PI-HMR: Towards Robust In-bed Temporal Human Shape Reconstruction with Contact Pressure Sensing

Supplementary Material

A. Introduction

In this material, we provide additional details regarding the network and implementation of our methods, as well as compared SOTAs. We further present more qualitative results to show the performance of PI-HMR and our re-generated p-GTs for TIP [17] and to explore their failure scenarios. The details include:

- Implementation details for SMPLify-IB, PI-HMR, cross-modal knowledge distillation, VQ-VAE, test-time optimization, and SOTA methods compared to PI-HMR.
- More quantitative and qualitative results about SMPLify-IB, PI-HMR, and failure cases.
- Limitations and future works.

The overall pipeline of our pressure-to-motion flow is shown in Fig. 1, and detailed architecture and implementation details will be elaborated below.

B. Preliminary

Body Model. The SMPL [8] model provides a differentiable function $V = \mathcal{M}(\theta, \beta, t)$ that outputs a posed 3D mesh with $N = 6890$ vertices. The pose parameter $\theta \in \mathbb{R}^{24\times3}$ includes a $\mathbb{R}^3$ global body rotation and the relative rotation of 23 joints with respect to their parents. The shape parameter $\beta \in \mathbb{R}^{10}$ represents the physique of the body shape. And $t \in \mathbb{R}^3$ means the root translation w.r.t the world coordinate.

C. Network and Implementation Details

C.1. Implementation details for SMPLify-IB

C.1.1. The first stage

In the first stage of our optimization algorithm, we jointly optimize body shape β, pose parameters θ, and translation t using a sliding-window (set as 128) approach, with overlap (set as 64) between adjacent windows. We minimize the following objective function:

$$L_{s1}(\theta,\beta,t) = \lambda_J\mathcal{L}_J + \lambda_p\mathcal{L}_p + \lambda_{sm}\mathcal{L}_{sm} + \lambda_{cons}\mathcal{L}_{cons} + \lambda_{bc}\mathcal{L}_{bc} + \lambda_g\mathcal{L}_g + \lambda_{sc}\mathcal{L} \quad (1)$$

1. **Reprojection constraint term $\mathcal{L}_J$:** This term penalizes the weighted robust distance between the projections of the estimated 3D joints and the annotated 2D joint ground truths. Instead of the widely used weak-perspective projection in [1] with presumed focal length, we apply the perspective projection with calibrated focal length and camera-bed distance provided by TIP.

2. **Prior constraint term $\mathcal{L}_p$:** This term impedes the unrealistic poses while allowing possible ones. $\mathcal{L}_{pose}$, $\mathcal{L}_{shape}$ penalizes the out-of-distribution estimated postures and shapes, which is similar to terms in SMPLify, and $\mathcal{L}_{torso}$ ensures correct in-bed torso poses, where the height of hips should be less than shoulders and the height of waist is below the mean height of shoulders and hips.

$$\mathcal{L}_p = \mathcal{L}_{pos} + \mathcal{L}_{sha} + \mathcal{L}_{tor} \quad (2)$$

$$\mathcal{L}_{pos} = \sum_i^T(\lambda_1^{pos}(\mathcal{G}(\theta(i)) + \sum_j \lambda_{2,j}^{pos} \cdot e^{\gamma_j \cdot \theta(i)_j}))$$

$$\mathcal{L}_{sha} = \lambda^{sha}\sum_i^T ||\beta(i)||^2$$

$$\mathcal{L}_{tor} = \sum_i^T(\lambda_1^{tor} \cdot e^{\omega_{hip} d_{hip}(i)} + \lambda_2^{tor} \cdot e^{\omega_{wai} d_{wai}(i)})$$

$$d_{hip}(i) = z_{hip}(i) - z_{sho}(i)$$

$$d_{wai}(i) = z_{wai}(i) - mean(z_{hip}(i), z_{sho}(i))$$

where $\mathcal{G}$ is the Gaussian Mixture Model pre-trained in SMPLify, and the second term in $\mathcal{L}_{pos}$ penalizes impossible bending of limbs, neck and torso, such as shoulder twist. z_{hip}, z_{sho}, z_{wai} are the height of hip joints, shoulder joints, and waist joint, and ω_{hip}, ω_{wai} are both set to 100.

3. **Smooth constraint term $\mathcal{L}_{sm}$:** This term reduces the jitters by minimizing the 3D joints velocity, acceleration and SMPL parameter differences.

$$\mathcal{L}_{smo} = \mathcal{L}_{par} + \mathcal{L}_{vel} + \mathcal{L}_{acc} \quad (3)$$

$$\mathcal{L}_{par} = \sum_{i=1}^{T-1}(\lambda_1^{par}||\beta(i+1) - \beta(i)||^2 + \lambda_2^{par}||\theta(i+1) - \theta(i)||^2 + \lambda_2^{par}||t(i+1) - t(i)||^2)$$

$$\mathcal{L}_{vel} = \sum_{i=1}^{T-1}(\lambda_1^{vel}||J(i+1)_{3D} - J(i)_{3D}||^2 + \lambda_2^{vel}||V(i+1) - V(i)||^2)$$

$$\mathcal{L}_{acc} = \sum_{i=2}^{T-1}||2J(i)_{3D} - J(i-1)_{3D} - J(i+1)_{3D}||^2$$

where $V(i)$ and $J(i)$ are the coordinates of SMPL vertex set V and 3D joints J in the frame i.

