Towards 3D Generative Models With Sparse Guidance

Sanjeev Muralikrishnan

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

June 11, 2025

I, Sanjeev Muralikrishnan, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Storytelling in its various forms has been a part of the human experience for several millenia. From cave impressions to contemporary virtual experiences, the tools for telling stories have evolved a long way. Today, digital 3D assets are the alphabets of modern storytelling, invaluable in several creative pipelines from games to movies to advertisements. They provide creative storytellers with a powerful medium to immerse their audiences in vivid worlds.

Creating rich 3D content typically calls for teams of highly skilled artists working with sophisticated and complex authoring tools. As such, creating high-quality 3D assets is quite expensive in time and cost. This barrier greatly limits the availability of 3D data compared to domains like images and text. Thus there has always been a need for automated or semi-automated creation of 3D content. In recent years, with the advent of generative machine learning, artists benefit from AI models that are trained to generate or edit 3D assets. These AI models aim to augment the artists' creativity and continually ease the process of 3D content creation. However, for contemporary generative models to be truly useful, they need to be trained on large datasets of 3D content leading to a cyclic chicken-and-egg problem wherein to aid artists create 3D content we require large volumes of difficult-to-author 3D content in the first place. In this thesis, we present algorithms to learn generative models for 3D that can be trained with sparse datasets that are more easily available. We introduce three such novel frameworks, and they tackle the problem of 3D generation for shapes, poses and animation in these sparse data settings.

Acknowledgements

This thesis is the culmination of a long, arduous yet satisfying journey that began in 2015 with my first foray into research. Its a journey I'm proud to have taken for it was one of self-discovery, of gratitude and immense personal growth beyond what I thought possible. I take this opportunity to express my gratitude to the people who knowingly or otherwise made this journey possible.

I thank my parents and my sister Vinitra for always believing in my dreams, for being there in persistent support as I chased them. Thank you for everything you did to make this dream possible.

I thank Siddhartha Chaudhuri for initiating this journey in 2015, for fueling it with passion and inculcating in me the belief that I can thrive on this path.

I thank my supervisor Niloy J. Mitra for helping me discover the scientist I want to be, for his incessant drive in pushing me to always aim for excellence in science and for being in the trenches with me through tough deadlines.

Special thanks to Vladimir Kim, Noam Aigerman, Matthew Fisher, Duygu Ceylan and Chun-Hao Paul Huang for believing in the research I proposed to undertake and for helping me stress-test every aspect of the work we produced.

A PhD can be a lonely journey, it wasn't for me. I'm immensely grateful to my colleagues at 169 Euston Road and the many friends in London and back home for their support and friendship through this ride. I thank Michael and Luca for the numerous times I needed their advice, the laughter and the lunchtime banters, you have been invaluable. I thank Yiftach and Shalini for the exciting board-game nights, for the laughs and their sagely advice. I particularly thank Yiftach for his wonderful friendship that set me at ease during this long walk.

Thank you Michael, Niladri, Karran, Romy for your companionship, the amazing climbing sessions and for answering some confounding questions I struggled to answer during these years. I also thank Shravan, Mohit and Nishchay for setting me on this journey through the Swayam Satellite project, for initiating me in the ways of science and for their constant support and guidance.

My deepest gratitude goes to my school friends Amey, Yogesh, Aditya, Suraj, Nimesh, Ishan, Vivek and Ashwin for being my everyday-family, for being my virtual home away from home. Thank you for your friendship, laughter and the countless cricket and random discussions that kept me going every single day.

I thank Amey for believing in me, for always lifting me up when I was down and prodding me along this road. I have been very lucky to have you as my friend.

I'm deeply grateful to my cats Ginny, Joey and Trubbsy for adopting me into their warm furry world. My deepest gratitude to the goodest boy Bruno who through serendipity brought me together with my wife.

Lastly, I am filled with immense gratitude for my wife Harshita. I feel incredibly lucky to have met you during this journey. In my quest to discover the scientist I want to be, you helped me become the person I want to be. Thank you for being the rock during these times, for being proud of me and believing in me.

I'm forever indebted to everyone who made this thesis possible.

Publications

The work presented in this thesis has been published as conference papers, references are listed below.

- Chapter 2:

  Sanjeev Muralikrishnan, Siddhartha Chaudhuri, Noam Aigerman, Vladimir G. Kim, Matthew Fisher, Niloy J. Mitra. GLASS: Geometric Latent Augmentation for Shape Spaces In *Proc. Computer Vision and Pattern Recognition, IEEE*, 2022.

- Chapter 3:

  Sanjeev Muralikrishnan, Chun-Hao Paul Huang, Duygu Ceylan, Niloy J. Mitra BLiSS: Bootstrapped Linear Shape Spaces In *Proc. International Conference on 3D Vision, IEEE*, 2024.

- Chapter 4:

  Sanjeev Muralikrishnan, Niladri Shekhar Dutt, Siddhartha Chaudhuri, Noam Aigerman, Vladimir G. Kim, Matthew Fisher, Niloy J. Mitra. Temporal Residual Jacobians for Rig-free Motion Transfer In *Proc. European Conference on Computer Vision , Springer Science*, 2024.

Code has been released for all these works.

Impact Statement

This thesis presents three works that introduce frameworks for generating static and dynamic 3D content from very sparse datasets. They were presented at CVPR, 3DV, and ECCV. These works primarily learn to generate human/animal shapes in varied poses and/or shapes. In this context, pose refers to the orientation of parts relative to others according to a hierarchical kinematic tree defined on body joints, while shape refers to geometric surface details – like curvature and normal orientation – that define the identity of the human or animal.

Chapter 2 (GLASS) introduces a method to iteratively grow pose spaces - the space of relative part orientations - seeded with 3–10 poses to rich spaces of up to 2500 pose variations (identity-preserving deformations), by injecting classical geometric priors into the neural learning process. Chapter 3 (BLiSS) extends this idea by injecting learned neural priors to model shape variation - identity-changing deformations. Starting with 200 artist-curated body shapes, BLiSS grows the shape space – the space of all identities – to 1000 through iterative growth. Chapter 4 (Temporal Residual Jacobians) boosts learning of spatio-temporal motion fields from very few motion examples by learning temporal signals that preserve motion consistency; the learned field, when applied to characters-in-the-wild, imparts complex motion to these stylistic characters, creating newly animated shapes from sparse motion datasets.

Each chapter is self-contained, with its associated literature review included. This thesis demonstrates that learning generative 3D models greatly benefits from task-specific priors, enabling such models to be trained from super-sparse guidance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As our tools for communicating with each other evolve, so does our reliance on newer, richer and more complex forms of data. One such relatively modern modality is digital 3D data. 3D data have become invaluable in gaming, movie-making (CGI, VFX), animated stories, engineering (CAD, simulation), news media, sports broadcasting and more recently in creating immersive experiences in Virtual/Augmented Reality settings. This reliance on 3D content is only expected to increase, calling for easier and faster ways of creating them.

Compared to other communication media like images or text (which either already have a large volume or are easier to capture), creating and/or capturing 3D content involve cumbersome pipelines with experts involved at various stages of 3D acquisition. Thus, while our need for 3D data increases we face the bottlenecks inherent with its creation or acquisition. The most common pipeline is that of artists authoring 3D data in authoring environments like Blender, Maya, 3DS Max and others. Each of these provide a complex set of tools that enable creating 3D assets and often require the artist to be fluent at employing these tools. An alternative pipeline is that of acquiring 3D content via RGBD scanners which output 3D shapes as a collection of points in space. However most graphics pipelines cannot use these scans in their raw form and require tedious post-processing and cleaning by highly-skilled artists.

Thus, research efforts have focused on easing the process of 3D content creation. Before the rise of deep learning, substantial research focused on intuitive,

artist-friendly methods for 3D content creation. These included sketch-based modeling systems [10, 11], example-based modeling using shape galleries [12, 13], and part-assembly-driven shape construction [14, 15]—all aimed at simplifying the modeling process through familiar and interactive paradigms. Complementary to these approaches, techniques that enable 3D generation and editing through semantic attributes allow artists to specify or adjust features using language, making it possible to emphasize or suppress certain characteristics in the resulting shapes [16, 17]. Together, these works highlight the longstanding goal of making 3D content creation more accessible, efficient, and intuitive.

More recent attempts offer to convert from easier-to-acquire modalities - text, image, videos - to 3D assets. These advances aim to create far less cumbersome workflows for creating 3D data, both for expert artists and relative novices. Both unconditional [18] and conditional 3D generation [19, 20] have seen a surge in research focus, aided by the progress in Transformer [21] based generative architectures. Strong 2D Diffusion models [22] trained on millions of images have further boosted text-driven 3D content generation [23, 24, 25, 26] allowing artists and novices to instantly create 3D assets using single or multi-view images and simple text prompts. However, being a relatively new field of research the assets created from these works are far from usable in standard graphics pipelines for two reasons - they often generate 3D content in non-mesh representations, while most real-time graphics pipelines rely on high quality mesh assets, or when they do generate meshes, do so with poor triangulation quality with elongated or very small triangles [18] making them unsuitable for animation and deformation tasks. More importantly, training these generative models require large volumes of 3D data in the first place, leading to a cyclic problem.

In this thesis, we focus on learning generative models for 3D content that can represent a wide range of poses and shapes, using very sparse and easily obtainable datasets. Within this context, we define “shape” as the selection of parts, along with their local geometric details, that together establish a unique identity—for example, the specific structural characteristics that distinguish chair ’A’ from any other chair.

In contrast, "pose" refers to the relative arrangement or orientation of these parts within a given shape. Varying the pose alters the configuration without changing the underlying identity—for instance, tilting the backrest of a chair relative to its seat transforms it into a reclining chair, while retaining its identity as the same chair. Thus, a single shape can appear in multiple poses, and each object category (e.g., chairs, airplanes, humans) can include a wide variety of distinct shapes.

For any given object category, the generative space can be viewed as spanning two primary axes: shape (also referred to as body shape), which defines the identity of an object within the category, and pose, which specifies the configuration or articulation of that shape. An ideal 3D dataset would comprehensively cover all object categories, with each shape observed across a wide range of poses. However, constructing such datasets is infeasible in practice, as the number of possible categories and the combinatorial variety of shape–pose configurations are effectively unbounded.

In reality, popular datasets such as [27, 28] typically take the form of a bag-of-shapes: diverse collections containing a broad assortment of objects, but only a few instances per category—and even fewer examples of the same shape in different poses. In contrast, articulated shape spaces such as SMPL [4], SMPL-X [29], and GHUM [5] are explicitly designed to provide dense, continuous coverage of the shape–pose space for a specific category—usually humans, hands, or faces. These models parameterize along both axes, enabling disentangled sampling of pose and shape. However, building such spaces requires large volumes of high-quality, finely annotated data for a single category, capturing both local geometric detail and articulation. As a result, these models are not easily generalizable to other object categories and are typically constrained to humanoid and animal forms.

This thesis similarly focuses on humans and animals, using mesh-based representations—that is, collections of 3D points connected via triangulated faces that define the object's surface geometry.

Given that most readily available shape datasets contain only a few shapes, each observed in even fewer poses, the goal of this thesis is to complete the

shape–pose matrix by discovering novel instances along each axis. Specifically, we address shape and pose generation independently across different works.

In GLASS (Chapter 2), we introduce a framework for discovering novel poses of a fixed shape, starting from as few as 5–10 exemplar poses. In BLiSS (Chapter 3), we present a method for generating novel shapes in a fixed pose, given a sparse set of shape exemplars. Finally, in Temporal Residual Jacobians (TRJ) (Chapter 4), we extend shape generalization—similar to BLiSS—to dynamic motion sequences, focusing on deformation over time for varying identities.

The central challenge in completing the sparse shape–pose matrix lies in its inherently circular nature: the very sparsity of the data makes it difficult to train models capable of overcoming that sparsity. Neural generative models typically require large datasets to learn meaningful and generalizable representations. However, when trained on limited data, such models tend to overfit, exhibiting poor capacity for generating novel content. In contrast, the works in this thesis seek to train generative models from sparse examples while still expecting them to extrapolate significantly beyond the observed data. This goal—robust generalization from limited supervision—stands in contrast to much of the existing deep learning literature, which rarely addresses such extreme data scarcity.

Our key insight is that we can learn rich shape spaces from sparse datasets by injecting plausibility priors in the training process. In our first work GLASS (Chapter 2), we build a rich pose-space of 2000+ poses starting from as few as 3-10 pose examples. The space is iteratively grown over several iterations; in each iteration we search the current version of the space, which is noisy due to the sparsity, for novel and plausible new poses. This search is guided by a classical geometric energy function which vastly constraints the search space. This energy function is the plausibility prior in this case, and the shapes discovered in each iteration are refined and projected back to obtain the new, richer version. After several iterations, we end up with a rich space of poses despite beginning with fewer than 10 initial poses.

In our next work BLISS (Chapter 3), we follow a similar iterative approach however here the prior is learned from a small dataset of artist-annotated shapes.

We first build a linear shape space using the initial sparse set. Being a sparse space it is limited in expressing the details on shape surfaces. In each iteration, the current version of the space is used to register raw noisy scans to the shape space resulting in coarse registrations. We train a surface deformation model using a small set (100) of artist-registered shapes to map from these coarse registrations to highly-detailed ones. Thus the new registrations are imparted fine detail through this learned prior and finally projected to the linear space, leading to its next version. Again, starting from 100, over several iterations we grow to a space of about 1000 shapes.

Finally, having presented generative models for static poses and shapes, we turn our attention in TRJ (Chapter 4) to learning spatio-temporal motion fields from a sparse set of motion examples. To address the challenge of limited data (5–10 examples), we design our model to learn motion-consistency priors by focusing on the temporal residuals of the motion field at each time step. These temporal residuals capture the underlying principles governing motion evolution, serving as learned priors that guide the generation process. By leveraging these priors, our model can effectively propagate the motion field across time, enabling the synthesis of complex, long-range animations. Moreover, these learned priors generalize to unseen static shapes, transforming them into dynamically animated characters.

To summarize, 3D asset generation remains a labor-intensive process for artists or requires large volumes of high-quality training data to support deep generative models. The latter often produce shapes in non-mesh formats—unsuitable for standard graphics pipelines—or generate meshes with poor triangulation, limiting their practical utility. A key source of data scarcity lies in the absence of comprehensive pose–shape matrix datasets for most object categories.

In this thesis, we introduce a set of novel tools that leverage geometric priors and localized deformations to enable 3D generative modeling in sparse-data regimes. Across the presented works, we demonstrate the effectiveness of using priors—both learned and handcrafted—to constrain the generative space. These priors guide the models toward plausible outputs and allow meaningful generalization to unseen cases, despite limited training supervision.

Chapter 2

GLASS: Geometric Latent Augmentation for Shape Spaces

In this chapter, we begin our investigation of 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, in this work, 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 extensively evaluate our method against a set of strong baselines, provide ablation studies, and demonstrate application towards establishing shape correspondences. GLASS (Geometric Latent Augmentation for Shape Spaces) produces multiple interesting and meaningful pose variations even when starting from as few as 3-10 training poses.

In GLASS we focus on novel pose generation for given shape identities while in Chapter 3 we do shape generation for a fixed pose.

2.1 Introduction

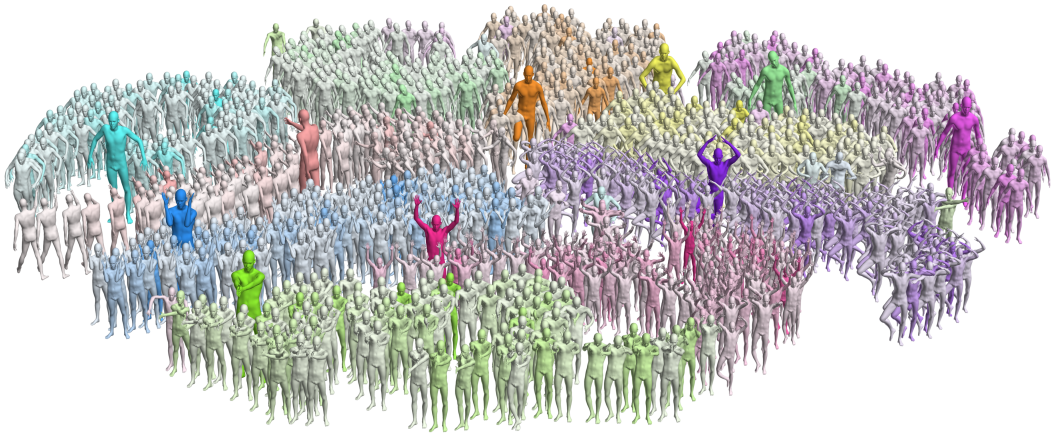

Figure 2.1: Starting from just 10 shapes (larger), our method iteratively augments the collection by alternating between training a VAE, and exploring random perturbations in its low-dimensional latent space guided by a purely geometric deformation energy. Here we show the 1000 most diverse shapes from 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.

This work is concerned with generating plausible deformations of a 3D shape from a very sparse set of examples. Fig. 2.1 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 landmarks 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 [30, 31]. An alternative approach is to jointly rig the given meshes using methods such as [32], [33], and [34], and then generate novel poses by sampling from the rig. However, as shown in [4] and [29], achieving reliable pose deformations requires identity-specific pose correctives, which are typically either handcrafted by artists or learned from data. Moreover, defining the plausibility of generated poses without relying on manually crafted metrics necessitates a data-driven prior. While rigs provide a compact, low-dimensional deformation space, discovering novel and plausible poses within this space remains a significant challenge—one that this work seeks to address. In our approach, the plausibility of a newly generated pose is guided by the current set of poses (which has grown iteratively from an initial set), while a latent encoding serves as a compact representation of the deformation space. Our work, like other recent approaches [35, 36], 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 [37]).

2.2 GLASS: Related Work

2.2.1 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 [38], blendshapes [39], skinned skeletons [40] and Laplacian eigenfunctions [41]. Semantically plausible shapes can be generated using these parameterizations by imposing constraints in the parameter space—for example, joint rotation limits for skeletal models [42]. However, such annotations must be defined separately for each new shape category, which limits scalability. Additionally, pose plausibility is often dependent on body shape (identity), requiring shape-specific pose correctives [4, 29]. In contrast, our data-driven approach generalizes across shape categories without the need for additional manual annotation and allows the data itself to define shape-specific plausibility. In contrast,

variational methods model deformations as minimizers of an energy functional – e.g. Dirichlet [43], isometric [31], conformal [44], Laplacian [45], As-Rigid-As-Possible (ARAP) [46], or As-Consistent-As-Possible (ACAP) [47] – 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 [48] (parametric) are equivalent to minimizing the Laplacian energy (variational). Please see surveys [49, 50] for a complete discussion. We are also inspired by work on modal analysis [51], which linearise the deformation space of a shape in terms of the least-significant eigenvectors of the Hessian of some energy functional. In the current work, 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. [52] 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. [31] 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.

2.2.2 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 work, please see surveys such as [53]. 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 [54], bodyshape [55] and pose [56] variations in a data-driven fash-

ion 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. [57] 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 [58, 59, 60, 61, 62], and use them for applications such as non-rigid correspondences [37]. Tan et al. [63] and Huang et al. [36] regularize a VAE with an energy-based loss. We use the latter [36] 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 training 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 [35] 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.

2.2.3 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 [64], time series [65], and NLP [66]. A particularly relevant recent technique is Deep Markov Chain Monte Carlo [67], 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 [68, 69]. In contrast, we seek to *preserve* an energy-based loss, and hence *eliminate* the gradient and other high-change directions from consideration.

2.3 Approach

2.3.1 Problem Setup

We assume all shapes in a particular input dataset are meshes with consistent topology and correspond to the same underlying identity—for example, a single individual such as person 'A'. 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", $W^1, \ldots W^n$. 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-Rigid-As-Possible (ARAP) energy [46] and its latent-space approximation ARAPReg [36] 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) [70]. Namely, during training, our method explores the current sample space, guided by the deformation energy $f(W)$,

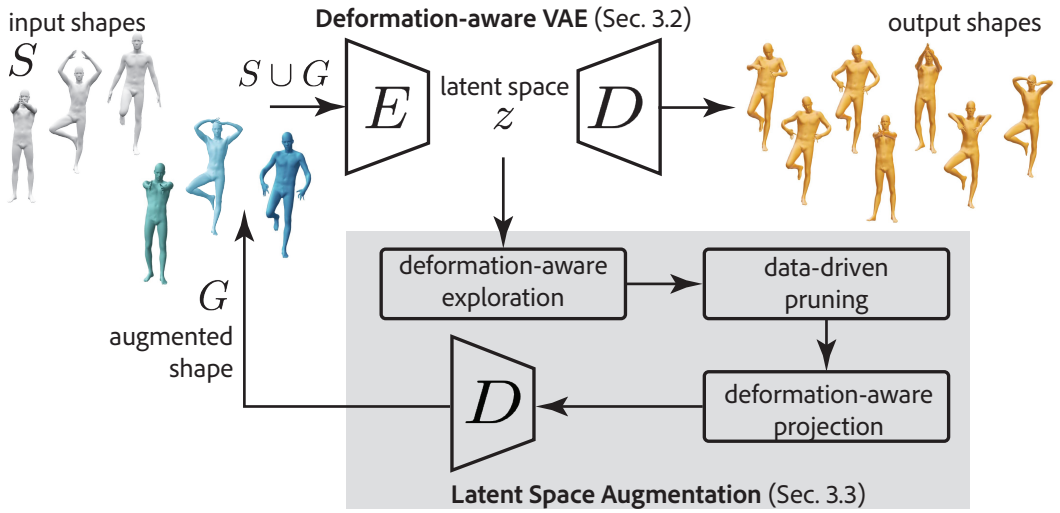

Figure 2.2: We present GLASS to iteratively build a deformation-aware VAE latent space and analyzing it to generate new training samples to augment the original training set. These enables generation of diverse yet plausible shape variations starting from very few input examples.

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.

