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:

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

and

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

subject to

$$\begin{aligned}\mathbf{T}_i =& \mathtt{F}_i(\mathbf{T}_{\mathrm{par}(i)},\,\boldsymbol{\beta}_{\mathrm{par}(i)},\,\boldsymbol{\Omega}_i)\\ \triangleq& \mathbf{T}_{\mathrm{par}(i)}\begin{bmatrix}\boldsymbol{\Omega}_i & \mathcal{S}_i\cdot\boldsymbol{\beta}_{\mathrm{par}(i)}+\mathbf{l}_i\\ \mathbf{0} & 1\end{bmatrix},\end{aligned} \tag{3a}$$

$$\boldsymbol{\beta}_i = \boldsymbol{\beta}_{\mathrm{par}(i)}. \tag{3b}$$

In Eqs. (1) and (2), $\mathbf{T}_i \in SE(3)$ 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. (1) and (2) are the dense unconstrained and sparse constrained formulations, respectively, for 3D human pose and shape estimation that are defined in the paper.

From Eq. (3b), if we let $\boldsymbol{\beta}_0 = \boldsymbol{\beta}$, then, $\boldsymbol{\beta}_i = \boldsymbol{\beta}$ for all $i = 1, \cdots, K$. Thus, Eq. (2) is reduced to

$$\min_{\{\mathbf{T}_i,\,\boldsymbol{\Omega}_i,\,\boldsymbol{\beta}_i\}_{i=0}^{K}} \mathtt{E} = \sum_{i=0}^{K} \frac{1}{2}\|\mathtt{r}_i(\mathbf{T}_i,\,\boldsymbol{\Omega}_i,\,\boldsymbol{\beta})\|^2 \tag{4}$$

subject to

$$\begin{aligned}\mathbf{T}_i =& \mathtt{F}_i(\mathbf{T}_{\mathrm{par}(i)},\,\boldsymbol{\beta},\,\boldsymbol{\Omega}_i)\\ =& \mathbf{T}_{\mathrm{par}(i)}\begin{bmatrix}\boldsymbol{\Omega}_i & \mathcal{S}_i\cdot\boldsymbol{\beta}+\mathbf{l}_i\\ \mathbf{0} & 1\end{bmatrix}.\end{aligned} \tag{5}$$

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. (5) 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.,

$$\mathbf{T}_i \triangleq \mathbf{T}_i\,(\mathbf{T}_0,\,\boldsymbol{\Omega},\,\boldsymbol{\beta}) \tag{6}$$

If we use Eq. (6) to cancel out non-root rigid body transformations $\mathbf{T}_i$ ($1 \le i \le K$), then, each $\mathtt{r}_i(\cdot)$ in Eq. (4) 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. (1). Therefore, it can be concluded that Eqs. (1) and (2) 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. (1) and (2) can be rewritten as

$$\min_{\mathbf{x}} \mathtt{E} = \sum_{i=0}^{K} \frac{1}{2} \|\mathtt{r}_i(\mathbf{x})\|^2, \tag{7}$$

and

$$\min_{\{\mathbf{x}_i,\, \boldsymbol{\Omega}_i\}_{i=0}^K} \mathtt{E} = \sum_{i=0}^{K} \frac{1}{2} \|\mathtt{r}_i(\mathbf{x}_i,\, \boldsymbol{\Omega}_i)\|^2 \tag{8}$$

subject to

$$\mathbf{x}_i = \begin{bmatrix} \mathtt{F}_i(\mathbf{x}_{\mathrm{par}(i)},\, \boldsymbol{\Omega}_i) \\ \boldsymbol{\beta}_{\mathrm{par}(i)} \end{bmatrix}, \tag{9}$$

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 1, which will be frequently used in the rest of this proof.

A.2.2 The Equivalence of the Gauss-Newton Direction

In Table 1, 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. (10) and (12), are equivalent. From Eq. (6), 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. (10) can be computed using $\mathbf{J}_{i,1}$ and $\mathbf{J}_{i,2}$ in Eq. (12):

$$\mathbf{J}_i \Delta\mathbf{x} = \mathbf{J}_{i,1} \begin{bmatrix} \frac{\partial \mathbf{T}_i}{\partial \mathbf{T}_0} \Delta\mathbf{T}_0 + \frac{\partial \mathbf{T}_i}{\partial \boldsymbol{\Omega}} \Delta\boldsymbol{\Omega} + \frac{\partial \mathbf{T}_i}{\partial \boldsymbol{\beta}} \Delta\boldsymbol{\beta} \\ \boldsymbol{\beta} \end{bmatrix} + \mathbf{J}_{i,2} \Delta\boldsymbol{\Omega}_i. \tag{21}$$

