

A RAPTR Architecture

Cross-view Encoder: Fig. 7a illustrates the layer structure of the cross-view encoder in our RAPTR architecture. The cross-view encoder associates multi-view radar features using the cross-attention and skip-connection mechanism. A shared-weight backbone extracts sets of multi-scale features for both horizontal and vertical radar heatmaps as $\mathbf{Z}_{\text{hor}} = \{\mathbf{Z}_{\text{hor},i}\}_{i=1}^S$ and $\mathbf{Z}_{\text{ver}} = \{\mathbf{Z}_{\text{ver},i}\}_{i=1}^S$ with S scale levels. Each scale-level feature map is enriched with spatial positional encoding and a learnable level embedding, following [50], and then fed into the cross-view encoder. The cross-view encoder consists of a stack of L_{enc} multi-head multi-scale deformable attention layers, and we denote the input horizontal/vertical features to the i -th layer as $\mathbf{F}_{\text{hor}}^{(i)}$, $\mathbf{F}_{\text{ver}}^{(i)}$, respectively. In our case, $\mathbf{F}_{\text{hor}}^{(0)} = \mathbf{Z}_{\text{hor}}$, $\mathbf{F}_{\text{ver}}^{(0)} = \mathbf{Z}_{\text{ver}}$. Each i -th layer runs a shared cross-attention bidirectionally: first with $\mathbf{F}_{\text{hor}}^{(i-1)}$ as key/value and $\mathbf{F}_{\text{ver}}^{(i-1)}$ as query, then vice versa. Residual connection, layer normalization, and an FFN follow as in standard Transformer encoders to further refine the features, and additional residual connections are incorporated to preserve view-specific details. As a whole, the encoding process in the i -th layer is written as:

$$\begin{aligned}\bar{\mathbf{F}}_a^{(i-1)} &= \mathbf{F}_a^{(i-1)} + \text{CrossAttn}(\mathbf{F}_a^{(i-1)}, \mathbf{F}_b^{(i-1)}), \\ \mathbf{F}_a^{(i)} &= \bar{\mathbf{F}}_a^{(i-1)} + \text{FFN}(\text{layernorm}(\bar{\mathbf{F}}_a^{(i-1)})), \quad (a, b) \in \{(\text{hor}, \text{ver}), (\text{ver}, \text{hor})\}.\end{aligned}\quad (10)$$

The output of the last layer, or encoder memory, is denoted as $\mathbf{F}_{\text{hor}}, \mathbf{F}_{\text{ver}}$.

Pose/Joint Decoder: Fig. 7b illustrates the layer structure of the pose/joint decoder in RAPTR. The pose decoder and the joint decoder share the architecture: they receive pose/joint queries $\mathbf{Q}_{\text{pose}}, \mathbf{Q}_{\text{joint}}$, multi-scale encoder memory from the cross-view encoder $\mathbf{F}_{\text{hor}}, \mathbf{F}_{\text{ver}}$, and reference points $\hat{\mathbf{P}}_{\text{radar}}$. The decoders then generate refined embeddings through multi-head self-attention and pseudo-3D deformable attention layers, which is the process denoted as $\mathcal{D}_{\text{pose}}$ and $\mathcal{D}_{\text{joint}}$ in Section 4. N pose queries correspond to N pose predictions in the pose decoder, whereas K joint queries correspond to K joints on the same subject in the joint decoder. The queries are first fed into self-attention, followed by residual connection and layer normalization, and then passed into the pseudo-3D deformable attention layer, as defined in Section 4.2. The pseudo-3D deformable attention layer is a cross-attention layer, using encoder memory to produce keys and values, which correlate with the refined queries. In addition, reference points are fed into this layer to determine the sampling locations and aggregate sparse features on the multi-view encoder memory across space and scales in the pseudo-3D deformable attention mechanism. The outputs are then passed through

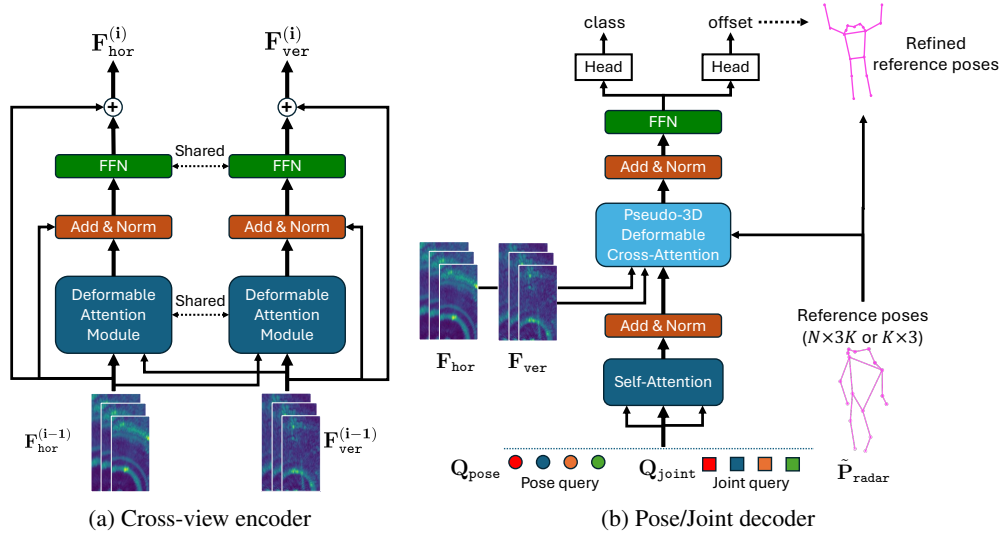


Figure 7: Transformer design in RAPTR.

another residual connection, layer normalization, and an FFN. In the pose decoder, a class regression head takes the resulting N queries to calculate confidence scores \hat{c} for each corresponding person. A pose regression head also takes the resulting queries to calculate pose coordinate offsets $\mathbb{R}^{N \times 3K}$ to refine the reference poses. In the joint decoder, a pose regression head, shared with the pose decoder, takes the resulting K queries to calculate the joint coordinate offsets $\mathbb{R}^{K \times 3}$ to refine the reference points. The pose decoder and the joint decoder consist of $L_{\text{pose}}, L_{\text{joint}}$ decoder layers, respectively, and their outputs are the initial pose estimates and the refined pose estimates, as shown in Fig. 3.

In the implementation, we have a technically involved step regarding the application of bipartite matching to the batched estimations from the pose decoder. We train the model using mini-batches, where the reference poses are first shaped as $B \times N \times 3K$, with B denoting the mini-batch size. While Section 4.3 describes the bipartite matching in general terms, it is in practice applied to the initial pose estimations $\tilde{\mathbf{P}}_{\text{world}}$ output by the pose decoder and the corresponding 3D keypoint labels $\mathbf{P}_{\text{world}}$ so that the computational cost of the subsequent joint decoder would be reduced. The matching is guided by the regressed confidence scores \hat{c} and the Euclidean distance between the initial estimations and the labels. Out of all estimations in the mini-batch, we retain only the N' matched ones and reshape them to $N' \times K \times 3$ to serve as the reference poses for the joint decoder, in which N' is considered the new mini-batch size.

