Revitalizing Optimization for 3D Human Pose and Shape Estimation: A Sparse Constrained Formulation

Taosha Fan[1,2], Kalyan Vasudev Alwala[1], Donglai Xiang[3,4], Weipeng Xu[3],
Todd Murphey[2], Mustafa Mukadam[1]

[1]Facebook AI Research, [2]Northwestern University,
[3]Facebook Reality Labs, [4]Carnegie Mellon University

Abstract

We propose a novel sparse constrained formulation and from it derive a real-time optimization method for 3D human pose and shape estimation. Our optimization method, SCOPE (Sparse Constrained Optimization for 3D human Pose and shapE estimation), is orders of magnitude faster (avg. 4ms convergence) than existing optimization methods, while being mathematically equivalent to their dense unconstrained formulation under mild assumptions. We achieve this by exploiting the underlying sparsity and constraints of our formulation to efficiently compute the Gauss-Newton direction. We show that this computation scales linearly with the number of joints and measurements of a complex 3D human model, in contrast to prior work where it scales cubically due to their dense unconstrained formulation. Based on our optimization method, we present a real-time motion capture framework that estimates 3D human poses and shapes from a single image at over 30 FPS. In benchmarks against state-of-the-art methods on multiple public datasets, our framework outperforms other optimization methods and achieves competitive accuracy against regression methods. Project page with code and videos: https://sites.google.com/view/scope-human/.

1. Introduction

Estimating 3D human poses and shapes from an image has a broad range of applications in embodied AI, robotics, AR/VR, and has seen remarkable progress in recent years. Among leading techniques, optimization methods [5, 17, 19, 24, 25, 39] have been successful. However, they can still take up to tens of seconds to fit 3D human poses and shapes given an image, which is not ideal for real-time applications. Deep learning based regression methods [12, 15] have significantly reduced the computation times down to just tens of milliseconds, but often rely on optimization during training or for refining the network outputs. With a novel formulation, we revitalize optimization towards solving this problem in real-time.

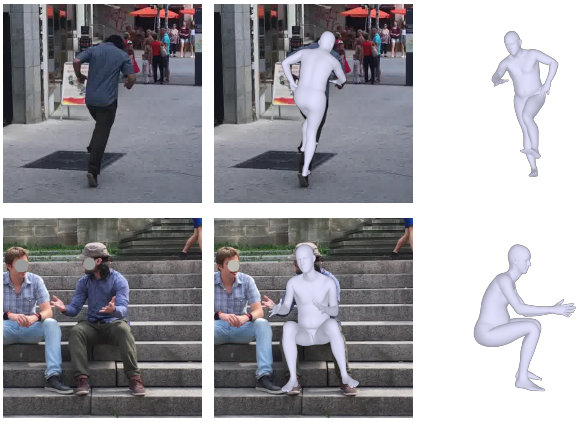


Figure 1: Example solutions from our motion capture framework based on our proposed sparse constrained optimization. (left) input image from the 3DPW [37] dataset, (middle) 3D pose and shape reconstruction overlayed on the input image, (right) 3D reconstruction shown from a rotated viewpoint.

Most optimization methods [5, 17, 19, 24, 25, 39] formulate 3D human pose and shape estimation as dense unconstrained optimization problems, differing only in terms of the objective functions. These formulations are dense as they result in dense Hessian matrices and unconstrained as the optimization variables are unconstrained. To optimize the objective they use iterative techniques like Gauss-Newton [26] to find a local minimum given an initial guess. These formulations however, suffer from high computation times due to the dense Hessian matrices that lead to $O(K^3) + O(K^2N)$ time to compute the Gauss-Newton direction for a 3D human model with K joints and N measurements. In particular, computing this direction involves the steps of linearization to find the Jacobian, building and then solving the linear system, where a dense formulation renders all these steps expensive. Therefore, it is critical to improve the efficiency of the Gauss-Newton direction computation to develop real-time optimization methods for 3D human pose and shape estimation.

In this work, instead of using the dense unconstrained formulation from existing optimization methods, we present a sparse constrained formulation that is mathematically equivalent under mild assumptions. We show how the underlying sparsity and constraints of our formulation can be exploited leading to sparse Hessian matrices and ultimately computing the Gauss-Newton direction in $O(K) + O(N)$ time for a 3D human model with K joints and N measurements. Our optimization method, *SCOPE (Sparse Constrained Optimization for 3D human Pose and shapE estimation)*, is thus orders of magnitude faster (average 4 ms convergence) than existing optimization methods, particularly when the number of joints K and measurements N is large.

Based on our optimization method, we present a real-time 3D motion capture framework (illustrated in Figure 2) that estimates 3D human poses and shapes from a single image at over 30 FPS. Example solutions are shown in Figure 1. Our method allows using a modified SMPL model [20] that has 75 degrees of freedom and 10 shape parameters, and estimates both human poses and shapes with which the 3D human mesh can be fully reconstructed. In contrast, several real-time 3D motion capture frameworks using optimization methods [24, 25] adopt a much simpler 3D skeleton model with 33 degrees of freedom and no shape parameters to reduce the computation complexity and are therefore unable to reconstruct the 3D human mesh. We compare our real-time 3D motion capture framework with numerous state-of-the-art methods [5, 12, 15, 16, 17, 39] on public benchmark datasets [10, 23, 37]. Our framework achieves accuracies that outperform optimization methods [5, 17, 25, 39] and are competitive to regression methods [12, 15].

In summary, our contributions are: (i) we propose a sparse constrained formulation for 3D human pose and shape estimation that is mathematically equivalent to the dense unconstrained formulation of existing optimization methods under mild assumptions; (ii) we develop an efficient algorithm that computes the Gauss-Newton direction in linear-time complexity with respect to the number of joints and measurements; and (iii) we present a real-time 3D motion capture framework that estimates 3D human poses and shapes from a single image.

2. Related work

Optimization methods estimate human poses and shapes by matching 3D joints on the human body to 2D keypoints on the image. Works in human body modeling [2, 20, 27] and 2D keypoint detection [6, 8, 36] have made substantial contributions, but the resulting optimization problem remains challenging due to the ambiguity in the 3D information from an image and the uncertainty of 3D human poses. To address this, recent works have incorporated 3D information, such as 3D keypoint positions [24, 25], part orientation fields [39], silhouette [9], etc, as additional fitting terms. Additionally, human 3D pose priors in the form of mixture of Gaussians [5], variational autoencoder [28], and normalizing flow [40] have been trained from numerous datasets [10,11,21] and successfully applied to human 3D pose and shape estimation. A closer look at these optimization methods [5,17,24,25,39,40] does reveal that they primarily differ in their loss terms of the objective function while still utilizing the same underlying dense unconstrained formulation. We show that such a formulation is inherently inefficient in computing the Gauss-Newton direction. Thus despite the considerable progress, these methods still take tens of seconds to converge and are impractical for real time applications.

Regression methods use deep neural networks to regress human poses and shapes directly from images. In most cases, regression methods [12, 15, 16, 34] take only tens of milliseconds to process one image and meet the real-time requirements. Unlike [22, 29, 30, 31, 32] that lift 2D keypoints to 3D keypoints, regression methods for 3D human pose and shape estimation face a challenge in having access to large datasets with ground truth labels of 3D human pose and shape. To address this, regressions methods often employ optimization methods to precompute 3D ground truth for supervision [12] or even have optimization methods in the loop [15] during training. Other examples like [34] rely on optimization methods to refine the network outputs. In these aforementioned scenarios, the computational efficiency of optimization methods play an important role both during training and deployment.

3. Problem Formulation

3.1. SMPL Model

The SMPL model [20] is a vertex-based linear blend skinning 3D human model. In this paper, we use a SMPL model that has $K = 23$ rotational joints, $N = 6890$ vertices, and $P = 10$ shape parameters.

The SMPL model represents the human body using a kinematic tree with $K + 1$ inter-connected body parts indexed with $i = 0, 1, \cdots, K$. In the rest of this paper, we use $\mathrm{par}(i)$ to denote the parent of body part i, and $\mathbf{T}_i \in SE(3)$ the pose of body part i, and $\boldsymbol{\Omega}_i \in SO(3)$ the state of joint i, and $\boldsymbol{\beta} \in \mathbb{R}^P$ the shape parameters. Note that body part i is connected to its parent body part $\mathrm{par}(i)$ through joint i.

In the Supplementary Material, we show that it is possible to extract $\mathcal{S}_i \in \mathbb{R}^{3\times P}$ and $\mathbf{l}_i \in \mathbb{R}^3$ from the SMPL model such that the relative pose $\mathbf{T}_{\mathrm{par}(i),i} \in SE(3)$ between body part i and its parent body part $\mathrm{par}(i)$ is

$$\mathbf{T}_{\mathrm{par}(i),i} \triangleq \begin{bmatrix} \boldsymbol{\Omega}_i & \mathcal{S}_i \cdot \boldsymbol{\beta} + \mathbf{l}_i \\ \mathbf{0} & 1 \end{bmatrix}. \tag{1}$$

Furthermore, if $\mathbf{T}_i \in SE(3)$ of body part i is represented as $\mathbf{T}_i \triangleq \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3)$ in which $\mathbf{R}_i \in SO(3)$ is the rotation and $\mathbf{t}_i \in \mathbb{R}^3$ is the translation, then $\mathbf{T}_i$ can be recursively computed as

$$\mathbf{T}_i = \mathbf{T}_{\mathrm{par}(i)}\mathbf{T}_{\mathrm{par}(i),i} = \mathbf{T}_{\mathrm{par}(i)} \begin{bmatrix} \boldsymbol{\Omega}_i & \mathcal{S}_i \cdot \boldsymbol{\beta} + \mathbf{l}_i \\ \mathbf{0} & 1 \end{bmatrix}. \quad (2)$$

3.2. Rigid Skinning Assumption of Keypoints

We need to select a set of joints and vertices on the SMPL model as keypoints to calculate 2D and 3D keypoint losses, part orientation field losses, etc. [5,24,25,39]. In this paper, we modify the SMPL model and make the following assumption of the selected keypoints for loss calculation.

Assumption 1. Each keypoint j is rigidly attached to a body part i, i.e., the position $\mathbf{v}_j \in \mathbb{R}^3$ of keypoint j is

$$\mathbf{v}_j = \mathbf{R}_i \overline{\mathbf{v}}_j + \mathbf{t}_i, \quad (3)$$

in which $\mathbf{R}_i \in SO(3)$ and $\mathbf{t}_i \in \mathbb{R}^3$ are the rotation and translation of pose $\mathbf{T}_i \in SE(3)$, and $\overline{\mathbf{v}}_j \in \mathbb{R}^3$ is the relative position of keypoint j with respect to body part i. Furthermore, there exists $\mathcal{V}_j \in \mathbb{R}^{3\times P}$ and $\overline{\mathbf{v}}_{j,0} \in \mathbb{R}^3$ such that the relative position $\overline{\mathbf{v}}_j \in \mathbb{R}^3$ in Eq. (3) is evaluated as

$$\overline{\mathbf{v}}_j = \mathcal{V}_j \cdot \boldsymbol{\beta} + \overline{\mathbf{v}}_{j,0}. \quad (4)$$

For simplicity, we use $\mathcal{V}_j$ and $\overline{\mathbf{v}}_{j,0}$ extracted from the joint and vertex positions at the rest pose of the SMPL model, whose derivation is similar to that of $\mathcal{S}_i$ and $\mathbf{l}_i$ in Eq. (1). We remark that Assumption 1 is important for our sparse constrained formulation presented later in this paper.

Compared to the SMPL model, Assumption 1 keeps rigid skinning (shape blend shapes) while dropping non-rigid skinning (pose blend shapes) for the vertex keypoints. We argue that Assumption 1 is a reasonable and mild modification for human pose and shape estimation. First, the SMPL model evaluates the joint keypoints, such as wrists, elbows, knees, etc, using Eq. (2), which is essentially equivalent to Eqs. (3) and (4) of rigid skinning. While the SMPL model has each vertex position depend on the poses of all the body parts, the vertices selected as keypoints, such as nose, eyes, ears, etc., are mainly affected by a single body part. Finally, we note that inaccuracies are also present in 2D and 3D keypoint measurements used for estimation, which are usually much larger than those induced by the SMPL model modification using Eqs. (3) and (4).

3.3. Objective Function

Given an RGB image, we use the following objective for human pose and shape estimation:

$$\mathtt{E} = \sum_{0\le i\le K} \big(\mathtt{E}_{\mathrm{2D},i} + \lambda_{\mathrm{3D}} \cdot \mathtt{E}_{\mathrm{3D},i} + \lambda_{\mathrm{p}} \cdot \mathtt{E}_{\mathrm{p},i} + \lambda_{\mathrm{T}} \cdot \mathtt{E}_{\mathrm{T},i} + \lambda_{\Omega} \cdot \mathtt{E}_{\Omega,i}\big) + \lambda_{\beta} \cdot \mathtt{E}_{\beta}, \quad (5)$$

in which λ_{3D}, λ_{p}, λ_{T}, λ_{Ω} and λ_{β} are scalar weights and joint state $\boldsymbol{\Omega}_0 \in SO(3)$ for body part 0 is a dummy variable. Each loss term in Eq. (5) is defined as follows:

1. $\mathtt{E}_{\mathrm{2D},i} \triangleq \frac{1}{2}\sum_{j\in \mathrm{V}_{\mathrm{2D},i}} \|\Pi_{\mathbf{K}}(\mathbf{v}_j) - \hat{\mathbf{v}}_{\mathrm{2D},j}\|^2$ is the 2D keypoint loss, where $\mathrm{V}_{\mathrm{2D},i}$ is the set of indices of keypoints attached to body part i and selected to calculate the 2D keypoint loss, $\Pi_{\mathbf{K}}(\cdot)$ is the 3D to 2D projection map with camera intrinsics $\mathbf{K}$, $\mathbf{v}_j \in \mathbb{R}^3$ is the keypoint position, and $\hat{\mathbf{v}}_{\mathrm{2D},j} \in \mathbb{R}^2$ is the 2D keypoint measurement.
2. $\mathtt{E}_{\mathrm{3D},i} \triangleq \frac{1}{2}\sum_{j\in \mathrm{V}_{\mathrm{3D},i}} \|\mathbf{v}_j - \hat{\mathbf{v}}_{\mathrm{3D},j}\|^2$ is the 3D keypoint loss, where $\mathrm{V}_{\mathrm{3D},i}$ is the set of indices of keypoints attached to body part i and selected to calculate the 3D keypoint loss, $\mathbf{v}_j \in \mathbb{R}^3$ is the keypoint position and $\hat{\mathbf{v}}_{\mathrm{3D},j} \in \mathbb{R}^3$ is the 3D keypoint measurement.
3. $\mathtt{E}_{\mathrm{p},i} \triangleq \frac{1}{2}\sum_{j\in \mathrm{P}_i} \left\| \frac{\mathbf{v}_j - \mathbf{t}_i}{\|\mathbf{v}_j - \mathbf{t}_i\|} - \hat{\mathbf{p}}_j \right\|^2$ is the part orientation field loss [39], where P_i is the set of indices of keypoints attached to body part i and selected to calculate the part orientation field loss, $\mathbf{v}_j \in \mathbb{R}^3$ is the keypoint position, and $\mathbf{t}_i \in \mathbb{R}^3$ is the position of body part i as well as the translation of pose $\mathbf{T}_i \in SE(3)$, and $\hat{\mathbf{p}}_i \in \mathbb{R}^3$ is the part orientation field measurement.
4. $\mathtt{E}_{\mathrm{T},i} \triangleq \frac{1}{2}\|\mathbf{T}_i - \hat{\mathbf{T}}_i\|^2$ is the prior loss of pose $\mathbf{T}_i \in SE(3)$, where $\hat{\mathbf{T}}_i \in SE(3)$ is a known prior estimate.
5. $\mathtt{E}_{\Omega,i} \triangleq \frac{1}{2}\|\mathtt{r}_{\Omega_i}(\boldsymbol{\Omega}_i)\|^2$ is the prior loss of joint state $\boldsymbol{\Omega}_i \in SO(3)$, where $\mathtt{r}_{\Omega_i}(\cdot)$ is a normalizing flow of $SO(3)$ trained on the AMASS dataset [21]. Please see the Supplementary Material for more details on $\mathtt{E}_{\Omega,i}$.
6. $\mathtt{E}_{\beta} \triangleq \frac{1}{2}\|\boldsymbol{\beta}\|^2$ is the prior loss of shape parameters $\boldsymbol{\beta} \in \mathbb{R}^P$.

