LBS Autoencoder: Self-supervised Fitting of Articulated Meshes to Point Clouds

Chun-Liang Li[1], Tomas Simon[2], Jason Saragih[2], Barnabás Póczos[1], Yaser Sheikh[1,2]
[1]Carnegie Mellon University and [2]Facebook Reality Labs
{chunlial, bapoczos}@cs.cmu.edu {firstname.lastname}@fb.com

Abstract

We present LBS-AE; a self-supervised autoencoding algorithm for fitting articulated mesh models to point clouds. As input, we take a sequence of point clouds to be registered as well as an artist-rigged mesh, i.e. *a template mesh equipped with a linear-blend skinning (LBS) deformation space parameterized by a skeleton hierarchy. As output, we learn an LBS-based autoencoder that produces registered meshes from the input point clouds. To bridge the gap between the artist-defined geometry and the captured point clouds, our autoencoder models pose-dependent deviations from the template geometry. During training, instead of using explicit correspondences, such as key points or pose supervision, our method leverages LBS deformations to bootstrap the learning process. To avoid poor local minima from erroneous point-to-point correspondences, we utilize a* structured *Chamfer distance based on part-segmentations, which are learned concurrently using self-supervision. We demonstrate qualitative results on real captured hands, and report quantitative evaluations on the FAUST benchmark for body registration. Our method achieves performance that is superior to other unsupervised approaches and comparable to methods using supervised examples.*

1. Introduction

The registration of unstructured point-clouds to a common mesh representation is an important problem in computer vision and has been extensively studied in the past decades. Works in this area can be coarsely grouped together based on how much prior knowledge and supervision is incorporated into the fitting method. On one end of the spectrum, there are entirely unsupervised and object-agnostic models, such as FoldingNet [50] or AtlasNet [13]. These methods learn to deform a flat 2D surface to match the target geometry, while making no assumptions about the objects being modeled other than that they can be represented as a 2D surface. Adding slightly more prior knowledge, 3D-CODED [12] uses a template mesh (*e.g.* hand or body) with a topology better suited to the object of interest.

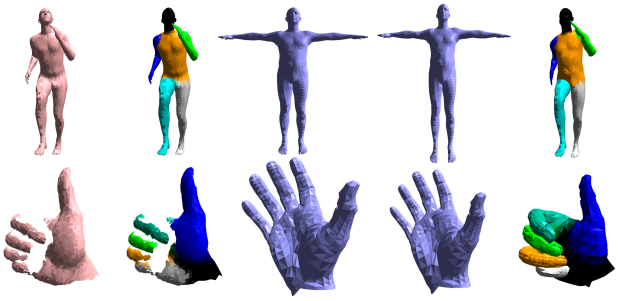


Figure 1: Given point clouds sampled from the surface of an input shape (a), our model infers a coarse segmentation (b), and learns to deform a given template (c), through a combination of deformations of the template (d) as well as pose deformation parameterized by LBS to match the reconstruction (e). We use a *structured Chamfer distance* that uses the inferred segmentation of the data (b) as coarse correspondence to measure distance between matching regions to avoid local optima in the Chamfer distance.

On the other end of the spectrum are highly specialized models for specific objects, such as hands and bodies. Works of this kind include SCAPE [2], Dyna [34], SMPL [28], and MANO [38]. These models are built using high-resolution 3D scans with correspondence and human curation. They model correctives for different poses and modalities (*e.g.* body types) and can be used as high-quality generative models of geometry. A number of works learn to manipulate these models to fit data based on different sources of supervision, such as key points [6, 22, 31, 43, 15] and/or prior distributions of model parameters [18, 17].

In this paper, we present an unsupervised/self-supervised algorithm, LBS Autoencoder (LBS-AE), to fit such articulated mesh models to point cloud data. The proposed algorithm is a middle ground of the two ends of spectrum discussed above in two senses.

First, we assume an articulated template model of the object class is available, but not the statistics of its articulation in our dataset nor the specific shape of the object instance. We argue that this prior information is widely available for many common objects of interest in the form of "rigged"

or "skinned" mesh models, which are typically created by artists for use in animation. In addition to a template mesh describing the geometric shape, these prior models have two more components: (1) a kinematic hierarchy of transforms describing the degrees of freedom, and (2) a skinning function that defines how transforms in the hierarchy influence each of the mesh vertices. This enables registration to data by manipulating the transforms in the model. One common example is Linear Blending Skinning (LBS). Therefore, instead of relying on deep networks to learn the full deformation process from a single template [12], we leverage LBS as part of the decoder to model coarse joint deformations. Different from hand-crafted models such as SMPL [28], LBS by itself does not model pose-dependent correctives between the template and data, nor does it model the space of non-articulated shape variation (*e.g.* body shape). To model these, we also allow our network to learn deformations of the template mesh which, when posed by LBS, result in a better fit to the data. The encoder therefore learns a latent representation from which it can infer both joint angles for use by the LBS deformation, as well as corrective deformations to the template mesh.

Second, for fitting models to data during the training, existing works either rely on explicit supervision (*e.g.* correspondence [12] and key points [15]) or unsupervised nearest neighbors search (*e.g.* Chamfer Distance (CD) [50]) to find point correspondence between the model and data for measuring reconstruction loss. Rather than using external supervision, we introduce a "Structured Chamfer Distance" (SCD), which improves the blind nearest neighbor search in CD based on an inferred coarse correspondence. The idea is to segment the point clouds into corresponding regions (we use regions defined by the LBS weighting). After inferring the segmentation on the input point cloud and the template, we then apply nearest neighbor search between corresponding regions as high-level correspondence. The challenge is we do not assume external supervision to be available for the input point clouds. Instead, we utilize the learned LBS-AE model to generate *self-supervision* to train the segmentation network from scratch. As the LBS-AE fitting is improved during training, the training data from self-supervision for segmentation also improves, leading to improved segmentation of the real data. We are then able to use the improved segmentation to achieve better correspondence and in turn better LBS-AE model fitting. In this paper, we present a joint training framework to learn these two components simultaneously. Since LBS-AE does not require any explicit correspondence nor key points, it is similar to approaches which are sometimes referred to as "unsupervised" in the pose estimation literature [42, 9], but it is different from existing unsupervised learning approach [50] in that it leverages LBS deformation to generate *self-supervision* during training.