Computational Complexity: The cross-view encoder takes two sets of multi-scale feature maps $\mathbf{F}_{\text{hor}}, \mathbf{F}_{\text{ver}}$ for horizontal-depth and vertical-depth radar perceptions. As shown in Fig. 7a, we apply deformable cross-attention in both directions: from \mathbf{F}_{hor} to \mathbf{F}_{ver} and vice versa, treating one set as queries and the other as keys and values in each direction. For each direction, given S -level feature scales, each with N_s spatial positions, and N_{offset} sampling points per head, the total computational cost is

$$\mathcal{O}\left(2(d^2 + N_{\text{offset}}d) \sum_{s=1}^S N_s\right) \approx \mathcal{O}\left((d^2 + N_{\text{offset}}d) \sum_{s=1}^S N_s\right), \quad (11)$$

where d is the feature dimension.

The pose decoder takes N object queries and the encoded memory and performs self-attention and pseudo-3D deformable cross-attention. Therefore, the computational cost is written as

$$\mathcal{O}(N^2d + Nd^2 + NN_{\text{offset}}Sd). \quad (12)$$

Here, we omit the constant factor associated with bilinear interpolation and regression of sampling offsets and attention weight matrices in the last term, as it does not affect the asymptotic complexity.

Finally, the joint decoder takes K joint queries for N' poses, selected through a bipartite matching procedure out of N , and the encoded memory and performs pseudo-3D deformable attention as well, and thus the cost is

$$\mathcal{O}\left((N'K)^2d + (N'K)d^2 + N'KN_{\text{offset}}Sd\right). \quad (13)$$

In conclusion, the total computational cost of our RAPTR is

$$\mathcal{O}\left((d^2 + N_{\text{offset}}d) \sum_{s=1}^S N_s + (N^2 + (N'K)^2)d + (N + N'K)d^2 + (N + N'K)N_{\text{offset}}Sd\right). \quad (14)$$

B Details of Pseudo-3D Deformable Attention

Multi-scale Multi-head Extension of Pseudo-3D Deformable Attention: We can extend the pseudo-3D deformable attention defined by Eq. 8 in Section 4.2 to multi-scale and multi-head operation. First, given M heads and S feature scale levels, Eq. 7 is extended as

$$\mathbf{f}_{ms,\text{hor}}^{(i)} = \mathbf{F}_{s,\text{hor}}(x + \Delta x_{msi}, z + \Delta z_{msi}), \quad \mathbf{f}_{ms,\text{ver}}^{(i)} = \mathbf{F}_{s,\text{ver}}(y + \Delta y_{msi}, z + \Delta z_{msi}), \quad (15)$$

where $\mathbf{F}_{s,\text{hor}}, \mathbf{F}_{s,\text{ver}}$ are the s -th level feature maps for horizontal and vertical view, respectively, and $\Delta\{x, y, z\}_{msi}$ is the i -th sampling offset in the m -th head on the s -th level feature. Subsequently,

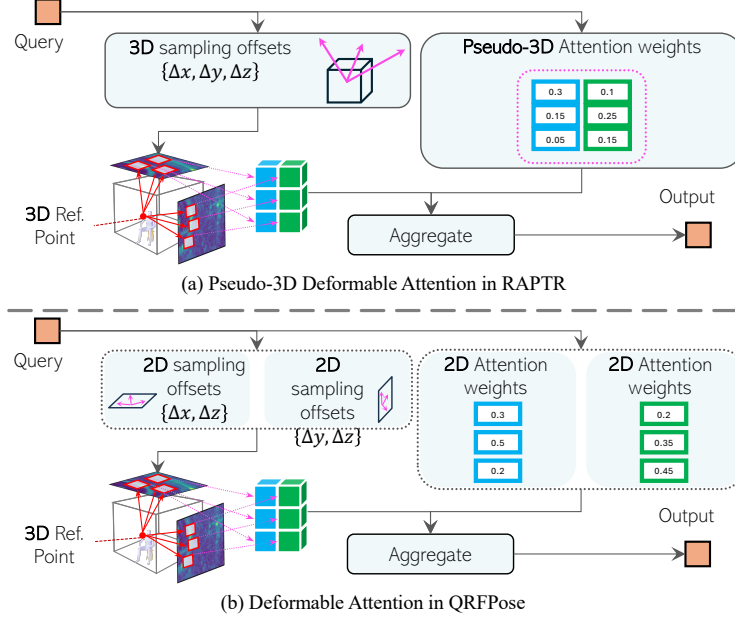


Figure 8: Comparison between (a) pseudo-3D and (b) decoupled 2D deformable attention mechanisms.

we collect the sampled features as $\mathbf{F}_{m,s,\text{attn}} = \{\mathbf{f}_{m,s,\text{hor}}^{(1)}, \mathbf{f}_{m,s,\text{ver}}^{(1)}, \dots, \mathbf{f}_{m,s,\text{hor}}^{(N_{\text{offset}})}, \mathbf{f}_{m,s,\text{ver}}^{(N_{\text{offset}})}\}$, and then extend Eq. 8 as

$$\bar{\mathbf{F}}_{\text{attn}} = \sum_{m=1}^M \mathbf{W}_m \left[\sum_{s=1}^S \sum_{i=1}^{N_{\text{offset}}} \left(A_{m,si,0} \mathbf{W}'_m \mathbf{F}_{m,s,\text{attn}}^{(2i-1)} + A_{m,si,1} \mathbf{W}'_m \mathbf{F}_{m,s,\text{attn}}^{(2i)} \right) \right], \quad (16)$$

where $\mathbf{W}_m \in \mathbb{R}^{d \times d_v}$ and $\mathbf{W}'_m \in \mathbb{R}^{d_v \times d}$ with $d_v = d/M$ are learnable weight matrices, and $A_{m,si,0}$ and $A_{m,si,1}$ are the attention weights of the i -th sampled deformable radar feature on the s -th level feature in the m -th attention head, for the horizontal and vertical radar views. Attention weights are normalized per head by $\sum_s \sum_i^{N_{\text{offset}}} A_{m,si} = 1$.

Implementation of Sampling Location Determination: We describe the pseudo-3D deformable attention in Section 4.2 as computing sampling locations by adding N_{offset} offsets, which are derived from each query, to a corresponding reference point. From an implementation point of view, our pose decoder adopts a structurally extended design. In our implementation, each query corresponds not to a single keypoint, but to the entire pose of a person, represented as $K \times 3$ coordinates. Accordingly, we replace the notion of a reference point with a reference *pose*, formatted as an $N \times 3K$ tensor for N queries. Sampling locations are determined by calculating the $3K$ offsets per query and adding them element-wise to the corresponding reference pose $K \times 3$. In this way, we virtually have K sampling points for each pose. This effectively enables a single query to take care of K spatial locations, allowing feature aggregation within the context of a unified pose.