From the definitions above, each loss term $\mathtt{E}_{(\#),i}$ in Eq. (5) can be in general formulated as

$$\mathtt{E}_{(\#),i} = \sum_j \frac{1}{2}\|\mathtt{r}_{(\#),ij}(\mathbf{T}_i, \boldsymbol{\Omega}_i, \boldsymbol{\beta}, \mathbf{v}_j)\|^2, \quad (6)$$

in which $\mathtt{r}_{(\#),ij}(\cdot)$ is a function of $\mathbf{T}_i$, $\boldsymbol{\Omega}_i$, $\boldsymbol{\beta}$ and $\mathbf{v}_j$. Since keypoint j in Eq. (6) is attached to body part i, then Eqs. (3) and (4) indicate that $\mathbf{v}_j$ is a function of $\mathbf{T}_i$ and $\boldsymbol{\beta}$:

$$\mathbf{v}_j = \mathbf{R}_i\big(\mathcal{V}_j \cdot \boldsymbol{\beta} + \overline{\mathbf{v}}_{j,0}\big) + \mathbf{t}_i. \quad (7)$$

As a result of Eq. (7), we might cancel out $\mathbf{v}_j$ in Eq. (6) and simplify $\mathtt{r}_{(\#),ij}(\cdot)$ as a function of $\mathbf{T}_i$, $\boldsymbol{\Omega}_i$ and $\boldsymbol{\beta}$:

$$\mathtt{E}_{(\#),i} = \sum_j \frac{1}{2}\|\mathtt{r}_{(\#),ij}(\mathbf{T}_i, \boldsymbol{\Omega}_i, \boldsymbol{\beta})\|^2. \quad (8)$$

We remark that $\mathtt{r}_{(\#),ij}(\cdot)$ in Eq. (8) is related to $\mathbf{T}_i \in SE(3)$ and $\boldsymbol{\Omega}_i \in SO(3)$ of a single body part i. Then, Eq. (8) immediately suggests that Eq. (5) takes the form of

$$\mathtt{E} = \sum_{0\le i\le K} \frac{1}{2}\|\mathtt{r}_i(\mathbf{T}_i, \boldsymbol{\Omega}_i, \boldsymbol{\beta})\|^2, \quad (9)$$

in which each $\mathtt{r}_i(\cdot)$ is a function of $\mathbf{T}_i \in SE(3)$, $\boldsymbol{\Omega}_i \in SO(3)$ and $\boldsymbol{\beta} \in \mathbb{R}^P$. Besides those in Eq. (5), a number of losses can be written in the form of Eqs. (6) and (8) as well.

3.4. Dense Unconstrained Optimization

With Eqs. (1) and (2), we might recursively compute each $\mathbf{T}_i \in SE(3)$ through a top-down traversal of the kinematics tree. Thus, each $\mathbf{T}_i$ can be written as a function of the root pose $\mathbf{T}_0 \in SE(3)$, the joint states $\boldsymbol{\Omega} \triangleq (\boldsymbol{\Omega}_0, \boldsymbol{\Omega}_1, \cdots, \boldsymbol{\Omega}_K) \in SO(3)^{K+1}$ and the shape parameters $\boldsymbol{\beta} \in \mathbb{R}^P$:

$$\mathbf{T}_i \triangleq \mathbf{T}_i\left(\mathbf{T}_0, \boldsymbol{\Omega}, \boldsymbol{\beta}\right). \tag{10}$$

In existing optimization methods [5, 17, 24, 25, 28, 39], Eq. (10) is substituted into Eq. (9) to cancel out non-root poses $\mathbf{T}_i \in SE(3)$ $(1 \leq i \leq K)$, which results in a dense unconstrained optimization problem of $\mathbf{T}_0 \in SE(3)$, $\boldsymbol{\Omega} \in SO(3)^K$ and $\boldsymbol{\beta} \in \mathbb{R}^P$:

$$\min_{\mathbf{T}_0, \boldsymbol{\Omega}, \boldsymbol{\beta}} \mathtt{E} = \sum_{0 \leq i \leq K} \frac{1}{2} \|\mathbf{r}_i(\mathbf{T}_0, \boldsymbol{\Omega}, \boldsymbol{\beta})\|^2. \tag{11}$$

In general, Gauss-Newton is the preferred method to solve optimization problems of the kind in Eq. (11). This consists of linearization to find the Jacobian matrix, building and then solving the linear system to find the Gauss-Newton direction. In the Supplementary Material we show that Eq. (11) yields a dense linear system when computing the Gauss-Newton direction. Since the complexity of dense linear system computation increases superlinearly with their size, the dense unconstrained formulation of Eq. (11) has poor scalability when the human model has large numbers of joints and measurements.

4. Method

In this section, we present a sparse constrained formulation for 3D human pose and shape estimation that is mathematically equivalent to the dense unconstrained one in Section 3.4. From our formulation, we derive a method that scales linearly with the number of joints and measurements to compute the Gauss-Newton direction.

4.1. Sparse Constrained Optimization

We introduce $\boldsymbol{\beta}_i \in \mathbb{R}^P$ with $\boldsymbol{\beta}_i = \boldsymbol{\beta}_{\mathrm{par}(i)}$ for each body part i in the SMPL model. Since $\boldsymbol{\beta}_i = \boldsymbol{\beta}_{\mathrm{par}(i)}$ indicates $\boldsymbol{\beta}_i = \boldsymbol{\beta}$, and $\mathbf{T}_i$, $\boldsymbol{\Omega}_i$ and $\boldsymbol{\beta}$ satisfy the kinematic constraints of Eq. (2), we formulate 3D human pose and shape estimation of Eq. (9) as a sparse constrained optimization problem on $\{\mathbf{T}_i, \boldsymbol{\beta}_i, \boldsymbol{\Omega}_i\}_{i=0}^K \in \left(SE(3) \times \mathbb{R}^P \times SO(3)\right)^{K+1}$:

$$\min_{\{\mathbf{T}_i, \boldsymbol{\beta}_i, \boldsymbol{\Omega}_i\}_{i=0}^K} \sum_{0 \leq i \leq K} \frac{1}{2} \|\mathbf{r}_i(\mathbf{T}_i, \boldsymbol{\Omega}_i, \boldsymbol{\beta}_i)\|^2 \tag{12}$$

subject to

$$\begin{aligned} \mathbf{T}_i &= \mathtt{F}_i(\mathbf{T}_{\mathrm{par}(i)}, \boldsymbol{\beta}_{\mathrm{par}(i)}, \boldsymbol{\Omega}_i) \\ &\triangleq \mathbf{T}_{\mathrm{par}(i)} \begin{bmatrix} \boldsymbol{\Omega}_i & \mathcal{S}_i \cdot \boldsymbol{\beta}_{\mathrm{par}(i)} + \mathbf{l}_i \\ \mathbf{0} & 1 \end{bmatrix}, \end{aligned} \tag{13a}$$

$$\boldsymbol{\beta}_i = \boldsymbol{\beta}_{\mathrm{par}(i)}. \tag{13b}$$

In Eq. (13a), note that $\mathtt{F}_i(\cdot) : SE(3) \times \mathbb{R}^P \times SO(3) \rightarrow SE(3)$ is a function corresponding to Eq. (2) and maps $\mathbf{T}_{\mathrm{par}(i)}$, $\boldsymbol{\beta}_{\mathrm{par}(i)}$, $\boldsymbol{\Omega}_i$ to $\mathbf{T}_i$. For notational simplicity, we define $\mathbf{x}_i \triangleq (\mathbf{T}_i, \boldsymbol{\beta}_i) \in SE(3) \times \mathbb{R}^P$. Then, Eqs. (12) and (13) are equivalent to

$$\min_{\{\mathbf{x}_i, \boldsymbol{\Omega}_i\}_{i=0}^K} \sum_{0 \leq i \leq K} \frac{1}{2} \|\mathbf{r}_i(\mathbf{x}_i, \boldsymbol{\Omega}_i)\|^2 \tag{14}$$

subject to

$$\mathbf{x}_i = \begin{bmatrix} \mathtt{F}_i(\mathbf{x}_{\mathrm{par}(i)}, \boldsymbol{\Omega}_i) \\ \boldsymbol{\beta}_{\mathrm{par}(i)} \end{bmatrix}. \tag{15}$$

In spite of additional optimization variables and kinematic constraints compared to Eq. (11), we have the following proposition for our sparse constrained formulation.

Proposition 1. Eqs. (14) and (15) are equivalent to Eq. (11) (under Assumption 1).

Proof. Please refer to the Supplementary Material. □

In the remainder of this section, we will make use of the sparsity and constraints of Eqs. (14) and (15) to simplify the computation of the Gauss-Newton direction.

4.2. Gauss-Newton Direction

The computation of the Gauss-Newton direction for Eqs. (14) and (15) is summarized as follows.

Step 1: The linearization of Eqs. (14) and (15) results in

$$\min_{\{\Delta\mathbf{x}_i, \Delta\boldsymbol{\Omega}_i\}_{i=0}^K} \Delta\mathtt{E} = \sum_{0 \leq i \leq K} \frac{1}{2} \left\|\mathbf{J}_{i,1}\Delta\mathbf{x}_i + \mathbf{J}_{i,2}\Delta\boldsymbol{\Omega}_i + \mathbf{r}_i\right\|^2, \tag{16}$$

subject to

$$\Delta\mathbf{x}_i = \mathbf{A}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{B}_i, \tag{17}$$

in which $\Delta\mathbf{x}_i \triangleq (\Delta\mathbf{T}_i, \Delta\boldsymbol{\beta}_i) \in \mathbb{R}^{6+P}$ and $\Delta\boldsymbol{\Omega}_i \in \mathbb{R}^3$ are the Gauss-Newton directions of $\mathbf{x}_i$ and $\boldsymbol{\Omega}_i$, respectively, and $\mathbf{r}_i$ in Eq. (16) is the residue, and

$$\mathbf{J}_{i,1} \triangleq \frac{\partial \mathbf{r}_i}{\partial \mathbf{x}_i} = \begin{bmatrix} \frac{\partial \mathbf{r}_i}{\partial \mathbf{T}_i} & \frac{\partial \mathbf{r}_i}{\partial \boldsymbol{\beta}_i} \end{bmatrix} \text{ and } \mathbf{J}_{i,2} \triangleq \frac{\partial \mathbf{r}_i}{\partial \boldsymbol{\Omega}_i}, \tag{18}$$

in Eq. (16) are the Jacobians, and

$$\mathbf{A}_i \triangleq \begin{bmatrix} \frac{\partial \mathtt{F}_i}{\partial \mathbf{T}_{\mathrm{par}(i)}} & \frac{\partial \mathtt{F}_i}{\partial \boldsymbol{\beta}_{\mathrm{par}(i)}} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \text{ and } \mathbf{B}_i \triangleq \begin{bmatrix} \frac{\partial \mathtt{F}_i}{\partial \boldsymbol{\Omega}_i} \\ \mathbf{0} \end{bmatrix} \tag{19}$$

in Eq. (17) are the partial derivatives of Eq. (15). For $\Delta\mathbf{x}_i = (\Delta\mathbf{T}_i, \Delta\boldsymbol{\beta}_i) \in \mathbb{R}^{6+P}$ in Eqs. (16) and (17), note that $\Delta\mathbf{T}_i \in \mathbb{R}^6$ and $\Delta\boldsymbol{\beta}_i \in \mathbb{R}^P$ are the Gauss-Newton direction of $\mathbf{T}_i$ and $\boldsymbol{\beta}_i$, respectively.

Step 2: We reformulate Eqs. (16) and (17) as

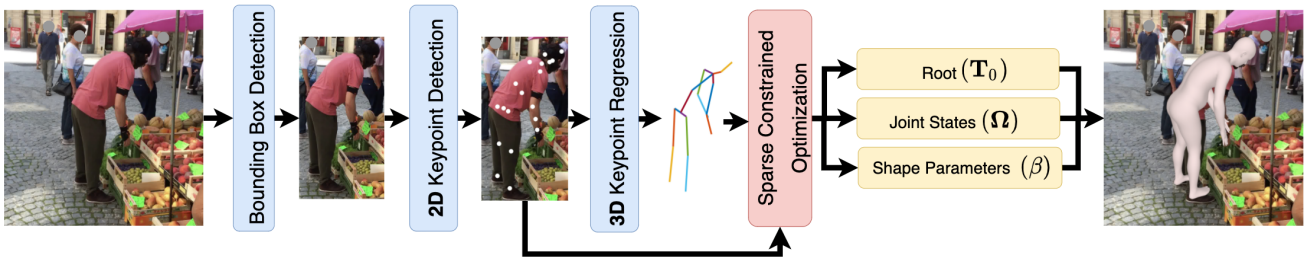


Figure 2: Overview of our motion capture framework. Given an image, our preprocessing pipeline estimates a bounding box, 2D and 3D keypoints. The 2D and 3D keypoints are then sent to our fast sparse constrained optimizer for 3D pose and shape reconstruction. Note that 3D keypoints are used to compute the part orientation fields [39].

$$\min_{\{\Delta\mathbf{x}_i,\Delta\mathbf{\Omega}_i\}_{i=0}^K} \Delta\mathbf{E}=\sum_{i=0}^{K}\Big[\frac{1}{2}\Delta\mathbf{x}_i^\top\mathbf{H}_{i,11}\Delta\mathbf{x}_i+\Delta\mathbf{\Omega}_i^\top\mathbf{H}_{i,21}\Delta\mathbf{x}_i+ \frac{1}{2}\Delta\mathbf{\Omega}_i^\top\mathbf{H}_{i,22}\Delta\mathbf{\Omega}_i+\mathbf{g}_{i,1}^\top\Delta\mathbf{x}_i+\mathbf{g}_{i,2}^\top\Delta\mathbf{\Omega}_i\Big], \quad (20)$$

subject to

$$\Delta\mathbf{x}_i = \mathbf{A}_i\Delta\mathbf{x}_{\mathrm{par}(i)}+\mathbf{B}_i\Delta\mathbf{\Omega}_i, \quad (21)$$

in which $\mathbf{H}_{i,11} \triangleq \mathbf{J}_{i,1}^\top\mathbf{J}_{i,1}$, $\mathbf{H}_{i,21} \triangleq \mathbf{J}_{i,2}^\top\mathbf{J}_{i,1}$ and $\mathbf{H}_{i,22} \triangleq \mathbf{J}_{i,2}^\top\mathbf{J}_{i,2}$ are the Hessians, and $\mathbf{g}_{i,1} \triangleq \mathbf{J}_{i,1}^\top\mathbf{r}_i$ and $\mathbf{g}_{i,2} \triangleq \mathbf{J}_{i,2}^\top\mathbf{r}_i$ are the gradients.

