IKOL: Inverse Kinematics Optimization Layer for 3D Human Pose and Shape Estimation via Gauss-Newton Differentiation

Juze Zhang[1,2,3,4], Ye Shi[1,4*], Yuexin Ma[1,4], Lan Xu[1,4], Jingyi Yu[1,4], Jingya Wang[1,4*]

[1] ShanghaiTech University
[2] Shanghai Advanced Research Institute, Chinese Academy of Sciences
[3] University of Chinese Academy of Sciences
[4] Shanghai Engineering Research Center of Intelligent Vision and Imaging
{zhangjz,shiye,mayuexin,xulan1,yujingyi,wangjingya}@shanghaitech.edu.cn

Abstract

This paper presents an inverse kinematic optimization layer (IKOL) for 3D human pose and shape estimation that leverages the strength of both optimization- and regression-based methods within an end-to-end framework. IKOL involves a nonconvex optimization that establishes an implicit mapping from an image's 3D keypoints and body shapes to the relative body-part rotations. The 3D keypoints and the body shapes are the inputs and the relative body-part rotations are the solutions. However, this procedure is implicit and hard to make differentiable. So, to overcome this issue, we designed a Gauss-Newton differentiation (GN-Diff) procedure to differentiate IKOL. GN-Diff iteratively linearizes the nonconvex objective function to obtain Gauss-Newton directions with closed form solutions. Then, an automatic differentiation procedure is directly applied to generate a Jacobian matrix for end-to-end training. Notably, the GN-Diff procedure works fast because it does not rely on a time-consuming implicit differentiation procedure. The twist rotation and shape parameters are learned from the neural networks and, as a result, IKOL has a much lower computational overhead than most existing optimization-based methods. Additionally, compared to existing regression-based methods, IKOL provides a more accurate mesh-image correspondence. This is because it iteratively reduces the distance between the keypoints and also enhances the reliability of the pose structures. Extensive experiments demonstrate the superiority of our proposed framework over a wide range of 3D human pose and shape estimation methods. Code is available at https://github.com/Juzezhang/IKOL.

Introduction

Recent years have seen the widespread application of 3D human pose and shape (3D-HPS) estimation to a range of fields, including video analysis, camera surveillance, human computer interaction, virtual/augmented reality, and others. Among the leading techniques, two popular paradigms for 3D-HPS have been investigated: optimization-based methods and regression-based methods. Optimization-based methods (Bogo et al. 2016; Fan et al. 2021) iteratively fit parametric models (such as SMPL (Loper et al. 2015)) to 2D observations. They can produce accurate mesh-image alignments, and they can also preserve consistency between some input data and a prediction. However, fitting the parametric models to 2D observations is an inherently ill-posed problem; it is also non-convex. This means various optimization techniques are required to find a local minimum and compute an initial guess. Worse still, this procedure is time-consuming and the final accuracy of the estimation relies heavily on the initial guess. Conversely, regression-based methods treat 3D-HPS as a regression problem, where a deep network is trained to directly regress the model's parameters thanks to the powerful nonlinear mapping capability of neural networks. However, despite efficient and promising results, regression-based methods tend to suffer from misalignments between the estimated meshes and the image.

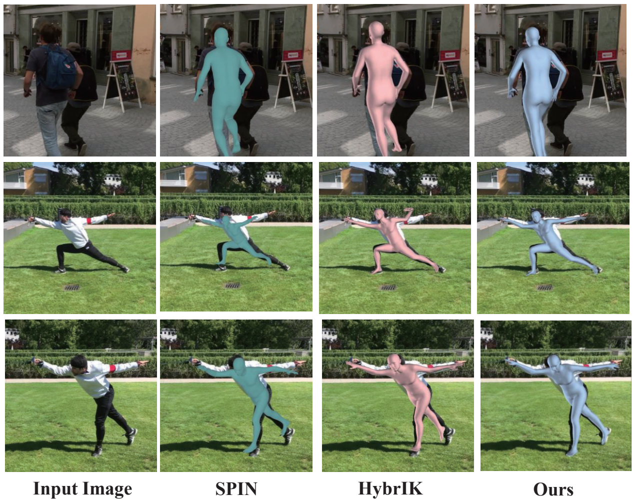

Figure 1: Qualitative comparison with the state-of-the-art method SPIN (Kolotouros et al. 2019) and HybrIK (Li et al. 2021).

Some pioneering studies have attempted to integrate these two paradigms, coupling an optimization process with a regression technique (Zanfir et al. 2021; Guler and Kokkinos 2019; Kolotouros et al. 2019). These works either use post-

[*] Corresponding author.

optimization schemes to refine the regression results during the inference stage, or they use optimization to provide a supervision signal to the optimization scheme in the training loop. The former approaches ((Zanfir et al. 2021; Guler and Kokkinos 2019)) tends to rely heavily on the quality of the regression result, while the latter approach (SPIN, (Kolotouros et al. 2019)) can suffer from an instability during training process due to inaccurate supervision and initialization sensitivity. Thus, to date, trying to leverage the best of both works still requires further investigation. Recently, (Li et al. 2021) proposed HybrIK by deriving analytical solutions to handle inverse kinematics via a twist-and-swing decomposition based on the assumption that the bone lengths are consistent. Although this simplification is effective, using this strategy can sacrifice some accuracy of the inverse kinematics. This is because assuming that bone lengths are consistent will usually result in non-trivial errors, stemming from potential regression errors and the measurements of the 2D and 3D keypoints.