On the other hand, the implementation in the joint decoder more aligns with the description in Section 4.2: a joint query in the joint decoder corresponds to a single joint so that the sampling locations are determined by calculating the N_{offset} sampling offsets per query and adding them to the corresponding reference joint. We set $N_{\text{offset}} = 4$ as listed in Table 7.

Computational Complexity Comparison with Decoupled 2D Deformable Attention: As illustrated in Fig. 8 (a), the pseudo-3D deformable attention adapts a 3D reference point with 3D sampling offsets and the pseudo-3D attention weights are computed over multiple radar views. In comparison, the QRFPos of Fig. 8 (b) adapts a 3D reference point with projected 2D sampling offsets and 2D attention weights are separately computed over each radar view [33]. This simple lifting operation may lead to better computational complexity of the pseudo-3D attention over the

Table 6: Complexity comparison of pseudo-3D vs. decoupled 2D deformable attention.

Queries (N)	Views (V)	2D Att	Pseudo-3D Att	Ratio (3D/2D)	Savings
10	2	$160NC$	$150NC$	0.94 ↓	6.25%
10	5	$400NC$	$330NC$	0.83 ↓	17.5%
10	10	$800NC$	$630NC$	0.79 ↓	21.3%

number of radar views. In the following, we provide a computational complexity analysis for the two types of deformable attention mechanisms, given V radar views:

- Decoupled 2D deformable attention: $\mathcal{O}(8VNN_{\text{offset}}d)$, where
 - 3D reference point projected to V 2D radar views: $\mathcal{O}(6VN)$,
 - Offset estimation: $\mathcal{O}(2VNN_{\text{offset}}d)$, where 2 is due to the computation of 2D (x, y) offsets,
 - Attention weights: $\mathcal{O}(VNN_{\text{offset}}d)$,
 - Feature aggregation: $\mathcal{O}(5VNN_{\text{offset}}d)$, where 5 is due to bilinear interpolation and weighted sum,
- Pseudo-3D deformable attention: $\mathcal{O}(6VNN_{\text{offset}}d + 3NN_{\text{offset}}d)$, where
 - Offset estimation: $\mathcal{O}(3NN_{\text{offset}}d)$, where 3 is due to the computation of the 3D (x, y, z) offsets,
 - 3D offset projected to V 2D radar views: $\mathcal{O}(6VN)$,
 - Attention weights: $\mathcal{O}(VNN_{\text{offset}}d)$,
 - Feature aggregation: $\mathcal{O}(5VNN_{\text{offset}}d)$.

Note that, in the above analysis, $\mathcal{O}(6VN)$ is excluded from the final complexity expressions as $6VN \ll 5VNN_{\text{offset}}d$ in practice. Table 6 shows the complexity comparison with a specific number of queries and an increasing number of views. It is observed that the pseudo-3D attention achieves computational savings of 17.5% with $V = 5$ radar views and 21.3% with $V = 10$ radar views, compared to the decoupled 2D attention.

C Optional View Mask Module

As an extension of our pseudo-3D deformable attention, we introduce an optional view mask module that aims to put more attention weights on the feature on the more important view with a hard-thresholding approach. The view mask module first computes a view selection mask \mathbf{M}_{attn} from a query \mathbf{q} corresponding to reference points of interest, as:

$$\mathbf{M}_{\text{attn}} = \sigma(\lambda \cdot \text{FFN}(\mathbf{q})) \in \mathbb{R}^{N_{\text{offset}} \times 2}, \quad (17)$$

where σ is the Sigmoid function, and $\lambda (\approx 1e5) \in \mathbb{R}$ makes the sigmoid output very close to 0 or 1 while preserving gradients. The element $m_{i,j} (\approx 1)$ signals that the j th view ($j = 0$ to the horizontal and $j = 1$ to the vertical) should retain its share of attention at the i th sampling point, whereas $m_{i,j} (\approx 0)$ marks it for suppression.

For each i th sampling point, we can consider three patterns to adjust the attention weights \mathbf{A}_{attn} for that point $A_{i,0}$, $A_{i,1}$ and, potentially, other sampling points according to the corresponding row in the view selection mask $[m_{i,0}, m_{i,1}]$.

- $[m_{i,0}, m_{i,1}] = [1, 1]$: use both views, weights unchanged;
- $[m_{i,0}, m_{i,1}] = [0, 1]$ or $[m_{i,0}, m_{i,1}] = [1, 0]$: ignore one view and transfer its weight to the other, e.g., when $[m_{i,0}, m_{i,1}] = [0, 1]$, the adjusted attention weights are $\hat{A}_{i,0} = A_{i,0} + A_{i,1}$, $\hat{A}_{i,1} = 0$, to ensure that the view selection decision at one sampling point does not influence the other sampling points;
- $[m_{i,0}, m_{i,1}] = [0, 0]$: ignore this sampling point in both views. Its weight is evenly redistributed so that $\sum_{i,j} \hat{A}_{i,j} = 1$ still holds.

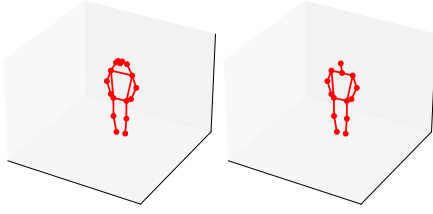


Figure 9: Template keypoints $\mathbf{K}_{\text{world}}$ for MMVR (left) and HIBER (right) dataset.

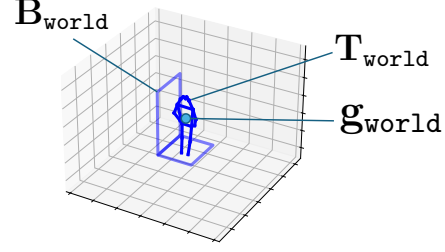


Figure 10: An example of a template pose located in the 3D world coordinate.

In this way, the view selection mask adaptively changes through training so that the attention mechanism associates with the view that is more informative for each sampling point. We can also utilize this view selection module to control the use of multi-view features by manually setting the values in the view selection mask, which we conduct a relating ablation study in the following Appendix I.

D Details of Loss Functions

In this section, we give a supplementary explanation of the loss functions in RAPTR provided in Section 4. For simplicity, we consider a single corresponding sample from each set, denoted as $\{\mathbf{b}_{\text{world}} \in \mathbf{B}_{\text{world}}, \mathbf{p}_{\text{image}} \in \mathbf{P}_{\text{image}}, \hat{\mathbf{p}}_{\text{image}} \in \hat{\mathbf{P}}_{\text{image}}, \tilde{\mathbf{p}}_{\text{world}} \in \tilde{\mathbf{P}}_{\text{world}}, \hat{\mathbf{p}}_{\text{world}} \in \hat{\mathbf{P}}_{\text{world}}\}$.

