CLOAF: CoLlisiOn-Aware Human Flow

Andrey Davydov* Martin Engilberge Mathieu Salzmann Pascal Fua
CVLab, EPFL
{name}.{surname}@epfl.ch

Abstract

Even the best current algorithms for estimating body 3D shape and pose yield results that include body self-intersections. In this paper, we present CLOAF, which exploits the diffeomorphic nature of Ordinary Differential Equations to eliminate such self-intersections while still imposing body shape constraints. We show that, unlike earlier approaches to addressing this issue, ours completely eliminates the self-intersections without compromising the accuracy of the reconstructions. Being differentiable, CLOAF can be used to fine-tune pose and shape estimation baselines to improve their overall performance and eliminate self-intersections in their predictions. Furthermore, we demonstrate how our CLOAF strategy can be applied to practically any motion field induced by the user. CLOAF also makes it possible to edit motion to interact with the environment without worrying about potential collision or loss of body-shape prior.

1. Introduction

Feed-forward approaches to estimating human body 3D shape and pose from a single image have become remarkably effective [6, 25, 35]. The very recent transformer-based architecture of [10] embodies the current state-of-the-art. It is pre-trained on 300 million images and fine-tuned on most SMPL data sets in existence. However, as good as these methods have become, they can still produce unrealistic poses with substantial self-intersections of body parts, as illustrated by Fig. 1. This is a serious issue if video-based motion capture is to be used in fields, such as robotics or realistic animation, where preventing self-intersections is of utmost importance.

Most current approaches to addressing this issue [4, 13, 24, 28] are iterative. They penalize self-intersections explicitly by minimizing an interpenetration loss. This requires explicitly detecting self-intersections and then performing a separate optimization step, which makes the whole process non-differentiable and precludes its use during training. Another approach is to eliminate self-intersections in the training databases [26]. While all these methods help, they do not guarantee the absence of self-collisions at inference time.

In this paper, we propose a different approach. It prevents self-intersections in a differentiable manner and without an explicit detection step. To this end, we rely on the fact that if the scene flow from one body to another is the

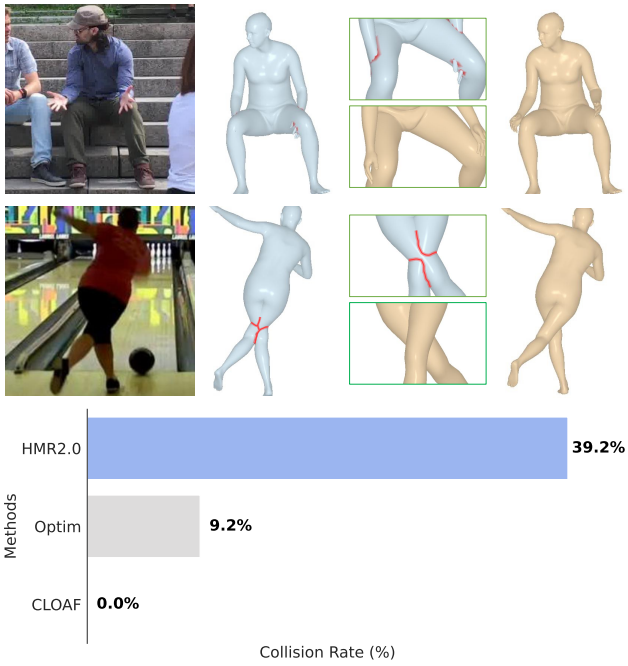

Figure 1. **Self-intersections in SOTA methods.** *Top rows.* HMR2.0 [10] (*first row*) and PARE [19] (*second row*), two of the best current methods, produce bodies shown in blue with self-intersections. CLOAF removes them and generates the results shown in gold. *Bottom row.* HMR 2.0 [10] recovers bodies with self-intersections in 39.2% of frames of the 3DPW-test set. A recent post-processing method such as [28] brings this down to 9.2%. CLOAF drops this number all the way to zero.

*This work was supported in part by the Swiss National Science Foundation

solution of an Ordinary Differential Equation (ODE), then there *cannot* be any self-intersections. Thus, given a volumetric representation of body shapes, we formulate an ODE that models their deformation over time and show how it can be solved with respect to the parameters of the body model we use to represent humans. This means that we can prevent self-intersections while imposing the proper geometric priors on our reconstructions. In other words, our method is able to map any motion flow, even those that may seem implausible, to a flow without any self-intersections. None of the existing methods can achieve this.

In its simplest form, our CoLlisiOn-Aware Flow (CLOAF) method can be used to interpolate between two non self-intersecting body representations so that the intermediate body shapes are both realistic and self-intersection free. It can also be used in a more sophisticated manner to remove self-intersections from the output of single-frame pose estimators, such as [10], while remaining as close as possible to the original poses. Fig. 1 illustrates this. Because it is differentiable, CLOAF can also be integrated into the training pipeline of a deep network to improve its performance. Additionally, we demonstrate how our CLOAF integration procedure can utilize practically any customized motion field to move towards a target area and model interactions with surrounding objects.

In short, our contribution is to use the diffeomorphic nature of ODEs to build a flow-based pipeline to compute human body trajectories without self-intersections. This is a very generic method and, because it is differentiable, it can be used in conjunction with any body pose estimation scheme. The code will be made available at https://github.com/cvlab-epfl/CLOAF.

2. Related Work

Feed-Forward Pose and Shape Estimation. While recent approaches to pose and shape estimation from images have become spectacularly good [6, 10, 19, 25, 26, 35], they still do not guarantee that the resulting body models are self-penetration free. The latest transformer-based architecture of [10] underwent pre-training using 300 million images and further refinement on the majority of existing SMPL datasets, making it the current state-of-the-art. Nevertheless, self-intersections can easily be found in its output, as shown in Fig. 1.

In part, this is because most current approaches favor the accuracy of the body reprojection in the image, potentially at the expense of plausibility, which includes preventing self-intersection. We are not the first to notice this problem and attempts have been made to fix it. For example, the approach of [26] aims to make the self-contacts natural. To this end, it generates pseudo ground-truth data that features them. A feed-forward model trained on such data handles self-contacts better than previous methods. However, it still produces a significant number of self-intersections, as shown in Table 1.

