Supplementary Materials for ARAPReg: An As-Rigid-As Possible Regularization Loss for Learning Deformable Shape Generators

Qixing Huang*
UT Austin
huangqx@cs.utexas.edu

Xiangru Huang*
UT Austin & MIT
xiangruhuang816@gmail.com

Bo Sun*
UT Austin
bosun@cs.utexas.edu

Zaiwei Zhang
UT Austin
zaiweizhang@utexas.edu

Junfeng Jiang
Hohai University
jiangjf.hhu@gmail.com

Chandrajit Bajaj
UT Austin
bajaj@cs.utexas.edu

	DFAUST	SMAL	Bone
W.o. Decoupling	4.90	7.23	3.82
With Decoupling	**4.52**	**6.68**	**3.76**

Table 1: Ablation study on shape and pose variation. In w.o. decoupling setting, all directions are penalized equally. With decoupling setting is the setting in the main paper, where pose directions are penalized more than shape directions.

A. More Quantitative Results

A.1. Ablation Study on Pose and Shape Variation in Section 4.3

In the Section 4.2, we introduced decoupling shape and pose variations to improve ARAPReg. Here we show an ablation study of this decoupling. In Table.1, we show MSE reconstruction error in AD framework w/w.o shape and pose decoupling. Specifically, in the non-decoupling setting, we use the L2 formulation in Proposition 2, where all directions are penalized equally.

A.2. Comparison with ARAP deformation from the base mesh

Here we show the comparison between our method and the traditional ARAP deformation method, where an ARAP deformation is applied between the base mesh and the output mesh for regularization (c.f. [1, 2, 4]). In Table 2, we show results on DFAUST and SMAL datasets. On DFAUST dataset, there are large deformations among the underlying shapes, and the approach of enforcing an ARAP loss to the base shape is significantly worse than without the ARAP loss. In the SMAL dataset, we pick all samples with the same shape but different poses, the ARAP loss to the base shape offers slight performance gains. However, ARAPReg still outperforms this simple baseline considerably.

	DFAUST	SMAL
No ARAP	5.17	8.74
ARAP Deform.	24.55	7.67
Ours	**4.52**	**6.68**

Table 2: Comparison between our method and the traditional ARAP deformation method. We show reconstruction errors of AD model without ARAP, with traditional ARAP and with our method. The traditional method couldn't handle large pose variation and shape distortion.

B. More Implementation Details

B.1. Model Architecture

Our VAE model consists of a shape encoder and a decoder. Our AD model only contains a decoder. Both encoder and decoder are composed of Chebyshev convolutional filters with $K = 6$ Chebyshev polynomials [3].The VAE model architecture is based on [3]. We sample 4 resolutions of the mesh connections of the template mesh. The encoder is stacked by 4 blocks of convolution + down-sampling layers. The decoder is stacked by 4 blocks of convolution + up-sampling layers. There's two fully connected layers connecting the encoder, latent variable and the decoder. For the full details, please refer to our Github repository.

B.2. Reconstruction evaluation

In the AD model, there's no shape encoder to produce latent variables so we add an in-loop training process to optimize shape latent variables, where we freeze the decoder parameters and optimize latent variables for each test shape. In the VAE training, we also add some refinement steps on the latent variable optimization where we freeze the decoder. We apply this refinement step to both methods w/w.o ARAPReg.

C. More Results

In this section, we show more results of reconstruction (Fig.1), interpolation (Fig.2) and extrapolation (Fig.3) of our methods in variational auto-encoder (VAE) and auto-decoder (AD) frameworks, with and without ARAPReg. We also show more closest shapes for randomly generated shapes in VAE framework with ARAPReg in Fig. 4.

D. Proofs of Propositions in Section 4.2

D.1. Proof of Prop.1

For a shape $\mathbf{g} \in \mathcal{R}^{3n}$ with an infinitesimal vertex displacement $\mathbf{x} \in \mathcal{R}^{3n}$ and $\|\mathbf{x}\|_2 \leq \epsilon$, the local rigidity energy is

$$E(\mathbf{g},\mathbf{x}) = \min_{\{A_i \in SO(3)\}} \sum_{(i,j)\in\mathcal{E}} w_{ij}\|(A_i - I_3)(\mathbf{g}_i - \mathbf{g}_j) - (\mathbf{x}_i - \mathbf{x}_j)\|^2 \tag{1}$$

where A_i is a 3D rotation matrix denoting the local rotation from $\mathbf{g}_i - \mathbf{g}_j$ to $(\mathbf{g}_i + \mathbf{x}_i) - (\mathbf{g}_j + \mathbf{x}_j)$. Note that here vector indexing is vertex indexing, where $\mathbf{g}_i = \mathbf{g}_{3i:3(i+1)}$.