Coarse-grained 3D Loss: For 3D gravity loss, we compute the centroid of a 3D BBox label in $\mathbf{b}_{\text{world}} = [x_{\min}, y_{\min}, z_{\min}, x_{\max}, y_{\max}, z_{\max}]$ as the 3D gravity center label $\mathbf{g}_{\text{world}} \in \mathbb{R}^{1 \times 3}$ as

$$\mathbf{g}_{\text{world}} = \left[\frac{x_{\max} - x_{\min}}{2}, \frac{y_{\max} - y_{\min}}{2}, \frac{z_{\max} - z_{\min}}{2} \right]. \quad (18)$$

Given a refined 3D pose estimate $\hat{\mathbf{p}}_{\text{world}} = \{(\hat{\mathbf{p}}_{\text{world},x}^{(k)}, \hat{\mathbf{p}}_{\text{world},y}^{(k)}, \hat{\mathbf{p}}_{\text{world},z}^{(k)})\}_{k=1}^K$ at the joint decoder, we also collapse it into its centroids as $\hat{\mathbf{g}}_{\text{world}} \in \mathbb{R}^{1 \times 3}$ as

$$\hat{\mathbf{g}}_{\text{world}} = \left[\frac{1}{K} \sum_k \hat{\mathbf{p}}_{\text{world},x}^{(k)}, \frac{1}{K} \sum_k \hat{\mathbf{p}}_{\text{world},y}^{(k)}, \frac{1}{K} \sum_k \hat{\mathbf{p}}_{\text{world},z}^{(k)} \right], \quad (19)$$

where $\hat{\mathbf{p}}_{\text{world},x/y/z}^{(k)}$ is the x -, y -, and z -coordinate for the k -th joint. The 3D gravity loss $\mathcal{L}_{\text{gravity}}$ is defined as the Euclidean distance between the two 3D gravity centers, $\mathbf{g}_{\text{world}}$ and $\hat{\mathbf{g}}_{\text{world}}$.

For 3D template loss, we construct a 3D template keypoint label for each $\mathbf{b}_{\text{world}}$ using template keypoints at the coordinate origin $\mathbf{K}_{\text{world}} \in \mathbb{R}^{K \times 3}$. The template pose $\mathbf{T}_{\text{world}} \in \mathbb{R}^{K \times 3}$ is given as $\mathbf{T}_{\text{world}} = \mathbf{K}_{\text{world}} + \mathbf{1}^\top \mathbf{g}_{\text{world}}$. Fig. 9 shows the template keypoints $\mathbf{K}_{\text{world}}$ with different numbers of keypoints for the MMVR and HIBER datasets, and Fig. 10 provides an example of locating these template keypoints in the 3D world coordinate based on the location of a 3D BBox label. The 3D template loss $\mathcal{L}_{\text{template}}$ is defined as the Euclidean distance between the template poses $\mathbf{T}_{\text{world}}$ and the initial 3D pose estimates $\tilde{\mathbf{P}}_{\text{world}}$ from the pose decoder.

Fine-grained 2D Loss: Specifically in the fine-grained 2D loss, OKS loss \mathcal{L}_{OKS} is the loss function based on object keypoint similarity (OKS), a metric used to evaluate the accuracy of keypoint estimations taking into account the object scale and keypoint visibility, which is defined as

$$\text{OKS}(\mathbf{p}_{\text{image}}, \hat{\mathbf{p}}_{\text{image}}) = \sum_k \exp\left(-\frac{d_k^2}{2s^2\psi_k^2}\right), \quad (20)$$

where d_k is the distance between the k -th estimated joint $\hat{\mathbf{p}}_{\text{image}}^{(k)}$ and the corresponding label $\mathbf{p}_{\text{image}}^{(k)}$, s is the object scale, and ψ_k is a pre-defined constant for the k -th joint. Here, we assume that all keypoint labels are annotated as visible points. Since OKS is a metric in which higher values indicate greater similarity, its negative logarithm $-\log(\text{OKS})$ is taken when used as a loss function \mathcal{L}_{OKS} .

E Datasets

HIBER: HIBER [35] is an open-source multi-view radar dataset for indoor human perception tasks including detection, segmentation, and keypoint estimation. They provide horizontal and vertical radar heatmaps and corresponding labels such as 2D BBoxes, 2D segmentation masks, 2D keypoints, 3D BBoxes, and 3D keypoints. Among its data environments, WALK and MULTI are currently available. WALK comprises frames that feature a single individual, while frames in MULTI consistently depict two individuals walking concurrently. The frames provided are captured from ten distinct viewpoints within a single room, designated as “view01” through “view10.” We use “view02” to “view10” for training, validating, and testing the models, with the data splits provided, and the specific number of frames is listed in Table 7.

MMVR: MMVR [24] is a more recent open-source multi-view radar dataset for indoor human perception. They provide horizontal and vertical radar heatmaps and corresponding labels, such as 2D BBoxes, 2D segmentation masks, 2D keypoints, and 3D BBoxes. They collected data from 25 subjects in 6 different scenarios (e.g., open/cluttered office spaces) spanning over 9 days. MMVR consists of 1) P1: single-person scenarios in an open space without any obstacles, and 2) P2: multi-person scenarios in a cluttered office spaces, including sitting postures. P1 is designed to establish fundamental benchmarks for radar-based human perception tasks, while P2 is designed to challenge with more realistic and complicated indoor scenarios and cross-environment, cross-subject generalization. The data split we use is S1 that they provide, and the specific number of frames is listed in Table 7.

F Hyper Parameters

In the evaluations presented in Section 5, the model training for all baselines and our RAPTR and its variants share the hyper parameters outlined in Table 7, unless otherwise specified.

G Baseline Implementations

We provide the specific implementation for the baseline methods that we use in the evaluation.

Person-in-WiFi 3D: We refer to the official implementation [40] and modify some parts of the code to make them compatible with the datasets that we use. We employ a ResNet backbone to extract multi-scale features from the radar heatmaps, as well as our RAPTR does. We then take the C4 feature map, flatten it as the N_{token} tokens with dimensions of d , and feed it into the network. Specifically, $N_{\text{token}} = 260$ for the HIBER dataset and 256 for the MMVR dataset. Since the original study uses Wi-Fi channel state information (CSI), which inherently lacks explicit spatial structure, and transforms it into 180 tokens for input, our approach of converting C4 feature maps derived from radar heatmaps into approximately 200 tokens can be reasonably justified in terms of fairness and comparability. Regarding the loss function, we implement the loss as the summation of class loss, 2D keypoint loss, refined 2D keypoint loss, and 3D gravity loss, with loss weights of 1.0, 5.0, 10.0, and 1.0, respectively.

HRRadarPose: We refer to the official implementation [11] and modify some parts of the code to make them compatible with the datasets that we use. First, we exclusively utilize horizontal view radar heatmaps, excluding vertical view heatmaps. This decision stems from the disparity in angular resolution between the elevation and azimuth axes reported in the HRRadarPose paper. The resolution of the elevation axis is 18 degrees, while the resolution of the azimuth axis is 1.4 degrees, and only the azimuth resolution is comparable to that of HIBER and MMVR (1.3 degrees). We presume that we could solely use the horizontal view heatmaps while ensuring fairness in our evaluations. In addition, we expand the original codes to multi-person scenarios by implementing Non-Maximum Suppression (NMS) on the predictions. Regarding the loss function, we implement the loss as the summation of heatmap loss, 2D keypoint loss, and 3D gravity loss, with loss weights of 5.0, 1.0, and 1.0, respectively.