2.3.2 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,

$$L_{\text{Reconstruction}} := \|D(E(W)) - W\|^2. \tag{2.1}$$

(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 [71]. We found that for a small sample size this batch-based loss leads to faster convergence, compared to the standard KL-Divergence. We denote this loss as,

$$L_{\text{Gaussian}} := \frac{1}{b}\sum_{i=1}^{b}\left(\|\mu_i\|^2 + \|\Sigma_i - \mathbb{I}\|^2\right), \tag{2.2}$$

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,

$$L_{\text{Deformation}} := f(D(E(W))). \tag{2.3}$$

In summary, our network training loss is

$$L := L_{\text{Reconstruction}} + L_{\text{Gaussian}} + \sigma L_{\text{Deformation}} \tag{2.4}$$

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

2.3.3 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 (2.4) 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*

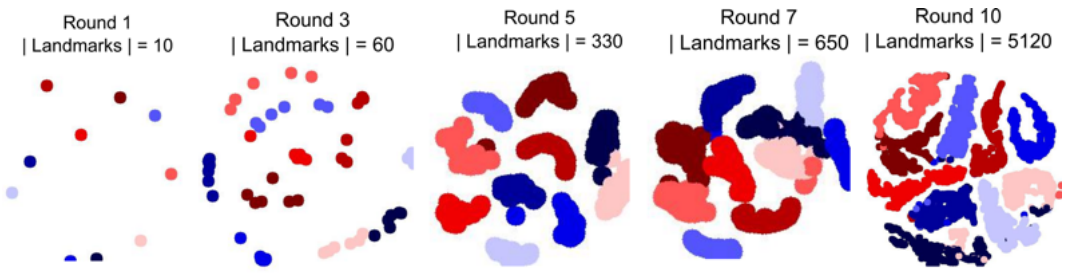

Figure 2.3: tSNE embedding of generated samples shows progressive augmentation of the shape space. Sample color indicates originating (parent) shape. See also Fig. 2.1.

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 $\in$ [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

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 2.3 illustrates how the latent space is progressively populated with new deformations over iterations, where colors indicate the base shapes.

(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,

$$H := \nabla_l \nabla_l f(D(l)). \tag{2.5}$$

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,

$$l_t := l + \alpha \sum_{i=1}^{k} \hat{\beta}_i U_i^{\uparrow}(H), \tag{2.6}$$

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 C^2 continuity for deformation energy $f()$, we can obtain the following bound on the step size.

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

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

Proof: 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 2.6 under a small (update) step (i.e., $\|l_t - l\|$ can be considered to be infinitesimal). Assuming that the deformation energy is C^2 continuous (and single valued), $f(l_t)$ can be approximated using Taylor series expansion of the up to the 2 order as,

$$f(l_t) \approx f(l) + (l_t - l)^T \nabla_l f(l_t) + \frac{1}{2}(l_t - l)^T H (l_t - l), \tag{2.7}$$

where H is the Hessian from Equation 2.5. 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,

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

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,

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

The step-size α is therefore obtained in relation to the threshold change in energy value δ; in our experiments we set δ to 10^{-4}

(ii) Data-driven pruning of the perturbed deformations. In order to add diverse samples, given the set of candidate deformations W_d, we select one example to be added to the dataset, via Maximal Marginal Relevance (MMR) ranking [72]. 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,

$$F(w) = \gamma M(w, W) - (1-\gamma) \max_{r \in R} M(w, r), \tag{2.8}$$

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^*. We perform five explorations per W, resulting in five candidates that are added to W_d. This choice provides sufficient variation for meaningful selection.

(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.

2.4 Implementation Choices

Choice of energy f. For training the VAE, we set f as the $L2$ formulation of the ARAPReg energy [36]. ARAPReg is an efficient approximation of the ARAP energy [46] 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 $\overline{H}$.

Approximation of Hessian. Similarly, for step (i) in Section 2.3.3, we use the approximate Hessian $\overline{H} \approx J^T H J$ proposed in [36] in our Equations 2.5 and 2.6, 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.

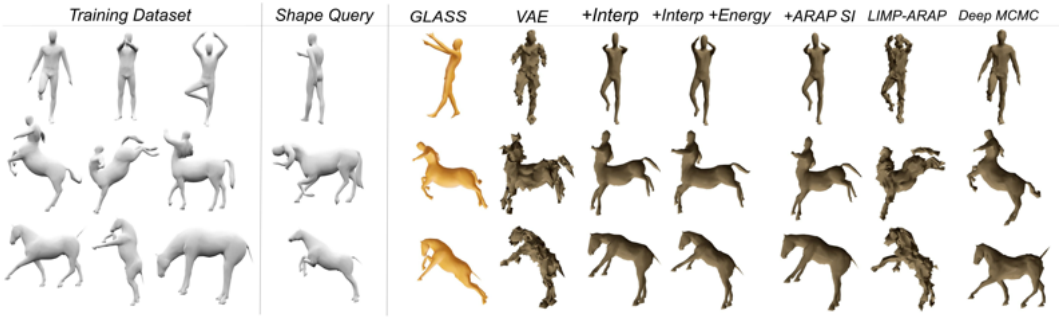

Figure 2.4: *Generation results evaluated by coverage.* We train different methods on the same training data (col 1) and generate comparable numbers of shapes. Given two shapes from the holdout data (col 2), we evaluate the methods by finding the closest generated shape (cols 3-9). Note how the baselines exhibit strong artifacts and usually do not match the query shape.

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 [46].

Our network architecture is visualized in Figure A.9 in Appendix A.

2.5 Evaluation

We evaluate GLASS on public data of humans (FAUST [73]) and creatures (TOSCA [74]) 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 (see A.5 in Appendix A for the landmark poses in each "SubjectName-X"); most results use

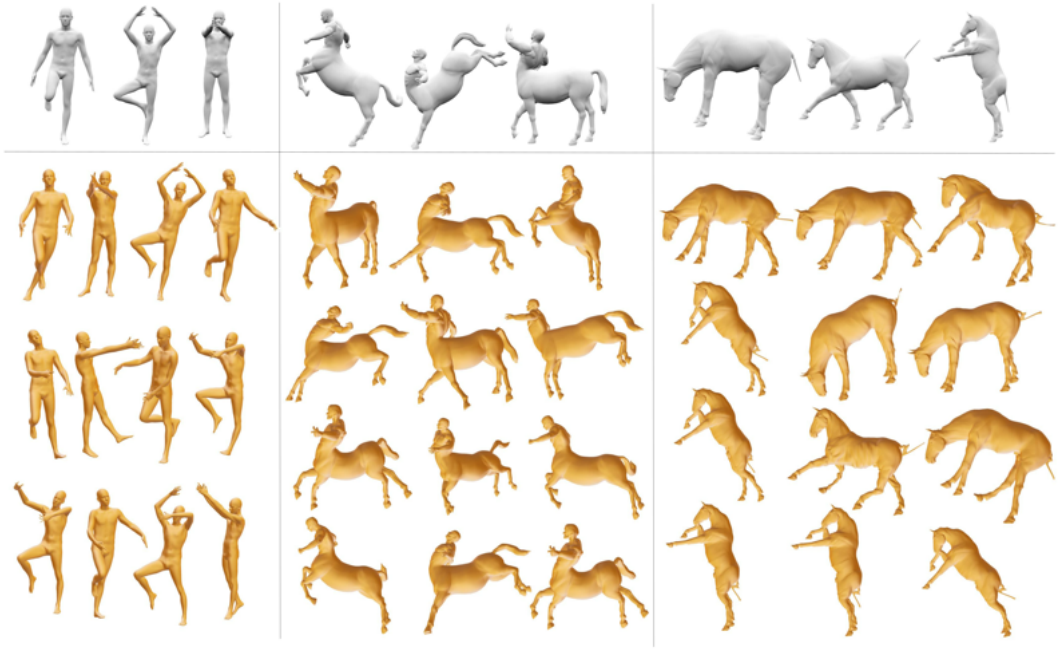

Figure 2.5: Training GLASS on the human, centaur, and horse meshes using the 3 examples each (top). (Bottom) We show random samples from the latent space, which combine different properties learned from the example deformations. Please see Appendix A for more generation results.

between 3 and 10 landmarks.

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 dis-

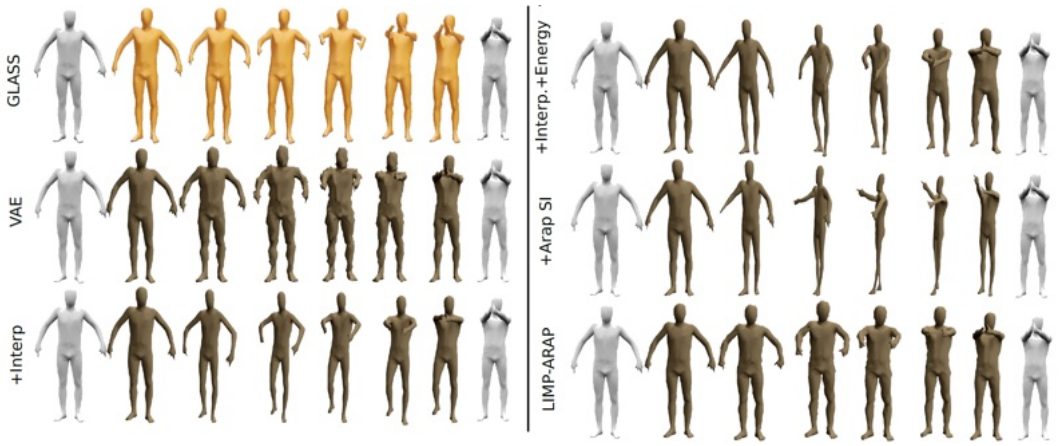

Figure 2.6: We compare the interpolation results between our method, several ablations of our method, and prior work.

Table 2.1: `Surface smoothness/Coverage` with respect to excluded set, of generated samples. Lower is better.

Data	Vanilla VAE	+Interpolation	+Interp. +Energy	LIMP	Deep-MCMC	+ARAP SI [30]	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**

tance between corresponding vertices of shapes s and g. We use Euclidean distance to measure shape similarity, as the shapes in both sets share consistent topology and are in correspondence.

(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. Additionally, since these are deformations of the same underlying identity and share a common topology, we use Euclidean distance to measure point-wise motion, and its standard deviation to identify sudden jumps.

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 2.4), we compare methods using

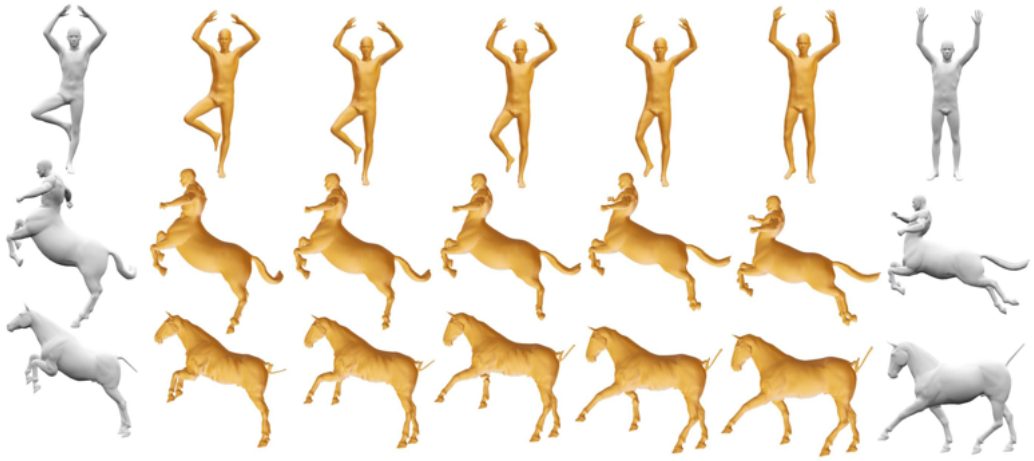

Figure 2.7: *Interpolation results.* In gray, we show two landmark shapes. In gold, we show the decoded meshes after we linearly interpolate the latent space between these two landmarks. All models are trained on only 5 landmarks. See Appendix A for more interpolation results.

Table 2.2: L2 Error wrt excluded DFaust frames and reconstruction error of those excluded frames. Lower is better. The table shows results with GLASS trained on 5 frames from 5 different motion sequences (indicated by "DFaust-index")

Data	Vanilla VAE	+Interpolation	+Interp. +Energy	LIMP	+ARAP SI [30]	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**

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 [35]:* This method is motivated by the training strategy proposed in LIMP [35]. 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 LIMP to use ARAP energy. (v) + *ARAP Shape Interpolation [30]:* 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 [67]:* 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 2.1 and 2.5 and additional generation results in Appendix A). Our generated poses look substantially different from the training data and combine features from multiple input examples.

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 2.4 and quantitative evaluations in Table 2.1. 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). In particular, we ensure plausibility of generated results by a combination of taking small adaptive steps from shapes in the current set and the data-driven pruning by Maximal Marginal Relevance (Eq 2.8). The lower Mesh Smoothness scores indicate that our method generates shapes free from common artifacts such as pinching and face-flipping—issues often seen in rig-based methods when joint limits are exceeded and shape-specific pose correctives are absent. Importantly, we achieve this without explicitly applying pose correctives, instead leveraging data to learn and enforce shape-specific plausibility.

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

Table 2.3: `Surface smoothness/ARAP energy/Interpolation Smoothness` 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 2.4: Ablation study results – `Surface smoothness/ARAP energy/Interpolation Smoothness`

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**

our results in Figure 2.7 (more results in Appendix A) and comparisons in Figure 2.6 with corresponding stats in Table 2.3. 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. In particular, the lower Interpolation Smoothness score indicates that GLASS produces shapes that closely follow the interpolation path, maintaining plausibility without relying on annotations such as joint-angle limits typically required in rig-based generation [42]. *ARAP Shape Interpolation* is performing consistently worse than all baselines except Vanilla-VAE - this is because while the underlying deformation is non-linear, this baseline follows a linear path in the rotation space. Interpolation-based baselines sometimes yield smoother interpolations, something we would expect to be true for simpler, linear motions. However, as seen in Figure 2.6, 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. As a data-driven method, GLASS does not explicitly prevent unnatural joint bending at arbitrary angles. However, this is implicitly mitigated by the data constraint, as we only take small steps from the current set.

Table 2.5: 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**

Dynamic Faust Interpolation Dynamic Faust [75] (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 2.2. 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 [37], as a reference. Originally, this method was trained on 230k deformations, most of them sampled from the SMPL model [4] 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 [4] that are augmented with a number of additional deformations generated from our model (see Table 2.5). 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 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 ARAP-Reg [36]. Table 2.4(1,2) shows this improves all metrics.

Next, we consider our perturbation strategy (Section 2.3.3 i, ii) and add it to both vanilla and energy-guided VAE (Table 2.4.3, 2.4.4). 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 2.3.3 iii). We add it to both baseline techniques that have perturbation, and report results in Table 2.4.5, 2.4.6, 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.

2.6 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. Additionally, the diversity of our generated results is inherently limited by the variability present in the initial dataset. For example, if a bending pose is absent from the starting

set, our method is unlikely to generate such a pose. To mitigate this limitation, we sample the initial set using farthest point sampling where applicable, as described in the Dynamic FAUST Interpolation experiment (Section 2.5).

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.

In the next chapter, we employ learnable priors (albeit not as energy functions) in aiding the creation of dense shape spaces from sparse initial sets.

Chapter 3

BLiSS: Bootstrapped Linear Shape Space

In this chapter, we continue our investigation into learning generative models from sparse datasets in the context of morphable models. Specifically, this chapter focuses on learning generative spaces for human bodyshapes (with fixed pose). Morphable models are fundamental to numerous human-centered processes as they offer a simple yet expressive shape space. Creating such morphable models, however, is both tedious and expensive. For example, SMPL [4] and the follow-up models were learned from thousands of shapes acquired via expensive capturing setups and registered by expert artists. The main challenge is establishing dense correspondences across raw scans that capture sufficient shape variation. This is often addressed using a mix of significant manual intervention and non-rigid registration. We observe that creating a shape space and solving for dense correspondence are tightly coupled – while dense correspondence is needed to build shape spaces, an expressive shape space provides a reduced dimensional space to regularize the search. We introduce BLISS, a method to solve both progressively. Starting from a small set of manually registered scans to bootstrap the process, we enrich the shape space and then use that to get new unregistered scans into correspondence automatically. The critical component of BLISS is a non-linear deformation model that captures details missed by the low-dimensional shape space, thus allowing progressive enrichment of the space.

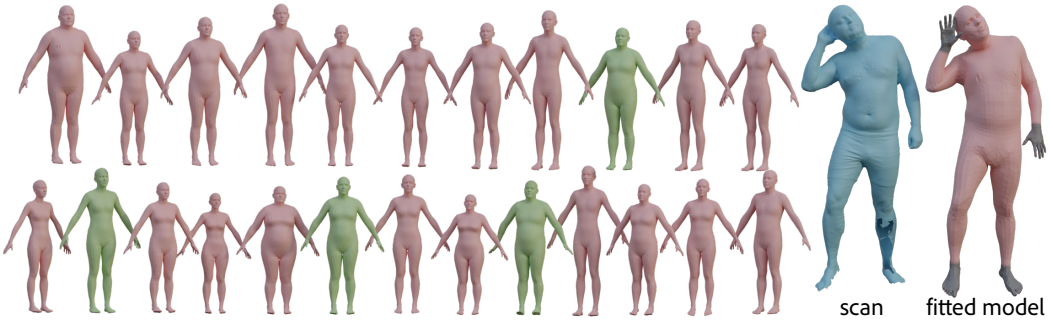

Figure 3.1: We present BLISS, which progressively builds a human body shape space and brings *unregistered* scans into correspondence to a given template mesh. Starting from as few as 200 manually registered scans (green samples), BLISS creates an expressive shape space (pink samples), performing on par with state-of-the-art models such as SMPL, STAR, and GHUM, while requiring only 5% of annotations compared to the others. (Right) Our space can then recover the body-shape parameters of raw scans by projecting them directly to ours.

3.1 Introduction

Morphable models [76, 4, 77] continue to strongly influence research towards human-centric workflows. This success is explained by the simple and versatile encoding of the underlying shape space, while providing interpretable handles for both shape and pose variations. The compact shape space has been extensively used for a variety of applications, including retexturing [78], shape editing [79], pose and illumination manipulation [80], animation [81], avatar creation [82], to name only a few.

While morphable models are widely considered useful, creating them is surprisingly tedious. Theoretically, given a set of 3D shapes (e.g., scans of human bodies) with vertex-level correspondence, morphable models can 'simply' be built using linear (e.g., principal component analysis (PCA)) or nonlinear (e.g., autoencoder [83, 5]) dimensionality reduction methods. The hurdles lie first in getting scans of many subjects, with a wide coverage of body shape and pose variations, and second in establishing vertex-level dense correspondence across the scans. Given these challenges, not surprisingly, only very few high-quality morphable models (e.g., SMPL [4], STAR [7], GHUM [5]) are publicly available.

The first hurdle has been significantly lowered with rapid advances [84, 85, 86]

in affordable, portable, fast, and robust (hardware) 3D scanning solutions (e.g., RGBD sensors, range scanners etc.). The second hurdle is algorithmic. The dominant approach to establish dense correspondence across the raw scans is to use non-rigid registration [56] to align scans with a template (body) mesh. This works well when the input shapes have limited variations and are clean. Unfortunately, when shape variability is large (as among scans capturing representative variations across a population) or contains holes and noise, successful registration must rely on manual intervention or strong shape priors. Thus, either users have to annotate landmark correspondence across the scans, or provide shape priors to regularize the registration step. Manual annotation is expensive and does not scale easily. Providing a shape prior is also tricky as generating one requires shapes in correspondence – this leads to a chicken-and-egg problem.

We provide a solution that, starting from a small set of registered scans, alternates between building an underlying linear shape space and utilizing the current shape space model to automatically bring new (raw) scans into correspondence. At the core of our approach is a nonlinear deformation setup, expressed in the form of a neural network, i.e., Neural Jacobian Fields (NJF) [9], that helps to predict dense correspondence for scans *close* to the currently modeled shape space. NJF is trained to add information beyond the current PCA space which is critical for registering new target scans, especially in the early stages where our linear shape space may yet not be sufficiently expressive. Once such correspondences are established, they enrich the shape space with additional scans. We repeat this process iteratively until all scans are brought into correspondence and a final shape space is achieved. We term this bootstrapping scheme Bootstrapped Linear Shape Space (BLISS). In its current form, BLISS does not handle pose correctives. We leverage the pose space of SMPL-H [4], optimizing over its pose parameters, while using BLISS's shape space to fit body shape.

We evaluate BLISS on the commonly used CEASAR dataset [87] and show that starting from only as few as 200 manually registered scans, we can jointly learn a shape space and automatically bring additional scans into correspondence. We

evaluate the expressive power of the learned shape space on held out test scans and show that our model performs on par with models that require all scans to be registered manually. We also compare with standard non-registration methods and demonstrate that our method is more robust against noise and holes in the scans. Finally, we apply the methodology of BLISS in the context of face shape space construction to emphasize its generalizability.

In summary, we make the following contributions: (i) an on-the-fly PCA shape space and correspondence learning framework: starting from only 200 registered shapes, we progressively enrich the model with new shapes and eventually reach a shape space that is on par with the one trained with 3800 registrations directly; and (ii) a novel combination of linear PCA and non-linear Neural Jacobian Field (NJF) deformation model that brings the target scans into better correspondence.

3.2 BLiSS: Related Work

3.2.1 Non-rigid registration

Registering two sets of raw scans (i.e., point clouds) is a long-standing problem [88, 89], typically consisting of two steps: (i) estimating correspondence between the source and the target scans; and (ii) minimizing the distances between each correspondence pair to bring the source closer to the target. Since this work is concerned with human bodies that often deform non-rigidly, we review how correspondences are estimated in non-rigid registration of 3D human data.

Optimization-based (ICP). When the source and the target points are roughly aligned in the ambient 3D space, correspondences can be approximated by seeking nearest points. Following this intuition, existing methods [90, 91, 92, 93, 94] alternate between searching the closest point and deforming the source points, which

Table 3.1: Comparison with SMPL [4], GHUM [5], DenseRac [6], and STAR [7] w.r.t. the number of registrations used in training respective morphable models.

Method	SMPL	GHUM	DenseRac	STAR	BLISS
# shape	3800	64000	2500	15000	200
space	PCA	PCA/VAE	PCA	PCA	PCA

can be seen as non-rigid variants of the classical Iterative-Closest-Point (ICP) algorithm [95, 96]. For fast convergence, such methods assume the two sets of points to be close enough, or require an "oracle guess" to initialize the correspondences.

Furthermore, these methods often require additional regularization terms to avoid local minima, *e.g.*, Laplacian [97] and ARAP [98]. They impose extrinsic heuristics to constrain the deformation, which do not always apply to the target tasks. In contrast, we employ the recently introduced Neural Jacobian Fields (NJF) [9] that implicitly learns an appropriate regularization in a data-driven manner. We also use NJF in our method as it has been shown to better distribute error by having a global Poisson solve to integrate local gradient (*i.e.*, Jacobian) information.

