Geometric Disentanglement for Generative Latent Shape Models: Supplementary Material

Tristan Aumentado-Armstrong*, Stavros Tsogkas, Allan Jepson, Sven Dickinson
University of Toronto Vector Institute for AI Samsung AI Center, Toronto
taumen@cs.utoronto.ca, {stavros.t,allan.jepson,s.dickinson}@samsung.com

1. Dataset Details

1.1. MNIST Dataset

As a simple dataset on which to test our method, we generate point clouds from the greyscale MNIST images. We first produce triangle meshes by placing vertices at each pixel, with the x and y coordinates normalized by image width, and the z coordinate given by 0.1 times the normalized pixel value. Edges are added between each horizontal and vertical neighbour, as well as one diagonal, thus defining 2 triangles per set of four vertex neighbours. This is essentially the mesh of the height field defined by the image of the digit. We then threshold the mesh, deleting any vertices with a z height value less than 0.01. Finally, viewing the mesh as a graph, we take the largest connected component, giving us the final triangle mesh from which we sample our point clouds. This resulted in 59483 training and 9914 testing meshes with spectra. For the MNIST dataset, we fixed $N_T = 2000$, $N_\lambda = 40$, and $N_S = 1000$.

1.2. MPI Dyna Dataset

The Dyna dataset [12] consists of 3D scans of 10 individuals performing various sequences of simple actions (e.g., holding up their arms). In total, the dataset contains approximately 40K triangle meshes of multiple body types in a large variety of poses. During training, we fixed the total number of samples to $N_T = 6000$, the spectrum length as $N_\lambda = 100$, and the size of the input point clouds as $N_S = 2000$ for the Dyna dataset.

1.3. SMAL-derived Dataset

Using the SMAL model [15], we generated a dataset of animal meshes, including random body types and articulated motions. In detail, we use the fitted multivariate Gaussians computed by the authors of SMAL, which each act as a distribution over clusters of the shape parameters of the same animal species. We generate 3200 meshes for each of the five categories by sampling from each cluster distribution. Following the generation procedure in other works [4], we then sample the pose (here, the joint angles) via a Gaussian distribution with a standard deviation of 0.2. We then split the resulting dataset into 15000 training and 1000 testing meshes (each comprised of equal numbers of meshes per species). For SMAL, the total number of samples was $N_T = 8000$, the spectrum length was $N_\lambda = 50$, and the size of the input point clouds during training was $N_S = 1500$.

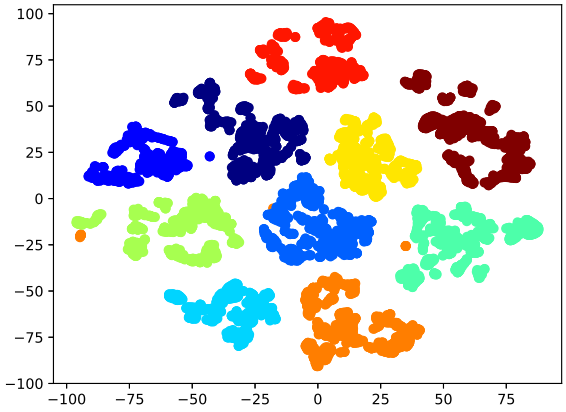

Figure 1. A t-SNE plot of the LBO spectra of the human shapes in the Dyna dataset. Each color corresponds to a different individual. Note the natural clusters formed by individuals.

1.4. SMPL-derived Dataset

Similar to the dataset derived from SMAL, we generate a dataset of human meshes with random body types and articulations via the SMPL model [8]. We largely follow the protocol from 3D-CODED [4]. Briefly, we sampled 20500 meshes from each of the male and female models using random samples from the SURREAL dataset [14]. We augmented this data with 3100 meshes of "bent" humans per

*The work in this article was done while Tristan A.A. was a student at the University of Toronto. Sven Dickinson, Allan Jepson, and Stavros Tsogkas contributed in their capacity as Professors and Postdoc at the University of Toronto, respectively. The views expressed (or the conclusions reached) are their own and do not necessarily represent the views of Samsung Research America, Inc.

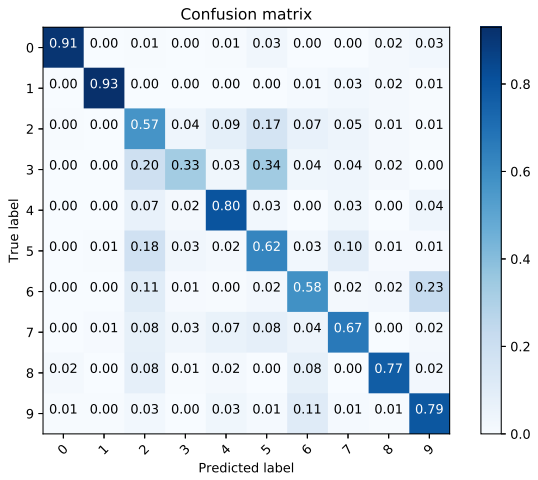

Figure 2. Confusion matrix of a shallow classifier, mapping mesh spectra to digit identity. Note the main confusions, e.g., between 9 and 6, match our intuition for the shape of the digit geometrically.

gender, using the alterations from Groueix et al [4]. We then assigned 500 unbent and 100 bent meshes (per gender) to the held-out test set, and the remaining meshes to the training set. This resulted in 45992 meshes and 1199 testing meshes after spectral calculations. Using these meshes, we derived point clouds with $N_T = 8000$, $N_\lambda = 40$, and $N_S = 2000$.

2. Spectral Geometry Intuition

In this section, we provide some intuition for the geometric meaning of the spectrum and why it can be used as a geometrically disentangled prior for shape representation, particularly for shapes that undergo isometric articulated pose transforms.