QRFPose: Currently, the authors of the paper have not released official codes. Therefore, we independently replicated the implementation based on the architectures and parameters outlined in

Table 7: Hyper parameters for RAPTR

Name	Notation	Value	
		HIBER	MMVR
Data			
Radar image resolution	W, H, D	160, 160, 200	128, 128, 256
# of training samples	-	59000 / 54280 (WALK / MULTI)	86579 / 190441(P1S1 / P2S1)
# of validation samples	-	6490 / 5900 (WALK / MULTI)	10538 / 23899 (P1S1 / P2S1)
# of test samples	-	3540 / 3540 (WALK / MULTI)	10785 / 23458 (P1S1 / P2S1)
# of keypoints	K	14	17
Model params			
Backbone	-	ResNet 18	
# of feature scale	S	3	
Feedforward dimension in Transformer	-	1024	
# of encoder layers	L_{enc}	3	
# of pose decoder layers	L_{pose}	2	
# of joint decoder layers	L_{joint}	3	
# of deformable sampling offsets	N_{offset}	$K / 4$ (pose / joint decoder)	
# of heads in multi-head attention	-	8	
Feature dimension	d	128	
# of input frames	T	4	
# of pose query	N	10	
Training params			
Optimizer	-	AdamW	
Base learning rate	-	$2e-4$	
Weight decay	-	$1e-4$	
LR scheduler	-	Cosine Decay	
Batch size	-	32	
Epochs	-	50	
Gradient clip norm	-	0.1	
Early stopping patience	-	5 epochs	
Loss weights			
3D template loss	λ_1	1.0	
3D gravity loss	λ_2	1.0	
2D Keypoint loss	λ_3	5.0	
2D OKS loss	λ_4	1.0	
Class loss	λ_5	1.0	
Computational Resource			
GPU		NVIDIA A40	
# of workers		8	
Approximate training time		3 hours / 10K samples	

the paper. We verify that our implementation replicates performance similar to that of the original report using 3D keypoint labels. To ensure a fair comparison, we set the number of Transformer decoder layers to 5, which is equivalent to the total number of layers in the pose decoder and the joint decoder in our RAPTR model. Although the original implementation uses RLE loss [16] as the keypoint regression loss function, we employ the conventional Euclidean distance loss in our implementation so that we can integrate the loss with the 3D gravity loss in a more balanced way. Regarding the loss function, we implement the loss as the summation of class loss, 2D keypoint loss, and 3D gravity loss, with loss weights of 1.0, 5.0, and 1.0, respectively.

H Metrics

For simplicity, we omit the subscripts that indicate the coordinate system in which the keypoints or BBoxes are defined (radar, world) in this section. In addition, $\mathbf{p} \in \mathbf{P}$, $\mathbf{b} \in \mathbf{B}$, and $\hat{\mathbf{p}} \in \hat{\mathbf{P}}$ denote the corresponding samples taken from the 3D keypoint labels, the 3D BBox labels, and the 3D pose estimates, respectively.

MPJPE: We employ Mean Per Joint Position Error (MPJPE) as the performance metric to evaluate the 3D pose estimation capabilities of the models. Given a 3D keypoint label $\mathbf{p} = \{\mathbf{p}^{(k)} | (x^{(k)}, y^{(k)}, z^{(k)})\}_{k=1}^K$ and the corresponding estimate $\hat{\mathbf{p}} = \{\hat{\mathbf{p}}^{(k)} | (\hat{x}^{(k)}, \hat{y}^{(k)}, \hat{z}^{(k)})\}_{k=1}^K$. MPJPE is defined as:

$$\text{MPJPE} = \frac{1}{K} \sum_{k=1}^K \|\mathbf{p}^{(k)} - \hat{\mathbf{p}}^{(k)}\|_2. \quad (21)$$

The unit for MPJPE that we use is the centimeter in the world coordinate system. We also evaluate MPJPE along each axis: horizontal (h), vertical (v), and depth (d).

3D BBox-based Metrics for MMVR: For the evaluation of the MMVR dataset, due to the absence of 3D keypoint labels in the dataset, we approximate the pose estimation performance in a different way from that for the HIBER dataset. Specifically, we calculate 1) the distance between the center of the 3D pose estimate $\hat{\mathbf{P}}$ and the 3D BBox label \mathbf{B} , and 2) the absolute error in the edge lengths along each axis of the box. Specifically, we first construct a 3D BBox that encloses the estimated 3D keypoints as $\hat{\mathbf{b}} = [\min(\mathbf{p}_x), \min(\mathbf{p}_y), \min(\mathbf{p}_z), \max(\mathbf{p}_x), \max(\mathbf{p}_y), \max(\mathbf{p}_z)]$ where $\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z$ is the set of x -, y - and z -coordinates of the estimated keypoints. We then calculate the center coordinate of the 3D BBox label $\mathbf{b} = [x_{\min}, y_{\min}, z_{\min}, x_{\max}, y_{\max}, z_{\max}]$ and $\hat{\mathbf{b}}$ as

$$\mathbf{g} = \left[\frac{x_{\max} - x_{\min}}{2}, \frac{y_{\max} - y_{\min}}{2}, \frac{z_{\max} - z_{\min}}{2} \right],$$

$$\hat{\mathbf{g}} = \left[\frac{\max(\mathbf{p}_x) - \min(\mathbf{p}_x)}{2}, \frac{\max(\mathbf{p}_y) - \min(\mathbf{p}_y)}{2}, \frac{\max(\mathbf{p}_z) - \min(\mathbf{p}_z)}{2} \right]. \quad (22)$$

The center distance between the BBoxes is the Euclidean distance between \mathbf{g} and $\hat{\mathbf{g}}$. We also calculate the edge lengths of \mathbf{b} and $\hat{\mathbf{b}}$ as

$$\mathbf{l} = (x_{\max} - x_{\min}, y_{\max} - y_{\min}, z_{\max} - z_{\min}),$$

$$\hat{\mathbf{l}} = (\max(\mathbf{p}_x) - \min(\mathbf{p}_x), \max(\mathbf{p}_y) - \min(\mathbf{p}_y), \max(\mathbf{p}_z) - \min(\mathbf{p}_z)), \quad (23)$$

and we calculate the absolute error of the edge length along each axis.

I Additional Ablation Studies and Visualization

To validate the effectiveness of RAPTR, we conduct additional ablation studies. Unless otherwise specified, we conduct the studies with the hyper parameters in Table 7.

I.1 Numerical Results

Additional Results for MMVR on P2S1: Table 8 shows the evaluation results for the MMVR dataset on P2S1. P2S1 includes cluttered indoor scenarios with multiple subjects, which is thus more challenging than P1S1. RAPTR outperforms baselines and shows improvements in center distance by 71.54%, 85.28%, and 69.47% compared to Person-in-WiFi 3D, QRFPose, and HRRadarPose, respectively. We defer the qualitative evaluation for MMVR P2S1 to Appendix I.2.

