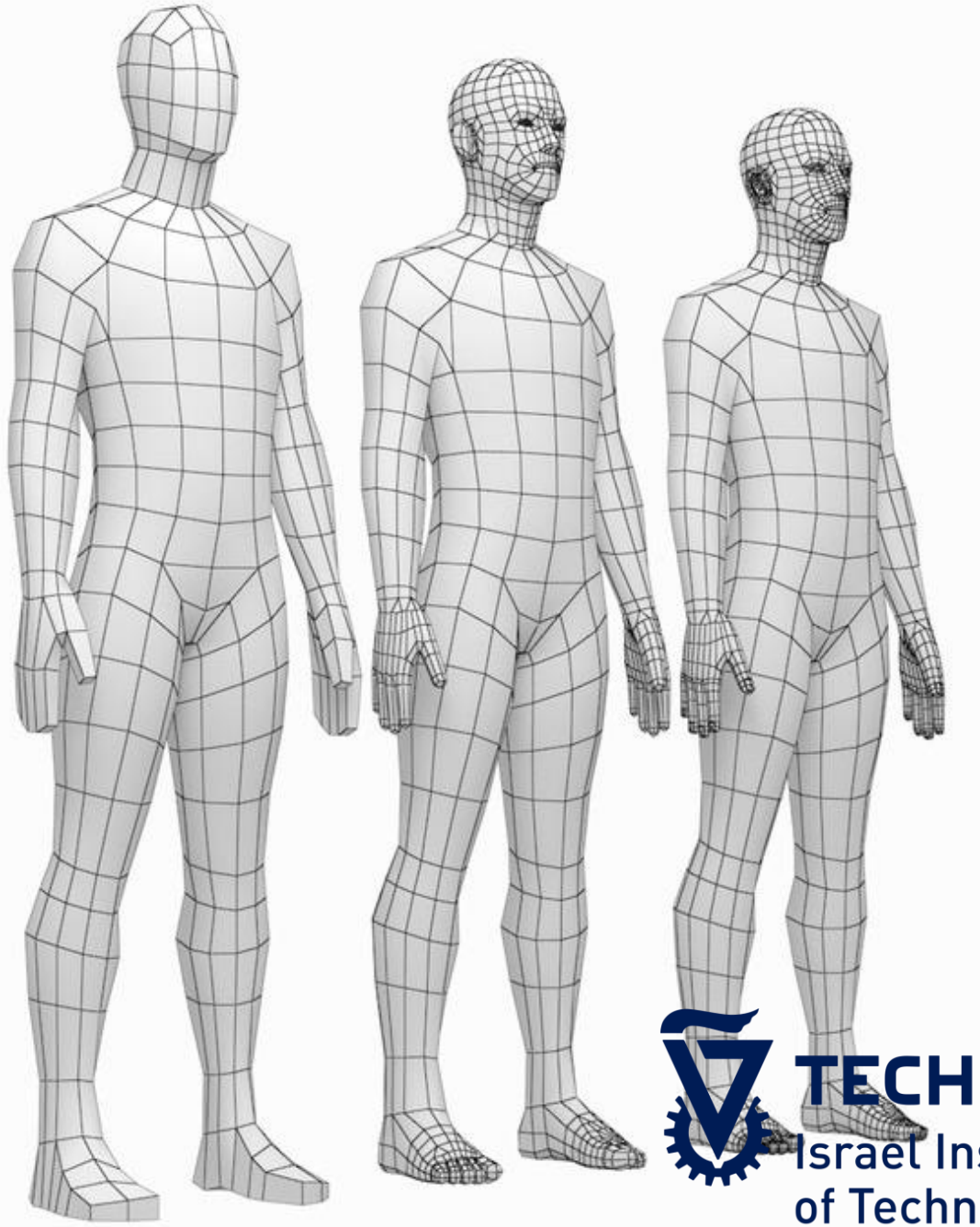


Learning Unique Invariant Signatures of Non-Rigid Point Clouds

Authors: Sari Hleihil & Idan Shenfield

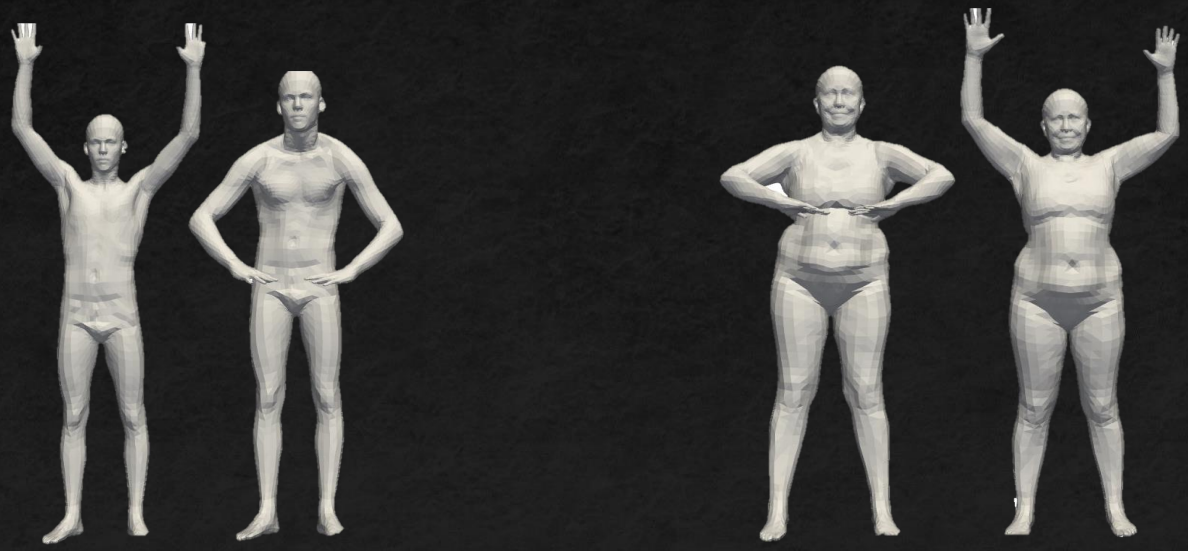
Supervisor: Ido Imanuel

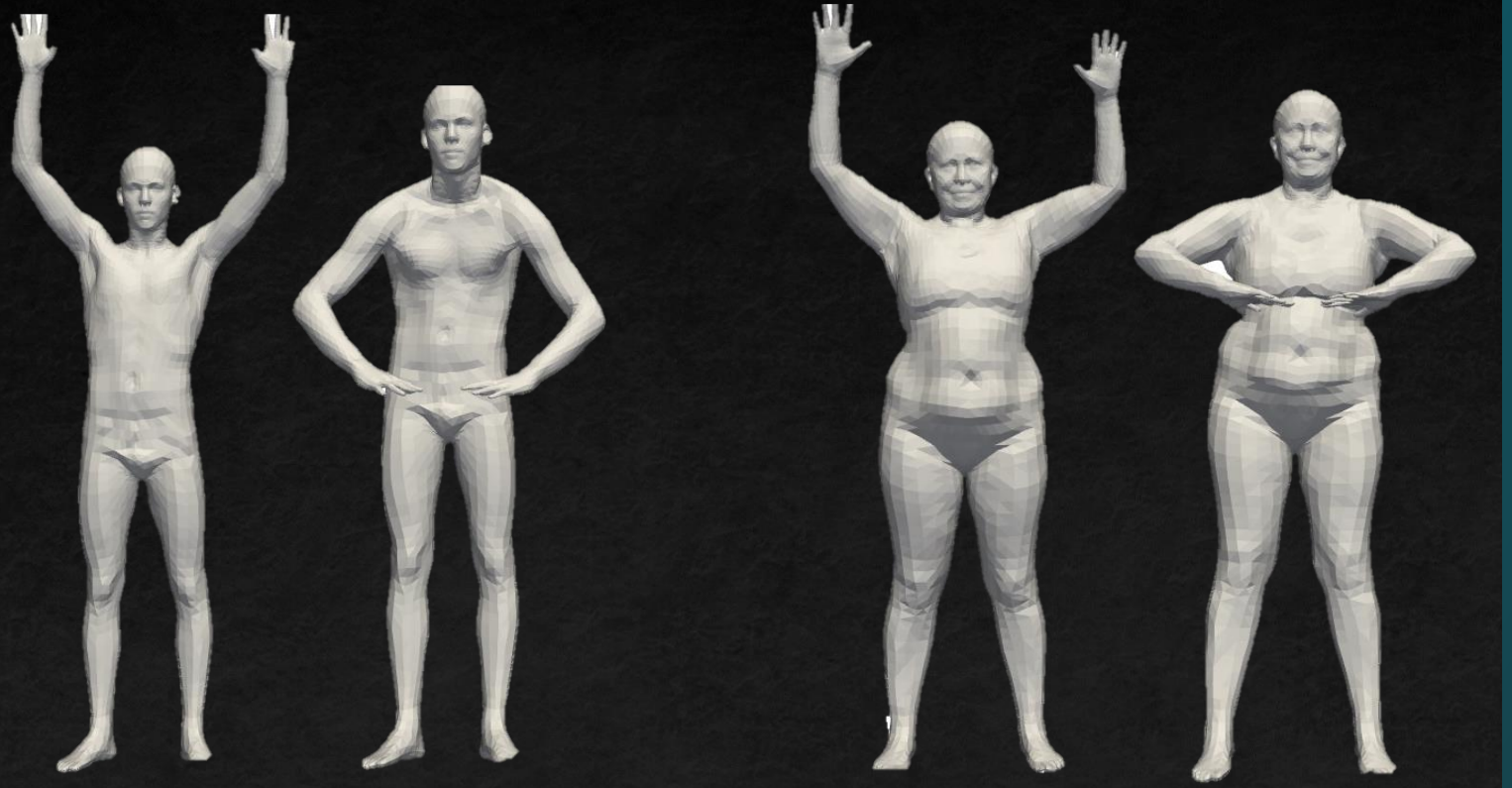
TECHNION
Israel Institute of Technology

Presentation Outline

Motivation

- Human beings perceive objects independently of their pose, can machines do that too?





Motivation

Intro - Abstract

- We propose a metric learning framework for the construction of invariant signatures of non-rigid 3D point clouds under the isometry transformations group.
- We achieve results superior to the SOTA, by achieving descriptors that are more pose invariant yet more descriptive.
- Our method is more efficient that the SOTA, thus allowing for computations on larger point clouds.