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. (21) are obtained by the recursive implementation of Eq. (13). Therefore, it can be concluded that Eqs. (10) and (12) 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. (19)

In Table 1, 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. (19) in **Step 3**, which is nontrivial. In this subsection, we derive a dynamic programming algorithm that exploits the sparsity and constraints of Eq. (13) 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 $j \in \{i\} \cup \mathrm{des}(i)$

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

subject to

$$\Delta\mathbf{x}_j = \mathbf{A}_j \Delta\mathbf{x}_{\mathrm{par}(j)} + \mathbf{B}_j \Delta\boldsymbol{\Omega}_j,\ \forall j \in \{i\} \cup \mathrm{des}(i), \tag{23}$$

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. (22) that $\mathcal{E}_i(\cdot)$ can be reduced to an optimization problem of $\Delta\mathbf{x}_i$ and $\Delta\boldsymbol{\Omega}_i$

$$\begin{aligned} \mathcal{E}_i(\Delta\mathbf{x}_{\mathrm{par}(i)}) \triangleq \min_{\Delta\mathbf{x}_i,\, \Delta\boldsymbol{\Omega}_i} \Big[\frac{1}{2} \Delta\mathbf{x}_i^\top \mathbf{H}_{i,11} \Delta\mathbf{x}_i + \\ \Delta\boldsymbol{\Omega}_i^\top \mathbf{H}_{i,21} \Delta\mathbf{x}_i + \frac{1}{2} \Delta\boldsymbol{\Omega}_i^\top \mathbf{H}_{i,22} \Delta\boldsymbol{\Omega}_i + \mathbf{g}_{i,1}^\top \Delta\mathbf{x}_i + \\ \mathbf{g}_{i,2}^\top \Delta\boldsymbol{\Omega}_i + \sum_{j \in \mathrm{chd}(i)} \mathcal{E}_j(\Delta\mathbf{x}_i) \Big] \end{aligned} \tag{24}$$

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. (24) is an intermediate procedure that is essential for our dynamic programming algorithm.