SOTA method	@0	@100	@100 (w/o col.)	P-MPJPE
HMR2.0 [10]	39.2	21.6	12.3	54.3
PARE [19]	36.1	24.3	11.0	50.9
TUCH [26]	23.7	11.7	8.6	55.5
EFT [16]	17.8	6.5	4.5	58.1
SPIN [20]	15.5	5.5	2.8	59.2

Table 1. **Collisions in SOTA methods.** We report the Col.Rate across samples of 3DPW-test set with at least one collision (@0), at least 100 collisions (@100), and at least 100 collisions among samples of 3DPW-test that do not have collisions.

Collision-Aware Optimization. Since preventing feed-forward methods from producing self-intersections is hard, an alternative is to post-process the results to eliminate them. For example, in SMPLify [4], limbs are modeled as ellipsoids and inter-penetrations are explicitly penalized with the corresponding loss. The more recent SMPLify-X [28] uses Bounding Volume Hierarchies (BVHs) for fast collision detection and introduces local conic 3D distance fields to penalize the penetration [3, 31]. The first method is simple to use but modeling body parts as ellipsoids is an oversimplification that can yield unrealistic results. The second avoids this problem but the computation of the BVHs is costly. SMPLify-DC [26] extends SMPLify by modeling self-contacts more precisely. Furthermore, PROX [13] introduces inter-penetration constraints to prevent collisions between bodies and surrounding objects. COAP [24] suggests using independent body part-aware volumetric occupancy networks. The self-collision then can be seen as the intersection between the neighboring occupancy volumes.

In any event, none of these schemes are differentiable with respect to the input pose estimate. Hence, they cannot be incorporated in an end-to-end trainable pipeline. Additionally, they do not guarantee removal of all self-intersections because they rely on minimizing a loss. By contrast, our flow-based approach never produces self-intersections, is fully differentiable, and can be used during training.

Motion Field Integration. Central to our work is the idea of integrating an ODE-based field to prevent self-intersections.

This concept has been explored since well before the deep-learning era, especially for shape transfer purposes. In [33], shape deformations are modeled as local path line integrations. This enables volume-preserving transformations between shapes while avoiding self-intersections for practically any given input transformation. To preserve diffeomorphisms, NMF [12] employs a series of learnable ODE

integrations to morph a spherical mesh into various shapes, conditioned by a point cloud. It is shown that the generated meshed objects retain feasible physical properties. MeshODE [14] and ShapeFlow [15] directly learn the volumetric field between pairs of meshed objects, then transferring one shape to another without collisions. ODE integration is also discussed in human motion modeling. OccFLow [27] models the temporal deformation sequence for a single human subject using a flow field.

However, none of these methods exploits any data-driven prior model. Hence, intermediate shapes are not guaranteed to be realistic. In essence, we extend these concepts to the problem of transitioning between human body shapes while maintaining a valid parametric representation that preserves realism at every step of the integration.

Inverse Kinematics. One of our main contributions is the coupling of ODEs with a parametric model. The recovery of the underlying pose from spatial points resembles an inverse kinematics (IK) problem. HybriK [21] incorporates an iterative IK module into the image-to-mesh recovery network to better align the 3D keypoints with a parametric body representation. Even though IK is solvable with iterative techniques, it is usually highly restricted by the spatial rig and the structure of the kinematic chains [1]. Instead of performing exact but sometimes overly rigid IK, [7] proposes to explicitly optimize the tangent vector using first-order approximations of the input field. In CLOAF, we project the input motion onto the most plausible velocity of the parametric state. Since we work with velocities instead of displacements, our inverse projection is exact and does not require iterating. In simple terms, we take the best of both worlds: integrating the motion flow without penetrations, while preserving a parametric representation of the body at every moment.

3. Method

When predicting body shape and pose in terms of a parameterized body model such as SMPL [22], the simplest way to discourage self-intersections is to introduce loss functions that make them costly [4, 28]. This can be effective but suffers the same fate as all soft constraints: they can still be violated. In this work, we exploit diffeomorphism, a key property of Ordinary Differential Equations (ODEs), to truly prevent self-intersections.

Motion Flow as an ODE. Let us consider a volume containing a body deforming from a first position B_0 to a second one B_1 between times t_0 and t_1. Each point in that volume follows a specific trajectory. Let us assume the existence of a function $\boldsymbol{f}_\omega$ such that we can write for every point $\mathbf{x}$ in the volume:

$$\begin{aligned} &\frac{d\mathbf{x}}{dt} = \boldsymbol{f}_\omega(\mathbf{x}, t|B_0, B_1) \quad \forall t, t_0 \leq t \leq t_1 \ , \\ &\mathbf{x}(t_0) = \mathbf{x}_0 \ , \end{aligned} \tag{1}$$

where $\mathbf{x}_0$ the initial position of $\mathbf{x}$, while the parameters ω control the behavior of $\boldsymbol{f}$. Then, according to the Picard-Lindelöf theorem [8], the trajectories of two initially distinct points can *never* intersect. This requires the right-hand side of the ODE, the velocity field $\boldsymbol{f}$, to be Lipschitz continuous with respect to $\mathbf{x}$, which neural networks satisfy [29, 32].

Given the formulation of Eq. 1, a start position B_0, and an end position B_1, computing trajectories for points in the volume is a classic Cauchy problem that can be solved efficiently using numerical solvers. This means that if we start from a valid position B_0, it will remain valid throughout $t_0 < t < t_1$. In the remainder of this section and the experiment section, we show that the function $\boldsymbol{f}_\omega$ exists and can be learned from data.

Introducing a Body Model. The function $\boldsymbol{f}_\omega$ of Eq. 1 could be implemented by a neural network with weights ω. Then, given N pairs of start and end positions $\{(B_0^i, B_1^i), 1 \leq i \leq N\}$, the weights could be learned so that the final body positions obtained by solving the ODE starting from B_0^i are as close as possible to B_1^i. This amounts to minimizing

$$\sum_i \| B_{0\to1}^i - B_1^i \|^2 \ , \tag{2}$$

with respect to ω, where $B_{0\to1}^i$ is the body position in the coordinate space estimated at time t_1, starting from B_0 at t_0. Hereafter, we omit the explicit dependency of the results of the integration on ω for notational simplicity.

