# DiLO: Disentangled Latent Optimization for Learning Shape and Deformation in Grouped Deforming 3D Objects

**Mostofa Rafid Uddin**[1], **Jana Armouti**[1], **Umong Sain**[2], **Md Asib Rahman**[2], **Xingjian Li**[1], **Min Xu**[1] *

[1]Carnegie Mellon University, Pittsburgh, PA 15213, USA
[2]Bangladesh University of Engineering and Technology, Dhaka 1000, Bangladesh
mxu1@cs.cmu.edu

## Abstract

In this work, we propose a disentangled latent optimization-based method for parameterizing grouped deforming 3D objects into shape and deformation factors in an unsupervised manner. Our approach involves the joint optimization of a generator network along with the shape and deformation factors, supported by specific regularization techniques. For efficient amortized inference of disentangled shape and deformation codes, we train two order-invariant PointNet-based encoder networks in the second stage of our method. We demonstrate several significant downstream applications of our method, including unsupervised deformation transfer, deformation classification, and explainability analyses. Extensive experiments conducted on 3D human, animal, and facial expression datasets demonstrate that our simple approach is highly effective in these downstream tasks, comparable or superior to existing methods with much higher complexity.

**Extended version** — https://arxiv.org/pdf/2511.06115

## Introduction

Parameterizing 3D objects with distinct generative factors, such as shape and deformation, has garnered considerable attention in computer graphics and vision research (Cosmo et al. 2020; Chen et al. 2021b; Huang et al. 2021; Aumentado-Armstrong et al. 2019). In this context, shape typically refers to the intrinsic properties of 3D objects, such as height, body structure, and surface geometry, while deformation refers to extrinsic properties, including pose, motion, twisting, and morphing. By disentangling these generative factors and parameterizing 3D objects accordingly, it is possible to achieve efficient 3D deformation transfer, shape manipulation, and generation (Zhou, Bhatnagar, and Pons-Moll 2020; Song et al. 2023; Sun, Chen, and Kim 2023; Cosmo et al. 2020; Aumentado-Armstrong et al. 2019). This capability has practical applications in industries such as content creation, gaming, and AR/VR (Chen et al. 2021a, 2023).

For nearly a decade, parameterizing specific 3D objects such as humans (Anguelov et al. 2005; Loper et al. 2023; Pons-Moll et al. 2015), hands (Romero, Tzionas, and Black 2022), and faces (Li et al. 2017; Ploumpis et al. 2019), into generative factors like shape and deformation has been achieved using hand-crafted features, such as landmarks, skeletons, or manually estimated point-wise distances. However, obtaining such features requires significant manual effort and expert knowledge. Additionally, for many deforming 3D objects (e.g., organs, proteins), it is often impractical to define clear skeletons or landmark features. Addressing these limitations, recent advancements have led to the development of high-fidelity deep representation learning models (Chen et al. 2021b; Zhou, Bhatnagar, and Pons-Moll 2020; Cosmo et al. 2020; Aumentado-Armstrong et al. 2019) that can parameterize 3D objects into distinct generative factors, shape and deformation codes, in an unsupervised manner, eliminating the need for manually provided features.

These deep learning-based methods are generally trained on grouped 3D object collections, where multiple deforming 3D objects are grouped based on shape, deformation, or other characteristics. The methods leverage group information of their shapes and utilize assumptions specific to deformation to train their models. For example, (Cosmo et al. 2020) enforces that 3D objects with two different deformations of the same shape preserve the geodesic distance between their vertices. (Zhou, Bhatnagar, and Pons-Moll 2020) model deformation in 3D objects as As-Rigid-As-Possible (ARAP) deformation and enforces it during model training. Despite some success, these methods leave much room for improvement. Moreover, implementing deformation-specific constraints during training creates a lot of computational overhead and is thus resource-intensive.

In this work, we approach the problem from a different perspective. We leverage the grouping information of the shapes of the 3D objects to learn two generative factors, one responsible for the commonalities within each group, in other words, shape, and the other accountable for intra-group instance-wise variation, which, in our datasets, is deformation. To this end, we developed a novel method, called **disentangled latent optimization (DiLO)**. DiLO is a two-stage framework. In the first stage, we perform latent optimization of the generative factors using an autodecoder network (Huang et al. 2021). Instead of optimizing individual latent factors for both shape and deformation for each 3D object, we perform **shared optimization for the shape factors** since they are responsible for commonalities within the group. In this way, all objects belonging to a shape