In this work, we show that the space of deformations described by an artist-defined rig may sometimes already be sufficiently constrained to allow fitting to real data without any additional labeling. Such a model-fitting pipeline without additional supervision has the potential to simplify geometric registration tasks by requiring less human labeling effort. For example, when fitting an artist-defined hand rig to point clouds of hands, our method allows for unsupervised hand pose estimation. When fitting a body model to 3D scans of body data, this allows recovering the joint angles of the body as well as registering the mesh vertices. In the experiments, we present the results on fitting real hands as well as benchmark body data on the SURREAL and FAUST datasets.

2. Proposed Method

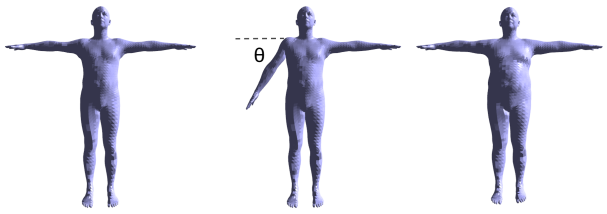
We propose to learn a function $\mathcal{F}(\cdot)$ that takes as input an unstructured point cloud $\mathbf{X}=\{x\}_{i=1}^{n}$, where each x_i is a 3D point and n is a variable number, and produces as output a fixed number m of corresponded vertices $\mathbf{V}=\{v_i\}_{i=1}^{m}$, where $\mathbf{V} = \mathcal{F}(\mathbf{X})$. The vertices $\mathbf{V}$ form a mesh with fixed topology whose geometry should closely match that of the input[1]. Rather than allowing $\mathcal{F}(\cdot)$ to be any arbitrary deformation produced by a deep neural network (as in [50, 13]), we force the output to be produced by Linear Blending Skinning (LBS) to explicitly encode the motion of joints. We allow additional non-linear deformations (also given by a neural network) to model deviations from the LBS approximation. However, an important difference with respect to similar models, such as SMPL [28] or MANO [38], is that we do not pre-learn the space of non-LBS deformations on a curated set (and then fix them) but rather learn these simultaneously on the data that is to be aligned, with no additional supervision.

Linear Blending Skinning We start by briefly introducing LBS [29], which is the core building component of the proposed work. LBS models deformation of a mesh from a rest pose as a weighted sum of the skeleton bone transformations applied to each vertex. We follow the notation outlined in [28], which is a strong influence on our model. An LBS model with J joints can be defined as follows

$$\mathbf{V} = M(\Theta, \mathbf{U}), \tag{1}$$

with $\mathbf{V}$ the vertices of the deformed shape after LBS. The LBS function M takes two parameters, one is the vertices $\mathbf{U} = \{u_i\}_{i=1}^{m}$ of a base mesh (template), and the other are the relative joint rotation angles $\Theta \in \mathbb{R}^{J\times 3}$ for each joint j with respect to its parents. If $\Theta = \mathbf{0}$, then $M(\mathbf{0}, \mathbf{U}) = \mathbf{U}$. Two additional parameters, the skinning weights w and the joint hierarchy K, are required by LBS. We will consider them fixed by the artist-defined rig. In particular,

[1]Note that, although we assume the inputs are point clouds, they could also be the vertices of a mesh without using any topology information.



(a) $\mathbf{U}$ (b) $M(\theta, \mathbf{U})$ (c) $\mathbf{U}^d$

Figure 2: (a) Template mesh, (b) LBS deformation of the template using joint angles θ, and (c) a deformed template.

$w \in \mathbb{R}^{m \times J}$ defines the weights of each vertex contributing to joint j and $\sum_j w_{i,j} = 1$ for all i. K is the joint hierarchy. Each vertex $v_i \in \mathbf{V}$ can then be written as

$$v_i = (\mathbf{I}_3, \mathbf{0}) \cdot \sum_{j=1}^{J} w_{i,j} \mathcal{T}_j(\Theta, K) \begin{pmatrix} u_i \\ 1 \end{pmatrix},$$

where $\mathcal{T}_j(\Theta, K) \in \mathrm{SE}(3)$ is a transformation matrix for each joint j, which encodes the transformation from the rest pose to the posed mesh in world coordinate, constructed by traversing the hierarchy K from the root to j. Since each v_i is constructed by a sequence of linear operations, the LBS $M(\Theta, \mathbf{U})$ is differentiable respect to Θ and $\mathbf{U}$. A simple example constructed from the LBS component in SMPL [28] is shown in Figure 2a and 2b.

In this work, both the joint angles and the template mesh used in the LBS function are produced by deep networks from the input point cloud data,

$$\mathbf{V} = M(f(\mathbf{X}), d(\mathbf{X}, \mathbf{U})), \tag{2}$$

where we identify a *joint angle estimation* network f, and a *template deformation* network d which we describe below.

Joint Angle (Pose) Estimation Given an LBS model defined in (1), the goal is to regress joint angles based on input $\mathbf{X}$ via a function $f : \mathbf{X} \to \Theta$ such that $M(f(\mathbf{X}), \mathbf{U}) \approx \mathbf{X}$. We use a deep neural network, which takes set data (*e.g.* point cloud) as input [35, 51] to f, but we must also specify how to compare $\mathbf{X}$ and $\mathbf{V}$ from $M(\cdot)$. Losses that assume uniformly sampled surfaces (such as distribution matching [26] or optimal transport) are less suitable, because reconstructed point clouds typically exhibit some amount of missing data and non-uniform sampling.