However, without any body shape prior, the intermediate body positions would be completely unrealistic [15, 27]. Thus, we propose to incorporate a body model, specifically the SMPL model [22], in this formalism. To this end, we reformulate Eq. 1 in terms of the parameters Θ of the body model, rather than the points $\mathbf{x}$ as follows.

When using the SMPL model, each 3D point $\mathbf{x}$ on the body surface is parameterized by the underlying SMPL vector $\Theta \in \mathbb{R}^d$. For each one, we can write

$$\begin{aligned} &\mathbf{x}(t) = \mathbf{x}(\Theta(t)) \ , \\ \Rightarrow &\frac{d\mathbf{x}}{dt} = \frac{d\mathbf{x}}{d\Theta}\frac{d\Theta}{dt} = \mathbf{J}\frac{d\Theta}{dt} \ , \end{aligned} \tag{3}$$

where $\mathbf{J} \in \mathbb{R}^{3\times d}$ is the Jacobian of the SMPL transformation $\mathbf{x}(\Theta)$, computed given the body state $\Theta(t)$. Injecting this into the time derivative of Eq. 1 yields

$$\mathbf{J}\frac{d\Theta}{dt} = \boldsymbol{f}_\omega(\mathbf{x}, t|\Theta_0, \Theta_1) \ . \tag{4}$$

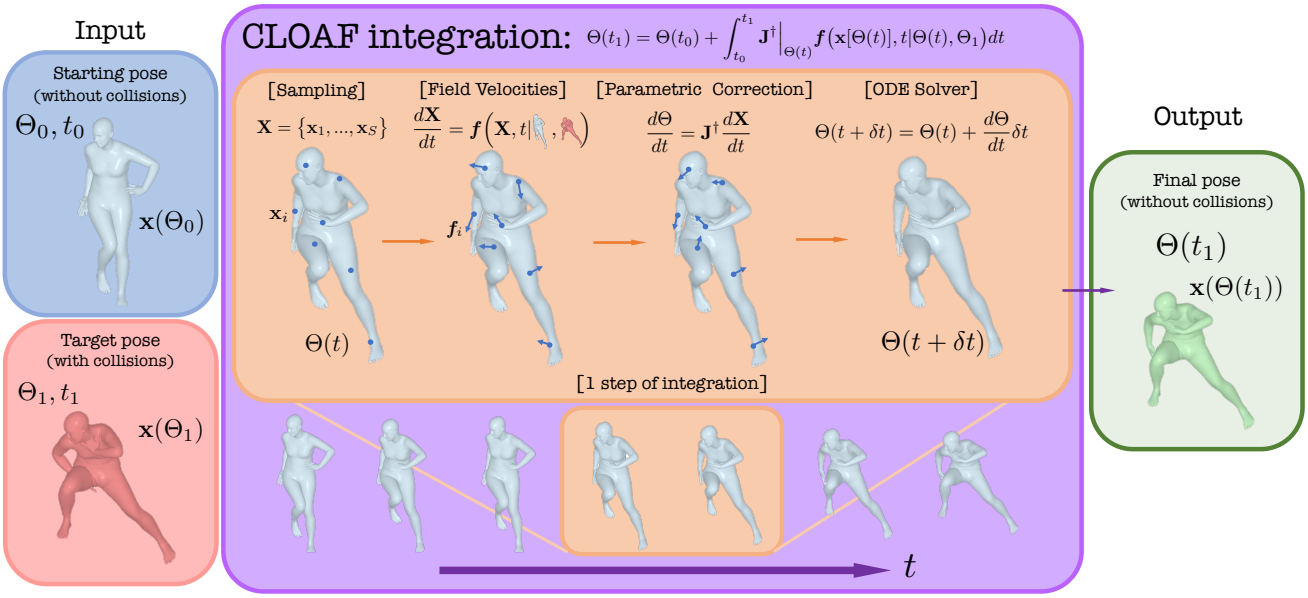

Figure 2. **Method overview.** CLOAF integrates from an initial body pose without self-intersections towards a target one that may feature some. Every integration step involves sampling points from the body surface, calculating approximate spatial velocities, correcting these velocities in the parametric space, and performing an integration step using the ODE solver. When the integration is complete, the body shape is without self-intersections and its pose is taken to be the corrected pose and the output of our method.

Since the number of points S from the SMPL mesh can be taken to be much larger than the dimension of Θ, writing Eq. 4 for each point produces an over-constrained system of linear equations. It can be solved in the least-squares sense, which yields

$$\frac{d\Theta}{dt} = \mathbf{J}^\dagger \cdot \boldsymbol{f}_\omega(\mathbf{X}, t|\Theta_0, \Theta_1) \ , \tag{5}$$

where $\mathbf{X} \in \mathbb{R}^{3S}$ is the vector formed by concatenating the coordinates of all S points and $\mathbf{J}^\dagger = (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T$ is the Moore-Penrose pseudo-inverse of $\mathbf{J} \in \mathbb{R}^{3S\times d}$. This computation is akin to solving an inverse kinematic problem [2, 7, 11].

We can now reformulate the ODE of Eq. 1 in terms of the parameters Θ of the SMPL model using Eq. 3. It becomes

$$\begin{aligned} \frac{d\Theta}{dt} &= \mathbf{J}^\dagger \boldsymbol{f}_\omega(\mathbf{X}(\Theta), t|\Theta_0, \Theta_1) \ , \\ \Theta(t_0) &= \Theta_0 \ , \end{aligned} \tag{6}$$

where Θ_0 parameterizes the initial body pose, Θ_1 the final one, and f_ω approximates $\frac{d\mathbf{X}}{dt}$. Solving Eq. 6 yields an evolution in the SMPL parameter space instead of the coordinate space of Eq. 1. Thus, it enforces the learned shape prior.