To alleviate these issues, we designed an inverse kinematics optimization layer (IKOL) that leverages the strengths of both optimization and regression for end-to-end 3D-HPS estimation. IKOL employs a twist-and-swing decomposition and an optimization process that does not rely on the simplified assumption of bone lengths. In contrast to SPIN, IKOL's optimization process of is very fast and very effective because it benefits from decomposition. It is worth noting that IKOL involves a nonconvex optimization with the 3D keypoints and body shape as inputs and the relative body-part rotations as the solutions, which is hard to make differentiable. In recent years, some solvers such as Optnet (Amos and Kolter 2017), CvxpyLayer (Agrawal et al. 2019) and Alt-Diff (Sun et al. 2022a) have been developed to differentiate convex optimizations. Although these solvers have achieved considerable success, differentiation schemes for nonconvex optimizations are still quite open. The problem with 3D-HPS estimation is that it is highly nonlinear and also nonconvex. This motivated us to develop an efficient differentiation procedure. Called Gauss-Newton differentiation (GN-Diff), this scheme iteratively linearizes the nonconvex objective function to obtain Gauss-Newton directions with closed form solutions. Thus, an automatic differentiation procedure can be directly applied to generate a Jacobian matrix for end-to-end training. Notably, the GN-Diff procedure is quite fast because it does not rely on a time-consuming implicit differentiation procedure. In summary, our work makes three main contributions to the literature:

- IKOL: which leverages the strengths of both the optimization and regression for end-to-end 3D-HPS estimation. IKOL has a much lower computational overhead than most existing optimization-based methods and provides a more accurate mesh-image correspondence than existing regression-based methods.
- GN-Diff: which efficiently differentiates the nonconvex optimization in IKOL. Notably, existing solvers of optimization layers cannot differentiate nonconvex optimization problems. In addition, our GN-Diff procedure is very fast because it benefits from iterative closed form solutions via Gauss-Newton iterations.
- State-of-the-art performance: IKOL with GN-Diff yields state-of-the-art performance with a wide range of 3D human pose and shape benchmarks. Notably, the time cost of the optimization branch is much faster than the competing optimization procedure used in SPIN and is even competitive with the analytical solution proposed in HybrIk.

Related work

Optimization-Based Methods

The current optimization methods usually fit a parametric model (such as SMPL (Loper et al. 2015)) to image cues. Based on this formulation, several image cues have been designed as fitting terms for object functions, including 2D keypoints (Bogo et al. 2016), 3D keypoints (Mehta et al. 2017b, 2020; Zhang et al. 2022), silhouettes (Huang et al. 2017; Lassner et al. 2017), part segmentation(Zanfir, Marinoiu, and Sminchisescu 2018), part orientation fields (Xiang, Joo, and Sheikh 2019), and more(Han et al. 2023; Liang et al. 2023). However, only using fitting term may lead to unnatural and unrealistic shapes and poses. Alleviating this issue relies on several prior terms, such as a mixture of Gaussians (Bogo et al. 2016), a variational autoencoder (Pavlakos et al. 2019), or a normalizing flow (Zanfir et al. 2020). Yet, despite the good performance of optimization-based methods with image-mesh alignment, most treat the fitting scheme as a separate task. As a consequence, accuracy relies heavily on estimated observations. By contrast, our method formulates the problem as an optimization layer and learns the entire regression and optimization together in an end-to-end manner.

Regression-Based Methods

There has been a recent trend to treat 3D-HPS estimation as a regression problem, i.e., where the model's parameters are directly regressed thanks to the powerful nonlinear mapping capability of deep neural networks. Since the ground truth of the model's parameters, or the model mesh, is rarely available, some have used 2D annotations as a form of weak supervision by enforcing different kinds of reprojection loss. The 2D annotations used include 2D keypoints(Kanazawa et al. 2018; Pavlakos et al. 2018), silhouettes(Pavlakos et al. 2018; Varol et al. 2018), parts segmentation(Omran et al. 2018; Rueegg et al. 2020) and dense correspondences (Xu, Zhu, and Tung 2019; Zhang et al. 2020). To overcome occlusion issues, a few studies (Zhang et al. 2021; Sun et al. 2021, 2022b; Jiang et al. 2020) have involved one-stage networks, which simultaneously regress multiple 3D people. Yet, despite their effectiveness, mappings from an image to a parametric model are hard to learn, and so the strategy tends not to preserve accurate image-mesh alignments. By comparison, IKOL involves an optimization that does provide an accurate mesh-image correspondence by iteratively reducing the distance between keypoints.

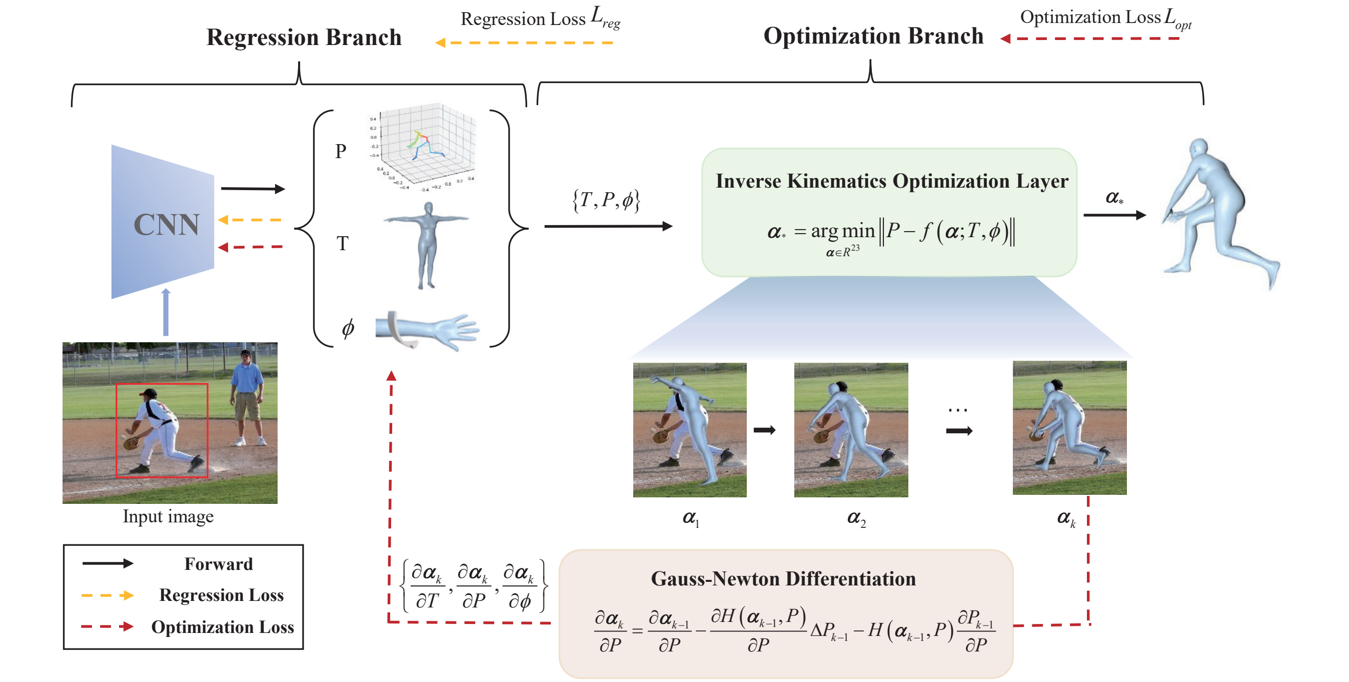
Regression Loss L_{reg}
Regression Branch
Optimization Branch
Optimization Loss L_{opt}
CNN
P
T
ϕ
$\{T,P,\phi\}$
Inverse Kinematics Optimization Layer
$\alpha_* = \arg\min_{\alpha\in R^{23}} \|P - f(\alpha;T,\phi)\|$
α_*
Input image
α_1
α_2
…
α_k
Forward
Regression Loss
Optimization Loss
$\left\{\frac{\partial\alpha_k}{\partial T},\frac{\partial\alpha_k}{\partial P},\frac{\partial\alpha_k}{\partial\phi}\right\}$
Gauss-Newton Differentiation
$\frac{\partial\alpha_k}{\partial P}=\frac{\partial\alpha_{k-1}}{\partial P}-\frac{\partial H(\alpha_{k-1},P)}{\partial P}\Delta P_{k-1}-H(\alpha_{k-1},P)\frac{\partial P_{k-1}}{\partial P}$