Since the zero and first-order derivatives from E to $\mathbf{x}$ around zero is 0:

$$E(\mathbf{g},\mathbf{x})|_{\mathbf{x}=\mathbf{0}} = 0, \quad \frac{\partial E(\mathbf{g},\mathbf{x})}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{0}} = \mathbf{0} \tag{2}$$

We can use second-order Taylor expansion to approximate the energy E when $\mathbf{x}$ is around zero:

$$E(\mathbf{g},\mathbf{x}) \approx \frac{1}{2}\mathbf{x}^T \frac{\partial^2 E}{\partial \mathbf{x}^2}\mathbf{x} \tag{3}$$

Proposition 1 *Given a function $g(\boldsymbol{x}) = \min_{\boldsymbol{y}} f(\boldsymbol{x},\boldsymbol{y})$, and define $\boldsymbol{y}(\boldsymbol{x}) = (argmin)_{\boldsymbol{y}} f(\boldsymbol{x},\boldsymbol{y})$ such that $g(\boldsymbol{x}) = f(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{x}))$,*

$$\frac{\partial^2 g}{\partial \boldsymbol{x}^2} = \frac{\partial^2 f}{\partial \boldsymbol{x}^2} - \frac{\partial^2 f}{\partial \boldsymbol{x}\partial \boldsymbol{y}}\left(\frac{\partial^2 f}{\partial \boldsymbol{y}^2}\right)^{-1}\frac{\partial^2 f}{\partial \boldsymbol{y}\partial \boldsymbol{x}} \tag{4}$$

By treating each A_i as a function of $\mathbf{x}$, we can rewrite our energy as

$$E(\mathbf{g},\mathbf{x}) = f_{\mathbf{g}}(\mathbf{x}, A(\mathbf{x})) \tag{5}$$

where A is the collection of all A_i.

By using Prop.1, we can get the Hessian from E to $\mathbf{x}$.

In the above formulation, A_i is in the implicit form of $\mathbf{x}$. Now we use Rodrigues' rotation formula to write is explicitly. For a rotation around an unit axis k with an angle θ, its rotation matrix is

$$A_i = I + \sin_\theta \mathbf{k}\times + (1-\cos_\theta)(\mathbf{k}\times)^2 \tag{6}$$

where $\mathbf{k}\times$ is the cross product matrix of vector $\mathbf{k}$.

Since here we apply infinitesimal vertex displacement, rotation angle θ is also infinitesimal. We can approximate 6 as

$$A_i \approx I + \theta\mathbf{k}\times + \frac{1}{2}(\theta\mathbf{k}\times)^2 \tag{7}$$

Let $\mathbf{c} = \theta\mathbf{k}$ and only preserve the first two terms:

$$E(\mathbf{x}) \approx \min_{\{\mathbf{c}_i\}} \sum_{(i,j)\in\mathcal{E}} w_{ij}\|\mathbf{c}_i \times \mathbf{e}_{ij} - (\mathbf{x}_i - \mathbf{x}_j)\|^2 \tag{8}$$

$$= \min_{\{\mathbf{c}_i\}} \sum_{(i,j)\in\mathcal{E}} w_{ij}\|\mathbf{e}_{ij} \times \mathbf{c}_i + (\mathbf{x}_i - \mathbf{x}_j)\|^2 \tag{9}$$

where $\mathbf{e}_{ij} = \mathbf{p}_i - \mathbf{p}_j$

From Prop. 1, we can compute the hessian from E to $\mathbf{x}$ by writing $E(\mathbf{g},\mathbf{x}) = f_{\mathbf{g}}(\mathbf{x},\mathbf{c}(\mathbf{x}))$.

We rewrite our energy function in matrix form

$$E = \begin{bmatrix}\mathbf{x}^T & \mathbf{c}^T\end{bmatrix} \begin{pmatrix} L\otimes I_3 & B \\ B^T & C\end{pmatrix} \begin{bmatrix}\mathbf{x} \\ \mathbf{c}\end{bmatrix} \tag{10}$$

where $\otimes$ denotes the kronecker product or tensor product.