To train the network, we learn its weights ω by adapting the training scheme introduced above as follows. Given N pairs of start and end poses $\{(\Theta_0^i, \Theta_1^i), 1 \leq i \leq N\}$, the weights are learned so that the final body positions corresponding to the parameters obtained by solving the ODE starting from the initial Θ_0^i are as close as possible to the body positions corresponding to the target parameters Θ_1^i. Hence, we reformulate Eq, 2 in terms of Θ, which induces body points $\mathbf{x}$ using Eq. 3. This amounts to minimizing

$$\sum_i \|\mathbf{x}(\Theta_{0\to1}^i) - \mathbf{x}(\Theta_1^i)\|^2 \ , \tag{7}$$

with respect to ω, where $\Theta_{0\to1}^i$ is the body position in the parametric space estimated at time t_1, starting from Θ_0 at t_0.

Points Sampling. Solving Eq. 6 requires multiple computations of the Jacobian, which means performing both the forward and backward pass of the SMPL transformation. On modern GPUs, this can be efficiently parallelized. In practice, we found that $S = 1000$ points is sufficient to cover the full body shape and compute meaningful Jacobians. We ablate the sampling number S in Sec. 4.5.

In [26], it was pointed out that some regions of the body, such as the crotch or the armpits, are more prone to natural self-intersections. Furthermore, it was shown that the SMPL blending parameters do not adequately compensate for that, being insufficiently precise [4, 28]. Hence, we exclude these areas from the sampling procedure.

Network Architecture and Training. The input vector to the motion field network $\boldsymbol{f}_\omega$ consists of the points $\mathbf{x}$ and

the trajectory description $\{t, t_0, t_1, \Theta_0, \Theta_1\}$. Specific input strategies are discussed in the following paragraph. All input values are extended with Fourier features [30] to better accommodate to slight variations in the input signal, the number of frequencies is $n_f = 20$. Regarding the model, we use a variant of the ShapeFlow network architecture [15] to implement $\boldsymbol{f}_\omega$. It relies on a 6-layer MLP, where the output of each layer (except the last) is concatenated with an input vector. Note that such architecture is Lipschitz continuous, as required by the theory, because all the constituent layers are Lipschitz.

To solve ODEs, we use NeuralODE [5] with the adaptive Dormand-Prince solver. For all computations in this paper, we use one Tensor Core GPU NVidia A100.

As our training data, we use the AMASS dataset [23] that contains pose sequences stored in the SMPL format. During training, we sample pairs of poses and integrate between them. In our experiments, we found that using the absolute time t as input does not provide any sufficient information to the model, preventing robust convergence. Instead, we use the "time left" variable, $\Delta t = t_1 - t$; this is crucial to give the model a sense of speed, and it significantly improves convergence.

At each step, the field $\boldsymbol{f}_\omega$ is computed at S points $\mathbf{x}$ from the body surface, using Eq. 3. Following [36], points are sampled uniformly from the mesh surface. Finally, the motion field network can be written as

$$\boldsymbol{f}_\omega = \boldsymbol{f}_\omega(\mathbf{x}|\Theta(t), \Theta_1, \Delta t) \;, \tag{8}$$

where all points $\mathbf{x}$ are stacked together and concatenated with the current pose $\Theta(t)$, the target pose Θ_1 and the time gap Δt that are the same for a given body.

In our experiments, we found that the flow model trained from scratch is prone to produce unrealistic velocity values. To avoid this, we scale the output predictions by the expected speed averaged across all the points, we see that it substantially stabilizes the training and converges faster to plausible fields.

Solving Eq. 6 is a two-step operation: field estimation followed by parametric re-projection, with the latter taking time. We found that replacing $\Theta(t)$ by its approximation computed via linear interpolation $\tilde{\Theta}(t) = \Theta_0 + \frac{t-t_0}{t_1-t_0}(\Theta_1 - \Theta_0)$ significantly stabilizes the training, makes it much faster, and does not bring any detrimental effect at inference time. This means that the model implicitly learns a linear interpolation in the parametric space through integration in the coordinate space, where non self-intersection is preserved.

When integrating the field for one pair of poses we induce the loss only for the ending point of the trajectory, Θ_1, as shown in Eq. 7. We found that the model is prone to get stuck and not move towards the target, especially in the first stages of training. To address this, we extend the aforementioned approximation and compute the following "trajectory" loss for every ith pair of poses:

$$L^i_{traj} = \frac{1}{M}\sum_{m=0}^{M} \|\Theta_{0\to t_m} - \mathbf{x}(\tilde{\Theta}(t_m))\|^2 \;, \tag{9}$$

where $t_m = t_0 + \frac{m}{M}(t_1 - t_0)$, M is the number of steps in the integration, and $\Theta_{0\to t_m}$ is the body position in the parametric space estimated at time t_m, starting from Θ_0 at t_0. Such loss significantly helps the model to stay on the trajectory and not diverge from the target.

4. Experiments

4.1. Datasets and Metrics

Datasets. To train the motion field network, we utilize the AMASS dataset [23], which contains more than 40 hours of motion sequences in the common SMPL body representation. To evaluate the baseline methods and our approach, we use the 3DPW [34] dataset. It contains 60 video sequences of various activities in the wild. All samples have a ground-truth in the SMPL format as well. We use the common train/test split, the training part is used for unsupervised fine-tuning (Sec. 4.3), while the test set is a common benchmark and is used in all our experiments for evaluation. The COCO_{EFT} [16] pseudo ground-truth dataset of in-the-wild images is used for supervised part of the EFT baseline training in Sec. 4.3.

Metrics. To evaluate the 3D pose, we use the standard 3D Mean-Per-Joint Position Error (MPJPE, *mm*) and its Procrustes Aligned version (P-MPJPE, *mm*). To assess the smoothness of the motion, we compute the acceleration error (Accel.Err, mm/s^2) between the predicted and ground-truth 3D keypoints. This metric has been used in previous works [6, 17, 18] to quantify the trade-off between 3D pose accuracy and motion consistency. Since our main goal is to eliminate self-intersections, we also report the collision rate (Col.Rate@C, %) computed on the 3DPW-test [34] set. It reflects the proportion of samples across the dataset that have more than C vertices inside the body.