Step 3: Solve Eqs. (20) and (21) to compute the Gauss-Newton direction $\{\Delta\mathbf{x}_i, \Delta\mathbf{\Omega}_i\}_{i=0}^K$.

Here, **Steps 1 to 3** compute the Gauss-Newton direction $\{\Delta\mathbf{x}_i, \Delta\mathbf{\Omega}_i\}_{i=0}^K$ by solving a constrained quadratic optimization problem. The following proposition is for its completeness and complexity.

Proposition 2. The resulting $\{\Delta\mathbf{x}_i, \Delta\mathbf{\Omega}_i\}_{i=0}^K$ for Eqs. (14) and (15) is also the Gauss-Newton direction for Eq. (11). Furthermore, Eqs. (14) and (15) take $O(K)+O(N)$ time to compute $\{\Delta\mathbf{x}_i, \Delta\mathbf{\Omega}_i\}_{i=0}^K$ using **Steps 1 to 3**, in which K and N are the number of joints and measurements of the 3D human model, respectively. In contrast, Eq. (11) has a complexity of $O(K^3)+O(K^2N)$.

Proof. Please refer to the Supplementary Material. □

In general, the computation of the Gauss-Newton direction occupies a significant portion of workloads in optimization. Since our sparse constrained formulation improves this computation by two orders in terms of the number of joints and has the number of joints and measurements decoupled for the complexity, it is expected that our resulting method greatly improves the efficiency of optimization.

5. Evaluation

In this section, we present quantitative and qualitative evaluation of our method against state-of-the-art optimization and regression methods on multiple public benchmark datasets. All experiments are done on an Intel Xeon E3-1505M 3.0GHz CPU and a NVIDIA Quadro GP 100 GPU.

5.1. Datasets

We evaluate all methods on the following datasets.

Human3.6M (H36M) [7, 10] is one of the most commonly used datasets for 3D human pose (and shape) estimation (it was obtained and used by coauthors affiliated with academic institutions). Following the standard training-testing protocol established in [29], we use subjects S9 and S11 for evaluation.

MPI-INF-3DHP [23] is a markerless dataset with multiple viewpoints. We use subjects TS1-TS6 for evaluation where the first four (TS1-TS4) are in a controlled lab environment and the last two are in the wild (TS5-TS6).

3DPW [37] is an in-the-wild dataset captured from a moving single hand-held camera. IMU sensors are also used to compute ground-truth poses and shapes using the SMPL model. We use its defined test dataset for evaluation.

5.2. Real-time Motion Capture Framework

We design a real-time monocular motion capture framework, illustrated in Figure 2, based on our fast optimization method to recover 3D human poses and shapes from a single image. Similar to the other frameworks [24, 25], ours consists of a preprocessing pipeline with the input image fed to YOLOv4-CSP [4, 38] for human detection, then to AlphaPose [8] for 2D keypoint estimation, and finally to a light-weight neural network that is a modification of VideoPose3D [30] for 2D-to-3D lifting. The output of the preprocessing pipeline is then sent to our fast optimizer for 3D reconstruction. The Python API of NVIDIA TensorRT 7.2.1 is used to accelerate the inference of the preprocessing neural networks. Please refer to the Supplementary Material for more details on our motion capture framework.

5.3. Computation Times

We evaluate all methods on their computation or inference times on the Human3.6M dataset [10] dataset. We compare optimization methods against ours on the optimization only time and compare all methods on the total computation time per image.

Optimization time is reported in column 4 of Table 1.

	Method	Time (s)				Protocol 1		Protocol 2
		Preprocessing	Optimization	Regression	Total	MPJPE ↓	PA-MPJPE ↓	PA-MPJPE ↓
Pose only	Rogez et al. [31]	–	n/a	–	–	–	–	87.3
	Rogez et al. [32]	–	n/a	–	–	87.7	71.6	–
	Pavlakos et al. [29]	–	n/a	–	–	71.9	51.2	51.9
	Martinez et al. [22]	–	n/a	–	–	–	–	**47.7**
	Pavllo et al. [30]	–	n/a	–	–	**51.8**	**40**	–
	*VNect [25]	0.026	0.008	n/a	0.034	80.5	–	–
Pose and shape	HMR [12]	0.017	n/a	0.032	0.049	88.0	58.1	56.8
	Kolotouros et al. [16]	0.017	n/a	0.023	0.040	74.7	51.9	50.1
	SPIN [15]	0.017	n/a	**0.012**	**0.029**	**65.6**	**44.6**	**41.1**
	*SMPLify [5]	0.029	45	n/a	45	–	–	82.3
	*UP-P91 [17]	0.029	40	n/a	40	–	–	80.7
	*MTC [39]	0.029	20	n/a	20	64.5	–	–
	*Ours	0.029	**0.004**	n/a	**0.033**	**61.5**	48.2	**46.3**

(*) optimization method (n/a) not applicable (–) unreported statistic

Table 1: Evaluation on the Human3.6M dataset comparing computational times (s) and accuracy (mm) with Protocols 1 and 2. Overall, our method significantly outperforms all optimization methods with orders of magnitude speed up, and is competitive against the best performing regression method SPIN [15]. Preprocessing time for regression methods is the generation of human bounding boxes with YOLOv4-CSP [38], and for optimization methods is the inference time of the front-end neural network. All the optimization is run on CPU. VNect, MTC and ours are in C++, and SMPLify and UP-P91 are in Python.

Our method converges in 20-50 iterations taking less than 4ms on average to reconstruct 3D human poses and shapes. In contrast to existing optimization methods that estimate pose and shape [5, 17, 39] in 20-45s, ours is 4 orders of magnitude faster. As discussed earlier, our method uses the SPML model with 2.6 times as many variables (75 degrees of freedom and 10 shape parameters) as the 3D skeleton in VNect [25] (33 degrees of freedom and no shape parameters)—note that the complexity of optimization problems typically increases superlinearly with the number of optimization variables. Our optimization method is still twice as fast as VNect that only estimates poses (with an objective function with fewer loss terms). We attribute the significant improvements in optimization times to our sparse constrained formulation whose computation of the Gauss-Newton direction has linear rather than cubic complexity with the number of joints and measurements. The ablation studies in Section 5.6 and the Supplementary Material further support our complexity analysis.

Total time includes the preprocessing time and any optimization or regression time and reflects the overall time it takes for a method to produce estimates given an image. All timings are reported in columns 3-6 of Table 1. The regression methods [12, 15, 16] use ground-truth bounding boxes during evaluation. Therefore, we assume YOLOv4-CSP [4, 38] (17ms) is used in practice to obtain bounding boxes from images and count it as the preprocessing time per image. For the optimization methods, the preprocessing time of VNect [25] is computed from its own neural networks while for others [5, 17, 39] the preprocessing pipeline is similar to ours and we assume their times (29ms) are close to ours. Note that in our method the 29ms preprocessing time is a significant portion of the total time, while for the other optimization methods (that estimate pose and shape) it is negligible compared to their optimization times. SPIN [15] has the lowest total time of 29ms and ours is a close second with 33ms. Our motion capture framework thus has a speed of over 30 FPS which is sufficient for real-time applications.

5.4. Accuracy

Human3.6M. We evaluate all methods on the Mean Per-Joint Position Errors without (MPJPE) and with (PA-MPJPE) Procrustes Alignment on two common protocols. Protocol 1 uses all the four cameras and Protocol 2 only uses the frontal camera. The results are reported in columns 7-9 of Table 1. Our framework outperforms the other methods on Protocol 1 MPJPE, and achieve the second lowest PA-MPJPE slightly behind SPIN [15] on both Protocols 1 and 2. Though not presented in Table 1, our method also has the lowest MPJPE on Protocol 2, which is 60.3 mm.

MPI-INF-3DHP. This is a more challenging dataset than Human3.6M dataset. In addition to MPJPE, we also compare on Percentage of Correct Keypoints (PCK) with a threshold of 150 mm and Area Under the Curve (AUC) for a range of PCK thresholds as alternate metrics for evaluation. The results of MPI-INF-3DHP without and with rigid alignment are presented in Table 2. Our method achieves the state-of-the-art performance on all metrics.

3DPW. The results are reported in Table 3. Our method has the second lowest MPJPE and PA-MPJPE, and is competitive against the regression method SPIN [15]. Our method also outperforms regression methods that use multiples frames [3, 13].

Method	PCK ↑	AUC ↑	MPJPE ↓
Absolute (w/o rigid alignment)			
Mehta et al. [5]	75.7	39.3	117.6
HMR [12]	72.9	36.5	124.2
SPIN [15]	76.4	37.1	105.2
*XNect [24]	77.8	38.9	115.0
*VNect [25]	76.6	40.4	124.7
*Ours	**83.0**	**41.9**	**91.5**
Rigid aligned			
HMR [12]	86.3	47.8	89.8
SPIN [15]	92.5	55.6	67.5
*VNect [25]	83.9	47.3	98.0
*Ours	**94.6**	**59.0**	**62.1**

Table 2: Evaluation on the MPI-INF-3DHP dataset. Our method outperforms optimization (denoted by *) and regression methods over multiple accuracy metrics before and after rigid alignment.

Method	MPJPE ↓	PA-MPJPE ↓
HMR [12]	130	81.3
Kolotouros et al. [16]	–	70.2
SPIN [15]	**96.9**	**59.2**
‡Arnab et al. [3]	–	72.2
‡Kanazawa et al. [13]	116.5	72.6
*XNect [24]	134.2	80.3
*Ours	98.6	68.0

Table 3: Evaluation on the 3DPW dataset. Our method is competitive against the best regression method SPIN. * denotes optimization method and ‡ indicates that the method uses multiple frames.

5.5. Qualitative Results

We present typical failure cases due to inaccurate detection of our preprocessing pipeline in Fig. 3 and qualitative comparisons with SPIN [15] and SMPLify [5] on difficult examples from the Human3.6M, MPI-INF-3DHP and 3DPW datasets in Fig. 4. For a fair comparison, we add extra 3D keypoint measurements to SMPLify to improve its performance. We also show more qualitative results in the Supplementary Material. In Fig. 4 and Supplementary Material, it can be seen that our method has better pixel alignment than SPIN [15] and generates results of higher quality than SMPLify [5].

5.6. Ablation Studies

In the ablation stuidies, we perform the following experiments on the SMPL model [20] with $K = 23$ joints and SMPL+H model [33] with $K = 51$ joints to compute the Gauss-Newton direction.

Experiment 1. The number of shape parameters P is 0 and the number of measurements N increases from 120 to 600 for both of the SMPL and SMPL+H models.

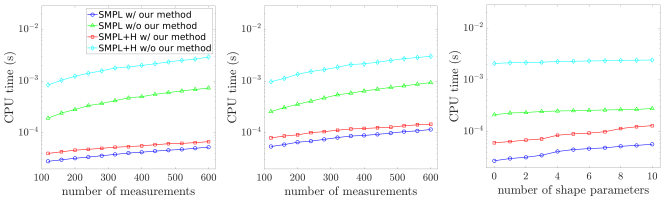
Experiment 2. The number of shape parameters P is 10 and the number of measurements N increases from 120 to 600 for both of the SMPL and SMPL+H models.

Figure 3: Typical failure cases of our method due to (left) body part occlusion, (middle) incorrect body orientation detection, (right) depth ambiguity of monocular camera.

Figure 4: Qualitative comparisons of our method (second row in pink), SPIN [15] (third row in gray), and SMPLify [5] (fourth row in purple) on the Human3.6M, MPI-INF-3DHP and 3DPW datasets. Please see Supplementary Material for more qualitative comparisons.

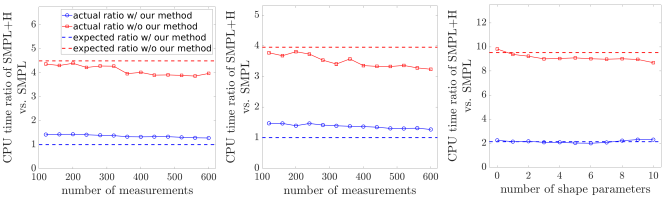
Experiment 3. The number of shape parameters P increases from 0 to 10, and each joint of the SMPL and SMPL+H models is assigned with a 2D keypoint, a 3D keypoint, and a part orientation field as measurements.

The SMPL and SMPL+H models have different numbers



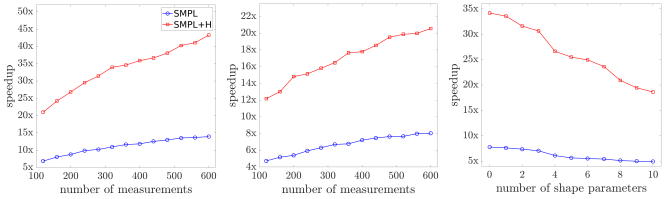
(a) Experiment 1 (b) Experiment 2 (c) Experiment 3

Figure 5: The CPU times on the SMPL and SMPL+H models w/ and w/o our method in Experiments 1 to 3.



(a) Experiment 1 (b) Experiment 2 (c) Experiment 3

Figure 6: The CPU time ratios of the SMPL+H vs. SMPL models w and w/o our method in Experiments 1 to 3.



(a) Experiment 1 (b) Experiment 2 (c) Experiment 3

Figure 7: The speedups on the SMPL and SMPL+H models w/ our method in Experiments 1 to 3.

of joints, and Experiments 1 to 3 have varying numbers of measurements and shape parameters. Thus, these experiments are sufficient to evaluate the impacts of the number of joints K, measurements N and shape parameters P on the computation of the Gauss-Newton direction. A more complete analysis of ablation studies is presented in the Supplementary Material.