The Hessian from E to $\mathbf{x}$ around zero is

$$H_R(\mathbf{g}) = L \otimes I_3 - B^T C^{-1} B \tag{11}$$

Now we compute each term of $H_R(\mathbf{g})$. Expand $f_{\mathbf{g}}(\mathbf{x},\mathbf{c}(\mathbf{x}))$:

$$f(\mathbf{x},\mathbf{c}(\mathbf{x})) = \sum_{(i,j)\in\mathcal{E}} w_{ij}\|\mathbf{e}_{ij}\times\mathbf{c}_i + (\mathbf{x}_i - \mathbf{x}_j)\|^2$$

$$= \sum_{(i,j)\in\mathcal{E}} w_{ij}(\mathbf{x}_i^2 + \mathbf{x}_j^2 - 2\mathbf{x}_i\mathbf{x}_j + 2(\mathbf{e}_{ij}\times\mathbf{c}_i)^T(\mathbf{x}_i - \mathbf{x}_j)$$

$$+ (\mathbf{e}_{ij}\times\mathbf{c}_i)^T(\mathbf{e}_{ij}\times\mathbf{c}_i))$$

L is the weighted graph Laplacian,

$$\mathbf{L}_{ij} = \begin{cases} \sum_{k\in\mathcal{N}_i} w_{ik}, & i = j \\ -w_{ij}, & i \neq j \text{and} (i,j)\in\mathcal{E} \\ 0, & otherwise \end{cases} \tag{12}$$

The matrix $\mathbf{B}$ is a block matrix whose 3×3 blocks are defined as

$$B_{ij} = \begin{cases} \sum_{k\in\mathcal{N}_i} w_{ik}\mathbf{e}_{ik}\times, & i = j \\ -w_{ij}\mathbf{e}_{ij}\times, & i \neq j, (i,j)\in\mathcal{E} \\ 0, & otherwise \end{cases} \tag{13}$$

Finally,C = diag($C_1 \ldots C_{|P|}$) is a block diagonal matrix

$$C_i = \sum_{j\in\mathcal{N}_i} w_{ij}(\mathbf{e}_{ij}\times)^T(\mathbf{e}_{ij}\times) \tag{14}$$

$$= \sum_{j\in\mathcal{N}_i} w_{ij}\|\mathbf{e}_{ij}\|_2^2\mathbf{I}_3 - \mathbf{e}_{ij}\mathbf{e}_{ij}^T \tag{15}$$