Table 8: 3D pose estimation performance on MMVR (P2S1).

Method	Center distance (cm)	Edge length error (cm)		
		(h)	(v)	(d)
Person-in-WiFi 3D	103.43	48.29	112.75	152.88
QRFPose	200.03	115.80	126.14	335.30
HRRadarPose	96.43	32.19	51.02	175.04
RAPTR (ours)	29.44	18.74	27.14	40.29

Full 3D Supervision: We compare the performance of our RAPTR under (i) full supervision with fine-grained 3D keypoint labels and (ii) weak supervision with 2D keypoint and 3D BBox labels for HIBER (MULTI). Table 9 shows the performance comparison in MPJPE. Under full 3D supervision, the RAPTR architecture achieves an MPJPE of 8.93 cm. Even when trained under weak supervision, MPJPE increases by only about 10 cm, indicating that our structured loss design with two-stage decoding approach effectively learns reliable 3D body structures.

Effect of 3D Templates and Their Scales: We evaluate the impact of 3D templates and their scale on the final MPJPE performance. Specifically, we experiment with

- A **standing pose** scaled by two factors: $0.5\times$ and $1\times$,

Table 9: RAPTR performance with full and weak supervision (HIBER MULTI).

Method	Head	Neck	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	MPJPE	(h)	(v)	(d)
RAPTR (Full 3D supervision)	7.90	5.28	6.41	7.94	12.43	5.75	8.80	14.65	8.93	4.97	2.86	5.12
RAPTR (Weak supervision)	18.39	13.13	16.44	20.12	24.62	15.01	17.76	23.22	18.99	7.80	4.38	14.54

Table 10: Effect of 3D Templates and their scales on the performance (HIBER MULTI).

3D Template	MPJPE
Standing (scale=1)	18.99 \pm 0.16
Standing (scale=0.5)	20.11 \pm 0.54
Sitting (scale=1)	20.84 \pm 0.91
Standing (learned scale)	23.13 \pm 0.33

- A **sitting pose** of a 1.6 m-tall person,
- A **learned scaling factor** applied to the standard standing pose.

Table 10 suggests that the choice of 3D template has minor impacts on the final MPJPE, likely due to the refinement capability of the second-stage joint decoder, as long as the first-stage decoder generates a reasonable, human-like initial pose.

Effect of Loss Weighting Factors: We assess the RAPTR performance under varying loss weighting factors λ_i . Three configurations are evaluated: 1) equal weights for all loss terms, 2) increased weights on 3D losses terms, and 3) increased weights on 2D keypoint loss. Table 11 provides the performance comparison among these settings.

When all weighting factors are set to 1.0, RAPTR achieves an average MPJPE of 19.91 cm. Increasing the weights of the 3D losses to 5.0 degrades performance, resulting in an MPJPE of 24.04 cm. In contrast, emphasizing the 2D keypoint loss yields the best performance with an MPJPE of 18.99 cm. These results suggest that appropriately balancing the loss terms, particularly by increasing the weight of the 2D keypoint loss, plays a crucial role in enhancing joint localization accuracy.

Table 11: Effect of loss weighting factors on the RAPTR performance (HIBER MULTI).

λ_1 3D template	λ_2 3D gravity	λ_3 2D keypoint	λ_4 2D OKS	λ_5 class	MPJPE
1.0	1.0	1.0	1.0	1.0	19.91 \pm 0.65
5.0	5.0	1.0	1.0	1.0	24.04 \pm 0.75
1.0	1.0	5.0	1.0	1.0	18.99 \pm 0.16

Impact of View Selection: To investigate how the use of features from horizontal and vertical views affects performance, we evaluate the performance of 3D pose estimation by configuring the view selection mask $\mathbf{M}_{\text{attn}} \in \mathbb{R}^{N_{\text{offset}} \times 2}$, integrated into pseudo-3D deformable attention as described in Appendix C, according to several predefined patterns. Specifically, given that the first column of the view selection mask $\mathbf{M}_{\text{attn}}[:, 0]$ and the second column $\mathbf{M}_{\text{attn}}[:, 1]$ correspond to horizontal and vertical views, respectively, we set the values in the mask as

- **Both Views:** all values in \mathbf{M}_{attn} to 1 so that the attention weight matrix \mathbf{A}_{attn} is used as is;
- **Horizontal View Only:** $\mathbf{M}_{\text{attn}}[:, 0] = \mathbf{1}$ and $\mathbf{M}_{\text{attn}}[:, 1] = \mathbf{0}$ so that the features sampled from the horizontal feature map are aggregated and those of the vertical feature are omitted;
- **Vertical View Only:** The reversed one of “horizontal only”: $\mathbf{M}_{\text{attn}}[:, 0] = \mathbf{0}$ and $\mathbf{M}_{\text{attn}}[:, 1] = \mathbf{1}$;
- **Random Mask:** The values in the mask are randomly assigned for each mini-batch step in the training process;
- **Adaptive View Selection:** The mask values are adaptively determined by the corresponding queries, described in Appendix C.

Table 12: Effect of view selection patterns on the performance.

	MPJPE	(h)	(v)	(d)
Both Views	20.31 \pm 0.34	8.44 \pm 0.34	5.02 \pm 0.05	15.21 \pm 0.67
Horizontal View Only	20.55 \pm 2.05	8.74 \pm 0.67	5.42 \pm 0.82	14.97 \pm 1.99
Vertical View Only	23.78 \pm 0.77	12.24 \pm 1.45	4.66 \pm 0.14	16.61 \pm 1.15
Random Mask	21.18 \pm 0.33	8.75 \pm 0.78	4.96 \pm 0.72	16.04 \pm 0.10
Adaptive View Selection (ours)	18.99 \pm 0.16	7.80 \pm 0.31	4.38 \pm 0.25	14.54 \pm 0.13

Table 12 shows the performance comparison in MPJPE. When using only a single view as input, restricting the input to either the horizontal or vertical view results in a noticeable increase in MPJPE, averaging 20.55 cm and 23.78 cm, respectively. In contrast, the adaptive view selection scheme achieves the lowest average MPJPE of 18.99 cm, outperforming both the “random mask” and “both views” settings.

When using only a single view as input, restricting the input to either the horizontal or vertical view leads to a noticeable increase in MPJPE, averaging 20.55 cm and 23.78 cm, respectively. In contrast, under the multi-view input setting, the adaptive view selection strategy achieves the lowest average MPJPE of 18.99 cm, outperforming both the “random mask” and “both views” configurations.

Analysis of Attention Weight Assignment: We investigate the contribution of each radar view by analyzing the distribution of attention weights and view-selection patterns in the pose and joint decoders. Table 13 shows the joint-wise breakdown of attention weight assignment. Note that weights in the pose decoder are normalized over all joints and views, while those in the joint decoder are normalized per joint. Due to averaging over the test set, values may not sum to exactly 1. In the pose decoder, both horizontal and vertical views receive relatively balanced attention across joints. On the other hand, in the joint decoder, vertical views consistently receive higher weights. Specifically, the edge joints exhibit larger disparities, such as 25.07% difference for the knee and 18.39% for the wrist. Since the pose decoder merely aligns the estimates to template poses, it only requires a rough estimate of the overall pose structure, and thus the attention weights are almost evenly distributed for both views. In contrast, the joint decoder is tasked with refining each joint, which makes the vertical view more critical, as it provides more comprehensive visibility across all joints.