Learning-based shape matching. Global registration methods exist that match two human shapes without assuming they are close in 3D space. Instead of matching points in 3D space, they measure the similarity in a pre-defined feature space [99, 100, 101, 102] and leverage machine-learning techniques to estimate correspondences [103, 104, 105, 106], optionally refined with a global optimization [107, 108, 109]. The quality of these methods degrades significantly when the shapes are outside the distribution of the training data. More importantly, such methods do not yet handle noise in raw scans, and hence cannot be easily used in our setting.

3.2.2 3D Morphable Models for Humans

A standard human body model has to account for pose and shape deformation. In this work, we are particularly interested in the latter – the anthropometric variability across identities, and we focus the discussion on this aspect.

Table 3.2: Comparison with SMPL [4], GHUM [5], DenseRac [6], and STAR [7] w.r.t. the number of registrations used in training respective morphable models.

Method	SMPL	GHUM	DenseRac	STAR	GLASS
# shape	3800	64000	2500	15000	200
space	PCA	PCA/VAE	PCA	PCA	PCA

Parametric mesh. Representing human body parts as statistical shape models dates back to Cootes *et al*. [110] in 2D and Blanz & Vetter [76] in 3D. The latter has become the de facto standard for modeling 3D human shapes [111, 112], often called 3D morphable models (3DMM). The goal of a 3DMM is to adapt the template to each person by controlling the shape variations in a low-dimensional space. In the context of whole body, a myriad of work [56, 113, 4, 7, 114, 3, 5, 6] has been proposed for this purpose and has led to rapid progress in monocular and multi-view human body reconstruction [112].

Learning such a parametric shape space, however, requires firstly, a large database of body scans, and more difficultly, bringing them into correspondence by registering a common template mesh to them. Most models above are trained with thousands or ten thousands of registrations to body scans in CAESAR [87] and/or SizeUSA [115], curated with manual intervention for quality control (see Table 3.2). Another frequently overlooked challenge is that databases have each subject scanned in similar but not exactly the same pose (*e.g*., A pose in CAESAR) while the template is desired to be in one canonical pose (typically T pose). To factor out the pose variation in the data, an un-posing process is performed to bring registrations to the canonical pose [4], which we refer to as "canonical shapes" in the rest of the text. Any artifact introduced in this step will be kept in the learned shape space. Our formulation can take A-posed scans as input and output the canonical shapes in T-pose, requiring no un-posing before including them to training.

The most relevant work of Hirshberg *et al*. [91] explores a "semi-supervised" setting of co-registration of multiple scans similar to ours. However, their approach differs from us in several aspects: (i) they aim to "simultaneously" learn a morphable model and bring scans into correspondence in one shot, without iterations. Consequently, it amounts to a big optimization problem where one has to provide good initialization (e.g., via manual landmarks) and carefully anneal the weights of each term, which may be easy to break. Due to the complexity of the optimization, this method can handle hundreds of scans whereas we can scale our iterative pipeline to thousands of scans; (ii) they rely on model-free registrations with a

nearly isometric regularization term to capture information beyond the model space. While there is no publicly available code for us to perform an exact comparison, we compare to baselines where we employ a similar edge-preserving non-rigid registration approach and demonstrate superior performance.

Implicit surfaces. A well-known limitation of meshes is that it is limited in handling deformations that require changes in the topology. Recent work [116, 117, 118, 119] explores representing human bodies with neural fields [120], which does not assume a consistent mesh topology. They take the translations and rotations of body joints as input and estimate whether a query 3D point is inside the body or not. However, so far the effort has been devoted primarily in generalization of articulated poses, where large-scale motion capture datasets [121, 122] are used for training.

To help generalize to multiple subjects, it is encouraged to condition the networks on body shapes. However, in these methods, shape information is carried only in the locations of input joints, which is a very coarse anthropometric feature as two bodies can share the same joints but different surface shapes. In this work, we consider explicit surface meshes in order to better capture details in human bodies.

3.3 Approach

3.3.1 Overview

Given a large set of raw scans S of varied human body shapes in roughly a similar pose (e.g., A-pose), our goal is to learn a shape space in a canonical but different pose (e.g., T-pose) that captures the variation of plausible body shapes. We learn a shape space with respect to a predefined shape template topology; in our setup, we use the SMPL [4] template, denoted as T_{SMPL} with N vertices. We also assume having access to a small set of registered shapes R where a small set of scans S_R have been brought to the same topology as T_{SMPL} in the canonical pose via a manual non-rigid registration process to avoid any registration artifacts. Starting with R and T_{SMPL}, we iteratively expand R with new shapes from the unregistered scans U that are automatically brought into correspondence with T_{SMPL} and learn an enhanced

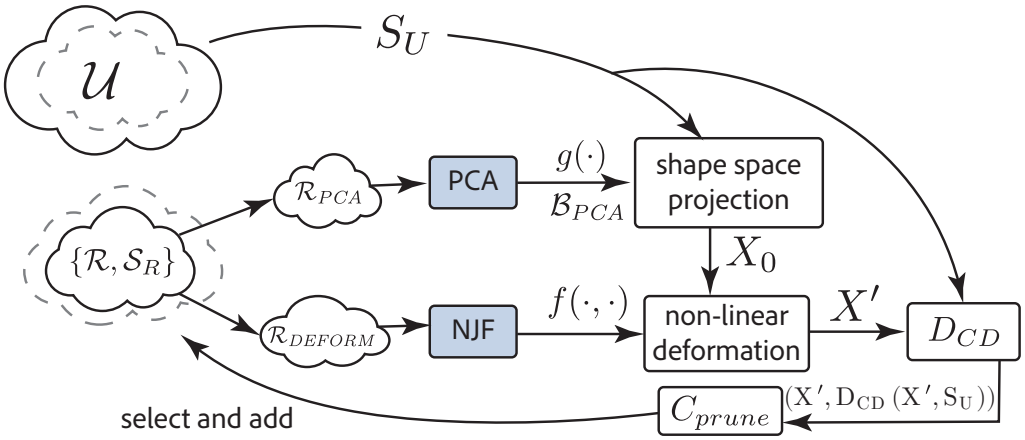

Figure 3.2: Given a sparse set of scans S_R, and their registrations R to a common template, we learn a linear shape space B_{PCA} using R_{PCA} and train a non-linear NJF-based deformation model using R_{DEFORM}. Then, given a scan S_U from a set of unregistered scans U, we project it to the PCA basis to obtain X_o and utilize NJF-based deformation to recover its registration to the template X' in the canonical pose. To enhance our shape space, we calculate the Chamfer Distance (D_{CD}) of registrations to target scans. We add all registrations where the distance falls within one standard deviation of the minimum distance to R_{PCA}. We repeat this process to jointly register raw scans and enrich our shape space.

shape space. Note that we always have $S = S_R \cup U$.

Our method takes as input a set of raw scans U (approximately in A-pose) and a set of manually registered scans R (in T-pose). Using the latter, we construct an initial shape space, which is then used to iteratively register the shapes in U to this space, aligning them to the T-pose. After several iterations, the process yields an expanded shape space that captures a wider range of body shapes, all in T-pose.

Our method works by deforming the template T_{SMPL} to closely match a new raw scan $S_U \in U$. This deformation model consists of two parts: (i) a PCA-based shape space B_{PCA} that provides a search space for shapes; and (ii) a neural deformation model f that maps shapes obtained by searching B_{PCA} to targets that better capture the shape details of the raw scan. The two deformation models work in tandem to jointly register scans and yield correspondence with T_{SMPL}, resulting in registered scans based on the current shape space. We then 'close the loop' by selecting a few new registrations based on their distance to the scan and adding them to recompute a PCA basis B_{PCA}, thus enriching the shape space further. We repeat this process for multiple iterations, with each pass progressively learning a richer

Algorithm 2

procedure BLISS(R, S_R, U, f, n)
▷ R = Registered Set, S_R = Corresponding set of scans
▷ U = Unregistered set of scans
▷ f = Deformation Model, n = Number of rounds
  for Round $\in [1,n]$: **do**
    $R_{PCA}, R_{DEFORM} \leftarrow R$
    $B_{PCA}, g(\cdot) \leftarrow PCA(R_{PCA})$ ▷ build shape sp.
    for $X, S_X \in (R, S_R)$: **do** ▷ build deform mod.
      $\{\alpha_i^\star\}, \theta^\star = g(S_X)$ ▷ Fit using basis
      $X_o = \bar{S} + \sum_{i=1}^{k} \alpha_i^\star v_{s_i}$ ▷ init. canon. shape
      $X' = f(X_o, S_X)$ ▷ Register with NJF
      $L_{vertex} = \|X' - X\|^2$ ▷ Vertex Loss
      $L_{Jacobian} = \|J' - J\|^2$ ▷ Jacobian Loss
      $f(...;\gamma_i) = f(...;\gamma_{i-1})$ ▷ Backprop
    end for
    $C = \emptyset$ ▷ New Candidate Registrations
    for $S_U \in U$: **do**
      $\{\alpha_i^\star\}, \theta^\star = g(S_U)$
      $X_o = \bar{S} + \sum_{i=1}^{k} \alpha_i^\star v_{s_i}$
      $X' = f(X_o, S_U)$
      $C \leftarrow C \cup X'$
    end for
    $D \leftarrow \text{ChamferDist}(C, U)$
    $\text{th} \leftarrow \min(D) + \sigma(D)$
    $C_{prune} \leftarrow \{c | c \in C, \text{ChamferDist}(c, U) < \text{th}\}$
    $R \leftarrow R \cup C_{prune}$ ▷ Update Basis
  end for
  return R
end procedure

shape space and using it to register raw scans. Figure 3.2 illustrates the pipeline of BLISS.

3.3.2 PCA-based Shape Space

We use a subset of the shapes in R, R_{PCA} to compute a PCA basis in a similar fashion to classical works like SMPL [4] and STAR [7]. Note that we use only a subset of the R and save the rest for the data-driven deformation model (see Section 3.3.3). Our shape space B_{PCA}, similar to others, is composed of a pose-corrective deformation basis allowing for pose-conditioned deformations and a shape basis that enables body-shape deformations. In our work, since we are primarily interested in learning a space of body shapes, we borrow the pose corrective directly from SMPL, which is denoted as $B_P(\theta) : \mathbb{R}^{\|pose\|} \rightarrow \mathbb{R}^{3N}$ as well as a rigged skeleton to pose T_{SMPL}, where $\|pose\| = 24 \times 3$, corresponding to 3 axis-angles for each of the 24 joints.

We represent the shape basis B_{PCA} with k shape eigenvectors $B_{PCA} := \{v_{s_i}\}$, where k is selected such that the shape variation in the dataset is explained using the k basis vectors. In this computed space, we define a new shape S_c in any particular pose θ as,

$$S_c(\{\alpha_i\},\theta) := \bar{S} + \sum_{i=1}^{k} \alpha_i v_{s_i} + B_P(\theta) S_p(\{\alpha_i\},\theta) := W\left(S_c(\{\alpha_i\},\theta), J, \theta, W_s\right), \quad (3.1)$$

where $\bar{S}$ is the mean shape, $S_p(\{\alpha_i\},\theta)$ is the posed shape before applying pose correctives, J is the joint regressor that provides the joint locations given the vertex positions in the shape, W_s is a fixed set of skinning weights, and finally, W is the skinning function as defined in [4].

Now, given a target scan S_U and a current set of shape basis vectors v_{s_i}, we optimize for the pose parameters and shape coefficients:

$$g(S_U) := (\{\alpha_i^\star\},\theta^\star) = \arg\min_{\{\alpha_i\},\theta} D_{CD}(W(S_c, J, \theta, W_s), S_U), \quad (3.2)$$

where D_{CD} is the Chamfer Distance and S_U is an unregistered raw scan.

We optimize Eq. 3.2 to find the shape in B_{PCA} that best matches the scan S_U while also optimizing for the pose parameters θ. In other words, the function g "projects" the raw scan S_U onto the shape space B_{PCA}. After optimization, we obtain the canonical shape that corresponds to the scan as $X_o := \bar{S} + \sum_{i=1}^{k} \alpha_i^\star v_{s_i}$. Note that due to the limited expressivity of the linear basis, X_o may not accurately represent S_U. We now seek a deformation model that can further enrich X_o with the details from S_U. This shape space optimization also provides a good initial point to seed our subsequent nonlinear deformation model, as explained next.

3.3.3 Neural Deformation with NJF

In our work, a nonlinear deformation is simply an assignment of new 3D positions to the vertices of the given (template) mesh. We adopt Neural Jacobian Fields (NJF) [9] as our nonlinear deformation model f. NJF trains an MLP to map triangles on a source mesh to a corresponding deformed triangle on a target mesh using only

local information. The key step is to have this training receive gradients through a differentiable global Poisson Solve layer to then directly predict the positions of the vertices.

We consider another subset of shapes in R, R_{DEFORM} where $R_{DEFORM} \subset R \setminus R_{PCA}$, and their corresponding raw scans to train NJF. For each shape $X \in R_{DEFORM}$, we first optimize for parameters $g(S_X)$ where S_X is the raw scan corresponding to X, giving us a shape space projection X_o. We then train NJF to map the canonical X_o to the canonical X, *conditioned* on the scan S_X that can be in any pose. Essentially, we ask our deformation model f to deform the result of our (current) shape space projection X_o to the target registration X that contains richer details. The deformation function f is conditioned on the raw scan representing the target, and can fix any residues not covered by the optimization step. Specifically, we train f with per-vertex L_2 loss,

$$L_{vertex} := \|f(X_o, S_X; \gamma) - X\|^2, \tag{3.3}$$

where γ represents learnable parameters, and a per-triangle Jacobian loss $L_{Jacobian}$ which supervises the ground-truth Jacobians J (see [9]), with our total training signal being,

$$L_{total} = 10 \cdot L_{vertex} + L_{Jacobian}. \tag{3.4}$$

We slightly abuse notation in Equation 3.3 – in practice, X_o and S_X are represented as features and not by vertex locations themselves. Specifically, we use Pointnet encodings [123] of both shapes. Details of the network architectures and the features used are in Appendix B.

With our initial shape space defined by R_{PCA} and a nonlinear deformation model trained with R_{DEFORM}, we can now use these in tandem to register new scans.

3.3.4 Closing the Loop

For each unregistered scan $S_U \in U$, we first fit the template T_{SMPL} to it by optimizing parameters $g(S_U)$ in Equation 3.2, to obtain the shape projection X_o in canonical pose. We then use the trained NJF to predict the final registration as $f(X_o, S_U; \gamma) \to X'$. The model X' is then "posed" to match the pose of S_U by using

Table 3.3: Evaluating ours against alternatives. (i) Learning a one-time static shape space from 400 available registrations provides an upper bound; (ii) and (iii) provide baselines replacing our non-linear deformation model with classical non-rigid registration. Errors are in cm.

Method	$\lvert R_{EVAL}\rvert$	initial $\lvert R_{PCA}\rvert$	$\lvert R_{DEFORM}\rvert$	regularizations	# shapes $\in U$	v2v (↓)
(i) FULLANNOTATION- PCA+NJF	29	400	400	✗	✗	*0.87*
(ii) BASELINE1- PCA + non-rigid	229	100	✗	small $\lVert\Delta v\rVert$	800	3.11
(iii) BASELINE2- PCA + non-rigid	229	100	✗	edge-preserving	800	3.26
BLISS (PCA only)	229	100	100	✗	800	1.31
BLISS (PCA+NJF)	229	100	100	✗	800	**0.90**

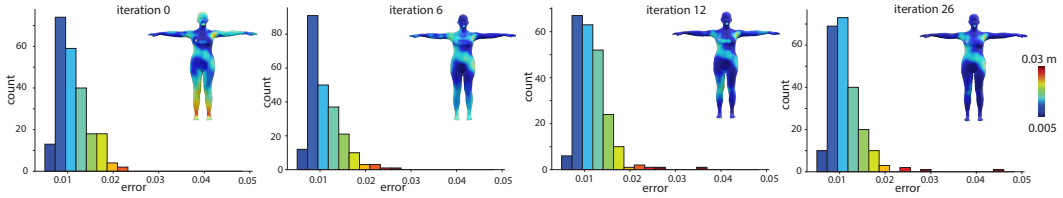

Figure 3.3: We show the histogram of the v2v error of the scans in our test set at different iterations of our method. We also color code the per-vertex error for an example scan. As our method progresses, the error decreases, and we observe a slight left shift in the histogram as the shape space improves. Insets show residue error on one scan over iterations.

the optimized pose parameters $\theta^{\star}$. We compute the Chamfer Distance to their corresponding scans, if the error falls within one standard deviation from the minimum error, we augment R_{PCA} with the new shapes X' in T-pose. Following a common statistical convention, we set the threshold at one standard deviation above the minimum error, which yields a reasonable number of new shapes (typically 30–50) while maintaining acceptable registration accuracy.

In the next iteration, the shape space will be updated by computing PCA with the augmented set R_{PCA}. The updated basis also provides new initial states for training our deformation model f. Note that we do not add the PCA projections X_o to R_{PCA} as it does not carry "fresh" information like X'.

We repeat the steps of constructing a PCA basis, learning an NJF based deformation model, and registering new scans for several rounds, with each round expanding the shape space (see Algorithm 2).

3.4 Evaluation

3.4.1 Dataset and Protocols

Dataset. We picked 429 full-body scans from the CAESAR dataset and had them registered by a professional artist, *i.e.*, $|R| = 429$. Note that the artist took 40-60 min per scan using a combination of landmark point specification, running nonrigid ICP, and then manually fine-tuning dense correspondence correction/specification (*e.g.*, around fingers, armpit, etc.), costing around $25 for each scan. We consider these artist-registered meshes as Ground Truth for evaluation, training, and, in the case of some baselines, as targets. Specifically, we first randomly sample two mutually exclusive sets R_{PCA} and R_{DEFORM} from R to train the initial PCA space and NJF respectively, where $|R_{PCA}| = 100$ and $|R_{DEFORM}| = 100$. We use all the remaining 229 registrations as R_{EVAL} for evaluation purposes unless noted otherwise. Given the limited size of the dataset (only 429 shapes), we select 100 samples each for the PCA and deformation sets. This provides a sufficient number of examples to train our deformation MLP, while reserving a substantial portion for evaluation.

Since the original CAESAR dataset consists of around 4000 scans, 429 of which we have registrations from the artist, we consider the rest 3.5k scans as unregistered scans U. In Algorithm 2 Ln. 14-18, scans in U are brought to correspondence, added to R_{PCA}, and contribute to the new shape space B_{PCA}, whereas R_{DEFORM} is kept fixed to the initial 100 artist registrations. Throughout the experiment, we always use k=11 basis for any PCA-based shape space, denoted as PCA. Note that despite R_{DEFORM} being fixed, in each round, since the basis of the shape space B_{PCA} changes, X_o changes, and consequently, the amount of details that NJF needs to compensate also changes. Hence, our NJF is rebuilt in each iteration (Algorithm 2 Ln. 5-11). At test time, since BLISS consists of a PCA shape space and an NJF network, whenever we evaluate only the learned PCA space, we denote it as "PCA only" to distinguish from running the full pipeline.

Metrics. For each registration in R_{EVAL}, we take the corresponding raw scan and obtain a registered shape using either our method or the baselines below. We measure the vertex-to-vertex (v2v) distance (since they share the same topology)

between the ground truth and the estimated canonical shapes, using the artist-annotated scan-to-template correspondences. When comparing shape spaces with different topology, such as GHUM, we start by non-linearly registering the target and our mean body shape models in the same pose. Then, we use barycentric coordinates on the corresponding face to map each point on our template to the target body model. Additionally, we calculate the vertex-to-plane error (v2p), which does not require meshes to have the same topology. We choose vertex-to-point (v2p) distance over alternatives like Chamfer Distance, as it more accurately captures differences between surfaces (as opposed to point wise distances) —making it better suited for evaluating how well the underlying identity has been predicted.

(i) **FULLANNOTATION**: We spare 29 registered shapes for evaluation and use all the remaining 400 annotated meshes to train a PCA model. We further train an NJF with the same 400 scans to add the missing details not covered by the PCA model, denoted as "FULLANNOTATION- PCA+NJF". This baseline represents scenarios where one trains the model *in one go* with all available registrations, without any bootstrapping schemes that leverage the unregistered scans. Hence, this can be seen as an *upper bound*. Since the goal of this baseline is to set the *upper bound* we chose the split such that a significant proportion is used for training, while still reserving some for evaluation.

Baselines. We consider several baselines, where NJF is replaced in our pipeline with classical non-rigid registration methods. Given an unregistered scan S_U, we first obtain the projection in the PCA space X_o and then optimize the location of each vertex on X_o, such that when posed with $\theta^\star$, the shape yields low Chamfer Distance to the scan S_U. This "free-form deformation" scheme can fall into local minimum easily even if we provide X_o as close initialization. Therefore we define new baselines where we constrain it with standard regularization terms: (ii) **BASELINE1**: vertices should not be deviating too far from the canonical shapes X_o, *i.e.*, $\|\Delta v\|$ should be small favoring smooth surfaces; (iii) **BASELINE2**: deformation should preserve edge lengths, *i.e.*, favor near-isometric deformations.

Finally, we also compare to existing shape spaces including SMPL [4],

STAR [7], and GHUM [5].

3.4.2 Results and Discussions

Progressive improvement of the shape space. First, we illustrate how our shape space is progressively improved, *i.e.*, becomes more expressive. In Figure 3.3, we show the histograms of the v2v residue error at different iterations of our method as it consumes new scans from U and visualize the error as heat maps in each inset image. One can observe that the error gets reduced as we have more rounds, *i.e.*, the correspondence quality improves progressively.

Further, we compare with a method that simultaneously burns all available 400 scans into model training – FULLANNOTATION. As shown in Table 3.3 upper part, it attains the lowest v2v error of 0.87cm on a smaller evaluation set of 29. In the bottom part, we can observe that, despite being trained initially with only 100 registrations, BLISS yields v2v error that is on par with the upper bound by automatically extending the set of registered scans to 800.

Effect of NJF as a non-linear registration module. In BASELINE1-2, we run Algo. 2 but estimate detailed shapes X' at Ln. 16 by non-rigid registration methods instead of NJF f. We then evaluate how well the resulting PCA shape space explains the test set R_{EVAL} and compare with our PCA shape space. In Table 3.3 bottom part, we observe that after consuming 800 unregistered scans, our shape space explains scans in R_{EVAL} with 1.31cm v2v error (PCA only), while NJF further reduces it to 0.90cm. Consuming the same amount of scans, shape spaces enriched with X' from non-rigid registration yield errors of 3.11cm and 3.26cm, respectively. This suggests using a data-driven NJF in the loop recovers better correspondence than optimization-based registration methods, and when included in R_{PCA}, it leads to a new PCA space B_{PCA} with richer information.