which ends the proof. $\square$

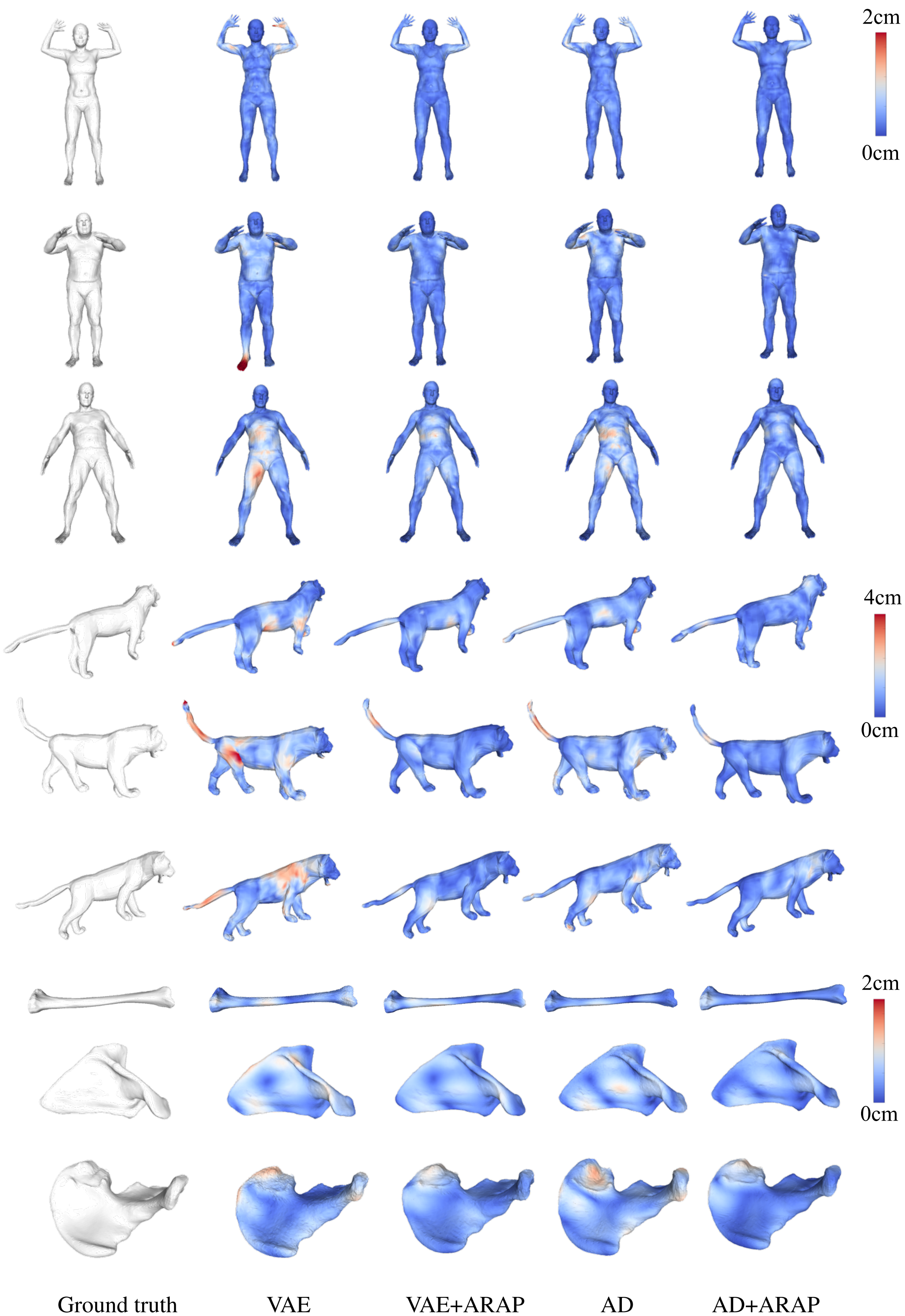

Figure 1: More qualitative results of reconstruction. We show results using VAE and AD generator w/w.o ARAPReg.