Prior Work

Learning Invariant Representations of Planar Curves - Gautam Pai et al.

Using CNNs to learn geometric invariants under various transformation groups on planar curves. Successfully learned the gaussian curvature of curves.

MDS

A classical algorithm that attempts to embed a non-Euclidian high dimensional metric space into a Euclidean space of a lower dimension while preserving the metric.

Geodesic Distance Descriptors – Shamai et al.

Approximates a truncated basis for the geodesic distance functions of the metric space and uses it to embed the shape into a lower dimension.

Preserves the information encoded in the metric indirectly

Learning Invariant Representations Of Planar Curves

- The goal of this work is to learn an invariant to the set of Euclidean transformations on planar curves.

- This is done using a Siamese learning scheme and a contrastive loss:

$$\mathcal{L}(W_1, W_2, Y) = (1 - Y)\frac{1}{2}D_{W_{12}}^2 + Y\frac{1}{2}\max\left(0, \mu - D_{W_{12}}\right)^2$$

- They use resampling techniques and smoothing in order to achieve re-parametrization and scale invariance.
- They show that they were able to learn a signature strongly correlated to the Euclidean curvature.

MDS

- Definition: let $\{x_i\}_{i=1}^n \subseteq \mathbb{R}^3$ be the vertices of a mesh, and $\forall i,j \in [n]$ $\delta_{ij} = d_{geo}(x_i, x_j)$, then let $\{X_i\}_{i=1}^n \subseteq \mathbb{R}^k$ be a mapping of $\{x_i\}_{i=1}^n$, the stress is defined as:

$$Stress(\{x_i\}_{i=1}^n) = \frac{\sum_{i<j} \left(\delta_{ij} - d(X_i, X_j)\right)^2}{\sum_{i<j} \delta_{ij}^2}$$

- The stress is could be thought of as the total energy of a spring system.
- Goal: minimize Stress!

MDS

- Objective (reminder): min $\sum_{i<j}\left(\delta_{ij}-d(X_i,X_j)\right)^2$
- Definition: $D_{sr}\coloneqq\delta_{sr}^2$, $\widetilde{D}_{sr}=d(X_s,X_r)=\|X_r-X_s\|_2^2$.
- Definition: B $\coloneqq JDJ, \tilde{B}_{sr}\coloneqq X_s^T X_r=\left(\underline{X}^T\underline{X}\right)_{sr}$.
- W.l.o.g $\sum_{i=1}^{n}X_i=\vec{0}$, thus it could be proven that $\tilde{B}=\frac{1}{2}J\widetilde{D}J$ where $J\coloneqq I-\frac{1}{n}1\,1^T$ and thus the objective could be written as:

$$\underset{\widetilde{D}}{\operatorname{argmin}}\left\{\left\|D-\widetilde{D}\right\|_F^2\right\}=\underset{\widetilde{D}}{\operatorname{argmin}}\left\{\left\|J\left(D-\widetilde{D}\right)J\right\|_F^2\right\}=\underset{\widetilde{D}}{\operatorname{argmin}}\left\{\left\|B-\tilde{B}\right\|_F^2\right\}$$

- It is known that this problem is solved by a truncated SVD approximation, thus using spectral decomposition: $B=V\Lambda V^T$ where $\Lambda=diag\{\lambda_1,\dots,\lambda_k,0,\dots,0\}$, the optimal solution is $V_{n\times k}\tilde{\Lambda}_{k\times k}^{\frac{1}{2}}$ *where* $\tilde{\Lambda}$ is a truncated version of Λ.

GDD

- The LBO basis is optimal for general smooth functions, but what if we constrain our functions to be the geodesic distance functions, another basis might be optimal.
- This work aims to find such a basis efficiently.
- Using furthest point-sampling an estimation of the geodesic distance matrix could be obtained.
- By clever mathematical manipulation (matrix decompositions and factorizations) an approximation of Q, Λ could be obtained such that:

$$D \approx \hat{Q}^T \Lambda \hat{Q}$$

  - a matrix X could be written such that

$$X = \hat{Q}\sqrt{\Lambda} \Longrightarrow XX^T \approx D$$

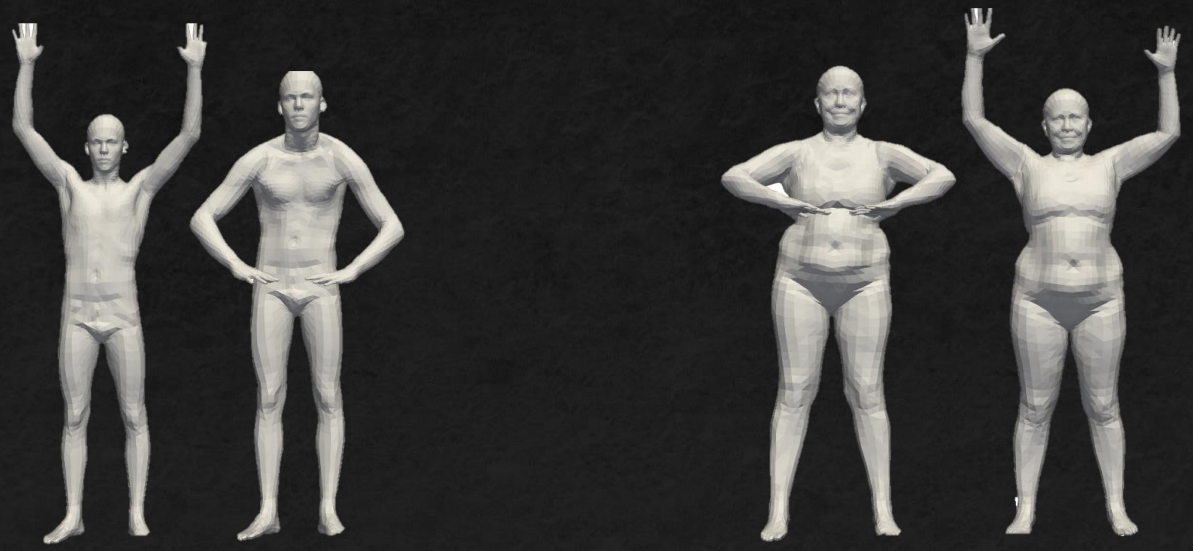
  - X is a set of complex point-descriptors, and the above equation shows that the geodesic distances are preserved in the Euclidean distances between the descriptors of the points.