	Dense Unconstrained Formulation	Sparse Constrained Formulation
Step 1	The linearization of Eq. (7) results in $$\min_{\Delta\mathbf{x}} \Delta\mathtt{E} = \sum_{i=0}^{K} \frac{1}{2}\|\mathbf{J}_i\Delta\mathbf{x} + \mathtt{r}_i\|^2, \quad (10)$$ 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 $$\mathbf{J}_i \triangleq \begin{bmatrix} \frac{\partial \mathtt{r}_i}{\partial \mathbf{T}_0} & \frac{\partial \mathtt{r}_i}{\partial \boldsymbol{\Omega}} & \frac{\partial \mathtt{r}_i}{\partial \boldsymbol{\beta}} \end{bmatrix} \in \mathbb{R}^{N_i \times (6+3K+P)} \quad (11)$$ is the Jacobian of $\mathtt{r}_i(\cdot)$, and $\mathtt{r}_i \in \mathbb{R}^{N_i}$ is the residue.	The linearization of Eq. (8) results in $$\min_{\{\Delta\mathbf{x}_i, \Delta\boldsymbol{\Omega}_i\}_{i=0}^{K}} \Delta\mathtt{E} = \sum_{i=0}^{K} \frac{1}{2}\|\mathbf{J}_{i,1}\Delta\mathbf{x}_i + \mathbf{J}_{i,2}\Delta\boldsymbol{\Omega}_i + \mathtt{r}_i\|^2 \quad (12)$$ subject to $$\Delta\mathbf{x}_i = \mathbf{A}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{B}_i\Delta\boldsymbol{\Omega}_i, \quad (13)$$ 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 $$\mathbf{J}_{i,1} \triangleq \begin{bmatrix} \frac{\partial \mathtt{r}_i}{\partial \mathbf{T}_i} & \frac{\partial \mathtt{r}_i}{\partial \boldsymbol{\beta}_i} \end{bmatrix} \in \mathbb{R}^{N_i \times (6+P)} \quad (14)$$ and $$\mathbf{J}_{i,2} \triangleq \frac{\partial \mathtt{r}_i}{\partial \boldsymbol{\Omega}_i} \in \mathbb{R}^{N_i \times 3} \quad (15)$$ are the Jacobians of $\mathtt{r}_i(\cdot)$, and $$\mathbf{A}_i \triangleq \begin{bmatrix} \frac{\partial \mathtt{F}_i}{\partial \mathbf{T}_{\mathrm{par}(i)}} & \frac{\partial \mathtt{F}_i}{\partial \boldsymbol{\beta}_{\mathrm{par}(i)}} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \in \mathbb{R}^{(6+P)\times(6+P)} \quad (16)$$ and $$\mathbf{B}_i \triangleq \begin{bmatrix} \frac{\partial \mathtt{F}_i}{\partial \boldsymbol{\Omega}_i} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(6+P)\times 3} \quad (17)$$ are the partial derivatives of Eq. (13), and $\mathtt{r}_i \in \mathbb{R}^{N_i}$ is the residue.
Step 2	Reformulate Eq. (10) as $$\min_{\Delta\mathbf{x}} \Delta\mathtt{E} = \frac{1}{2}\Delta\mathbf{x}^\top\mathbf{H}\Delta\mathbf{x} + \mathbf{g}^\top\Delta\mathbf{x} \quad (18)$$ 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. (12) as $$\min_{\{\Delta\mathbf{x}_i, \Delta\boldsymbol{\Omega}_i\}_{i=0}^{K}} \Delta\mathtt{E} = \sum_{i=0}^{K}\Big[\frac{1}{2}\Delta\mathbf{x}_i^\top\mathbf{H}_{i,11}\Delta\mathbf{x}_i + \Delta\boldsymbol{\Omega}_i^\top\mathbf{H}_{i,21}\Delta\mathbf{x}_i + \frac{1}{2}\Delta\boldsymbol{\Omega}_i^\top\mathbf{H}_{i,22}\Delta\boldsymbol{\Omega}_i + \mathbf{g}_{i,1}^\top\Delta\mathbf{x}_i + \mathbf{g}_{i,2}^\top\Delta\boldsymbol{\Omega}_i\Big] \quad (19)$$ 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. (18), which has a closed-form solution $$\Delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{g}. \quad (20)$$	Compute the Gauss-Newton direction from Eq. (19), which can be exactly solved by Algorithm 1.

Table 1: Steps to compute the Gauss-Newton direction for the dense unconstrained and sparse constrained formulations.

Algorithm 1 Solve Eq. (19) 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

$$\mathcal{E}_j(\Delta\mathbf{x}_i) = \frac{1}{2}\Delta\mathbf{x}_i^\top \mathbf{M}_j\Delta\mathbf{x}_i + \mathbf{m}_j^\top\Delta\mathbf{x}_i + \Delta\mathrm{E}_j. \tag{25}$$

Applying Eq. (25) to Eq. (24), we obtain

$$\begin{aligned}\mathcal{E}_i(\Delta\mathbf{x}_{\mathrm{par}(i)}) = \min_{\Delta\mathbf{x}_i,\,\Delta\boldsymbol{\Omega}_i} \frac{1}{2}\Delta\mathbf{x}_i\mathbf{N}_{i,11}\Delta\mathbf{x}_i + \\ \Delta\boldsymbol{\Omega}_i^\top\mathbf{N}_{i,21}\Delta\mathbf{x}_i + \frac{1}{2}\Delta\boldsymbol{\Omega}_i^\top\mathbf{N}_{i,22}\Delta\boldsymbol{\Omega}_i + \\ \mathbf{n}_{i,1}^\top\Delta\mathbf{x}_i + \mathbf{n}_{i,2}^\top\Delta\boldsymbol{\Omega}_i + \Delta\bar{\mathrm{E}}_i\end{aligned} \tag{26}$$

subject to

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

in which

$$\mathbf{N}_{i,11} = \mathbf{H}_{i,11} + \sum_{j\in\mathrm{chd}(i)} \mathbf{M}_i, \tag{27a}$$

$$\mathbf{N}_{i,21} = \mathbf{H}_{i,21}, \tag{27b}$$

$$\mathbf{N}_{i,22} = \mathbf{H}_{i,22}, \tag{27c}$$

$$\mathbf{n}_{i,1} = \mathbf{g}_{i,1} + \sum_{j\in\mathrm{chd}(i)} \mathbf{m}_j, \tag{27d}$$

$$\mathbf{n}_{i,2} = \mathbf{g}_{i,2}, \tag{27e}$$

$$\Delta\bar{\mathrm{E}}_i = \sum_{j\in\mathrm{chd}(i)} \Delta\mathrm{E}_j. \tag{27f}$$