Figure 3.3 shows how the model improves over iterations/rounds (from left to right). While NJF helps add lost details, it is still limited by the sparse set of training pairs (100 in our case) and may not recover all the details.

Comparison to other shape spaces. We compare BLISS with the following existing shape spaces: (i) the classical SMPL [4] shape space trained with the regis-

trations of 3800 CAESAR scans, (ii) its follow-up STAR [7] which uses additional 11000 registrations of the SizeUSA dataset [115], (iii) GHUM which uses registrations for an additional proprietary dataset of 64000 scans (where a majority consists of body, hand, and, facial pose variations) along with CAESAR. GHUM presents a linear shape space as well as a VAE-based non-linear shape space, both of which we include in our comparisons. For each corresponding scan in R_{EVAL}, we optimize for the pose and shape parameters of each body model and report both the v2v and v2p errors in Table 3.4. We also show qualitative comparison in Figure 3.4 where we color code each optimized body model using the v2p error with respect to the ground truth artist provided registration.

Table 3.4: Ours, after absorbing 800 shapes from CEASER, outperformed SMPL, STAR, and GHUM even though we only used 200 registered scans, compared to their much larger number of scans.

Method	# shapes $\in R$	v2v (↓) (cm)	v2p (↓) (cm)
SMPL (PCA)	3800	1.72	0.62
STAR (PCA)	15000	2.15	**0.58**
GHUM-PCA	64000	6.74	3.01
GHUM-VAE	64000	5.89	2.63
BLISS	200	**0.90**	0.65

We observe that BLISS yields consistently lower v2v error than other shape spaces. We use 11 PCA components for SMPL, STAR, and BLISS while all the PCA components for GHUM. Our Ground Truth are artist-annotated registrations, which can still potentially contain errors which might have an effect on this gap. Nevertheless, our method performs on par with SMPL and STAR based on the v2p error and better than GHUM. Hence, the primary observation in Table 3.4 is that, despite starting from only a small amount of registrations (100+100), BLISS yields on-par expressivity compared to a model trained with an order of magnitude more registrations. We attribute this to the novel combination of linear PCA and non-linear NJF deformation model, as well as the progressive scheme leveraging such a hybrid deformation model for better correspondence.

Diversity of body shape spaces. We qualitatively show the diversity of body shapes represented by our shape space by randomly sampling our final PCA space

using farthest sampling in terms of vertex differences. Sampled shapes are shown in Figure 3.1. In Figure 3.5, we show shape variations captured by our shape space's top three PCA modes. Note that while gender is represented as a binary attribute in the dataset we trained on, the geometric features associated with gender (e.g., shoulder width) are continuous and can be interpolated smoothly.

In order to compare the diversity of our and existing shape spaces, we sample

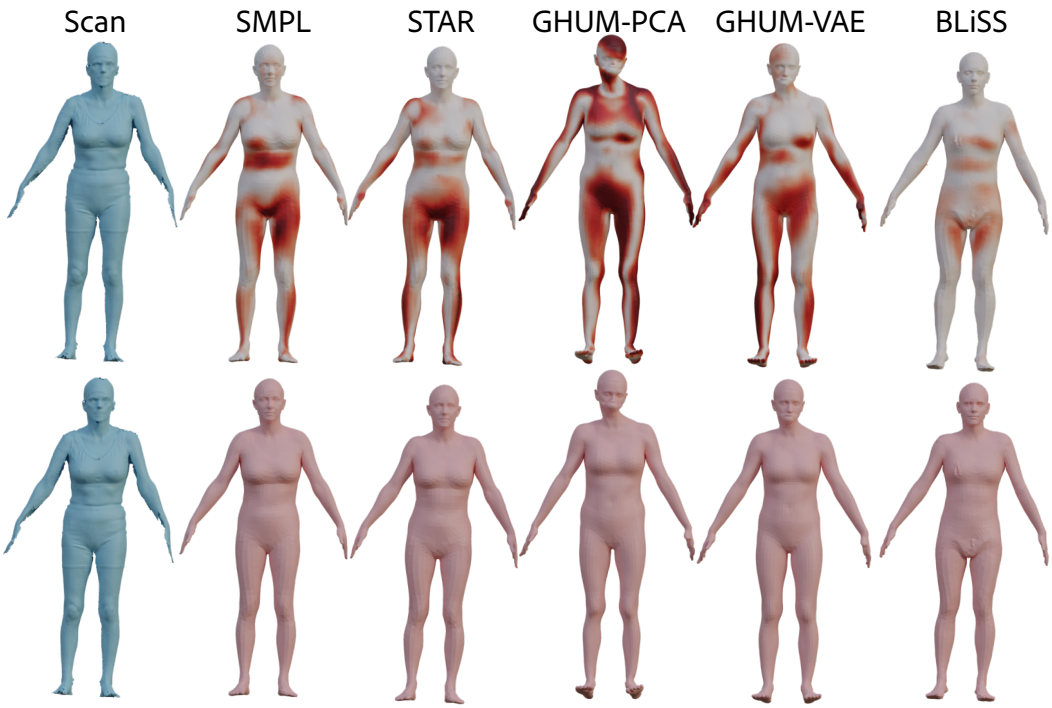

Figure 3.4: For a given raw scan, we register each body model by predicting pose and body shape parameters. (Top) Each result is color coded based on the v2p error in meters w.r.t. the ground truth registration provided by the artist.

Table 3.5: *Left:* We use Farthest Sampling to gather 500 shapes from each space. To determine the similarity between the different spaces, we calculate the distance between each shape in one space and its closest counterpart in all other spaces, including off-diagonal entries. We then report the average distance (in centimeters) for each possible pairing of spaces in both directions. Low values for (A, B) and (B, A) suggest the two spaces are similar. *Right:* We compute the diversity of samples inside each space, with higher values indicating more diversity.

Space	Ours	GHUM	STAR	SMPL
Ours	0	3.57	1.38	1.46
GHUM	4.03	0	3.65	3.71
STAR	1.79	3.74	0	1.37
SMPL	1.90	3.75	1.36	0

Space	Ours	GHUM	STAR	SMPL
Diversity	4.10	4.48	3.96	4.14

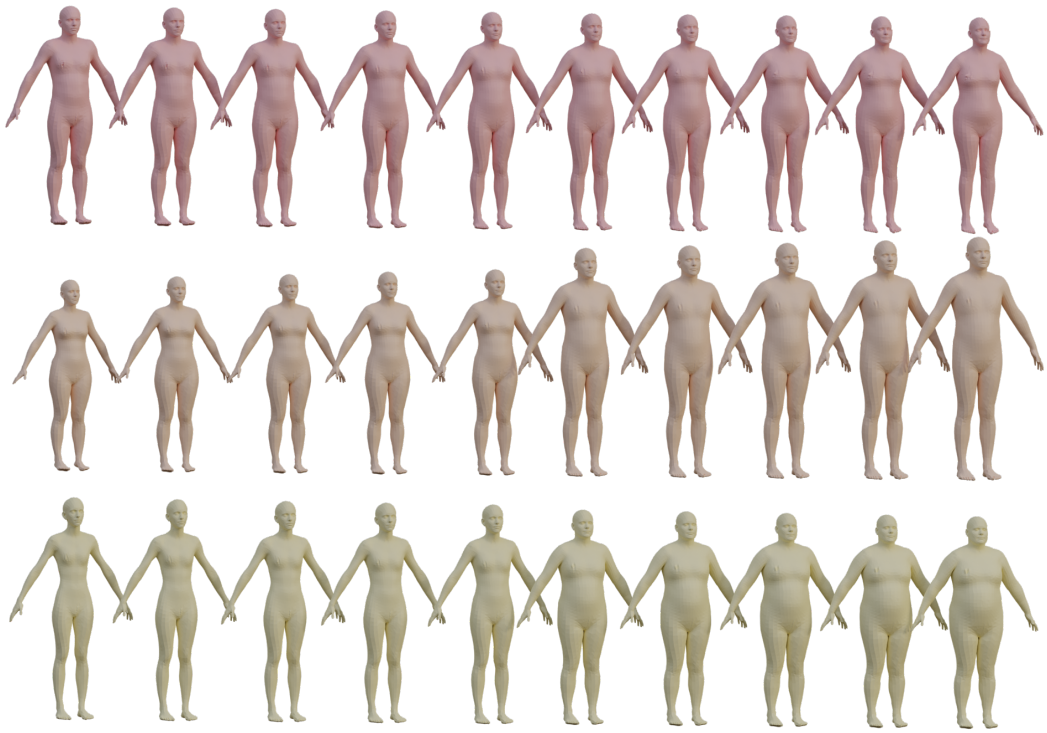

Figure 3.5: We show shapes along the top three principal directions in different rows, and observe variations in gender, height, and weight along the respective PCA modes.

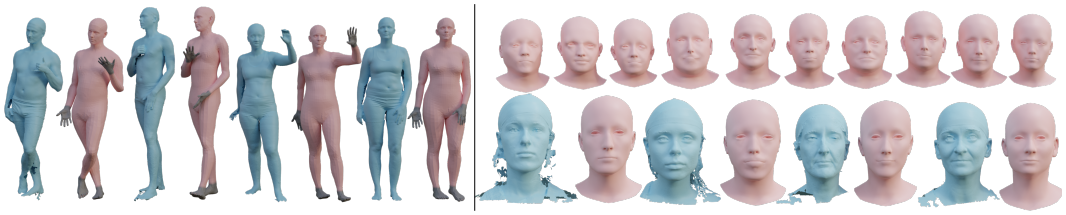

Figure 3.6: *Left:* Registration (pink) of noisy scans (blue) with our final shape space. Since our model does not capture finger-level details, after optimization, the joints corresponding to the greyed-out regions are reset to default poses. *Right*: We show sampled faces from our final face-shape space after growing it from 20 $\rightarrow$ 800 shapes. We observe a variety of face changes in the cheek and nose regions. (Bottom) We take the test scans from the COMA dataset (in blue) and register them in our final face-shape space, which is shown in pink.

500 body shapes in each shape space by furthest point sampling. For each sampled body shape, we compute is nearest sample within the same shape space by measuring the v2v error. We report the average of such pairwise sample error on the right in Table 3.5 where higher pairwise distance means a diverse shape space. As shown quantitatively, the diversity of our shape space is on par with existing

shape spaces. We also compute, for each sample in a given shape space, its nearest neighbor among all samples in the other shape spaces (by Chamfer distance instead of v2v or v2p, as GHUM is higher resolution and follows a different topology). For each shape space in each row, we then compute the pairwise sample distance to each shape space in the columns. Smaller numbers indicate higher similarity between shape spaces. We observe that our shape space is closer to the SMPL and STAR shape spaces. We observe higher distances for GHUM, indicating lower similarity to the other three shape spaces. This is likely because GHUM was not trained on CAESAR shapes or scans.

Number of PCA components. While we use k=11 basis consistently for all PCA-bsaed methods, we also analyze the effect of using more bases with k=30, 50, 100. Using higher number of basis increases the expressivity of PCA, but empirically we observe very little change in our metrics – to the order of 10^{-5} – by considering more PCA components, and thus stick to using just k=11 in all experiments.

Compute. Our method takes approximately 6 hours to process 800 CAESAR scans on a single NVIDIA GTX 1060 Ti. The primary bottleneck is the optimization step described in Eq. 3.2. In contrast, the PCA computation is relatively fast at our resolution of 6890 vertices (similar to SMPL), and the neural deformation with NJF is efficient since it involves a simple feedforward pass through a shallow MLP.

Application. A typical application of a body shape space is to predict a given raw scan's shape parameters. We demonstrate the use of BLISS shape space in such an application in Figures 3.6 and 3.1 (right) . Since our work focuses on capturing body shape variation, we optimize for pose in SMPL's pose space. For each raw scan we use 9 manually annotated landmarks to estimate the initial pose, then we estimate the body pose (with SMPL) and shape parameters in our shape space. We observe that our space accurately estimates the body shape despite the scans being noisy.

Face registration. To demonstrate the generalization of our approach, we sample 20 faces from FLAME [124] to create an initial face shape space. We then

iteratively register faces from the COMA [125] dataset. Note that we use the NJF module trained on full-body human scans. Registrations shown in Fig. 3.6. We do *not* assume paired data for this task and instead exploit NJF's invariance to topology and use the pre-trained NJF to deform the faces. NJF was trained on centroids and Wave Kernel Signature as input features; these features are computed for the linearly registered face scans and fed to NJF to add further details. We iteratively train BLISS until we add 800 face-shapes to ours.

3.5 Conclusions

In this chapter, we presented a framework that takes in a small set of artist-annotated scans along with a much larger corpus of unregistered scans, and jointly learns a (linear) shape space while progressively bringing the unregistered scans into correspondence. At the core of our approach is a novel formulation that continuously refines the underlying shape space and a learned nonlinear module that automatically registers models 'close' to the current shape space. The learned module encodes the artist's registration and refinement process, thus providing a meaningful prior to enable learning from sparse guidance. We demonstrate that our approach trained only with 200 registered scans, can produce competitive performance compared to established shape space models trained using thousands of registered scans.

One limitation of our method is that it does not capture pose corrective shape space. Learning a pose-corrective space in conjunction with the shape space would require all unregistered scans to be available in multiple poses; such a scan dataset is hard to curate in the first place. Another limitation is that the learned space is linear. We can explore the use of nonlinear shape spaces, such as AE or VAE, but it poses a challenge of growing robustly on limited data, especially in the initial rounds of the approach. It is also worth noting that our method cannot handle hands in complex poses due to the lack of finger articulation in SMPL's pose space.

Thus far, this thesis has focused on generating static variations from sparse datasets. In the final work, we shift our attention to creating dynamic 3D data under similar sparse data conditions. Unlike GLASS and BLISS, which focus on learning

shape spaces, Temporal Residual Jacobians introduces a novel approach to learn a spatio-temporal field from sparse guidance, enabling motion transfer to previously unseen characters.

Chapter 4

Temporal Residual Jacobians for Rig-free Motion Transfer

In this chapter we extend our exploration of sparsely guided generative models to the problem of generating dynamic 3D content. To that end, we introduce Temporal Residual Jacobians as a model to enable sparse-data-driven motion transfer. In this work, we train our models on 5-10 instances of a motion category and effectively transfer the learned motion to unseen characters. The method presented here does not assume access to any rigging or intermediate shape keyframes, produces geometrically and temporally consistent motions, and can be used to transfer long motion sequences. Central to our approach are two coupled neural networks that individually predict local geometric and temporal changes that are subsequently integrated, spatially and temporally, to produce the final animated meshes. The two networks are jointly trained, complement each other in producing spatial and temporal signals, and are supervised directly with 3D positional information. During inference, in the absence of keyframes, our method essentially solves a motion extrapolation problem. We test our setup on diverse meshes (synthetic and scanned shapes) to demonstrate its superiority in generating realistic and natural-looking animations on unseen body shapes against SoTA alternatives. Since this work generates dynamic content, the results are best judged by viewing the motion-transfer videos available at `https://temporaljacobians.github.io/`.

4.1 Introduction

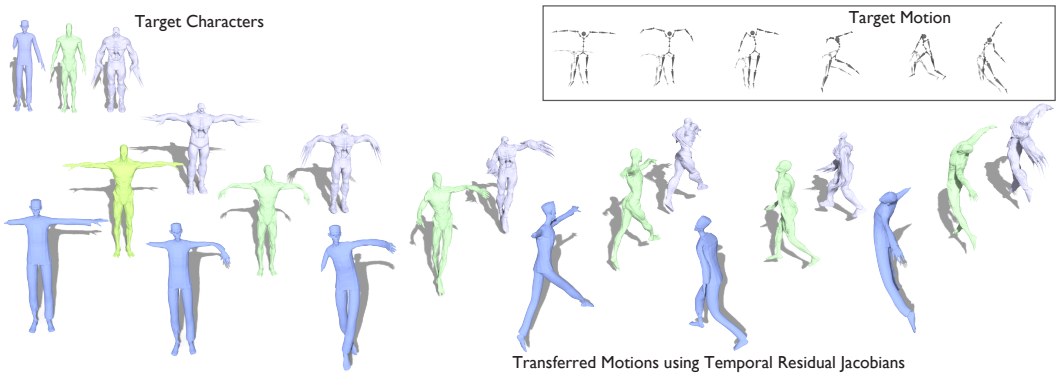

Figure 4.1: Given a stick figure dance motion (top-right), *Temporal Residual Jacobians* retarget the animation to unseen, unrigged meshes (top-left) across time, producing realistic motion dynamics. Please refer to the webpage for videos. Our method can be trained on limited data, does not require rigged models or skinning weights during training or inference, and does not assume paired sequences or registration to any canonical template mesh. The method was trained on other bodyshapes: no target characters were seen during training. All results in the paper and the webpage were obtained with automatic feature correspondences and without any postprocessing or smoothing applied.

A major challenge in character animation is to transfer the motion of a source (skeletal) system to a diverse range of target characters in a realistic manner. The traditional approach to achieve this involves using a *rig*, which connects a skeleton to the character's surface and manages the surface motion through a variety of constraints and parameters. Target movement is then conveyed from the skeleton to the surface by either simulating the physics of the muscles and fat or by using tailored sets of skinning weights. These weights can be manually created by skilled artists or derived from pre-existing rigged models [126, 127]. Despite its simplicity, rigging can be time consuming to set up and complicated to transfer to new target shapes; it may also fail to accurately capture dynamics.

There has been growing interest in developing surface deformation techniques that are more flexible and efficient than traditional rigging-based approaches and can utilize available volumes of full-body motion capture data. One possibility is to train data-driven methods to learn a low-dimensional parameterization, such as a morphable humanoid template [4] or a neural space deformation [128], to provide controllable handles for shape and pose-aware manipulations. However, such meth-

ods do not capture continuity over time and can overlook subtle motion dynamics essential for enhancing the realism of the generated motion sequences. To add time-dependent effects, corrective vertex deformations, similar to DMPLs [4] and Soft-SMPL [129], have been introduced in multistage workflows. Unfortunately, such methods do not account for elementwise temporal inter-relations and have limited generalization to unseen characters.

We aim to transfer a source motion, expressed as joint angles on a stick figure, onto a target shape, specified as a (rig-free) mesh. We want to do so without access to any rigging on the target shape, and we also want to avoid any fixed template or morphable shapes. A desirable solution should address several challenges: (i) handle rig-free meshes and/or scans with arbitrary topology; (ii) produce plausible transfers to diverse shapes; (iii) achieve continuity over space and time and thus avoid artifacts (e.g., broken meshes, jittery motion); and (iv) work with long motion sequences without significant drift. The first two problems are partially handled by rig-free pose transfer (i.e., transferring a pose to a target mesh) methods [9, 130, 131]. While such methods produce plausible single frames, they lead to jittery motion transfer and motion-induced geometric artifacts like shearing.

We propose a novel approach that learns local spatio-temporal changes to produce natural-looking motion transfers. Technically, we achieve this via a new representation in the form of *Temporal Residual Jacobians* that temporally links spatial predictions and is directly supervised using example motion sequences. We jointly train two neural networks to individually predict local spatial and temporal changes. They are coupled by spatial integration with a differentiable Poisson solve, and temporal integration with a neural ODE. A **key technical insight** is that instead of having the neural ODE predict per-frame mesh deformations, it is more effective to predict initial deformations independently of time via a base posing model (we use Neural Jacobian Fields [9]), and then have the neural ODE predict *residual deformations*, linked over time, as corrective factors that improve temporal coherence. Figure 4.1 shows how a stick-figure control motion produces target motions for different characters, without requiring registration to any standard template or

character rigs.

We evaluate the effectiveness of our method for motion generation to different character bodies (e.g., humanoids, animals, Mixamo characters, scans) and different motions (walk, run, jump, punch, dance). We compare our approach to alternatives, when available. We provide qualitative and quantitative results using the AMASS [122], COP3D [2], and 4DComplete [132] datasets.

In summary, our primary contributions are: (i) a novel method that enables motion transfer via Temporal Residual Jacobians and can be trained directly using positional data; (ii) local predictors that can be integrated in space and time to create natural looking character animations; and (iii) a robust pathway to transfer realistic character motion without the need for explicit rigging or learning a parameterization using any canonical template shape.

4.2 TRJ: Related Work

4.2.1 Parametric Shape Deformation.

These 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, explicit [38] or neural [128], blendshapes [39], skinned skeletons [40], Laplacian eigenfunctions [41], and several other variations. Linking the parameters to the shape's surface often requires manual annotation of weights (commonly known as weight painting) in 3D authoring tools. Alternately, given sufficient data (i.e., meshes, rigs, skinning weights), end-to-end training can produce realistic neural rigs, as demonstrated by Pinocchio [133], RigNet framework [127], skinning-based human motion retargeting [134], and skeletal articulations with neural blend shapes [135]. Unsupervised shape and pose disentanglement [136] proposes to learn a disentangled latent representation for shape and pose, which can be further used to transfer motion using shape codes. This requires meshes to be registered and have the same connectivity. To animate these parametrized shapes through time, the parameters are varied over time and the dynamic weights animate the mesh. These methods require access to body templates and/or rigs and

can produce results that are jittery due to loose coupling of the individual frame predictions.

4.2.2 Dynamic Motion.

It is possible to model temporal surface effects by simulating the underlying soft tissues using finite element methods (FEM) [137, 138]. This direct simulation is typically slow and requires artists to design the underlying bone and muscle structure [139]. Approaches have been developed to overcome the stiffness problem in FEM to accelerate simulating these systems [140] or to solve the problem in a lower-dimensional subspace [141]. For the specific case of rigged human characters, Santesteban et al. [129] add soft-tissue deformation as an additive per-vertex bump map on top of a primary motion model; AMASS [122] imparts secondary motion using the blending coefficients of the DMPL shape space [4]; and Dyna [142] learns a data-driven model of soft-tissue deformations using a linear PCA subspace. These efforts, however, assume access to primary motion via a skeleton rig and are restricted to humans, registered to a canonical template. Complementary dynamics [143, 144] models physically-based secondary motion in the subspace complementary to that spanned by an animation rig. This approach adds automatic secondary motion to arbitrary animated objects, but requires the target shape to be rigged, is not designed for deformation transfer (the base animation is part of the input), and is tied to a specific hand-coded secondary physics model. DeepEmulator [145] achieves a similar effect using a local-patch-based neural network to learn the secondary behavior, but again requires the primary motion as input and does not support deformation transfer.