Intro – Mathematical Framework

- Let X be the set of all non-rigid shapes and $S \subseteq X^X$ be the group of all isometric transformations, our objective is to find a function ϕ such that:

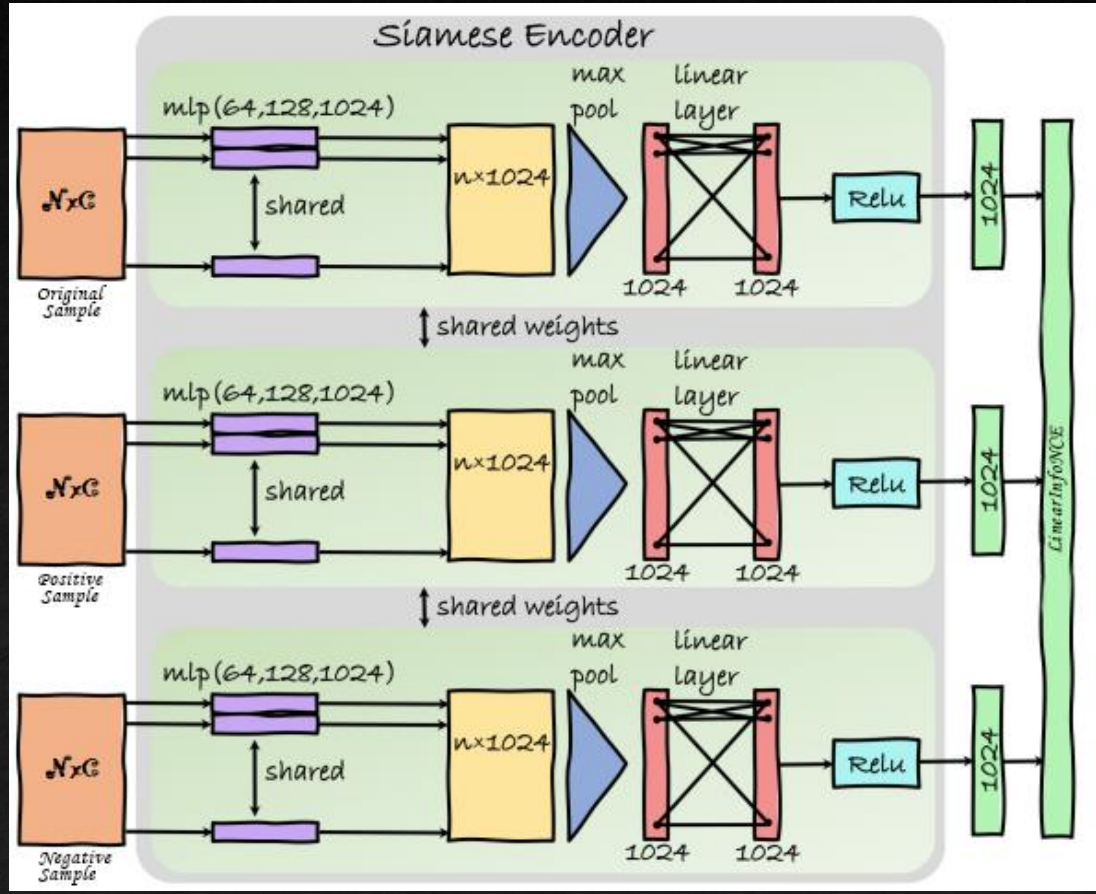
$$(1)\ \ \forall x, y \in X: \quad \phi(x) = \phi(y) \Leftrightarrow \exists \psi \in S, \psi(x) = y$$

- equivalently:
  - Given an object in some pose $x \in X$, $\forall \psi \in S\ \phi(x) = \phi\big(\psi(x)\big)$. (2)
  - Given two different objects $x, y \in X$ (i.e., $\forall \psi \in S, \psi(x) \neq y$), $\phi(x) \neq \phi(y)$. (3)



Training For Invariance

- Since we have no known target signatures, we must learn them in an unsupervised fashion.
- We use a Siamese learning scheme.
  - The same network is used to encode 3 samples: the original, a positive and a negative example.
  - The signatures are fed into our loss function, that minimizes the difference between the original and the positive sample and maximizes the difference with the negative sample.