Substitute Eq. (13) into Eq. (26) to cancel out $\Delta\mathbf{x}_i$ and simplify the resulting equation to an unconstrained optimization problem on $\Delta\boldsymbol{\Omega}_i \in \mathbb{R}^3$:

$$\begin{aligned}\mathcal{E}_i(\Delta\mathbf{x}_{\mathrm{par}(i)}) = \min_{\Delta\boldsymbol{\Omega}_i} \frac{1}{2}\Delta\mathbf{x}_{\mathrm{par}(i)}\mathbf{Q}_{i,11}\Delta\mathbf{x}_{\mathrm{par}(i)} + \\ \Delta\boldsymbol{\Omega}_i^\top\mathbf{Q}_{i,21}\Delta\mathbf{x}_{\mathrm{par}(i)} + \frac{1}{2}\Delta\boldsymbol{\Omega}_i^\top\mathbf{Q}_{i,22}\Delta\boldsymbol{\Omega}_i + \\ \mathbf{q}_{i,1}^\top\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{q}_{i,2}^\top\Delta\boldsymbol{\Omega}_i + \Delta\bar{\mathrm{E}}_i,\end{aligned} \tag{28}$$

in which

$$\mathbf{Q}_{i,11} = \mathbf{A}_i^\top\mathbf{N}_{i,11}\mathbf{A}_i, \tag{29a}$$

$$\mathbf{Q}_{i,21} = \mathbf{B}_i^\top\mathbf{N}_{i,11}\mathbf{A}_i + \mathbf{N}_{i,21}\mathbf{A}_i, \tag{29b}$$

$$\begin{aligned}\mathbf{Q}_{i,22} = \mathbf{B}_i^\top\mathbf{N}_{i,11}\mathbf{B}_i + \mathbf{N}_{i,21}\mathbf{B}_i + \\ \mathbf{B}_i^\top\mathbf{N}_{i,21}^\top + \mathbf{N}_{i,22},\end{aligned} \tag{29c}$$

$$\mathbf{q}_{i,1} = \mathbf{A}_i^\top\mathbf{n}_{i,1}, \tag{29d}$$

$$\mathbf{q}_{i,2} = \mathbf{B}_i^\top\mathbf{n}_{i,1} + \mathbf{n}_{i,2}. \tag{29e}$$

It is obvious that Eq. (28) has a closed-form solution

$$\Delta\boldsymbol{\Omega}_i = \mathbf{K}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \mathbf{k}_i, \tag{30}$$

in which

$$\mathbf{K}_i = -\mathbf{Q}_{i,22}^{-1}\mathbf{Q}_{i,21} \quad\text{and}\quad \mathbf{k}_i = -\mathbf{Q}_{i,22}^{-1}\mathbf{q}_{i,2}. \tag{31}$$

If we use Eq. (30) to eliminate $\Delta\boldsymbol{\Omega}_i$ in Eq. (28), 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

$$\begin{aligned}\mathcal{E}_i(\Delta\mathbf{x}_{\mathrm{par}(i)}) = \frac{1}{2}\Delta\mathbf{x}_{\mathrm{par}(i)}^\top\mathbf{M}_i\Delta\mathbf{x}_{\mathrm{par}(i)} + \\ \mathbf{m}_i^\top\Delta\mathbf{x}_{\mathrm{par}(i)} + \Delta\mathrm{E}_i,\end{aligned} \tag{32}$$

in which

$$\mathbf{M}_i = \mathbf{Q}_{i,11} - \mathbf{Q}_{i,21}^\top\mathbf{Q}_{i,22}^{-1}\mathbf{Q}_{i,21}, \tag{33a}$$

	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 2: 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).

$$\mathbf{m}_i = \mathbf{q}_{i,1} - \mathbf{Q}_{i,21}^\top \mathbf{Q}_{i,22}^{-1}\mathbf{q}_{i,2}, \tag{33b}$$

$$\Delta \mathrm{E}_i = \Delta\bar{\mathrm{E}}_i - \frac{1}{2}\mathbf{q}_{i,2}^\top \mathbf{Q}_{i,22}^{-1}\mathbf{q}_{i,2}. \tag{33c}$$

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. (25) 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. (32).