Table 13: Attention weight assignment in the pose/joint decoders ($\times 1e-2$). The larger value indicates the more attention is weighted on the view for each joint. While the pose decoder assigns balanced attention across joints, the joint decoder makes the vertical view more critical with larger weights.

		Head	Neck	Shoulder	Elbow	Wrist	Hip	Knee	Ankle
Pose	Horizontal	4.59	3.07	3.53	3.07	2.82	3.56	2.30	1.57
	Vertical	4.01	3.35	2.33	3.74	3.68	4.68	2.36	3.63
Joint	Horizontal	34.08	40.32	41.96	40.18	31.45	38.40	32.06	35.72
	Vertical	50.20	46.83	46.31	46.84	49.84	49.94	57.13	47.69

In RAPTR, we adopt a view selection approach in which the view selection mask is adaptively determined by the corresponding queries, as described in Appendix C, and Table 14 shows the joint-wise breakdown of view selection assignments in the pose and joint decoders. “Omit Both”, “Use Horizontal”, “Use Vertical”, and “Use Both” are represented by the view selection mask values $[0, 0]$, $[1, 0]$, $[0, 1]$, and $[1, 1]$ for each corresponding row in the mask, respectively. While the pose decoder shows no prominent value imbalance across the assigned patterns, the joint decoder tends to rely more on both views, with high “Use Both” ratios observed for almost all joints, such as 31.06% for shoulder, 32.48% for hip, and 32.44% for neck. This trend reflects that the joint decoder prefers to integrate information from both views when refining keypoint locations.

I.2 Additional Visualizations

Visualization of Pseudo-3D Deformable Attention: Fig. 11 shows the visualization of pseudo-3D deformable attention at the last layers in the pose and joint decoders. For each view (horizontal on the left, vertical on the right), close-up regions around the bright radar reflections corresponding to subjects are extracted and visualized. The red dots on the plots indicate the sampling locations

Table 14: The joint-wise breakdown of view-selection assignments in the pose/joint decoders (%). The joint decoder relies on both views more than the pose decoder, especially for edge joints.

		Head	Neck	Shoulder	Elbow	Wrist	Hip	Knee	Ankle
Pose	Omit Both	25.81	21.09	23.59	22.50	26.16	18.99	25.31	28.42
	Use Horizontal	22.61	25.47	27.15	31.84	22.74	23.29	19.35	26.55
	Use Vertical	29.65	26.70	22.72	20.79	24.43	21.76	22.42	23.33
	Use Both	21.92	26.74	26.54	24.86	26.67	35.96	32.92	21.70
Joint	Omit Both	20.73	17.87	19.94	19.24	24.20	21.54	19.13	23.48
	Use Horizontal	22.75	21.55	22.94	23.18	21.24	25.86	23.17	25.25
	Use Vertical	27.99	28.14	26.06	27.69	28.20	20.12	26.52	24.98
	Use Both	28.53	32.44	31.06	29.88	26.37	32.48	31.18	26.29

selected by deformable attention. The pseudo-3D deformable attention mechanism primarily samples features from regions surrounding human subjects. Specifically, in the vertical view, the sampling points are clearly divided and distributed by joint. Moreover, as the sampling offsets are computed across the x , y , and z axes at once in our pseudo-3D deformable attention, the sampling locations maintain consistent alignment in the depth direction across the views for the same subject.

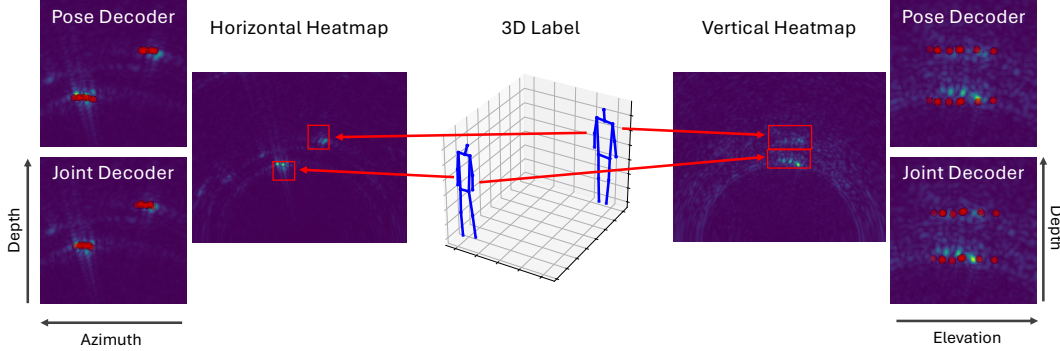
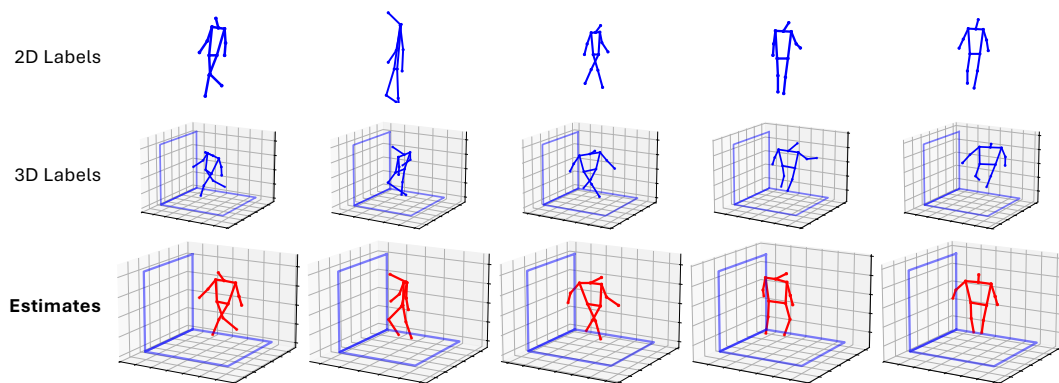


Figure 11: Visualization of pseudo-3D deformable attention. The attention mechanism samples features around the bright signals caused by body reflection, represented as red dots on the plots.