4. **Consistency constraint term $\mathcal{L}_{cons}$:** This term enhances the consistency between the overlapping parts of the

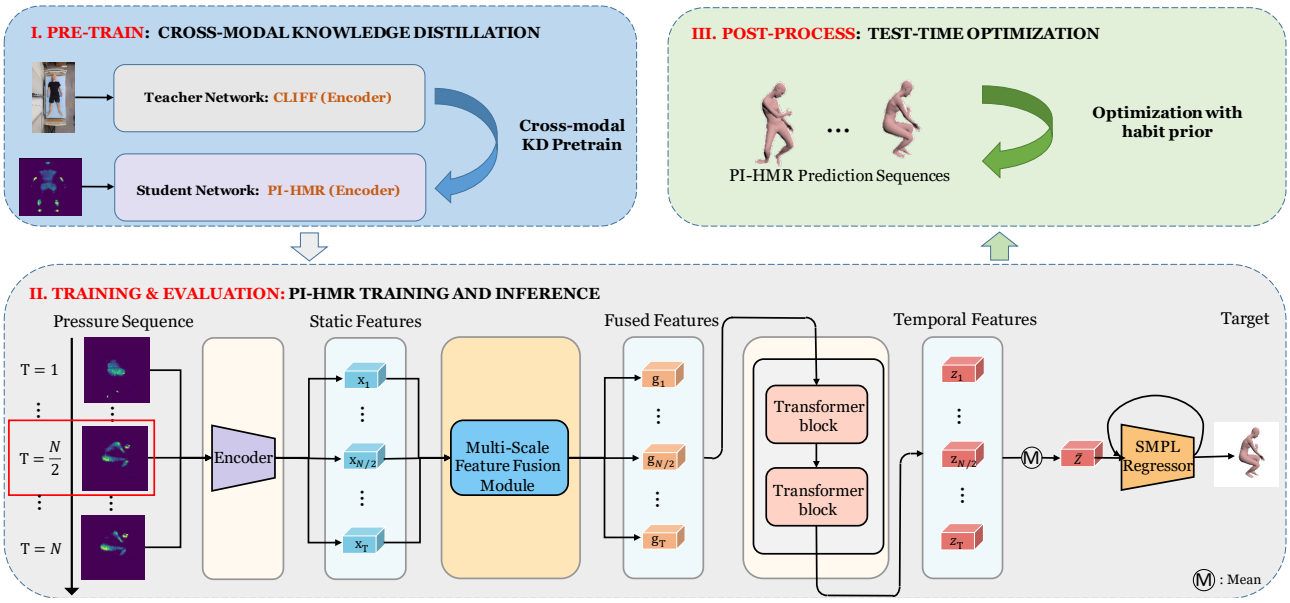

Figure 1. Our data flow includes three stages: (1) pre-train: knowledge distillation-based cross-modal pre-training; (2) train & evaluation: train the PI-HMR network with pressure sequences; (3) post-process: improve the estimates with the Test-Time Optimization strategy.

current window and the previously optimized window.

$$\begin{aligned}\mathcal{L}_{cons} = \sum_{\substack{i \in overlap \\ frames}} & (\lambda_1^{cons}||\theta(i,b_1) - \theta(i,b_2)||^2 \\ & + \lambda_2^{cons}||t(i,b_1) - t(i,b_2)||^2 \\ & + \lambda_3^{cons}||V(i,b_1) - V(i,b_2)||^2 \\ & + \lambda_4^{cons}||J(i,b_1)_{3D} - J(i,b_2)_{3D}||^2) \end{aligned} \quad (4)$$

where $t(i,b)$, $\theta(i,b)$ is the translation parameters and pose parameters in frame i of window b, and $V(i,b)$, $J(i,b)_{3D}$ is the coordinates of vertex set V, 3D joints J in frame i of window b. b_1, b_2 means the previous window and the present window, respectively.

5. **Bed contact constraint term $\mathcal{L}_{bc}$:** This term improves the plausibility of human-scene contact. We consider vertices that are close to the bed to be in contact with bed and encourage those vertices to contact with the bed plane while penalizing human-bed penetration.

$$\begin{aligned}\mathcal{L}_{bc} = \sum_{i}^{T}(\lambda^{in\_bed} & \sum_{0<z(i)_v<thre_{bed}} \tanh^2(\omega_{in\_bed} z(i)_v) \\ & + \lambda^{out\_bed} \sum_{z(i)_v<0} \tanh^2(-\omega_{out\_bed} z(i)_v)) \end{aligned} \quad (5)$$

where $z(i)_v$ is the signed distance to the bed plane of vertex v in frame i, and $thre_{bed}$ is the contact threshold and set to 0.02m.

6. **Gravity constraint term $\mathcal{L}_g$:** This term penalizes abnormal limb-lifting and reduces depth ambiguity.

$$\mathcal{L}_g = \sum_{i}^{T} \sum_{\substack{j \in G_J \\ z(i)_j > 0}} \mathbb{I}(vel(i)_j < thre_{vel}) e^{\omega(i)_j (z(i)_j)} \quad (6)$$

where $thre_{vel}$ is set to $\sqrt{110}$, and $vel(i)_j$ denotes the velocity of joint j in frame i, which is calculated from 2D annotations. $\omega(i)_j$ is a dynamic weight depends on the state of annotated 2D joint ground truths. Specifically, in addition to the velocity-based criterion, we have more complicated settings for potential corner cases. For example, when a person is seated on the bed, supporting the bed surface with both hands, the shoulders will be incorrectly judged as implausible lifts by sole velocity-based criterion (This scenario is rarely encountered in TIP, yet it still exists). In that case, we alleviate the impact of gravity constraints on this scenario by dynamically adjusting $\omega(i)_j$. In practice, when the 2D projection lengths of limbs are less than 60% of the projection lengths in the rest pose, according to geometry, we consider the corresponding limb to be normally lifted even if the corresponding joint speed is below $thre_v$, and thus $\omega(i)_j$ takes a smaller value. Besides, $\omega(i)_j$ takes a smaller value for hand joints whose 2D projections are inside the torso to avoid severe hand-torso intersection.

7. **Self-contact constraint term $\mathcal{L}_{sc}$:** This term is proposed to obtain plausible self-contact and abbreviate self-penetration. In the first stage, we only deal with the intersection between the hand and the torso. The self-contact between other body parts is optimized in the second stage.

body parts	joints & virtual joints
head	left ear, right ear, nose
torso-upper arm	left shoulder, right shoulder, spine2
left arm	left elbow, left hand, mid of left elbow and hand, $\frac{2}{5}$ point from left elbow to shoulder
right arm	right elbow, right hand, mid of right elbow and hand, $\frac{2}{5}$ point from right elbow to shoulder
torso-thigh	left hip, right hip
left thigh	left knee, left ankle, mid of left knee and ankle, $\frac{2}{5}$ point from left ankle to hip
right thigh	right knee, right ankle, mid of right knee and ankle, $\frac{2}{5}$ point from right ankle to hip

Table 1. **Positions of our selected segment centers**.

$$\mathcal{L}_{sc} = \lambda^{p\_con}\mathcal{L}_{p\_con} + \lambda^{p\_isect}\mathcal{L}_{p\_isect} + \lambda^{pull}\mathcal{L}_{pull} + \lambda^{push}\mathcal{L}_{push} \tag{7}$$

$$\mathcal{L}_{p\_con} = \sum_{0<sdf_v<thre_{dist}} \tanh^2(\omega_{p\_con} sdf_v)$$

$$\mathcal{L}_{p\_isect} = \sum_{sdf_v<0} \tanh^2(\omega_{p\_isect}|sdf_v|)$$

where sdf_v is the value of the signed distance field(SDF) at vertex v, which is calculated by our self-penetration detection algorithm. The details of $\mathcal{L}_{pull}$, $\mathcal{L}_{push}$ are given in the main body of the manuscript.