4.2.3 Discrete Time Motion Models.

Given their ability to model time, deep recurrent neural networks have also been used to model shape sequences. Fragkiadaki et al. [146] use LSTMs to predict short human joint motions given initial frames. Harvey et al. [147] leverage LSTMs for in-betweening to predict intermediate joint motions. He et al. [148] learn a motion field for joints through time. In all these methods, the mesh itself is deformed via rigging, and since joint motion does not encode bodyshape, they do not suffi-

ciently represent secondary motion. Also, being discrete time representations, these approaches must train on large datasets of joint motion. Qiao et al. [149] instead use mesh convolutions with LSTMs to deform vertices through time. In our work, we use neural ODEs as they can model time continuously instead of discretizing time and modeling the sequence using LSTMs. Further, vertex-based deformation models are susceptible to artifacts like normal-inversion as we demonstrate in our evaluation section.

4.3 Approach

Given an unrigged, triangulated mesh of a 3D character, we aim to animate it by motion transfer from an available motion described by relative joint angles (stick figure motion) at each time step. The relative joint angles are represented as Euler angles and are defined at each joint with respect to its hierarchy in SMPL's kinematic tree (cf. [148]). We obtain these joint-based motion representations from the AMASS dataset [122]. Since we aim to animate the mesh itself from the joints' motion, we seek to learn a mapping from the joint representation to the positions of the given mesh's vertices, and to do so at each time step while ensuring we generate a smooth and artifact-free mesh animation. We supervise our setup with ground-truth meshes from the AMASS dataset [122].

We parameterize such a character $X \in \mathbb{R}^{N\times 3}$ as assigning positions to each of its N vertices of the underlying mesh. Thus to impart a motion to a mesh, we perform this assignment at every time step. Given a shape X_0 as a triangulated mesh, in its initial pose configuration, along with per-frame pose configuration M_t that describes the target pose at time t, we aim to predict the shape X_t at each time t generating the full motion sequence; essentially learning a mapping from relative joint orientations (as defined by SMPL's kinematic tree[4]) to mesh deformations (typically, from 30-50 joints to 50-100k vertices). We aim for a data-driven method that generalizes to new characters and does *not* rely on shape rigging or intermediate shape keyframes. Another desirable property is to do so with limited data (e.g., working with 5-10 motion examples), as obtaining full-body 3D motion data is

non-trivial.

Our key observation is that we can robustly train neural networks to predict changes *local* in both space and time, which can then be integrated across space, using a differentiable Poisson solve; and across time, using Euler equations to handle an ODE numerically. Integrating across space helps maintain plausibility of the entire shape, while integrating across time helps model a realistic, time-coherent motion. Further, local predictions help higher generalization capability to unseen body shape; while the spatio-temporal integration ensures smoothness of the sequence, without jitters or undesirable shearing artifacts. Our end-to-end differentiable formulation allows training the networks directly using example training sequences, without requiring factorized spatial and temporal motion signals. For spatial handling, we extend the representation from local deformation encoding [150] and affine mapping framework Neural Jacobian Fields (NJF) [9], which learns an affine transformation field that is sampled at each mesh face and integrated into vertex positions via a Poisson solve. For temporal coherence and consistency in these predicted Jacobians, we couple temporal signals across the character motions using a neural ODE framework [8] via a novel Temporal Residual Jacobian representation. Predicting *residual deformations* that correct the predictions of a base model for time-coherence turns out to be more effective than trying to make the base model itself time-coherent.

4.3.1 Overview

Training: Since our goal is to map temporally varying joint angles to full mesh animation, during training we assume this mapping is available i.e., we have one-to-one mapping across time between joint orientations and the corresponding meshes. Thus, we use the motion sequences in AMASS [122] as our training dataset for humanoid shapes. Thus, during training, our algorithm takes in the joint angles at each time step, the mesh itself at time $t = 0$ in its starting pose, and the full mesh sequence is used for supervising our neural networks.

Inference: At inference time, our algorithm simply takes in an unrigged character

provided as a triangulated mesh. Our trained method is then used to animate this mesh from joint-sequences of motions sampled from AMASS. These meshes can be obtained in-the-wild or from character datasets like Mixamo. For shapes in the AMASS dataset since we have the ground-truth motions, we evaluate our framework quantiatively. We only show qualitative motion generation for other shapes in the work, as there is no ground-truth solution.

Motion and Shape parameters: The motion sequences in AMASS were obtained from live captures of subjects performing different motions; these were then parametrized in SMPL's [4] shape and pose space, thus providing a mapping from joint orientations to 3D mesh pose - i.e., vertices of the mesh oriented to match the pose represented by the provided joint orientations. We use this mapping to supervise our framework. Specifically, this data framework allows us to sample motions across both shapes and poses, by varying the respective low-dimensional parameters β and α. In our work, the α translate to joint angles at each time step and we use the source live-capture subject's β as is, and pass it as a conditioning variable to our method.

4.3.2 Preliminaries

Our framework uses two components – one to learn spatial mesh deformations (reposing) and the other to learn temporal signals to generate a temporally coherent mesh motion. We describe these below.

Neural Jacobian fields, as introduced in [9], encode local deformations by defining a field via map ϕ, defined over space, that can be sampled at the N vertices of the surface mesh. Specifically, given ϕ, we compute Jacobians in the basis of the triangles of the surface mesh as,

$$J_i = \phi \nabla_i^T \tag{4.1}$$

where ∇_i is defined as the gradient of triangle t_i in its basis B_i defined at the triangle's centroid. Thus, given a learned map, we obtain each triangle's (estimated) affine transformations J_i. Recovering the vertex positions from this per-triangle as-

signment of affine transformations is then done via a least-squares formulation that reduces to solving a Poisson system given by (cf. [151]),

$$\phi^* = L^{-1}\rho\nabla^T J \tag{4.2}$$

where ρ is the mesh's mass matrix, $L = \nabla^T\rho\nabla$ is the cotangent Laplacian, and J is a stack of (estimated) Jacobians J_i. This solution gives a unique mesh up to a translation, which we fix using X_0, the mesh at $t = 0$. Mesh positions directly supervise the neural map ϕ via a differentiable Poisson solver [152]. Note that, by leveraging sparse matrix representations in the Poisson solve, we can efficiently handle high-resolution meshes (up to 200k vertices) without encountering memory overflow on standard GPU hardware.

Thus, NJF allows us to learn affine deformations of a given mesh. In our work, the map ϕ is a trainable neural network that predicts local deformations of the mesh's triangles. Given conditioning parameters, the trained map can then be used to predict the Jacobians of each triangle in the mesh at each time step, generating a time-coherent animated 3D mesh.

Neural ODEs Chen et al. [8] represent differential equations with neural networks to model the dynamics of time-varying systems (motion sequences in our setting). Specifically, given a function $f(t;\theta) := \frac{\partial J}{\partial t}$, where f is a neural network with parameters θ and t being time, the variable J can be integrated out at t as

$$J(t) = \int_0^t \frac{\partial J}{\partial t} dt = \int_0^t f(t;\theta)dt + J_0. \tag{4.3}$$

The neural network can be directly supervised with J values as the network output utilizes a black box differential equation solver without explicitly discretizing the dynamic system. This leads to better estimation, benefits from constant memory updates (as opposed to explicit hidden states in RNNs, LSTMs, etc.), and trades numerical precision for speed.

Our work seeks to learn how the triangle Jacobians move through time. Thus, our neural ODE operates in the Jacobian space, with Jacobians being the variables

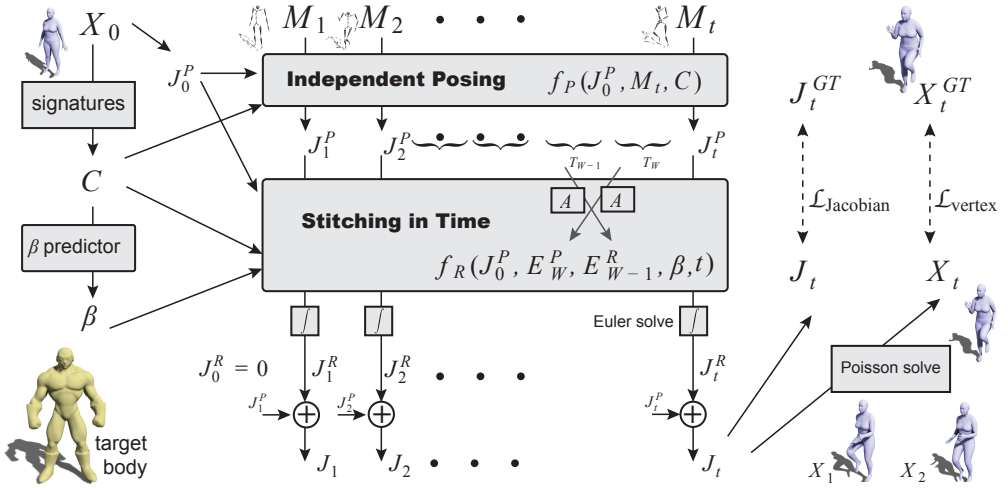

Figure 4.2: Method overview. Starting from input stick figure motion ($\{M_i\}$) and a target body shape (X_0), Temporal Residual Jacobians makes local predictions, using primary f_P and residual f_R MLPs, to predict spatial and temporal changes to per-triangle Jacobians respectively. f_P (middle, top) predicts per-face deformations independently at each time step. These are attended to in fixed windows by attention blocks (A) and encoded as previous (E_{W-1}) and current window (E_W) motion features. These features along with bodyshape signature β are input to our temporal residue module f_R (middle, bottom) which predicts the Residual Jacobians to be added to per-frame Jacobians, to make them temporally coherent. These residuals are then integrated in time, via numerical Euler stepping to predict the stitched Jacobians at time t, followed by a spatial integration of these Jacobians via a Poisson Solve.

These two learnable modules - f_P and f_R - are trained simultaneously with only direct object-level supervision using a combination of positional and Jacobian losses. We use the ground-truth meshes from AMASS to supervise the predictions. The time t input is positionally encoded.

to integrate across time.

4.3.3 Motion Transfer with Space-time Integration

Without keyframes, we have to solve a temporal extrapolation problem. Our goal is to produce shape-preserving realistic motion sequences, as well as to learn such a motion model from a very sparse dataset. Not surprisingly, if each frame were independently predicted, the resulting motion would not be consistent across time, resulting in a jittery and artifact-ridden sequence (see Section 4.4). We, therefore, propose a novel framework wherein independent frame predictions are improved by providing temporal signals from a neural ODE. The two networks work in lockstep, each boosting the other's predictions, and are trained together.

Independent posing: We estimate each frame's Jacobians independent of temporal information. Given a shape X_0, pose configuration (relative joint angles with respect to SMPL's kinematic tree) M_t and features describing each mesh face, we predict J_t^P as the primary Jacobians at time t. Specifically,

$$\Delta J_t^P = f_P(J_0^P, M_t, C; \theta_P) \tag{4.4}$$

$$J_t^P = J_0^P + \Delta J_t^P \tag{4.5}$$

where J_0^P is the first frame Jacobian computed from X_0, M_t is the pose configuration given as relative joint angles, and C are per-face features defined at the centroid of the mesh faces, of the mesh at $t = 0$. In our tests, as features C, we jointly learned PointNet features on the centroids and normals of X_0 and augmented them with pre-computed Wave Kernel Signatures [99]. Our feature network (a shallow PointNet) learns geometric features that enable mapping to an unseen shape via correspondence in this learned feature space. We note that conditioning the posing network on J_0^P and adding the predicted delta via a residual connection significantly improves pose prediction and generalization to unseen pose configurations. We additionally note that the Jacobians are computed in the local basis defined at each face's centroid. Thus, J_0^P is the Jacobian of the identity deformation in the local basis (a rotation). Henceforth, we do not update the basis in the sequence and express all Jacobians on this chosen basis.

In practice, we analytically compute the triangle Jacobians of only the first frame X_0 in the sequence. We then predict the Jacobians at each time instance t in the coordinate frame of X_0 and solve Equation 4.2 to obtain the shape X_t at time t. Importantly, we augment NJF with temporal learning signals, by linking these independent per-frame predictions via a neural ODE, as described next.

Stitching across time: Our key observation is that learning *local* changes in time generalizes better to unseen shapes. Central to our method is a neural ODE that provides temporal training signals to the primary NJF and integrates across time to learn a smooth, arbitrary-length motion sequence. We found independent per-frame

predictions are prone to artifacts and do not generalize well (see Section 4.4). We also observe that a neural ODE, in isolation, cannot predict the entire sequence due to the drift problem inherent to estimating functions using numerical ODE methods. Specifically, given an initial state and per-frame control parameters (in our case, joint angles), predicting the mesh sequence is an extrapolation problem. As such, ODEs are prone to drifting away from the underlying function. This problem is exacerbated as the length of the motion increases – the longer the motion, the larger the accumulated drift.

As a solution, we propose a novel formulation to address both the incoherence of per-frame predictions and the drift problem. Specifically, we direct the neural ODE to learn only *Residual Jacobians* at each time step conditioned on the predictions from Eq 4.5 and on a window of past Jacobians. The Residual Jacobians are corrective factors which are directly added to the per-frame Jacobians. We predict Residual Jacobians local in time as outputs of a Neural ODE to ensure temporal coherence. This allows us to handle much longer motions, spanning 1-3k frames, without noticeable drift. We describe our method below.

ODE formulation: To handle arbitrary length sequences, in the interest of memory and training speed, we train and infer a given sequence in windows of consecutive frames. A given sequence is broken into fixed window sizes. We initialize J_0 in Eq 4.3 as the first frame's Residual Jacobian. Specifically, J_0 is the identity transformation projected on the local basis of each face of X_0 by rotation. Since the first frame is stationary and given as input, we set the first frame's Residual Jacobian as,

$$J_0^R = \mathbf{0} \in \mathbb{R}^{3\times 3}. \tag{4.6}$$

We then task the neural ODE to learn Residual Jacobian J_t^R at each t, by extrapolating from J_0^R using Euler's integration. We then extract the final Jacobians in terms of the base Jacobians corrected by the Residual Jacobians as

$$J_t = J_t^P + J_t^R. \tag{4.7}$$

We predict the residuals J_t^R by integrating over time the bodyshape-specific local changes predicted by f_R which is defined as,

$$\frac{\partial J_t^R}{\partial t} = f_R(J_0^P, E_W^P, E_{W-1}^R, \beta, t; \theta_R) \tag{4.8}$$

where β is the shape signature that defines the given body shape, and J_0^P, as defined previously, are the Jacobians of the first frame; $E_W^P$and $E_{W-1}^R$are attention encodings of current-window pose predictions and previous-window residual predictions. We integrate the local changes (see Eq 4.8) over time using Euler's method to obtain J_t^R at each t as,

$$J_t^R = \int_0^t \frac{\partial J_t^R}{\partial t} dt + J_0^R = \int_0^t f_R(J_0^P, E_W^P, E_{W-1}^R, \beta, t; \theta_R) dt. \tag{4.9}$$

Our jointly trained attention encoders are defined as,

$$E_W^P = A^P(J_W^P, T_W) \tag{4.10}$$

$$E_{W-1}^R = A^R(J_{W-1}^R, T_{W-1}) \tag{4.11}$$

where A^P and A^R are multi-head attention networks, J_W^P and J_{W-1}^R are a block of sequential Jacobians in the current window W and past window $W-1$, respectively; T_W and T_{W-1} are correspondingly blocks of time instances in these windows and are positionally encoded using time. In all our experiments we use a window size of 32 frames.

Note that we use the attention networks to encode a window of Jacobians to a single encoding. Since the encoding sizes are thus constant, we can handle arbitrary window/sequence lengths without overflowing memory. These encoders distill the current posed Jacobian predictions from Eq 4.5 and previously predicted Residual Jacobians obtained from Eq 4.9.

We pass the output of the attention networks as conditioning to Eq 4.8 to integrate and obtain the residuals in Eq 4.9. The predicted residuals are then added to the posed Jacobians in Eq 4.7. Finally, we spatially integrate the predicted Jaco-

bians J_t using a differentiable Poisson solve [152], in the coordinate frame of the first frame, to obtain the predicted shape X_t at t.

Loss function: Our pipeline is trained end-to-end using only a shape loss over vertices of X_t and a Jacobian loss. Thus, our final objective function is simply

$$\mathscr{L}_{\text{vertex}} = \|X_t - X_t^{GT}\|^2 \quad \text{and} \quad \mathscr{L}_{\text{Jacobian}} = \|J_t - J_t^{GT}\|^2, \tag{4.12}$$

$$\mathscr{L} = \mathscr{L}_{\text{vertex}} + \alpha \mathscr{L}_{\text{Jacobian}}. \tag{4.13}$$

We use $\alpha = 0.05$ in our tests.

4.4 Evaluation

Motion datasets. We train and evaluate our method on motion sequences from three different datasets. First, the AMASS dataset [122] for humanoid motions, which is based on the SMPL body model [4] that allows us to vary body-shapes by changing its shape parameter β, for a given sequence. Thus, we train our model for a given motion category (like running) to predict the sequences on varying body-shapes (indicated as the β-predictor in Figure 4.2). During training, each sequence in the category is trained with only one body-shape. During inference, given an unknown test shape X_0 and a motion $\{M_t\}$, we synthesize the full body motion for the target shape. Second, we also tested on a synthetic DeformingThings4D dataset [1], which provides animal 4D meshes as deforming sequences. Finally, we also present results on motions extracted from video recordings of animals in the COP3D dataset [2] where our inputs are meshes fitted to the video recordings to train Temporal Residual Jacobians.

Target shapes. We evaluate our method on various forms of target shapes, namely sampled SMPL models, scans from the FAUST dataset, characters from the Mixamo library, various models from online 3D repositories (e.g., wolf, triceratop, monster, hole-man, etc.). Non-manifold inputs were converted to manifold meshes before running our algorithm. All correspondences were as inferred by the signature

Figure 4.3: Generalization across bodyshapes. We show results of different motion transfers on meshes found in-the-wild (blue), FAUST scans (pink) and Mixamo characters (green). We observe a smooth motion consistent with the target geometry in each case. Please see the webpage for the videos.

module (i.e., combination of PointNet features on centroids and normals of faces along with WaveKernel Signatures, previously defined as C).

Implementation details. All the networks we use are shallow MLPs to aid training speed. For our independent posing network, we use a 4-layer MLP with ReLU activation, with the final layer being linear. For the residual Jacobian prediction, we use a 3-layer MLP similarly with intermediate ReLU activations and a final linear layer. Both our attention networks follow the same architecture – we use two atten-

tion heads, with a 32-neuron wide feed forward layer and 32-dimensional features for the keys and values. Our PointNet network from which we obtain per-face features similarly has 3 ReLU layers followed by a linear layer at the end. Our method and the baselines are trained until $\mathscr{L}_{\text{vertex}}$ converges to less than 3e-4 or upto 300 iterations. On a 12 GB TitanX our typical training time is 6-8 hours, varying by the lengths of the sequences.

For the `walking` and `dancing` motions, we set the root orientation in AMASS to zero, so all subjects are front facing, making it easier to train (similar to [148]), as our Jacobian-based formulation cannot, on its own, infer global rotation and/or translation. We do not, however, predict global transformations; we obtain the global transforms from the source AMASS sequences and apply them to our final outputs at inference after appropriate scaling according to the target's height.

Qualitative results. We show various examples of deformation transfer by our method in Figures 4.1, 4.3, 4.4 and 4.5. Please refer to the videos in the web-

Figure 4.4: Generalization across shapes with very sparse training sets. Here, we show motion transfer from two animal sequences (in yellow) sampled from the DeformingThings4D dataset [1], to animal meshes found in the wild (in blue). Our method was trained on only two sequences from this dataset and yet generates plausible motion transfer to unseen shapes. Rigs were *not* available to our algorithm at training and/or test time. (Note: blue sequences have been slightly globally rotated for visibility.)

page. Our method generalizes to new shapes of varied body types, including non-humans *completely unseen during training*, while preserving the source motion. We show examples on monsters, a 4-armed character, etc. The correspondences in these cases are implicitly established using our learned per-face features and Wave Kernel Signatures, and the method works well because the characters are all four-limbed. Motion transfer to characters with different limb topologies—such as an eight-limbed spider or octopus—may yield unpredictable results. Additionally, as different humans perform the same motion in different manners, we capture those varying dynamics in the deformation of the new shape as well. We can also learn and apply motions from animals, both from synthetically generated mesh sequences as well as motions extracted from monocular video recordings.

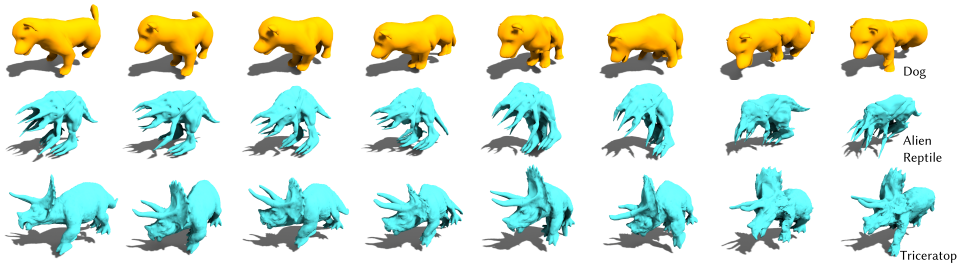

Figure 4.5: Motion transfer from COP3D dataset. We train on only four sequences of dogs obtained from the COP3D dataset [2], which are monocular video recordings of animal motion, and transfer the observed regressed motion (in yellow) to creature meshes found in the wild (in blue).

Comparisons. Our method Temporal Residual Jacobians, due to the space-time coupled formulation, is designed to produce natural-looking motion retargeting. Although we are unaware of any other method that performs the specific motion extrapolation problem (i.e., *without* access to keyframes or skeletal rigs), we compare with possible baseline design variations. First, *VertexODE*, where an MLP predicts the rate of change of vertex positions (velocity) at time t, followed by a Neural ODE [8] that takes numerical steps by Euler's method to integrate to the vertex positions at t. This baseline directly displaces the vertices (i.e., without triangle Jacobians). As seen in Figure 4.6, this method produces significant artifacts and leads to degenerate shapes. It also loses the intended motion after some time, as seen in the videos. Second, *NJF(M_t)*, where we extend the original NJF frame-

work to be additionally conditioned on per-frame relative joint orientations M_t, to predict the Jacobians at each time step. Framewise prediction leads this approach to suffer from jitters (please refer to the videos on the webpage) and a tendency to overfit to training data. Although the individual frames are mostly plausible, they occasionally suffer from frame-level artifacts as shown in Figure 4.6. The primary limitation, however, is jittery output as seen in the video results since the frames are independently generated. Note that a method without spatial or temporal derivatives wherein a simple MLP is trained to predict vertex deformations given the initial geometry and time t performs poorly, with numerous artefacts due to vertex-level predictions.