Details: The signature

- As described, we aim for an invariant signature under the isometry transform group.
- We observe that point clouds are unordered sets, and thus to achieve invariance to the reparameterization, we aim for a global signature.
- We chose an embedding space of 1024 dimensions but constrained the embedding to be on the 1024-dimensional unit sphere.
- This constraint allows for more stable training, and more meaningful descriptors.

Details:
The Loss Function
LinearInfoNCE

- We aimed for a simple separation, preferably a linear one, and thus we used the angle between the signatures as our measure of distance.

- Since the signature is normalized, the inner product is equal to the cosine of the angle, and since the cosine is in $[-1,1]$,

$$\frac{\langle \phi(s), \phi(O) \rangle + 1}{2} \in [0,1]$$

and thus, could be used as a probability measure.

  - Note that if the vectors are identical, the probability is 1, if they are opposite, it is 0.

Details:
The Loss Function
LinearInfoNCE

- Meaning that the probability could be interpreted as the probability of the signatures being of the same object, and thus **BCE** could be used.

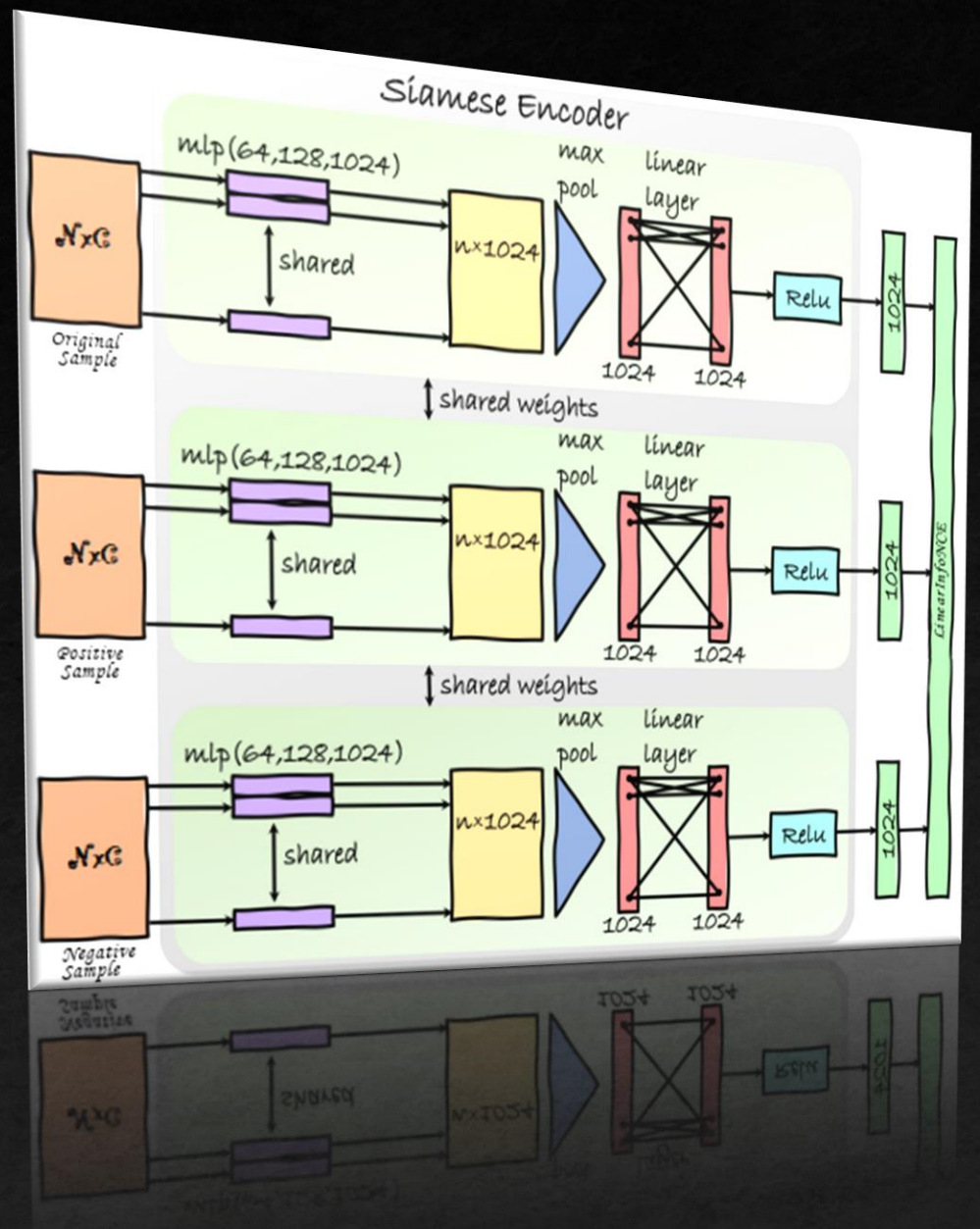
- the final loss function is:

$$l(\phi; G) \coloneqq \frac{1}{|G| - 1}\left[\sum_{s \in G_+} -\log\left(\frac{\langle \phi(s), \phi(O)\rangle + 1}{2}\right) + \sum_{s \in G_-} -\log\left(1 - \frac{\langle \phi(s), \phi(O)\rangle + 1}{2}\right)\right]$$

Where:

- G is the set of examples, G_+, G_- are the sets of positive and negative examples, and ϕ is the parametric function.