4.2. Eliminating Self-Intersections

As shown in Table 1, self-collisions are prevalent in the output of some of the best current techniques. CLOAF can be used to remove them.

Let Θ be the body shape estimate produced by a neural network, which can contain self-intersections such as those depicted by Fig. 1. To eliminate them in any given frame of a video sequence, we start from a body shape estimate Θ_0 that does not contain any and solve the ODE of Eq. 6 with the target shape Θ_1 being Θ. Body shapes along the resulting trajectory are guaranteed to be self-intersection-free and we take the final one to be our refined estimate. Note that

method	MPJPE	P-MPJPE	Accel.Err	Col.Rate@0
HMR2.0 [10]	82.0	52.7	16.1	39.2%
Opt.Cones [28]	**81.3**	**52.4**	14.3	9.2%
Opt.Cones$^+$ [28]	82.5	53.3	16.4	9.1%
Opt.Contact [26]	81.7	53.1	14.9	8.6%
COAP [24]	81.9	52.8	14.5	5.7%
CLOAF	82.3	53.2	**9.4**	**0.0%**

Table 2. **Comparison against collision penalizing techniques.** Our method yields smoother motion flow compared to previous approaches and guarantees collision-free predictions. In terms of pose estimation metrics, it is on par with existing methods. The "$^+$" symbol denotes that the method uses previous frames for initialization, as we do. We use 3DPW-test [34] for evaluation.

method	P-MPJPE	Col.Rate
EFT [16] (baseline)	58.1	6.5
+CLOAF (post-proc.)	57.9	0.0
Opt.Cones [28]	57.5	6.1
CLOAF (*no diff*)	57.8	6.3
CLOAF (*diff*)	**55.4**	**3.4**

Table 3. **Using the differentiability of CLOAF to fine-tune a network.** This is less efficient to remove self-intersections than using CLOAF to post-process but better than the comparable baselines. Fine-tuning with differentiable CLOAF significantly improves the accuracy of the model. We use the 3DPW-test set [34] for evaluation.

for this to work properly Θ_0 must be self-intersection-free. As we work with complete video sequences, when processing a frame, we take the corrected pose in the previous one to be Θ_0. In the case when we do not have access to sequences, we can use different strategies that we discuss in Section 4.5.

In Table 2, we compare our results against those of the baselines. For fairness, we test the baseline of [28] initialized using both, the original approach (the body estimate from the current frame) and ours (the body estimated from the neighboring frame). While the optimization-based methods and their soft constraints cannot completely eliminate the self-intersections, ours does while at the same time providing poses that are temporally more consistent. This is achieved at the cost of a very slight drop in reconstruction accuracy, which is not clearly significant because there are some self-intersections in the so-called ground-truth data. Fixing such errors, while actually correct, makes our results appear to be further from the ground-truth than some incorrect ones.

4.3. Self-Intersection-Aware Fine-Tuning

CLOAF is differentiable with respect to the corrupted input poses. Therefore, it can be added at the end of any pose and shape estimation network to fine-tune it in an end-to-end manner to reduce its self-intersection rates.

We demonstrate this using the EFT pose and shape estimation model [16] parameterized by Ω. Its predictions $\hat{\mathbf{x}}_\Omega$ can have self-intersections. We can remove them by computing $CLOAF(\hat{\mathbf{x}}_\Omega)$. The differentiability of CLOAF allows us to introduce the loss $L_{\text{cloaf}} = \|\hat{\mathbf{x}}_\Omega - CLOAF(\hat{\mathbf{x}}_\Omega)\|$, whose computation does not require any new annotations. We then use the 3DPW training set [34] to refine the network weights by minimizing a composite loss that is the sum of L_{cloaf} and the usual supervised loss.

In Table 3, we compare our approach to two baselines. The first is designed for comparison against a traditional optimization method [28]. In this scenario, the network is fine-tuned using the loss $L = \|\hat{\mathbf{x}}_\Omega - \tilde{\mathbf{x}}\|$, where $\tilde{\mathbf{x}}$ is the self-intersection-free result of the optimization method. Since it is not differentiable[1], the gradient of the collision correcting operation cannot be used, which hurts performance. The second baseline involves CLOAF but with the gradient detached when computing L, meaning that the right term of the loss is only used in the computation of the loss, but no gradient from the ODE solver is used.

All three fine-tuning strategies help reduce the self-intersection rate, but our approach (CLOAF (*diff*)) significantly outperforms the optimization-based method in both self-intersection rate and P-MPJPE. Our ablated method with the detached gradient (CLOAF (*no diff*)) is comparable to the optimization-based method. This confirms that the benefit of our approach lies in its differentiability.

4.4. Simplified Motion Fields

So far, we have trained the motion field network $\boldsymbol{f}_\omega$ of Eq. 8 to produce realistic motion fields and have constrained the trajectories to go from a start pose towards a final one. However, there are scenarios in which someone might wish to use simpler fields described by rough displacements that are defined only locally. A simple example is a piecewise field that "moves the left arm up". It can be defined by a vector $\boldsymbol{f}(\mathbf{x}) \neq 0$ around the left arm and zero elsewhere, as illustrated by the gold ovals in Fig. 3. Such a field does not satisfy the Picard-Lindeöf theorem, which we relied in the derivation of Eqs. 1 and 6. Nevertheless, it can be approximated by one that does. Specifically, as in [33], we blend the inner and outer regions in a Lipschitz-continuous way. To this end, we use a Bézier blending function

$$b(r) = \sum_{p=0}^{4} \alpha_p B_p^4\left(\frac{r - r_{\text{in}}}{r_{\text{out}} - r_{\text{in}}}\right), \tag{10}$$

where B_p^4 are the Bernstein polynomials [9], r_{in} and r_{out} are the thresholds for the inner and outer regions, respectively, and α_p are the blending coefficients, $\alpha_0 = \alpha_1 = \alpha_2 = 0$ and $\alpha_3 = \alpha_4 = 1$. Then, the blended field is split into three

[1] It should be noted that here we refer to the approach of [28], which is based on iterative optimization. Its final output is not differentiable with respect to the initial pose, while CLOAF's estimate is.