The CPU times on the SMPL and SMPL+H models w/ and w/o our method are reported in Fig. 5. In all the experiments, our method using the sparse constrained formulation is a lot faster than that using the dense unconstrained formulation regardless of the number of joints, measurements and shape parameters.

The CPU time ratios of the SMPL+H vs. SMPL models w and w/o our method are reported in Fig. 6. As mentioned before, the SMPL and SMPL+H models have $K = 23$ and $K = 51$ joints, respectively, and as a result, such CPU time ratios reflect the influences of the number of joints K on the computation of the Gauss-Newton direction. The calculation of the expected CPU ratios w/ and w/o method in Fig. 6 is provided in the Supplementary Material. In Fig. 6, it can be seen that the impacts of the number of joints is around $O(K^2)$ times less on our method, which is consistent with the $O(K)$ complexity of our sparse constrained formation against $O(K^3)$ of the dense unconstrained one.

The speedups on the SMPL and SMPL+H models w/ our method are reported in Fig. 7. In Figs. 7(a) and 7(b), our method has greater speedup if there are more measurements, and achieves better performance on the SMPL+H model with more joints, whose results are expected since our sparse constrained formulation has $O(N)$ complexity—note that N is not coupled with K—in contrast to the dense unconstrained formulation with $O(K^2N)$ complexity, in which K and N are the number of joints and measurements, respectively. In Fig. 7(c), it can be seen that that speedup decreases with more shape parameters, and this is due to that both formulations have the same complexities for the shape parameters.

6. Discussion

We revitalized the optimization approach to address the problem of 3D human pose and shape estimation by presenting a sparse constrained formulation that performs on par with regression methods. We demonstrated how to exploit the sparsity in our formulation and build an optimizer that can compute the Gauss-Newton direction in only linear complexity (with respect to the number of joints and measurements in the human model). This was a key contributing factor in bringing down the computation times of existing optimization methods by orders of magnitude to 4ms. In benchmarks across multiple datasets on several metrics our framework, that uses a preprocessing neural network plus our optimizer, was highly competitive against the best performing regression method in terms of speed and accuracy.

We note that our fast framework can also benefit regression methods by quickly refining their outputs or by reducing training times for methods that train with some optimization in the loop.

The qualitative results illustrate that our framework was mainly limited by the reliability of the preprocessor. While our primary focus in this work was on the optimization side, some investment in engineering the preprocessor could yield further improvements in performance. Although we employed the SMPL model in our current implementation, our optimizer has the flexibility to support other types of 3D human models if the appropriate loss terms are specified for the objective. In particular, sparse 3D human models such as STAR [27] would be well suited for our method. With an additional preprocessor, and model and loss terms to support human hands and facial expressions, our framework can also be extended to address the total 3D human capture problem.

Acknowledgments. For this work authors affiliated with Northwestern University were partially supported by the National Science Foundation under award DCSD-1662233.

Supplementary Materials
Revitalizing Optimization for 3D Human Pose and Shape Estimation: A Sparse Constrained Formulation

Taosha Fan[1,2], Kalyan Vasudev Alwala[1], Donglai Xiang[3,4], Weipeng Xu[3],
Todd Murphey[2], Mustafa Mukadam[1]

[1]Facebook AI Research, [2]Northwestern University,
[3]Facebook Reality Labs, [4]Carnegie Mellon University

Abstract

In this supplementary material, we present the proofs of the propositions in the paper and a comprehensive complexity analysis of the dense unconstrained and sparse constrained formulations for 3D human pose and shape estimation, from which we derive an efficient algorithm to compute the Gauss-Newton direction. In addition, we present more results of qualitative comparisons and ablation studies to validate our work. Finally, we provide a more detailed description of our real-time motion capture framework, the prior loss of joint states, and how to implement our method on similar articulated tracking problems.

A. Proofs

A.1. Proof of Proposition 1

In this proof, we show the following two optimization problems are equivalent:

$$\min_{\mathbf{T}_0,\,\boldsymbol{\Omega},\,\boldsymbol{\beta}} \mathtt{E} = \sum_{i=0}^{K} \frac{1}{2} \|\mathtt{r}_i(\mathbf{T}_0,\,\boldsymbol{\Omega},\,\boldsymbol{\beta})\|^2, \tag{22}$$

and

$$\min_{\{\mathbf{T}_i,\,\boldsymbol{\Omega}_i,\,\boldsymbol{\beta}_i\}_{i=0}^{K}} \mathtt{E} = \sum_{i=0}^{K} \frac{1}{2} \|\mathtt{r}_i(\mathbf{T}_i,\,\boldsymbol{\Omega}_i,\,\boldsymbol{\beta}_i)\|^2 \tag{23}$$

subject to

$$\begin{aligned} \mathbf{T}_i =& \mathtt{F}_i(\mathbf{T}_{\mathrm{par}(i)},\,\boldsymbol{\beta}_{\mathrm{par}(i)},\,\boldsymbol{\Omega}_i) \\ \triangleq& \mathbf{T}_{\mathrm{par}(i)} \begin{bmatrix} \boldsymbol{\Omega}_i & \mathcal{S}_i \cdot \boldsymbol{\beta}_{\mathrm{par}(i)} + \mathbf{l}_i \\ \mathbf{0} & 1 \end{bmatrix}, \end{aligned} \tag{24a}$$

$$\boldsymbol{\beta}_i = \boldsymbol{\beta}_{\mathrm{par}(i)}. \tag{24b}$$

In Eqs. (22) and (23), $\mathbf{T}_i \in SE(3)$ is the rigid body transformation of body part i, and $\boldsymbol{\Omega}_i$ is the state of joint i, and $\boldsymbol{\Omega} \triangleq (\boldsymbol{\Omega}_1, \cdots, \boldsymbol{\Omega}_K) \in SO(3)^K$ are the joint states, and $\boldsymbol{\beta}$ and $\boldsymbol{\beta}_i \in \mathbb{R}^P$ are the shape parameters, and $\mathtt{F}_i(\cdot) : SE(3) \times \mathbb{R}^P \times SO(3) \to SE(3)$ is a function that maps $\mathbf{T}_{\mathrm{par}(i)}$, $\boldsymbol{\beta}_{\mathrm{par}(i)}$ and $\boldsymbol{\Omega}_i$ to $\mathbf{T}_i$. Note that Eqs. (22) and (23) are the dense unconstrained and sparse constrained formulations, respectively, for 3D human pose and shape estimation that are defined in the paper.

From Eq. (24b), if we let $\boldsymbol{\beta}_0 = \boldsymbol{\beta}$, then, $\boldsymbol{\beta}_i = \boldsymbol{\beta}$ for all $i = 1, \cdots, K$. Thus, Eq. (23) is reduced to

$$\min_{\{\mathbf{T}_i,\,\boldsymbol{\Omega}_i,\,\boldsymbol{\beta}_i\}_{i=0}^{K}} \mathtt{E} = \sum_{i=0}^{K} \frac{1}{2} \|\mathtt{r}_i(\mathbf{T}_i,\,\boldsymbol{\Omega}_i,\,\boldsymbol{\beta})\|^2 \tag{25}$$

subject to

$$\begin{aligned} \mathbf{T}_i =& \mathtt{F}_i(\mathbf{T}_{\mathrm{par}(i)},\,\boldsymbol{\beta},\,\boldsymbol{\Omega}_i) \\ =& \mathbf{T}_{\mathrm{par}(i)} \begin{bmatrix} \boldsymbol{\Omega}_i & \mathcal{S}_i \cdot \boldsymbol{\beta} + \mathbf{l}_i \\ \mathbf{0} & 1 \end{bmatrix}. \end{aligned} \tag{26}$$

Next, as mentioned in the paper, if we perform a top-down traversal of the kinematic tree of the SMPL model and recursively exploit Eq. (26) for each body part $i = 1, \cdots, K$, then, all of $\mathbf{T}_i \in SE(3)$ can be represented as a function of the root pose $\mathbf{T}_0 \in SE(3)$, and the joint states $\boldsymbol{\Omega} \in SO(3)^K$, and the shape parameter $\boldsymbol{\beta} \in \mathbb{R}^P$, i.e.,

$$\mathbf{T}_i \triangleq \mathbf{T}_i\,(\mathbf{T}_0,\,\boldsymbol{\Omega},\,\boldsymbol{\beta}) \tag{27}$$

If we use Eq. (27) to cancel out non-root rigid body transformations $\mathbf{T}_i$ ($1 \leq i \leq K$), then, each $\mathtt{r}_i(\cdot)$ in Eq. (25) is rewritten as a function of $\mathbf{T}_0 \in SE(3)$, and $\boldsymbol{\Omega} \in SO(3)^K$, and $\boldsymbol{\beta} \in \mathbb{R}^P$, from which we obtain an optimization problem of a dense unconstrained formulation

$$\min_{\mathbf{T}_0,\,\boldsymbol{\Omega},\,\boldsymbol{\beta}} \mathtt{E} = \sum_{i=0}^{K} \frac{1}{2} \|\mathtt{r}_i(\mathbf{T}_0,\,\boldsymbol{\Omega},\,\boldsymbol{\beta})\|^2$$

that is the same as Eq. (22). Therefore, it can be concluded that Eqs. (22) and (23) are equivalent. The proof is completed.

A.2. Proof of Proposition 2

The proof of proposition 2 is organized as follows: we present an overview of the steps to compute the Gauss-Newton direction in Section A.2.1, and show that the steps for the two formulations result in the same Gauss-Newton direction in Section A.2.2, and derive a dynamic programming algorithm to solve the quadratic program of the sparse constrained formulation in Section A.2.3, and analyze the complexity of the aforementioned steps to compute the Gauss-Newton direction in Section A.2.4.

A.2.1 Steps to Compute the Gauss-Newton Direction

With similar notation to the paper, we introduce $\mathbf{x} \triangleq (\mathbf{T}_0, \boldsymbol{\Omega}, \boldsymbol{\beta}) \in SE(3) \times SO(3)^K \times \mathbb{R}^P$ and $\mathbf{x}_i \triangleq (\mathbf{T}_i, \boldsymbol{\beta}_i) \in SE(3) \times \mathbb{R}^P$. Then Eqs. (22) and (23) can be rewritten as

$$\min_{\mathbf{x}} \mathtt{E} = \sum_{i=0}^{K} \frac{1}{2} \|\mathtt{r}_i(\mathbf{x})\|^2, \tag{28}$$

and

$$\min_{\{\mathbf{x}_i,\, \boldsymbol{\Omega}_i\}_{i=0}^K} \mathtt{E} = \sum_{i=0}^{K} \frac{1}{2} \|\mathtt{r}_i(\mathbf{x}_i,\, \boldsymbol{\Omega}_i)\|^2 \tag{29}$$

subject to

$$\mathbf{x}_i = \begin{bmatrix} \mathtt{F}_i(\mathbf{x}_{\mathrm{par}(i)},\, \boldsymbol{\Omega}_i) \\ \boldsymbol{\beta}_{\mathrm{par}(i)} \end{bmatrix}, \tag{30}$$

respectively. For analytical clarity, we assume with no loss of generality that the residues $\mathtt{r}_i(\mathbf{x})$ and $\mathtt{r}_i(\mathbf{x}_i,\, \boldsymbol{\Omega}_i)$ are $N_i \times 1$ vectors for $i = 0, \cdots, K$.

Following the procedure originally given in the paper, an overview of steps to compute the Gauss-Newton direction for the dense unconstrained and sparse constrained formulations is given in Table 4, which will be frequently used in the rest of this proof.

A.2.2 The Equivalence of the Gauss-Newton Direction

In Table 4, since Steps 2 and 3 are the reformulation of Step 1, we only need to show that the linearizations of dense unconstrained and sparse constrained formulations in Step 1, i.e., Eqs. (31) and (33), are equivalent. From Eq. (27), the rigid body transformation $\mathbf{T}_i \in SE(3)$ of body part i can be written as a function of $\mathbf{T}_0$, $\boldsymbol{\Omega}$ and $\boldsymbol{\beta}$. Furthermore, it is by the definition of $\mathtt{r}_i(\cdot)$ that

$$\mathtt{r}_i(\mathbf{T}_0,\, \boldsymbol{\Omega},\, \boldsymbol{\beta}) = \mathtt{r}_i\big(\mathbf{T}_i(\mathbf{T}_0,\, \boldsymbol{\Omega},\, \boldsymbol{\beta}),\, \boldsymbol{\Omega}_i,\, \boldsymbol{\beta}\big).$$

From the equation above, $\mathbf{J}_i \Delta\mathbf{x}$ in Eq. (31) can be computed using $\mathbf{J}_{i,1}$ and $\mathbf{J}_{i,2}$ in Eq. (33):

$$\mathbf{J}_i \Delta\mathbf{x} = \mathbf{J}_{i,1} \begin{bmatrix} \frac{\partial \mathbf{T}_i}{\partial \mathbf{T}_0} \Delta\mathbf{T}_0 + \frac{\partial \mathbf{T}_i}{\partial \boldsymbol{\Omega}} \Delta\boldsymbol{\Omega} + \frac{\partial \mathbf{T}_i}{\partial \boldsymbol{\beta}} \Delta\boldsymbol{\beta} \\ \boldsymbol{\beta} \end{bmatrix} + \mathbf{J}_{i,2} \Delta\boldsymbol{\Omega}_i. \tag{42}$$

Note that the partial derivatives $\frac{\partial \mathbf{T}_i}{\partial \mathbf{T}_0}$, $\frac{\partial \mathbf{T}_i}{\partial \boldsymbol{\Omega}}$ and $\frac{\partial \mathbf{T}_i}{\partial \boldsymbol{\beta}}$ in the right-hand side of Eq. (42) are obtained by the recursive implementation of Eq. (34). Therefore, it can be concluded that Eqs. (31) and (33) are equivalent to each other, which suggests that the dense unconstrained and sparse constrained formulations result in the same Gauss-Newton direction.

A.2.3 Algorithm to Solve Eq. (40)

In Table 4, it is straightforward to follow **Steps 1–2** of the sparse constrained formulation to compute the Gauss-Newton direction. Next, we need to solve the quadratic program of Eq. (40) in **Step 3**, which is nontrivial. In this subsection, we derive a dynamic programming algorithm that exploits the sparsity and constraints of Eq. (34) such that the Gauss-Newton direction can be exactly computed.

For notational simplicity, we let $\mathrm{par}(i)$, $\mathrm{chd}(i)$ and $\mathrm{des}(i)$ be the parent, children and descendants of body part i in the kinematics tree, and assume $i > \mathrm{par}(i)$ for all $i = 1, \cdots, K$.