D.2. Proof of Prop.2

Consider the eigen-decomposition of

$$\overline{H}_R(\boldsymbol{g}, J) := U\Lambda U^T,$$

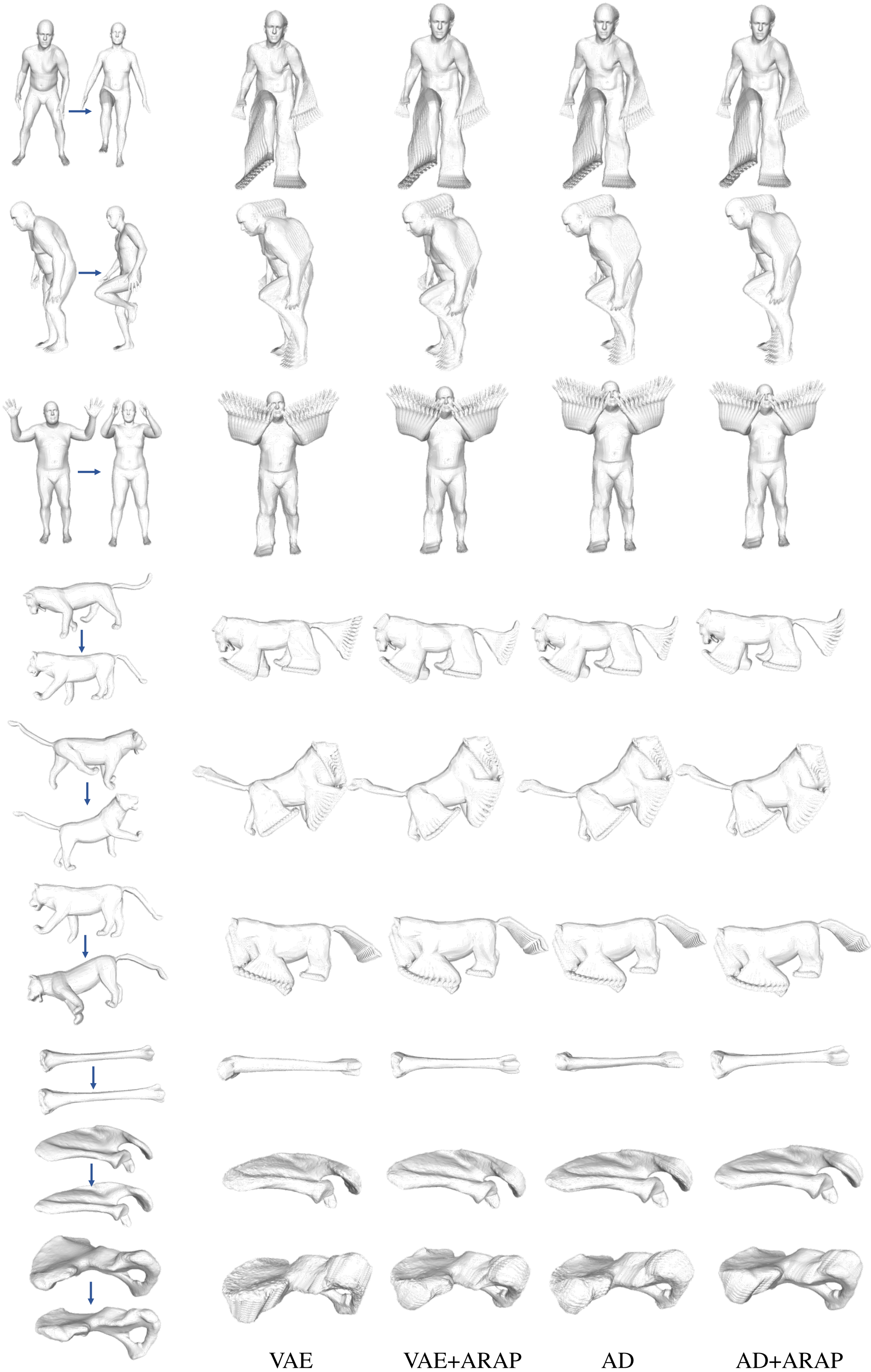

Figure 2: More interpolation results. We show results using VAE and AD generator w/w.o ARAPReg.

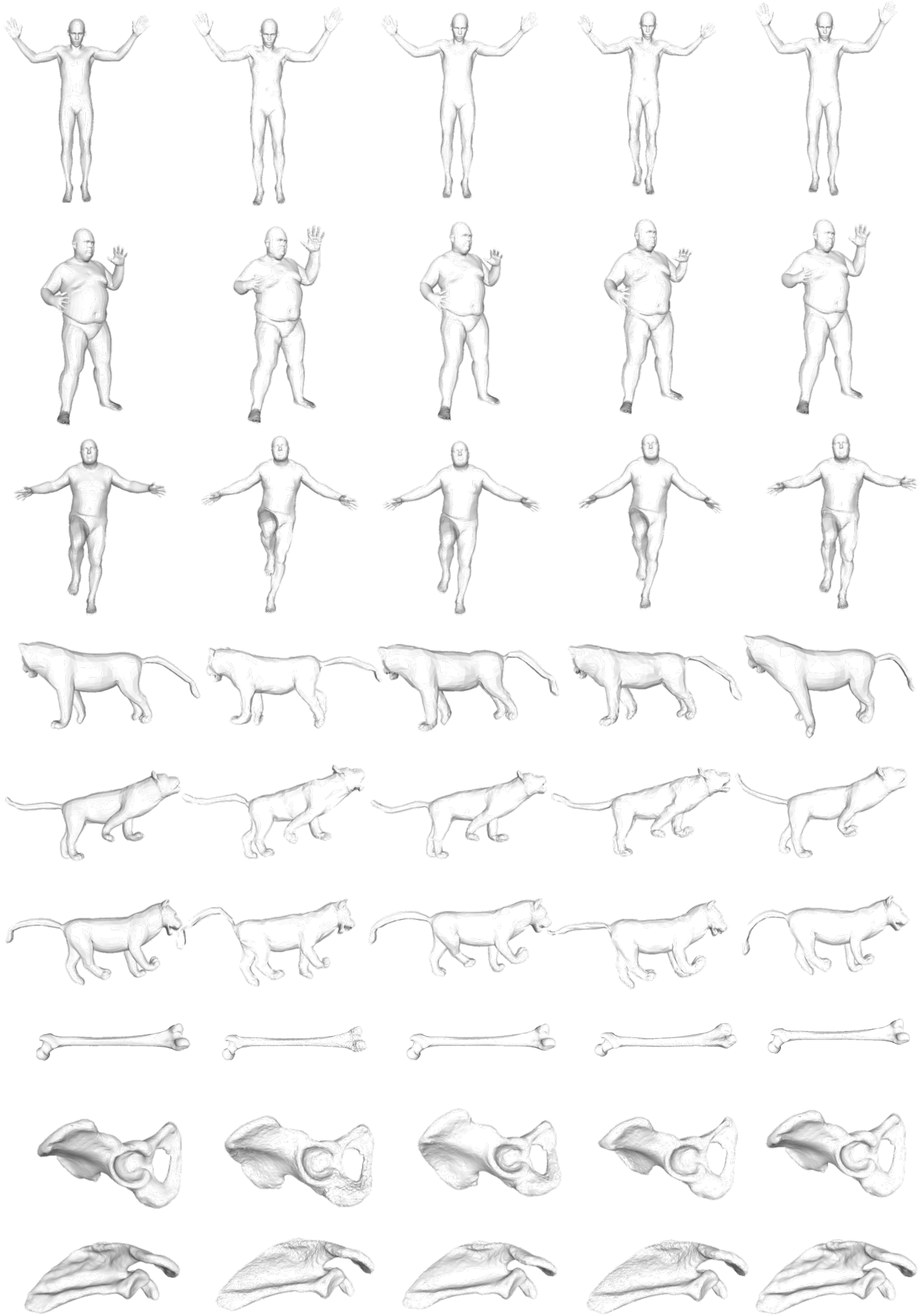

Figure 3: More extrapolation results. We show results using VAE and AD generator w/w.o ARAPReg.