C.1.2. The second stage

We treat the results of the first stage as initialization for the second stage. Specifically, we use the mean β of each subject and fix the shape parameters in the second stage. We optimize θ and t to obtain more plausible human meshes. The objective function L_{s2} is as follows:

$$L_{s2}(\theta,\beta,t) = \lambda_J\mathcal{L}_J + \lambda_p\mathcal{L}_p + \lambda_{sm}\mathcal{L}_{sm} + \lambda_{cons}\mathcal{L}_{cons} + \lambda_{bc}\mathcal{L}_{bc} + \lambda_g\mathcal{L}_g + \lambda_{sc}\mathcal{L}_{sc} \tag{8}$$

$\mathcal{L}_J$, $\mathcal{L}_p$, $\mathcal{L}_{sm}$, $\mathcal{L}_{cons}$, $\mathcal{L}_g$, $\mathcal{L}_{bc}$ are the same as the first stage, while L_{sc} penalizes self-intersection in all body segments rather than only hands and torso.

C.1.3. Implementation details

We use Adam as the optimizer with a learning rate of 0.01, and each stage involves 500 iterations. The length of the sliding window is 128, with 50% overlapping to prevent abrupt changes between windows. The joints and virtual joints we use for the segmentation of SMPL mesh is listed in Tab. 1.

C.2. Implementation details for PI-HMR

Before the aforesaid modules in the main body of our manuscript, PI-HMR also contains three different Transformer blocks for AttentionPooling, cross-attention for sampling features in MFF, and temporal consistency extraction. We will provide detailed designs of these Transformer layers. (1) For AttentionPooling, we use the same structure in CLIP [12]. (2) For the cross-attention module in MFF, we apply a one-layer Transformer block as the attention module with one attention head and Dropout set as 0. (3) For the temporal encoder, we apply a two-layer Transformer block to extract the temporal consistency from the fusion feature sequence. In detail, each transformer layer contains a multi-head attention module with $N = 8$ heads. These learned features are then fed into the feed-forward network with 512 hidden neurons. Dropout ($p = 0.1$) and DropPath ($p_d = 0.2$) are applied to avoid overfitting.

The loss of PI-HMR is defined as:

$$\mathcal{L}_{pi} = \lambda_{\text{SMPL}}\mathcal{L}_{\text{SMPL}} + \lambda_{3D}\mathcal{L}_{3D} + \lambda_{2D}\mathcal{L}_{2D} \tag{9}$$

where $\mathcal{L}_{\text{SMPL}}$, $\mathcal{L}_{3D}$, $\mathcal{L}_{2D}$ are calculated as:

$$\mathcal{L}_{\text{SMPL}} = \omega_s^{\text{SMPL}}||\beta-\hat{\beta}||^2 + \omega_p^{\text{SMPL}}||\theta-\hat{\theta}||^2 + \omega_t^{\text{SMPL}}||t-\hat{t}||^2$$

$$\mathcal{L}_{3D} = ||J_{3D} - \hat{J}_{3D}||^2$$

$$\mathcal{L}_{2D} = ||J_{2D} - \hat{J}_{2D}||^2$$

where $\hat{x}$ represents the ground truth for the corresponding estimated variable x, and λ and ω are hyper-parameters. We set $\lambda_{\text{SMPL}} = 1$, $\lambda_{3D} = 300$, $\lambda_{2D} = 0.5$, $\omega_t^{\text{SMPL}} = \omega_p^{\text{SMPL}} = 60$, and $\omega_s^{\text{SMPL}} = 1$ for PI-HMR's training.

Before training, we first pad pressure images to 64×64 and set $T = 15$ as the sequence length. No data augmentation strategy is applied during training. During the training process, we train PI-HMR for 100 epochs with a batchsize of 16, using the AdamW optimizer with a learning rate of 3e-4 and weight decay of 5e-3. We adopt a warm-up strategy in the initial 5 epochs and schedule periodically in a cosine-like function as [17]. The weight decay is set to 5e-3 to abbreviate overfitting. All implementation codes are implemented in the PyTorch 2.0.1 framework and run on an RTX4090 GPU.

C.3. Implementation details for cross-modal KD

We conduct a HMR-based network (with a ResNet50 as encoder and an IEF [5] SMPL regressor) to pre-train the ResNet50 encoder with SOTA vision-based method Cliff [7]. The detailed structure is presented in Fig. 3 where we concurrently introduce label supervision, as well as distillation from Cliff's latent feature maps and prediction outcomes, to realize cross-modal knowledge transfer.

To train the KD-based network, like PI-HMR, we first pad pressure images to 64×64. No data augmentation