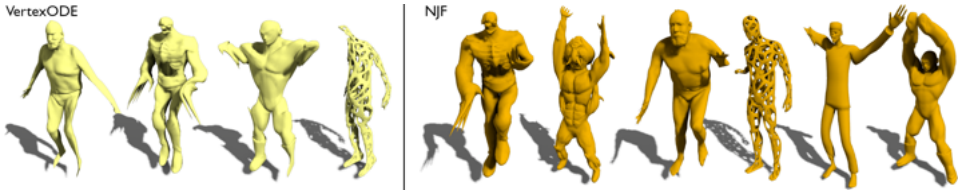

Figure 4.6: Observed artifacts in baselines. For each motion, we show results from an intermediate frame of the motion transfer for our baselines. VertexODE (left, yellow) completely distorts the shape, while not following the target motion. NJF (right, brown) suffers from temporal discontinuity resulting in motion-driven geometric artifacts – extended or shrunken parts as it tends to a linear path in later time steps – and jitter.

Quantitative comparison: We evaluate our method on several types of sequences and compare ours against other baselines in Table 4.1 by computing the residual error with respect to the ground-truth sequences. We compare five different motion categories from the AMASS dataset - `running`, `jumping jacks`, `walking`, `hopping`, and `punching`. We achieve the best on all metrics.

Human motion to non-humanoid characters. We show results on human motion applied to non-humanoid characters (e.g., 4-armed monster, alien-reptile) in the webpage videos. Please note the quality of automatic transfer without manual landmark specification.

Handling long sequences. One advantage of our space-time coupled formulation is the ability to handle long motion sequences. We show examples of motion transfer from a few hundred to a couple of thousand frames (dance and walk) on the

Table 4.1: Quantitative evaluation. Average vertex-to-vertex error in cm, L2 error of predicted Jacobians and angular error of normals in degrees, measured against ground truth sequences, for different motion categories and averaged over multiple sequences within the same target motion category. Here we compare against neural ODE [8] and an extended version of NJF [9]. Lower values indicate better generalization.

Method	Jump			Run			Punch			Walk			Dance		
	L2-V	L2-J	L2-N	L2-V	L2-J	L2-N	L2-V	L2-J	L2-N	L2-V	L2-J	L2-N	L2-V	L2-J	L2-N
VertexODE	22.59	1.22	48.21	14.23	0.83	42.11	17.66	0.65	40.04	23.92	1.01	46.12	26.15	1.26	50.17
NJF(M_t)	5.52	0.41	9.66	3.61	0.32	7.38	4.95	0.38	7.42	6.85	0.34	8.12	4.86	0.44	11.24
TRJ (Ours)	**2.64**	**0.28**	**7.31**	**1.73**	**0.24**	**5.63**	**2.86**	**0.26**	**6.65**	**1.48**	**0.22**	**6.33**	**3.96**	**0.37**	**9.88**

webpage.

Error Accumulation. While our method significantly reduces drift from the true motion, some error still accumulates as the poses deviate more from the initial pose. This is primarily due to the challenge of accurately predicting the initial Jacobians using f_P when the pose difference is large—for example, between a standing and a bending human. Our temporal residue module f_R is trained to predict small, local corrections that help stitch together frame-wise predictions. However, it is not designed to handle large, global pose transitions, and therefore cannot predict arbitrarily large residues. As a result, error accumulates gradually over time. One possible solution is to introduce intermediate keyframes as additional inputs to the initial pose, thus replacing "stale" Jacobians with "fresh" ones closer to the pose being predicted. We do not explore this direction in the current work, as our method already yields satisfactory results on long motion sequences such as `dance` and `walk`.

Design variation. A seemingly possible variation of ours is to choose a residual Jacobian representation, but expressed in terms of the previous predicted frame. While this seems attractive, we found it very slow to train as we have to backpropagate through time, and convergence is very slow (when using moderate computational resources). Hence, we found this approach to be infeasible with the current memory requirements for attention (and transformer) modules.

Alternatives to Euler Solve. We experimented with higher-order ODE solvers, namely Runge-Kutta and Runge-Kutta-Fehlberg. However, we noticed no significant improvements, even with the noticeably increased training time. We use the

simpler Euler's method for temporal integration for our tests.

Metrics & Perceptual Evaluation. The proposed metrics evaluate the accuracy of predicted motion based on point trajectories (L2-V), joint transformations (L2-J), and mesh consistency (L2-N). An additional useful metric is L2-Delta-V, which measures pointwise velocity and captures temporal smoothness. This is particularly relevant because independent per-frame predictions can introduce jitter, and velocity-based or higher-order metrics can help reveal such artifacts. Our work could also benefit from a perceptual user study to assess the plausibility of predicted motions. The same sequence of poses can be executed differently depending on body shape, making motion inherently shape-dependent. Since this variation is subjective, a user study focused on motion plausibility would provide valuable insight into the perceptual quality of the results.

Importance of Initial Shape. In our method, we assume the initial pose to be a natural rest position—typically close to an A-pose. This assumption aligns with common practice, as such poses are widely used in artist-designed character meshes and raw 3D human scans. However, since our model predicts local residuals in both space and time, it can still generate motion even from different initial poses. That said, the plausibility of the resulting motion becomes less predictable. For example, the model may learn the motion of `punching` as lifting the arms from a rest pose in preparation for a punch. If the initial pose instead has both arms extended straight overhead, the predicted motion may no longer resemble `punching`, and it becomes unclear how the model would proceed to lift the arms further or transition plausibly. This makes the quality of motion less reliable when starting from atypical poses.

4.5 Conclusion

We have presented Temporal Residual Jacobians, a spatially-coupled, neural ODE-based, motion transfer framework conditioned on body type and target motion to produce local Jacobians that are subsequently integrated across space and time to deform the target geometry. The resultant motions are robust, realistic, and gener-

alize to different body types. We extensively tested our method on both synthetic and real data captures, and enabled generic motion transfers to an extent which is not possible using existing methods.

Our method has limitations. (i) We do not impose physics constraints, therefore, our animation can have self-intersections (see result videos on the webpage). An interesting direction is to incorporate constraints for collision detection, e.g., via subspace-based contact handling [153]. We want to incorporate such an approach directly into the method, possibly via an attention mechanism, as motion dynamics are affected by earlier collisions. (ii) Although our Temporal Residual Jacobians, along with windowed attention modules, keep drifts low, error still accumulates over long motion sequences. A possible solution is to couple our method with a keyframe-based workflow [154]. (iii) Finally, our current formulation implicitly establishes correspondences and uses them to infer temporal Jacobians for surface triangles. Any spurious correspondences can be overridden by artists, possibly directly by "paintbrushing" correspondences or materials (e.g., indicating that the ear of the bunny model should be floppy) or using semantic features learned from untextured meshes [155, 156].

Chapter 5

Discussion

This thesis demonstrates that 3D generative models can indeed be effectively trained on sparse datasets, achieving competitive generalization by leveraging known or learned priors to impose additional constraints. We introduced three novel frameworks: GLASS for pose generation, BLiSS for learning morphable models for body shape generation, and Temporal Residual Jacobians (TRJ) for rig-free motion transfer. Together, these contributions address key challenges in 3D modeling and animation, particularly in data-scarce scenarios.

GLASS introduced a deformation-aware latent space that enables the generation of diverse poses from minimal examples by embedding geometric priors. BLiSS extended this approach by learning an artist-registration prior from a small set of registered shapes and iteratively augmenting the shape space with registrations from a large collection of raw scans. Finally, TRJ tackled dynamic motion generation by embedding motion-consistency priors through a Neural ODE that predicts corrective temporal residuals.

Despite their strengths, the proposed methods have inherent limitations. GLASS requires input meshes with consistent triangulation, cannot accommodate significant shape variation—thereby necessitating datasets with consistent identity and pose, which are uncommon for many shape categories—and is computationally intensive to scale from small to medium-sized datasets. Recent advances offer promising directions to address these challenges. For example, state-of-the-art correspondence frameworks such as [156] can be used to re-mesh shapes into

a common topology, mitigating the requirement for consistent triangulation. Another promising avenue involves defining data-driven energy functions—for instance, capturing a notion of "chairness" from a collection of chairs and using this as an energy prior to guide the generation of novel, chair-like shapes. While such approaches are theoretically compatible with GLASS, they introduce new requirements: shape deformation across varying topologies and energy functionals that remain differentiable in the latent space.

A key computational bottleneck in GLASS lies in the calculation of the analytical Hessian of the mesh-based energy function in latent space. This is currently done by backpropagating through a dense, mesh-resolution-dependent multi-layer perceptron (MLP). Higher-resolution meshes necessitate wider MLPs, and the size of the Hessian grows quadratically with the number of vertices. To alleviate this bottleneck, future work could explore data-driven, classification-based energy functions, topology-agnostic deformation methods, and numerical approximations of latent-space Hessians—each of which could help decouple performance from mesh resolution and improve scalability.

BLiSS relies on accurate initial correspondences and registrations—typically provided by artists—to initialize a shape space. While it can effectively augment a sparse shape space into a medium-sized one, achieving downstream performance comparable to established models such as SMPL [4] and GHUM [5], its primary limitation lies in its dependence on a borrowed pose space supplemented with pose correctives. One potential solution is to jointly learn both the pose and shape spaces directly from sparse data.

BLiSS is designed to learn from a limited set of identity shapes in a canonical pose, whereas joint learning of pose and shape spaces requires datasets in which each body shape is observed in multiple poses. The availability of such data would enable the integration of GLASS (for pose exploration) with BLiSS (for shape exploration), allowing the construction of a complete space from just a few examples. However, datasets that provide even a sparse matrix of identity-pose combinations are rare. Addressing this data scarcity is a central motivation behind the contribu-

tions of this thesis.

Temporal Residual Jacobians (TRJ) suffers from error accumulation over long motion sequences, as predicted poses progressively drift from the starting pose. A potential remedy is the intermittent renewal of Jacobians by introducing additional keyframes as checkpoints, allowing TRJ to handle arbitrarily long sequences. However, this approach increases the burden on animation artists, who must manually provide these keyframes.

Currently, TRJ learns a motion field specific to each motion category, limiting its generalization to unseen motions. For example, transferring a novel motion such as a yoga sequence would likely yield unpredictable or inaccurate results. This limitation stems from the use of SMPL pose parameters as the direct input motion representation, which lacks structure for interpolation or generalization across diverse motions.

A promising direction to address this is to learn a motion embedding, similar to approaches like [148], that provides a continuous, interpolatable, and parameterized motion space. Such a representation could serve as the input to TRJ, enabling it to generalize to a broader range of motions, including those unseen during training. This would significantly extend TRJ's applicability—allowing it to generate plausible deformations for novel motions on previously unseen characters.

TRJ relies on manifold meshes, and while theoretically adaptable to disconnected meshes, this remains untested. Additionally, the ARAP energy used in GLASS limits scalability to higher-resolution meshes, leading to geometric artifacts. Generating production-ready assets may ultimately require leveraging larger datasets, such as Objaverse, to overcome these barriers.

Moreover, for all three works presented in this thesis, the reliance on mesh representations presents another limitation, as extending these methods to alternative 3D representations like point clouds, voxels, radiance fields, or Gaussian splats remains non-trivial but holds exciting potential. A more fundamental constraint arises from the sparse dataset paradigm itself. Each method proposed in this thesis reaches a saturation point, beyond which generating novel, high-quality out-

puts becomes challenging. For instance, GLASS plateaus after generating approximately 2,500 novel poses, and BLiSS similarly encounters a threshold of 1,000 body shapes. In both GLASS and BLiSS, saturation is detected using metrics that signal reduced diversity in discovered shapes. Specifically, GLASS employs the Maximal Marginal Relevance (MMR) metric as a filter to determine whether to include or discard newly generated shapes. Saturation is identified when this filter admits only shapes with high similarity—measured as a cosine similarity greater than 0.8—to those already in the set. In BLiSS, we use a one-standard-deviation threshold from the minimum reconstruction error to filter shapes, and declare saturation when fewer than 10 shapes fall within this error range. We observe that saturation is reached more quickly when the initial set of shapes lacks diversity. Conversely, a more diverse starting set enables the discovery of substantially larger shape spaces, as both GLASS and BLiSS rely on small deformation steps from the initial examples to explore new regions of the space.

Addressing the discussed challenges through advancements such as physics-based constraints, artist-driven refinements, or hybrid approaches integrating learned and procedural techniques remains a promising direction for future research.

While developed within the context of sparse datasets, the core principle of integrating prior constraints can also be applied to data-rich training paradigms. Embedding domain knowledge in the form of constraints into models trained on extensive datasets could further reduce artifacts and enhance 3D generation quality, offering an exciting avenue for future exploration.

In conclusion, this thesis highlights the critical role of task-specific priors and innovative algorithmic designs in advancing 3D generative modeling. By demonstrating the power of prior-driven approaches, it paves the way for building more efficient, versatile, and high-quality 3D generation systems—an essential step toward unlocking the full creative potential of 3D content creation.

Appendix A

GLASS: Geometric Latent Augmentation for Shape Spaces

A.1 Additional Generated Samples

Figures A.1, A.2, and A.3 show some example generated deformations starting from sparse sets of Faust, Centaur, and Horse models, respectively.

A.2 Expression generation using COMA dataset

Despite facial expressions not being perfectly locally rigid, we show below that GLASS generates plausible novel expressions on the COMA dataset in figures A.4 and A.5

Table A.1: 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 [30]	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**

A.3 Additional Interpolation Examples

Figures A.6, A.7 and A.8 show example interpolated shapes between end poses (shown in gray), using the discovered latent space revealed by GLASS.

Figure A.1: 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.

A.4 Network Architecture

Figure A.9 shows the VAE architecture used by GLASS. The main pseudocode for GLASS is provided in Chapter 2.

A.5 Dataset Images

We include images of the various input datasets we used to highlight the diversity of pose variations in the input. Note that the datasets are all very sparse, consisting

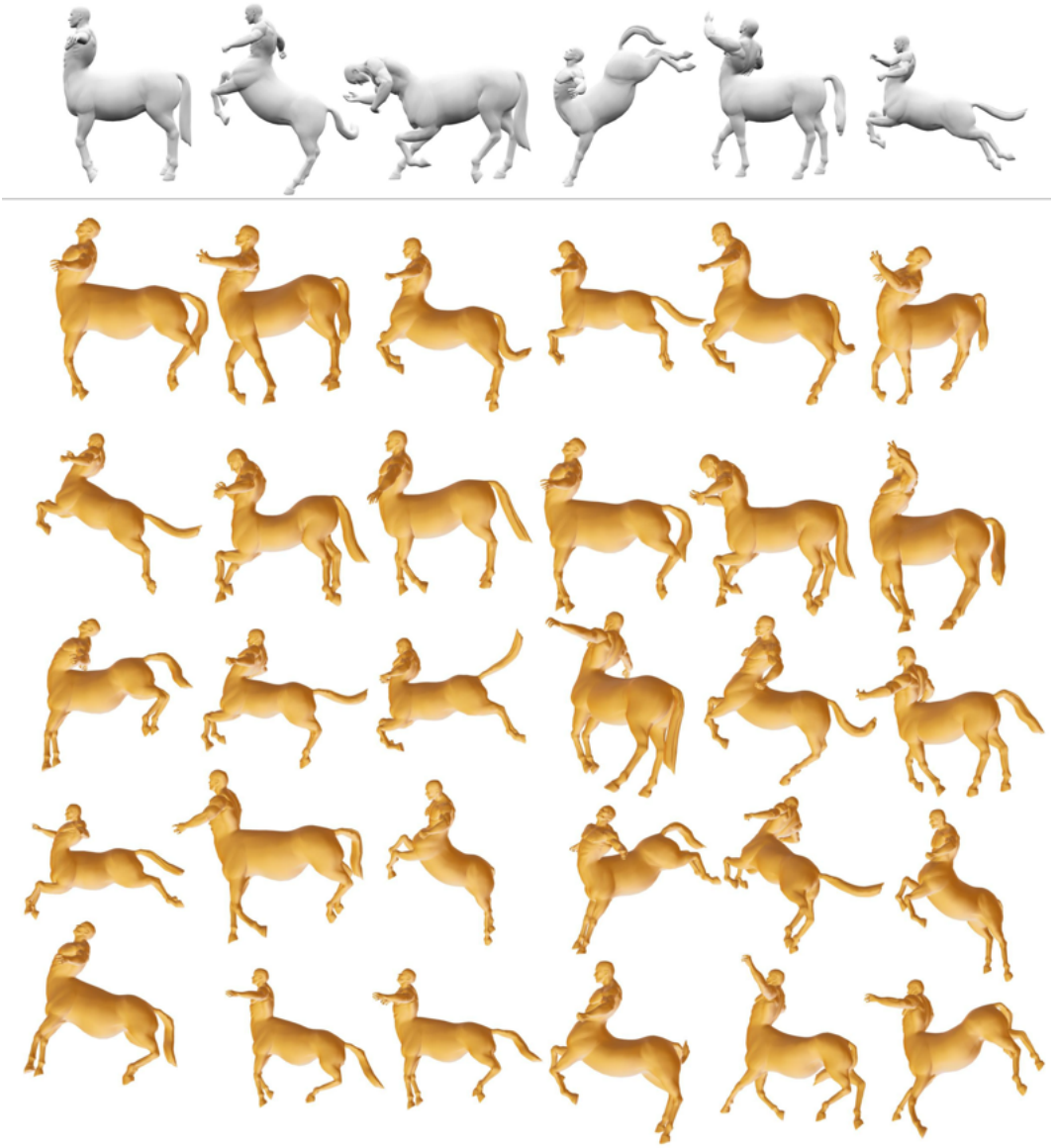

Figure A.2: We show some of the samples generated by GLASS (gold), from only 6 Centaur poses (gray). We see novel poses like bent back legs and torso facing upwards.

of 3-10 models.

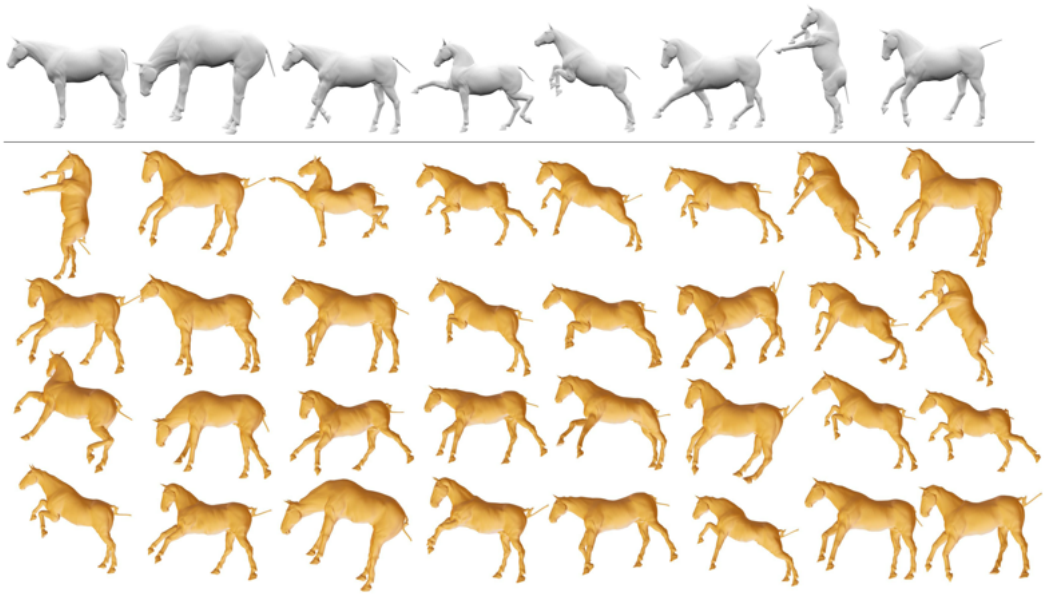

Figure A.3: We show some of the samples generated by GLASS (gold), from only 8 Horse poses (gray). We see novel poses like front legs raised beyond what's seen in the training set, upright torso and back legs stretching farther.

Figure A.4: We show facial expressions generated (gold) by training GLASS on 3 expressions from the COMA dataset (gray)

Figure A.5: We show facial expressions generated (gold) by training GLASS on 6 expressions from the COMA dataset (gray)

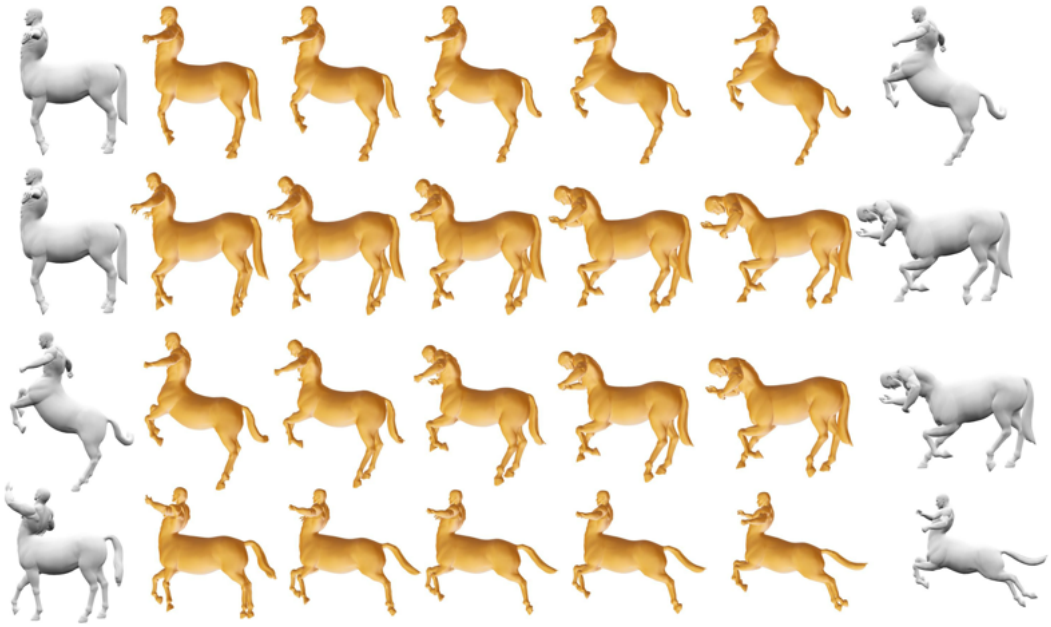

Figure A.6: Interpolated shapes (gold) between 2 Centaurs Poses (gray) inside the latent space generated using GLASS.