*Corresponding Author

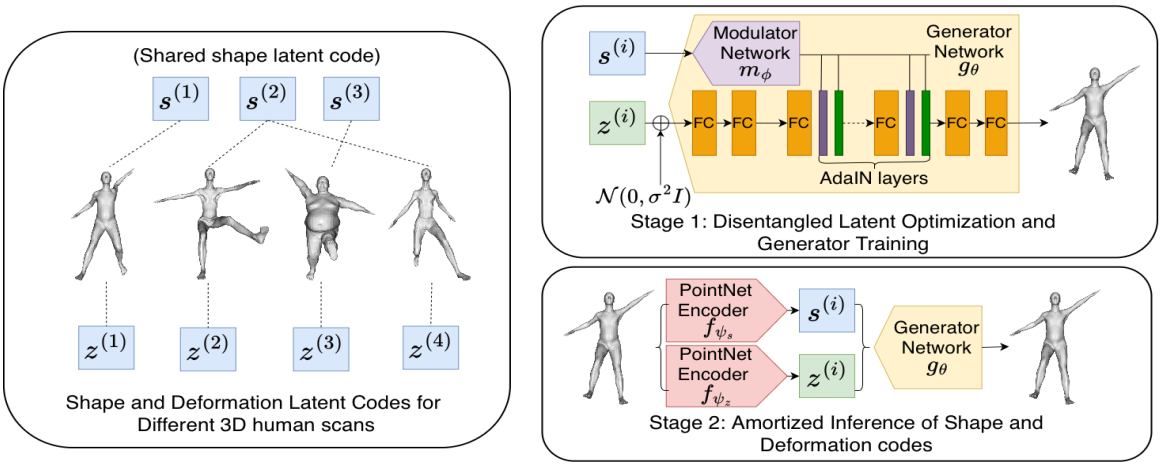


Figure 1: An overview of our proposed unsupervised shape-deformation disentanglement method. On the left, we show the conceptualization of shape codes $s^{(i)}$ and deformation codes $z^{(i)}$. On the right, we demonstrate the two learning stages of our method. In stage 1, we optimize $s^{(i)}$ and $z^{(i)}$ together with a generator network. In stage 2, we infer the optimized codes $s^{(i)}$ and $z^{(i)}$ from the input 3D object using two PointNet (Qi et al. 2017) encoders.

group have the same shape factor, whereas they have different deformation factors. This design ensures disentanglement between the shape and deformation codes. For training the autodecoder network generator, we incorporate adaptive instance normalization (AdaIN) layers commonly used for image style transfer (Gabbay and Hoshen 2021, 2020). Whereas the deformation code is directly passed as input to the generator, the shape code is used to predict the parameters of the AdaIN layers of the generator. In the second stage, we train two permutation-invariant PointNet encoder (Qi et al. 2017) networks that can infer the optimized latent codes from any given 3D object, enabling fast amortized inference. Our framework is lightweight, easy to train, and does not suffer from training instability, unlike many methods that rely on adversarial training.

We conducted extensive experiments to evaluate our method and baseline approaches across multiple 3D datasets, specifically 3D human models (SMPL (Loper et al. 2023)), 3D animal models (SMAL (Zuffi et al. 2017)), and 3D face models (COMA (Ranjan et al. 2018)). The evaluations were performed on three primary tasks: 1) unsupervised 3D deformation transfer, 2) deformation classification from latent codes, and 3) explainability analyses. Both qualitative and quantitative results consistently demonstrated the efficacy and superiority of our method compared to existing baselines. Additionally, we conducted extensive ablation experiments that demonstrate the effectiveness of individual components in our method. We summarize our major contributions as follows:

- We introduce **disentangled** latent optimization to learn shape and deformation for grouped deforming 3D shapes in an unsupervised manner.
- We demonstrate multiple downstream applications of our method, including unsupervised 3D deformation transfer, deformation classification, and explainability analyses using benchmark 3D shape datasets.
- Our method is computationally efficient, practically explainable, and also offers strong performance compared to the complex baseline methods in downstream tasks.

## Related Works

**Unsupervised shape-deformation disentanglement:** Parameterizing 3D shapes into shapes and deformation-specific components has been studied in computer graphics and vision for quite some time. Earlier methods used variants of principal component analysis to disentangle the shape and deformation-specific factors (Rustamov et al. 2013; Corman et al. 2017; Gao et al. 2017). Such disentanglement works when shape and deformation are linearly separable but not when non-linearity is present. Tan et al. (Tan et al. 2018) applied variational autoencoders (VAE) on 3D data and captured factors that are not linearly separable across different dimensions of the VAE latent code. Nevertheless, there was still no explicit disentanglement of shape and deformation-specific factors. GD-VAE (Aumentado-Armstrong et al. 2019) was among the first methods to perform explicit disentanglement of shape and deformation factors into two different latent codes. However, they did not exploit the shape group information of the datasets. Subsequently, several methods (Chen et al. 2021b; Zhou, Bhatnagar, and Pons-Moll 2020; Cosmo et al. 2020) based on VAEs or GANs were developed to disentangle shape and deformation by leveraging shape group information. These approaches consistently outperformed GD-VAE, highlighting the effectiveness of using group information. DiLO also utilizes shape group information for disentanglement; however, unlike prior methods, it avoids computationally expensive operations such as geodesic distance calculations or ARAP deformation, achieving efficient disentanglement without sacrificing performance.

**Unsupervised 3D deformation transfer:** 3D deformation transfer is a major downstream application of our un-

supervised 3D shape-deformation disentanglement method. The deformation transfer task aims to transfer the deformation of one 3D object into another while keeping the same shape or identity. Deformation transfer methods (Song et al. 2023, 2021) directly infer the deformation transferred object. Most existing deformation transfer methods (Song et al. 2021; Wang et al. 2020; Sumner and Popović 2004) are supervised, using the "group truth" transferred mesh as the target. Very recently, a few unsupervised 3D deformation transfer methods (Song et al. 2023; Sun, Chen, and Kim 2023) have been developed. X-DualNet (Song et al. 2023) uses dual reconstruction and consistency losses similar to (Zhou, Bhatnagar, and Pons-Moll 2020) to perform unsupervised deformation transfer. MAPConNet (Sun, Chen, and Kim 2023), uses mesh-level and point-level contrastive learning. Our shape-deformation disentanglement method can also be used for the unsupervised deformation transfer through latent manipulation and 3D generation (details on Section ). However, unlike these methods, our method is generative and capable of various 3D shape analysis tasks through latent space manipulation and 3D generation.

Further discussions on several other related works can be found in the extended version.

# Method

Given a set of $N$ deforming 3D objects $\{x^{(i)}\}_{i=1}^{N}$ and their shape group information, the goal of our method is to learn disentangled latent codes for shape and deformation-specific information. Simultaneously, our method aims to learn a generator that allows controllable generation of 3D objects. Considering a 3D point-cloud or mesh input space $X$, shape space $S$, and deformation space $Z$ disentangled from $S$, our method learns the spaces $S$ and $Z$ as well as a mapping $g_\theta : S \times Z \rightarrow X$.