First, we define $\mathcal{E}_i(\cdot) : \mathbb{R}^{6+P} \to \mathbb{R}$ to be a function of $\Delta\mathbf{x}_{\mathrm{par}(i)} \in \mathbb{R}^{6+P}$ in the form of an optimization problem of $\{\Delta\mathbf{x}_j,\, \Delta\boldsymbol{\Omega}_j\}$ for $j \in \{i\} \cup \mathrm{des}(i)$

$$\begin{aligned} &\mathcal{E}_i(\Delta\mathbf{x}_{\mathrm{par}(i)}) \triangleq \\ &\min_{\{\Delta\mathbf{x}_j,\, \Delta\boldsymbol{\Omega}_j\}_{j \in \{i\} \cup \mathrm{des}(i)}} \sum_{j \in \{i\} \cup \mathrm{des}(i)} \Big[\frac{1}{2} \Delta\mathbf{x}_j^\top \mathbf{H}_{j,11} \Delta\mathbf{x}_j + \\ &\Delta\boldsymbol{\Omega}_j^\top \mathbf{H}_{j,21} \Delta\mathbf{x}_j + \frac{1}{2} \Delta\boldsymbol{\Omega}_j^\top \mathbf{H}_{j,22} \Delta\boldsymbol{\Omega}_j + \\ &\mathbf{g}_{j,1}^\top \Delta\mathbf{x}_j + \mathbf{g}_{j,2}^\top \Delta\boldsymbol{\Omega}_j \Big] \end{aligned} \tag{43}$$

subject to

$$\Delta\mathbf{x}_j = \mathbf{A}_j \Delta\mathbf{x}_{\mathrm{par}(j)} + \mathbf{B}_j \Delta\boldsymbol{\Omega}_j,\ \forall j \in \{i\} \cup \mathrm{des}(i), \tag{44}$$

in which $\Delta\mathbf{x}_{\mathrm{par}(i)} \in \mathbb{R}^{6+P}$ is given. Furthermore, if $\mathcal{E}_j(\cdot) : \mathbb{R}^{6+P} \to \mathbb{R}$ is defined for all $j \in \mathrm{chd}(i)$, then, it is from Eq. (43) that $\mathcal{E}_i(\cdot)$ can be reduced to an optimization problem of $\Delta\mathbf{x}_i$ and $\Delta\boldsymbol{\Omega}_i$

$$\begin{aligned} &\mathcal{E}_i(\Delta\mathbf{x}_{\mathrm{par}(i)}) \triangleq \min_{\Delta\mathbf{x}_i,\, \Delta\boldsymbol{\Omega}_i} \Big[\frac{1}{2} \Delta\mathbf{x}_i^\top \mathbf{H}_{i,11} \Delta\mathbf{x}_i + \\ &\Delta\boldsymbol{\Omega}_i^\top \mathbf{H}_{i,21} \Delta\mathbf{x}_i + \frac{1}{2} \Delta\boldsymbol{\Omega}_i^\top \mathbf{H}_{i,22} \Delta\boldsymbol{\Omega}_i + \mathbf{g}_{i,1}^\top \Delta\mathbf{x}_i + \\ &\mathbf{g}_{i,2}^\top \Delta\boldsymbol{\Omega}_i + \sum_{j \in \mathrm{chd}(i)} \mathcal{E}_j(\Delta\mathbf{x}_i) \Big] \end{aligned} \tag{45}$$

subject to

$$\Delta\mathbf{x}_i = \mathbf{A}_i \Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{B}_i \Delta\boldsymbol{\Omega}_i,$$

in which $\Delta\mathbf{x}_{\mathrm{par}(i)} \in \mathbb{R}^{6+P}$ is given. Note that Eq. (45) is an intermediate procedure that is essential for our dynamic programming algorithm.

	Dense Unconstrained Formulation	Sparse Constrained Formulation
Step 1	The linearization of Eq. (28) results in $$\min_{\Delta\mathbf{x}} \Delta\mathtt{E} = \sum_{i=0}^{K} \frac{1}{2}\|\mathbf{J}_i\Delta\mathbf{x} + \mathtt{r}_i\|^2, \tag{31}$$ in which $\Delta\mathbf{x} \triangleq (\Delta\mathbf{T}_0, \Delta\boldsymbol{\Omega}, \Delta\boldsymbol{\beta}) \in \mathbb{R}^{6+3K+P}$, $\Delta\mathbf{T}_0 \in \mathbb{R}^6$, $\Delta\boldsymbol{\Omega} \in \mathbb{R}^{3K}$ and $\Delta\boldsymbol{\beta} \in \mathbb{R}^P$ are the Gauss-Newton directions of $\mathbf{x}$, $\mathbf{T}_0$, $\boldsymbol{\Omega}$ and $\boldsymbol{\beta}$, respectively, and $$\mathbf{J}_i \triangleq \begin{bmatrix} \frac{\partial \mathtt{r}_i}{\partial \mathbf{T}_0} & \frac{\partial \mathtt{r}_i}{\partial \boldsymbol{\Omega}} & \frac{\partial \mathtt{r}_i}{\partial \boldsymbol{\beta}} \end{bmatrix} \in \mathbb{R}^{N_i \times (6+3K+P)} \tag{32}$$ is the Jacobian of $\mathtt{r}_i(\cdot)$, and $\mathtt{r}_i \in \mathbb{R}^{N_i}$ is the residue.	The linearization of Eq. (29) results in $$\min_{\{\Delta\mathbf{x}_i, \Delta\boldsymbol{\Omega}_i\}_{i=0}^K} \Delta\mathtt{E} = \sum_{i=0}^{K} \frac{1}{2}\|\mathbf{J}_{i,1}\Delta\mathbf{x}_i + \mathbf{J}_{i,2}\Delta\boldsymbol{\Omega}_i + \mathtt{r}_i\|^2 \tag{33}$$ subject to $$\Delta\mathbf{x}_i = \mathbf{A}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{B}_i\Delta\boldsymbol{\Omega}_i, \tag{34}$$ in which $\Delta\mathbf{x}_i \triangleq (\Delta\mathbf{T}_i, \Delta\boldsymbol{\beta}_i) \in \mathbb{R}^{6+P}$, $\Delta\mathbf{T}_i \in \mathbb{R}^6$, $\Delta\boldsymbol{\Omega}_i \in \mathbb{R}^3$ and $\Delta\boldsymbol{\beta}_i \in \mathbb{R}^P$ are the Gauss-Newton directions of $\mathbf{x}_i$, $\mathbf{T}_i$, $\boldsymbol{\Omega}_i$ and $\boldsymbol{\beta}_i$, respectively, and $$\mathbf{J}_{i,1} \triangleq \begin{bmatrix} \frac{\partial \mathtt{r}_i}{\partial \mathbf{T}_i} & \frac{\partial \mathtt{r}_i}{\partial \boldsymbol{\beta}_i} \end{bmatrix} \in \mathbb{R}^{N_i \times (6+P)} \tag{35}$$ and $$\mathbf{J}_{i,2} \triangleq \frac{\partial \mathtt{r}_i}{\partial \boldsymbol{\Omega}_i} \in \mathbb{R}^{N_i \times 3} \tag{36}$$ are the Jacobians of $\mathtt{r}_i(\cdot)$, and $$\mathbf{A}_i \triangleq \begin{bmatrix} \frac{\partial \mathtt{F}_i}{\partial \mathbf{T}_{\mathrm{par}(i)}} & \frac{\partial \mathtt{F}_i}{\partial \boldsymbol{\beta}_{\mathrm{par}(i)}} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \in \mathbb{R}^{(6+P)\times(6+P)} \tag{37}$$ and $$\mathbf{B}_i \triangleq \begin{bmatrix} \frac{\partial \mathtt{F}_i}{\partial \boldsymbol{\Omega}_i} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(6+P)\times 3} \tag{38}$$ are the partial derivatives of Eq. (34), and $\mathtt{r}_i \in \mathbb{R}^{N_i}$ is the residue.
Step 2	Reformulate Eq. (31) as $$\min_{\Delta\mathbf{x}} \Delta\mathtt{E} = \frac{1}{2}\Delta\mathbf{x}^\top\mathbf{H}\Delta\mathbf{x} + \mathbf{g}^\top\Delta\mathbf{x} \tag{39}$$ in which $\mathbf{H} \triangleq \sum_{i=0}^{K}\mathbf{J}_i^\top\mathbf{J}_i \in \mathbb{R}^{(6+3K+P)\times(6+3K+P)}$ is the Hessian, and $\mathbf{g} \triangleq \sum_{i=0}^{K}\mathbf{J}_i^\top\mathtt{r}_i \in \mathbb{R}^{(6+3K+P)}$ is the gradient.	Reformulate Eq. (33) as $$\min_{\{\Delta\mathbf{x}_i, \Delta\boldsymbol{\Omega}_i\}_{i=0}^K} \Delta\mathtt{E} = \sum_{i=0}^{K}\Big[\frac{1}{2}\Delta\mathbf{x}_i^\top\mathbf{H}_{i,11}\Delta\mathbf{x}_i + \Delta\boldsymbol{\Omega}_i^\top\mathbf{H}_{i,21}\Delta\mathbf{x}_i + \frac{1}{2}\Delta\boldsymbol{\Omega}_i^\top\mathbf{H}_{i,22}\Delta\boldsymbol{\Omega}_i + \mathbf{g}_{i,1}^\top\Delta\mathbf{x}_i + \mathbf{g}_{i,2}^\top\Delta\boldsymbol{\Omega}_i\Big] \tag{40}$$ subject to $$\Delta\mathbf{x}_i = \mathbf{A}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{B}_i\Delta\boldsymbol{\Omega}_i,$$ in which $\mathbf{H}_{i,11} \triangleq \mathbf{J}_{i,1}^\top\mathbf{J}_{i,1} \in \mathbb{R}^{(6+P)\times(6+P)}$, $\mathbf{H}_{i,21} \triangleq \mathbf{J}_{i,2}^\top\mathbf{J}_{i,1} \in \mathbb{R}^{3\times(6+P)}$, and $\mathbf{H}_{i,22} \triangleq \mathbf{J}_{i,2}^\top\mathbf{J}_{i,2} \in \mathbb{R}^{3\times3}$ are the Hessians, and $\mathbf{g}_{i,1} \triangleq \mathbf{J}_{i,1}^\top\mathtt{r}_i \in \mathbb{R}^{6+P}$ and $\mathbf{g}_{i,2} \triangleq \mathbf{J}_{i,2}^\top\mathtt{r}_i \in \mathbb{R}^{6+P}$ are the gradients.
Step 3	Compute the Gauss-Newton direction from Eq. (39), which has a closed-form solution $$\Delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{g}. \tag{41}$$	Compute the Gauss-Newton direction from Eq. (40), which can be exactly solved by Algorithm 1.

Table 4: Steps to compute the Gauss-Newton direction for the dense unconstrained and sparse constrained formulations.

Algorithm 1 Solve Eq. (40) and compute the Gauss-Newton direction

Input: $\{\mathbf{H}_{i,11}, \mathbf{H}_{i,21}, \mathbf{H}_{i,22}, \mathbf{g}_{i,1}, \mathbf{g}_{i,2}\}_{i=0}^{K}$

Output: $\{\Delta\mathbf{x}_i, \Delta\boldsymbol{\Omega}_i\}_{i=0}^{K}$ and $\Delta\mathrm{E}_0$

for $i = K \to 1$ **do**
  $\mathbf{N}_{i,11} = \mathbf{H}_{i,11} + \sum_{j\in\mathrm{chd}(i)} \mathbf{M}_{j,11}$
  $\mathbf{N}_{i,21} = \mathbf{H}_{i,21}$
  $\mathbf{N}_{i,22} = \mathbf{H}_{i,22}$
  $\mathbf{n}_{1,i} = \mathbf{g}_{i,1} + \sum_{j\in\mathrm{chd}(i)} \mathbf{m}_{j,1}$
  $\mathbf{n}_{i,2} = \mathbf{g}_{i,2}$
  $\Delta\bar{\mathrm{E}}_i = \sum_{j\in\mathrm{chd}(i)} \Delta\mathrm{E}_j$
  $\mathbf{Q}_{i,11} = \mathbf{A}_i^\top \mathbf{N}_{i,11}\mathbf{A}_i$
  $\mathbf{Q}_{i,21} = \mathbf{B}_i^\top \mathbf{N}_{i,11}\mathbf{A}_i + \mathbf{N}_{i,21}\mathbf{A}_i$
  $\mathbf{Q}_{i,22} = \mathbf{B}_i^\top \mathbf{N}_{i,11}\mathbf{B}_i + \mathbf{N}_{i,21}\mathbf{B}_i + \mathbf{B}_i^\top \mathbf{N}_{i,21}^\top + \mathbf{N}_{i,22}$
  $\mathbf{q}_{i,1} = \mathbf{A}_i^\top \mathbf{n}_{i,1}$
  $\mathbf{q}_{i,2} = \mathbf{B}_i^\top \mathbf{n}_{i,1} + \mathbf{n}_{i,2}$
  $\mathbf{K}_i = -\mathbf{Q}_{i,22}^{-1}\mathbf{Q}_{i,21}$
  $\mathbf{k}_i = -\mathbf{Q}_{i,22}^{-1}\mathbf{q}_{i,2}$
  $\mathbf{M}_{i,11} = \mathbf{Q}_{i,11} - \mathbf{Q}_{i,21}^\top \mathbf{Q}_{i,22}^{-1}\mathbf{Q}_{i,21}$
  $\mathbf{m}_{1,i} = \mathbf{q}_{i,1} - \mathbf{Q}_{i,21}^\top \mathbf{Q}_{i,22}^{-1}\mathbf{q}_{i,2}$
  $\Delta\mathrm{E}_i = \Delta\bar{\mathrm{E}}_i - \frac{1}{2}\mathbf{q}_{i,2}^\top \mathbf{Q}_{i,22}^{-1}\mathbf{q}_{i,2}$
end for
$\Delta\boldsymbol{\Omega}_0 = \mathbf{0}$
$\mathbf{M}_0 = \mathbf{H}_{0,11} + \sum_{j\in\mathrm{chd}(0)} \mathbf{M}_{j,11}$
$\mathbf{m}_0 = \mathbf{g}_{0,1} + \sum_{j\in\mathrm{chd}(0)} \mathbf{m}_{j,1}$
$\Delta\bar{\mathrm{E}}_0 = \sum_{i\in\mathrm{chd}(0)} \Delta\mathrm{E}_i$
$\mathbf{x}_0 = -\mathbf{M}_0^{-1}\mathbf{m}_0$
$\Delta\mathrm{E}_0 = \Delta\bar{\mathrm{E}}_0 - \frac{1}{2}\mathbf{m}_0^\top \mathbf{M}_0^{-1}\mathbf{m}_0$
for $i = 1 \to K$ **do**
  $\Delta\boldsymbol{\Omega}_i = \mathbf{K}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{k}_i$
  $\Delta\mathbf{x}_i = \mathbf{A}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{B}_i\Delta\boldsymbol{\Omega}_i$
end for