Figure 2: Overview of the proposed inverse kinematics optimization layer(IKOL) framework. First, a CNN backbone is utilized to regress the 3D joints P, the rest pose T and the twist angle ϕ. Second, the inverse kinematics optimization layer involves an optimization scheme that establishes the mapping from the outputs of the regression branch to the swing angle $\boldsymbol{\alpha}$. IKOL is trained in an end-to-end differentiable framework by using GN-Diff.

Hybrid of Optimization & Regression Methods

Several research teams have tried to integrate optimization processes with regression methods for 3D-HPS estimation. So as to use the iterative fitting strength of an optimization method, some studies have adopted a post-optimization scheme which refines their results after the regression is complete. This ensures the output aligns with the intermediate estimations, such as part segmentation(Zanfir et al. 2021), and dense correspondences(Guler and Kokkinos 2019). However, typically, the post-optimization results rely heavily on the quality of the regression results and are sensitive to the initialization. Unlike the above post-optimization methods, SPIN (Kolotouros et al. 2019) performs its optimization in the training loop as a way to provide a supervision signal for the regression branch. However, this type of supervision has a tendency to be inaccurate and is quite sensitive to the initialization settings. Often, this means the training process is unstable. To avoid an iterative fitting process to find the optimal solution, recent studies (Li et al. 2021; Moon and Lee 2020; Yu et al. 2021) have suggested building a collaboration between the 3D joints and the body mesh. Although an analytical solution can be produced to replace the optimization procedure by removing the twist components learned from a neural network, this method assumes the bone length consistency hypothesis. Consequently, inverse kinematic accuracy may suffer. In contrast to HybrIK, IKOL formulates the inverse kinematics problem as an iterative optimization process instead of as an approximated analytical derivation.

Optimization Layer

Embedding the optimization process as a layer in a DNN has emerged as a promising new direction in building models (Amos and Kolter 2017; Agrawal et al. 2019; Gould, Hartley, and Campbell 2021; Sun et al. 2022a). Optnet (Amos and Kolter 2017), for example, integrates quadratic optimization into a DNN and implicitly differentiates the Karush–Kuhn–Tucker (KKT) conditions for an end-to-end training scheme. Subsequently, CvxpyLayer (Agrawal et al. 2019) was devised. This framework differentiates a general convex optimization based on conic operators and has been applied to many tasks, such as optimizing rank metrics (Rolínek et al. 2020a) and graph matching (Rolínek et al. 2020b). However, these methods only work for convex optimization; they are not suitable for the nonconvex problems associated with 3D-HPS estimation. For this reason, we designed GN-Diff, which efficiently does this differentiation work.

Method

In this section, we present the IKOL method and the GN-Diff procedure, which together create an end-to-end learning framework for 3D-HPS estimation. Within the model, a regression branch predicts the 3D joints, and the IKOL layer estimates the 3D keypoints to boost the 3D body mesh estimation. Our pipeline for IKOL is illustrated in Fig. 2

Inverse Kinematics Optimization Layer

In this paper, we used SMPL(Loper et al. 2015) as the body model, which provides a differentiable function $\mathcal{M}(\theta, \beta)$ to control an artist-created mesh with $N = 6890$ vertices and $K = 23$ joints. The body shape is parameterized by the first 10 principal components $\beta \in \mathbb{R}^{10}$. Following standard skinning practice, the zero pose θ^* results in the mean template shape $\hat{T} \in \mathbb{R}^{6890}$. The rest pose of joints $T \in \mathbb{R}^{23}$ are obtained via a linear regressor mapping from $\hat{T}$. The relative axis angle of the joints are parameterized by $\theta \in \mathbb{R}^{3\times 23}$ and usually transformed into a relative rotation matrix $R = \{R_{par(i),i}\}_{i=1}^{K}$ using Rodrigues' formula. Here, par(i) denotes the parent of the body joint i and $R_{par(i),i} \in \mathbb{SO}(3)$ denotes the i-th local rotation matrix with respect to its parent joint. We denote the ith global rotation matrix as $R_i \in \mathbb{SO}(3)$. With this, the global rotation matrix of the body joint i can be recursively computed from relative rotation matrix as:

$$R_i = R_{par(i)} R_{par(i),i}. \tag{1}$$

It is worth noting that the rotation estimated from the regressed joints P are inherently ambiguous as the twist angle is missing from the skeleton representation. Hence, we followed (Li et al. 2021) and decomposed the rotation R into a twist rotation R^{tw} and a swing rotation R^{sw} by applying twist-and-swing decomposition (Baerlocher and Boulic 2001). Thus, each rotation matrix can be computed as:

$$R_i = R_i^{sw}(\boldsymbol{\alpha}_i) R_i^{tw}(\phi_i) \tag{2}$$

where $\boldsymbol{\alpha}_i \in \mathbb{R}$ denotes the swing angle of ith rotation that will optimized by our optimization branch and $\phi_i \in \mathbb{R}$ is the twist angle of ith rotation that will be predicted by the regression branch. Readers interested in further derivation details of the swing rotation $R^{sw}(\boldsymbol{\alpha})$ and twist rotation $R^{tw}(\phi)$ are referred to (Li et al. 2021). Notably, pelvis joints and the leaf joints are excluded in this decomposition. Because the pelvis joints have three children which can be derived by closed-form solution with the children joints of the spine, left hip and right hip. Thus, the forward kinematics of SMPL are defined as:

$$Q = FK(R^{sw}(\boldsymbol{\alpha}); R^{tw}(\phi), T, P) := f(\boldsymbol{\alpha}), \tag{3}$$

where $Q \in \mathbb{R}^{23}$ denotes the reconstructed 3D joints from the SMPL outputs, P denotes the regressed 3D joints estimated from the regression branch, $\boldsymbol{\alpha} \in \mathbb{R}^{23}$ denotes the swing angle, and $f(\cdot)$ denotes the forward model operator for the simplification.