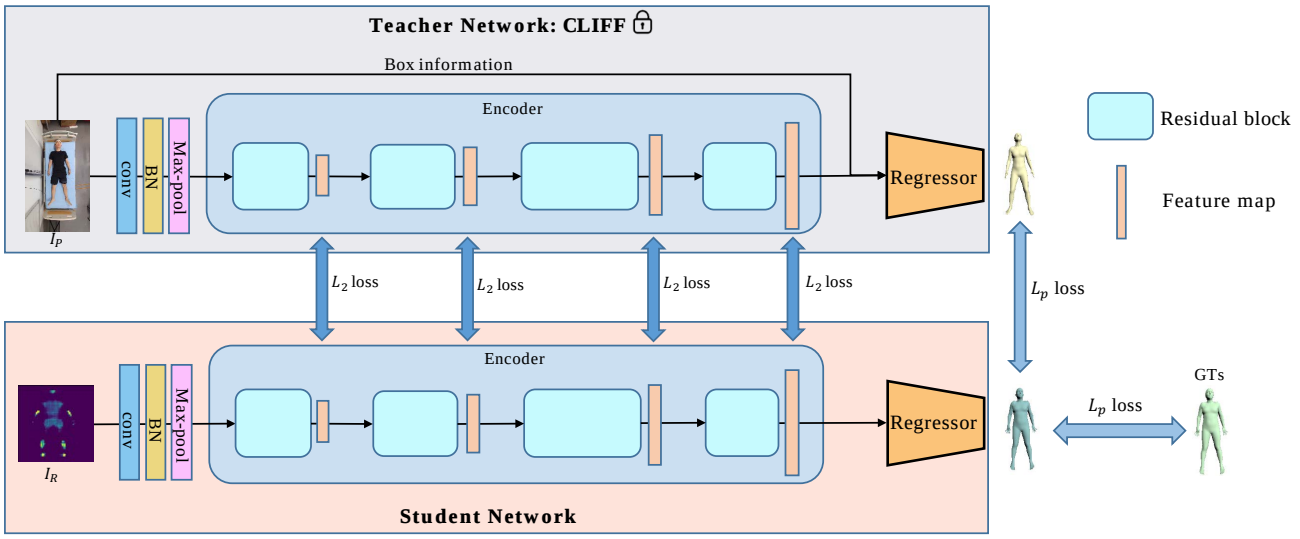

Figure 2. **An overview of our KD-based network.**

strategy is applied during training. The training process is performed for 100 epochs with an AdamW optimizer in a minibatch of 256 on the same training and validation dataset of PI-HMR. We adopt a warm-up strategy in the initial 5 epochs and schedule periodically in a cosine-like function. The weight decay is set to 5e-3 to abbreviate overfitting. All implementation codes are implemented in the PyTorch 2.0.1 framework and run on an NVIDIA. RTX4090 GPU.

C.4. Implementation details for VQ-VAE

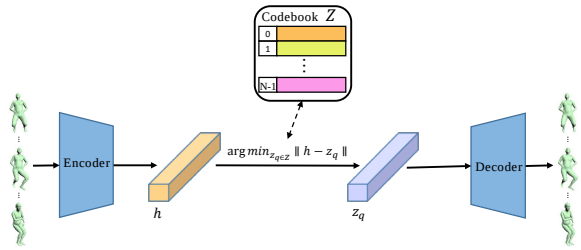

Figure 3. **An overview of our VQ-VAE network.**

The VQ-VAE follows the architecture in [3], which incorporates two 4-layer Transformer blocks as the encoder and decoder, respectively, and a $\mathbb{R}^{512\times384}$ codebook with 512 entries and $\mathbb{R}^{384}$ for the discrete latent of each entry. Each Transformer layer consists of a 4-head self-attention module and a feedforward layer with 256 hidden units.

To train the VQ-VAE network, we only input the pose parameter sequence $\Theta = \{\theta_1, ..., \theta_T\}$, without the translation and shape parameters, to push the model learning the motion continuity of the turn-over process. The pose sequences will firstly be encoded to motion features H in the Transformer encoder, quantized into discrete latent sequence Z by finding its closest element in the codebook, and reconstruct the input motion sequence in the follow-up Transformer decoder. We follow the loss setting in [3] and minimize the following loss function in Eq. (10).

$$\begin{aligned}\mathcal{L}_{vq} = \lambda_{\theta}^{vq}\text{Smooth}_{L1}(\Theta, \hat{\Theta}) \\ + \lambda_{J}^{vq}\text{Smooth}_{L1}(J_{3D}(\Theta), J_{3D}(\hat{\Theta})) \\ + \lambda_{d}^{vq}(||sg[Z] - H||_2 + \omega_{b}^{vq}||Z - sg[H]||_2)\end{aligned} \tag{10}$$

where $\hat{x}$ represents the ground truth for the corresponding estimated variable x, $\mathcal{J}(\Theta)$ means 3D joint locations of given SMPL pose parameter sequences Θ (β and t are default all-0 tensors), sg denotes the stop gradient operator, and λ and ω are hyper-parameters. We set $\lambda_{\theta}^{vq} = 1$, $\lambda_{J}^{vq} = 5$, $\lambda_{d}^{vq} = 0.25$, and $\omega_{b}^{vq} = 0.5$.

The VQ-VAE is trained with a batchsize of 64 and a sequence length of 64 frames for 100 epochs on the same training and validation dataset of PI-HMR. Adam optimizer is adapted for training, with a fixed learning rate of 1e-4, and [0.9, 0.999] for β of the optimizer. All implementation codes are implemented in the PyTorch 2.0.1 framework and run on an NVIDIA. RTX4090 GPU.

C.5. Implementation details for Test-Time Optimization

We use the VQ-VAE to act as the only motion prior and supervision in our TTO routine. For terminological convenience, given a VQ-VAE $\mathbb{M}$ and PI-HMR initial predictions $\Theta^0 = \{\theta_1^0, ...\theta_T^0\}$. For the i_{th} iteration, we calculate the loss by Eq. (11) and update the Θ by stochastic gradient descent. The result of i_{th} iteration will be input into $\mathbb{M}$ and optimized in the $i + 1_{th}$ iteration:

$$\mathcal{L}_{TTO}^{i} = \alpha\mathcal{L}_m(\Theta^i, \Theta^0) + (1-\alpha)\mathcal{L}_m(\Theta^i, \mathbb{M}(\Theta^i)) + \mathcal{L}_{sm}(\Theta^i) \tag{11}$$

where each term is calculated as:

$$\begin{aligned}\mathcal{L}_m(\Theta_1, \Theta_2) = \lambda_{\text{smpl}}^{TTO}||\Theta_1, \Theta_2||^2 \\ + \lambda_{3D}^{TTO}||\mathcal{J}(\Theta_1) - \mathcal{J}(\Theta_2)||^2\end{aligned} \tag{12}$$

$$\mathcal{L}_{sm}(\Theta) = \lambda_{sm}^{TTO} \frac{1}{T-1} \sum_{t=2}^{T} (|\Theta(t) - \Theta(t-1)| + |\mathcal{J}(\Theta(t)) - \mathcal{J}(\Theta(t-1)|$$

where $\mathcal{J}(\Theta)$ means 3D joint locations of given SMPL pose parameters Θ (β and t are the initial predictions and won't be updated during the optimization), α is a balance weight to balance initial PI-HMR predictions and reconstructions of VQ-VAE, and λs are hyperparameters. We set $\alpha = 0.5$, $\lambda_{\text{smpl}}^{TTO} = 0.5$, $\lambda_{3D}^{TTO} = 0.1$, and $\lambda_{sm}^{TTO} = 0.1$ for the test-time optimization.