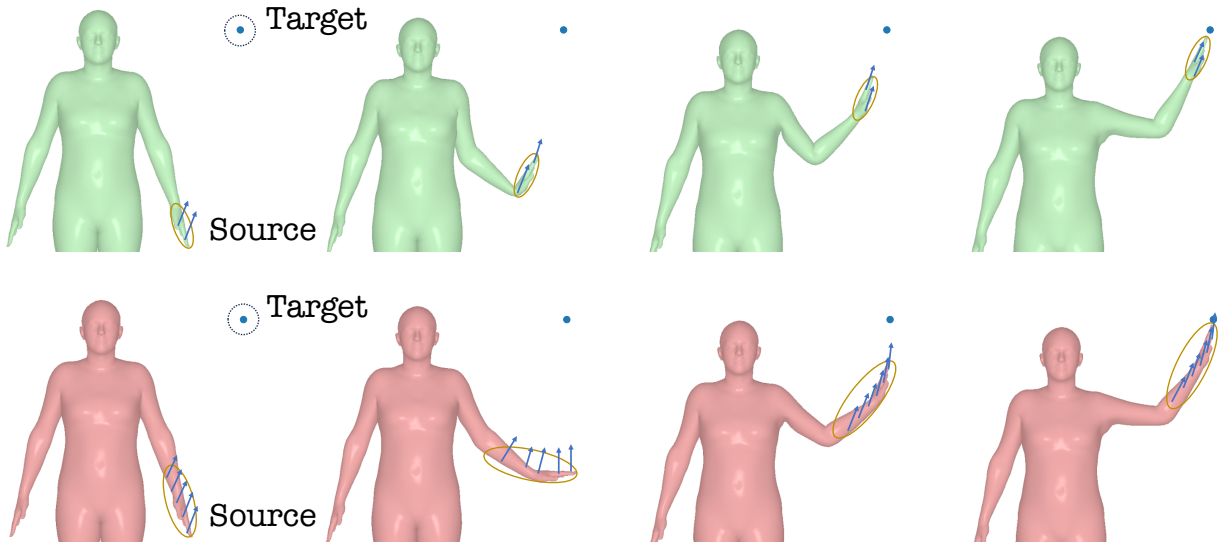

Figure 3. **Moving towards a target.** CLOAF can be used to integrate practically any field, even those that are induced locally. Here the motion is set by the source region on the body (gold oval around the left arm) and the target point (blue dot). The field comprises the direction towards the target point (blue arrows) and adapts during the integration. In the top row, the source region is smaller than in the bottow row, which affects the behavior of the field and induces different motion.

regions as follows:

$$\boldsymbol{f}_b = \begin{cases} \boldsymbol{f}(\mathbf{x}) & r(\mathbf{x}) < r_{\text{in}} \\ \boldsymbol{f}(\mathbf{x}) \cdot (1-b) + \mathbf{0} \cdot b & r_{\text{in}} \leq r(\mathbf{x}) \leq r_{\text{out}} \\ \mathbf{0} & r(\mathbf{x}) > r_{\text{out}} \end{cases} \quad (11)$$

This blended field can be handled by CLOAF. We follow the same procedure as in Sec. 3, but instead of using the neural network to estimate the motion field, we use $\boldsymbol{f}_b$.

We provide two examples in Fig. 3. In both cases, we start with the same body posture, and the goal is to move the left arm towards the blue dot, but we change the size of the non-zero component of the field (gold ovals). The smaller the region, the less realistic the motion is, since fewer body parameters are affected.

Another potential application of such customized integration is the interaction with objects. The subject must be able to move in the field, while not going through objects. This can be achieved by modeling the space inside the object as a region of zero field. When blending during integration, the moving points get stuck in the non-moving area, preventing further penetration.

We illustrate this in the example in Fig. 4. The target point for both hands is located in front of the face (red dot). As in the previous experiment, the field exists only for selected areas of the arms (gold ovals). During integration, the local fields of the hands are blended with the zero field of the box, and the hands stop at the surface of the box (green body on the right). If no constraints are imposed on the field (gold body in the middle), both hands successfully reach the target point.

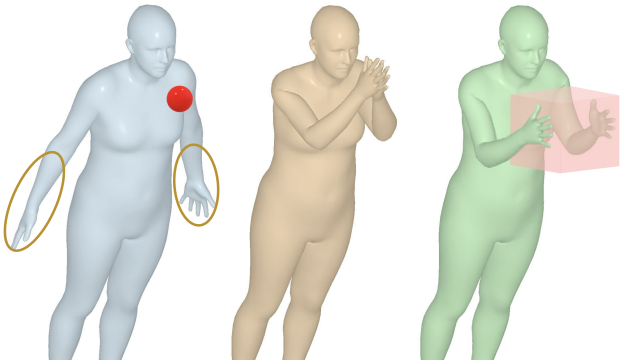

Figure 4. **Interacting with the objects.** One can build a blended field (Eq. 11), where non-zero component (defined at arms, gold ovals) induces the motion, while zero component impedes it. Starting from the same posture (*blue body on the left*), the motion towards the target (*red dot in front of the face*) is computed without any constraints (*gold body in the middle*) and with zeroing out the field inside the red box (*green body on the right*). The blending with the area empty from the field successfully prevents penetrations inside the area. Without constraints, both hands reach the target point.

4.5. Ablation Study

Picking the Initial Body Posture. Recall from Section 4.2 that, when solving our ODE, we took the initial body position to be the corrected one in the previous frame. We refer to this strategy as *Successive Frames*. This works well for video sequences but there are cases where this would be impractical, for example when dealing with single

method	MPJPE	P-MPJPE	Accel.Err
HMR2.0 [10]	82.0	52.7	16.1
Jitter	**81.6**	**52.6**	15.9
Key Poses	82.9	53.6	16.4
Successive Frames	82.3	53.2	**9.4**

Table 4. **Picking the initial body posture.** All variations of CLOAF produce 0.0% collision rate. The *Jitter* strategy improves position error, while *Successive Frames* variant better preserves the overall motion smoothness. Experiments are performed on 3DPW-test set [34].

images. We have therefore explored two alternatives.