Directly regressing the model parameters is non-trivial due to the highly non-linear mapping between the images cues and the articulated pose. Usually, such a procedure leads to image-model misalignment. By contrast, 3D keypoint representation predicts the per-pixel likelihood for each joint's location as a 3D heatmap, which can preserve the spatial relationships in the input image.

Similar to the network design in HybrIK (Li et al. 2021), we also use a deep neural network to predict the 3D joints P, the twist angle ϕ, and the shape parameters β. We use the common supervision of 3D joints, and shape and twist angles as follows:

$$L_{reg} = w_1 L_{shape} + w_2 L_{twist} + w_3 L_{joint}, \tag{4}$$

where L_{shape} is the ℓ_2 loss of the shape parameters β, L_{twist} is the ℓ_2 loss of the twist angle β, L_{joint} is the ℓ_1 loss of the regressed joints between the groundtruth and $w_{(\cdot)}$ is the corresponding loss weights. Once the regressed joints P, the twist angle ϕ, and the shape parameters β have been obtained from regression neural network, IKOL can be used to obtain the corresponding solution angle:

$$\boldsymbol{\alpha}_* = \arg\min_{\boldsymbol{\alpha}} \frac{1}{2} \|P - f(\boldsymbol{\alpha})\|_2^2. \tag{5}$$

Then, the optimization loss is designed to compute the loss with ground truth of the rotation as follows:

$$L_{opt} = \|R^{sw}(\boldsymbol{\alpha}^*) - R^{gt}\|_1, \tag{6}$$

where R^{gt} denotes the ground truth of the rotation. To sum up, IKOL is supervised by the weighted sum of the regression loss L_{reg} and optimization loss L_{opt}, The total loss L_{total} is derived as

$$L_{total} = L_{reg} + w_4 L_{opt}, \tag{7}$$

where w_4 denotes the corresponding optimization loss weights.

The above optimization (Eq. 5) is highly nonconvex, which is hard to train together with the regression branch. As we analyzed before, existing optimization differentiation solvers, such as OptNet (Amos and Kolter 2017) and CvxPylayer (Agrawal et al. 2019), are not suitable the nonconvex optimization. In this paper, we develop a Gauss-Newton Differentiation to train the above nonconvex optimization with the regression branch under an end-to-end framework.

Gauss-Newton Differentiation

As the forward model operator $f(\cdot)$ is nonlinear, optimization first requires linearizing this problem. Denotes $\boldsymbol{\alpha}_{k-1}$ as the swing angle at $k-1$th iteration. To this end, Taylor's series expansion is used to approximate $f(\cdot)$ at $\boldsymbol{\alpha}_{k-1}$ in the Eq. 3, i.e.:

$$f(\boldsymbol{\alpha}) \approx f(\boldsymbol{\alpha}_{k-1}) + J_{k-1}(\boldsymbol{\alpha} - \boldsymbol{\alpha}_{k-1}), \tag{8}$$

where J_{k-1} denotes the Jacobian matrix calculated from the $(k-1)$ th iteration. Here the Jacobian matrix J_{k-1} is associated with the input of swing angle $\boldsymbol{\alpha}$, twist rotation R^{tw}, rest pose T and regressed joints P. To ensure the whole process is differentiable, the key is to derive J_{k-1} with analytical formulation in the kinematics tree. Detailed derivation for J_{k-1} is provided in supplementary.

Linearizing the Eq. 3 results in

$$\Delta\boldsymbol{\alpha}_k = \arg\min_{\Delta\boldsymbol{\alpha}} \frac{1}{2} \|J_{k-1}\Delta\boldsymbol{\alpha} - \Delta P_{k-1}\|_2^2, \tag{9}$$

where $\Delta P_{k-1} = P - f(\boldsymbol{\alpha}_{k-1})$ is the residue, and $\Delta\boldsymbol{\alpha}_k = \boldsymbol{\alpha} - \boldsymbol{\alpha}_{k-1}$ are the Gauss-Newton directions of $\boldsymbol{\alpha}_{k-1}$. Here Eq. 9 is a linear problem, and the Gauss-Newton directions are computed in each iteration:

$$
\begin{aligned}
\Delta\boldsymbol{\alpha}_k &= -(J_{k-1}^T J_{k-1} + \sigma I)^{-1} J_{k-1}^T \Delta P_{k-1} \\
&= -H(\boldsymbol{\alpha}_{k-1}, P)\Delta P_{k-1},
\end{aligned} \tag{10}
$$

where σ is a positive value to avoid ill-posed solutions and $H(\boldsymbol{\alpha}_k, P) := (J_{k-1}^T J_{k-1} + \sigma I)^{-1} J_{k-1}^T$. Given an initial guess $\boldsymbol{\alpha}_0$, the swing angle is iteratively updated as follows:

$$
\boldsymbol{\alpha}_k = \boldsymbol{\alpha}_{k-1} + \Delta\boldsymbol{\alpha}_k. \tag{11}
$$

Note that $\boldsymbol{\alpha}^{k-1}$ and $\Delta\boldsymbol{\alpha}^k$ are analytically differentiable. The whole process is summarized in Alg. 1.