Next, suppose that there exists $\mathbf{M}_j \in \mathbb{R}^{(6+P)\times(6+P)}$, $\mathbf{m}_j \in \mathbb{R}^{6+P}$ and $\Delta\mathrm{E}_j \in \mathbb{R}$ for all $j \in \mathrm{chd}(i)$ such that $\mathcal{E}_j(\Delta\mathbf{x}_i)$ can be written as

$$\mathcal{E}_j(\Delta\mathbf{x}_i) = \frac{1}{2}\Delta\mathbf{x}_i^\top \mathbf{M}_j \Delta\mathbf{x}_i + \mathbf{m}_j^\top \Delta\mathbf{x}_i + \Delta\mathrm{E}_j. \tag{46}$$

Applying Eq. (46) to Eq. (45), we obtain

$$\begin{aligned}\mathcal{E}_i(\Delta\mathbf{x}_{\mathrm{par}(i)}) = \min_{\Delta\mathbf{x}_i,\,\Delta\boldsymbol{\Omega}_i} \frac{1}{2}\Delta\mathbf{x}_i\mathbf{N}_{i,11}\Delta\mathbf{x}_i + \\ \Delta\boldsymbol{\Omega}_i^\top \mathbf{N}_{i,21}\Delta\mathbf{x}_i + \frac{1}{2}\Delta\boldsymbol{\Omega}_i^\top \mathbf{N}_{i,22}\Delta\boldsymbol{\Omega}_i + \\ \mathbf{n}_{i,1}^\top\Delta\mathbf{x}_i + \mathbf{n}_{i,2}^\top\Delta\boldsymbol{\Omega}_i + \Delta\bar{\mathrm{E}}_i\end{aligned} \tag{47}$$

subject to

$$\Delta\mathbf{x}_i = \mathbf{A}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{B}_i\Delta\boldsymbol{\Omega}_i,$$

in which

$$\mathbf{N}_{i,11} = \mathbf{H}_{i,11} + \sum_{j\in\mathrm{chd}(i)} \mathbf{M}_i, \tag{48a}$$

$$\mathbf{N}_{i,21} = \mathbf{H}_{i,21}, \tag{48b}$$

$$\mathbf{N}_{i,22} = \mathbf{H}_{i,22}, \tag{48c}$$

$$\mathbf{n}_{i,1} = \mathbf{g}_{i,1} + \sum_{j\in\mathrm{chd}(i)} \mathbf{m}_j, \tag{48d}$$

$$\mathbf{n}_{i,2} = \mathbf{g}_{i,2}, \tag{48e}$$

$$\Delta\bar{\mathrm{E}}_i = \sum_{j\in\mathrm{chd}(i)} \Delta\mathrm{E}_j. \tag{48f}$$

Substitute Eq. (34) into Eq. (47) to cancel out $\Delta\mathbf{x}_i$ and simplify the resulting equation to an unconstrained optimization problem on $\Delta\boldsymbol{\Omega}_i \in \mathbb{R}^3$:

$$\begin{aligned}\mathcal{E}_i(\Delta\mathbf{x}_{\mathrm{par}(i)}) = \min_{\Delta\boldsymbol{\Omega}_i} \frac{1}{2}\Delta\mathbf{x}_{\mathrm{par}(i)}\mathbf{Q}_{i,11}\Delta\mathbf{x}_{\mathrm{par}(i)} + \\ \Delta\boldsymbol{\Omega}_i^\top \mathbf{Q}_{i,21}\Delta\mathbf{x}_{\mathrm{par}(i)} + \frac{1}{2}\Delta\boldsymbol{\Omega}_i^\top \mathbf{Q}_{i,22}\Delta\boldsymbol{\Omega}_i + \\ \mathbf{q}_{i,1}^\top\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{q}_{i,2}^\top\Delta\boldsymbol{\Omega}_i + \Delta\bar{\mathrm{E}}_i,\end{aligned} \tag{49}$$

in which

$$\mathbf{Q}_{i,11} = \mathbf{A}_i^\top \mathbf{N}_{i,11}\mathbf{A}_i, \tag{50a}$$

$$\mathbf{Q}_{i,21} = \mathbf{B}_i^\top \mathbf{N}_{i,11}\mathbf{A}_i + \mathbf{N}_{i,21}\mathbf{A}_i, \tag{50b}$$

$$\begin{aligned}\mathbf{Q}_{i,22} = \mathbf{B}_i^\top \mathbf{N}_{i,11}\mathbf{B}_i + \mathbf{N}_{i,21}\mathbf{B}_i + \\ \mathbf{B}_i^\top \mathbf{N}_{i,21}^\top + \mathbf{N}_{i,22},\end{aligned} \tag{50c}$$

$$\mathbf{q}_{i,1} = \mathbf{A}_i^\top \mathbf{n}_{i,1}, \tag{50d}$$

$$\mathbf{q}_{i,2} = \mathbf{B}_i^\top \mathbf{n}_{i,1} + \mathbf{n}_{i,2}. \tag{50e}$$

It is obvious that Eq. (49) has a closed-form solution

$$\Delta\boldsymbol{\Omega}_i = \mathbf{K}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{k}_i, \tag{51}$$

in which

$$\mathbf{K}_i = -\mathbf{Q}_{i,22}^{-1}\mathbf{Q}_{i,21} \quad \text{and} \quad \mathbf{k}_i = -\mathbf{Q}_{i,22}^{-1}\mathbf{q}_{i,2}. \tag{52}$$

If we use Eq. (51) to eliminate $\Delta\boldsymbol{\Omega}_i$ in Eq. (49), there exists $\mathbf{M}_i \in \mathbb{R}^{(6+P)\times(6+P)}$, $\mathbf{m}_i \in \mathbb{R}^{6+P}$ and $\Delta\mathrm{E}_i \in \mathbb{R}$ such that

$$\begin{aligned}\mathcal{E}_i(\Delta\mathbf{x}_{\mathrm{par}(i)}) = \frac{1}{2}\Delta\mathbf{x}_{\mathrm{par}(i)}^\top \mathbf{M}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \\ \mathbf{m}_i^\top\Delta\mathbf{x}_{\mathrm{par}(i)} + \Delta\mathrm{E}_i,\end{aligned} \tag{53}$$

in which

$$\mathbf{M}_i = \mathbf{Q}_{i,11} - \mathbf{Q}_{i,21}^\top \mathbf{Q}_{i,22}^{-1}\mathbf{Q}_{i,21}, \tag{54a}$$

	Dense Unconstrained Formulation	Sparse Constrained Formulation
Step 1	$O\big(N(6+3K+P)\big)$	$O\big(K(9+P)\big)+O\big(N(9+P)\big)$
Step 2	$O\big(N(6+3K+P)^2\big)$	$O\big(N(9+P)^2\big)$
Step 3	$O\big((6+3K+P)^3\big)$	$O\big(K(9+P)^2\big)+O\big((6+P)^3\big)$
Total	$O\big((6+3K+P)^3\big)+O\big(N(6+3K+P)^2\big)$	$O\big(K(9+P)^2\big)+O\big((6+P)^3\big)+O\big(N(9+P)^2\big)$

(a)

	Dense Unconstrained Formulation	Sparse Constrained Formulation
Step 1	$O\big(KN\big)$	$O(K)+O(N)$
Step 2	$O\big(K^2N\big)$	$O\big(N\big)$
Step 3	$O(K^3)$	$O(K)$
Total	$O(K^3)+O\big(K^2N\big)$	$O(K)+O\big(N\big)$

(b)

Table 5: The summary of the computational complexities for the steps to compute the Gauss-Newton direction for the dense unconstrained and sparse constrained formulations, in which K is the number of joints, P is the number of shape parameters, N is the number of measurements for all the body parts. Note that the number of shape parameters P is assumed to be varying in (a) and constant in (b).

$$\mathbf{m}_i = \mathbf{q}_{i,1} - \mathbf{Q}_{i,21}^\top \mathbf{Q}_{i,22}^{-1}\mathbf{q}_{i,2}, \tag{54b}$$

$$\Delta \mathrm{E}_i = \Delta \bar{\mathrm{E}}_i - \frac{1}{2}\mathbf{q}_{i,2}^\top \mathbf{Q}_{i,22}^{-1}\mathbf{q}_{i,2}. \tag{54c}$$

Therefore, if there exists $\mathbf{M}_j \in \mathbb{R}^{(6+P)\times(6+P)}$, $\mathbf{m}_j \in \mathbb{R}^{6+P}$ and $\Delta\mathrm{E}_j \in \mathbb{R}$ for all $j \in \mathrm{chd}(i)$ such that Eq. (46) holds, we might further obtain $\mathbf{M}_i \in \mathbb{R}^{(6+P)\times(6+P)}$, $\mathbf{m}_i \in \mathbb{R}^{6+P}$ and $\Delta\mathrm{E}_i \in \mathbb{R}$ with which $\mathcal{E}_i(\Delta\mathbf{x}_{\mathrm{par}(i)})$ can be written as Eq. (53).

In the kinematic tree, a body part i at the leaf node has no children, for which Eq. (48) is simplified to $\mathbf{N}_{i,11} = \mathbf{H}_{i,11}$, $\mathbf{N}_{i,21} = \mathbf{H}_{i,21}$, $\mathbf{N}_{i,22} = \mathbf{H}_{i,22}$, $\mathbf{n}_{i,1} = \mathbf{g}_{i,1}$, $\mathbf{n}_{i,2} = \mathbf{g}_{i,2}$ and $\Delta\bar{\mathrm{E}}_i = 0$, then, it is possible to recursively compute $\mathbf{M}_i \in \mathbb{R}^{(6+P)\times(6+P)}$, $\mathbf{m}_i \in \mathbb{R}^{6+P}$ and $\Delta\mathrm{E}_i \in \mathbb{R}$ for each $i = 1, \cdots, K$ following Eqs. (48), (50) and (54) through the bottom-up traversal of kinematic tree.

It is by definition that $\mathbf{\Omega}_0$ is a dummy variable and $\Delta\mathbf{\Omega}_0 = \mathbf{0}$. Thus, if $\mathcal{E}_i(\Delta\mathbf{x}_0)$ in Eq. (53) is known for each $i \in \mathrm{chd}(0)$, Eq. (40) is equivalent to an unconstrained optimization problem on $\Delta\mathbf{x}_0 \in \mathbb{R}^{6+P}$:

$$\min_{\Delta\mathbf{x}_0} \frac{1}{2}\Delta\mathbf{x}_0^\top \mathbf{H}_{0,11}\Delta\mathbf{x}_0 + \mathbf{g}_{0,1}^\top\Delta\mathbf{x}_0 + \sum_{j\in\mathrm{chd}(0)} \mathcal{E}_i(\Delta\mathbf{x}_0).$$

From Eq. (53), the equation above is equivalent to

$$\min_{\Delta\mathbf{x}_0} \frac{1}{2}\Delta\mathbf{x}_0^\top \mathbf{M}_0\Delta\mathbf{x}_0 + \mathbf{m}_0^\top\mathbf{x}_0 + \Delta\bar{\mathrm{E}}_0 \tag{55}$$

in which

$$\mathbf{M}_0 = \mathbf{H}_{0,11} + \sum_{j\in\mathrm{chd}(0)} \mathbf{M}_j, \tag{56a}$$

$$\mathbf{m}_0 = \mathbf{g}_{0,1} + \sum_{j\in\mathrm{chd}(0)} \mathbf{m}_j, \tag{56b}$$

$$\Delta\bar{\mathrm{E}}_0 = \sum_{i\in\mathrm{chd}(0)} \Delta\mathrm{E}_i. \tag{56c}$$

It is straightforward to show that

$$\Delta\mathbf{x}_0 = -\mathbf{M}_0^{-1}\mathbf{m}_0 \tag{57}$$

solves Eq. (55) with

$$\Delta\mathrm{E}_0 = \Delta\bar{\mathrm{E}}_0 - \frac{1}{2}\mathbf{m}_0^\top\mathbf{M}_0^{-1}\mathbf{m}_0 \tag{58}$$

to be the expected cost reduction as well as the minimum objective value of Eq. (40).

At last, we recursively compute $\{\Delta\mathbf{x}_i, \Delta\mathbf{\Omega}_i\}_{i=1}^K$ using Eqs. (34), (51) and (52) through a top-down traversal of the kinematics tree, from which the Gauss-Newton direction is exactly retrieved.

From our analysis, the resulting algorithm to solve Eq. (40) and compute the Gauss-Newton direction is summarized in Algorithm 1. In the next subsection, we show that Algorithm 1 scales linearly with respect to the number of joints.

A.2.4 Complexity Analysis

In Table 5, we present a short summary of the computational complexities for each step to compute the Gauss-Newton direction, and in Table 6, we present a comprehensive analysis of the computational complexities that leads to

	Dense Unconstrained Formulation	Sparse Constrained Formulation
Step 1	(a) It takes $O\big(N_i(6+3K+P)\big)$ time to compute $\mathbf{J}_i \in \mathbb{R}^{N_i\times(6+3K+P)}$ in Eq. (32) for each $i = 0, \cdots, K$. (b) In total, it takes $O\big(N(6+3K+P)\big)$ time to compute $\mathbf{J}_i \in \mathbb{R}^{N_i\times(6+3K+P)}$ for all $i = 0, \cdots, K$.	(a) It takes $O\big(9+P\big)$ time to compute $\mathbf{A}_i \in \mathbb{R}^{(6+P)\times(6+P)}$ and $\mathbf{B}_i \in \mathbb{R}^{(6+P)\times 3}$ in Eqs. (37) and (38) for each $i = 0, \cdots, K$. Note that the bottom of $\mathbf{A}_i$ and $\mathbf{B}_i$ in Eqs. (37) and (38) are either zero or identity matrices, which simplifies the computation. (b) It takes $O\big(N_i(9+P)\big)$ time to compute $\mathbf{J}_{i,1} \in \mathbb{R}^{N_i\times(9+P)}$ and $\mathbf{J}_{i,2} \in \mathbb{R}^{N_i\times 3}$ in Eqs. (35) and (36) for each $i = 0, \cdots, K$. (c) Note that $\mathbf{J}_{i,1}$, $\mathbf{J}_{i,2}$, $\mathbf{A}_i$ and $\mathbf{B}_i$ are intermediates to compute $\mathbf{J}_i$ in Eq. (32) using the chain rule. (d) In total, it takes $O\big(K(9+P)+O\big(N(9+P)\big)\big)$ time to compute $\mathbf{J}_{i,1}$, $\mathbf{J}_{i,2}$, $\mathbf{A}_i$ and $\mathbf{B}_i$ for all $i = 0, \cdots, K$.
Step 2	(a) It takes $O\big(N_i(6+3K+P)^2\big)$ to compute $\mathbf{J}_i^\top\mathbf{J}_i \in \mathbb{R}^{(6+3K+P)\times(6+3K+P)}$ for each $i = 0, \cdots, K$. (b) In total, it takes $O\big(N(6+3K+P)^2\big)$ time to compute $\mathbf{H} = \sum_{i=0}^{K}\mathbf{J}_i^\top\mathbf{J}_i \in \mathbb{R}^{(6+3K+P)\times(6+3K+P)}$ in Eq. (39).	(a) It takes $O\left(N_i(9+P)^2\right)$ time to compute $\mathbf{H}_{i,11} \in \mathbb{R}^{(6+P)\times(6+P)}$, $\mathbf{H}_{i,21} \in \mathbb{R}^{3\times(6+P)}$ and $\mathbf{H}_{i,22} \in \mathbb{R}^{3\times 3}$ in Eq. (40) for each $i = 0, \cdots, K$. (b) In total, it takes $O\big(N(9+P)^2\big)$ time to compute $\mathbf{H}_{i,11} \in \mathbb{R}^{(6+P)\times(6+P)}$, $\mathbf{H}_{i,21} \in \mathbb{R}^{3\times(6+P)}$ and $\mathbf{H}_{i,22} \in \mathbb{R}^{3\times 3}$ for all $i = 0, \cdots, K$.
Step 3	(a) In total, it takes $O\big((6+3K+P)^3\big)$ to compute the matrix inverse of $\mathbf{H} \in \mathbb{R}^{(6+3P+K)\times(6+3P+K)}$ and solve Eq. (41).	(a) It takes $O\big((9+P)^2\big)$ time to run lines 2-17 and lines 26-27 in Algorithm 1 for each $i = 1, \cdots, K$. Note that $\mathbf{A}_i$ and $\mathbf{B}_i$ in Eqs. (37) and (38) are zero and identity matrices at the bottom, which can be exploited to simplify the computation. (b) It takes $O\big((6+P)^3\big)$ time to compute the matrix inverse of $\mathbf{M}_0 \in \mathbb{R}^{(6+P)\times(6+P)}$ in line 23 of Algorithm 1. (c) In total, it takes $O\big(K(9+P)^2\big)+O\big((6+P)^3\big)$ to compute the Gauss-Newton direction.
Total	The overall complexity is $O\big((6+3K+P)^3\big)+O\big(N(6+3K+P)^2\big)$.	The overall complexity is $O\big(K(9+P)^2\big)+O\big((6+P)^3\big)+O\big(N(9+P)^2\big)$.