The Architecture



- Pointclouds are unordered sets, and thus we need a network invariant to re-permutation.
- We use a PointNet encoder with max-pooling, and a MLP head.
- The same encoder is shared amongst the 3 examples.
- We try feeding second moments as features to the encoder as well.

Experiments

- Our focus was not finding the best architecture possible, but rather a general training scheme that could be used with any architecture.
- Two important components are the loss function & normalization layers, thus we searched for the best combination.

Experiments -loss functions

- Contrastive Loss:

$$\mathcal{L}(W_1, W_2, Y) = (1 - Y)\frac{1}{2}D_{W_{12}}^2 + Y\frac{1}{2}\max\left(0, \mu - D_{W_{12}}\right)^2$$

- Triplet Loss:

$$\mathcal{L}(x, x_p, x_n) \coloneqq \max\left(0, \gamma\left\|f(x_p) - f(x)\right\|_2^2 - (1 - \gamma)\left\|f(x_p) - f(x)\right\|_2^2 + \alpha\right)$$

- SigmoidInfoNCE:

$$l(\phi; G) \coloneqq \frac{1}{|G| - 1}\left[\sum_{s \in G_+} -\log\left(\sigma\left(\frac{\langle\phi(s), \phi(O)\rangle}{\tau}\right)\right) + \sum_{s \in G_-} -\log\left(1 - \sigma\left(\frac{\langle\phi(s), \phi(O)\rangle}{\tau}\right)\right)\right]$$

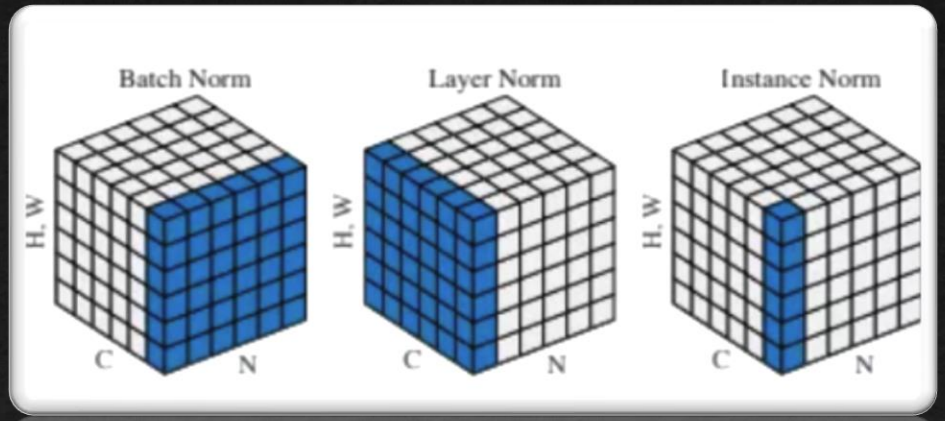
- LinearInfoNCE:

$$l(\phi; G) \coloneqq \frac{1}{|G| - 1}\left[\sum_{s \in G_+} -\log\left(\frac{\langle\phi(s), \phi(O)\rangle + 1}{2}\right) + \sum_{s \in G_-} -\log\left(1 - \frac{\langle\phi(s), \phi(O)\rangle + 1}{2}\right)\right]$$

Experiments -Normalization

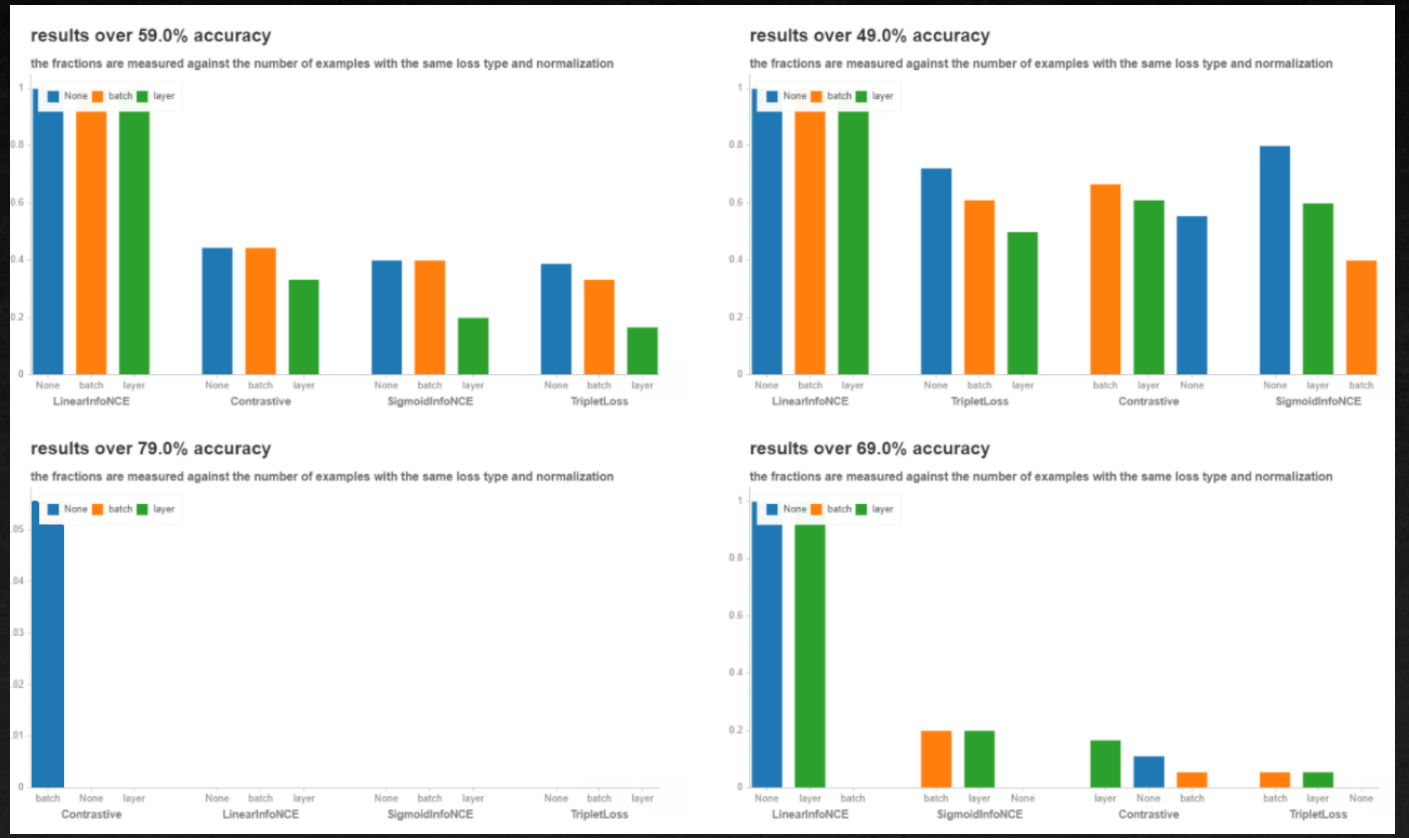
- Normalization is a very important part of any deep learning architecture it usually provides a more stable learning process, and better overall results.