For any $(i, j)$, with $s^{(i)}, s^{(j)} \in S$ and $z^{(i)}, z^{(j)} \in Z$, the outputs $g_\theta(s^{(i)}, z^{(i)})$ and $g_\theta(s^{(j)}, z^{(j)})$ should satisfy:

$$\begin{cases} s^{(i)} = s^{(j)},\ z^{(i)} \neq z^{(j)} & \Leftrightarrow \text{same shape, distinct def.,} \\ s^{(i)} \neq s^{(j)},\ z^{(i)} = z^{(j)} & \Leftrightarrow \text{same def., distinct shape.} \end{cases}$$

## Overview

At a high level, our method is built on an auto-decoder (Huang et al. 2021) based architecture (Figure 1). For each point-cloud or mesh $x^{(i)}$ in the input space $X \subseteq \mathbb{R}^{N \times V \times 3}$, we learn a shape latent code $s^{(i)}$ in the shape space $S \subseteq \mathbb{R}^{N \times d_s}$ and deformation latent code $z^{(i)}$ in the deformation space $Z \subseteq \mathbb{R}^{N \times d_z}$, where $d_s$ and $d_z$ are dimensions of shape code and deformation code respectively and $V$ is the number of points in each $x^{(i)}$. Simultaneously, we learn the mapping $g_\theta$ as a decoder or generator network.

## Disentanglement of Shape and Deformation

Merely optimizing two latent codes with the autodecoder network does not result in the disentanglement of the latent codes. Specialized techniques are needed to ensure the shape code represents only shape information, and the deformation code represents only deformation information. As described below, this is achieved by processing the shape and deformation codes differently.

**Shape Code Optimization** If two 3D objects $x^{(i)}$ and $x^{(j)}$ have the same shape, their shape codes $s^{(i)}$ and $s^{(j)}$ should also be the same. This constraint is enforced through the shape group information. If two 3D objects belong to the same shape group, they are assigned the same shape identity label (Figure 1). While optimizing the shape latent code, we explicitly constrain the shape latent codes to be shared within the same shape group of 3D objects. Such explicit constraint makes it difficult to have any deformation information in the shape code. Additionally, optimizing the shape latent codes for each shape group directly, rather than inferring the shape code for every 3D object and then averaging the codes for each group to create a template, allows us to simply use random sampling for creating mini-batches during training.

**Deformation Code Optimization** For each 3D object $x^{(i)} \in X$ in the dataset, we optimize a deformation code $z^{(i)} \in Z$. Since information represented by the deformation code should be minimal and not exhibit shape-specific attributes, the code must be regularized. To this end, we use the following two ways to regularize the deformation codes.

First, we optimize the deformation latent codes with $L_2$ regularization. Such regularization encourages the values of the deformation latent codes to be small and near zero. Second, we add a Gaussian noise of zero mean and fixed variance to the deformation codes before passing them to the generator network $g_\theta$. This is unlike variational autoencoders (VAEs), where the variances are learned. This mechanism ensures the variance does not decrease to a very small value for any particular component and thus prevents partial posterior collapse (He et al. 2019).

**Generator Network Optimization** Our generator network $g_\theta$ consists of multiple fully connected linear layers with several adaptive instance normalization (AdaIN) layers. During training, we pass the deformation code with additive noise $z^{(i)} + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2 I)$ directly as input to the generator. However, for shape latent code $s^{(i)}$, we do not directly pass it to the generator $g_\theta$. Instead, we use it to predict the parameter values of the adaptive instance normalization (AdaIN) layers in $g_\theta$. We achieve this by passing the shape code to a modulator network $m_\phi$ that predicts the parameter values for the AdaIN layers in $g_\theta$. The parameters of AdaIN layers are used to scale and shift the input features. Further details on AdaIN layers are in the supplementary document.

The output of the final AdaIN layer is passed through a sequence of linear layers and activation functions which generates the output point cloud $y^{(i)}$ of size $\mathbb{R}^{N \times 3}$. The similarity between the output point cloud $y^{(i)}$ and the point cloud $x^{(i)}$ is maximized using a reconstruction loss $L_{\text{recon}}$. Overall, the loss function $L_1$ of our method at this stage becomes:

$$\begin{aligned} L_1 &= L_{\text{recon}}(y^{(i)}, x^{(i)}) + \lambda \|z^{(i)}\|_2^2 \qquad (1) \\ &= L_{\text{recon}}\big(g_\theta(z^{(i)} + \epsilon,\ s^{(i)}),\ x^{(i)}\big) + \lambda \|z^{(i)}\|_2^2, \\ &\quad \epsilon \sim \mathcal{N}(0, \sigma^2 I). \qquad (2) \end{aligned}$$

We optimize $\{z^{(i)}, s^{(i)}\}_{i=1}^{N}$ along with the parameters $\theta$ in $g_\theta$ and $\phi$ in $m_\phi$ using loss function $L_1$ and gradient descent.

### Inference of Shape and Deformation Codes

In the above steps, we optimize the shape and deformation codes for each 3D object in the input dataset. However, we should be able to infer the shape and deformation codes for any point cloud or mesh not present in the input dataset. To this end, in the second stage, we learn two inverse mappings $f_{\psi_s} : X \to S$ and $f_{\psi_z} : X \to Z$ with two encoder networks. Given a point-cloud or mesh as input $x^{(i)}$, the encoder $f_{\psi_s}$ outputs the corresponding shape code $s^{(i)}$ and the encoder $f_{\psi_z}$ outputs the corresponding deformation code $z^{(i)}$ that has been learned in the first stage. We ensure this with a distance loss $L_{\text{dis}}$ between the encoder predictions and the latent codes learned in the first stage. We also use a reconstruction loss to ensure that the encoder-predicted latent codes can be used to reconstruct $x^{(i)}$. Overall, the loss function $L_2$ in the second stage of our method becomes:

$$\begin{aligned} L_2 = {} & L_{\text{recon}}(g_\theta(f_{\psi_z}(x^{(i)}),\, f_{\psi_s}(x^{(i)})), x^{(i)}) \\ & + L_{\text{dis}}(f_{\psi_z}(x^{(i)}), z^{(i)}) + L_{\text{dis}}(f_{\psi_s}(x^{(i)}), s^{(i)}). \end{aligned} \tag{3}$$

Since the learned shape code and deformation code in the first stage are disentangled, the inferred latent codes also remain disentangled.