Table 6: The analysis of the computational complexities for the steps to compute the Gauss-Newton direction for the dense unconstrained and sparse constrained formulations. In this table, K is the number of joints, P is the number of shape parameters, N is the number of measurements for all the body parts, and N_i is the number of measurements associated with body part i.

results in Table 5. The analysis also proves the complexity conclusions in Proposition 2.

In Tables 5 and 6, it can be concluded that our sparse constrained formulation is $O(K)$ times faster for Step 1, and $O(K^2)$ times for Steps 2 and 3 than the dense unconstrained formulation in terms of the number of joints K. In total, our sparse constrained formulation scales linearly with respect to the number of joints instead of cubically as the dense unconstrained formulation.

Furthermore, in terms of the number of measure-

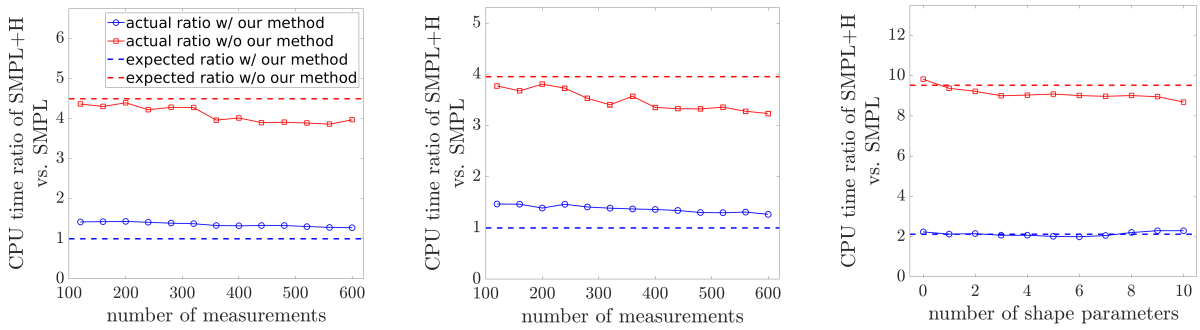


Figure 8: The CPU time ratio of the SMPL+H and SMPL models to compute the Gauss-Newton direction with (a) different numbers of measurements and no shape parameters, (b) different numbers of measurements and 10 shape parameters, and (c) different numbers of shape parameters. The SMPL and SMPL+H models have $K = 23$ and $K = 51$ joints, respectively. In Figs. 8 (a) to 8(c), the solid lines denote the actual CPU time ratio of the SMPL+H and SMPL models that is obtained from the experiments, whereas the dashed lines denote the expected CPU time ratio that is approximated from the complexity analysis in Tables 5 and 6. It can be seen the impact of the number of joints is around two orders of magnitude less on our method.

ments N, Tables 5 and 6 indicate that the complexity of our sparse constrained formulation is $O\big(N(9+P)^2\big)$ or $O(N)$, whereas that of the dense constrained formulation is $O\big(N(6+3K+P)^2\big)$ or $O(K^2N)$. This suggests that our sparse constrained formulation has the the number of joints K and measurements N decoupled in the computation, and as a result, is much more efficient to handle optimization problems with more measurements. Note that it is common in [5, 17, 24, 25, 28, 39] to introduce extra measurements to improve the estimation accuracy.

B. Ablation Studies

In addition to the results of ablation studies in the paper, we present a more complete analysis on the impact of the number of joints K, the number of measurements N, and the number of shape parameters P on the computation of the Gauss-Newton direction.

B.1. Experiments

As mentioned in the paper, the CPU time to compute the Gauss-Newton direction w/ and w/o our method is recorded for the SMPL and SMPL+H models in the following experiments.

Experiment 1. The number of shape parameters P is 0 and the number of measurements N increases from 120 to 600 for both of the SMPL and SMPL+H models.

Experiment 2. The number of shape parameters P is 10 and the number of measurements N increases from 120 to 600 for both of the SMPL and SMPL+H models.

Experiment 3. The number of shape parameters P increases from 0 to 10, and each joint of the SMPL and SMPL+H models is assigned with a 2D keypoint, a 3D keypoint, and a part orientation field as measurements.

B.2. Number of the Joints

The CPU time ratio of the SMPL+H and SMPL models to compute the Gauss-Newton direction is used as the metric to evaluate the impact of the number of joints K. Note that the SMPL and SMPL+H models have $K = 23$ and $K = 51$ joints, respectively. The CPU time ratio reflects the additional time induced as a result of the more joints on the SMPL+H model. The CPU time ratios of the three experiments are reported in Fig. 8 and discussed as follows:

1. In Experiment 1, there are no shape parameters and the computation of the Gauss-Newton direction is dominated by the number of measurements N. From Tables 5 and 6, it is known that our method has $O(N)$ complexity, which is not related with the number of joints K, and thus, the expected CPU time ratio with our method should be
$$\frac{1}{1} = 1.$$
In contrast, the CPU time without our method is approximately $O\left((3K+6)^2\right)$, which suggests an expected CPU time ratio of
$$\left(\frac{3\times 51+6}{3\times 23+6}\right)^2 = 4.49.$$
The numbers of 1 and 4.49 in the two equations above are consistent with the results in Fig. 8(a).

2. In Experiment 2, there are 10 shape parameters. However, the analysis is still similar to that of Experiment 1. From Tables 5 and 6, the expected CPU time ratio of the SMPL+H and SMPL models w/ and w/o our method should be around
$$\frac{1}{1} = 1$$

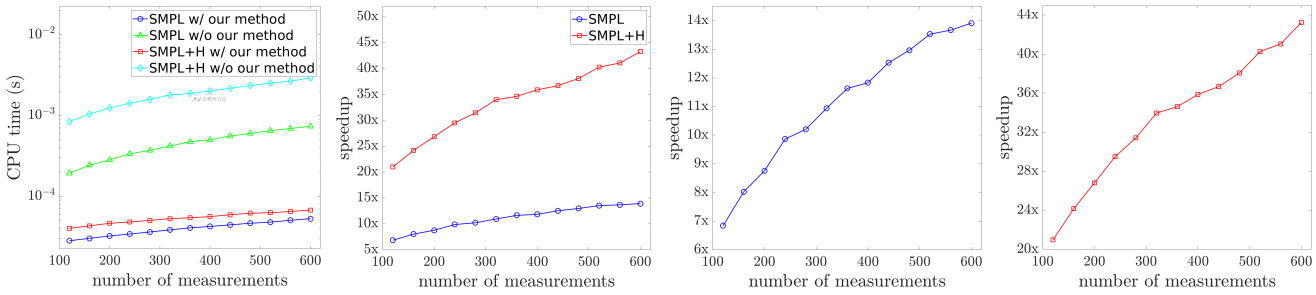


Figure 9: The computation of the Gauss-Newton direction with different numbers of measurements and no shape parameters. The results are (a) the CPU time with and without our method on the SMPL and SMPL+H models, and (b) the speedup of our method on the SMPL and SMPL+H models, and (c) the speed up of our method on the SMPL model, and (d) the speed up of our method on the SMPL+H model.

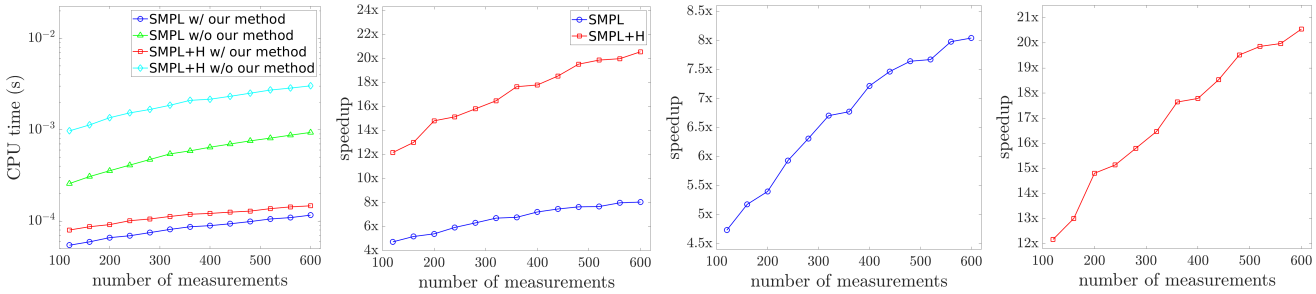


Figure 10: The computation of the Gauss-Newton direction with different numbers of measurements and 10 shape parameters. The results are (a) the CPU time with and without our method on the SMPL and SMPL+H models, and (b) the speedup of our method on the SMPL and SMPL+H models, and (c) the speed up of our method on the SMPL model, and (d) the speed up of our method on the SMPL+H model.

and

$$\left(\frac{3\times 51+6+10}{3\times 23+6+10}\right)^2 = 3.95,$$

respectively, which is consistent with the results in Fig. 8(b).

3. In Experiment 3, the number of measurements N is proportional to the number of joints of the SMPL and SMPL+H models. Then, as a result of Tables 5 and 6, the CPU time w/ and w/o our method to compute the Gauss-Newton direction should be around $O(K)$ and $O\left((3K+6)^3\right)$, respectively, and the corresponding expected CPU time can be also approximated by

$$\frac{51}{23} = 2.22$$

and

$$\left(\frac{3\times 51+6}{3\times 23+6}\right)^3 = 9.53,$$

which is consistent with the results in Fig. 8(c).

4. From Fig. 8 and the discussions above, it can be further concluded that the number of joints has around $O(K^2)$ times less impact on our method, which suggests that our sparse constrained formulation is more suitable for human models with more joints.

B.3. Number of the Measurements

The CPU time w/ and w/o our method to compute the Gauss-Newton direction and the corresponding speedup in Experiments 1 and 2 are reported in Figs. 9 and 10. It can be seen from Figs. 9 and 10 that our method has $4.73 \sim 13.91$x speedup on the SMPL model and a $12.17 \sim 43.24$x speedup on the SMPL+H model. Furthermore, no matter whether there are shape parameters or not, the speedup of our method is greater as the number of measurements increases, which means that our sparse constrained formulation is more efficient to solve optimization problems with more more measurements.

B.4. Number of the Shape Parameters

The CPU time w/ and w/o our method to compute the Gauss-Newton direction and the corresponding speedup in Experiment 3 are reported in Fig. 11. It can be seen from Fig. 11 that our method has a $4.92 \sim 7.78$x speedup on the SMPL model and a $18.63 \sim 34.18$x speedup on the SMPL+H model, which is consistent with the analysis that our sparse constrained formulation has better scalability on human models with more joints. On the SMPL+H model, the CPU time taken to compute the Gauss-Newton direction without our method is as many as 2.5 ms, which is difficult to be used in real time considering that most optimization methods need around $20 \sim 30$ iterations to converge. As

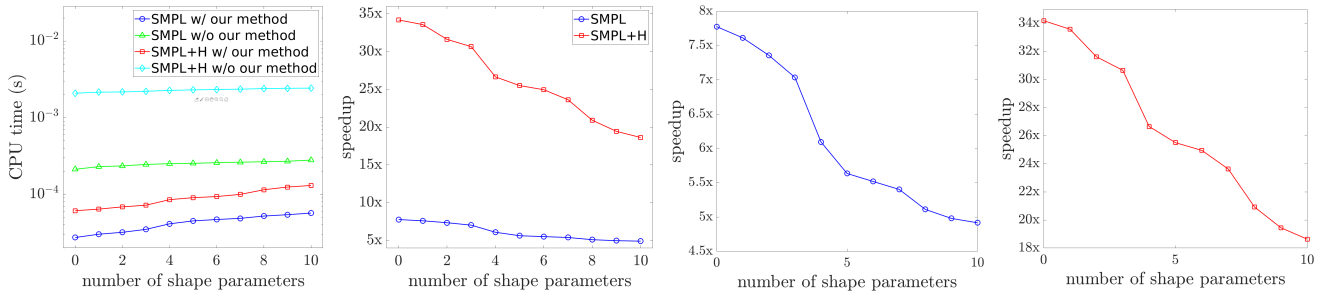


Figure 11: The computation of the Gauss-Newton direction with different number of shape parameters. The results are (a) the CPU time with and without our method on the SMPL and SMPL+H models, and (b) the speedup of our method on the SMPL and SMPL+H models, and (c) the speed up of our method on the SMPL model, and (d) the speed up of our method on the SMPL+H model.

a comparison, our method is significantly faster on both of the SMPL and SMPL+H models, for which the CPU time is $0.027 \sim 0.13$ ms. In particular, note that if there are no shape parameters, our method has a further acceleration of the computation—this has is important for real-time video tracking of 3D human pose and shape, in which the shape parameters that are estimated from the first few frames can be reused.

C. Qualitative Results

In this section, we present more qualitative comparisons with SPIN [15] and SMPLify [5] on the Human3.6M, MPI-INF-3DHP and 3DPW datasets. The results are shown in Figs. 12 to 14.

D. Real-Time Motion Capture Framework

D.1. Human Detection

The YOLOv4-CSP [4,38] is used for human detection to make a balance between accuracy and efficiency. The size of input images for YOLOv4-CSP is 512×512.