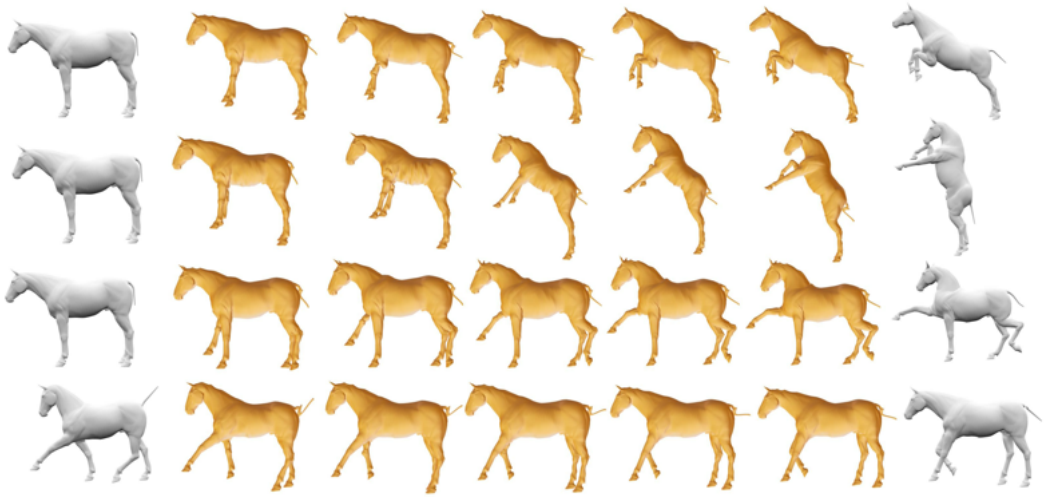

Figure A.7: Interpolated shapes (gold) between 2 Horse Poses (gray) inside the latent space generated using GLASS.

Figure A.8: Interpolated shapes (gold) between 2 Faust Poses (gray) inside the latent space generated using GLASS.

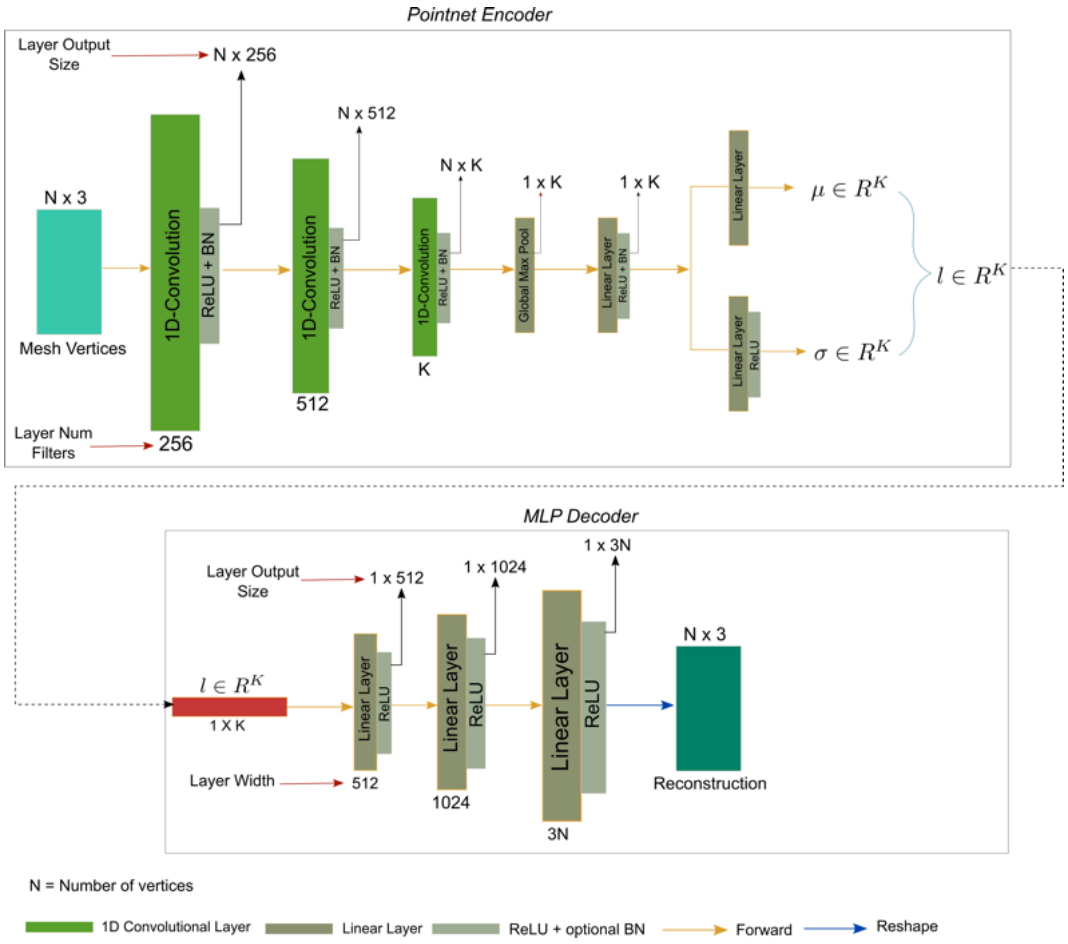

Figure A.9: The VAE architecture used by GLASS.

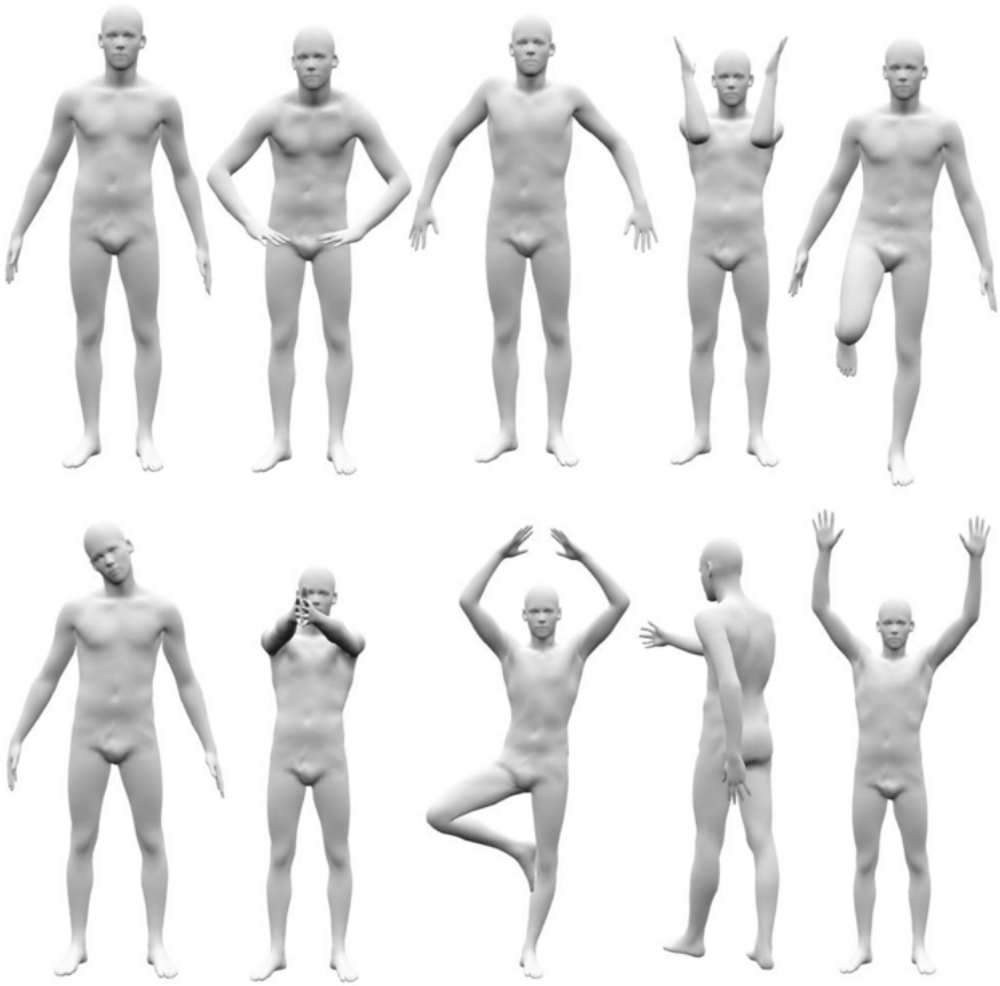

Figure A.10: The Faust-10 dataset with 10 poses. Please refer to Figure A.11 for the corresponding even sparser versions of the dataset used in our experiments.

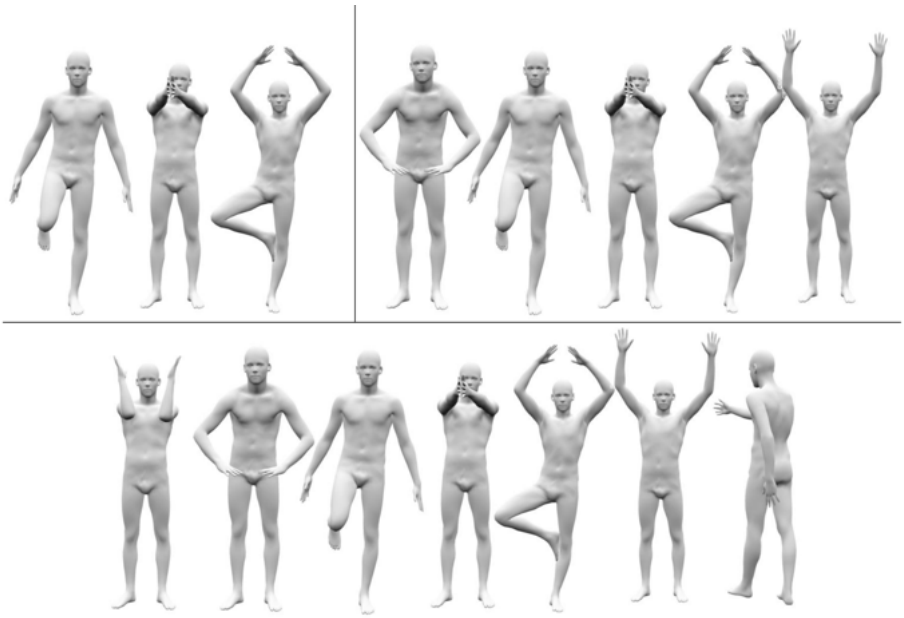

Figure A.11: Faust-3 (top left), Faust-5 (top right) and Faust-7 (bottom).

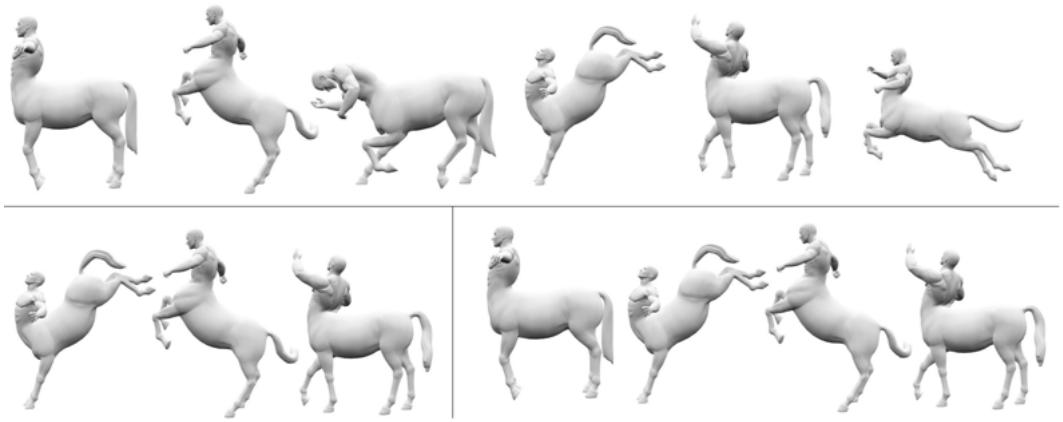

Figure A.12: Centaurs-6 (top), Centaurs-3 (bottom left), Centaurs-4 (bottom right).

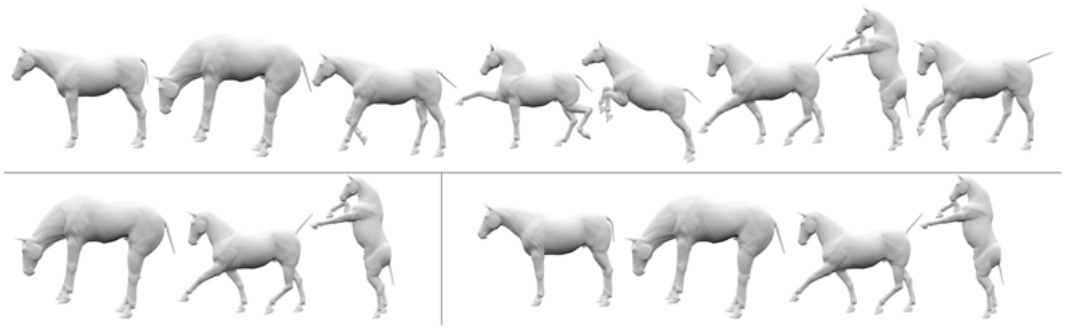

Figure A.13: Horses-8 (top), Horses-3 (bottom left), Horses-4 (bottom right).

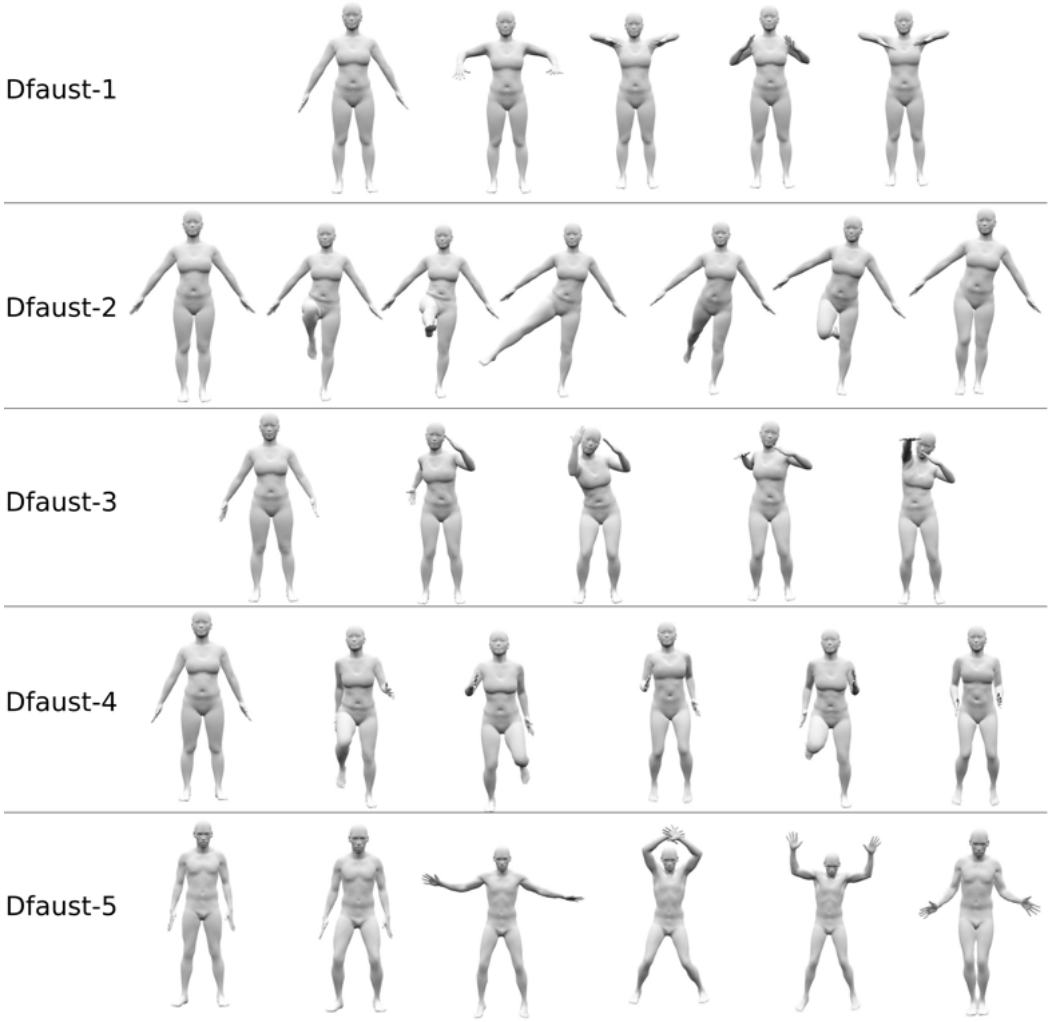

Figure A.14: The keypoints of 5 different Dynamic Faust sequences used for training, for Table 2.2 in Chapter 2

Appendix B

BLiSS: Bootstrapped Linear Shape Space

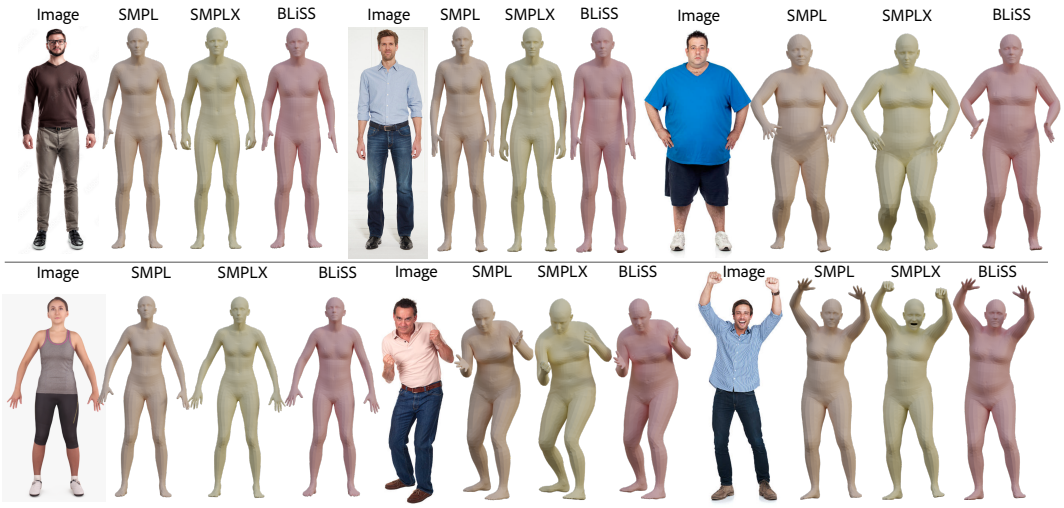

Figure B.1: Here we use the trained SMPLify-X [3] model to estimate the shape from a single image. For BLISS, we plugin our shape space as a drop-in replacement for SMPL's space, while using SMPL's pose space.

B.1 Shape Estimation from Single Image

We additionally compare our shape space with SMPL's by plugging them into an optimization-based body reconstruction framework SMPLify-X [3]. Since BLISS doesn't handle pose, we use SMPL's pose space. SMPLify-X then searches the given space to fit the shape to the given image. We observe in Figure B.1, that BLISS's estimated shape is comparable with that of SMPL and SMPL-X, despite having started from only 200 given registrations. Note that unlike SMPL-X, SMPL

(and hence BLISS) does not articulate hand poses.

B.2 Iterative improvements

Registrations in the initial rounds of our shape-space are often coarse, lacking details, as shown in the two pale colored faces in the middle in Figure 2 B.2. As the space is more densely populated, later registration more closesly capture finer details of the given scan.

Figure B.2: Iterative Shape Corrections: Lightly colored faces in the middle are our registrations in earlier iterations of the space, and the pink-colored face on the right is our registration after five rounds of BLISS. As the rounds progress, registrations in later rounds more accurately capture the scan (left, in Blue), as observed by the broadening of the nose and jawline.

B.3 Hand Registration - MANO

We sample 20 hands from MANO to create an initial shape space. We then iteratively register scans from the MANO dataset using the pre-trained NJF *without* assuming paired data. Since NJF is a topologically invariant mapping module, we are able to train the network for full body registrations, and use the trained MLP to register out-of-domain scans. Registrations are shown in Figure B.3.

Figure B.3: Registering Hand scans: We use our final hand-shape space to register hand scans (blue); registrations in canonical pose (i.e., default) shown in pink.

B.4 Implementation Details

B.4.1 Features for Neural Jacobian Field

At the core of NJF is a Multi-layer Perceptron (MLP) that processes the input features on each triangle of the given mesh to produce a per-triangle Jacobian, which is used in a differentiable Poisson solve to compute the deformed vertex positions. We use NJF to deform our PCA projection X_o conditioned on the scan S_X, similarly to the shape morphing experiments proposed in the original paper [9]. In our tests, we used the following features:

- Since S_X is a raw scan, the PointNet encoding of its coordinates represents it. We use both the global encoding of the scan and its per-point features from PointNet. Since the scan and X_o are *not* in correspondence, we choose features of those points that are closest to a point on X_o. We note that despite X_o and S_X having different poses, the nearest neighbor feature look-up provides an indication to the MLP of the kind of shape transformation that is required.

- We associate with each triangle of X_o, the PointNet encoding of S_X and Ss_X's points as obtained above, and of X_o itself. Specifically for X_o as in the original work, we encode each triangle's centroids, normals and top 50 Wave Kernel Signatures [99]. In addition, we also pass the Jacobians of X_o itself as we observe that it significantly improves the mapping to target. Note that S_X and X_o are processed via different PointNets as their input features are different

We pass the above features to a 4-layer MLP, with each hidden layer being 128 wide and activated by ReLU. The final Linear layer produces a 9-dimensional vector for each triangle (as a Jacobian is a 3×3 matrix). Both PointNets - one for X_o and one for S_X - and the MLP are trained jointly to produce the mapping from X_o to the desired shape.