- Different types of normalization have been proposed over the years, we considered a few:
  - Batch normalization.
  - Layer normalization.
  - Instance normalization.
  - No normalization.



Experiments – The Search!

- We observe that the loss function and normalization type might not be independent, and thus we run a grid search, allowing us to find the best pair.
- We run a K-Fold validation on each pair (loss function & normalization type) and calculate the percentage of models that reached certain accuracy thresholds.



Testing Method #1

- We want to test weather our two goals are satisfied:
  - Given an object in some pose $x \in X$, $\forall \psi \in S\ \phi(x)=\phi(\psi(x))$. (2)
  - Given two different objects $x,y \in X$ (i.e., $\forall \psi \in S$, $\psi(x) \neq y$), $\phi(x) \neq \phi(y)$. (3)

- If (3) is satisfied, two the signatures of different shapes coming from two different objects are necessarily unique, and thus allow for classification of the objects by their signature.
  - An MLP is used to classify the objects by their signatures, a high accuracy means that (3) is satisfied.

- If (2) is satisfied, the signatures of two shapes of the same object (different pose) are similar, and thus would not allow for classification of the pose given the signature.
  - An MLP is used to classify the poses by their signatures, a low accuracy means that (2) is satisfied.

Testing Method #2

- We use 3 types of encoders:
  - Baseline classifier (after removing the linear head).
    - Uses our architecture but is trained end-to-end for classification.
  - Geodesic distance descriptors.
  - Our encoder (w\ and w\o second moments).
- The learning-based methods are all trained on DFaust.
- The encoders are all evaluated on:
  - DFaust in shape classification with a 1- and 2-layer MLP.
  - Faust in shape classification with a 2-layer MLP.
  - Faust in pose classification with a 2-layer MLP.

Results w\o Rotation

- Running the previously mentioned evaluation technique without applying a random rotation yields the following results:

c	Shape classification (↑)			Pose Classification (↓)
Algorithm \ Test	DFaust – 1 layer head	DFaust – 2 layer head	Faust – 2 layer head	Faust – 2 layer head
Baseline classifier	91.1%	91.08%	65%	40%
SN w/o moments	99.54%	99.1%	65%	15.8%
SN w/ moments	99.86%	99.54%	40%	35%
GDD	12%	92%	25%	10%

- Our model is superior to the SOTA (GDD) by a significant margin.

Rotation Invariance

- In some settings general rotations are a natural isometry transformation, and thus we try to make our encoder rotation invariant.

- Running our algorithm unchanged produces the following results:

	Shape classification (↑)			Pose Classification (↓)
Algorithm \ Test	DFaust – 1 layer head	DFaust – 2-layer head	Faust – 2-layer head	Faust – 2-layer head
Baseline classifier	82%	82%	30%	70%
SN w/o moments	75.1%	74.6%	25%	33%
SN w/ moments	77%	80%	50%	36.6%
GDD	12%	92%	25%	10%

- The performance drops significantly, putting the algorithm below the previous SOTA which is completely rotation invariant as it is based on the geodesic distance matrix.
  - We avoid using geodesic distances as this information cannot always be obtained reliably.

Rotation Invariance

PCA Normalization

Using second moment information about the shape in order to align it.
This is done as a pre-processing step, and thus alleviate the burden from the network itself.

Alignment Network

Learning a function that can extract the rotation transform that aligns a given mesh.
We note that this an ill posed problem since there is not one correct orientation for a mesh.

