GLASS: Geometric Latent Augmentation for Shape Spaces

Sanjeev Muralikrishnan[1], Siddhartha Chaudhuri[2,3], Noam Aigerman[2], Vladimir G. Kim[2], Matthew Fisher[2], and Niloy J. Mitra[1,2]

[1]University College London, [2]Adobe Research, [3]IIT Bombay

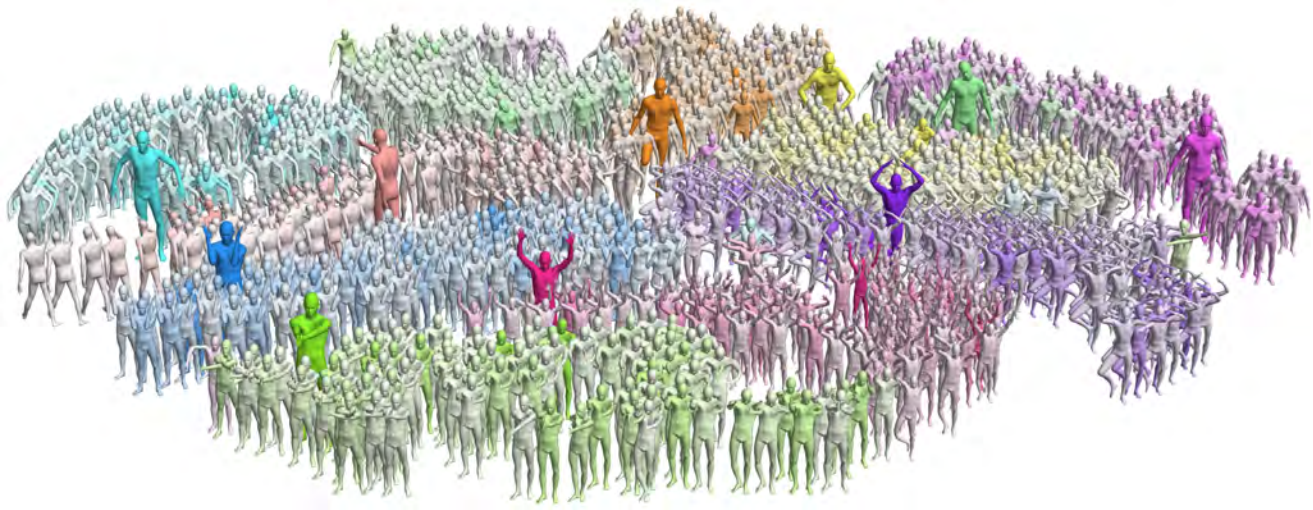

Figure 1. Starting from just 10 shapes (larger), our method iteratively augments the collection by alternating between training a VAE, and exploring random perturbations in its low-dimensional latent space guided by a purely geometric deformation energy. Here we show the 1000 most diverse shapes from the first 5K discovered by our method, positioned according to their latent embedding (projected to 2D via t-SNE). Shapes are colored according to the initial landmark they trace back to, with shapes added in later iterations lighter (greyer) in color. The augmentation effectively fills in the space between the sparse initial landmarks, and even extrapolates beyond them. It manages to also interpolate global rotations for samples near the back-facing exemplar, and yields shapes with larger feet-strides (far left), and crossed arms or feet (front, left and center) even though there are no such initial landmarks.

Abstract

We investigate the problem of training generative models on very sparse collections of 3D models. Particularly, instead of using difficult-to-obtain large sets of 3D models, we demonstrate that geometrically-motivated energy functions can be used to effectively augment and boost only a sparse collection of example (training) models. Technically, we analyze the Hessian of the as-rigid-as-possible (ARAP) energy to adaptively sample from and project to the underlying (local) shape space, and use the augmented dataset to train a variational autoencoder (VAE). We iterate the process, of building latent spaces of VAE and augmenting the associated dataset, to progressively reveal a richer and more expressive generative space for creating geometrically and semantically valid samples. We evaluate our method against a set of strong baselines, provide ablation studies, and demonstrate application towards establishing shape correspondences. GLASS *produces multiple interesting and meaningful shape variations even when starting from as few as 3-10 training shapes. Our code is available at* https://sanjeevmk.github.io/glass_webpage/.

1. Introduction

This paper is concerned with generating plausible deformations of a 3D shape from a very sparse set of examples. Fig. 1 shows an input of 10 human 3D meshes in different poses, and the additional deformations generated by our method. 3D deformations have a strong *semantic* element to them – e.g., a human's limbs should only bend at the joints, and then, under normal circumstances, not beyond certain angular ranges. Arguably, this can only be deduced in general via learning by example from a dataset.

Unfortunately, in contrast to 2D images, the 3D domain poses several challenges for data-driven frameworks. Probably the most significant one is that data acquisition is complex and tedious, making datasets both scarcer and sparser.

Given this data paucity, we tackle the challenge of generating additional meaningful deformations from a given (very) sparse set of landmark deformations. Our method meaningfully augments the sparse sets to create larger datasets that, in turn, can be leveraged by other techniques that cannot operate on sparse datasets.

Producing plausible deformations from a few landmarks is difficult. Linearly interpolating the vertices of two land-

marks yields highly implausible intermediates. A key insight is that while meaningful deformations are semantic, they often have a very strong pure-geometric element, e.g., they are smooth (i.e., preserve local details) and don't distort the shape too much (i.e., local distances are preserved). However, simply perturbing vertices while minimizing a geometric energy (e.g., smoothness or metric distortion)

generates artifacts such as smooth global bending or surface ripples because by itself, the energy is not a sufficient constraint. Interpolating landmark pairs, while preserving the energy, fares better but produces limited variations [1, 25]. Our paper, like other recent approaches [12, 20], advocates *learning a low-dimensional generative latent space* which maps out the underlying manifold *jointly* defined by the landmarks, while simultaneously minimizing a deformation energy. However, these prior methods still require a large dataset to learn a rich set of variations.

Our core contribution is to address this difficulty with a novel *data augmentation* approach that alternates between latent space training and *energy-guided exploration*. We employ *supervised* learning of a generative space from a training dataset, a very sparse one, but augment that set in an *unsupervised*, geometry-aware way. Specifically, we train a Variational Autoencoder (VAE) on the given dataset.

After training, we use the eigenmodes of a deformation energy's Hessian to *perturb and project* latent codes of training shapes in a way that ensures they yield smooth, low-distortion deformations, which we add back as data augmentation to the input set. We then re-train the VAE on the augmented dataset and repeat the process iteratively until the space has been densely sampled. In addition to reducing spurious deformations, the use of a low-dimensional, jointly-trained latent space allows low-energy perturbations of one landmark to be influenced by other landmarks, yielding richer variations. We call our method GLASS.

We evaluate GLASS on several established datasets and compare performance against baselines using multiple metrics. The experiments show the effectiveness of GLASS to recover meaningful additional deformations from a mere handful of exemplars. We also evaluate the method in the context of shape correspondence, demonstrating that our sampling process can be used as a data augmentation technique to improve existing strongly-supervised correspondence algorithms (e.g., 3D-CODED [18]).

2. Related Work

Geometric shape deformation. *Parametric* deformation methods express 2D or 3D shapes as a known function of a set of common parameters, and model deformations as variations of these parameters. Such methods include cages [24], blendshapes [29], skinned skeletons [23] and Laplacian eigenfunctions [32]. In contrast, *variational* methods model deformations as minimizers of an energy functional – e.g. Dirichlet [19], isometric [25], conformal [28], Laplacian [7], As-Rigid-As-Possible (ARAP) [36], or As-Consistent-As-Possible (ACAP) [15] – subject to user constraints. In our work we focus on minimizing the ARAP energy, although our method supports any twice-differentiable energy function. There are strong connections between the parametric and variational approaches, for instance biharmonic skinning weights [22] (parametric) are equivalent to minimizing the Laplacian energy (variational). Please see surveys [27, 42] for a complete discussion. We are also inspired by work on modal analysis [21], which linearize the deformation space of a shape in terms of the least-significant eigenvectors of the Hessian of some energy functional. In the current paper, we effectively perform *learned non-linear modal analysis*: starting with a variational formulation – the implicitly-defined manifold of low-energy perturbations of a few landmark shapes – we *learn* the corresponding parametric representation as the latent space of an autoencoder by iteratively exploring locally linear perturbations.

