χ: Symmetry Understanding of 3D Shapes via Chirality Disentanglement

Supplementary Material

In the supplementary materials, we include information about the data generation (Sec. A), architectural decisions (Sec. B), additional results (Sec. C & Sec. D) and implementation details (Sec. E).

A. BeCoS Data Generation

When generating BeCoS [18] with default settings, each shape is used multiple times to generate a wide variety of shape matching pairs. This results in a train/validation/test split of $20370/274/284$ shapes. Since we use the framework to propagate annotations to all shapes, effectively ignoring the pairings, we restrict BeCoS to use each shape at most once, resulting in the train/validation/test split of $1980/284/274$ shapes. To generate BeCoS$_{\text{-h}}$ and BeCoS$_{\text{-a}}$, we filter the resulting dataset for human and animal entries, respectively. Note that we assign the *centaur* shapes from TOSCA dataset [7] to BeCoS$_{\text{-a}}$.

B. Architectural Design

In our method, each vertex gets assigned two scalar features χ_v and $\bar{\chi}_v$, derived from a non-linear projection $\tilde{g}$ of its original and mirrored Diff3F features. Specifically, we extract a specific component from $\tilde{g}(\mathcal{F}_v)$ and normalise it by the L2 Norm of the whole feature vector. The normalisation ensures that χ_v reflects the relative magnitude of the chosen component compared to the whole feature, while remaining invariant to scaling. Compared to a fixed non-linearity such as $\tanh{[\tilde{g}(\mathcal{F}_v)]_0}$, which only uses the first component, the normalisation incorporates additional information and promotes more stable learning dynamics. To empirically confirm our choice, we provide an ablation by training both architectures and evaluating them in the left-right disentanglement task. The results can be found in Table T.1.

C. Different shape representations.

Our $\mathcal{L}_{\text{var}}$ loss requires mesh connectivity to regularize the smoothness and boundary of our solution, effectively restricting the applicability of our method to meshes. However, connectivity information for different kinds of shape representations, like point clouds, can be approximated. For example using mutual k-nearest neighbors among vertices. To get a first impression of the applicability of our method on point clouds, we train a network using SD + Dino features generated from *rendered point cloud images*. We replace the edges used in $\mathcal{L}_{\text{var}}$ with a primitive mutual k-nearest neighbor ($k = 5$) connectiviy approximation. The models are evaluated in the left/right classification on FAUST. We also include results for the point cloud setting without $\mathcal{L}_{\text{var}}$. The table and figure below show that our method with k-nn approximation performs robustly on most vertices of the point cloud.

Input	Point cloud		Mesh
Losses	w/o $\mathcal{L}_{\text{var}}$	KNN- $\mathcal{L}_{\text{var}}$	Full
Acc	67.08	92.87	94.76

Table T.2. The model trained on point clouds using approximate k-NN connectivity information reaches a high accuracy, compared to a model trained without connectiviy information.

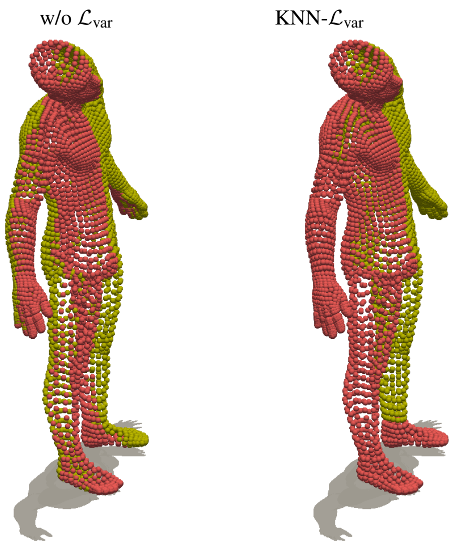
w/o $\mathcal{L}_{\text{var}}$
KNN-$\mathcal{L}_{\text{var}}$

Figure F.1. Qualitative results of models trained on point clouds with and without approximated k-NN connectivity information. Without the $\mathcal{L}_{\text{var}}$ loss, the length of the boundary is not regularized, resulting in an inaccurate left/right split. With the approximate $\mathcal{L}_{\text{var}}$ loss, the model is able to correctly classify most of the points.