Instead, we adopt a Chamfer distance (CD) [50] defined as $\mathcal{L}_c(\mathbf{X}, \mathbf{V}) =$

$$\frac{1}{n} \sum_{i=1}^{n} \|x_i - \mathcal{N}_{\mathbf{V}}(x_i)\|^2 + \frac{1}{m} \sum_{j=1}^{m} \|v_j - \mathcal{N}_{\mathbf{X}}(v_j)\|^2, \tag{3}$$

where $\mathcal{N}_{\mathbf{V}}(x_i) = \arg\min_{v_j \in \mathbf{V}} \|x_i - v_j\|$ is the nearest neighbor of x_i in $\mathbf{V}$. This is also called Iterative Closest Point (ICP) in the registration literature [5]. After finding nearest neighbors, we learn f by back-propagating this point-wise loss through the differentiable LBS $\mathbf{V} = M(f(\mathbf{X}), \mathbf{U})$. Also note that we only sample a subset of points for estimating (3) under SGD training schemes.

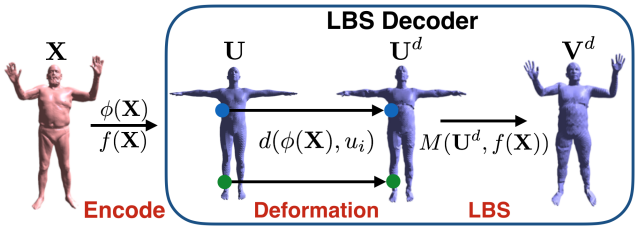


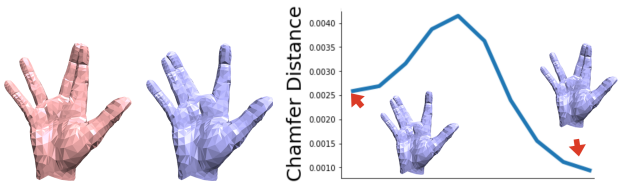
Figure 3: LBS-AE. Given a point cloud $\mathbf{X}$ of a input shape, we encode $\mathbf{X}$ into a latent code $\phi(\mathbf{X})$ and the inferred joint angles $f(\mathbf{X})$. The decoder contains a deformation network d to deform the template $\mathbf{U}$ into $\mathbf{U}^d$, then uses a LBS to pose $\mathbf{U}^d$ into $\mathbf{V}^d$ as the reconstruction.

In practice, we observe that it takes many iterations for PointNet [35] or DeepSet [51] architectures to improve if the target loss is CD instead of corresponded supervision. Similar behaviors were observed in [50, 26], where the algorithms may take millions of iterations to converge. To alleviate this problem, we utilize LBS to generate data based on a given Θ' for *self-supervision* by optimizing

$$\min_f \mathcal{L}_\Theta = \|f(M(\Theta', \mathbf{U})) - \Theta'\|^2.$$

It is similar to the *loop-back* loss [9] that ensures f can correctly reinterpret the model's own output from M. Different from [9, 17], we do not assume a prior pose distribution is available. Our Θ' comes from two sources of randomness. One is uniform distributions within the given joint angle ranges (specified by the artist-defined rig) and the second is we uniformly perturb the inferred angles from input samples with a small uniform noise on the fly, which can gradually adapt to the training data distribution when the estimation is improved as training progresses (see Section 2.1 and Figure 6).

Template Deformation Although LBS can represent large pose deformations, due to limitations of LBS as well as differences between the artist modeled mesh and the real data, there will be a large residual in the fitting. We refer to this residual as a *modality gap* between the model and reality, and alleviate this difference by using a neural network d to produce the template mesh to be posed by LBS. The deformation network $d(\phi(\mathbf{X}), u_i)$ takes two sources as input, where u_i is each vertex in the template mesh $\mathbf{U}$, and $\phi(\mathbf{X})$ are features from an intermediate layer in f, which contains information about the state of $\mathbf{X}$. This yields a deformed template $\mathbf{U}^d = \{d(\phi(\mathbf{X}), u_i)\}_{i=1}^m$. One example is shown in Figure 2c. After LBS, we denote the deformed and posed mesh as $\mathbf{V}^d = M(f(\mathbf{X}), \mathbf{U}^d)$, and denote by



(a) Input (b) Estimate (c)

Figure 4: When we try to move the middle finger of the current estimate (b) toward the target (a), the Chamfer distance increases before decreasing, showing a local optimum that is difficult to overcome.

$\mathbf{V} = M(f(\mathbf{X}), \mathbf{U})$ the posed original template.

If d is high-capacity, $f(\mathbf{X})$ can learn to generate all-zero joint angles for the LBS component (ignoring the input $\mathbf{X}$), and explain all deformations instead with d. That is, $M(f(\mathbf{X}), \mathbf{U}^d) = M(\mathbf{0}, \mathbf{U}^d) = \mathbf{U}^d \approx \mathbf{X}$, which reduces to the unsupervised version of [12]. Instead of using explicit regularization to constrain d (*e.g.* $\|d(\phi(\mathbf{X}), \mathbf{U}^B)\|$), we propose a composition of two Chamfer distances as

$$\mathcal{L}_{c^2,\lambda} = \mathcal{L}_c\left(\mathbf{X}, \mathbf{V}^d\right) + \lambda \mathcal{L}_c\left(\mathbf{X}, \mathbf{V}\right). \tag{4}$$

The second term in (4) enforces $f(\mathbf{X})$ to learn correct joint angles even without template deformation.

Lastly, we follow [18, 12] and apply Laplacian regularization $\mathcal{L}_{\text{lap}} = \|L\mathbf{V}^d\|$ to encourage smoothness of the deformed template, where L is the discrete Laplace-Beltrami operator constructed from mesh $\mathbf{U}$ and its faces.