Our work on data augmentation from a sparse set of landmark shapes is related to *interpolation/morphing* between, and *extrapolation* from, sets of shapes. As in our scenario, the set typically comprises articulations of a common template. See e.g. [39] for a survey of classical (non-learning-based) methods for shape interpolation. Plausible extrapolation is less well-defined, and less studied, in the classical literature. Kilian et al. [25] extend geodesics of an isometric energy in the deformation space, though it is restricted to exploring (and extrapolating) paths between shapes rather than the full deformation space.

Learned deformation models. Various types of generative models based on graphical models, GANs, VAEs etc have been developed to probabilistically synthesize shape variations. A full treatment is beyond the scope of this paper, please see surveys such as [11]. Here, we focus on models which capture the space of smooth deformations of a given shape. The best-studied domain is that of virtual humans, beginning with seminal works capturing face [4], bodyshape [2] and pose [3] variations in a data-driven fashion from scanned exemplars. These works, like several subsequent ones, rely on variations of principal component analysis (PCA) to parameterize the deformation space. Yumer et al. [43] learn a common set of deformation handles for a dataset. More recent work uses deep neural networks to learn shape deformation models from training sets [13,16,31,37,40], and use them for applications such as non-rigid correspondences [18]. Tan et al. [38] and Huang et al. [20] regularize a VAE with an energy-based loss. We use the latter [20] as our choice for energy. However, the primary role of the energy in our method is to guide exploration for data augmentation.

Crucially, all the above methods rely on extensive train-

ing data. In contrast, we specifically aim to learn meaningful data-driven deformation models under *extreme sparsity constraints*, from just a handful of landmarks indicating modes of the distribution. While this is broadly related to few-shot learning scenarios, only a few other papers consider these requirements in the context of geometric shape synthesis, or without any auxiliary data from other domains. LIMP [12] is an important recent work that tries to regularize the latent space of a 3D shape VAE by requiring points sampled on the line segment between two latent codes to minimize geometric distortion relative to the endpoints. Unlike our method, LIMP does not explore the full volume of the hull bounding the training landmarks, or extrapolate beyond it – regularization is limited to the web of pairwise paths. We modified LIMP to work with ARAP energy, and demonstrate that our method significantly outperforms their approach on a variety of metrics.

Unsupervised data augmentation. Our work is part of a wide class of methods for synthetically increasing the size of training datasets for data-hungry machine learning, without additional supervision. For broad coverage, we refer the reader to surveys on images [35], time series [41], and NLP [10]. A particularly relevant recent technique is Deep Markov Chain Monte Carlo [33], which samples perturbations of training data using MCMC on an energy functional, trains an autoencoder on these samples, and uses the resulting latent space for lower-dimensional (and hence faster) MCMC. We observed that on very sparse and high-dimensional datasets (only a few landmark 3D shapes), the initial samples of Deep MCMC do not capture meaningful variations, and hence it does not adequately augment the dataset. Also related are methods that augment classification datasets with adversarial perturbations along the gradients of loss functions [17, 34]. In contrast, we seek to *preserve* an energy-based loss, and hence *eliminate* the gradient and other high-change directions from consideration.

3. Method

3.1. Problem Setup

We assume all shapes in a particular input dataset are meshes with consistent topology. Given a mesh with N vertices $V \in \mathbb{R}^{N\times 3}$ and triangle faces T, a mesh deformation is simply an assignment of a new position to each vertex, denoted as $W \in \mathbb{R}^{N\times 3}$. We consider the input dataset itself as deformations of a base topology and we are given a sparse set of n deformation "examples", $W^1, \dots W^n$. We assume access to a deformation energy $f(W, W')$ which measures the distortion of candidate deformation W with respect to an exemplar deformation W', with higher values indicating more severe distortion induced by the candidate. For brevity, we omit W' and simply write $f(W)$ to mean energy with respect to the relevant base shape. We use the As-

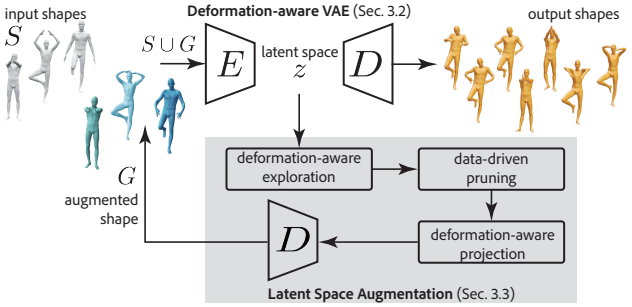

Figure 2. We present GLASS to iteratively build a deformation-aware VAE latent space and analyzing it to generate new training samples to augment the original training set. This enables generation of diverse yet plausible shape variations starting from very few input examples.

Rigid-As-Possible (ARAP) energy [36] and its latent-space approximation ARAPReg [20] to measure the deviation of a deformation from isometry, i.e., how much do geodesic lengths change with respect to the rest pose V.

We devise a subspace-sampling strategy that adheres to two properties: (i) it should be data-driven, and contain deformations from the given sparse set; and (ii) it should be geometrically-meaningful, i.e., the deformations should have low energy, wrt the given deformation energy $f(W)$.

Our main contribution is a method for online data augmentation during the training of a variational autoencoder (VAE) [26]. Namely, during training, our method explores the current sample space, guided by the deformation energy $f(W)$, to discover additional meaningful deformation samples. These are progressively used as additional sample points to form an augmented dataset that is used to retrain the VAE, and the process is iterated until convergence.

3.2. Deformation-Aware VAE

Let $E : \mathbb{R}^{N\times 3} \to \mathbb{R}^K$ be the encoder in the standard VAE architecture, mapping a deformation W into vectors of mean μ and variance Σ into a distribution $E(W) \sim \mathcal{N}(\mu, \Sigma)$. These vectors define the mean and variance of a multivariate Gaussian distribution from which the latent code z, of dimension K, is sampled, i.e., $z \sim \mathcal{N}(\mu, \Sigma)$. Similarly, let $D : \mathbb{R}^K \to \mathbb{R}^{N\times 3}$ be the decoder mapping the latent code to a deformation, $D(z) = W$. We shall slightly abuse notation and use $D(E(W))$ to denote the full autoencoding process of W, including the step of sampling from the Gaussian. We define three losses to be used in training.

(i) Reconstruction Loss: We require that the VAE reduces to an identity map for any sample deformation,

$$L_{\text{Reconstruction}} := \|D(E(W)) - W\|^2. \tag{1}$$

(ii) Gaussian Regularizer Loss: Instead of the standard the KL divergence regularizer used in VAEs, we constrain the sample mean and covariance of the mini-batch to that of a unit Gaussian, as proposed in [14]. We found that for a

Algorithm 1 Pseudocode for searching the latent space, starting from deformation W, for a new augmenting shape.