For MNIST, an obvious question is how the intrinsic geometry of a digit's shape captured by its LBO spectrum is related to its semantic label (i.e., numeric value). Intuitively, a natural notion of the shapes of the digits should be closely related to their numeric identity, while minor perturbations that change the style of the digits should have less of an effect on the intrinsic shape. We examine this relation by training a simple classifier, which learns to map from the spectrum vector to the digit identity. We use a neural network with one hidden layer (size 100), using the ReLU nonlinearity and otherwise default parameters from the scikit-learn library [11]. This obtains an accuracy of 0.69, suggesting that the intrinsic digit shape alone is capable of significant discriminatory power, but not as much as the complete shape information. The confusion matrix is shown in Figure 2. The observed results agree with our intuition: for instance, 9 and 6 are close to rotations of each other, while 2 and 5 also clearly have very similar intrinsic shapes; unsurprisingly, these are relatively more common misclassifications. The most misclassified shapes, however, are 3 and 5, which differ only in the placement of the "bridge" between the lower curved part and the upper line making up the digits.

For datasets of some objects, such as people, articulations (i.e., non-rigid changes in pose for a single person) are nearly isometric transformations. Just as one naturally divides rigid and non-rigid deformations of objects, so too can one separate isometries from non-isometric deformations. To illustrate, in Figure 1, a t-Distributed Stochastic Neighbor Embedding (t-SNE) [9] plot (via scikit-learn [11]) shows the embeddings of the spectra of a portion of the Dyna dataset (every sixth shape in each activity sequence). Notice that the embedding naturally clusters by individual, as the intrinsic shape of each individual's body is largely unchanged by the pose articulations of each action sequence. This corresponds to an intuitive view of intrinsic shape as being both continuous (as opposed to labelling each individual) and, in this case, articulation invariant.

3. Architectures

All models were implemented in PyTorch [10]. For exact values for each dataset, see Table 1.

3.1. Autoencoder Architecture

The encoder consists of a PointNet model [13], followed by fully-connected (FC) layers. The convolutional layers were of sizes $S_{\text{AE,C}}$. There were two affine transformers ("T-networks") after the first and third layers, with convolutional layers of size 64, 128, and 512, followed by an FC layer with hidden size 256. After the convolutional layers and affine transforms, 1D max pooling was applied, followed by processing by a FC network with hidden layers of sizes $S_{\text{AE,FE}}$. The final latent representation of the AE was of dimensionality L_{AE}. The decoder directly generated a fixed size point cloud (with $N_{\text{AE,OUT}}$ points), via a series of FC layers with hidden layer sizes $S_{\text{AE,FD}}$

We fixed the loss weighting parameters as follows: $\alpha_C = 0.75, \alpha_H = 0.5$. For MNIST and Dyna, FC and 1D convolutional layers utilized a BatchNorm layer [6] and the ReLU non-linearity between linear transforms. For SMPL and SMAL, no normalization and layer normalization [2] were used, respectively, in the decoder (similar to the encoder-only usage of batch norm in prior work [1]). Data augmentation consisted of random rotations about the "height" axis of the models (e.g., see Figure 5), with rotations in $[0, 2\pi]$ for MNIST and Dyna and in $[-\pi/6, \pi/6]$ for SMPL and SMAL. A batch-size of S_B and a learning rate of $\eta_{\text{AE,LR}}$ was used, with Adam [7] as the optimizer, for N_{EP} epochs.

		MNIST	Dyna	SMAL	SMPL
AE	$S_{\text{AE,C}}$	50, 100, 200, 300, 400	50, 100, 200, 300, 400	50, 100, 200, 400, 500	50, 100, 200, 400, 500
	$S_{\text{AE,FE}}$	300	300	800	500
	L_{AE}	250	250	400	300
	$S_{\text{AE,FD}}$	300, 500, 800	300, 500, 800	800, 800, 2000	500, 800, 2000
	r_C, r_H	10, 1	10, 1	20, 0.1	20, 0.1
	$N_{\text{AE,OUT}}$	1000	2000	1500	2000
	S_B, N_{EP}	32, 100	32, 200	30, 500	20, 100
	$\eta_{\text{AE,LR}}$	0.00005	0.00005	0.00025	0.00025
VAE	S_λ	500, 400	500, 400	300, 300	300, 400
	$S_{\text{V,B}}$	200	200	250	250
	$N_{\text{V,EP}}$	150	150	200	100
	$\lvert z_E\rvert, \lvert z_I\rvert$	5, 5	10, 10	8, 5	12, 5
	β_4, γ_I, w_J	50, 1, 1	25, 5, 5	50, 100, 10	50, 10, 10

Table 1. Architectural parameters for the models across different datasets. See Sections 3.1 and 3.2 for details concerning the AE and VAE parameters, respectively.

3.2. GDVAE Architecture

The encoder consists of two parts: one for the rotation R and one for the shape X. Each part consists of two sub-parts: a set of shared layers, followed by a separate network for the variational parameters μ and Σ (i.e., $z_R \sim \mathcal{N}(\mu(g_{\text{shared,r}}(R)), \Sigma(g_{\text{shared,r}}(R)))$ and $(z_E, z_I) \sim \mathcal{N}(\mu(g_{\text{shared,x}}(X)), \Sigma(g_{\text{shared,x}}(X)))$). For the rotation mapping, the shared layers consisted of FC layers with hidden sizes $300, 200$, while the mean and variance mappings are each given by a single affine transform. For the X mapping, the shared layers consisted of FC layers with hidden sizes 1000, 750, and 500, while the mean and variance are parameterized by a FC network with one hidden layer of size 250 (MNIST/Dyna) or 300 (SMPL/SMAL).

The decoder also consisted of two parts: the rotation decoder (FC layers with sizes 200 and 300) and the shape decoder (FC layers of sizes 500, 750, and 1000). The spectral predictor defined on the latent intrinsic z_I space was an FC network with layer sizes S_λ.

Models used a batch-size of $S_{\text{V,B}}$ for $N_{\text{V,EP}}$ epochs, using Adam with a learning rate of 0.0001. A small L_2 weight decay with coefficient 5×10^{-6} was used for MNIST and Dyna. Data was again augmented by random rotations about the height axis, but limited to an angular magnitude within $[-\pi/5, \pi/5]$ for MNIST and Dyna, and $[-\pi/12, \pi/12]$ for SMPL and SMAL.