LBS-based Autoencoder The proposed algorithm can be interpreted as an encoder-decoder scheme. The joint angle regressor is the encoder, which compresses $\mathbf{X}$ into style codes $\phi(\mathbf{X})$ and interpretable joint angles $f(\mathbf{X})$. The decoder, different from standard autoencoders, is constructed by combining a human designed LBS function and a style deformation network d on the base template. We call the proposed algorithm LBS-AE as shown in Figure 3.

2.1. Structured Chamfer Distance

To train an autoencoder, we have to define proper reconstruction errors for different data. In LBS-AE, the objective that provides information about input point clouds is only CD (3). However, it is known that CD has many undesirable local optima, which hinders the algorithm from improving.

A local optimum example of CD is shown in Figure 4. To move the middle finger from the current estimate towards the index finger to fit the input, the Chamfer distance must increase before decreasing. This local optimum is caused by incorrect correspondences found by nearest neighbor search (the nearest neighbor of the middle finger of the current estimate is the ring finger of the input).

High-Level Correspondence Given a pair of sets $(\mathbf{V}, \mathbf{X})$, for each $v \in \mathbf{V}$, we want to find its correspondence $\mathcal{C}_{\mathbf{X}}(v)$ in $\mathbf{X}$. In CD, we use the nearest neighbor $\mathcal{N}_{\mathbf{X}}(v)$ to approximate $\mathcal{C}_{\mathbf{X}}(v)$, which can be wrong, as shown in Figure 4. Instead of searching for nearest neighbors $\mathcal{N}_{\mathbf{X}}(v)$ over the entire set X, we propose to search within a subset $X' \subset X$, where $\mathcal{C}_{\mathbf{X}}(v) \in X'$, by eliminating irrelevant points in $\mathbf{X}$. Following this idea, we partition $\mathbf{X}$ into k subsets, $\mathbf{X}^1 \dots \mathbf{X}^k$, where we use $s(x; \mathbf{X}) \in \{1, \dots, k\}$ to denote which subset x belongs to. A desirable partition should ensure $s(v; \mathbf{V}) = s(\mathcal{C}_{\mathbf{X}}(v); \mathbf{X})$; then, to find the nearest neighbor of v, we need only consider $\mathbf{X}^{s(v)} \subset \mathbf{X}$. We then define the *Structured Chamfer Distance* (SCD) as $\mathcal{L}_s(\mathbf{X}, \mathbf{V}) =$

$$\frac{1}{n}\sum_{i=1}^{n} \|x_i - \mathcal{N}_{\mathbf{V}^{s(x)}}(x_i)\|^2 + \frac{1}{m}\sum_{j=1}^{m} \|v_j - \mathcal{N}_{\mathbf{X}^{s(v)}}(v_j)\|^2, \tag{5}$$

where we ease the notation of $s(x, \mathbf{X})$ and $s(v, \mathbf{V})$ to be $s(x)$ and $s(v)$. Compared with CD, which finds nearest neighbors from *all to all*, SCD uses *region to region* based on the high-level correspondence by leveraging the structure of data. Similar to (4), we define

$$\mathcal{L}_{s^2,\lambda} = \mathcal{L}_s\left(\mathbf{X}, \mathbf{V}^d\right) + \lambda \mathcal{L}_s\left(\mathbf{X}, \mathbf{V}\right). \tag{6}$$

In this paper, we partition the vertices based on the LBS skinning weights at a chosen granularity. Examples of hand and body data are shown in Figure 5, which use the structure and our prior knowledge of the human body. These satisfy the property that the true correspondence is within the same partition. With the proposed SCD, we can improve the local optimum in Figure 4.

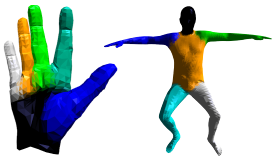


Figure 5: Joint Partitions.

Segmentation Inference For the deformed mesh $\mathbf{V}$, we can easily infer the partition $s(v; \mathbf{V})$, because the mapping between vertices and joints is defined by the LBS skinning weights w. We directly use $\text{argmax}_j\, w_{i,j}$ as labels. Without additional labeling or keypoint information, the difficulty is to infer $s(x; \mathbf{X})$ for $x \in \mathbf{X}$, which is a point cloud segmentation task [35]. However, without labels for $\mathbf{X}$, we are not able to train a segmentation model on $\mathbf{X}$ directly. Instead, similar to the self-supervision technique used for training the joint angle regressor, we propose to train a segmentation network s with the data $(\mathbf{V}^d, \mathbf{Y})$ generated by LBS, where $\mathbf{Y}$ are the labels for w defined in LBS and $\mathbf{V} = M(\Theta, \mathbf{U}^d)$. Note that Θ follows the same distribution as before, which contains uniform sampling for exploration and perturbation of the inferred angles $f(\mathbf{X})$, as shown in Figure 6. Instead of using the base template $\mathbf{U}$ only, we use the inferred deformed template $\mathbf{U}^d$ to adapt to the real data modality, which improves performance (see Section 4.1).

The final objective for training the shape deformation

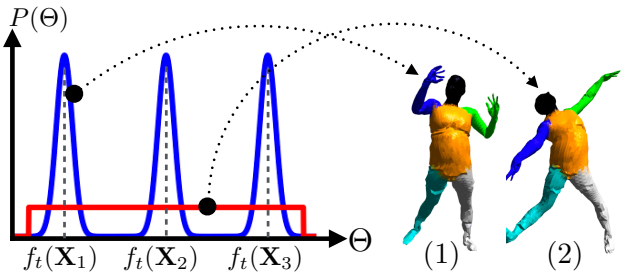


Figure 6: The mixture distribution of self-supervision data from the LBS at iteration t. We sample from (1) the perturbed distribution centered at the $f_t(\mathbf{X}_i)$ and (2) a uniform distribution.