procedure LATENTAUGMENT(W,E,D,f,R)
▷ E = Encoder, D= Decoder
▷ W = Initial deformation, f = Energy
▷ R = All previously input or generated shapes
$l = E(W)$ ▷ latent code
$\overline{H} \approx \nabla_l \nabla_l f(D(l))$ ▷ approximate Hessian
$\lambda, U^{\uparrow}(H) = \text{EigenDecomposition}(H)$
$\lambda_k, U_k^{\uparrow}(H) \leftarrow \lambda, U^{\uparrow}(H)$ ▷ retain k components
$W_d = \emptyset$
for j ∈ [1,s]: **do**
$\beta \sim \mathcal{N}(0, \mathbb{I}) \in R^k$ ▷ sample β, $k \ll K$
$\hat{\beta} = \beta / \sum_{i=1}^{k} \beta_i^2$
$\alpha = \sqrt{2\delta / \sum_{i=1}^{i=k} \hat{\beta}_i^2 \lambda_i}$
$\hat{W}_j = D(l + \alpha \sum_{i=1}^{k} \hat{\beta}_i U_i^{\uparrow}(H))$
$W_d \leftarrow W_d \cup \hat{W}_j$ ▷ add to candidates
end for
$W^* = \text{MMR}(W, R, W_d)$ ▷ prune candidates
$W_{\text{Projected}} = \arg\min f(W^*)$ ▷ project
$R \leftarrow R \cup \{W_{\text{Projected}}\}$ ▷ augment training set
end procedure

small sample size this batch-based loss leads to faster convergence, compared to the standard KL-Divergence. We denote this loss as,

$$L_{\text{Gaussian}} := \frac{1}{b} \sum_{i=1}^{b} \left(\|\mu_i\|^2 + \|\Sigma_i - \mathbb{I}\|^2 \right), \tag{2}$$

where b is the mini-batch size, μ_i, Σ_i are the predicted mean and covariance for the i-th sample in the mini-batch, and $\mathbb{I}$ is the identity matrix.

(iii) Deformation Energy: Lastly, we require the resulting deformation to have low deformation energy,

$$L_{\text{Deformation}} := f(D(E(W))). \tag{3}$$

In summary, our network training loss is

$$L := L_{\text{Reconstruction}} + L_{\text{Gaussian}} + \sigma L_{\text{Deformation}} \tag{4}$$

where σ is a scalar weight applied to the energy function.

3.3. Augmenting via Latent Space Exploration

We now describe the main step of our technique – adding additional deformation examples W^j to the latent space to reinforce training (Algorithm 1). Simply optimizing (4) is not enough to cover the deformation space. Instead, we continuously introduce new low-energy deformations into the training set, by which we make the *data term aware of the deformation energy*. We achieve this through three steps: (i) deformation-aware *perturbation* of the latent code in directions that locally *least* modify the deformation energy; (ii) data-driven *pruning* of perturbed codes that do not introduce variance to the current dataset; and (iii) deformation-aware *projection* of the new codes to further lower their deformation energy, with an optional *high-resolution projection* to transfer the deformation from a low-resolution mesh to a higher resolution one. Figure 3 illustrates how the latent space is progressively populated with new deformations over iterations, where colors indicate the base shapes.

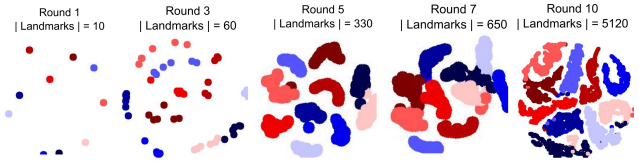

Figure 3. tSNE embedding of generated samples shows progressive augmentation of the shape space. Sample color indicates originating (parent) shape. See also Fig. 1.

(i) Deformation-aware perturbation in latent space. Our goal is to create variations of a given code in latent space without modifying its deformation energy significantly: let W be a deformation, and $l = E(W) \in \mathbb{R}^K$ a latent code achieved from encoding it. We aim to find the low-energy perturbation modes of l. In short, we aim to perturb the deformation while not changing the deformation energy too much, or in other words – we wish to stay on the current level set of the energy. To achieve that, we can restrict ourselves to perturbations on the local *tangent space* of the energy's level set. This tangent space simply comprises of all directions orthogonal to the gradient $\nabla_l f(D(l))$.

In the tangent space, we can pick better directions using a second order analysis. Let H denote the Hessian of the deformation energy with respect to the latent code,

$$H := \nabla_l \nabla_l f(D(l)). \tag{5}$$

Let λ_i, and $U_i^{\uparrow}(H)$ respectively denote the eigenvalues and eigenvectors of H, in ascending order of eigenvalues. Since smaller eigenvalues correspond to directions of less change in energy, we retain only the k ($k << K$) smallest λ_i and $U_i^{\uparrow}(H)$. We then draw a random perturbation, by sampling a random vector $\beta \in \mathbb{R}^k$ from a normal distribution. Each $\beta_i \in \beta$ corresponds to step along the eigenvector $U_i^{\uparrow}(H)$. We normalize β to $\hat{\beta}$ such that $\sum_{i=1}^{k} \hat{\beta}_i U_i^{\uparrow}(H)$ is a unit vector, i.e., $\sum_{i=1}^{k} \hat{\beta}_i^2 = 1$. We then take a step,

$$l_t := l + \alpha \sum_{i=1}^{k} \hat{\beta}_i U_i^{\uparrow}(H), \tag{6}$$

where α denotes the step-size and l_t is in the tangent plane. We repeat this process s times for each latent code l to get s perturbed codes $\tilde{l}^1, \tilde{l}^2, \ldots \tilde{l}^s$. Let $\{\tilde{W}^1, \tilde{W}^2 \ldots \tilde{W}^s\}$ denote the decoded perturbed deformations where $\tilde{W}^j = D(\tilde{l}^j)$.

Variable step size: Different regions of the latent space have different local deformation landscapes, e.g., curvatures along different directions on the tangent plane. Hence, we

should adapt α to the nature of the local landscape around l. To that end, we formulate the step size α in terms of the eigenvalues λ_i and a user-prescribed threshold δ on the allowed deformation energy f (i.e., $f() \leq \delta$). Assuming C2 continuity for deformation energy $f()$, we can obtain the following bound on the step size (see supplemental),

$$\alpha \leq \sqrt{\frac{2\delta}{\sum_{i=1}^{k} \hat{\beta}_i^2 \lambda_i}}.$$

This gives an upper bound on the step size along any deformation direction.

(ii) Data-driven pruning of the perturbed deformations. In order to add diverse samples, given the set of candidate deformations $\{\tilde{W}^j\}$, we select one example to be added to the dataset, via Maximal Marginal Relevance (MMR) ranking [9]. Specifically, MMR gives a higher score to perturbations that are similar to the unperturbed W, but different from the existing deformations set R, containing the landmarks and deformations generated so far. We compute as,

$$F(w) = \gamma M(w, W) - (1 - \gamma) \max_{r \in R} M(w, r), \quad (7)$$

where $M(x, y)$ is the cosine similarity between x and y. Thus, we choose the deformation $W^* \in W_d$ that maximizes the MMR function with its latent code denoted l^*.

(iii) Deformation-aware projection to smooth, low-energy deformations. Although the deformation-aware perturbation somewhat avoids high-energy states, the perturbed deformation W^* may still exhibit undesirable artifacts such as lack of smoothness or high deformation energy. Hence, we project the code to have lower deformation energy *with respect to the unperturbed* W. We achieve this by treating W as the rest pose, defining a deformation energy with respect to it, f_W. We perform a fixed number of gradient descent steps starting from W^* to lower the energy, which yields the final deformation $W_{\text{Projected}}$. In our experiments, we optimize up to the threshold of 10^{-5}.

(iv) Augment and iterate: Finally, we append the newly generated deformations to the current (training) set, and continue training. We repeat this augmentation and retraining several times, until we reach a target training set size.

Implementation Choices

Choice of energy f. For training the VAE, we set f as the $L2$ formulation of the ARAPReg energy [20]. ARAPReg is an efficient approximation of the ARAP energy [36] that is conducive for network training. This energy computes an approximate Hessian $\overline{H}$ of ARAP with respect to the latent space, and directly minimizes the eigenvalues of $\overline{H}$.

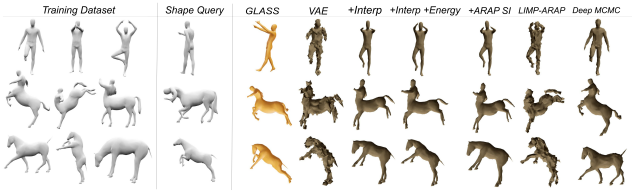

Figure 4. *Generation results evaluated by coverage.* We train different methods on the same training data (col 1) and generate comparable numbers of shapes. Given two shapes from the hold-out data (col 2), we evaluate the methods by finding the closest generated shape (cols 3-9). Note how the baselines exhibit strong artifacts and usually do not match the query shape.