During the optimization, we freeze the shape parameters β and translation parameters t of the initial PI-HMR's outputs, and only optimize pose parameters θ. We employ a sliding window of size 64 to capture the initial PI-HMR predictions and update them in 30 iterations with a learning rate of 0.01 and Adam as the optimizer. All optimization codes are implemented in the PyTorch 1.11.0 framework and run on an NVIDIA. RTX3090 GPU.

C.6. Implementation details for SOTA methods

In this section, we will provide implementation details of compared SOTA networks.

HMR [5] and HMR + KD: The implementation details of HMR series are introduced in Sec. C.3. The distinction between the two lies in whether knowledge distillation supervision is employed during the training process.

TCMR [2] and MPS-NET [16]: We choose TCMR and MPS-NET as the compared vision-based architecture because they follow the same paradigm of VIBE [6], which incorporates a static encoder for texture feature extraction, a temporal encoder for temporal consistency digestion, and a regressor for final SMPL predictions. We use the same architecture and loss weights of the default setting, except converting the initial ResNet50 input to a single channel and adjusting the first convolution layer's kernel size to 5×5 to fit the single-channel pressure images.

PI-Mesh [17]: PI-Mesh is the first-of-its-kind temporal network to predict in-bed human motions from pressure image sequences. We follow the codes and implementation details provided in [17] with a ResNet50 as the static encoder and a two-layer Transformer block as the temporal encoder.

BodyMAP-WS: BodyMap [14] is a SOTA dual-modal method to predict in-bed human meshes and 3D contact pressure maps from both pressure images and depth images. We realize a substitute version provided in their paper, named BodyMap-WS, because we don't have 3D pressure map labels. It is worth mentioning that we notice the TIP dataset fails to converge on the algorithm provided in their GitHub repository. So we remove part of the codes including rotation data augmentation and post-processing of the network outputs (Line 139-150 and Line 231-242 in the *PMM/MeshEstimator.py* of the GitHub repository) to ensure convergence.

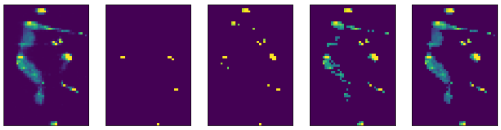

Figure 4. **Visualization of TopK Sampling.**

All methods are trained on the same training-validation dataset of PI-HMR. For TCMR, MPS-NET, and PI-Mesh, we adopt the same training routine as PI-HMR. To be specific, we first pad pressure images to 64×64 and set $T = 15$ as the sequence length. No data augmentation strategy is applied during training. During the training process, we train these approaches for 100 epochs with a batchsize of 16, using the AdamW optimizer with the learning rate of 3e-4 and weight decay of 5e-3 (we firstly conduct a simple grid-search for the best learning rate selection on these methods), and adopt a warm-up strategy in the initial 5 epochs and scheduled periodically in a cosine-like function. For BodyMap-WS, we follow the training routine provided in [14], resize the pressure images to 224×224, and apply RandomAffine, RandomCutOut, and PixelDropout as data augmentation strategies. The training process is performed for 100 epochs with an Adam optimizer in a minibatch of 32, a learning rate of 1e-4 and weight decay of 5e-4. All codes are implemented in the PyTorch 2.0.1 framework and run on an NVIDIA. RTX4090 GPU.

D. More ablations

D.1. Discussion on TopK sampling

The sampling functions as a low-value filter, freeing the model's attention from redundant, noisy backgrounds and focusing more on high-value regions. We provide a visualization in Fig. 4, where, with 128 points, the pressure image can retain the human's outline while highlighting the core contact areas.

D.2. Comparisons with single-input models

For vision methods, single-image models usually exhibit lower MPJPE compared to temporal models (*e.g.* CLIFF vs PMCE). However, for pressure data, temporal models show superiority, likely due to their ability to leverage temporal context, mitigating information ambiguity. This implies the strength of temporal models in pressure data processing compared to single ones. For fair comparisons, we implemented a single-input-based PI-HMR, achieving a 62.01mm MPJPE (71.48mm for BodyMAP-WS), showing the efficacy of our architecture framework.

Method	TCMR	PI-Mesh	PI-HMR
MPJPE/ACC-ERR	67.9/14.6	79.2/18.2	**68.38/5.24**

Table 2. **Quantitative results on the original TIP dataset.**

α	0.1	0.3	0.5	0.7	0.9
MPJPE	56.94	55.93	55.50	**55.43**	55.67

Table 3. **Ablations on balance weight α.**

iters	10	30	50	70	90
MPJPE	56.14	55.50	55.25	55.15	**55.10**

Table 4. **Ablations on the number of iterations.**

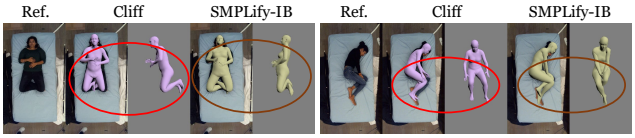

Figure 5. **Visualizations of SMPLify-IB on SLP.**

D.3. Results on the original TIP dataset

The results are shown in Tab. 2, which demonstrate a comparable magnitude of MPJPE reduction, proving the efficacy of PI-HMR.

D.4. Ablations of TTO.

We conducted ablations involving the selection of the balance weight α in Tab. 3 and the number of iterations in Tab. 4. We also explored integrating the pre-trained VQ-VAE into PI-Mesh during training (as it regresses the sequence rather than the mediate frame, making it suitable for VQ-VAE) and calculating the reconstruction loss. However, MPJPE drops limitedly (0.06mm). We will explore more potential methods (*e.g.* SPIN-like) in the future work.

D.5. Generalization of SMPLify-IB on the SLP dataset.