Concerning hyper-parameters, unless otherwise specified, we used a spectral weight of $\zeta = 1000$, a relative quaternionic loss weight of $w_Q = 10$, hierarchically factorized VAE loss coefficients of $\beta_1 = \beta_2 = \beta_3 = 1.0$, and a reconstruction loss weight of $\eta = \dim(X)$ (on MNIST and Dyna), $\eta = 200$ (on SMPL), and $\eta = 1$ (on SMAL). On SMPL, we used $\zeta = 500$, however. Recall that the weights on the disentanglement penalties were written β_4 for the inter-group TC, γ_I for the inter-group covariance, and w_J for the Jacobian.

	Beta-VAE	Jacobian	Cov	TC	All
β_4	1	1	1	100	100
γ_I	0	0	100	0	100
w_J	0	100	0	0	100

Table 2. Hyper-parameter values varied in the test of various loss weightings. (See Figure 4 for visualization of results).

4. Rotation Disentanglement

We examine the performance of the deterministic autoencoder, visualizing a number of reconstructions in Figure 5. Qualitatively, the points sampled over the reconstructions are largely uniformly spread out. However, thin or protruding areas tend to have lower densities of points, an issue identified by other works [1].

One question is how the presence of the rotation quaternion affects the representation. We visualize this by "derotating" the shapes (shown in the last row of each set of shapes in Figure 5), where the R portion of their representation is set to the same value. We can see that the derotated shapes tend to approximately fall into two or three groups with similar orientation. To confirm this, we also sample random shapes, rotate them, and then embed their X encodings via t-SNE in Figure 6. Qualitatively, we can see that rotating the point sets does not appear to lead to a single representation X; instead, it forms a small number of latent groups.

We also tried adding a "rotational consistency" penalty to the loss function, which was computed by placing an L_2 loss between the X representation of pairs of shapes that differ only by a rigid rotation (i.e., $||X - X_r||_2^2$, where X and X_r are generated by rotations of the same point cloud). However, this seemed to lead to higher reconstruction error and did not entirely prevent the differing representations

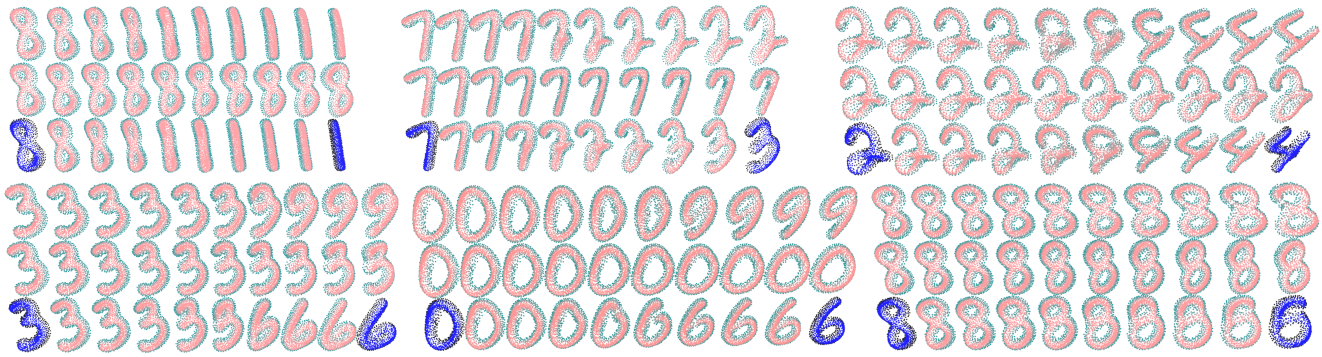

Figure 3. Disentangled latent interpolations for the MNIST dataset. Colours per digit denote depth. For each inset, we are interpolating between the blue-black digits in the bottom row. The first row corresponds to only moving through z_I (with z_E constant), while the second corresponds to only moving through z_E (holding z_I fixed). The third row traverses the full latent z space. In all cases, z_R is set to zero.