where

$$\Lambda = \mathrm{diag}\big(\lambda_1(\overline{H}_R(\boldsymbol{g}, J)), \cdots, \lambda_k(\overline{H}_R(\boldsymbol{g}, J))\big).$$

Let $\overline{y} = U^T \boldsymbol{y}$. Then

$$\int_{\boldsymbol{y}} \boldsymbol{y}^T \overline{H}_R(\boldsymbol{g}, J)\boldsymbol{y} = \int_{\boldsymbol{y}} \overline{\boldsymbol{y}}^T \Lambda \overline{\boldsymbol{y}} = \int_{\boldsymbol{y}} \sum_{i=1}^{k} \lambda_i(\overline{H}_R(\boldsymbol{g}, J))\overline{y}_i^2$$

$$= \sum_{i=1}^{k} \lambda_i(\overline{H}_R(\boldsymbol{g}, J)) \int_{\overline{\boldsymbol{y}}} \overline{\boldsymbol{y}}_i^2 d\overline{\boldsymbol{y}}$$

$$\frac{1}{\ } \sum^{k} \lambda_i(\overline{H}_R(\boldsymbol{g}, J)) \int \sum^{k} \overline{y}^2 d\overline{y}$$

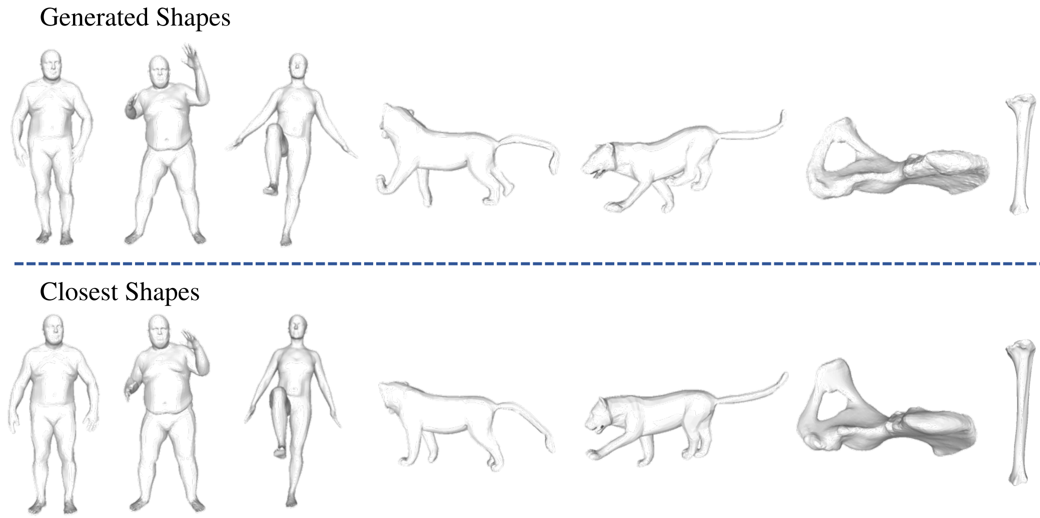

Figure 4: Randomly generated shapes from our VAE frame work and their closed shapes in the training set.

□

E. Gradient of Loss Terms

This section presents the gradients of the loss to the rigidity term.

For simplicity, we will express formulas for gradient computation using differentials. Moreover, we will again replace $\boldsymbol{g}^{\theta}$ and $\frac{\partial \boldsymbol{g}^{\theta}}{\partial \boldsymbol{z}}(\boldsymbol{z})$ with $\boldsymbol{g}$ and J whenever it is possible. The following proposition relates the differential of $r_R(\boldsymbol{g}, J)$ with that of $\overline{H}_R(\boldsymbol{g}, J))$.

Proposition 2

$$dr_R(\boldsymbol{g}, J) = \alpha \sum_{i=1}^{k} \frac{\boldsymbol{u}_i^T d(\overline{H}_R(\boldsymbol{g}, J))\boldsymbol{u}_i}{\lambda_i^{1-\alpha}(\overline{H}_R(\boldsymbol{g}, J))}. \tag{16}$$

Recall that λ_i and $\boldsymbol{u}_i$ are eigenvalues of eigenvectors of $\overline{H}_R(\boldsymbol{g}, J))$.