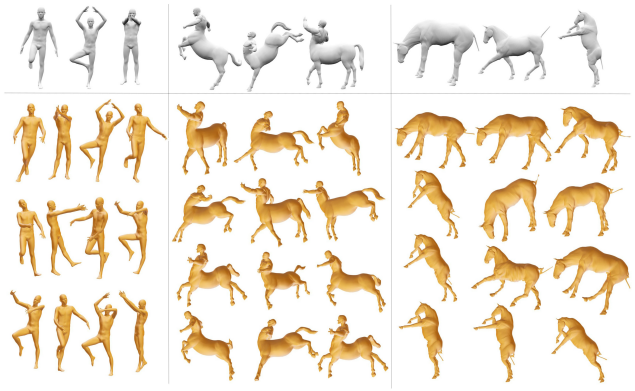

Figure 5. Training GLASS on the human, centaur, and horse meshes using the 3 examples each (top). (Bottom) We show random samples from the latent space, which combine different properties learned from the example deformations.

Approximation of Hessian. Similarly, for step (i) in Section 3.3, we use the approximate Hessian $\overline{H} \approx J^T H J$ proposed in [20] in our Equations 5 and 6, where H is the exact Hessian of ARAP and J is the stochastic Jacobian of the ARAP with respect to the $\mathbb{R}^K$ latent space.
The above choices speed up training and yield better results than classical ARAP. We still use ARAP for step (iii) where Hessian approximation is not needed.
Upper bound for α. The eigenvalues allow us to choose an appropriate local step-size, but they only yield a local approximation of the deformation energy. As our method continues adding low-energy shapes, the eigenvalues become lower, leading to an extremely large step size α. Thus, we set an upper bound of 2 on α.
High-resolution projection. To speed up training and avoid deformations that are too high-frequency, we use low-resolution meshes (low vertex count). We decimate the high-resolution meshes by preserving a subset of the vertices, thus retaining correspondence from low to high. The generated low-res deformations can later be projected back to original high-res meshes by treating the chosen subset of vertices as deformation constraints in the ARAP optimization proposed in [36].

4. Experiments

We evaluate GLASS on public data of humans (FAUST [5]) and creatures (TOSCA [8]) in different poses. In our experiments, we sample X landmark poses of a model from a dataset and train our method. We evaluate quality and diversity of newly generated poses as well as interpolation sequences between landmark poses. We denote our experiments as "SubjectName-X" to indicate the type of the subject and the number of landmark poses provided as an input to our method; most results use between 3 and 10 landmarks.

Evaluation metrics. We use three metrics - *Coverage*, *Mesh smoothness* and *Interpolaiton smoothness* - to evaluate our performance. For each metric, lower values are better. The metrics are detailed in the supplemental.

Method Comparison While existing methods are not designed to learn generative latent spaces from sparse data, we adapt them as baselines. Since ARAP projection to high resolution is specific to our method (see 3.3), we compare methods using the low-res version that we train with.

(i) *Vanilla VAE:* We train a VAE using only the sparse set of shapes, with no data augmentation. (ii) *+Interpolation:* We generate new shapes by interpolating between all pairs of available landmarks by simply averaging coordinates of corresponding vertices. We interpolate such that the amount of augmented data is equivalent to what is generated with GLASS (2500 shapes). Then we train a VAE with these poses. (iii) *+Interp. +Energy:* This is an extension of the previous method. Since raw interpolation can deviate from the underlying shape space, in addition to interpolation, we perform projection by minimizing the sum of ARAP energies with respect to both shapes in the pair. (iv) *LIMP-ARAP [12]:* This method is motivated by the training strategy proposed in LIMP [12]. They train a VAE with pairs of shapes in every iteration - for each pair, they pick a random latent code on the line between the two, decode it to a new shape, and minimize its energy. Since we only want to compare augmentation strategies we adapt

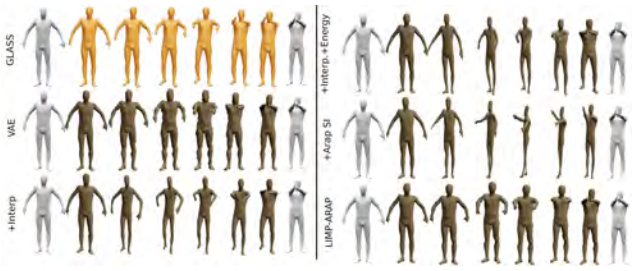

Figure 6. We compare the interpolation results between our method, several ablations of our method, and prior work.

Figure 7. *Interpolation results.* In gray, we show two landmark shapes. In gold, we show the decoded meshes after we linearly interpolate the latent space between these two landmarks. All models are trained on only 5 landmarks.

Figure 8. In gray, are 2 subsets of 3 and 6 facial expressions from the COMA dataset. Training GLASS on each of them, correspondingly generates the novel expressions in gold.

LIMP to use ARAP energy. (v) + *ARAP Shape Interpolation [1]:* This method morphs a shape from a source pose to a target pose. The morph is rigid in the sense that local volumes are least-distorting as they vary from their source to target configurations. (vi) *Deep-MCMC [33]:* This method explores parameter variations via a latent space and generate samples by performing HMC steps in the latent space and decoding the generated codes.

Generation Experiments After training, we sample latent codes from a Unit Gaussian in $\mathbb{R}^K$, and decode with our decoder to generate samples (see Figures 1, 5 and 8). Our generated poses look substantially different from the training data and combine features from multiple input examples.

Table 1. Surface smoothness and coverage with respect to excluded set, of generated samples. Lower is better.

Data	Vanilla VAE	+Interpolation	+Interp. +Energy	LIMP	Deep-MCMC	+ARAP SI [1]	GLASS
Faust-3	1.0 / 1.0	0.73 / 0.96	0.75 / 0.89	1.06 / 1.01	1.04 / 0.95	0.77 / 1.09	**0.63 / 0.77**
Faust-5	1.0 / 1.0	0.65 / 1.04	0.62 / 0.88	1.0 / 0.96	1.06 / 0.94	0.62 / 0.97	**0.59 / 0.78**
Faust-7	1.0 / 1.0	1.00 / 1.51	0.57 / 1.61	1.07 / 0.93	1.01 / 0.96	0.85 / 1.84	**0.54 / 0.81**
Centaurs-3	1.0 / 1.0	0.83 / 0.96	0.85 / 0.94	1.03 / 0.97	1.04 / 0.98	0.85 / 0.99	**0.69 / 0.84**
Centaurs-4	1.0 / 1.0	0.81 / 0.97	0.84 / 0.95	1.01 / 0.99	1.08 / 0.96	0.81 / 1.0	**0.69 / 0.84**
Horses-3	1.0 / 1.0	0.66 / 0.95	0.73 / 0.91	1.06 / 1.03	1.02 / 1.05	0.65 / 0.94	**0.61 / 0.89**
Horses-4	1.0 / 1.0	0.65 / 0.94	0.68 / 0.95	1.03 / 0.98	1.04 / 1.04	0.62 / 1.0	**0.60 / 0.80**

Table 2. L2 Error wrt excluded DFaust frames and reconstruction error of those excluded frames. Lower is better.