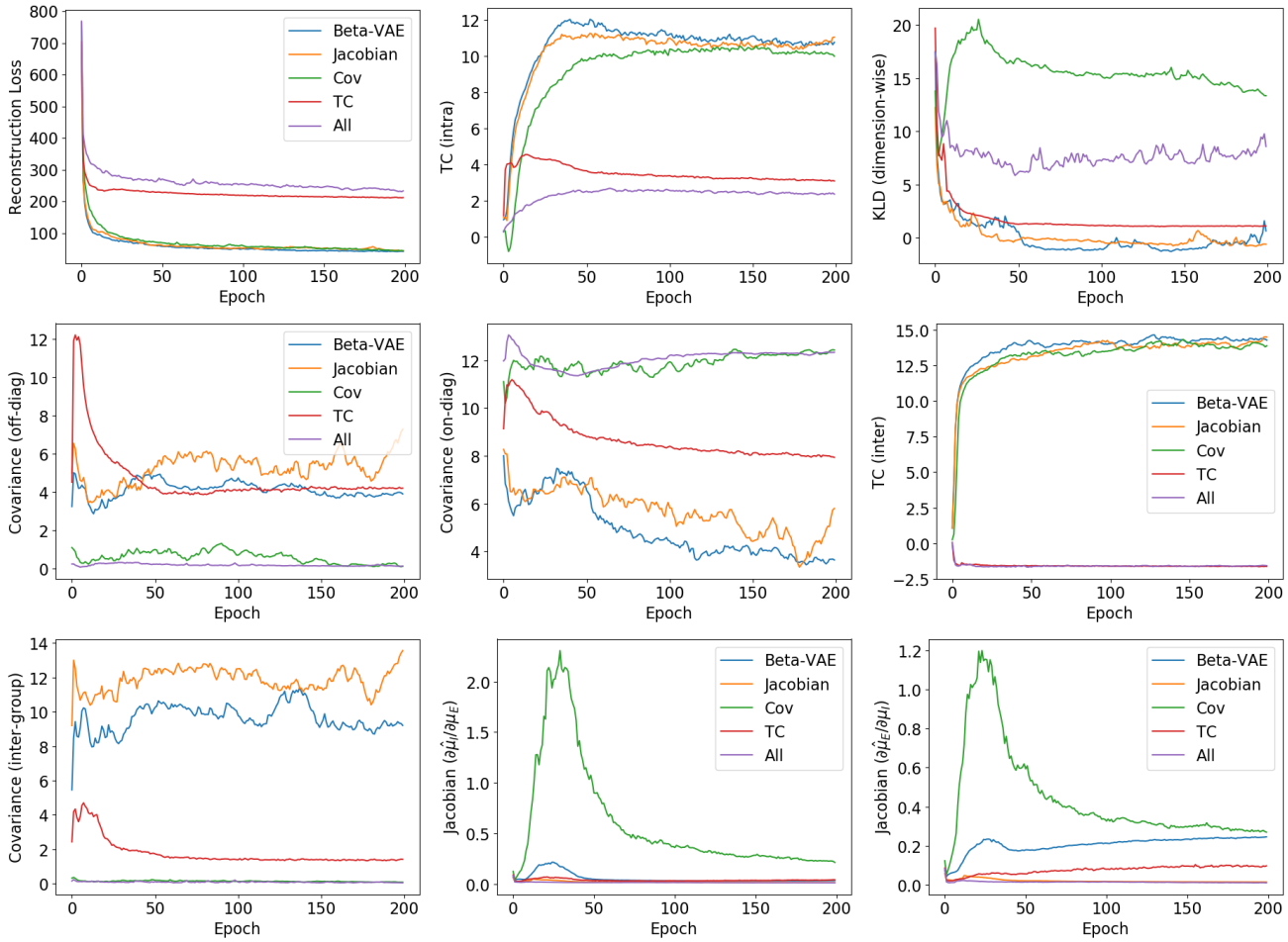

Figure 4. Empirical curves of loss terms during training across weight hyper-parameters on model learning. See Table 2 for weight values. Each colored curve represents a different set of weight values (i.e., model hyper-parameters). Top row: log-likelihood reconstruction loss, intra-group TC, dimension-wise KL divergence. Middle row: off-diagonal intra-group covariance, on-diagonal covariance terms (i.e., variances), inter-group TC. Bottom row: inter-group covariance, Jacobian penalty (intrinsics with respect to extrinsics), Jacobian penalty (extrinsics with respect to intrinsics).

forming across rotations. Furthermore, as noted in the main text, some shapes can naturally become a "new" shape, semantically speaking, due to rotation (e.g., a 9 becoming a 6), which this penalty does not allow.

Figure 5. Examples of autoencoder reconstructions across different rotations of the same object. Note that the encoding of the datum from the AE is not passed through the VAE. Colour denotes depth (z value). Within each group of shapes, the columns show different rotations of the left-most shape, while the top row shows input shapes (i.e., P, from the original data), the middle row shows the reconstruction $\hat{P}$, and the bottom row shows the derotated shape with the rotation component R set to the value given by the shape in the first column. Note that these models were trained with data augmentations across all rotations about the gravity axis.

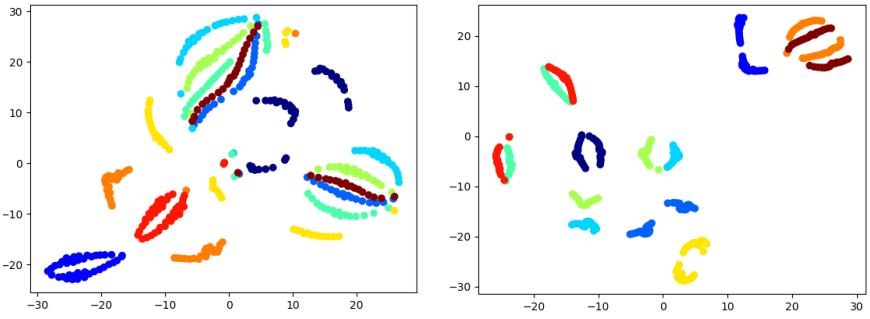

Figure 6. A t-SNE visualization of the rotated shapes from the deterministic AE. Left: plot of MNIST. Right: plot of Dyna. In both plots, 10 random shapes are selected, and rotated 40 times, evenly spread over $[0, 2\pi]$, with respect to the height axis. Each shape is then encoded by the AE, and only the X component of its representation is plotted. A single color corresponds to a single shape.

5. MNIST Disentanglement

5.1. MNIST Latent Interpolations

Compared to that of articulating shapes (e.g., humans or other animal), the geometric disentanglement of the latent MNIST digit representation is less intuitive. Often, however, we found that moving in z_E would tend to deform the digit in "style", while tranversing z_I would more affect digit scale/thickness and identity. Some examples are shown in Figure 3. For instance, in the first inset, moving in z_E simply shifts around the lines of the '8' without changing its digit identity (deforming it stylistically), while moving in

		X	z	z_E	z_I
SMAL	E_β	0.641	0.743	0.975	0.645
	E_θ	0.938	0.983	0.983	0.993
SMAL-NJ	E_β	0.642	0.829	0.980	0.734
	E_θ	0.938	0.979	0.980	0.996
SMAL-NC	E_β	0.642	0.670	0.962	0.656
	E_θ	0.938	0.966	0.969	0.991
SMAL-NJC	E_β	0.642	0.661	0.834	0.891
	E_θ	0.938	0.967	0.978	0.982
SMPL	E_β	0.856	0.922	0.997	0.928
	E_θ	0.577	0.726	0.709	0.947
SMPL-NJ	E_β	0.858	0.907	1.006	0.895
	E_θ	0.578	0.695	0.812	0.908
SMPL-NC	E_β	0.855	0.908	0.995	0.905
	E_θ	0.578	0.727	0.836	0.909
SMPL-NJC	E_β	0.857	0.888	0.992	0.921
	E_θ	0.578	0.693	0.722	0.965

Table 3. Different retrieval scores under various disentanglement penalty ablation conditions. NJ, NC, and NJC mean no Jacobian, no covariance, and neither Jacobian nor covariance cases respectively. Note that differences between scores under X are due to the random samplings of points from the shape; hence, each score is obtained by running the process three times (across point samplings). However, only the scores for SMAL and SMPL are averaged over multiple training runs.

z_I horizontally squishes the '8' into a '1'.