To deal with the permutation invariance of points in the input, we used the PointNet (Qi et al. 2017) architecture to implement the encoder networks $f_{\psi_z}$ and $f_{\psi_s}$. While more advanced architectures – PointNet++ (Qi et al. 2017), DGCNN (Wang et al. 2019), Point Transformer (Zhao et al. 2021), etc., exist for processing points, they are more effective for fine-grained tasks, *e.g.*, 3D scene segmentation, and do not provide any benefits in our disentanglement task.

### Implementation of Loss Functions

To implement $L_{\text{recon}}(y^{(i)}, x^{(i)})$ in Eq. 1 and Eq. 3, we used pairwise Euclidean distances between all points in the 3D object.

$$L_{\text{recon}}(y^{(i)}, x^{(i)}) = \left\| \mathbf{D}_{\mathbb{R}^3}(y^{(i)}) - \mathbf{D}_{\mathbb{R}^3}(x^{(i)}) \right\|_F^2 \tag{4}$$

Here, $\mathbf{D}_{\mathbb{R}^3}(x)$ is the matrix of pairwise Euclidean distances between all points in $x$, and $\| \cdot \|_F$ denotes the Frobenius norm of the matrix. Such reconstruction loss has also been used in (Cosmo et al. 2020) and have been found to be highly effective for objects with same connectivity.

## Experiments & Results

### Implementation Details

We used PyTorch to implement our method and the baselines. We trained and tested them on NVIDIA RTX A5000 GPUs. We used the Adam optimizer with a cosine annealing learning rate scheduler to optimize the latent codes and networks. We used a batch size of 6 for SMPL and 16 for SMAL and COMA, which had fewer vertices. In all our experiments, the generator $g_\theta$ used 5 AdaIN layers. Refer to the supplementary materials for further implementation details.

### Datasets

**SMPL-NPT:** This dataset (Wang et al. 2020) comprises 24,000 synthetic human meshes, each containing 6,890 vertices. It includes 30 shape identities represented in 800 unique deformations. For training, we draw 6,400 samples, covering 16 shapes and 400 deformations. For testing, following the literature (Song et al. 2023), we create a seen subset of 72 mesh pairs, sampled from the 14 shapes excluded from training and 400 deformations used in training. Similarly, we create an unseen subset of 72 mesh pairs, sampled from the remaining 14 shapes and 200 deformations which were excluded from the training data.

**SMAL:** This dataset (Zuffi et al. 2017) contains 24,600 synthetic animal meshes, each containing 3,889 vertices. It represents 41 shape identities, each in 600 unique deformations. For training, we randomly sample 9,000 meshes from 29 shapes and 400 deformations. For evaluation, similar to the literature (Song et al. 2023), an independent set of 400 unseen mesh pairs is used, taken from the 12 shapes and 200 deformations that were excluded from training.

**COMA:** This dataset (Ranjan et al. 2018) contains meshes of 12 human faces, each performing 12 different facial expressions. By splitting some expressions into left and right variants, we obtained 17 distinct expressions in total. 10 subjects were used for training and 2 for testing, resulting in a test set with seen deformations but unseen identities. Since not all expressions are available for every subject, the training set contains 162 meshes and the test set 34 meshes. All meshes share the same connectivity, with 5023 vertices each.

### Evaluation Metrics

We evaluated our method and the baseline methods in two different aspects:

1. How effectively can the disentangled shape and deformation codes be applied for unsupervised deformation transfer on unseen (zero-shot) shapes or deformations?
2. To what extent can the disentangled shape and deformation codes predict actual deformations, and how independent are the factors in this prediction?

**To assess criterion 1**, we utilize the evaluation metrics PMD and CD frequently used in the relevant literature. We define them as follows:

**PMD:** The average of the Euclidean distances between corresponding points in two point clouds.

$$L_{\text{PMD}}(y^{(i)}, x^{(i)}) = \frac{1}{N} \sum_{j=1}^{N} \|y_j^{(i)} - x_j^{(i)}\|^2 \tag{5}$$

**CD:** Measures the similarity between two point clouds by calculating the average distance from each point in one set to the closest point in the other set. It is in the same form as Eq. 6.

$$\begin{aligned} & L_{\text{CD}}(y^{(i)}, x^{(i)}) \\ & = M_{\alpha_c}\left( \frac{1}{|x^{(i)}|} \sum_{p \in x^{(i)}} \hat{d}(p), \frac{1}{|y^{(i)}|} \sum_{\hat{p} \in y^{(i)}} d(\hat{p}) \right) \end{aligned} \tag{6}$$