Data	Vanilla VAE	+Interpolation	+Interp. +Energy	LIMP	+ARAP SI [1]	GLASS
DFaust-1	1.0 / 1.0	0.48 / 0.51	0.48 / 0.40	0.53 / 0.56	0.94 / 0.92	**0.45 / 0.40**
DFaust-2	1.0 / 1.0	0.56 / 0.48	0.51 / 0.42	1.06 / 1.07	0.84 / 0.81	**0.47 / 0.40**
DFaust-3	1.0 / 1.0	0.61 / 0.57	0.51 / 0.44	1.08 / 2.0	0.77 / 0.8	**0.45 / 0.41**
DFaust-4	1.0 / 1.0	0.59 / 0.35	0.57 / 0.5	0.84 / 1.73	0.87 / 1.14	**0.46 / 0.32**
DFaust-5	1.0 / 1.0	0.27 / 0.38	0.3 / 0.27	0.53 / 1.28	0.67 / 1.2	**0.22 / 0.24**

We compare our approach to all the baselines. We sample from the unit Gaussian for all VAE-based techniques, where the only exception is Deep-MCMC where we use latent-space HMC as proposed in their work. We show qualitative results in Figure 4 and quantitative evaluations in Table 1. Each cell reports smoothness and coverage errors, normalized based on the corresponding errors for Vanilla-VAE. Note that our method outperforms all baselines in its ability to generate novel and plausible poses (i.e., the poses from the hold-out set of the true poses).

Interpolation Experiments We compare our method and the first five baselines by evaluating the quality of interpolations produced between all pairs of landmark shapes (we omit Deep-MCMC since it is not suitable for interpolation). We show our results in Figure 7 and comparisons in Figure 6 with corresponding stats in Table 3. Each cell reports smoothness, ARAP score, and interpolation quality, and to make results more readable we normalize the scores using the corresponding value for Vanilla VAE.

Ours performs the best with respect to ARAP score and also yields consistently smoother shapes with fewer artifacts, both in terms of individual surfaces, as well as discontinuities in interpolation sequences. *ARAP Shape Interpolation* is performing consistently worse than all baselines except Vanilla-VAE. Interpolation-based baselines sometimes yield smoother interpolations, something we would expect to be true for simpler, linear motions. However, as seen in Figure 6, an outlier global rotation can disturb the otherwise smooth interpolation, which does not happen with GLASS. Additionally, we demonstrate that they are limited in their ability to synthesize novel plausible poses.

Dynamic Faust Interpolation Dynamic Faust [6] (DFaust) provides meshed data of captured motion sequences, from which we selected 5 sequences with the most variance in vertex positions. Each sequence contains 100-200 shapes from which we select ≈ 5 keypoints and train GLASS and the baselines on these. We then evaluate interpolation by measuring the $L2$ distance between the generated interpolations and the ground-truth frames in the sequence that were excluded from training. Additionally, we measure the reconstruction error of the excluded frames. The results are compared in Table 2. Our method significantly outperforms the baselines.

Using GLASS for Learning Correspondences We now evaluate our data augmentation technique on the practical task of learning 3D correspondences between shapes. We pick a state-of-the-art correspondence learning method, 3D-CODED [18], as a reference. Originally, this method was trained on 230k deformations, most of them sampled from the SMPL model [30] and 30k synthetic augmentations. We evaluate how well this method could perform with a smaller training set, with and without the proposed augmentation.

We train 3D-CODED using 280 sampled shapes from SMPL [30] that are augmented with a number of additional deformations generated from our model (see Table 5). Ours consistently provides a significant improvement over training 3D-CODED with the original landmarks. For reference, the correspondence error of 3D-CODED trained on the full 230k pose dataset is 1.98cm.

Ablation study In this section we evaluate the contribution of various steps in GLASS to our interpolation metrics. We evaluate on the Faust-10, Centaurs-6 and Horses-8

Table 3. Surface smoothness, ARAP energy, and standard deviation of inter-frame spacing between landmarks by interpolation across different datasets. All results are normalized such that Vanilla VAE is 1.0, and lower numbers are better.

Data	Vanilla VAE	+Interpolation	+Interp. +Energy	LIMP	+ARAP SI	GLASS
Faust-3	1.0 / 1.0 / 1.0	0.47 / 0.22 / 0.43	0.44 / 0.24 / 0.29	0.45 / 0.32 / 0.7	0.95 / 1.49 / 0.91	**0.42/0.20 / 0.21**
Faust-5	1.0 / 1.0 / 1.0	0.53 / 0.25 / 0.49	0.53 / 0.25 / 0.41	0.52 / 0.27 / 0.5	0.99 / 1.13 / 0.87	**0.51 / 0.23 / 0.38**
Faust-7	1.0 / 1.0 / 1.0	0.64 / 0.31 / 0.49	**0.57** / **0.26** / 0.62	0.62 / 0.32 / 0.72	0.84 / 0.57 / 0.86	**0.57** / 0.3 / **0.23**
Faust-10	1.0 / 1.0 / 1.0	0.71 / 0.37 / 0.3	0.67 / 0.43 / 0.27	0.66 / 0.36 / 1.52	0.78 / 0.48 / 0.55	**0.55 / 0.27 / 0.54**
Centaurs-3	1.0 / 1.0 / 1.0	0.53 / 0.31 / 0.22	0.51 / 0.31 / 0.48	0.52 / 0.34 / 0.24	0.89 / 1.28 / 2.12	**0.5 / 0.28 / 0.18**
Centaurs-4	1.0 / 1.0 / 1.0	0.53 / 0.24 / 0.27	0.52 / 0.26 / 0.17	0.51 / **0.25** / 0.48	0.88 / 0.88 / 0.8	**0.49 / 0.25 / 0.1**
Centaurs-6	1.0 / 1.0 / 1.0	0.59 / 0.26 / **0.33**	0.65 / 0.38 / 0.5	0.6 / 0.27 / 0.47	0.98 / 1.02 / 0.44	**0.58 / 0.22** / 0.43
Horses-3	1.0 / 1.0 / 1.0	0.44 / 0.39 / 0.33	0.46 / 0.33 / **0.28**	0.44 / 0.4 / 0.43	0.9 / 1.41 / 1.28	**0.42 / 0.24** / 0.43
Horses-4	1.0 / 1.0 / 1.0	0.48 / 0.29 / 0.28	0.47 / 0.3 / 0.45	**0.45** / 0.36 / 0.85	0.94 / 1.25 / 1.42	0.48 / **0.22 / 0.44**
Horses-8	1.0 / 1.0 / 1.0	0.55 / 0.31 / 0.55	0.56 / 0.38 / 0.7	**0.53** / 0.43 / 1.29	0.83 / 0.76 / 0.74	**0.53 / 0.25 / 0.28**

Table 4. Ablation study results.

Data	1: Vanilla VAE	2: (1)+L_{deform}	3: (1) + perturb	4: (2) + perturb	5: (3)+project	6: (4)+project
Faust-10	1.0 / 1.0 / 1.0	0.63 / 0.39 / 0.6	0.61 / 0.35 / 0.6	0.59 / 0.32 / 0.59	0.56 / 0.32 / 0.57	**0.55 / 0.27 / 0.54**
Centaurs-6	1.0 / 1.0 / 1.0	0.69 / 0.34 / 0.54	0.62 / 0.31 / 0.47	0.59 / 0.29 / 0.47	0.59 / 0.27 / 0.46	**0.58 / 0.22 / 0.43**
Horses-8	1.0 / 1.0 / 1.0	0.66 / 0.37 / 0.43	0.59 / 0.37 / 0.33	0.56 / 0.29 / 0.32	0.54 / 0.27 / 0.29	**0.53 / 0.25 / 0.28**

Table 5. Correspondence error on the Faust INTRA benchmark, by GLASS-augmenting 3D-CODED with deformations sampled from our method.

Data	+GLASS augmentation	Error (cm)
Faust-3	0	26.90
Faust-3	3,065	**13.18**
Faust-7	0	22.10
Faust-7	3,573	**11.78**
SMPL-280	0	14.59
SMPL-280	40,000	**6.85**

datasets. Since these samples are all that is available we do not have a hold-out set, so we do not measure coverage. Starting with Vanilla-VAE (that only uses $L_{\text{Reconstruction}}$ and L_{Gaussian}), we first add the deformation energy loss ($L_{\text{Deformation}}$), which is ARAPReg [20]. Table 4(1,2) shows this improves all metrics.