In each step, e.g. the k-th step, the swing angle is iteratively updated as Eq. 11, while as $k \to \infty$, the local optimum of $\boldsymbol{\alpha}_*$ can be derived as follows:

$$
\boldsymbol{\alpha}_* = \lim_{k\to\infty} \boldsymbol{\alpha}_k = \lim_{k\to\infty} (\boldsymbol{\alpha}_{k-1} + \eta_k \Delta\boldsymbol{\alpha}_k). \tag{12}
$$

In practice, the Gauss-Newton-based solver is a second-order method, which just needs several iterations to converge. We have conducted several experiment to illustrate the convergence speed in the following section. To better understand the behavior of the optimization loss L_{opt}, we examined its gradient with respect to the one of the input parameters P, that is, $\frac{\partial \boldsymbol{\alpha}_k}{\partial P}$

$$
\frac{\partial \boldsymbol{\alpha}_k}{\partial P} = \frac{\partial \boldsymbol{\alpha}_{k-1}}{\partial P} + \frac{\partial \Delta\boldsymbol{\alpha}_k}{\partial P}. \tag{13}
$$

Since $\Delta\boldsymbol{\alpha}_k$ is derived from Eq. 10, its gradient can be calculated as:

$$
\begin{aligned}
\frac{\partial \Delta\boldsymbol{\alpha}_k}{\partial P} &= -\frac{\partial H(\boldsymbol{\alpha}_{k-1}, P)}{\partial P}\Delta P_{k-1} \\
&\quad - H(\boldsymbol{\alpha}_{k-1}, P)\frac{\partial \Delta P_{k-1}}{\partial P}.
\end{aligned} \tag{14}
$$

Note that the backward process in Alg. 1 is only used for illustration which is implemented with PyTorch's automatic differentiation engine.

Comparison to Implicit Differentiation Methods. Existing differentiation solvers are usually based on the implicit function theorem, which, however, is a quite time-consuming process and cannot handle the nonconvex optimization problem. The 3D-HPS problem in this paper involves highly nonconvex mapping from the 3D skeleton and body shapes to the relative body-part rotations. Our GN-Diff iteratively linearizes the objective function to obtain closed form Gauss-Newton directions $\Delta\boldsymbol{\alpha}_k$ and the automatic differentiation can be directly applied for end-to-end training.

Comparison to SPIN. Similar to our method, both the regression- and optimization-based modules are used in SPIN, where the optimization branch directly generates the supervision signal to train the regression one. Instead, IKOL treats the optimization problem Eq. 5 as an optimization layer that is differentiated by GN-Diff for end-to-end training. Benefiting from the fast convergence and close-form solution at each iteration of the Gauss-Newton method, GN-Diff is quite fast and effective compare to the optimization used in SPIN. Moreover, GN-Diff is not very sensitive to the initialization, which makes our training process more stable than SPIN.

Algorithm 1: The GN-Diff algorithm to differentiate IKOL

```
Input: $\boldsymbol{\alpha}_0$
Parameter:$T$, $P$, $R^{tw}$, $f(\cdot)$, $\sigma$
Output: $\boldsymbol{\alpha}_*$
Forward:
  for k = 1, 2, $\cdots$ do
      Compute $f(\boldsymbol{\alpha}_{k-1})$ and $J_{k-1}$
      Set $\Delta P_{k-1} = P - f(\boldsymbol{\alpha}_{k-1})$
      Compute $\Delta\boldsymbol{\alpha}_k$ using Eq. 10
      Update $\boldsymbol{\alpha}_k = \boldsymbol{\alpha}_{k-1} + \Delta\boldsymbol{\alpha}_k$
  end for
Backward:
  $\frac{\partial \boldsymbol{\alpha}^0}{\partial P} = -\frac{\partial H(\boldsymbol{\alpha}^0, P)}{\partial P}\Delta P^0 - H(\boldsymbol{\alpha}^0, P)\frac{\partial \Delta P^0}{\partial P}$
  for k = 1, 2, $\cdots$ do
      Compute $\frac{\partial \Delta\boldsymbol{\alpha}_k}{\partial P}$ using Eq. 14
      Update $\frac{\partial \boldsymbol{\alpha}_k}{\partial P} = \frac{\partial \boldsymbol{\alpha}_{k-1}}{\partial P} + \frac{\partial \Delta\boldsymbol{\alpha}_k}{\partial P}$
  end for
Returned values: $\boldsymbol{\alpha}_*$, $\frac{\partial \boldsymbol{\alpha}^*}{\partial P}$
```

Comparison to Hybrik. HybrIK also considered to reduce the error between the regressed joints P and the reconstructed joints Q. To achieve this, HybrIK relies on adaptively updating the newly reconstructed parent joints. However, HybrIK only reduces the error of children's joints in the kinematic tree if the parent joint is out of position, thus limiting the performance improvement. By contrast, our IKOL can significantly improve the 3D-HPS performance by iteratively optimizing all joints in the kinematic tree under an end-to-end differentiable framework.

Experiment

Implementation Details

For the purposes of experimentation, we implemented our framework as follows. Although the optimization is perform iteratively through the network, we observed that, with the Gauss-Newton solver, only several steps were needed for the model to converge. Hence, we empirically set the number of iteration steps to 5. It is worth noting that the objective function of our optimization only imposes a data fitting term; there is no regularization term. Therefore, unlike traditional optimization-based methods, our method is insensitive to the initialization parameter setting. We set the initialization parameter to zeros. Additionally, following (Li et al. 2021; Moon and Lee 2020), we used a ground truth bounding box in both the training and the testing stages. Mask R-CNN (He et al. 2017) was used to derive the bounding box if one was not available in the dataset. Following HybrIK as our baseline, we used ResNet-34 (He et al. 2016) pre-trained on the ImageNet dataset as our backbone network and adopted the same regression structure. All input images were padded to the same size of 256 × 256. We used Adam as our optimizer with an initial learning rate of 10^{-3}, reduced by a factor of