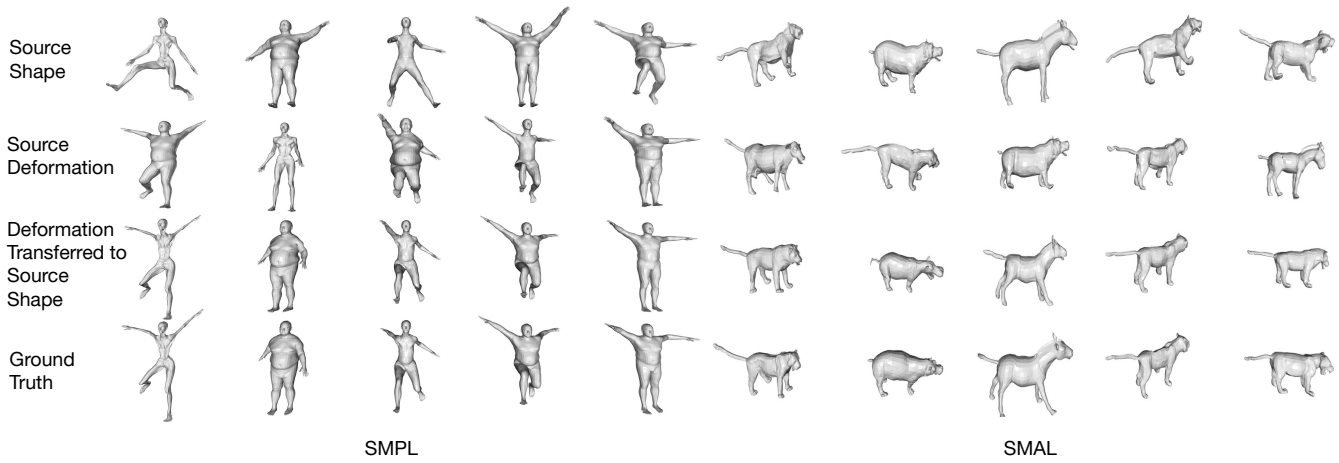


Figure 2: Unsupervised 3D deformation transfer in SMPL (left) and SMAL (right) datasets by our method. Additional visualizations can be found in the supplementary material.


Figure 3: Qualitative results of DiLO on COMA. Top row: identity sources (shape codes). Left column: expression sources (content codes). Middle: generated faces combining identity and expression, reflecting both source traits.

**For criterion 2**, we define the predictivity and disentanglement score for deformation and use them as the evaluation metric.

**Latent Space Disentanglement Score:** Measuring disentanglement of shape and deformation in latent space is inherently challenging. Nevertheless, a number of metrics, *e.g.*, MIG score, SAP, $D_{\text{score}}$, etc., have been proposed to this end in the disentangled representation learning literature (Locatello et al. 2019; Detlefsen and Hauberg 2019). Since (Locatello et al. 2019) showed that these scores are highly correlated, using any one of them often suffices. Following the image style disentanglement methods (Detlefsen and Hauberg 2019; Gabbay and Hoshen 2020), we used disentanglement score $D_{\text{score}}$ (Detlefsen and Hauberg 2019) as the metric to measure disentanglement in latent space. If there are two latent codes $lc_1$ and $lc_2$, then the $D_{\text{score}}$ for any ground truth factor is defined as:

$$D_{\text{score}}(\text{factor}) = |E(\text{factor}|lc_1) - E(\text{factor}|lc_2)| \qquad (7)$$

where $||$ denotes absolute value and $E(\text{factor}|lc)$ means the predictivity of factor given only latent code $lc$. The predictivity is usually measured with a simple linear classifier.

We further qualitatively assessed the efficacy of our method through explainability analyses using a surrogate model based explainability approach designed for 3D PointNet based neural networks. (Tan and Kotthaus 2022).

## Unsupervised 3D Deformation Transfer

Disentangled shape and deformation codes in 3D generative models can be readily used for deformation transfer, making it a natural way to evaluate disentanglement effectiveness. Given two 3D objects $x^{(s)}$ and $x^{(z)}$ where $x^{(s)}$ serves as the source shape and $x^{(z)}$ serves as the source deformation, the deformation transfer approach aims to predict the 3D object $x^{\text{new}}$ that has the shape of $x^{(s)}$ and deformation of $x^{(z)}$. Supervised 3D deformation transfer methods use the ground truth of the deformation transferred object $x^{\text{new}}$ during training, whereas unsupervised methods do not.

There exist two categories of unsupervised 3D deformation transfer methods- 1) unsupervised 3D methods directly predicting $x^{\text{new}}$ from the pair of objects $x^{(s)}$ and $x^{(z)}$ without disentanglement of latent codes and 2) unsupervised methods that disentangle shape and deformation code and can predict $x^{\text{new}}$ through latent manipulation and generation. In more detail, they infer the shape and deformation of latent codes with their encoder networks ($z = f_{\psi_z}(x^{(z)}), s = f_{\psi_s}(x^{(s)})$) and then predict $x^{\text{new}}$ using generator network by using codes $z$ and $s$ ($x^{\text{new}} = g_\theta(z, s)$).

The most recent state-of-the-art methods in category 1 are X-DualNet (Song et al. 2023) and MAPConNet (Sun, Chen, and Kim 2023). The existing unsupervised shape-deformation disentanglement methods leveraging the shape group information (Zhou, Bhatnagar, and Pons-Moll 2020; Cosmo et al. 2020; Chen et al. 2021b) along with our method all belong to the category 2.

We evaluated each method against the benchmark SMPL and SMAL datasets in unsupervised 3D deformation transfer experiments (Table 2). For the category 1 methods, we directly use their pretrained models to infer results on the benchmark test sets of SMPL and SMAL. For category 2

| Method | Time/epoch (mins) (↓) | Epochs trained trained | Total Training Time (↓) |
|---|---|---|---|
| Zhou et al. | 11.25 | 100 | 19 GPU hours |
| LIMP | 7.2 | 200 | 22 GPU hours |
| DiLO | **0.9** | 200 | **3** GPU hours |

Table 1: Computational cost comparison among DiLO and related methods

methods (Zhou, Bhatnagar, and Pons-Moll 2020; Cosmo et al. 2020; Chen et al. 2021b), we trained them ourselves on all the training datasets including COMA. However, we did not observe stable training with IEP-GAN (Chen et al. 2021b), most likely due to its use of GANs. A similar phenomenon was also reported by (Song et al. 2023). Consequently, we exclude it from the performance comparison among the methods. We report the training time of the category 2 methods in Table 1 which shows the sheer computational advantange of DiLO over other methods.