Next, we consider our perturbation strategy (Section 3.3 i, ii) and add it to both vanilla and energy-guided VAE (Table 4.3, 4.4). We observe that $L_{\text{Deformation}}$ performs better because it makes the latent-space conducive for sampling low energy shapes. Having $L_{\text{Deformation}}$ helps our perturbation strategy find low energy shapes that are suitable for our projection step. Due to this, we discover shapes with energy as low as 0.001, while without it, the discovered shapes can have energy > 0.1. This difference helps the subsequent projection step converge faster to our required threshold of 10^{-5}.

Finally, we look at the projection step (Section 3.3 iii). We add it to both baseline techniques that have perturbation, and report results in Table 4.5, 4.6, where column (6) corresponds to our final method. Adding the projection step improves the smoothness and ARAP scores. After projection, our shapes have very low ARAP, in the order of 10^{-5}. Since these are added back to the training set, we observe that perturbation steps in future iterations find lower energy shapes. This further improves convergence of the projection step in future iterations. Overall $L_{\text{Deformation}}$ helps both the perturbation and projection steps converge faster to low energy shapes, and since projected shapes are encoded again by training, both perturbation and projection steps require fewer iterations.

5. Conclusion

GLASS is shown to be an effective generative technique for 3D shape deformations, relying solely on a handful of examples and a given deformation energy. The main limitation of our method is its reliance on a given mesh with vertex correspondences, preventing its use on examples with different triangulations, and we set the goal of generalizing it to arbitrary geometries as important future work.

We believe our proposed technique opens many future directions. There are many other deformation energies that could be explored; e.g., densely sampling conformal (or quasi-conformal) deformations from a given sparse set can be an extremely interesting followup. More broadly, replacing the deformation energy with learned energies, such as the output of an image-discriminator, may enable generating plausible images, given a very sparse set of examples.

Acknowledgement This project has received funding from the UCL AI Center, gifts from Adobe Research, and EU Marie Skłodowska-Curie grant agreement No 956585.

References

[1] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible shape interpolation. In *SIGGRAPH*, 2000.

[2] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: Reconstruction and parameterization from range scans. In *SIGGRAPH*, 2003.

[3] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. SCAPE: Shape completion and animation of people. In *SIGGRAPH*, 2005.

[4] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH*, 1999.

[5] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. FAUST: Dataset and evaluation for 3D mesh registration. In *CVPR*, 2014.

[6] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *CVPR*, 2017.

[7] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *TVCG*, 14(1), 2008.

[8] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008.

[9] Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, 1998.

[10] Amit Chaudhary. A visual survey of data augmentation in NLP, 2020. https://amitness.com/2020/05/data-augmentation-for-nlp.

[11] Siddhartha Chaudhuri, Daniel Ritchie, Jiajun Wu, Kai Xu, and Hao Zhang. Learning generative models of 3D structures. *Comput. Graph. For. (Eurographics STAR)*, 2020.

[12] Luca Cosmo, Antonio Norelli, Oshri Halimi, Ron Kimmel, and Emanuele Rodolà. LIMP: Learning latent shape representations with metric preservation priors. *ECCV*, 2020.

[13] Matheus Gadelha, Giorgio Gori, Duygu Ceylan, Radomir Měch, Nathan Carr, Tamy Boubekeur, Rui Wang, and Subhransu Maji. Learning generative models of shape handles. In *CVPR*, 2020.

[14] Matheus Gadelha, Rui Wang, and Subhransu Maji. Multiresolution tree networks for 3D point cloud processing. *CoRR*, abs/1807.03520, 2018.

[15] Lin Gao, Yu-Kun Lai, Jie Yang, Ling-Xiao Zhang, Shihong Xia, and Leif Kobbelt. Sparse data driven mesh deformation. *TVCG*, 27(3), 2021.

[16] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. SDM-NET: Deep generative network for structured deformable mesh. *ACM Trans. Graph.*, 38(6), 2019.

[17] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[18] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. 3D-CODED: 3D correspondences by deep deformation. *ECCV*, 2018.

[19] Frédéric Hélein and John C. Wood. *Handbook of Global Analysis*, chapter Harmonic Maps. 2008.

[20] Qixing Huang, Xiangru Huang, Bo Sun, Zaiwei Zhang, Junfeng Jiang, and Chandrajit Bajaj. ARAPReg: An as-rigid-as possible regularization loss for learning deformable shape generators. *CoRR*, abs/2108.09432, 2021.

[21] Qi-Xing Huang, Martin Wicke, Bart Adams, and Leonidas Guibas. Shape decomposition using modal analysis. *Computer Graphics Forum*, 28(2), 2009.

[22] Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 30(4), 2011.

[23] Alec Jacobson, Zhigang Deng, Ladislav Kavan, and J. P. Lewis. Skinning: Real-time shape deformation. In *SIGGRAPH Courses*, 2014.

[24] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. In *SIGGRAPH*, 2005.

[25] Martin Kilian, Niloy J. Mitra, and Helmut Pottmann. Geometric modeling in shape space. *ACM Trans. Graph.*, 26(3), 2007.

[26] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[27] Hamid Laga. A survey on non-rigid 3D shape analysis. *CoRR*, abs/1812.10111, 2018.

[28] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérôme Maillot. Least squares conformal maps for automatic texture atlas generation. In *SIGGRAPH*, 2002.

[29] J. P. Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Fred Pighin, and Zhigang Deng. Practice and theory of blendshape facial models. *Eurographics State of the Art Reports*, 2014.

[30] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6), 2015.

[31] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3D faces using convolutional mesh autoencoders. In *ECCV*, 2018.

[32] Guodong Rong, Yan Cao, and Xiaohu Guo. Spectral mesh deformation. *The Visual Computer*, 24, 2008.

[33] Babak Shahbaba, Luis Martinez Lomeli, Tian Chen, and Shiwei Lan. Deep Markov Chain Monte Carlo. *CoRR*, abs/1910.05692, 2019.

[34] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. In *ICLR*, 2018.

[35] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 2019.

[36] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *SGP*, 2007.

[37] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational autoencoders for deforming 3D mesh models. In *CVPR*, 2018.

[38] Qingyang Tan, Zherong Pan, Lin Gao, and Dinesh Manocha. Realtime simulation of thin-shell deformable materials using CNN-based mesh embedding. *IEEE Robotics and Automation Letters*, 5(2), 2020.

[39] Christoph Von-Tycowicz, Christian Schulz, Hans-Peter Seidel, and Klaus Hildebrandt. Real-time nonlinear shape interpolation. *ACM Trans. Graph.*, 34(3), 2015.

[40] Yifan Wang, Noam Aigerman, Vladimir G. Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3D deformations. In *CVPR*, 2020.

[41] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: A survey. *CoRR*, abs/2002.12478, 2021.

[42] Yu-Jie Yuan, Yu-Kun Lai, Tong Wu, Lin Gao, and Ligang Liu. A revisit of shape editing techniques: from the geometric to the neural viewpoint. *CoRR*, abs/2103.01694, 2021.

[43] Mehmet Ersin Yumer and Levent Burak Kara. Co-constrained handles for deformation in shape collections. *ACM Trans. Graph.*, 33(6), 2014.