Method	Backbone	3DPW			MPI-INF-3DHP		
		PVE ↓	MPJPE ↓	PA-MPJPE ↓	PCK ↑	AUC ↑	MPJPE ↓
HMR (Kanazawa et al. 2018)	ResNet-50	-	130.0	81.3	72.9	36.5	124.2
Kolotouros et al. 2019	ResNet-50	-	-	70.2	-	-	-
Arnab et al. 2019	-	-	-	72.2	-	-	-
SPIN (Kolotouros et al. 2019)	ResNet-50	116.4	96.9	59.2	76.4	37.1	105.2
I2L-MeshNet (Moon and Lee 2020)	ResNet-50	110.1	93.2	58.6	-	-	-
HybrIK (Li et al. 2021)	ResNet-34	94.5	80.0	48.8	86.2	42.2	91.0
SPEC (Kocabas et al. 2021b)	ResNet-50	118.5	96.5	53.2	-	-	-
HMR-EFT (Joo et al. 2021)	ResNet-50	98.7	85.1	52.2	-	-	-
PARE (Kocabas et al. 2021a)	HRNet-w32	94.2	79.1	46.4	-	-	-
ROMP (Sun et al. 2021)	HRNet-w32	93.4	76.7	47.3	-	-	95.11
BEV (Sun et al. 2022b)	HRNet-w32	92.3	78.5	46.9	-	-	-
METRO (Lin et al. 2021a)	HRNet-w64	88.2	77.1	47.9	-	-	-
Mesh Graphormer (Lin et al. 2021b)	-	87.7	74.7	45.6	-	-	-
Ours(IKOL)	ResNet-34	86.4	73.3	45.5	**87.9**	**48.1**	**88.8**
Ours(IKOL*)	ResNet-34	**84.1**	**71.1**	**44.5**	87.0	47.6	89.7

Table 1: Performance comparison on 3DPW and MPI-INF-3DHP. ”*” means the results are fine tuned with the HybrIK model. ”-” means the results that are not available. Best in bold.

10 at the 50th and the 150th epoch. The model was trained for 400 epochs with a mini-batch size of 64 per GPU on four RTX 3090 GPUs. Notably, we empirically found HybrIK is a great method to pretrain our model and our performance can further boost if we use this method as our initial model weight for training. Accordingly, we conduct two experiment with and without pretrain from HybrIK.

Datasets and Evaluation Metrics

For a fair comparison with previous works, we trained our model on a mixture of 3D datasets (Human3.6M and MPI-INF-3DHP), and the 2D dataset COCO (Lin et al. 2014). We used the 3DPW training data when conducting experiments on this dataset. For evaluation Metrics, mean per joint position error(MPJPE) and procrustes aligned mean per joint position error (PA-MPJPE) are used to reflect the performance of 3D pose. Besides, per Vertex Error (PVE) are to evaluate the body mesh estimation ability. Additionly, we report the Percentage of Correct Keypoints (PCK) and Area Under Curve (AUC) on the MPI-INF-3DHP dataset. Our main result is shown in Table 1.

3DPW 3DPW is a challenging 3D-HPS benchmark that contains 60 video captured in indoor and outdoor conditions. We used this dataset as our major test set for evaluation.

MPI-INF-3DHP Following the setting of previous works, we report the MPJPE, the Percentage of Correct Keypoints (PCK) thresholded at 150mm and the Area Under the Curve (AUC) on MPI-INF-3DHP (Mehta et al. 2017a)

Human3.6M Dataset. The Human3.6M dataset (Ionescu et al. 2013) is the largest publicly available dataset for human 3D pose and shape estimation. Following (Li et al. 2021; Moon and Lee 2020), we used *Protocol* 2 and sampled every 5th frame in the videos for training.

	PVE ↓	MPJPE ↓	PA-MPJPE ↓
Analytical solution	94.5	80.0	48.8
Our	**92.5**	**78.2**	**48.5**

Table 2: Ablation study of the iterative optimization in our IKOL compared to the analytical method in HybrIK on 3DPW dataset. Best in bold.

Comparison with the State of the Art

To compare our method with state-of-the-art, we report our quantitative results on 3DPW and MPI-INF-3DHP dataset. For fair comparison, we use 14 joints for 3DPW and 17 joints for MPI-INF-3DHP dataset following the previous method. Tabel 1 shows that IKOL outperforms the existing regression and optimization methods in all sequence by a large margin. These results further demonstrate the effectiveness and efficiency of proposed IKOL model for 3D-HPS estimation.

Ablation Study

Analytical Method (HybrIK) VS Our Iterative Optimization To demonstrate the effectiveness of IKOL, we first compare the iterative optimization in our IKOL to the analytical method in HybrIK and take 3DPW dataset for an example to show the comparison. Since IKOL is completely compatible with the HybrIK module, we directly use the code from HybrIK and replace the analytical IK module in HybrIK with the iterative optimization in IKOL. It's worth noting that after the replacement we do not implement any fine tuning procedures to ensure a fair comparison. Tab. 2 shows that IKOL achieve better performance by using the same checkpoints. This means that our method can be directly applied to HybrIK to further improve the performance.

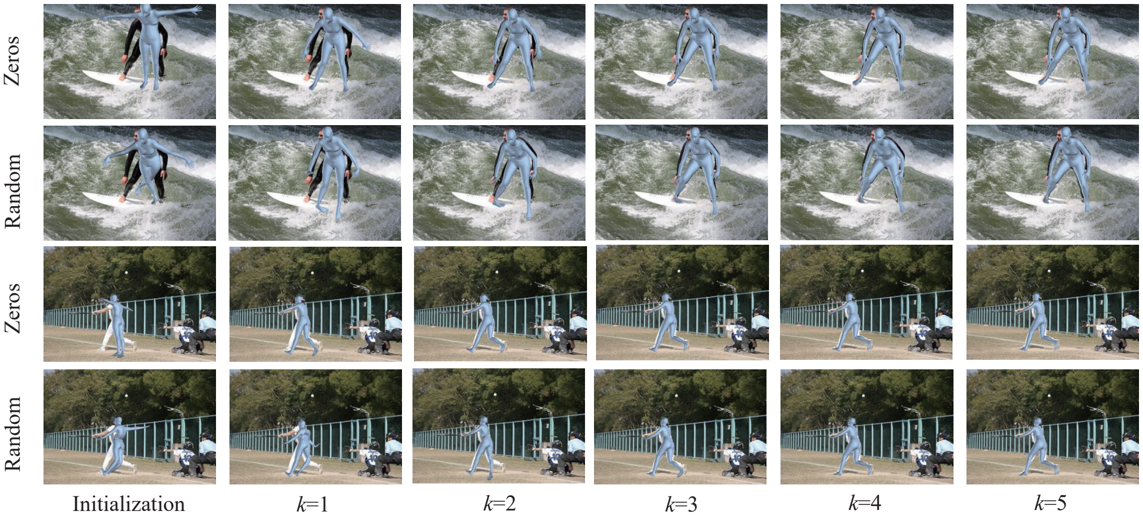

Figure 3: Visualization of reconstruction output results across each Gauss-Newton iteration k in the IKOL which show our method only take a few iterations to converge. Zero and random means use zero initialization and random initialization for fitting, which are used to verify the impact of the initialization

	PVE ↓	MPJPE ↓	PA-MPJPE ↓
Without GN-Diff	91.5	78.4	48.7
Full model	**86.4**	**73.3**	**45.5**

Table 3: Ablation study of the GN-Diff design on the 3DPW dataset by ablating the optimization loss during training. Best in bold.