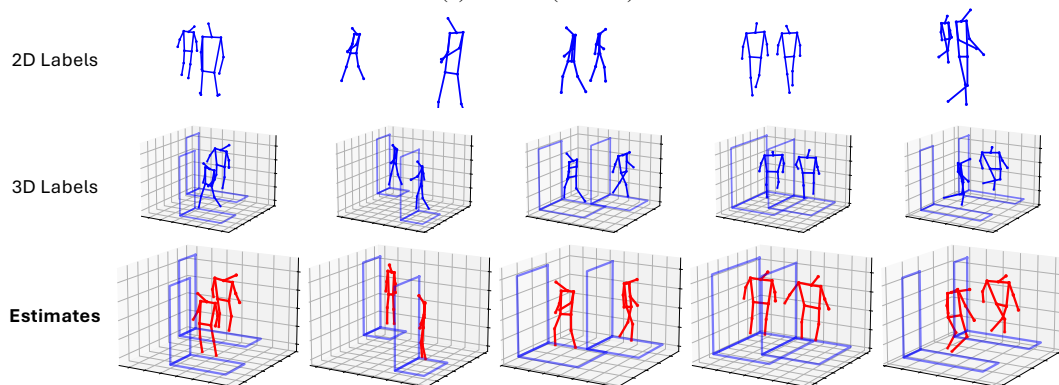
Additional Visualization Cases: We provide visualizations of the RAPTR estimation results for more cases in Fig. 12. We present visualizations of 2D and 3D labels and 3D pose estimates for HIBER WALK, MULTI, MMVR P1S1, and P2S1, arranged from top to bottom.

For the HIBER dataset, RAPTR maintains a stable estimation quality by capturing overall body orientation and limb articulation in (a), where only a single subject is presented, and (b) with multiple subjects. Although minor inaccuracies are observed in some cases, RAPTR preserves the spatial arrangement and relative depth of the subjects. As can be observed in the figure, the annotated 3D BBoxes are often significantly larger than the actual human body size, which originates from the dataset itself. Since RAPTR utilizes only the center of the 3D BBoxes as reference for coarse-grained 3D cue, it remains largely unaffected by such inaccuracies in box scale.

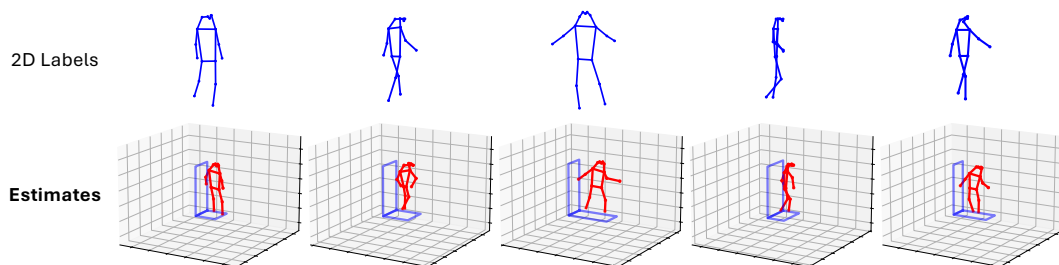
For the MMVR dataset, RAPTR has to deal with a diverse range of body configurations, including seated and crouched poses, and increased subject variability. In the P1S1 setting (c), RAPTR consistently estimates plausible 3D pose estimations that are well aligned with 2D keypoint labels, even for non-standard upright posture like spreading arms. On the other hand, in the P2S1 setting (d), RAPTR demonstrates reasonable performance for more complex body configurations with multiple subjects: it often captures the overall structure and spatial arrangement of each individual. In some cases, RAPTR even reconstructs plausible limb configurations where the 2D labels are inaccurate or incomplete, such as the seated person’s legs in the leftmost example of (d). This suggests that the architectural design, which first estimates initial poses using the pose decoder with a template pose, then refines joint positions via the joint decoder, effectively preserves a human-body prior throughout the process. However, there are visible imperfections in some cases, such as over-extended limbs or inaccurate limb orientations, illustrating the difficulty of 3D pose estimation in scenarios with occlusion and extreme articulation.



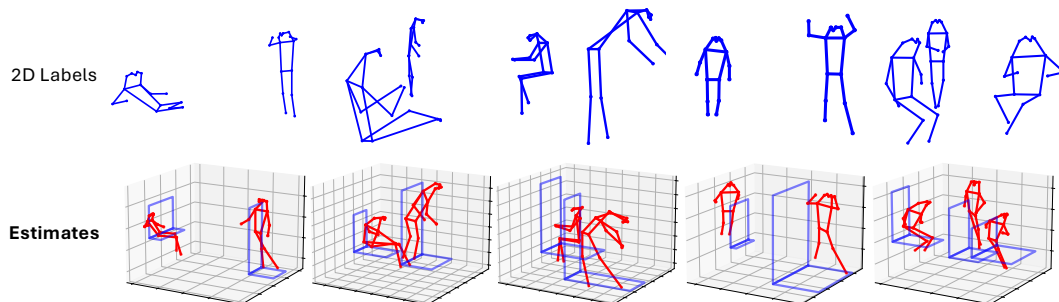
(a) HIBER (WALK)



(b) HIBER (MULTI)



(c) MMVR (P1S1)



(d) MMVR (P2S1)

Figure 12: Visualization for RAPTR 3D pose estimation.

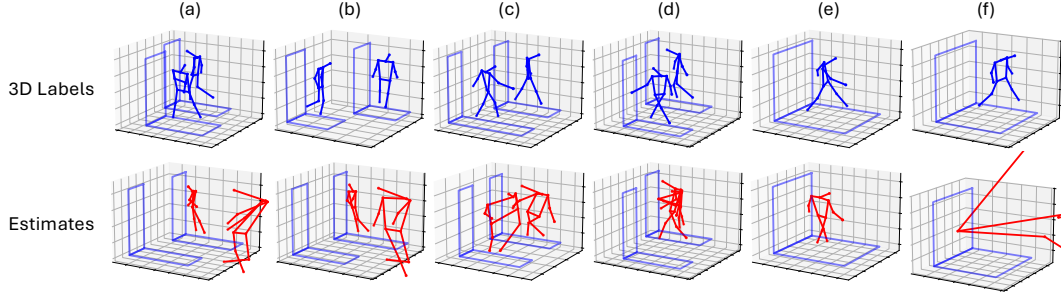


Figure 13: Failure cases for the HIBER (MULTI) dataset.

Failure Cases: We provide failure cases for the HIBER MULTI dataset in Fig. 13. In (a) and (b), RAPTR locates pose estimations significantly away from the 3D BBox labels, indicating errors in both subject localization and depth reasoning. In (c), the estimated poses appear severely distorted, failing to preserve the anatomical structure of the body. Despite being spatially close to the correct locations, the joint configurations are implausible, indicating a breakdown in fine-grained pose refinement. In (d), RAPTR fails to correctly associate the estimated poses with the 3D BBoxes, leading to redundant estimates that multiple predictions are assigned to the same individual, degrading the quality of the final estimation. In (e), RAPTR fails to recover the correct body orientation. While the label pose is stepping forward with the left leg in the back-right direction, the estimated pose incorrectly keeps the body facing forward and places the right leg in an unnatural cross-step position. In (f), RAPTR produces a completely corrupted estimate with no apparent correspondence to the human pose. These failures are frequently observed when the two individuals overlap on the 2D image plane and the 2D keypoint labels themselves exhibit pose inaccuracies, indicating the limitation of our RAPTR, which relies on the 2D keypoint labels to accurately represent human shapes.