We conducted both qualitative and quantitative evaluations of unsupervised 3D deformation transfer on benchmark datasets. Quantitative results (Table 2) show that DiLO achieves performance comparable to or better than baseline methods, despite its simplicity and low computational cost. Although MAPConNet consistently yields the best Chamfer Distance (CD) on SMPL and SMAL, it requires deformation labels during training—unlike category 2 methods such as DiLO. Among category 2 methods, DiLO and the approach by (Zhou, Bhatnagar, and Pons-Moll 2020) perform similarly on SMPL and SMAL, but DiLO significantly outperforms (Zhou, Bhatnagar, and Pons-Moll 2020) on COMA. DiLO also consistently outperforms LIMP (Cosmo et al. 2020) across all datasets. Notably, both DiLO and LIMP use a simple encoder-decoder architecture with PointNet encoder, whereas (Zhou, Bhatnagar, and Pons-Moll 2020) employ a complex multi-scale mesh-based encoder-decoder.

We provide a few representative qualitative results obtained with our method in Figures 2 and 3. More qualitative visualizations of our method and the baselines are available in the supplementary. Nevertheless, the figure demonstrates our method's successful unsupervised 3D deformation transfer without using any form of correspondence learning or ground truth target objects during training.

## Disentanglement and Deformation Classification

To effectively measure disentanglement in latent space, we assessed how predictive the latent codes are for the ground truth deformations unseen in training. Thus, we estimated the $D_{\text{score}}(\text{def.})$ evaluation metric as described in Eq 7.

We evaluated our method and other unsupervised shape-deformation disentanglement methods on the SMPL and SMAL datasets. After estimating latent codes for both train and test datasets, we trained a linear support vector classifier (SVC) model with latent codes and their labels from training dataset. The model's prediction on the test dataset was used to report $\mathbf{E}(\text{Def.}|\mathbf{z})$ and $\mathbf{E}(\text{Def.}|\mathbf{s})$, with $\mathbf{D_{score}}(\text{Def.})$ being the absolute difference between them. We did not report $D_{\text{score}}(\text{shape})$ since shape identities were already used during the generative model training in our method and the

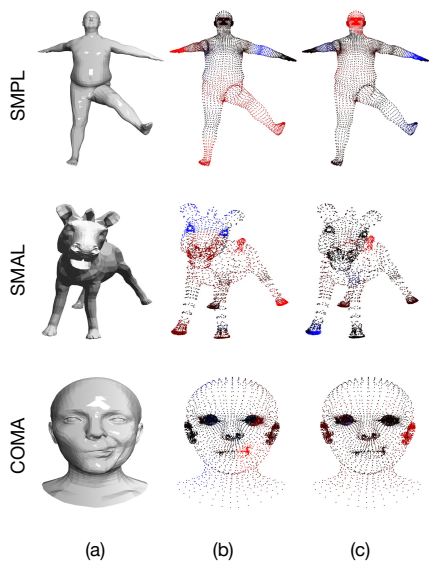


Figure 4: Results on explainability of DiLO. (a) A sample 3D mesh (b) Vertex importance learned by DiLO content encoder (c) Vertex importance learned by DiLO class encoder (Red = high importance, blue = low importance).

baseline methods. However, none used the deformation labels during training. It is important to note that the evaluation datasets for SMPL and SMAL used here differ from those in the deformation transfer experiments. Each evaluation dataset contains 400 random objects with unseen identities during training. For COMA, we use the same evaluation dataset used in the deformation transfer experiment.

We report the estimated values in Table 3, illustrating the superior latent space disentanglement achieved by our method compared to the baselines. It also demonstrates the efficacy of our method in accurately classifying deformation in 3D datasets using the inferred deformation latent code.

## Explainability Analyses

We further performed explainability analyses of our model using LIME3D (Tan and Kotthaus 2022). We assessed how the vertices at different location of the 3D objects in our datasets affects the shape encoder and the deformation encoder in DiLO. The details of how we leveraged LIME3D for this task is described in the supplementary document.

We provide the obtained results on Figure 4, which shows that our deformation encoder assigns high importance to the vertices that are affected by deformation. For instance, it focuses on legs and hands in SMPL; leg, hand, and mouth in SMAL; and lips in COMA. In contrast, our shape encoder assigns least importance to them. Instead, it prioritizes vertices associated with the identity of the 3D shapes- such as, face and body in SMPL, tail in SMAL, and ears in COMA.

These results indicate that our method not only produces feasible outputs but also makes rational and interpretable choices throughout the learning process. This enables DiLO to serve as a valuable tool for investigating regions of interest in 3D object datasets.