	IKOL	SMPLify	HybrIK
Runtime (s)	0.182	74	0.04

Table 4: Runtime comparisons on a 3080 GPU compared with the analytical solver of HybrIK and optimization scheme, SMPLify, used in SPIN. We set 5 iterations in our method for comparison.

Analysis of the Effectiveness of GN-Diff. To further verify the effectiveness of GN-Diff to train our network with optimization. We conduct the experiment on 3DPW dataset with and without GN-Diff during training. Two experimental setting were started from scratch. Table 3 shows that GN-Diff is able to improve the pose and shape estimation by 5.1 MPJPE, 3.2 PA-MPJPE and 4.6 PVE improvement.

Analysis of Initialization To show the robustness of our method, we conduct show the visualization of reconstruction output results across each iteration k in the IKOL with different initialization as shown in Fig. 3. We use zero initialization and random initialization to verify that our method is insensitive to the initialization. From our empiricism, the whole training phase of IKOL is stable and we do not need any hand-craft thresholding to remove the bad fits as introduced in (Kolotouros et al. 2019). Besides, although IKOL is insensitive to the initialization, we also need to keep our initialization parameter within reasonable values due to the non-convex of the object function. In practice, the absolute value of the initialization parameter is recommended to be set below 1. In our experiment, we set zero initialization parameters as default.

Time Cost of the Optimization Branch To analyze the time cost of the optimization branch in IKOL, we show some visualization results at each iteration k in the IKOL. As shown in Fig. 3, we can see that IKOL only takes a few iterations to converge with different initialization. Further, we also compare the running time of our iterative optimization solver with the traditional optimization-based method SMPLify (Bogo et al. 2016) that is used in SPIN and analytical solution on a desktop computer with RTX 3080 GPU and Intel i7-10700 CPU. Table 4 shows that IKOL achieves superior performance, significantly faster than the competing optimization-based methods used in SPIN and even competitive with the analytical solution used in HybrIK without sacrificing too much computational cost. Note that other benchmark methods in Table 1 do not involve the optimization branch, thus we do not need to compare with them here.

Conclusion

In this paper, we presented IKOL, which leverages the strength of both optimization-based and regression-based methods within an end-to-end framework for 3D-HPS estimation. Unlike existing hybrid optimization and regression methods, IKOL involves an iterative optimization process within a DNN coupled with a regression branch. To ensure the framework offers end-to-end training, we designed GN-Diff to iteratively linearize the nonconvex objective function. As a result, the Gauss-Newton directions are derived in their closed form solutions, which means the differentiation procedure is both automatic and can be applied directly.

Acknowledgements

This work was supported by the Shanghai Sailing Program (21YF1429400, 22YF1428800), Shanghai Local College Capacity Building Program (23010503100,22010502800), NSFC programs (61976138, 61977047), the National Key Research and Development Program (2018YFB2100500), STCSM (2015F0203-000-06), SHMEC (2019-01-07-00-01-E00003) and Shanghai Frontiers Science Center of Human-centered Artificial Intelligence (ShangHAI).

References

Agrawal, A.; Amos, B.; Barratt, S.; Boyd, S.; Diamond, S.; and Kolter, J. Z. 2019. Differentiable convex optimization layers. *NIPS*, 32.

Amos, B.; and Kolter, J. Z. 2017. Optnet: Differentiable optimization as a layer in neural networks. In *ICML*, 136–145. PMLR.

Baerlocher, P.; and Boulic, R. 2001. Parametrization and range of motion of the ball-and-socket joint. In *Deformable avatars*, 180–190. Springer.

Bogo, F.; Kanazawa, A.; Lassner, C.; Gehler, P.; Romero, J.; and Black, M. J. 2016. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *ECCV*, 561–578. Springer.

Fan, T.; Alwala, K. V.; Xiang, D.; Xu, W.; Murphey, T.; and Mukadam, M. 2021. Revitalizing optimization for 3D human pose and shape estimation: a sparse constrained formulation. In *CVPR*, 11457–11466.

Gould, S.; Hartley, R.; and Campbell, D. J. 2021. Deep declarative networks. *PAMI*.

Guler, R. A.; and Kokkinos, I. 2019. Holopose: Holistic 3d human reconstruction in-the-wild. In *CVPR*, 10884–10894.

Han, X.; Cong, P.; Xu, L.; Wang, J.; Yu, J.; and Ma, Y. 2023. LiCamGait: Gait Recognition in the Wild by Using LiDAR and Camera Multi-modal Visual Sensors. *AAAI*.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *ICCV*, 2961–2969.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. *CVPR*.

Huang, Y.; Bogo, F.; Lassner, C.; Kanazawa, A.; Gehler, P. V.; Romero, J.; Akhter, I.; and Black, M. J. 2017. Towards accurate marker-less human shape and pose estimation over time. In *3DV*, 421–430. IEEE.

Ionescu, C.; Papava, D.; Olaru, V.; and Sminchisescu, C. 2013. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *PAMI*, 36(7): 1325–1339.

Jiang, W.; Kolotouros, N.; Pavlakos, G.; Zhou, X.; and Daniilidis, K. 2020. Coherent reconstruction of multiple humans from a single image. In *CVPR*, 5579–5588.

Kanazawa, A.; Black, M. J.; Jacobs, D. W.; and Malik, J. 2018. End-to-end recovery of human shape and pose. In *CVPR*, 7122–7131.

Kocabas, M.; Huang, C.-H. P.; Hilliges, O.; and Black, M. J. 2021a. PARE: Part attention regressor for 3D human body estimation. In *ICCV*, 11127–11137.

Kocabas, M.; Huang, C.-H. P.; Tesch, J.; Müller, L.; Hilliges, O.; and Black, M. J. 2021b. SPEC: Seeing people in the wild with an estimated camera. In *ICCV*, 11035–11045.

Kolotouros, N.; Pavlakos, G.; Black, M. J.; and Daniilidis, K. 2019. Learning to reconstruct 3D human pose and shape via model-fitting in the loop. In *ICCV*, 2252–2261.