In the kinematic tree, a body part i at the leaf node has no children, for which Eq. (27) 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. (27), (29) and (33) 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. (32) is known for each $i \in \mathrm{chd}(0)$, Eq. (19) 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. (32), the equation above is equivalent to

$$\min_{\Delta\mathbf{x}_0} \frac{1}{2}\Delta\mathbf{x}_0^\top \mathbf{M}_0\Delta\mathbf{x}_0 + \mathbf{m}_0^\top \mathbf{x}_0 + \Delta\bar{\mathrm{E}}_0 \tag{34}$$

in which

$$\mathbf{M}_0 = \mathbf{H}_{0,11} + \sum_{j\in\mathrm{chd}(0)} \mathbf{M}_j, \tag{35a}$$

$$\mathbf{m}_0 = \mathbf{g}_{0,1} + \sum_{j\in\mathrm{chd}(0)} \mathbf{m}_j, \tag{35b}$$

$$\Delta\bar{\mathrm{E}}_0 = \sum_{i\in\mathrm{chd}(0)} \Delta\mathrm{E}_i. \tag{35c}$$

It is straightforward to show that

$$\Delta\mathbf{x}_0 = -\mathbf{M}_0^{-1}\mathbf{m}_0 \tag{36}$$

solves Eq. (34) with

$$\Delta\mathrm{E}_0 = \Delta\bar{\mathrm{E}}_0 - \frac{1}{2}\mathbf{m}_0^\top \mathbf{M}_0^{-1}\mathbf{m}_0 \tag{37}$$

to be the expected cost reduction as well as the minimum objective value of Eq. (19).

At last, we recursively compute $\{\Delta\mathbf{x}_i, \Delta\mathbf{\Omega}_i\}_{i=1}^K$ using Eqs. (13), (30) and (31) 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. (19) 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.

	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. (11) 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. (16) and (17) for each $i = 0, \cdots, K$. Note that the bottom of $\mathbf{A}_i$ and $\mathbf{B}_i$ in Eqs. (16) and (17) 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. (14) and (15) 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. (11) 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. (18).	(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. (19) 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. (20).	(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. (16) and (17) 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 3: 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.

A.2.4 Complexity Analysis

In Table 2, we present a short summary of the computational complexities for each step to compute the Gauss-Newton direction, and in Table 3, we present a comprehensive analysis of the computational complexities that leads to results in Table 2. The analysis also proves the complexity conclusions in Proposition 2.

In Tables 2 and 3, it can be concluded that our sparse constrained formulation is $O(K)$ times faster for Step 1,

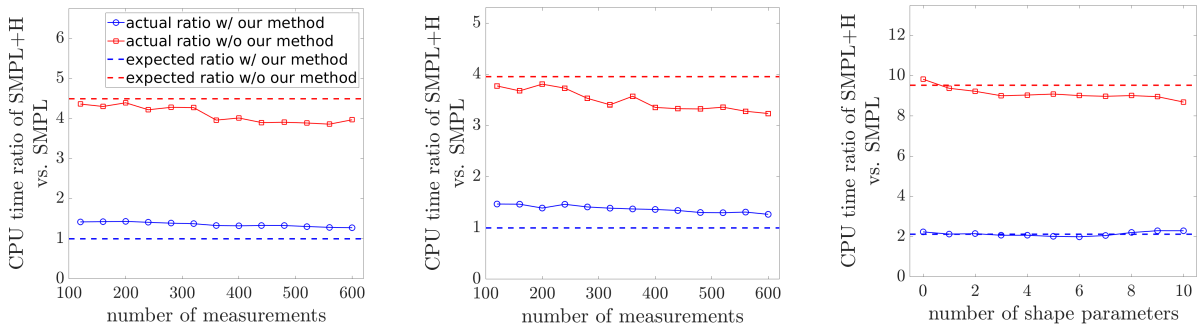

Figure 1: The CPU time ratio of the SMPL+H and SMPL models to compute the Gauss-Newton direction with (a) different numbers of measurements and no shape parameters, (b) different numbers of measurements and 10 shape parameters, and (c) different numbers of shape parameters. The SMPL and SMPL+H models have $K = 23$ and $K = 51$ joints, respectively. In Figs. 1 (a) to 1(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 2 and 3. It can be seen the impact of the number of joints is around two orders of magnitude less on our method.

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 measurements N, Tables 2 and 3 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 [3, 9, 14, 15, 16, 21] 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. 1 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 2 and 3, 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. 1(a).