5.2. MNIST Classification Results

We considered the information stored in the disentangled latent space segments by examining the performance of a classifier trained on various combinations of them, as representations of the digits. We show the confusion matrices for a linear SVM classifier on these sub-components of the latent spaces in Figure 7. Notice that utilizing z_R only has the poorest performance (and that adding it to z_{EI} has little effect), and that the most prominent mistakes when using only z_I are very similar to those in Figure 2 (e.g., mixing up 2, 3, 4; and 6, 9).

6. Model Analysis

6.1. Hyper-parameter Variation

Our generative model must balance three main terms: (1) autoencoding reconstruction, (2) latent prior sampling, and (3) disentanglement. We therefore trained five models on MNIST under five different hyper-parameter conditions (see Table 2) to showcase the relative effect of these loss weights. The choice of models was designed to check how the different penalties affected the metrics of disentanglement based on each loss (e.g., how penalizing covariance affects the pairwise Jacobian penalty), as well as autoencoding and generation.

Based on the results (loss curves are shown in Figure 4), we can make several observations. Firstly, a high TC penalty results in decreased inter-group covariance and Jacobian losses. However, while the covariance and Jacobian penalties effectively reduce their own penalties, they struggle to reduce the TC or each other. Nevertheless, the TC penalty alone does not drive the covariance or Jacobian values as low as having a penalty on them directly does. This suggests that the TC may be a more powerful penalty, but it can still be complemented by the other approaches. One explanation for this effect is that the Jacobian penalty is fundamentally local (penalizing the expected change with respect to an infinitesimal perturbation around each data point separately) and the covariance penalty only reduces linear correlations, whereas the TC penalty is information-theoretic (i.e., able to detect non-linear relations) and considers the estimated latent probability distributions on a more global level.

On the other hand, the models with high TC show the worst log-likelihood for reconstructions. The model with high weights for all terms also has poor dimension-wise KL divergence, meaning the ability to generate novel samples may be compromised, though it is not as high as that of the covariance-penalized model. Though our observations are limited to this dataset, hyper-parameters, and architecture, they suggest that the disentanglement term can be in conflict with reconstruction and sampling (just as the latter two are known to be in conflict with each other; e.g., [5]).

6.2. Disentanglement Penalty Ablation

To further examine the effect of the disentanglement penalties, in particular the covariance and Jacobian terms, we considered two ablation experiments on the SMPL and SMAL datasets: (1) looking at the loss curves of the estimated entanglement measures during training and (2) considering the pose-aware retrieval performance. We examined three conditions, in addition to the regular hyper-parameter values (REG): no Jacobian penalty (NJ), no covariance penalty (NC), and no Jacobian or covariance penalties (NJC).

The various loss terms are shown over training epochs in Figures 8 and 9, for SMAL and SMPL respectively. On SMAL, we see that REG and NJ have worse KL-divergence and reconstruction loss, but better inter-group TC and covariance. NJC scores the worst on all four entanglement measures, except $\partial\hat{\mu}_I/\partial\mu_E$ (where surprisingly NJ holds the smallest value). On SMPL, we see that NJ consistently has the highest Jacobian penalties, and that REG and NJ have the worst TC. On both datasets, we see that the covariance penalty is necessary to ensure the inter-group covariance is small, since NC and NJC always have much higher inter-group covariance than NJ and REG.