| Dataset | Category | Method | **PMD** ($10^{-3}$) (↓) | **CD** ($10^{-3}$) (↓) |
|---|---|---|---|---|
| SMPL (unseen identities, seen) deforms) | Deformation Transfer | X-DualNet (Song et al. 2023) | 0.82 | 1.27 |
| | | MAPConNet (Sun, Chen, and Kim 2023) | 0.52 | 1.02 |
| | Disentanglement | LIMP (Cosmo et al. 2020) | 20.21 | 32.24 |
| | | Zhou et al. (Zhou, Bhatnagar, and Pons-Moll 2020) | **0.06** | **0.18** |
| | | Ours (w/o latent optimization) | 4.276 | 12.9 |
| | | Ours (w/o AdaIN) | 5.86 | 21.40 |
| | | Ours | **0.06** | **0.18** |
| SMPL (unseen identities, unseen deforms) | Deformation Transfer | X-DualNet (Song et al. 2023) | 1.28 | 2.04 |
| | | MAPConNet (Sun, Chen, and Kim 2023) | **0.74** | **1.45** |
| | Disentanglement | LIMP (Cosmo et al. 2020) | 26.38 | 43.64 |
| | | Zhou et al. (Zhou, Bhatnagar, and Pons-Moll 2020) | 0.92 | 2.30 |
| | | Ours (w/o latent optimization) | 11.35 | 32.4 |
| | | Ours (w/o AdaIN) | 13.93 | 37.44 |
| | | Ours | 3.35 | 7.72 |
| SMAL | Deformation Transfer | X-DualNet (Song et al. 2023) | 4.36 | 8.18 |
| | | MAPConNet (Sun, Chen, and Kim 2023) | 3.66 | **6.94** |
| | Disentanglement | LIMP (Cosmo et al. 2020) | 26.77 | 24.97 |
| | | Zhou et al. (Zhou, Bhatnagar, and Pons-Moll 2020) | 3.46 | 7.02 |
| | | Ours (w/o latent optimization) | 9.66 | 23.6 |
| | | Ours (w/o AdaIN) | 6.61 | 13.14 |
| | | Ours | **3.45** | 7.40 |
| COMA | Disentanglement | LIMP (Cosmo et al. 2020) | 7.39 | 21.227 |
| | | Zhou et al. (Zhou, Bhatnagar, and Pons-Moll 2020) | 8.82 | 18.76 |
| | | Ours (w/o latent optimization) | 6.61 | 16.29 |
| | | Ours (w/o AdaIN) | 5.54 | 15.19 |
| | | Ours | **4.09** | **13.29** |

Table 2: Comparison of unsupervised deformation transfer accuracy for different methods on SMPL, SMAL, and COMA Datasets. PMD and CD are in units of $10^{-3}$. (↓) means lower values are better.

| Dataset | Method | $\mathbf{E}(\text{Def.}\|\mathbf{z})$ | $\mathbf{E}(\text{Def.}\|\mathbf{s})$ | $\mathbf{D}_{\textbf{score}}(\text{Def.})$ |
|---|---|---|---|---|
| SMPL | Zhou et al. | 0.918 | 0.085 | 0.833 |
| | LIMP | 0.960 | 0.940 | 0.020 |
| | Ours (w/o LO) | 0.991 | 0.185 | 0.806 |
| | Ours (w/o AdaIN) | 0.918 | **0.000** | 0.918 |
| | Ours | **1.000** | 0.003 | **0.997** |
| SMAL | Zhou et al. | 0.718 | 0.010 | 0.708 |
| | LIMP | 0.390 | 0.323 | 0.067 |
| | Ours (w/o LO) | 0.725 | 0.018 | 0.707 |
| | Ours (w/o AdaIN) | 0.933 | **0.003** | 0.930 |
| | Ours | **0.950** | 0.005 | **0.945** |
| COMA | Zhou et al. | 0.147 | 0.118 | 0.029 |
| | LIMP | 0.176 | 0.088 | 0.088 |
| | Ours (w/o LO) | 0.235 | 0.118 | 0.117 |
| | Ours (w/o AdaIN) | 0.375 | **0.063** | 0.312 |
| | Ours | **0.656** | **0.063** | **0.593** |

Table 3: Comparison of Deformation Probabilities for Different Methods on SMPL, SMAL, and COMA Datasets. For $\mathbf{E}(\text{Def.}|\mathbf{z})$ and $\mathbf{D}_{\textbf{score}}(\text{Def.})$ higher is better, and for $\mathbf{E}(\text{Def.}|\mathbf{s})$, lower is better. Def. and LO are abbreviations for deformation factor and latent optimization respectively.

Furthermore, the explainability is a unique advantage provided by DiLO over the baseline methods. The baseline LIMP (Cosmo et al. 2020) has limited explainability due to its use of a single PointNet encoder for jointly predicting shape and deformation. On the other hand, (Zhou, Bhatnagar, and Pons-Moll 2020) employs a multiscale mesh encoder-decoder architecture, where methods like LIME3D are not applicable. As a result, DiLO stands out from existing most relevant baselines by offering both strong performance, less computational cost, and practical explainability.

**Ablation Study:** We conducted ablation study to evaluate the contribution by individual components of DiLO, particularly 1) the latent optimization stage and 2) AdaIN layers in the generator. The ablative results provided in Table 2 and 3 underscores the importance of two-stage training and using adaptive instance normalization in our final method.

## Discussions & Conclusion

In this work, we proposed a novel method termed DiLO for parameterizing deforming grouped 3D objects into disentangled shapes and deformation factors in an unsupervised manner. Similar to other existing effective methods for unsupervised 3D shape-deformation disentanglement, it uses the group information of the 3D objects and performs disentangled latent optimization. In most practical applications, such shape grouping information is readily available. We empirically demonstrate the importance of our two-stage framework and the overall design of our method through an extensive ablation study. Our successful demonstrations of unsupervised 3D deformation transfer, deformation classification, and explainability analyses position our method as a promising practical tool to advance unsupervised 3D vision. As with all group-based disentanglement methods, DiLO assumes access to reliable shape-group information, and performance may degrade when group assignments are noisy or inconsistent – a problem worth exploring in the future.

## Acknowledgements

This work was supported in part by the U.S. NIH grant R35GM158094.

## References


Anguelov, D.; Srinivasan, P.; Koller, D.; Thrun, S.; Rodgers, J.; and Davis, J. 2005. Scape: shape completion and animation of people. In *ACM SIGGRAPH 2005 Papers*, 408–416. Association for Computing Machinery.

Aumentado-Armstrong, T.; Tsogkas, S.; Jepson, A.; and Dickinson, S. 2019. Geometric disentanglement for generative latent shape models. In *Proceedings of the IEEE/CVF international conference on computer vision*, 8181–8190.