- *Jitter.* Having localized the self-intersections, we can determine what part of the body and group of limbs are responsible for them. A straightforward solution is then to randomly jitter these parameters until a body without self-intersection is obtained. This method is simple yet proves to be highly effective in most cases.
- *Keyposes.* We first precompute a dictionary of poses with no self-intersections. To this end, we subsample the AMASS [23] pose dataset and cluster the poses using K-means with the keypoint distance as metric. We then take the closest neighbors to the K cluster centers as keyposes. Finally, given a body we wish to correct, we take the closest keypose to be our starting point. We found $K = 128$ to be sufficient to find a suitable keypose shape in most cases.

In Table 4, we compare these different ways to initialize the CLOAF process on the same video sequences of 3DPW-test set [34]. *Successive Frames* denotes the approach of Section 4.2, which is clearly best at preserving a smooth natural motion. However, the *Jitter* technique yields slightly more accurate pose estimates, even surpassing those of the baselines as reported in Table 2. The *Keyposes* approach performs worse than the others because it requires integration from the relatively distant body shapes. Yet, there may be some rare cases when it is the only usable, for example, when there is no neighboring frame without self-intersections and jitter in the parametric space does not yield a valid body shape.

Choosing Optimal Sampling. As discussed above, one step of CLOAF includes the Jacobian computation with the following solving the over-constrained system of equations. It brings in a natural tradeoff between time and accuracy, since the more points we sample, the more time is spent on the Jacobian computation, but the more accurate the result is. We explore the effect of the number of points S used to sample from the body shape following Eq. 5.

We show the results in Fig. 5. The running time is the time spent only on the Jacobian computation, as all other steps are negligible. To estimate the accuracy of the CLOAF step, we develop a simple yet effective metric. We sample an SMPL vector Θ_0 and add a random noise $\delta\Theta$ of the

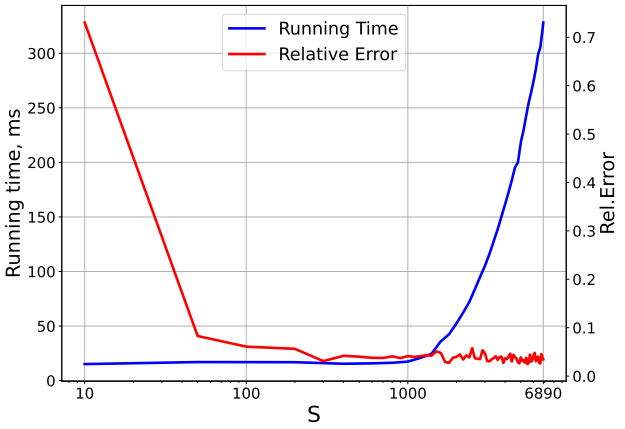

Figure 5. **Ablation on Points Sampling.** The Jacobian computation injects a tradeoff between the accuracy and the running time. Using too many points S for the evaluation slows down the calculations, while less do not provide enough accuracy. We choose $S = 1000$ as the best tradeoff.

known amount (we use $\|\delta\Theta\| = 10^{-2}$) to obtain the second posture $\Theta_1 = \Theta_0 + \delta\Theta$. For a pair of bodies and the sampling number S, one can compute the distance between the bodies in the coordinate space $\boldsymbol{f} \in \mathbb{R}^{3S}$. As the linear approximation of the SMPL transformation holds for such a small $\delta\Theta$, solving $\delta\hat{\Theta} = \mathbf{J}^{\dagger}\boldsymbol{f}$ must give a solution that is close to the "ground-truth" $\delta\Theta$. A more elaborate reasoning behind the linear assumption we make is discussed in the supplementary material. We define the relative error as

$$RE(\delta\Theta) = \frac{\|\delta\hat{\Theta} - \delta\Theta\|}{\|\delta\Theta\|} \,. \tag{12}$$

We choose $S = 1000$ as a good tradeoff between the accuracy and the time and use it in all our experiments.

5. Conclusion

We have presented an approach to reliably eliminating self-intersection from body shape and pose estimation by solving an ODE while imposing a body shape prior. Unlike methods that rely on minimizing a loss function, ours guarantees the complete disappearance of *all* self-intersections. Furthermore, it is differentiable, which means that it can be integrated into a deep learning training pipeline. We have shown how to exploit the differentiability of CLOAF to fine-tune networks and improve their performance while decreasing the amount of self-intersections they produce. Last but not least, we have demonstrated how our CLOAF strategy can be applied to practically any customized motion field, for example, enabling body interaction with the environment.

In future work, we will extend CLOAF to motion generation and exploit existing motion priors.

References

[1] A. Aristidou, J. Lasenby, Y. Chrysanthou, and A. Shamir. Inverse kinematics techniques in computer graphics: A survey. In *Computer Graphics Forum*. Wiley Online Library, 2018. 3

[2] P. Baerlocher and R. Boulic. An Inverse Kinematics Architecture for Enforcing an Arbitrary Number of Strict Priority Levels. *The Visual Computer*, 2004. 4

[3] L. Ballan, A. Taneja, J. Gall, L. Van Gool, and M. Pollefeys. Motion Capture of Hands in Action Using Discriminative Salient Points. In *European Conference on Computer Vision*, pages 640–653, 2012. 2

[4] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. Keep It SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image. In *European Conference on Computer Vision*, 2016. 1, 2, 3, 4

[5] R. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, 2018. 5

[6] H. Choi, G. Moon, J.Y. Chang, and K.M. Lee. Beyond Static Features for Temporally Consistent 3D Human Pose and Shape from a Video. In *Conference on Computer Vision and Pattern Recognition*, 2021. 1, 2, 5

[7] L. Ciccone, C. Öztireli, and R.W. Sumner. Tangent-space optimization for interactive animation control. *ACM Transactions on Graphics*, 2019. 3, 4