Proof: The proof is straight-forward using the gradient of the eigenvalues of a matrix, i.e.,

$$d\lambda = \boldsymbol{u}^T dH \boldsymbol{u}$$

where $\boldsymbol{u}$ is the eigenvector of H with eigenvalue λ. The rest of the proof follows from the chain rule. □

We proceed to describe the explicit formula for computing the derivatives of $\boldsymbol{u}_i^T d(\overline{H}_R(\boldsymbol{g}, J))\boldsymbol{u}_i$. First of all, applying the chain rule leads to

$$\begin{aligned} \boldsymbol{u}_i^T d(\overline{H}_R(\boldsymbol{g}, J))\boldsymbol{u}_i &= 2\Big((J\boldsymbol{u}_i)^T H_R(\boldsymbol{g})(dJ \cdot \boldsymbol{u}_i) \\ &- (A(\boldsymbol{g})J\boldsymbol{u}_i)^T D(\boldsymbol{g})^{-1} \cdot \big(dA(\boldsymbol{g}) \cdot (J\boldsymbol{u}_i)\big)\Big) \\ &+ \big(D(\boldsymbol{g})^{-1}A(\boldsymbol{g})J\boldsymbol{u}_i\big)^T dD(\boldsymbol{g})\big(D(\boldsymbol{g})^{-1}A(\boldsymbol{g})J\boldsymbol{u}_i\big). \end{aligned}$$

It remains to develop formulas for computing $dJ \cdot \boldsymbol{u}_i$, $dA(\boldsymbol{g}) \cdot (J\boldsymbol{u}_i)$, and $dD(\boldsymbol{g})$. Note that $J = \frac{\partial \boldsymbol{g}^{\theta}}{\partial \boldsymbol{z}}(\boldsymbol{z})$. We use numerical gradients to compute $dJ \cdot \boldsymbol{u}_i$, which avoid computing costly second derivatives of the generator:

$$d(\frac{\partial \boldsymbol{g}^{\theta}}{\partial \boldsymbol{z}}(\boldsymbol{z})) \cdot \boldsymbol{u}_i \approx \sum_{l=1}^{k} u_{il}(d\boldsymbol{g}^{\theta}(\boldsymbol{z} + s\boldsymbol{e}_l) - d\boldsymbol{g}^{\theta}(\boldsymbol{z})) \tag{17}$$

where $s = 0.05$ is the same hyper-parameter used in defining the generator smoothness term; $\boldsymbol{e}_l$ is the l-th canonical basis of $\mathcal{R}^k$; u_{il} is the l-th element of $\boldsymbol{u}_i$.

The following proposition provides the formulas for computing the derivatives that involve $A(\boldsymbol{g})$ and $D(\boldsymbol{g})$.

Proposition 3

$$\begin{aligned} &dA(\boldsymbol{g}) \cdot (J\boldsymbol{u}_i) = -\mathcal{A}(J\boldsymbol{u}_i) \cdot d\boldsymbol{g} \\ &\boldsymbol{c}^T dD(\boldsymbol{g}) \cdot \boldsymbol{c} = 2\sum_{i=1}^{n} \sum_{k \in \mathcal{N}(i)} \Big((\boldsymbol{g}_i - \boldsymbol{g}_k)^T(d\boldsymbol{g}_i - d\boldsymbol{g}_k)\|\boldsymbol{c}_i\|^2 \\ &\quad - \big(\boldsymbol{c}_i^T(d\boldsymbol{g}_i - d\boldsymbol{g}_k)\big) \cdot \big((\boldsymbol{g}_i - \boldsymbol{g}_k)^T\boldsymbol{c}_i\big)\Big) \end{aligned} \tag{18}$$

Proof:

(1). $dA(\boldsymbol{g}) \cdot (J\boldsymbol{u}_i)$:

Let's denote $J\boldsymbol{u}_i$ as $\boldsymbol{a}$. Now we prove $(A(\boldsymbol{g}) \cdot \boldsymbol{a}) = (A(\boldsymbol{a}) \cdot \boldsymbol{g})$. Then we will have $d(A(\boldsymbol{g}))vJ\boldsymbol{u}_i = d(A(\boldsymbol{g}) \cdot J\boldsymbol{u}_i) = d(A(J\boldsymbol{u}_i) \cdot \boldsymbol{g}) = A(J\boldsymbol{u}_i) \cdot d(\boldsymbol{g})$.