Algorithm 1 LBS-AE with SCD

Inputs: • Point Clouds: $\{\mathbf{X}\}$
• LBS: $M(;w,K,\mathbf{U})$ and angle ranges (R_l, R_u)
Pretrain s on uniformly sampled poses from LBS
while f and d have not converged:
1. Sample minibatch $\{\mathbf{X}_i\}_{i=1}^B$, $\{\mathbf{X}'_i\}_{i=1}^B$
2. $\Theta' = \{f(\mathbf{X}_i) + \epsilon_i\}_{i=1}^B \cup \Theta_r \sim \text{Unif}(R_l, R_u)$
3. Generate $(\mathbf{V}^d, \mathbf{Y})$ based on Θ' to update s
4. Infer segmentation labels $\{s(\mathbf{X}'_i)\}_{i=1}^B$
5. Update f and d based on (1)-(3) (Eq. (7))

pipeline including $f(\cdot)$ and $d(\cdot)$ is[2]

$$\mathcal{L} = \mathcal{L}_{c^2,0.5} + \lambda_s \mathcal{L}_{s^2,0.5} + \lambda_{\texttt{lap}} \mathcal{L}_{\texttt{lap}} + \lambda_\theta \mathcal{L}_\Theta, \quad (7)$$

and we use standard cross-entropy for training s. In practice, since s is noisy during the first iterations, we pretrain it for $50K$ iterations with poses from uniform distributions over joint angles. Note that, for pretraining, we can only use the base template $\mathbf{U}$ to synthesize data. After that, we then learn everything jointly by updating each network alternatively. The final algorithm, LBS-AE with SCD as reconstruction loss, is shown in Algorithm 1.

3. Related Works

LBS Extensions Various extensions have been proposed to fix some of the shortcomings of LBS [24, 41, 46, 20, 37, 16, 19, 23, 52, 28, 4], where we only name afew here. The proposed template deformation follows the idea of [21, 37, 52, 28] to model the modalities and corrections of LBS on the base template rest pose. [52, 28] use PCA-like algorithms to model modalities via a weighted sum of learned shape basis. Instead, our approach is similar to [4] by learning modalities via a deformation network. The main difference between LBS-AE and [52, 28, 4] is we do not rely on correspondence information to learn the template deformation d a priori. We simultaneously learn d and infer pose parameters without external labeling.

Deep Learning for 3D Data Many deep learning techniques have been developed for different types of 3D information, such as 3D voxels [10, 49, 48], geometry images [39, 40], meshes [8], depth maps [45] and point clouds [35, 36, 51]. Autoencoders for point clouds are explored by [50, 13, 26, 1].

Model Fitting with Different Knowledge Different works have studied to registration via fitting a mesh model by leveraging different levels of information about the data. [17] use SMPL [28] to reconstruct meshes from images by using key points and prior knowledge of distributions of pose parameters. [18] explore using a template instead of a controllable model to reconstruct the mesh with key points. [6, 15] also adopt pretrained key point detectors from other sources of data as supervision. Simultaneous training to improve model fitting and key point detection are explored by [22, 31]. The main difference from the proposed joint training in LBS-AE is we do not rely on an additional source of real-world data to pretrain networks, as needed to train these key point detectors. [47] share a similar idea of using segmentation for nearest neighbor search, but they trained the segmentation from labeled examples. [9] propose to control morphable models instead of rig models for modeling faces. They also utilize prior knowledge of the 3DMM parameter distributions for real faces. We note that most of the works discussed above aim to recover 3D models from images. [12] is the most related work to the proposed LBS-AE, but doesn't use LBS-based deformation. They use a base template and learn the full deformation process with a neural network trained by correspondences provided a priori or from nearest neighbor search. More comparison between [12] and LBS-AE will be studied in Section 4. Lastly, learning body segmentation via SMPL is studied by [44], but with a focus on learning a segmentation using SMPL with parameters inferred from real-world data to synthesize training examples.

Loss Function with Auxiliary Neural Networks Using auxiliary neural networks to define objectives for training targeted models is also broadly studied in GAN literature (*e.g.* [11, 30, 33, 3, 25, 32, 14]). [26] use a GAN loss for matching input and reconstructed point clouds. By leveraging prior knowledge, the auxiliary network adopted by LBS-AE is an interpretable segmentation network which can be trained without adversarial training.

4. Experiment

Datasets We consider hand and body data. For body data, we test on FAUST benchmark [7], which captures real human body with correspondence labeling. For hand data, we use a multi-view capture system to captured $1,524$ poses from three people, which have missing area and different

[2] We use $\lambda = 0.5$, $\lambda_{\texttt{lap}} = 0.005$, $\lambda_\theta = 0.5$ in all experiments.

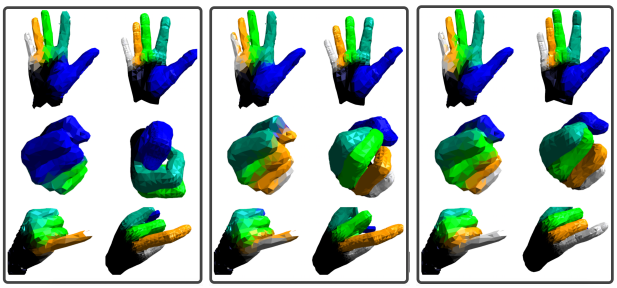
Figure 7: Examples of the captured hands.

densities of points across areas. The examples of reconstructed meshes are shown in Figure 7. For numerical evaluation, in addition to FAUST, we also consider synthetic data since we do not have labeling information on the hand data (*e.g.* key points, poses, correspondence). To generate synthetic hands, we first estimate pose parameters of the captured data under LBS. To model the modality gap, we prepare different base templates with various thickness and length of palms and fingers. We then generate data with LBS based on those templates and the inferred pose parameters. We also generate synthetic human body shapes using SMPL [6]. We sample $20,000$ parameter configurations estimated by SURREAL [44] and $3,000$ samples of bent shapes from [12]. For both synthetic hand and body data, the scale of each shape is in $[-1,1]^3$ and we generate 2300 and 300 examples as holdout testing sets.