[8] E.A. Coddington and N. Levinson. *Theory of Ordinary Differential Equations*. Tata McGrawHill Education, 1955. 3

[9] G. Farin. *Curves and Surfaces for Computer Added Geometric Design*. Academic Press, 1992. 6

[10] S. Goel, G. Pavlakos, J. Rajasegaran, A. Kanazawa, and J. Malik. Humans in 4D: Reconstructing and Tracking Humans with Transformers. In *International Conference on Computer Vision*, 2023. 1, 2, 6, 8

[11] K. Grochow, S.L. Martin, A. Hertzmann, and Z. Popovic. Style-Based Inverse Kinematics. In *ACM SIGGRAPH*, pages 522–531, 2004. 4

[12] K. Gupta and M. Chandraker. Neural Mesh Flow: 3D Manifold Mesh Generation via Diffeomorphic Flows. In *Advances in Neural Information Processing Systems*, 2020. 2

[13] M. Hassan, V. Choutas, D. Tzionas, and M. J. Black. Resolving 3D Human Pose Ambiguities with 3D Scene Constraints. In *International Conference on Computer Vision*, 2019. 1, 2

[14] J. Huang, C.M. Jiang, B. Leng, B. Wang, and L. Guibas. MeshODE: A Robust and Scalable Framework for Mesh Deformation. *arXiv Preprint*, 2020. 3

[15] C. Jiang, J. Huang, A. Tagliasacchi, and L.J. Guibas. Shapeflow: Learnable deformation flows among 3d shapes. In *Advances in Neural Information Processing Systems*, 2020. 3, 5

[16] H. Joo, N. Neverova, and A. Vedaldi. Exemplar Fine-Tuning for 3D Human Pose Fitting Towards In-the-Wild 3D Human Pose Estimation. In *International Conference on 3D Vision*, 2020. 2, 5, 6

[17] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-To-End Recovery of Human Shape and Pose. In *Conference on Computer Vision and Pattern Recognition*, 2018. 5

[18] M. Kocabas, N. Athanasiou, and M. J. Black. VIBE: Video Inference for Human Body Pose and Shape Estimation. In *Conference on Computer Vision and Pattern Recognition*, pages 5252–5262, 2020. 5

[19] M. Kocabas, C.-H.P. Huang, O. Hilliges, and M.J. Black. PARE: Part attention regressor for 3D human body estimation. In *International Conference on Computer Vision*, 2021. 1, 2

[20] N. Kolotouros, G. Pavlakos, M. J. Black, and K. Daniilidis. Learning to Reconstruct 3D Human Pose and Shape via Model-Fitting in the Loop. In *Conference on Computer Vision and Pattern Recognition*, 2019. 2

[21] J. Li, C. Xu, Z. Chen, S. Bian, L. Yang, and C. Lu. Hybrik: A hybrid analytical-neural inverse kinematics solution for 3d human pose and shape estimation. In *Conference on Computer Vision and Pattern Recognition*, 2021. 3

[22] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M.J. Black. SMPL: A Skinned Multi-Person Linear Model. *ACM SIGGRAPH Asia*, 34(6), 2015. 3

[23] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black. AMASS: Archive of Motion Capture as Surface Shapes. In *International Conference on Computer Vision*, pages 5442–5451, 2019. 5, 8

[24] M. Mihajlovic, S. Saito, A. Bansal, M. Zollhoefer, and S. Tang. COAP: Compositional articulated occupancy of people. In *Conference on Computer Vision and Pattern Recognition*, 2022. 1, 2, 6

[25] G. Moon and K.M. Lee. I2L-MeshNet: Image-to-Lixel Prediction Network for Accurate 3D Human Pose and Mesh Estimation from a Single RGB Image. In *European Conference on Computer Vision*, 2020. 1, 2

[26] L. Müller, A. Osman, S. Tang, C.-H. P. Huang, and M.J. Black. On self-contact and human pose. In *Conference on Computer Vision and Pattern Recognition*, 2021. 1, 2, 4, 6

[27] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Occupancy Flow: 4D Reconstruction by Learning Particle Dynamics. In *International Conference on Computer Vision*, 2019. 3

[28] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. Osman, D. Tzionas, and M.J. Black. Expressive Body Capture: 3D Hands, Face, and Body from a Single Image. In *Conference on Computer Vision and Pattern Recognition*, 2019. 1, 2, 3, 4, 6

[29] L. Smith, J. van Amersfoort, H. Huang, S. Roberts, and Y. Gal. Can Convolutional Resnets Approximately Preserve Input Distances? a Frequency Analysis Perspective. In *arXiv Preprint*, 2021. 3

[30] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 2020. 5

[31] D. Tzionas, L. Ballan, S. A., P. Aponte, M. Pollefeys, and J. Gall. Capturing Hands in Action Using Discriminative Salient Points and Physics Simulation. *International Journal of Computer Vision*, 118(2):172–193, 2016. 2

[32] A. Virmaux and K. Scaman. Lipschitz Regularity of Deep Neural Networks: Analysis and Efficient Estimation. In *Advances in Neural Information Processing Systems*, 2018. 3

[33] W. von Funck, H. Theisel, and H.-P. Seidel. Vector Field based Shape Deformations. *ACM Transactions on Graphics*, 2006. 2, 6

[34] T. von Marcard, R. Henschel, M. Black, B. Rosenhahn, and G. Pons-Moll. Recovering Accurate 3D Human Pose in the Wild Using IMUs and a Moving Camera. In *European Conference on Computer Vision*, 2018. 5, 6, 8

[35] W.-L. Wei, J.-C. Lin, T.-L. Liu, and H.-Y.M. Liao. Capturing Humans in Motion: Temporal-Attentive 3D Human Pose and Shape Estimation from Monocular Video. In *Conference on Computer Vision and Pattern Recognition*, 2022. 1, 2

[36] Y. Zhang, M.J. Black, and S. Tang. We are more than our joints: Predicting how 3d bodies move. In *Conference on Computer Vision and Pattern Recognition*, 2021. 5