We implemented SMPLify-IB on the SLP dataset. Results show the 2D MPJPE drops from 37.6 to 6.9 pixels compared to Cliff's outputs. Fig. 5 shows our pros in alleviating depth ambiguity. Meanwhile, we observed limb distortions in the optimization results, which may stem from erroneous initial estimations (CLIFF exhibits notable domain adaptation issues in an in-bed scene). In the absence of temporal context, these mis-predictions could exacerbate the likelihood of unreasonable limb angles, underscoring the significance of temporal information in in-bed human shape annotations.

E. Visualization results

In this section, we present additional visualization results to verify the efficiency of our general framework for the in-bed HPS task.

E.1. Visualizations for Time Consumption of self-penetration algorithms

(a) SMPLify-IB with Self-Contact (12796.8s) (b) SMPLify-IB with ours (28.8s)

Figure 6. **Time consumption when deploying the two self-penetration detection and computation algorithms in our optimization routine.** We count the time taken in an optimization stage with 500 iterations on a single batch (128 frames) and document the proportion of time spent by the self-penetration modules in the overall duration (in deep blue).

Fig. 6 provides quantitative comparisons on time consumption of our optimization routines with SOTA self-penetration algorithm (Self-Contact in SMPLify-XMC [9]) and our proposed light-weight approach (downsample 1/3 version). The experiment is conducted on a NVIDIA. 3090 GPU, with each optimization performing with 500 iterations on a single batch (128 frames). While the Self-Contact algorithm yields high detection accuracy, it comes at a significant time and computational expense (*i.e.*, nearly 100s per frame on a RTX3090 GPU). Our detection module brings nearly 450 times speed while archiving comparable self-penetration refinement.

E.2. Visualizations for gravity-based constraints.

Fig. 7 provides more visual evidence on the efficiency of our gravity constraints in SMPLify-IB. Traditional single-view regression-based method (yellow meshes by Cliff) and optimization-based method (red meshes by a SMPLify-like approach adopted in TIP) face serious depth ambiguity in the in-bed scene, especially when limbs overlap from the camera perspective, thus leading to implausible limb lifts (*e.g.*, hand lifts in the first and third rows in Fig. 7, and leg lifts when legs contact and overlap in the third row). Our proposed gravity constraints, accompanied by a strong self-penetration detection and penalty term, effectively alleviate the depth ambiguity issue while maintaining reasonable contact. This validates the feasibility of alleviating depth ambiguity issues with physical constraints in specific scenarios.

E.3. Failure cases for SMPLify-IB

About 1.6% samples of our optimization results might fail due to severely false initialization by CLIFF, wrong judgment in gravity constraints, and trade-offs in the multiple-term optimization, as presented in Fig. 8. Thus

Figure 7. **Qualitative comparisons on the p-GTs generated by Cliff (predicted on images), TIP and our generations by SMPLify-IB.** We highlight the implausible limb lifts by single-view depth ambiguity in red ellipses and our refinement with yellow ellipses.

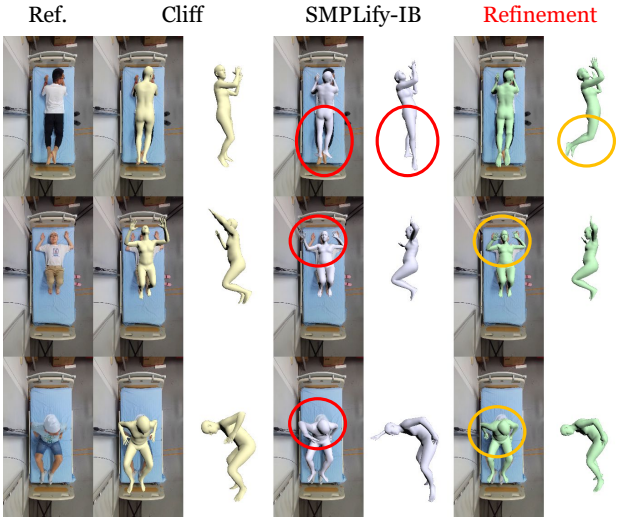

Figure 8. **Typical failure cases of SMPLify-IB.** We highlight the wrong generations with red markers and our refinement in the yellow ellipses.

we manually inspected all generated results and carried out another round of optimization to address these errors, aiming at generating reliable p-GTs for the TIP dataset. The refinement is highlighted with yellow ellipses in Fig. 8.

E.4. Failure cases for PI-HMR

In Fig. 9, we show a few examples where PI-HMR fails to reconstruct reasonable human bodies. The reason mainly falls in the information ambiguities, ranging from (a) PI-HMR mistakenly identifies the contact pressure between the foot and the bed as originating from the leg (shown in the red ellipse), (b) hand lifts and (c) leg lifts.

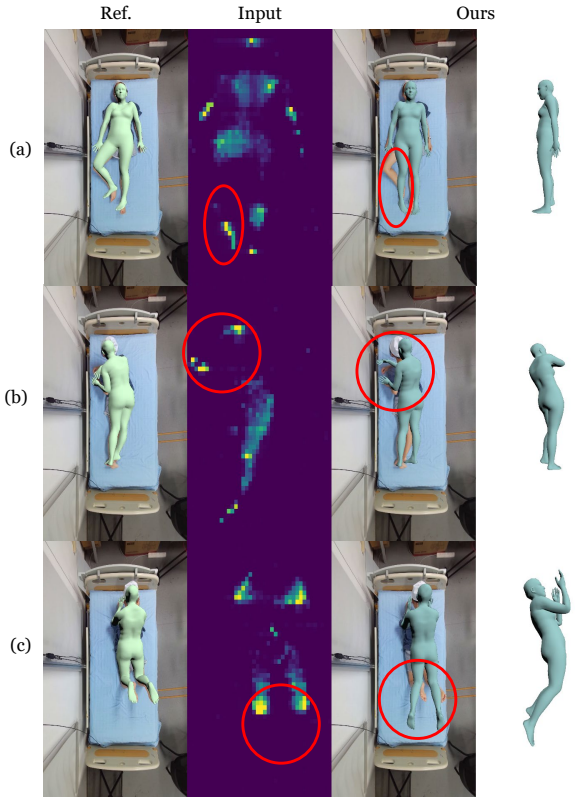

Figure 9. **Typical failure cases of PI-HMR.** We highlight the mis-predictions and corresponding pressure regions with red markers.