We next considered the pose-aware retrieval task under

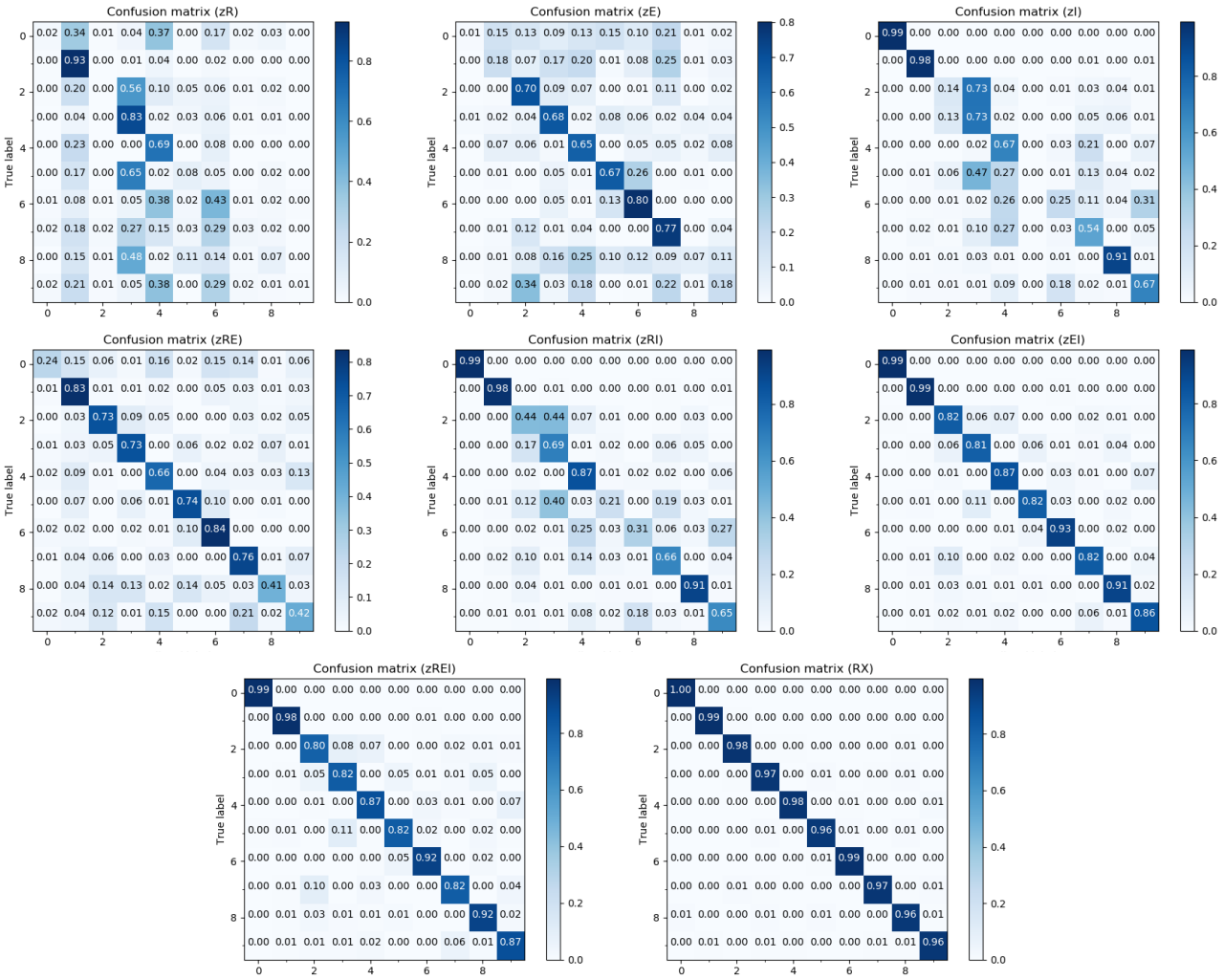

Figure 7. Confusion matrices for a linear SVM classifier accuracy on the MNIST test set. Top row: testing with z_R, z_E, z_I. Middle row: testing with z_{RE}, z_{RI}, z_{EI}. Bottom row: testing with, z_{REI}, (R, X).

the various ablation conditions. Results are shown in Table 3. Recall that low E_θ using z_E and low E_β using z_I are good (indicating z_I and z_E hold intrinsic shape and pose respectively), while low E_θ using z_I and low E_β using z_E are not (as it means entanglement is present). For SMAL, we see that REG has the best E_β using z_I, while NJC is worse than both NJ and NC. We can also see poor entanglement in the low E_β using z_E of NJC. The scores for E_θ are poor across all the representations, even including X. NC has the best E_θ using z_E, but it has a worse (i.e., lower) E_β using z_E, suggesting increased entanglement. For SMPL, REG has the best E_θ using z_E, but NC and NJ have better E_β values with z_I. However, NC and NJ also have lower E_θ with z_I, meaning pose is entangled with intrinsic shape. It's also worth noting that NJC on both SMAL and SMPL has lower error on both E_β and E_θ when using z. This suggests that the increased disentanglement penalties make holding information harder in general (i.e., allowing more entanglement can increase reconstruction and thus retrieval performance).

7. Pose-Aware Retrieval Variance

In Table 4, we consider the standard error of the mean (SEM) for the retrieval experiments. Each model was trained three times with the same hyper-parameters (to account for training stochasticity), and then each model was run three times to account for randomness in the point set sampling of the input shapes. We show the SEM across models (i.e., trainings) after averaging over point samplings per model. Since each model has its own SEM over samplings, we display the maximum SEM across models. Notice that all SEMs are less than 0.01, except for two (over model trainings): the E_β when using z and z_I on SMAL, which are the most unstable retrieval performances. In gen-