$$
\begin{aligned}
(A(\boldsymbol{g})\boldsymbol{a})_i &= \sum_j A_{ij}(\boldsymbol{g})\boldsymbol{a}_j \\
&= \sum_{k\in N(i)} \boldsymbol{v}_i k \times (\boldsymbol{a}_i - \boldsymbol{a}_k) = \sum_{k\in N(i)} \boldsymbol{v}_i k \times \boldsymbol{a}_i k \\
&= -\sum_{k\in N(i)} \boldsymbol{a}_i k \times \boldsymbol{v}_i k = \sum_j A_{ij}(\boldsymbol{a})\boldsymbol{g}_j \\
&= (A(\boldsymbol{a})\boldsymbol{g})_i
\end{aligned}
$$

This finishes the proof.

(2). $\boldsymbol{c}^T dD(\boldsymbol{g}) \cdot \boldsymbol{c}$:

We have $\boldsymbol{c}_i^T D_{ii}(\boldsymbol{g}) \cdot \boldsymbol{c}_i = \sum_{k\in N(i)}(\|\boldsymbol{v}_{ik}\|^2\|\boldsymbol{c}_i\|^2 - \boldsymbol{c}_i^T \boldsymbol{v}_{ik}\boldsymbol{v}_{ik}^T \cdot \boldsymbol{c}_i)$. We only need to compute the gradient of $\|\boldsymbol{v}_{ik}\|^2$ and $\boldsymbol{v}_{ik}\boldsymbol{v}_{ik}^T$. Note that $\|\boldsymbol{v}_{ik}\|^2 = \boldsymbol{v}_{ik}^T\boldsymbol{v}_{ik}$.

For a vector $\boldsymbol{a}$, we have $d(\boldsymbol{a}^T\boldsymbol{a}) = d(\boldsymbol{a}^T)\boldsymbol{a} + \boldsymbol{a}^T d(\boldsymbol{a}) = d(\boldsymbol{a})^T\boldsymbol{a} + \boldsymbol{a}^T d(\boldsymbol{a}) = 2\boldsymbol{a}^T d(\boldsymbol{a})$ and similarly, $d(\boldsymbol{a}\boldsymbol{a}^T) = 2d(\boldsymbol{a})\boldsymbol{a}^T$. We use these two results to our derivation and we will get the results above.

$$
\begin{aligned}
&\boldsymbol{c}^T dD(\boldsymbol{g}) \cdot \boldsymbol{c} \\
&= \sum_i \sum_{k\in N(i)} (d(\|\boldsymbol{v}_{ik}\|^2)\|\boldsymbol{c}_i\|^2 - \boldsymbol{c}_i^T d(\boldsymbol{v}_{ik}\boldsymbol{v}_{ik}^T)\cdot \boldsymbol{c}_i) \\
&= \sum_i \sum_{k\in N(i)} (d(\boldsymbol{v}_{ik}^T\boldsymbol{v}_{ik})\|\boldsymbol{c}_i\|^2 - \boldsymbol{c}_i^T d(\boldsymbol{v}_{ik}\boldsymbol{v}_{ik}^T)\cdot \boldsymbol{c}_i) \\
&= \sum_i \sum_{k\in N(i)} 2\Big((\boldsymbol{g}_i - \boldsymbol{g}_k)^T(d\boldsymbol{g}_i - d\boldsymbol{g}_k)\|\boldsymbol{c}_i\|^2 \\
&\quad - \big(\boldsymbol{c}_i^T(d\boldsymbol{g}_i - d\boldsymbol{g}_k)\big)\cdot\big((\boldsymbol{g}_i - \boldsymbol{g}_k)^T\boldsymbol{c}_i\big)\Big)
\end{aligned}
$$

References

[1] Marc Habermann, Weipeng Xu, Michael Zollhöfer, Gerard Pons-Moll, and Christian Theobalt. Deepcap: Monocular human performance capture using weak supervision. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 5051–5062. IEEE, 2020. 1

[2] Xueting Li, Sifei Liu, Shalini De Mello, Kihwan Kim, Xiaolong Wang, Ming-Hsuan Yang, and Jan Kautz. Online adaptation for consistent mesh reconstruction in the wild. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 1

[3] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 704–720, 2018. 1

[4] Keyang Zhou, Bharat Lal Bhatnagar, and Gerard Pons-Moll. Unsupervised shape and pose disentanglement for 3d meshes. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXII*, volume 12367 of *Lecture Notes in Computer Science*, pages 341–357. Springer, 2020. 1