E.5. More Qualitative Visualizations

We present more qualitative visualizations on the performance of our proposed optimization strategy SMPLify-IB in Fig. 10 and PI-HMR in Fig. 11 and Fig. 12.

F. Limitations and Future works

we conclude our limitations and future works in three main aspects:

(1) **Hand and foot parametric representations:** More diverse and flexible tactile interactions exist in the in-bed scenarios. For instance, the poses of the hands and feet vary with different human postures, thereby influencing the patterns of localized pressure. However, the SMPL model fails to accurately depict the poses of hands and feet, thereby calling for more fine-grained parametric body representations [10, 11] to precisely delineate the contact patterns between human bodies and the environment.

(2) **Explicit constraints from contact cues:** In this work, we propose an end-to-end learning approach to predict human motions directly from pressure data. The learning-based pipeline can rapidly sense the pressure distribution patterns and generate high-quality predictions from pressure sequences, yet it may lead to underutilization of contact priors from pressure sensors and cause misalignment between limb position and contact regions (*e.g.*, torso and limbs lift). In future works, we aim to explicitly incorporate contact priors through learning or optimization methods [13] to further enhance the authenticity of the model's predictions.

(3) **Efforts for information ambiguity:** In this work, we aspire to mitigate the information ambiguity issue through pressure-based feature sampling and habit-based Test-Time Optimization strategies, yielding accuracy improvement; however, challenges persist. Building upon the observation that users perform movements in certain habitual patterns, we expect to develop a larger-scale motion generation model reliant on VQ-VAE [15] or diffusion [4] techniques, to address the deficiencies in single-pressure modality based on users' motion patterns.

Figure 10. **Qualitative results of our generated p-GTs on the TIP dataset.** We compare our results with SOTA vision-based methods Cliff and TCMR (predicted on RGB images) and p-GTs provided in TIP.

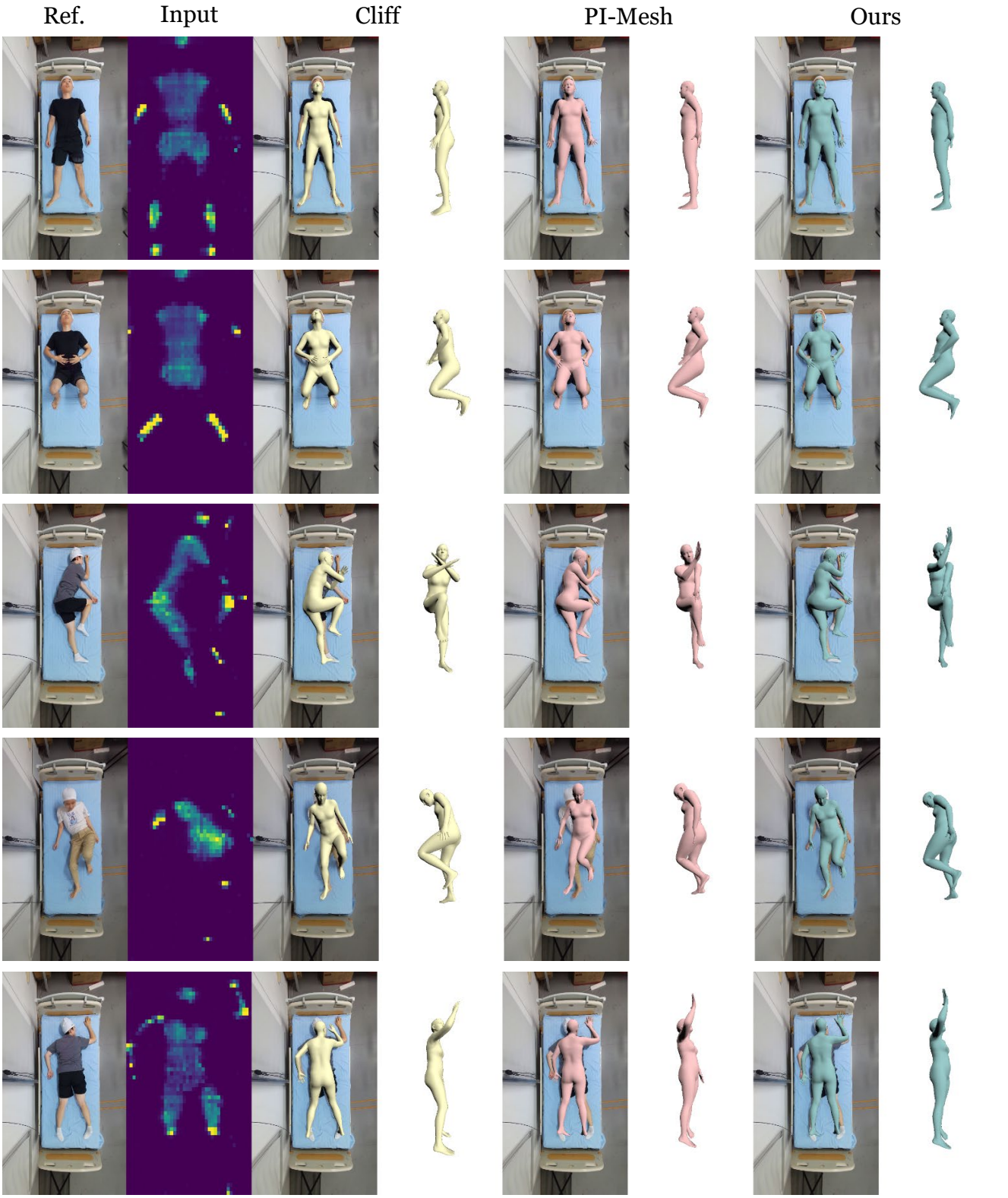

Figure 11. **Qualitative results of PI-HMR's performance on the TIP dataset.** We compare our results with SOTA vision-based methods Cliff (predicted on RGB images) and pressure-based method PI-Mesh.