The inaccurate assignment of the left foot shows that there are remaining open challenges. Since our main focus is on 3D meshes, we leave this exploration for future work.

D. Additional shape matching results.

We provide additional results for the shape matching task on the FAUST benchmark [6]. We compute vertex correspondences using cosine similarity between the vertex features of the source and target shape. When combined with Diff3F

Train	BeCoS	BeCoS-h		BeCoS-a		FAUST		SMAL	
Test	**BeCoS**	**BeCoS-h**	**BeCoS-a**	**BeCoS-h**	**BeCoS-a**	**FAUST**	**SCAPE**	**SMAL**	**TOSCA**
tanh	75.46	92.51	83.45	73.71	75.87	91.84	94.93	71.04	68.46
Normalisation	**91.84**	**94.09**	**84.19**	**90.36**	**91.10**	**94.76**	**95.51**	**96.59**	**94.09**

Table T.1. Normalisation of the chirality feature χ_v with respect to the whole vector results in higher accuracy across all datasets but FAUST, compared to using $\tanh$.

features, our features achieve a 50.0% decrease in error for the inter-subject and 42.2% for the intra-subject task, compared to Diff3F features. Qualitative results are shown in Fig. F.2.

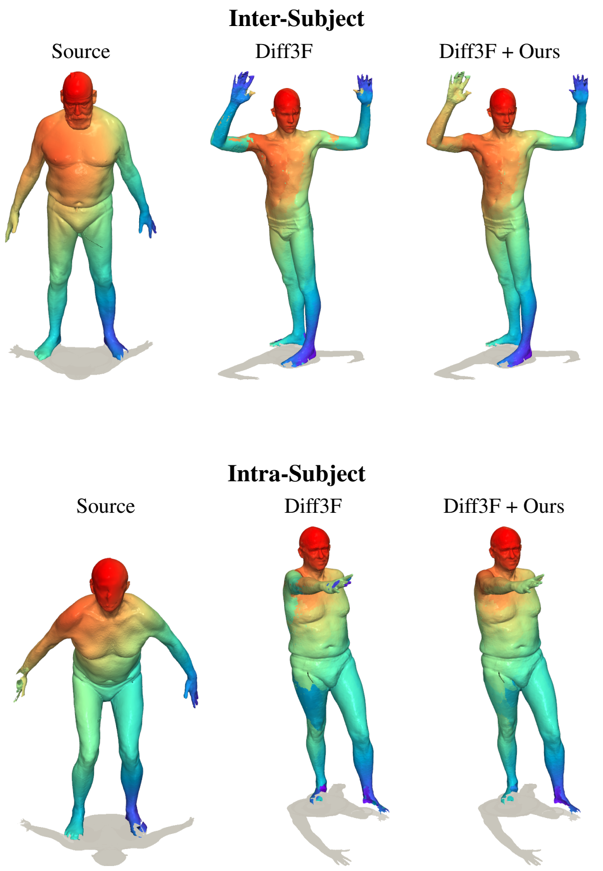

Figure F.2. Our method effectively resolves left/right ambiguity when matching the FAUST benchmark. Both in the inter- and intra-subject case.

E. Implementation details.

We employ a lightweight two-layer MLP to implement g_Φ, with a hidden dimension equal to the input dimension ($D = 3968$) and ReLU as the activation function. Experimentally, we find that using normalization on the output feature works better than sigmoid or tanh functions. The model is trained on a single NVIDIA A40 GPU using ADAM with a learning rate of 10^{-3}. We precalculate the input features and run the training for 20000 iterations, taking around 3h. All details can be found in the code on https://wei-kang-wang.github.io/chirality/.