Chen, H.; Shi, H.; Liu, X.; Li, X.; and Zhao, G. 2023. Smg: A micro-gesture dataset towards spontaneous body gestures for emotional stress state analysis. *International Journal of Computer Vision*, 131(6): 1346–1366.

Chen, H.; Tang, H.; Sebe, N.; Zhao, G.; et al. 2021a. AniFormer: Data-driven 3D Animation with Transformer. In *British Machine Vision Conference (BMVC'21)*, 1–13. BMVA.

Chen, H.; Tang, H.; Shi, H.; Peng, W.; Sebe, N.; and Zhao, G. 2021b. Intrinsic-extrinsic preserved gans for unsupervised 3d pose transfer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 8630–8639.

Corman, E.; Solomon, J.; Ben-Chen, M.; Guibas, L.; and Ovsjanikov, M. 2017. Functional characterization of intrinsic and extrinsic geometry. *ACM Transactions on Graphics (TOG)*, 36(2): 1–17.

Cosmo, L.; Norelli, A.; Halimi, O.; Kimmel, R.; and Rodola, E. 2020. Limp: Learning latent shape representations with metric preservation priors. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, 19–35. Springer.

Detlefsen, N. S.; and Hauberg, S. 2019. Explicit Disentanglement of Appearance and Perspective in Generative Models. In *33rd Conference on Neural Information Processing Systems*.

Gabbay, A.; and Hoshen, Y. 2020. Demystifying Inter-Class Disentanglement. In *International Conference on Learning Representations*.

Gabbay, A.; and Hoshen, Y. 2021. Scaling-up disentanglement for image translation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6783–6792.

Gao, L.; Chen, S.-Y.; Lai, Y.-K.; and Xia, S. 2017. Data-driven shape interpolation and morphing editing. In *Computer Graphics Forum*, volume 36, 19–31. Wiley Online Library.

He, J.; Spokoyny, D.; Neubig, G.; and Berg-Kirkpatrick, T. 2019. Lagging Inference Networks and Posterior Collapse in Variational Autoencoders. arXiv:1901.05534.

Huang, Q.; Huang, X.; Sun, B.; Zhang, Z.; Jiang, J.; and Bajaj, C. 2021. Arapreg: An as-rigid-as possible regularization loss for learning deformable shape generators. In *Proceedings of the IEEE/CVF international conference on computer vision*, 5815–5825.

Li, T.; Bolkart, T.; Black, M. J.; Li, H.; and Romero, J. 2017. Learning a model of facial shape and expression from 4D scans. *ACM Trans. Graph.*, 36(6): 194–1.

Locatello, F.; Bauer, S.; Lucic, M.; Raetsch, G.; Gelly, S.; Schölkopf, B.; and Bachem, O. 2019. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, 4114–4124. PMLR.

Loper, M.; Mahmood, N.; Romero, J.; Pons-Moll, G.; and Black, M. J. 2023. SMPL: A skinned multi-person linear model. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 851–866. Association for Computing Machinery.

Ploumpis, S.; Wang, H.; Pears, N.; Smith, W. A.; and Zafeiriou, S. 2019. Combining 3d morphable models: A large scale face-and-head model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10934–10943.

Pons-Moll, G.; Romero, J.; Mahmood, N.; and Black, M. J. 2015. Dyna: A model of dynamic human shape in motion. *ACM Transactions on Graphics (TOG)*, 34(4): 1–14.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Ranjan, A.; Bolkart, T.; Sanyal, S.; and Black, M. J. 2018. Generating 3D faces using convolutional mesh autoencoders. In *Proceedings of the European conference on computer vision (ECCV)*, 704–720.

Romero, J.; Tzionas, D.; and Black, M. J. 2022. Embodied hands: Modeling and capturing hands and bodies together. *arXiv preprint arXiv:2201.02610*.

Rustamov, R. M.; Ovsjanikov, M.; Azencot, O.; Ben-Chen, M.; Chazal, F.; and Guibas, L. 2013. Map-based exploration of intrinsic shape differences and variability. *ACM Transactions on Graphics (TOG)*, 32(4): 1–12.

Song, C.; Wei, J.; Li, R.; Liu, F.; and Lin, G. 2021. 3d pose transfer with correspondence learning and mesh refinement. *Advances in Neural Information Processing Systems*, 34: 3108–3120.

Song, C.; Wei, J.; Li, R.; Liu, F.; and Lin, G. 2023. Unsupervised 3d pose transfer with cross consistency and dual reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8): 10488–10499.

Sumner, R. W.; and Popović, J. 2004. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23(3): 399–405.

Sun, J.; Chen, Z.; and Kim, T.-K. 2023. MAPConNet: Self-supervised 3D Pose Transfer with Mesh and Point Contrastive Learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 14452–14462.

Tan, H.; and Kotthaus, H. 2022. Surrogate model-based explainability methods for point cloud nns. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2239–2248.

Tan, Q.; Gao, L.; Lai, Y.-K.; and Xia, S. 2018. Variational autoencoders for deforming 3d mesh models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5841–5850.

Wang, J.; Wen, C.; Fu, Y.; Lin, H.; Zou, T.; Xue, X.; and Zhang, Y. 2020. Neural pose transfer by spatially adaptive instance normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5831–5839.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5): 1–12.

Zhao, H.; Jiang, L.; Jia, J.; Torr, P. H.; and Koltun, V. 2021. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, 16259–16268.

Zhou, K.; Bhatnagar, B. L.; and Pons-Moll, G. 2020. Unsupervised shape and pose disentanglement for 3d meshes. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, 341–357. Springer.

Zuffi, S.; Kanazawa, A.; Jacobs, D. W.; and Black, M. J. 2017. 3D menagerie: Modeling the 3D shape and pose of animals. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6365–6373.