Appendix for GLASS: Geometric Latent Augmentation for Shape Spaces

A. Evaluation metrics.

We use the following to evaluate performance:
(i) Coverage: While it is difficult to evaluate whether generated poses are meaningful and diverse, we propose using the holdout data (S_H) that was not part of the input exemplars to see if the newly generated poses L_G cover every holdout example:
$M_{\text{coverage}} := \sum_{s \in S_H} \min_{g \in L_G} D(s,g)/|S_H|$, where $D(s,g)$ is the average Euclidean distance between corresponding vertices of shapes s and g.
(ii) Mesh smoothness: This metric measures how well a method preserves the original intrinsic structure of the mesh. We compute the mean curvature obtained from the discrete Laplace-Beltrami operator:
$M_{\text{smoothness}} := \sum_{i=1}^{N} \|\Delta(V_i)\|/2$ where Δ is the area-normalized cotangent Laplace-Beltrami operator and N is the number of mesh vertices.
(iii) Interpolation smoothness: In addition to measuring quality of individual meshes, we also evaluate the quality of interpolations between pairs of shapes. Since none of the existing methods guarantee a clear relationship between the distances in the latent space and differences between their 3D counterparts, we first densely sample 1000 poses between pairs of landmark deformations and then keep a subset of 30 poses so that they have approximately equal average Euclidean distance between subsequent frames. We then measure the standard deviation of these Euclidean distances, as a way to penalize interpolations that yield significant jumps between frames. While this measure is imperfect (e.g., variance could decrease as we increase the sampling rate), we found it to stabilize in practice after 1000 poses (we sampled up to 3000), which suggests that denser sampling would not reveal new poses in the latent space.

B. Variable step-size α

Let f be the energy function and l be the latent code corresponding to a zero/low energy shape in the training set, and l_t be the latent code obtained from the update in Equation 6 in the main paper under a small (update) step (i.e., $\|l_t - l\|$ can be considered to be infinitesimal). Assuming that the deformation energy is C2 continuous (and single valued), $f(l_t)$ can be approximated using Taylor series expansion of the up to the 2nd order as,

$$f(l_t) \approx f(l) + (l_t - l)^T \nabla_l f(l_t) + \frac{1}{2}(l_t - l)^T H (l_t - l), \quad (8)$$

where H is the Hessian from Equation 5. Note that H denotes the Hessian term with respect to the current pose l. Since $(l_t - l)$ and $\nabla_l f(l_t)$ are orthogonal, $(l_t - l)^T \nabla_l f(l_t) = 0$. The update step can be expressed in terms of the local eigenvectors as (see Equation 6 in the main paper),

$$(l_t - l) = \alpha \sum_{i=1}^{i=k} \hat{\beta}_i U_i^{\uparrow}(H).$$

Further, since eigenvalues and eigenvectors are related as $HU_i^{\uparrow}(H) = \lambda_i U_i^{\uparrow}(H)$ and the vectors $U_i^{\uparrow}(H)$ have unit length, we obtain,

$$f(l_t) \approx f(l) + \frac{1}{2}\alpha^2 \sum_{i=1}^{i=k} \hat{\beta}_i^2 \lambda_i.$$

By setting an upper bound on the change in deformation energy $f(l_t) - f(l) \leq \delta$, we obtain the relation for α as,

$$\alpha \leq \sqrt{\frac{2\delta}{\sum_{i=1}^{k} \hat{\beta}_i^2 \lambda_i}}.$$

C. Additional Generated Samples

Figures 9, 10, and 11 show some example generated deformations starting from sparse sets of Faust, Centaur, and Horse models, respectively.

Figure 9. We show some of the samples generated by GLASS (gold), from only 10 Faust poses (gray). Several poses of the limbs are unseen in the training set - crossed arms, long leg strides, half-lowered arms.

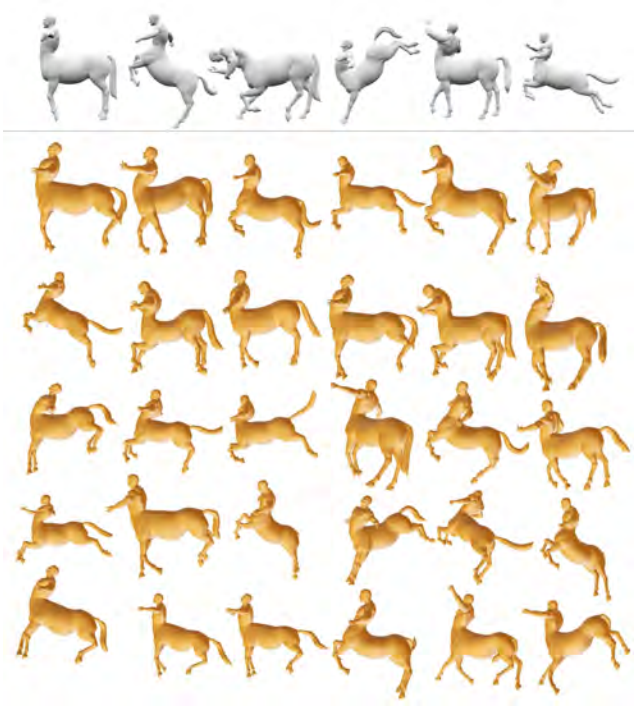

Figure 10. We show some of the samples generated by GLASS (gold), from only 6 Centaur poses (gray). We see novel poses like bent back legs and torso facing upwards.

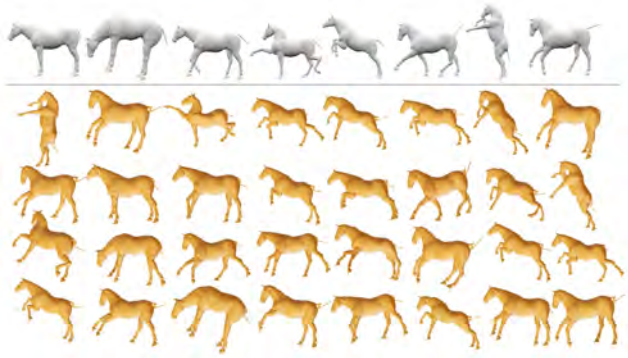

Figure 11. We show some of the samples generated by GLASS (gold), from only 8 Horse poses (gray). We see novel poses like front legs raised beyond what's seen in the training set, upright torso and back legs stretching farther.

D. Additional Interpolation Examples

Figures 12, 13 and 14 show example interpolated shapes between end poses (shown in gray), using the discovered latent space revealed by GLASS.

E. Dataset Images

We include images of the various input datasets we used to highlight the diversity of pose variations in the input.

Figure 12. Interpolated shapes (gold) between 2 Centaurs Poses (gray) inside the latent space generated using GLASS.

Figure 13. Interpolated shapes (gold) between 2 Horse Poses (gray) inside the latent space generated using GLASS.

Figure 14. Interpolated shapes (gold) between 2 Faust Poses (gray) inside the latent space generated using GLASS.

Note that the datasets are all very sparse, consisting of 3-10 models.

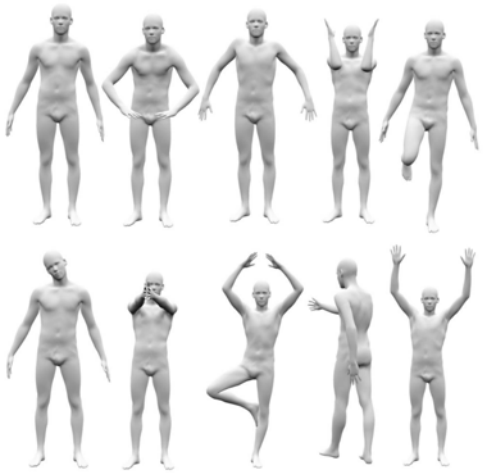

Figure 15. The Faust-10 dataset with 10 poses. Please refer to Figure 16 for the corresponding even sparser versions of the dataset used in our experiments.

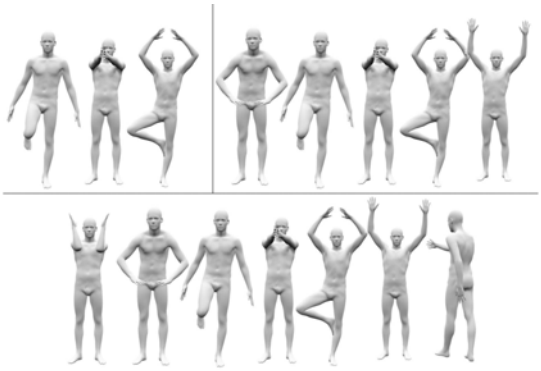

Figure 16. Faust-3 (top left), Faust-5 (top right) and Faust-7 (bottom).