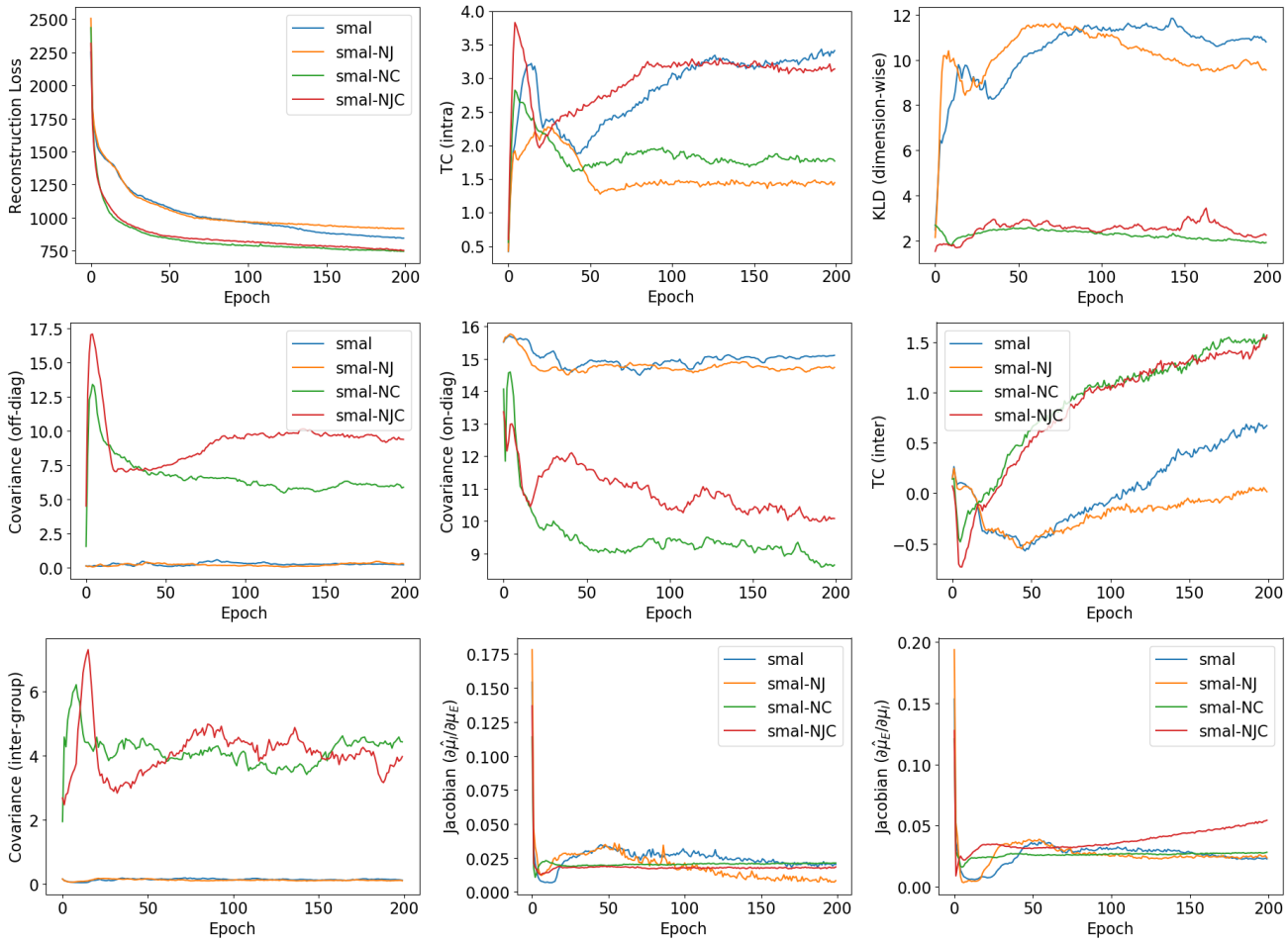

Figure 8. Empirical curves of loss terms for disentanglement ablations on SMAL. NC, NJ, and NJC mean no covariance, no Jacobian, or neither penalties, respectively. Each colored curve represents a different set of weight values (i.e., model hyper-parameters). Top row: log-likelihood reconstruction loss, intra-group TC, dimension-wise KL divergence. Middle row: off-diagonal intra-group covariance, on-diagonal covariance terms (i.e., variances), inter-group TC. Bottom row: inter-group covariance, Jacobian penalty (intrinsics with respect to extrinsics), Jacobian penalty (extrinsics with respect to intrinsics).

		X	z	z_E	z_I
SMAL	E_β	0.0006	0.0282	0.0035	0.0381
(M-SEM)	E_θ	0.0003	0.0015	0.0015	0.0015
SMPL	E_β	0.0003	0.0048	0.0015	0.0032
(M-SEM)	E_θ	0.0003	0.0032	0.0058	0.0059
SMAL	E_β	0.0028	0.0020	0.0030	0.0024
(S-SEM)	E_θ	0.0008	0.0005	0.0006	0.0008
SMPL	E_β	0.0015	0.0013	0.0020	0.0053
(S-SEM)	E_θ	0.0008	0.0017	0.0022	0.0070

Table 4. SEMs across model training runs (M-SEM) and shape samplings (S-SEM). Model training SEMs are computed over the mean of the shape sampling runs; shape sampling SEMs are computed by taking the SEM over point samplings per model, and then the maximum SEM across models.

eral, training instability is a useful consideration for future work.

8. Pose-Aware Retrieval with Spectral Noise

We also investigated the effect of spectral noise on model performance. We considered three forms of noise: (1) by directly injecting Gaussian multiplicative noise into the spectra, (2) replacing the spectrum with Gaussian noise (mean zero, sigma 50), and (3) by extracting the LBO from sampled point clouds (rather than meshes). To add the noise in (1), we multiply each eigenvalue with independent noise $\xi \sim \mathcal{N}(1, \sigma^2)$, and then enforced non-negativity (via clipping) and monotonicity of the spectrum (by re-sorting the spectrum). To obtain the spectra in (3), we used a simple Laplacian computed from an affinity matrix, via a radial basis function kernel on the inter-point L_2 distance, with bandwidth chosen as $b_\sigma = d_N^{-1/(2+\xi)}/4$, where d_N is the mean distance of each point to its nearest neighbour and $\xi = 0.01$ (similar to [3]).