Architectures The architecture of f follows [26] to use DeepSet [51], which shows competitive performance with PointNet [35] with half the number of parameters. The output is set to be $J \times 3$ dimensions, where J is the number of joints. We use the previous layer's activations as $\phi(\mathbf{X})$ for d. We use a three layer MLP to model d, where the input is the concatenation of v, $f(\mathbf{X})$ and $\phi(\mathbf{X})$, and the hidden layer sizes are 256 and 128. For segmentation network s, we use [35] because of better performance. For hand data, we use an artist-created LBS, while we use the LBS part from SMPL [28] for body data.

4.1. Study on Segmentation Learning

One goal of the proposed LBS-AE is to leverage geometry structures of the shape, by learning segmentation jointly to improve correspondence finding via nearest neighbor searching when measuring the difference between two shapes. Different from previous works (*e.g.* [47]), we do not rely on any human labels. We study how the segmentation learning with self-supervision interacts with the model fitting to data. We train different variants of LBS-AE to fit the captured hands data. The first is learning LBS-AE with CD only (LBS-AE$_{\text{CD}}$). The objective is (7) without $\mathcal{L}_{s^2,0.5}$. We then train the segmentation network s for SCD with hand poses sampled from uniform distributions based on $\mathbf{U}$ instead of $\mathbf{U}^d$. Note that there is no interaction between learning s and the other networks f and d. The segmentation and reconstructed results are shown in Figure 8a. We observe that the segmentation network trained on randomly sampled poses from a uniform distribution can only segment easy poses correctly and fail on challenging cases,



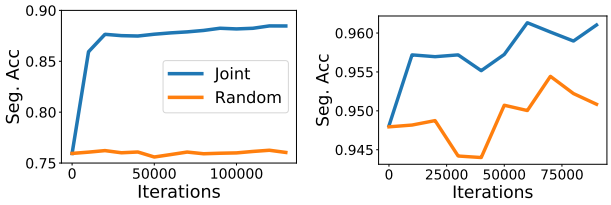
(a) LBS-AE$_{\text{CD}}$ (b) Modality Gap (c) LBS-AE

Figure 8: Ablation study of the proposed LBS-AE. For each block, the left column is the inferred segmentations of input shapes while the right column is the reconstruction.

such as feast poses, because of the difference between true pose distributions and the uniform distribution used as well as the modality gaps between real hands and synthetic hands from LBS. On the other hand, LBS-AE$_{\text{CD}}$ is stuck at different local optimums. For example, it recovers to stretch the ring finger instead of the little finger for the third pose.

Secondly, we study the importance of adapting to different modalities. In Figure 8b, we train segmentation and LBS fitting jointly with SCD. However, when we augment the data for training segmentation, we only adapt to pose distributions via $f(\mathbf{X})$, instead of using the deformed $\mathbf{U}^d$. Therefore, the training data for s for this case has a modality gap between it and the true data. Compared with Figure 8a, the joint training benefits the performance, for example, on the feast pose. It suggests how good segmentation learning benefits reconstruction. Nevertheless, it still fails on the third pose. By training LBS-AE and the segmentation jointly with inferred modalities and poses, we could fit the poses better as shown in Figure 8c. This difference demonstrates the importance of training segmentation adapting to the pose distributions and different modalities.

Numerical Results We also quantitatively investigate the learned segmentation when ground truth is available. We train s with (1) randomly sampled shapes from uniform distributions over joint angle ranges (`Random`) and (2) the proposed joint training (`Joint`). We use pretraining as initialization as describing in Section 2.1. We then train these two algorithms on the synthetic hand and body data and evaluate segmentation accuracy on the testing sets. The results are shown in Figure 9. `Random` is exactly the same as pretraining. After pretraining, `Random` is almost converged. On the other hand, `Joint` improves the segmentation accuracy in both cases by gradually adapting to the true pose distribution when the joint angle regressor f is improved. It justifies the effectiveness of the proposed joint training where we can infer the segmentation in a self-supervised manner. For hand data, as we show in Figure 8, there are many *touching-skin* poses where fingers are touched to each other. For those poses, there are strong correlations between



(a) Synthetic Hands (b) SMPL

Figure 9: Segmentation accuracy on holdout testing sets.

joints in each pose, which are hard to be sampled by a simple uniform distribution and results in a performance gap in Figure 9a. For body data, many poses from SURREAL are with separate limbs, which `Random` can generalize surprisingly well. Although it seems `Joint` only leads to incremental improvement over `Random`, we argue this gap is substantial, especially for resolving challenging touching-skin cases as we will show in Section 4.3.

4.2. Qualitative Study

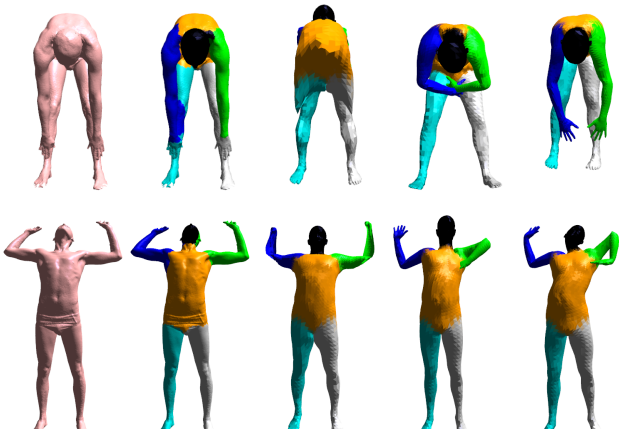
We compare the proposed algorithm with the unsupervised learning variant of [12], which learns the deformation by entirely relying on neural networks. Their objective is similar to (7), but using CD and Laplacian regularization only. For fair comparison, we also generate synthetic data on the fly with randomly sampled poses and correspondence for [12], which boosts its performance. We also compare with the simplified version of the proposed algorithm by using CD instead of SCD, which is denoted as LBS-AE$_{\text{CD}}$ as above.