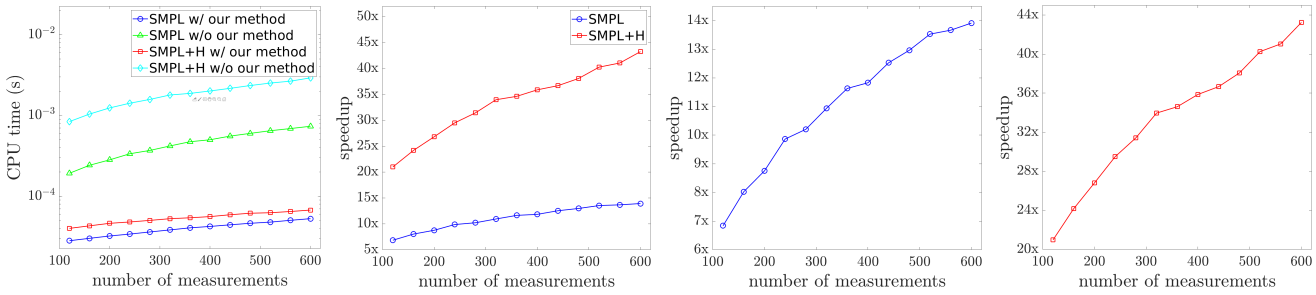

Figure 2: The computation of the Gauss-Newton direction with different numbers of measurements and no shape parameters. The results are (a) the CPU time with and without our method on the SMPL and SMPL+H models, and (b) the speedup of our method on the SMPL and SMPL+H models, and (c) the speed up of our method on the SMPL model, and (d) the speed up of our method on the SMPL+H model.

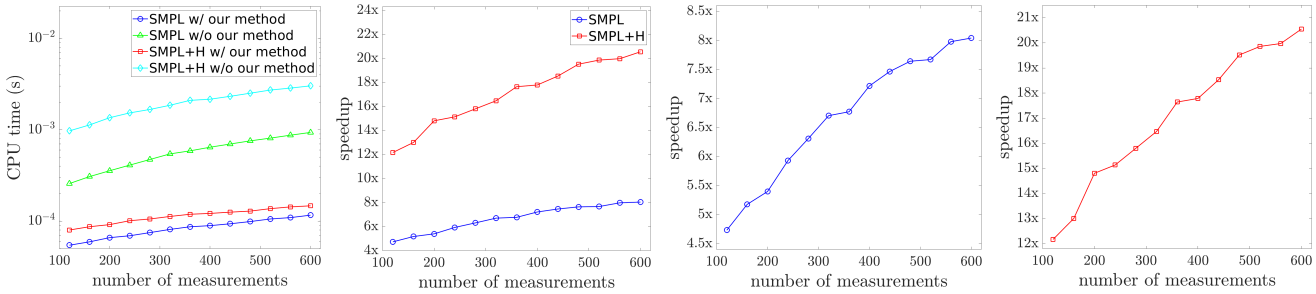

Figure 3: The computation of the Gauss-Newton direction with different numbers of measurements and 10 shape parameters. The results are (a) the CPU time with and without our method on the SMPL and SMPL+H models, and (b) the speedup of our method on the SMPL and SMPL+H models, and (c) the speed up of our method on the SMPL model, and (d) the speed up of our method on the SMPL+H model.

2. In Experiment 2, there are 10 shape parameters. However, the analysis is still similar to that of Experiment 1. From Tables 2 and 3, the expected CPU time ratio of the SMPL+H and SMPL models w/ and w/o our method should be around

$$\frac{1}{1} = 1$$

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. 1(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 2 and 3, 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. 1(c).

4. From Fig. 1 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. 2 and 3. It can be seen from Figs. 2 and 3 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. 4. It can be seen from Fig. 4 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