D.2. 2D Keypoint Estimation

The AlphaPose [8] is used for 2D keypoint estimation with 256×192 input images. The following datasets are used to train AlphaPose.

Human3.6M [7,10] is a popular dataset for 3D human pose estimation. Following the standard training-testing protocol in [29], we use subjects S1, S5-S8 for training.

MPI-INF-3DHP [23] is a multi-view markerless dataset with 8 training subjects and 6 test subjects. We use subjects S1-S8 that are downsampled to 10 FPS for training.

COCO [18] is a large-scale dataset for 2D joint detection. We use the COCO training datasets for training.

MPII [1] is a 2D human pose dataset that is extracted from online videos. We use the MPII training datasets for training.

D.3. 3D Keypoint Regression

In our real-time motion capture framework, we use a light-weight fully connected neural network for 2D-to-3D lifting. The 3D Keypoint regression network can be regarded as a modification of VideoPose3D [30]. From the 3D keypoint regression network, we further obtain the part orientation field [39] for each body part. We use the training datasets of Human3.6M [10] and MPI-INF-3DHP [23] that are downsampled to 10 FPS to train the 3D keypoint regression network.

E. Prior Loss of Joint States

We use the normalizing flow [14] to describe the joint state prior loss $\mathtt{E}_{\boldsymbol{\Omega},i}$. The normalizing flow is trained on the AMASS dataset [21] and has the structure of FC6 $\rightarrow$ PReLU $\rightarrow$ FC6 $\rightarrow$ PReLU $\rightarrow$ FC6 $\rightarrow$ PReLU $\rightarrow$ FC6 $\rightarrow$ PReLU $\rightarrow$ FC6 whose input is the 6D representation of rotation. We remark that the normalizing flow structure above to learn admissible joint states is inspired by the work of [40].

F. Implementation

F.1. Overview

While originally designed for 3D human pose and shape estimation, we emphasize that our method can be extended to any types of articulated tracking problems in computer vision and robotics [35]. The only requirement is that the objective can be written as

$$\mathtt{E} = \sum_{0 \leq i \leq K} \frac{1}{2} \|\mathbf{r}_i(\mathbf{T}_i, \boldsymbol{\Omega}_i, \boldsymbol{\beta})\|^2, \tag{59}$$

in which K is the number joints, $\mathbf{T}_i$ is the pose of body part i, $\boldsymbol{\Omega}_i$ is the joint state and $\boldsymbol{\beta}$ is the shape parameters. Empirically, such a requirement can be satisfied with ease, e.g., we might assume that the keypoints selected to calculate the losses are rigidly attached to a single body part. As a matter of fact, as long as the objective is in the form of Eq. (59),

Figure 12: Qualitative comparisons of our method (second row in pink), SPIN [15] (third row in gray), and SMPLify [5] (fourth row in purple) on the Human3.6M dataset.

the steps to compute the Gauss-Newton direction in Table 4 and the complexity analysis in Tables 5 and 6 hold as well. Thus, there are no difficulties to implement our method on practical articulated tracking problems.

F.2. Extract $\mathcal{S}_i$ and l_i from the SMPL Model

At the rest pose of the SMPL model [20], it is known that the joint positions linearly depend on the vertex positions, and the vertex positions also linearly depend on the shape parameters $\boldsymbol{\beta} \in \mathbb{R}^P$. Thus, we conclude that the joint positions $\bar{\mathbf{t}}_i \in \mathbb{R}^3$ at the rest pose linearly depend on the shape parameters, i.e., there exists $\mathcal{J}_i \in \mathbb{R}^{3\times P}$ and $\mathbf{c}_i \in \mathbb{R}^3$ in the SMPL model such that $\bar{\mathbf{t}}_i$ at the rest pose takes the form of

$$\bar{\mathbf{t}}_i = \mathcal{J}_i \cdot \boldsymbol{\beta} + \mathbf{c}_i. \tag{60}$$

Note that joint position $\mathbf{t}_i \in \mathbb{R}^3$ is also the translation of pose $\mathbf{T}_i = \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3)$ where $\mathbf{R}_i \in SO(3)$ is the rotation. Moreover, the relative joint position $\Delta\bar{\mathbf{t}}_i \in \mathbb{R}^3$ between any connected body parts is constant, and thus, we obtain $\Delta\bar{\mathbf{t}}_i = \bar{\mathbf{t}}_i - \bar{\mathbf{t}}_{\mathrm{par}(i)}$, in which $\mathrm{par}(i)$ denotes the index of the parent of body part i. Then, joint position $\mathbf{t}_i \in \mathbb{R}^3$ at any poses satisfies

$$\mathbf{t}_i = \mathbf{R}_{\mathrm{par}(i)}\Delta\bar{\mathbf{t}} + \mathbf{t}_{\mathrm{par}(i)} = \mathbf{R}_{\mathrm{par}(i)}\left(\bar{\mathbf{t}}_i - \bar{\mathbf{t}}_{\mathrm{par}(i)}\right) + \mathbf{t}_{\mathrm{par}(i)}. \tag{61}$$

In the equation above, $\mathbf{R}_{\mathrm{par}(i)}$ is rotation of pose $\mathbf{T}_{\mathrm{par}(i)} \in SE(3)$. Substituting Eq. (60) into Eq. (61) to cancel out $\bar{\mathbf{t}}_i$ and $\bar{\mathbf{t}}_{\mathrm{par}(i)}$, we obtain

$$\mathbf{t}_i = \mathbf{R}_{\mathrm{par}(i)}\left(\mathcal{S}_i \cdot \boldsymbol{\beta} + \mathbf{l}_i\right) + \mathbf{t}_{\mathrm{par}(i)}, \tag{62}$$

in which

$$\mathcal{S}_i = \mathcal{J}_i - \mathcal{J}_{\mathrm{par}(i)} \in \mathbb{R}^{3\times P} \tag{63}$$

and

$$\mathbf{l}_i = \mathbf{c}_i - \mathbf{c}_{\mathrm{par}(i)} \in \mathbb{R}^3. \tag{64}$$

It is immediate to show that $\mathcal{S}_i \cdot \boldsymbol{\beta} + \mathbf{l}_i$ is the relative joint position between body parts i and $\mathrm{par}(i)$, and thus, the corresponding relative pose $\mathbf{T}_{\mathrm{par}(i),i}$ is

$$\mathbf{T}_{\mathrm{par}(i),i} \triangleq \begin{bmatrix} \boldsymbol{\Omega}_i & \mathcal{S}_i \cdot \boldsymbol{\beta} + \mathbf{l}_i \\ \mathbf{0} & 1 \end{bmatrix}, \tag{65}$$

in which $\boldsymbol{\Omega}_i \in SO(3)$ is the state of joint i.

Figure 13: Qualitative comparisons of our method (second row in pink), SPIN [15] (third row in gray), and SMPLify [5] (fourth row in purple) on the MPI-INF-3DHP dataset.

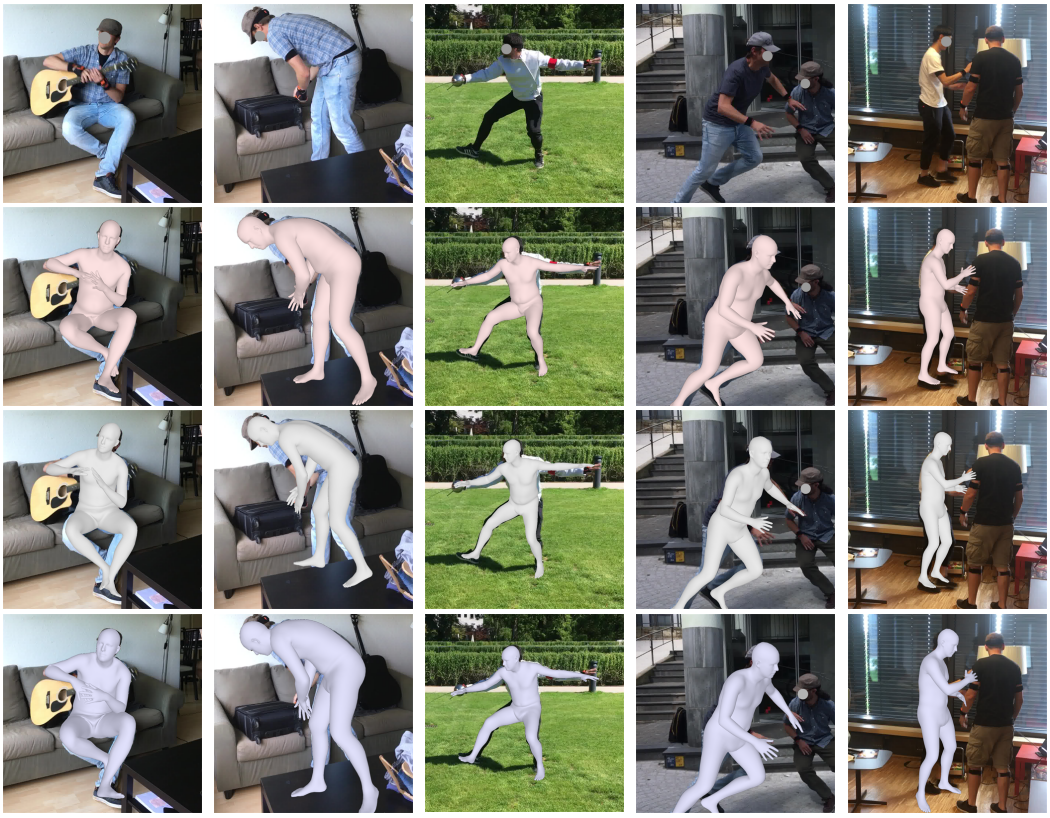


Figure 14: Qualitative comparisons of our method (second row in pink), SPIN [15] (third row in gray), and SMPLify [5] (fourth row in purple) on the MPI-INF-3DHP dataset.

References

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2D human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2014. 17

[2] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. SCAPE: Shape completion and animation of people. *ACM Trans. Graphics*, 24(3):408–416, July 2005. 2

[3] Anurag Arnab, Carl Doersch, and Andrew Zisserman. Exploiting temporal context for 3D human pose estimation in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 6, 7

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 5, 6, 17

[5] 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 *European conference on computer vision (ECCV)*, 2016. 1, 2, 3, 4, 6, 7, 15, 17, 18, 19

[6] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: Realtime multi-person 2D pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):172–186, 2019. 2

[7] Cristian Sminchisescu Catalin Ionescu, Fuxin Li. Latent structured models for human pose estimation. In *Procedings of the IEEE/CVF International Conference on Computer Vision*, pages 2220–2227, 2011. 5, 17

[8] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. RMPE: Regional multi-person pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2334–2343, 2017. 2, 5, 17

[9] Yinghao Huang, Federica Bogo, Christoph Lassner, Angjoo Kanazawa, Peter V Gehler, Javier Romero, Ijaz Akhter, and Michael J Black. Towards accurate markerless human shape and pose estimation over time. In *Proceedings of the International Conference on 3D Vision*, pages 421–430, 2017. 2

[10] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, 2013. 2, 5, 17

[11] Hanbyul Joo, Tomas Simon, and Yaser Sheikh. Total Capture: A 3D deformation model for tracking faces, hands, and bodies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8320–8329, 2018. 2

[12] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7122–7131, 2018. 1, 2, 6, 7

[13] Angjoo Kanazawa, Jason Y Zhang, Panna Felsen, and Jitendra Malik. Learning 3D human dynamics from video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5614–5623, 2019. 6, 7

[14] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 17

[15] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *Proceedings of the IEEE/CVF Conference on Computer Vision*, 2019. 1, 2, 6, 7, 17, 18, 19

[16] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. Convolutional mesh regression for single-image human shape reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 2, 6, 7

[17] Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J. Black, and Peter V. Gehler. Unite the People: Closing the loop between 3D and 2D human representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2, 4, 6, 15

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 17

[19] Matthew Loper, Naureen Mahmood, and Michael J Black. MoSh: Motion and shape capture from sparse markers. *ACM Transactions on Graphics (TOG)*, 2014. 1

[20] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, Oct. 2015. 2, 7, 18

[21] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5442–5451, Oct. 2019. 2, 3, 17

[22] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3D human pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2640–2649, 2017. 2, 6

[23] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. Monocular 3D human pose estimation in the wild using improved cnn supervision. In *Proceedings of the International Conference on 3D Vision*. IEEE, 2017. 2, 5, 17

[24] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Mohamed Elgharib, Pascal Fua, Hans-Peter Seidel, Helge Rhodin, Gerard Pons-Moll, and Christian Theobalt. XNect: Real-time multi-person 3D motion capture with a single RGB camera. *ACM Transactions on Graphics (TOG)*, 39(4):82–1, 2020. 1, 2, 3, 4, 5, 7, 15

[25] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. VNect: Real-time 3D human pose estimation with a single RGB camera. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017. 1, 2, 3, 4, 5, 6, 7, 15

[26] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006. 1

[27] Ahmed A A Osman, Timo Bolkart, and Michael J. Black. STAR: A spare trained articulated human body regressor. In *European Conference on Computer Vision (ECCV)*, 2020. 2, 8

[28] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 2, 4, 15

[29] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. Coarse-to-fine volumetric prediction for single-image 3D human pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7025–7034, 2017. 2, 5, 6, 17

[30] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3D human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 2, 5, 6, 17

[31] Gregory Rogez and Cordelia Schmid. MoCap-guided data augmentation for 3D pose estimation in the wild. *Advances in Neural Information Processing Systems*, 29:3108–3116, 2016. 2, 6

[32] Gregory Rogez, Philippe Weinzaepfel, and Cordelia Schmid. LCR-net: Localization-classification-regression for human pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017. 2, 6

[33] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics*, 36(6):1–17, 2017. 7

[34] Yu Rong, Takaaki Shiratori, and Hanbyul Joo. Frankmocap: Fast monocular 3D hand and body motion capture by regression and integration. *arXiv preprint arXiv:2008.08324*, 2020. 2

[35] Tanner Schmidt, Richard Newcombe, and Dieter Fox. DART: dense articulated real-time tracking with consumer depth cameras. *Autonomous Robots*, 39(3):239–258, 2015. 17

[36] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5693–5703, 2019. 2

[37] Timo von Marcard, Roberto Henschel, Michael Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3D human pose in the wild using IMUs and a moving camera. In *European Conference on Computer Vision (ECCV)*, sep 2018. 1, 2, 5

[38] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-YOLOv4: Scaling cross stage partial network. *arXiv preprint arXiv:2011.08036*, 2020. 5, 6, 17

[39] Donglai Xiang, Hanbyul Joo, and Yaser Sheikh. Monocular total capture: Posing face, body, and hands in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10965–10974, 2019. 1, 2, 3, 4, 5, 6, 15, 17

[40] Andrei Zanfir, Eduard Gabriel Bazavan, Hongyi Xu, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Weakly supervised 3d human pose and shape reconstruction with normalizing flows. In *European Conference on Computer Vision*, 2020. 2, 17