We fit and reconstruct the hand and body data as shown in Figure 10. For the thumb-up pose, due to wrong correspondences from nearest neighbor search, both [12] and LBS-AE$_{\text{CD}}$ reconstruct wrong poses. The wrong correspondence causes problems to [12]. Since the deformation from templates to targeted shapes fully relies on a deep neural network, when the correspondence is wrong and the network is powerful, it learns distorted deformation even with a Laplacian regularization. On the other hand, since LBS-AE$_{\text{CD}}$ still utilizes LBS, the deformation network d is easier to regularize, which results in better finger reconstructions. We note that [12] learns proper deformation if the correspondence can be found correctly, such as the third row in Figure 10. In both cases, the proposed LBS-AE can learn segmentation well and recover the poses better.

Lastly, we consider fitting FAUST, with only 200 samples, as shown in Figure 11. With limited and diverse poses, we have less hint of how the poses deform [47], a nearest neighbor search is easily trapped in bad local optimums as we mentioned in Figure 4. The proposed LBS-AE still results in reasonable reconstructions and segmentation, though the right arm in the second row suffers from the local optimum issues within the segmentation. A fix is to learn more fine-grained segmentation, but it brings the trade-off between task difficulty and model capacity, which we leave for future work.

(a) Input (b) Segment (c) [12] (d) CD (e) LBS-AE

Figure 10: Qualitative comparisons on captures hands and SURREAL (SMPL). Given point clouds sampled from the surfaces of input shapes (a), (c-e) are the reconstructions from different algorithms. (b) is the inferred segmentation of LBS-AE on the input shape.



(a) Input (b) Segment (c) [12] (d) CD (e) LBS-AE

Figure 11: Qualitative Comparison on FAUST.

	SMPL			Syn. Hand		
Algorithm	Recon	Pose	Corre.	Recon	Pose	Corre.
Unsup. [12]	0.076	0.082	0.136	0.099	0.035	0.176
Unsup.+Aug [12]	0.081	0.081	0.132	0.069	0.049	0.140
Sup. [12]	0.073	0.071	0.104	0.062	0.047	0.135
LBS-AE$_{\text{CD}}$	0.051	0.152	0.147	0.082	0.069	0.168
LBS-AE$_{\text{RAND}}$	0.041	0.058	0.100	0.069	0.050	0.137
LBS-AE	**0.037**	**0.048**	**0.091**	**0.053**	**0.035**	**0.111**

Table 1: Quantitative results on synthetic data.

4.3. Quantitative Study

We conduct quantitative analysis on reconstruction, pose estimation, and correspondence on synthetic hand and body data. We use $\sqrt{\text{CD}}$ as the proxy to reconstructions. Pose estimation compares the average ℓ_2 distance between true joint positions and inferred ones while correspondence also measures the average ℓ_2 between found and true correspondences. We randomly generate 4000 testing pairs from the testing data for correspondence comparison. Given two shapes, we fit the shapes via the trained models. Since we know the correspondence of the reconstructions, we project the data onto the reconstructions to find the correspondence. For more details, we refer readers to [12].

We compare three variants of [12], including the supervised version with full correspondence, and the unsupervised version with and without synthetic data augmentation aforementioned. For LBS-AE, we also consider three variants, including a simple CD baseline (LBS-AE$_{\text{CD}}$), a segmentation network s trained on poses from uniform distributions LBS-AE$_{\text{RAND}}$ and joint training version (LBS-AE). The results are shown in Table 1.

For LBS-AE variants, the jointly trained LBS-AE is better than LBS-AE$_{\text{CD}}$ and LBS-AE$_{\text{RAND}}$. It supports the hypothesis in Section 4.1, that joint training facilitates improving model fitting and segmentation. Also, as shown in Section 4.1, the pretrained segmentation network still has reasonable testing accuracy and brings an improvement over using CD loss only. On the other hand, the supervised version of [12] trained with full correspondence is worse than the proposed unsupervised LBS-AE due to generalization ability. For correspondence on the SMPL training set, supervised [12] achieves 0.065 while LBS-AE achieve 0.069. If we increase the training data size three times, supervised [12] improves its correspondence result to be 0.095. For hand data, supervised [12] generalizes even worse with only 1500 training examples. It suggests that leveraging LBS models into the model can not only use smaller networks but also generalize better than relying on an unconstrained deformation from a deep network.

Deformation Network We also investigate the ability of the deformation in LBS-AE. For data generated via SMPL, we know the ground truth of deformed templates $\mathbf{U}^{gt}$ of each shape. The average ℓ_2 distance between corresponding points from $\mathbf{U}^{gt}$ and $\mathbf{U}^{d}$ is 0.02, while the average distance between $\mathbf{U}^{gt}$ and $\mathbf{U}$ is 0.03.

Algorithm	Inter. error (cm)	Intra. err (cm)
FMNet [27]	4.826	2.44
Unsup. [12] (230K)	4.88	-
Sup. [12] (10K)	4.70	-
Sup. [12] (230K)	3.26	1.985
LBS-AE (23K)	4.08	2.161

Table 2: Correspondence results on FAUST testing set.

(a) (b) (c)

Figure 12: Inferred correspondence of FAUST testing data.

Real-World Benchmark. One representative real-world benchmark is FAUST [7]. We follow the protocol used in [12] for comparison, where they train on SMPL with SURREAL parameters and then fine-tune on FAUST. In [12], they use a different number of data from SMPL with SURREAL parameters, while we only use 23K. The numerical results are shown in Table 2. With only 23K SMPL data and self-supervision, we are better than unsupervised [12] with 50K data, supervised [12] with 10K data, and the supervised learning algorithm FMNet [27]. We show some visualization of the inferred correspondence in Figure 12.