B.4.2 Summary of the Use of CAESAR data

We have registration information for 429 of all 4000 CAESAR scans. We use 100 of them for training the initial PCA shape space and another 100 for NJF, withholding

the rest 229 for evaluation purposes, *i.e.*, $|R_{PCA}| = |R_{DEFORM}| = 100$, $|R_{EVAL}^{\text{FULL}}| = 229$ or $|R_{EVAL}^{\text{SMALL}}| = 29$. In each round of Algorithm 1, we sample random 100 among the rest of $\sim$ 3.5k (=4000-429) unregistered shapes U, bring it into correspondence and move it to R_{PCA} when selected by our pruning.

B.5 Nearest Lookup from Others to GHUM

In Table 3.4 of Chapter 3, we report a v2p error of 2.63 cms on 229 scans, while GHUM [5] report 1.91 cms (Chamfer distance) on the entire CAESAR [87] dataset. This difference may arise from the difference in the evaluation set and the different training sets. Further, we observe in Table 3.5 of Chapter 3, that the numbers reported with respect to GHUM are higher compared to others. In Figure B.4, we show samples from each of SMPL, STAR and BLISS and their corresponding nearest sample by vertex-to-vertex L2 in GHUM's shape space. Each space was randomly sampled to generate 10000 shapes, and we perform pairwise L2 distance across shapes in different spaces. We observe significant differences in body-shape indicating that there are non-overlapping regions between GHUM and other shape spaces.

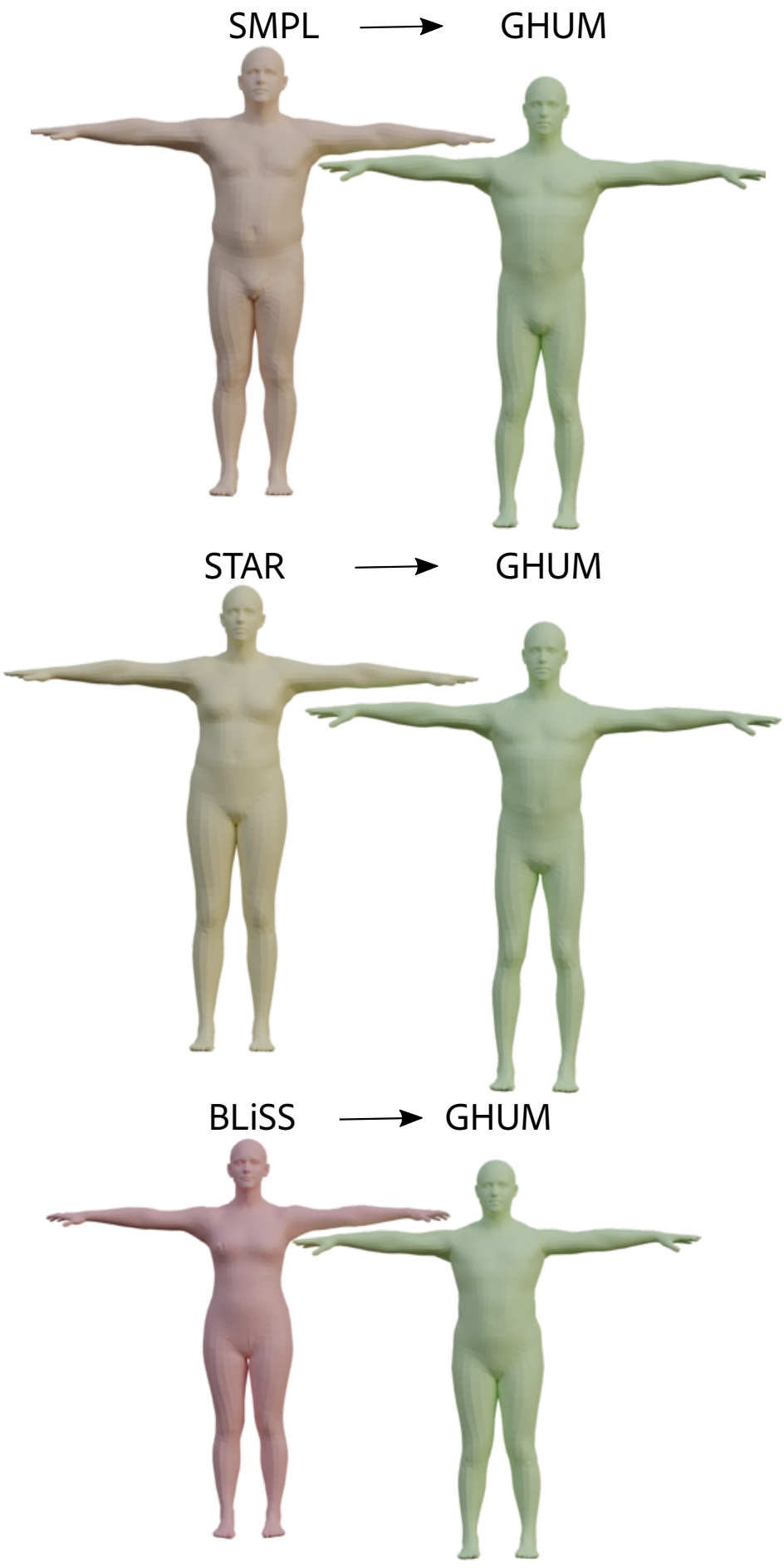

Figure B.4: For a sample in each of SMPL, STAR and BLiSS, we lookup the nearest shape in GHUM's space (green). We randomly sampled 10000 shapes in each space.

Bibliography

[1] Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. 2021.

[2] Samarth Sinha, Roman Shapovalov, Jeremy Reizenstein, Ignacio Rocco, Natalia Neverova, Andrea Vedaldi, and David Novotny. Common pets in 3d: Dynamic new-view synthesis of real-life deformable categories. *CVPR*, 2023.

[3] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *CVPR*, pages 10975–10985, 2019.

[4] 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.

[5] Hongyi Xu, Eduard Gabriel Bazavan, Andrei Zanfir, William T. Freeman, Rahul Sukthankar, and Cristian Sminchisescu. GHUM & GHUML: Generative 3D human shape and articulated pose models. In *CVPR*, pages 6183–6192, 2020.

[6] Yuanlu Xu, Song-Chun Zhu, and Tony Tung. DenseRaC: Joint 3D pose and shape estimation by dense render-and-compare. In *ICCV*, October 2019.

[7] Ahmed A A Osman, Timo Bolkart, and Michael J. Black. STAR: A sparse trained articulated human body regressor. In *ECCV*, pages 598–613, 2020.

[8] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[9] Noam Aigerman, Kunal Gupta, Vladimir G Kim, Siddhartha Chaudhuri, Jun Saito, and Thibault Groueix. Neural jacobian fields: Learning intrinsic mappings of arbitrary meshes. *SIGGRAPH*, 2022.

[10] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, page 409–416, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[11] Luke Olsen, Faramarz F. Samavati, Mario Costa Sousa, and Joaquim A. Jorge. Technical section: Sketch-based modeling: A survey. *Comput. Graph.*, 33(1):85–103, February 2009.

[12] Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R. Klemmer. Designing with interactive example galleries. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, page 2257–2266, New York, NY, USA, 2010. Association for Computing Machinery.

[13] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: a general approach to setting parameters for computer graphics and animation. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, page 389–400, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[14] Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and Vladlen Koltun. Probabilistic reasoning for assembly-based 3d modeling. *ACM Trans. Graph.*, 30(4), July 2011.

[15] R. Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. Shapeassembly: learning to generate programs for 3d shape structure synthesis. *ACM Trans. Graph.*, 39(6), November 2020.

[16] Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, and Thomas Funkhouser. Attribit: content creation with semantic attributes. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, page 193–202, New York, NY, USA, 2013. Association for Computing Machinery.

[17] Yassir Saquil, Qun-Ce Xu, Yong-Liang Yang, and Peter Hall. Rank3dgan: Semantic mesh generation using relative attributes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5586–5594, Apr. 2020.

[18] Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. Meshgpt: Generating triangle meshes with decoder-only transformers. *arXiv preprint arXiv:2311.15475*, 2023.

[19] Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. Polygen: An autoregressive generative model of 3d meshes. *ICML*, 2020.

[20] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. In *Advances In Neural Information Processing Systems*, 2022.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you

need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[22] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[23] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022.

[24] Fangzhou Hong, Jiaxiang Tang, Ziang Cao, Min Shi, Tong Wu, Zhaoxi Chen, Shuai Yang, Tengfei Wang, Liang Pan, Dahua Lin, and Ziwei Liu. 3dtopia: Large text-to-3d generation model with hybrid diffusion priors, 2024.

[25] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023.

[26] Yawar Siddiqui, Tom Monnier, Filippos Kokkinos, Mahendra Kariya, Yanir Kleiman, Emilien Garreau, Oran Gafni, Natalia Neverova, Andrea Vedaldi, Roman Shapovalov, and David Novotny. Meta 3d assetgen: Text-to-mesh generation with high-quality geometry, texture, and pbr materials. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 9532–9564. Curran Associates, Inc., 2024.

[27] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsanit, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13142–13153, 2023.

[28] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su,

Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository, 2015. cite arxiv:1512.03012.

[29] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

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

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

[32] Binh Huy Le and Zhigang Deng. Robust and accurate skeletal rigging from mesh sequences. *ACM Trans. Graph.*, 33(4), July 2014.

[33] Péter Borosán, Ming Jin, Doug DeCarlo, Yotam Gingold, and Andrew Nealen. Rigmesh: automatic rigging for part-based shape modeling and deformation. *ACM Trans. Graph.*, 31(6), November 2012.

[34] Andrew Feng, Dan Casas, and Ari Shapiro. Avatar reshaping and automatic rigging using a deformable model. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, MIG '15, page 57–64, New York, NY, USA, 2015. Association for Computing Machinery.

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

[36] 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.

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

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

[39] 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.

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

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

[42] Miroslav Purkrabek and Jiri Matas. Improving 2d human pose estimation in rare camera views with synthetic data. In *2024 IEEE 18th International Conference on Automatic Face and Gesture Recognition (FG)*. IEEE, May 2024.

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

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

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

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

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

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

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

[50] 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.

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

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

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

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

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

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

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

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

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

[60] 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.

[61] 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.

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

[63] 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.

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

[65] 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.

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

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

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

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

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

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

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

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

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

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

[76] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194, 1999.

[77] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM TOG*, 36(6):245:1–245:17, 2017.

[78] Pablo Garrido, Levi Valgaerts, Ole Rehmsen, Thorsten Thormahlen, Patrick Perez, and Christian Theobalt. Automatic face reenactment. In *CVPR*, pages 4217–4224, 2014.

[79] Kevin Dale, Kalyan Sunkavalli, Micah K. Johnson, Daniel Vlasic, Wojciech Matusik, and Hanspeter Pfister. Video face replacement. *ACM TOG*, 30(6):1–10, dec 2011.

[80] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Thomas J. Cashman, and Jamie Shotton. Fake it till you make it: Face analysis in the wild using synthetic data alone. In *ICCV*, pages 3681–3691, October 2021.

[81] Justus Thies, Michael Zollhöfer, Matthias Nießner, Levi Valgaerts, Marc Stamminger, and Christian Theobalt. Real-time expression transfer for facial reenactment. *ACM TOG*, 34(6):183–1, 2015.

[82] Liwen Hu, Shunsuke Saito, Lingyu Wei, Koki Nagano, Jaewoo Seo, Jens Fursund, Iman Sadeghi, Carrie Sun, Yen-Chun Chen, and Hao Li. Avatar digitization from a single image for real-time rendering. *ACM TOG*, 36(6):1–14, 2017.

[83] Sanjeev Muralikrishnan, Siddhartha Chaudhuri, Noam Aigerman, Vladimir Kim, Matthew Fisher, and Niloy Mitra. Glass: Geometric latent augmentation for shape spaces. In *CVPR*, pages 18552–18561, June 2022.

[84] Federica Bogo, Michael J. Black, Matthew Loper, and Javier Romero. Detailed full-body reconstructions of moving people from monocular RGB-D sequences. In *ICCV*, pages 2300–2308, December 2015.

[85] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. Fusion4D: Real-time performance capture of challenging scenes. *ACM TOG*, 35(4):1–13, 2016.

[86] Mingsong Dou, Jonathan Taylor, Henry Fuchs, Andrew Fitzgibbon, and Shahram Izadi. 3D scanning deformable objects with a single rgbd sensor. In *CVPR*, pages 493–501, 2015.

[87] Kathleen M. Robinette, Sherri Blackwell, Hein Daanen, Mark Boehmer, Scott Fleming, Tina Brill, David Hoeferlin, and Dennis Burnsides. Civilian American and European Surface Anthropometry Resource (CAESAR) final report. Technical Report AFRL-HE-WP-TR-2002-0169, US Air Force Research Laboratory, 2002.

[88] Ben Bellekens, Vincent Spruyt, Rafael Berkvens, and Maarten Weyn. A survey of rigid 3D pointcloud registration algorithms. In *AMBIENT 2014: the Fourth International Conference on Ambient Computing, Applications, Services and Technologies, August 24-28, 2014, Rome, Italy*, pages 8–13, 2014.

[89] Bailin Deng, Yuxin Yao, Roberto M Dyke, and Juyong Zhang. A survey of non-rigid 3D registration. In *Comput. Graph. Forum*, volume 41, pages 559–589. Wiley Online Library, 2022.

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

[91] D. Hirshberg, M. Loper, E. Rachlin, and M.J. Black. Coregistration: Simultaneous alignment and modeling of articulated 3D shape. In *ECCV*, pages 242–255, October 2012.

[92] Chun-Hao Huang, Cedric Cagniart, Edmond Boyer, and Slobodan Ilic. A bayesian approach to multi-view 4d modeling. *IJCV*, 116(2):115–135, 2016.

[93] Hao Li, Robert W Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *Comput. Graph. Forum*, volume 27, pages 1421–1430. Wiley Online Library, 2008.

[94] Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J. Black. Dyna: A model of dynamic human shape in motion. *ACM TOG*, 34(4):120:1–120:14, August 2015.

[95] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[96] Paul J Besl and Neil D McKay. A method for registration of 3-d shapes. *IEEE TPAMI*, 14(2):239–256, 1992.

[97] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, 2004.

[98] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007.

[99] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pages 1626–1633. IEEE, 2011.

[100] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, pages 652–660, 2017.

[101] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Comput. Graph. Forum*, volume 28, pages 1383–1392. Wiley Online Library, 2009.

[102] Samuele Salti, Federico Tombari, and Luigi Di Stefano. Shot: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:251–264, 2014.

[103] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *NeurIPS*, volume 29. Curran Associates, Inc., 2016.

[104] Chun-Hao Paul Huang, Benjamin Allain, Edmond Boyer, Jean-Sébastien Franco, Federico Tombari, Nassir Navab, and Slobodan Ilic. Tracking-by-detection of 3D human shapes: from surfaces to volumes. *IEEE TPAMI*, 40(8):1994–2008, 2017.

[105] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, pages 5115–5124, 2017.

[106] Lingyu Wei, Qixing Huang, Duygu Ceylan, Etienne Vouga, and Hao Li. Dense human body correspondences using convolutional networks. In *CVPR*, pages 1544–1553, 2016.

[107] Omri Azencot, Anastasia Dubrovina, and Leonidas Guibas. Consistent shape matching via coupled optimization. In *Comput. Graph. Forum*, volume 38, pages 13–25. Wiley Online Library, 2019.

[108] Qifeng Chen and Vladlen Koltun. Robust nonrigid registration by convex optimization. In *ICCV*, pages 2039–2047, 2015.

[109] Jing Ren, Simone Melzi, Peter Wonka, and Maks Ovsjanikov. Discrete optimization for shape matching. In *Computer Graphics Forum*, volume 40, pages 81–96. Wiley Online Library, 2021.

[110] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Active shape models-their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.

[111] Bernhard Egger, William AP Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhoefer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Ko-

rtylewski, Sami Romdhani, et al. 3D morphable face models—past, present, and future. *ACM TOG*, 39(5):1–38, 2020.

[112] Yating Tian, Hongwen Zhang, Yebin Liu, and Limin Wang. Recovering 3D human mesh from monocular images: A survey. *arXiv preprint arXiv:2203.01923*, 2022.

[113] Hanbyul Joo, Tomas Simon, and Yaser Sheikh. Total capture: A 3D deformation model for tracking faces, hands, and bodies. In *CVPR*, pages 8320–8329, 2018.

[114] Ahmed A A Osman, Timo Bolkart, Dimitrios Tzionas, and Michael J. Black. SUPR: A sparse unified part-based human body model. In *ECCV*, 2022.

[115] SizeUSA dataset. `https://www.tc2.com/size-usa.html`, 2017.

[116] Boyang Deng, JP Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. NASA: Neural articulated shape approximation. In *ECCV*, August 2020.

[117] Marko Mihajlovic, Shunsuke Saito, Aayush Bansal, Michael Zollhoefer, and Siyu Tang. COAP: Compositional articulated occupancy of people. In *CVPR*, pages 13191–13200, 2022.

[118] Marko Mihajlovic, Yan Zhang, Michael J Black, and Siyu Tang. LEAP: Learning articulated occupancy of people. In *CVPR*, June 2021.

[119] Garvita Tiwari, Nikolaos Sarafianos, Tony Tung, and Gerard Pons-Moll. Neural-GIF: Neural generalized implicit functions for animating people in clothing. In *ICCV*, October 2021.

[120] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Comput. Graph. Forum*, 2022.

[121] Saeed Ghorbani, Kimia Mahdaviani, Anne Thaler, Konrad Kording, Douglas James Cook, Gunnar Blohm, and Nikolaus F Troje. Movi: A large multi-purpose human motion and video dataset. *Plos one*, 16(6):e0253157, 2021.

[122] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *ICCV*, pages 5442–5451, October 2019.

[123] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[124] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6):194:1–194:17, 2017.

[125] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J. Black. Generating 3D faces using convolutional mesh autoencoders. In *European Conference on Computer Vision (ECCV)*, pages 725–741, 2018.

[126] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, and Karan Singh. Predicting animation skeletons for 3d articulated models via volumetric nets. In *2019 International Conference on 3D Vision (3DV)*, 2019.

[127] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. *ACM Trans. on Graphics*, 39, 2020.

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

[129] Igor Santesteban, Elena Garces, Miguel A. Otaduy, and Dan Casas. SoftSMPL: Data-driven Modeling of Nonlinear Soft-tissue Dynamics for Parametric Humans. *Computer Graphics Forum (Proc. Eurographics)*, 2020.

[130] Zhouyingcheng Liao, Jimei Yang, Jun Saito, Gerard Pons-Moll, and Yang Zhou. Skeleton-free pose transfer for stylized 3d characters. In *European Conference on Computer Vision (ECCV)*. Springer, October 2022.

[131] Jiashun Wang, Xueting Li, Sifei Liu, Shalini De Mello, Orazio Gallo, Xiaolong Wang, and Jan Kautz. Zero-shot pose transfer for unrigged stylized 3d characters, 2023.

[132] Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[133] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3D characters. *ACM Transactions on graphics (TOG)*, 26(3):72–es, 2007.

[134] Mathieu Marsot, Rim Rekik, Stefanie Wuhrer, Jean-Sébastien Franco, and Anne-Hélène Olivier. Correspondence-free online human motion retargeting, 2023.

[135] Peizhuo Li, Kfir Aberman, Rana Hanocka, Libin Liu, Olga Sorkine-Hornung, and Baoquan Chen. Learning skeletal articulations with neural blend shapes. *ACM Transactions on Graphics (TOG)*, 40(4):1–15, 2021.

[136] Keyang Zhou, Bharat Lal Bhatnagar, and Gerard Pons-Moll. 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*, pages 341–357. Springer, 2020.

[137] John E Chadwick, David R Haumann, and Richard E Parent. Layered construction for deformable animated characters. *ACM Siggraph Computer Graphics*, 23(3):243–252, 1989.

[138] Ye Fan, Joshua Litven, and Dinesh K Pai. Active volumetric musculoskeletal systems. *ACM Transactions on Graphics (TOG)*, 33(4):1–9, 2014.

[139] Rinat Abdrashitov, Seungbae Bang, David Levin, Karan Singh, and Alec Jacobson. Interactive modelling of volumetric musculoskeletal anatomy. *ACM Transactions on Graphics*, 40(4), 2021.

[140] V. Modi, L. Fulton, A. Jacobson, S. Sueda, and D.I.W. Levin. Emu: Efficient muscle simulation in deformation space. *Computer Graphics Forum*, Dec 2020.

[141] Sang Il Park and Jessica K Hodgins. Data-driven modeling of skin and muscle deformation. In *ACM SIGGRAPH 2008 papers*, pages 1–6. 2008.

[142] Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J Black. Dyna: A model of dynamic human shape in motion. *ACM Transactions on Graphics (TOG)*, 34(4):1–14, 2015.

[143] Jiayi Eris Zhang, Seungbae Bang, David I. W. Levin, and Alec Jacobson. Complementary dynamics. In *SIGGRAPH Asia*, 2020.

[144] Otman Benchekroun, Jiayi Eris Zhang, Siddhartha Chaudhuri, Eitan Grinspun, Yi Zhou, and Alec Jacobson. Fast complementary dynamics via skinning eigenmodes. *arXiv preprint arXiv:2303.11886*, 2023.

[145] Mianlun Zheng, Yi Zhou, Duygu Ceylan, and Jernej Barbic. A deep emulator for secondary motion of 3d characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5932–5940, 2021.

[146] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *ICCV*, 2015.

[147] Félix G. Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. Robust motion in-betweening. 39(4), 2020.

[148] Chengan He, Jun Saito, James Zachary, Holly Rushmeier, and Yi Zhou. Nemf: Neural motion fields for kinematic animation. In *NeurIPS*, 2022.

[149] Yi-Ling Qiao, Lin Gao, Yu-Kun Lai, and Shihong Xia. Learning bidirectional lstm networks for synthesizing 3d mesh animation sequences, 2018.

[150] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, 2004.

[151] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.*, 23(3):644–651, aug 2004.

[152] Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. Large steps in inverse rendering of geometry. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 40(6), December 2021.

[153] Cristian Romero, Dan Casas, Jesús Pérez, and Miguel Otaduy. Learning contact corrections for handle-based subspace dynamics. *ACM Trans. Graph.*, 40(4), jul 2021.

[154] Yangtuanfeng Wang, Tianjia Shao, Kai Fu, and Niloy Mitra. Learning an intrinsic garment space for interactive authoring of garment animation. *ACM Trans. Graph.*, 38(6), 2019.

[155] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023.

[156] Niladri Shekhar Dutt, Sanjeev Muralikrishnan, and Niloy J. Mitra. Diffusion 3d features (diff3f): Decorating untextured shapes with distilled semantic features, 2023.