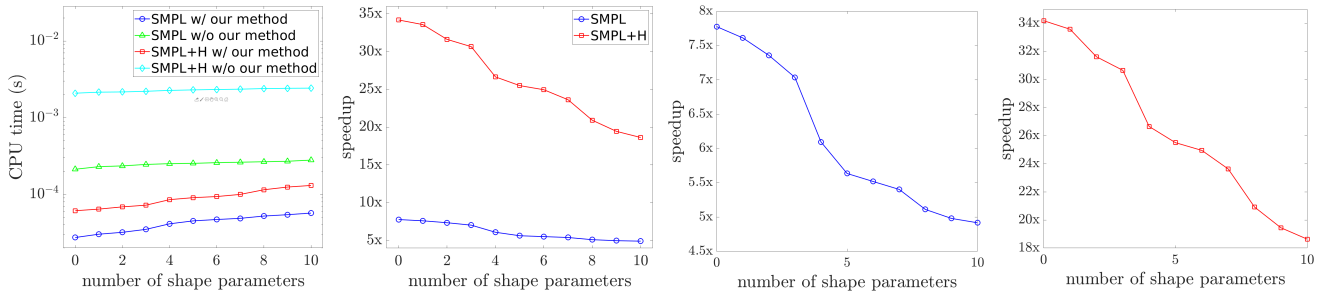

Figure 4: The computation of the Gauss-Newton direction with different number of shape parameters. The results are (a) the CPU time with and without our method on the SMPL and SMPL+H models, and (b) the speedup of our method on the SMPL and SMPL+H models, and (c) the speed up of our method on the SMPL model, and (d) the speed up of our method on the SMPL+H model.

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 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 [8] and SMPLify [3] on the Human3.6M, MPI-INF-3DHP and 3DPW datasets. The results are shown in Figs. 5 to 7.

D. Real-Time Motion Capture Framework

D.1. Human Detection

The YOLOv4-CSP [2, 20] 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 [5] is used for 2D keypoint estimation with 256×192 input images. The following datasets are used to train AlphaPose.

Human3.6M [4, 6] is a popular dataset for 3D human pose estimation. Following the standard training-testing protocol in [17], we use subjects S1, S5-S8 for training.

MPI-INF-3DHP [13] 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 [10] 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 [18]. From the 3D keypoint regression network, we further obtain the part orientation field [21] for each body part. We use the training datasets of Human3.6M [6] and MPI-INF-3DHP [13] that are downsampled to 10 FPS to train the 3D keypoint regression network.

E. Prior Loss of Joint States

We use the normalizing flow [7] to describe the joint state prior loss $\mathtt{E}_{\boldsymbol{\Omega},i}$. The normalizing flow is trained on the AMASS dataset [12] 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 [22].

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 [19]. The only requirement is that the objective can be written as

$$\mathtt{E} = \sum_{0 \leq i \leq K} \frac{1}{2} \|\mathbf{r}_i(\mathbf{T}_i, \boldsymbol{\Omega}_i, \boldsymbol{\beta})\|^2, \tag{38}$$

Figure 5: Qualitative comparisons of our method (second row in pink), SPIN [8] (third row in gray), and SMPLify [3] (fourth row in purple) on the Human3.6M dataset.

in which K is the number joints, $\mathbf{T}_i$ is the pose of body part i, $\mathbf{\Omega}_i$ 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. (38), the steps to compute the Gauss-Newton direction in Table 1 and the complexity analysis in Tables 2 and 3 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 [11], 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

$$\bar{\mathbf{t}}_i = \mathcal{J}_i \cdot \boldsymbol{\beta} + \mathbf{c}_i. \tag{39}$$

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

$$\mathbf{t}_i = \mathbf{R}_{\mathrm{par}(i)}\Delta\bar{\mathbf{t}} + \mathbf{t}_{\mathrm{par}(i)} = \mathbf{R}_{\mathrm{par}(i)}\left(\bar{\mathbf{t}}_i - \bar{\mathbf{t}}_{\mathrm{par}(i)}\right) + \mathbf{t}_{\mathrm{par}(i)}. \tag{40}$$

In the equation above, $\mathbf{R}_{\mathrm{par}(i)}$ is rotation of pose $\mathbf{T}_{\mathrm{par}(i)} \in SE(3)$. Substituting Eq. (39) into Eq. (40) to cancel out $\bar{\mathbf{t}}_i$ and $\bar{\mathbf{t}}_{\mathrm{par}(i)}$, we obtain

$$\mathbf{t}_i = \mathbf{R}_{\mathrm{par}(i)}\left(\mathcal{S}_i \cdot \boldsymbol{\beta} + \mathbf{l}_i\right) + \mathbf{t}_{\mathrm{par}(i)}, \tag{41}$$

in which

$$\mathcal{S}_i = \mathcal{J}_i - \mathcal{J}_{\mathrm{par}(i)} \in \mathbb{R}^{3\times P} \tag{42}$$

and

$$\mathbf{l}_i = \mathbf{c}_i - \mathbf{c}_{\mathrm{par}(i)} \in \mathbb{R}^3. \tag{43}$$

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

$$\mathbf{T}_{\mathrm{par}(i),i} \triangleq \begin{bmatrix} \mathbf{\Omega}_i & \mathcal{S}_i \cdot \boldsymbol{\beta} + \mathbf{l}_i \\ \mathbf{0} & 1 \end{bmatrix}, \tag{44}$$

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

Figure 6: Qualitative comparisons of our method (second row in pink), SPIN [8] (third row in gray), and SMPLify [3] (fourth row in purple) on the MPI-INF-3DHP dataset.