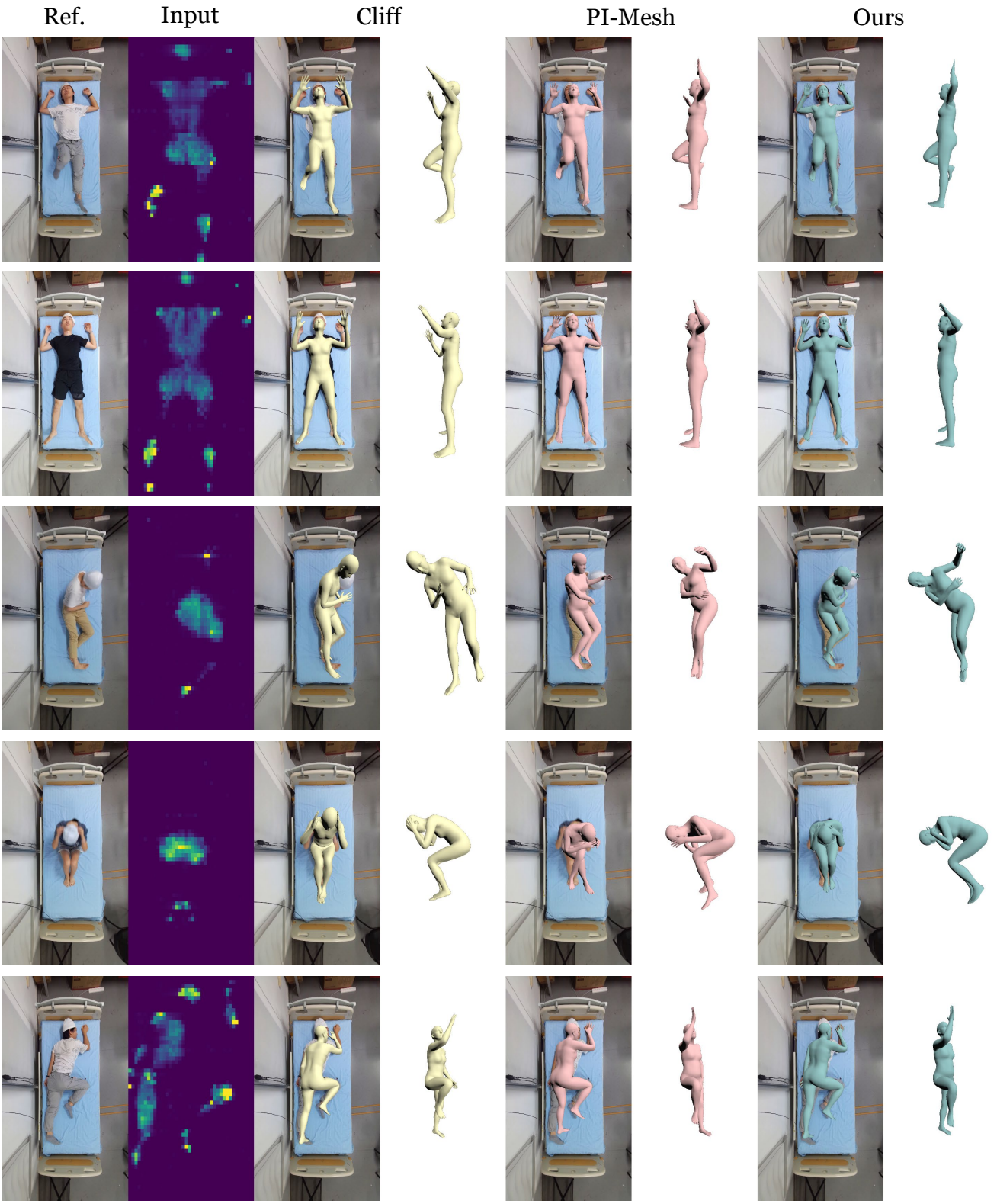

Figure 12. **More qualitative results of PI-HMR's performance on the TIP dataset.**

References

[1] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *ECCV*, pages 561–578. Springer, 2016. 1

[2] Hongsuk Choi, Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. Beyond static features for temporally consistent 3d human pose and shape from a video. In *CVPR*, pages 1964–1973, 2021. 5

[3] Han Feng, Wenchao Ma, Quankai Gao, Xianwei Zheng, Nan Xue, and Huijuan Xu. Stratified avatar generation from sparse observations. In *CVPR*, pages 153–163, 2024. 4

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NIPS*, 33:6840–6851, 2020. 8

[5] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *CVPR*, pages 7122–7131, 2018. 3, 5

[6] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. In *CVPR*, pages 5253–5263, 2020. 5

[7] Zhihao Li, Jianzhuang Liu, Zhensong Zhang, Songcen Xu, and Youliang Yan. Cliff: Carrying location information in full frames into human pose and shape estimation. In *ECCV*, pages 590–606. Springer, 2022. 4

[8] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 851–866. 2023. 1

[9] Lea Muller, Ahmed AA Osman, Siyu Tang, Chun-Hao P Huang, and Michael J Black. On self-contact and human pose. In *CVPR*, pages 9990–9999, 2021. 6

[10] Ahmed AA Osman, Timo Bolkart, Dimitrios Tzionas, and Michael J Black. Supr: A sparse unified part-based human representation. In *ECCV*, pages 568–585. Springer, 2022. 7

[11] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. Expressive body capture: 3d hands, face, and body from a single image. In *CVPR*, pages 10975–10985, 2019. 7

[12] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763. PMLR, 2021. 3

[13] Soshi Shimada, Vladislav Golyanik, Patrick Pérez, and Christian Theobalt. Decaf: Monocular deformation capture for face and hand interactions. *TOG*, 42(6):1–16, 2023. 8

[14] Abhishek Tandon, Anujraaj Goyal, Henry M Clever, and Zackory Erickson. Bodymap-jointly predicting body mesh and 3d applied pressure map for people in bed. In *CVPR*, pages 2480–2489, 2024. 5

[15] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *NIPS*, 30, 2017. 8

[16] Wen-Li Wei, Jen-Chun Lin, Tyng-Luh Liu, and Hong-Yuan Mark Liao. Capturing humans in motion: Temporal-attentive 3d human pose and shape estimation from monocular video. In *CVPR*, pages 13211–13220, 2022. 5

[17] Ziyu Wu, Fangting Xie, Yiran Fang, Zhen Liang, Quan Wan, Yufan Xiong, and Xiaohui Cai. Seeing through the tactile: 3d human shape estimation from temporal in-bed pressure images. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 8(2):1–39, 2024. 1, 3, 5