5. Conclusion

We propose a self-supervised autoencoding algorithm, LBS-AE, to align articulated mesh models to point clouds. The decoder leverages an artist-defined mesh rig, and using LBS. We constrain the encoder to infer interpretable joint angles. We also propose the structured Chamfer distance for training LBS-AE, defined by inferring a meaningful segmentation of the target data to improve the correspondence finding via nearest neighbor search in the original Chamfer distance. By combining LBS-AE and the segmentation inference, we demonstrate we can train these two components simultaneously without supervision (labeling) from data. As training progress, the proposed model can start adapting to the data distribution and improve with self-supervision. In addition to opening a new route to model fitting without supervision, the proposed algorithm also provides a successful example showing how to encode existing prior knowledge in a geometric deep learning model.

References

[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, 2018.

[2] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: Shape completion and animation of people. *TOG*, 2005.

[3] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *ICML*, 2017.

[4] S. W. Bailey, D. Otte, P. Dilorenzo, and J. F. O'Brien. Fast and deep deformation approximations. *TOG*, 2018.

[5] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. In *TPAMI*, 1992.

[6] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *ECCV*, 2016.

[7] F. Bogo, J. Romero, M. Loper, and M. J. Black. Faust: Dataset and evaluation for 3d mesh registration. In *CVPR*, 2014.

[8] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 2017.

[9] K. Genova, F. Cole, A. Maschinot, A. Sarna, D. Vlasic, and W. T. Freeman. Unsupervised training for 3d morphable model regression. In *CVPR*, 2018.

[10] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[12] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. 3d-coded: 3d correspondences by deep deformation. In *ECCV*, 2018.

[13] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. Atlasnet: A papier-m\^ ach\'e approach to learning 3d surface generation. In *CVPR*, 2018.

[14] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. In *NIPS*, 2017.

[15] H. Joo, T. Simon, and Y. Sheikh. Total capture: A 3d deformation model for tracking faces, hands, and bodies. In *CVPR*, 2018.

[16] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki. Harmonic coordinates for character articulation. In *TOG*, 2007.

[17] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *CVPR*, 2018.

[18] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018.

[19] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan. Geometric skinning with approximate dual quaternion blending. *TOG*, 2008.

[20] L. Kavan and J. Žára. Spherical blend skinning: a real-time deformation of articulated models. In *SI3D*, 2005.

[21] T. Kurihara and N. Miyata. Modeling deformable human hands from medical images. In *SCA*, 2004.

[22] C. Lassner, J. Romero, M. Kiefel, F. Bogo, M. J. Black, and P. V. Gehler. Unite the people: Closing the loop between 3d and 2d human representations. In *CVPR*, 2017.

[23] B. H. Le and Z. Deng. Smooth skinning decomposition with rigid bones. *TOG*, 2012.

[24] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH*, 2000.

[25] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos. Mmd gan: Towards deeper understanding of moment matching network. In *NIPS*, 2017.

[26] C.-L. Li, M. Zaheer, Y. Zhang, B. Poczos, and R. Salakhutdinov. Point cloud gan. *arXiv preprint arXiv:1810.05795*, 2018.

[27] O. Litany, T. Remez, E. Rodolà, A. M. Bronstein, and M. M. Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. In *ICCV*, 2017.

[28] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. *TOG*, 2015.

[29] N. Magnenat-Thalmann, R. Laperrire, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *GI*, 1988.

[30] X. Mao, Q. Li, H. Xie, R. Y. Lau, and Z. Wang. Least squares generative adversarial networks. In *ICCV*, 2017.

[31] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *TOG*, 2017.

[32] Y. Mroueh and T. Sercu. Fisher gan. In *NIPS*, 2017.

[33] S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *NIPS*, 2016.

[34] G. Pons-Moll, J. Romero, N. Mahmood, and M. J. Black. Dyna: A model of dynamic human shape in motion. *TOG*, 2015.

[35] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[36] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017.

[37] T. Rhee, J. P. Lewis, and U. Neumann. Real-time weighted pose-space deformation on the gpu. In *EUROGRAPHICS*, 2006.

[38] J. Romero, D. Tzionas, and M. J. Black. Embodied hands: Modeling and capturing hands and bodies together. *TOG*, 2017.

[39] A. Sinha, J. Bai, and K. Ramani. Deep learning 3d shape surfaces using geometry images. In *ECCV*, 2016.

[40] A. Sinha, A. Unmesh, Q. Huang, and K. Ramani. Surfnet: Generating 3d shape surfaces using deep residual networks. In *CVPR*, 2017.

[41] P.-P. J. Sloan, C. F. Rose III, and M. F. Cohen. Shape by example. In *SI3D*, 2001.

[42] A. Tewari, M. Zollhoefer, F. Bernard, P. Garrido, H. Kim, P. Perez, and C. Theobalt. High-fidelity monocular face reconstruction based on an unsupervised model-based face autoencoder. *TPAMI*, 2018.

[43] H.-Y. Tung, H.-W. Tung, E. Yumer, and K. Fragkiadaki. Self-supervised learning of motion capture. In *NIPS*, 2017.

[44] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid. Learning from synthetic humans. In *CVPR*, 2017.

[45] P. Wang, W. Li, Z. Gao, J. Zhang, C. Tang, and P. O. Ogunbona. Action recognition from depth maps using deep convolutional neural networks. *THMS*, 2016.

[46] X. C. Wang and C. Phillips. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *SCA*, 2002.

[47] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li. Dense human body correspondences using convolutional networks. In *CVPR*, 2016.

[48] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NIPS*, 2016.

[49] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shape modeling. In *CVPR*, 2015.

[50] Y. Yang, C. Feng, Y. Shen, and D. Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, 2018.

[51] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *NIPS*, 2017.

[52] S. Zuffi and M. J. Black. The stitched puppet: A graphical model of 3d human shape and pose. In *CVPR*, 2015.