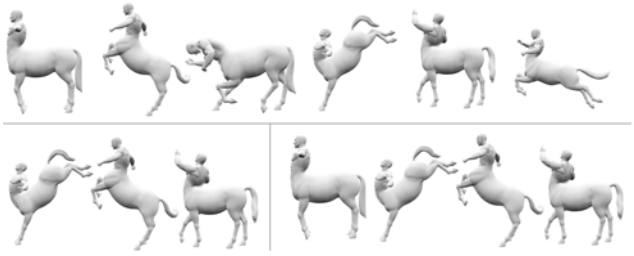

Figure 17. Centaurs-6 (top), Centaurs-3 (bottom left), Centaurs-4 (bottom right).

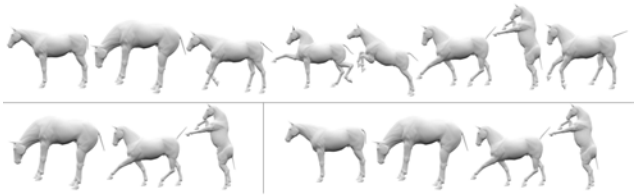

Figure 18. Horses-8 (top), Horses-3 (bottom left), Horses-4 (bottom right).

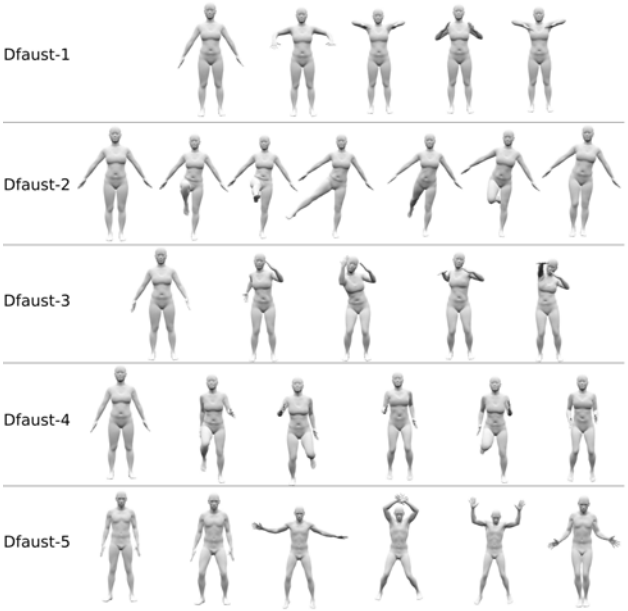

Figure 19. The keypoints of 5 different Dynamic Faust sequences used for training, for Table 2 of main paper.

F. Network Architecture

Figure 20 shows the VAE architecture used by GLASS. The main pseudocode for GLASS is provided in the main paper.

G. Extrapolation

We extrapolate beyond the endpoints of interpolation for the same length as interpolation and measure the smoothness of the mesh, and report the results in table 6. We observe that extrapolated meshes have a smoother surface for GLASS than the baselines.

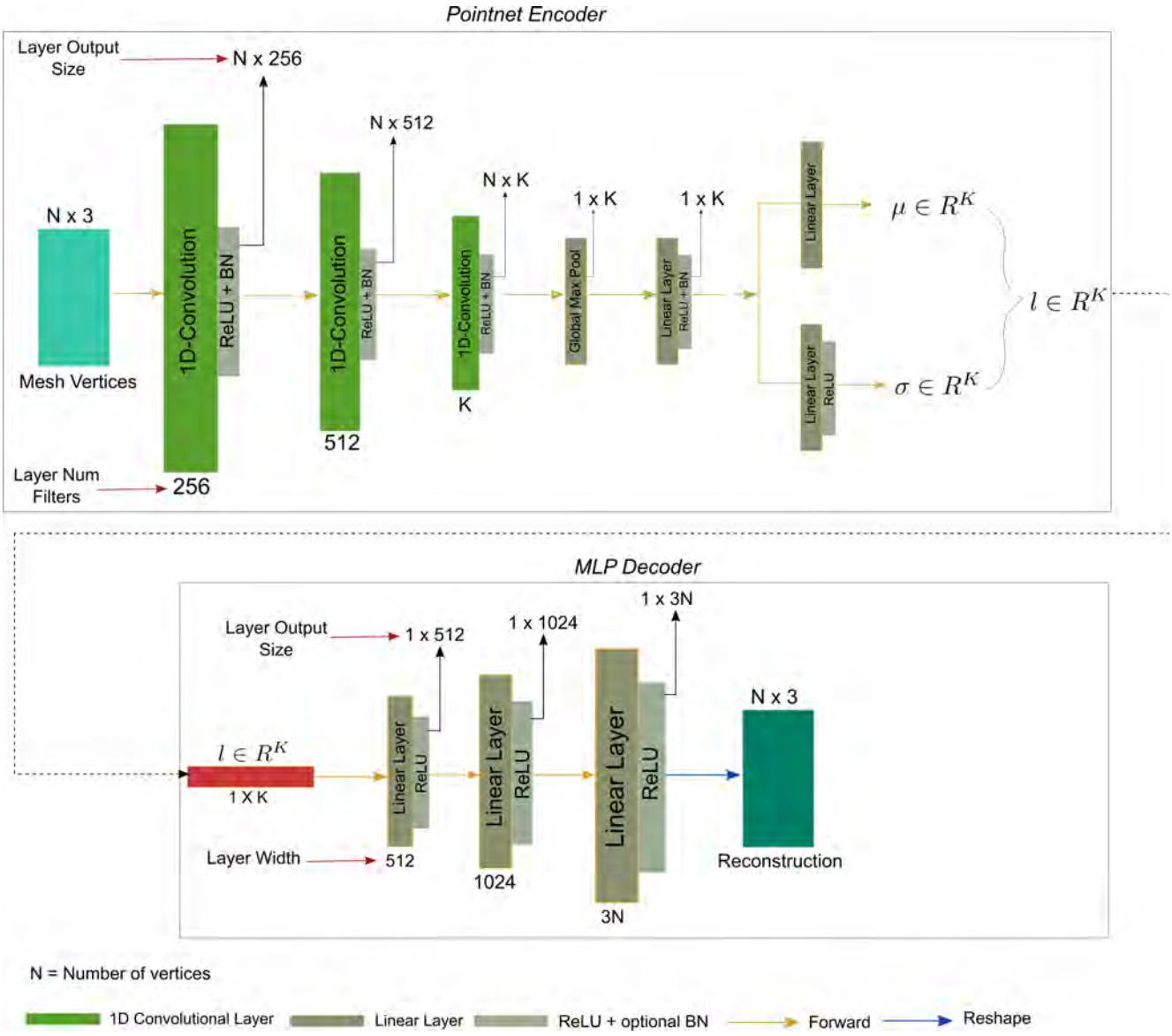

Figure 20. The VAE architecture used by GLASS.

Table 6. Ablation study. Surface smoothness of extrapolated shapes. All results are normalized such that Vanilla VAE is 1.0, and lower numbers are better.

Data	Vanilla VAE	+Interpolation	+Interp. +Energy	LIMP	+ARAP SI [1]	GLASS
Faust-3	1.0	0.74	0.79	0.87	1.18	**0.69**
Faust-5	1.0	0.64	0.6	0.63	1.07	**0.59**
Faust-7	1.0	0.66	0.66	0.64	0.94	**0.63**
Faust-10	1.0	0.65	**0.6**	0.62	0.93	**0.6**
Centaurs-3	1.0	1.13	1.03	1.11	1.21	**0.79**
Centaurs-4	1.0	0.88	0.88	0.9	1.09	**0.76**
Centaurs-6	1.0	0.7	**0.69**	0.79	1.07	**0.69**
Horses-3	1.0	1.1	1.06	1.1	1.22	**0.75**
Horses-4	1.0	0.87	0.82	0.8	1.09	**0.71**
Horses-8	1.0	0.62	0.62	0.61	1.0	**0.59**