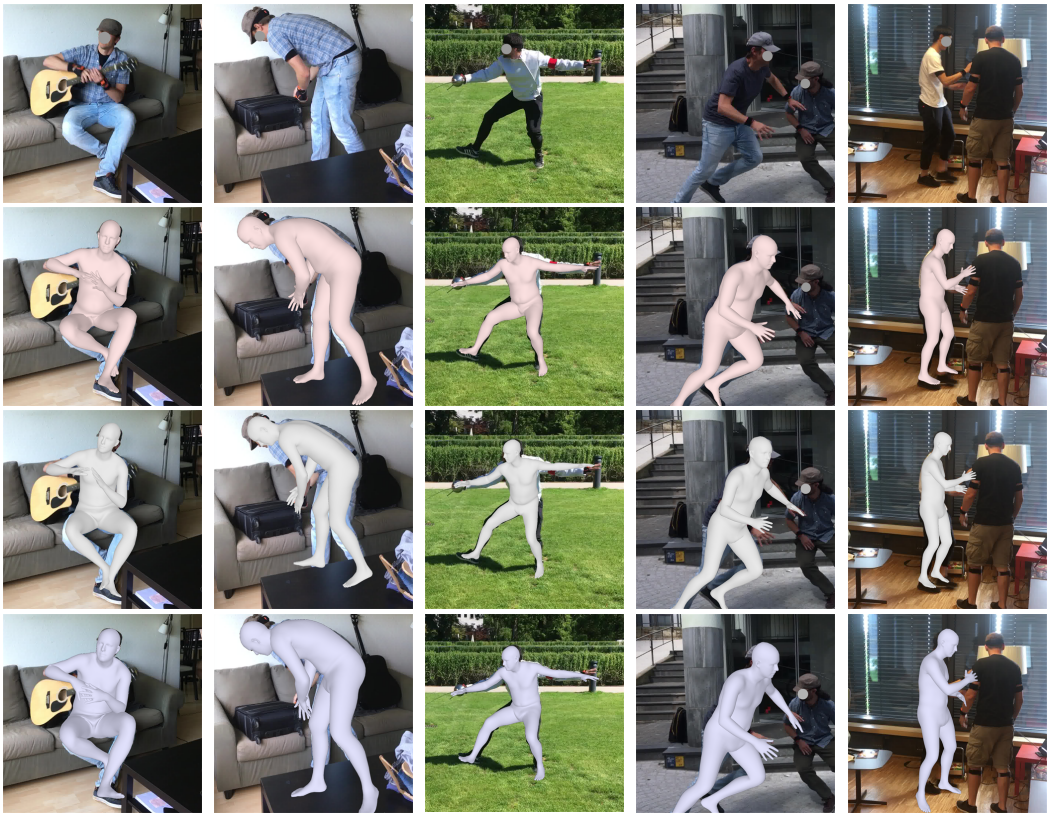

Figure 7: Qualitative comparisons of our method (second row in pink), SPIN [8] (third row in gray), and SMPLify [3] (fourth row in purple) on the MPI-INF-3DHP dataset.

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. 9

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 9

[3] 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. 7, 9, 10, 11

[4] 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. 9

[5] 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. 9

[6] 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. 9

[7] 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. 9

[8] 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. 9, 10, 11

[9] 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. 7

[10] 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. 9

[11] 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. 10

[12] 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. 9

[13] 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. 9

[14] 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. 7

[15] 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. 7

[16] 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. 7

[17] 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. 9

[18] 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. 9

[19] Tanner Schmidt, Richard Newcombe, and Dieter Fox. DART: dense articulated real-time tracking with consumer depth cameras. *Autonomous Robots*, 39(3):239–258, 2015. 9

[20] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-YOLOv4: Scaling cross stage partial network. *arXiv preprint arXiv:2011.08036*, 2020. 9

[21] 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. 7, 9

[22] 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. 9