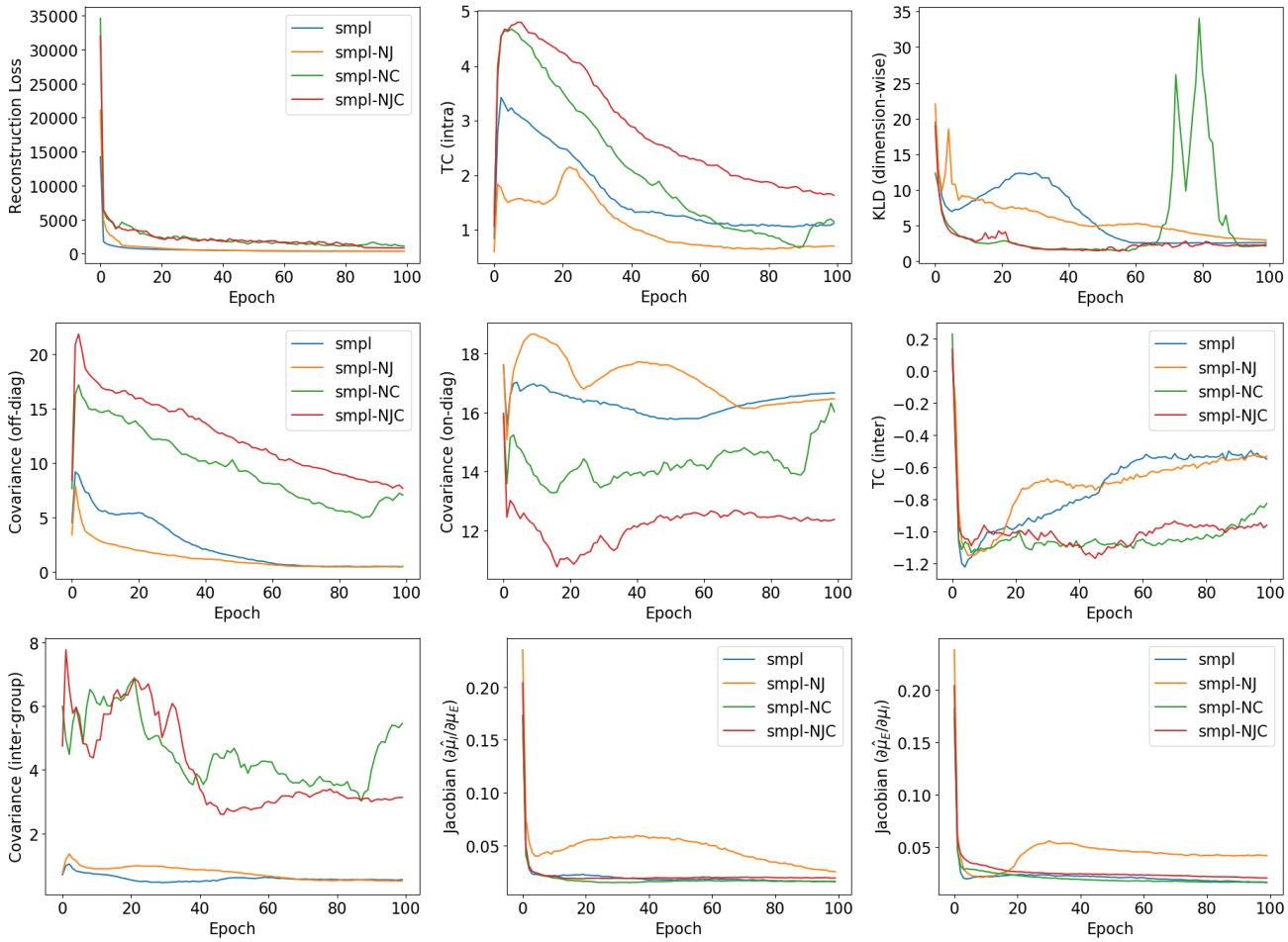

Figure 9. Empirical curves of loss terms for disentanglement ablations on SMPL. See Table 2 for weight values. NC, NJ, and NJC mean no covariance, no Jacobian, or neither penalties, respectively. Top row: log-likelihood reconstruction loss, intra-group TC, dimension-wise KL divergence. Middle row: off-diagonal intra-group covariance, on-diagonal covariance terms (i.e., variances), inter-group TC. Bottom row: inter-group covariance, Jacobian penalty (intrinsics with respect to extrinsics), Jacobian penalty (extrinsics with respect to intrinsics).

We note that the shapes (input point clouds) themselves do not have any additional noise (compared to the standard experimental setup), only the spectra do. Thus, poorer performance may manifest itself as increased retrieval accuracy with the *wrong* latent space segment (e.g., lower E_β when retrieving with z_E).

Results are shown in Table 5. However, we find that the network can tolerate moderate spectral noise, with decreasing performance as noise increases. For instance, one can see E_β with z_I on SMAL and E_θ with z_E on SMPL degrade as σ increases. Very extreme noise, as when replacing the spectrum with Gaussian random values ("SMAL-R" and "SMPL-R" in Table 5), destroys the disentanglement, as the network no longer has access to the isolated intrinsics. This is similar to ablating the spectral loss, except that predicting the random spectrum adds additional burden on z_I. Finally, for the spectra extracted from the point cloud Laplacians, the network degrades slightly on SMPL, compared to using the mesh LBO, but much more so on SMAL.

In practice, we note that our method does not require spectra at test time (for inference or novel sample generation). However, it will be affected by noise in the training data (whether in the meshes, spectra, or point clouds). One approach to improve generalization to noisy point clouds at test time is to use additional noise for data augmentation while training.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. 2018. 2, 3

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 2

[3] Mikhail Belkin, Jian Sun, and Yusu Wang. Constructing laplace operator from point clouds in $\mathbb{R}^d$. In *Proceedings*

		X	z	z_E	z_I
SMAL	E_β	0.641	0.724	0.961	0.666
$\sigma = 0.05$	E_θ	0.937	0.969	0.980	0.981
SMAL	E_β	0.641	0.723	0.900	0.868
$\sigma = 0.1$	E_θ	0.938	0.969	0.979	0.994
SMAL	E_β	0.636	0.810	0.893	0.838
$\sigma = 0.2$	E_θ	0.938	0.979	0.988	0.996
SMAL-R	E_β	0.643	0.822	0.791	0.913
	E_θ	0.938	0.977	0.985	0.998
SMAL-P	E_β	0.639	0.629	0.783	0.929
$n_p = 1.2$K	E_θ	0.939	0.973	0.973	0.995
SMAL-P	E_β	0.641	0.698	0.636	0.895
$n_p = 2$K	E_θ	0.938	0.980	0.984	0.993
SMPL	E_β	0.857	0.910	1.000	0.913
$\sigma = 0.05$	E_θ	0.578	0.735	0.774	0.966
SMPL	E_β	0.856	0.909	0.979	0.929
$\sigma = 0.1$	E_θ	0.578	0.694	0.720	0.979
SMPL	E_β	0.858	0.919	0.980	0.925
$\sigma = 0.2$	E_θ	0.578	0.710	0.810	0.946
SMPL-R	E_β	0.857	0.933	0.924	0.987
	E_θ	0.579	0.694	0.826	0.943
SMPL-P	E_β	0.856	0.946	0.972	0.948
$n_p = 1.2$K	E_θ	0.577	0.673	0.731	0.954
SMPL-P	E_β	0.856	0.926	0.990	0.944
$n_p = 2$K	E_θ	0.578	0.695	0.739	0.982

Table 5. Retrieval results in the presence of spectral noise. Each row corresponds to retrieval results in the presence of spectral noise either due to multiplicative Gaussian noise (with strength σ), replacing the spectrum with independent Gaussian noise (mean zero, sigma 50; denoted "-R"), or using an LBO estimated from a point cloud (denoted "-P", using a point cloud of size n_p, either 1200 or 2000).

of the twentieth annual ACM-SIAM symposium on Discrete algorithms, pages 1031–1040. Society for Industrial and Applied Mathematics, 2009. 8

[4] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. 3d-coded: 3d correspondences by deep deformation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 230–246, 2018. 1, 2

[5] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. 6

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 2

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[8] 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):248:1–248:16, Oct. 2015. 1

[9] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 2

[10] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 2

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 2

[12] Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J. Black. Dyna: A model of dynamic human shape in motion. *ACM Transactions on Graphics, (Proc. SIGGRAPH)*, 34(4):120:1–120:14, Aug. 2015. 1

[13] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017. 2

[14] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *CVPR*, 2017. 1

[15] Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1