Lassner, C.; Romero, J.; Kiefel, M.; Bogo, F.; Black, M. J.; and Gehler, P. V. 2017. Unite the people: Closing the loop between 3d and 2d human representations. In *CVPR*, 6050–6059.

Li, J.; Xu, C.; Chen, Z.; Bian, S.; Yang, L.; and Lu, C. 2021. Hybrik: A hybrid analytical-neural inverse kinematics solution for 3d human pose and shape estimation. In *CVPR*, 3383–3393.

Liang, H.; He, Y.; Zhao, C.; Li, M.; Wang, J.; Yu, J.; and Xu, L. 2023. HybridCap: Inertia-aid Monocular Capture of Challenging Human Motions. *AAAI*.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *ECCV*, 740–755. Springer.

Loper, M.; Mahmood, N.; Romero, J.; Pons-Moll, G.; and Black, M. J. 2015. SMPL: A skinned multi-person linear model. *TOG*, 34(6): 1–16.

Mehta, D.; Rhodin, H.; Casas, D.; Fua, P.; Sotnychenko, O.; Xu, W.; and Theobalt, C. 2017a. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *3DV*, 506–516. IEEE.

Mehta, D.; Sotnychenko, O.; Mueller, F.; Xu, W.; Elgharib, M.; Fua, P.; Seidel, H.-P.; Rhodin, H.; Pons-Moll, G.; and Theobalt, C. 2020. XNect: Real-time multi-person 3D motion capture with a single RGB camera. *TOG*, 39(4): 82–1.

Mehta, D.; Sridhar, S.; Sotnychenko, O.; Rhodin, H.; Shafiei, M.; Seidel, H.-P.; Xu, W.; Casas, D.; and Theobalt, C. 2017b. Vnect: Real-time 3d human pose estimation with a single rgb camera. *TOG*, 36(4): 1–14.

Moon, G.; and Lee, K. M. 2020. I2l-meshnet: Image-to-lixel prediction network for accurate 3d human pose and mesh estimation from a single rgb image. In *ECCV*, 752–768. Springer.

Omran, M.; Lassner, C.; Pons-Moll, G.; Gehler, P.; and Schiele, B. 2018. Neural body fitting: Unifying deep learning and model based human pose and shape estimation. In *3DV*, 484–494. IEEE.

Pavlakos, G.; Choutas, V.; Ghorbani, N.; Bolkart, T.; Osman, A. A.; Tzionas, D.; and Black, M. J. 2019. Expressive body capture: 3d hands, face, and body from a single image. In *CVPR*, 10975–10985.

Pavlakos, G.; Zhu, L.; Zhou, X.; and Daniilidis, K. 2018. Learning to estimate 3D human pose and shape from a single color image. In *CVPR*, 459–468.

Rolínek, M.; Musil, V.; Paulus, A.; Vlastelica, M.; Michaelis, C.; and Martius, G. 2020a. Optimizing rank-based metrics with blackbox differentiation. In *CVPR*, 7620–7630.

Rolínek, M.; Swoboda, P.; Zietlow, D.; Paulus, A.; Musil, V.; and Martius, G. 2020b. Deep graph matching via blackbox differentiation of combinatorial solvers. In *ECCV*, 407–424. Springer.

Rueegg, N.; Lassner, C.; Black, M.; and Schindler, K. 2020. Chained representation cycling: Learning to estimate 3D human pose and shape by cycling between representations. In *AAAI*, volume 34, 5561–5569.

Sun, H.; Shi, Y.; Wang, J.; Tuan, H. D.; Poor, H. V.; and Tao, D. 2022a. Alternating Differentiation for Optimization Layers. *arXiv preprint arXiv:2210.01802*.

Sun, Y.; Bao, Q.; Liu, W.; Fu, Y.; Black, M. J.; and Mei, T. 2021. Monocular, one-stage, regression of multiple 3d people. In *CVPR*, 11179–11188.

Sun, Y.; Liu, W.; Bao, Q.; Fu, Y.; Mei, T.; and Black, M. J. 2022b. Putting people in their place: Monocular regression of 3d people in depth. In *CVPR*, 13243–13252.

Varol, G.; Ceylan, D.; Russell, B.; Yang, J.; Yumer, E.; Laptev, I.; and Schmid, C. 2018. Bodynet: Volumetric inference of 3d human body shapes. In *ECCV*, 20–36.

Xiang, D.; Joo, H.; and Sheikh, Y. 2019. Monocular total capture: Posing face, body, and hands in the wild. In *CVPR*, 10965–10974.

Xu, Y.; Zhu, S.-C.; and Tung, T. 2019. Denserac: Joint 3d pose and shape estimation by dense render-and-compare. In *CVPR*, 7760–7770.

Yu, Z.; Wang, J.; Xu, J.; Ni, B.; Zhao, C.; Wang, M.; and Zhang, W. 2021. Skeleton2Mesh: Kinematics Prior Injected Unsupervised Human Mesh Recovery. In *ICCV*, 8619–8629.

Zanfir, A.; Bazavan, E. G.; Xu, H.; Freeman, W. T.; Sukthankar, R.; and Sminchisescu, C. 2020. Weakly supervised 3d human pose and shape reconstruction with normalizing flows. In *ECCV*, 465–481. Springer.

Zanfir, A.; Bazavan, E. G.; Zanfir, M.; Freeman, W. T.; Sukthankar, R.; and Sminchisescu, C. 2021. Neural descent for visual 3d human pose and shape. In *CVPR*, 14484–14493.

Zanfir, A.; Marinoiu, E.; and Sminchisescu, C. 2018. Monocular 3d pose and shape estimation of multiple people in natural scenes-the importance of multiple scene constraints. In *CVPR*, 2148–2157.

Zhang, H.; Cao, J.; Lu, G.; Ouyang, W.; and Sun, Z. 2020. Learning 3d human shape and pose from dense body parts. *PAMI*.

Zhang, J.; Wang, J.; Shi, Y.; Gao, F.; Xu, L.; and Yu, J. 2022. Mutual Adaptive Reasoning for Monocular 3D Multi-Person Pose Estimation. *ACMMM*.

Zhang, J.; Yu, D.; Liew, J. H.; Nie, X.; and Feng, J. 2021. Body meshes as points. In *CVPR*, 546–556.