Second Moments

We observe that the rotation information is all encoded into the covariance matrix which could be extracted from the second moments.
We try feeding the second moments to our network in hopes that it implicitly learns the rotation.

Second Moments

- Second Moment information can be very useful in determining orientation, and other statistics about objects.

- Neural networks have a hard time learning multiplications, and as such feeding this information to the network directly can reduce the amount of work that the network must do, and thus making the task simpler.
  - This allows us to use shallower networks.

Alignment Networks

- Objective: learn a function, such that given some shape, the functions outputs some representation of a rotation that when applied to the shape, gives the shape in some canonical orientation (e.g., shoulders along the x axis, spine along the y axis).

- We learn such a function in a supervised fashion, and we experimented with three losses:
  - L2 loss over the difference between the predicted Euler angles representation of the predicted and the ground truth angles.
  - L2 loss over the difference between the predicted rotation matrix and the ground truth rotation matrix.
  - L2 loss between the rotation matrix transposed multiplied by the ground truth matrix, and the unit matrix.

PCA - Normalization

- PCA finds the set of orthonormal directions of the highest variance of the shape's projection, then projects and expresses the shape in this basis.
  - These could be shown to be the eigen values of the covariance matrix.

- Denote the coordinates of the shape by X, and let R be an arbitrary rotation, $Y = RX$:

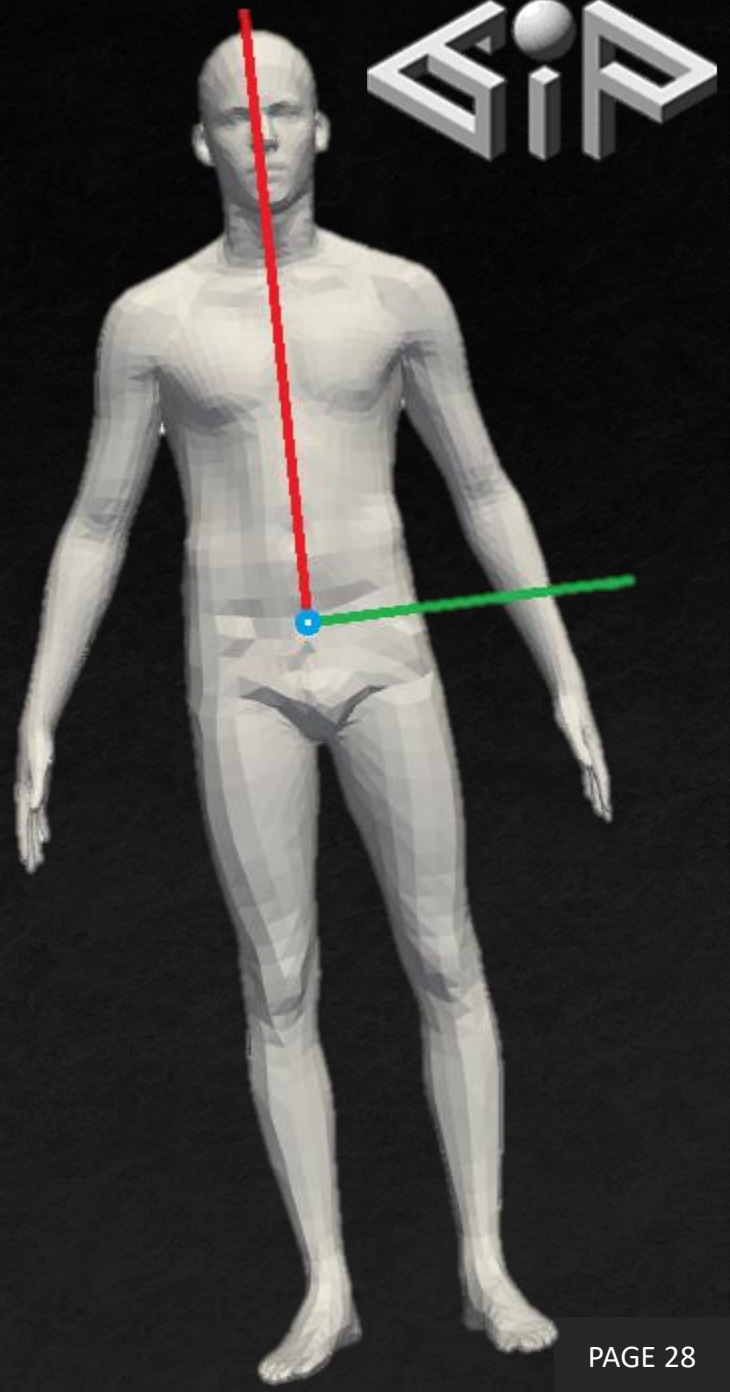
$$\mathbb{E}[YY^T] = \mathbb{E}[RXX^T R^T] = R\mathbb{E}[XX^T]R^T = R\mathbb{E}[XX^T]R^T$$

- Let u be an eigen vector with eigenvalue λ of $R\mathbb{E}[XX^T]$, denote $v = Ru$:

$$\mathbb{E}[YY^T]u = R\mathbb{E}[XX^T]R^T Rv = R\mathbb{E}[XX^T]v = \lambda Rv = \lambda u$$

- Thus, denoting the original PCA matrix as T_X:

$$T_y = RT_x \Longrightarrow PCA_Y = T_y^T Y = T_x R^T RX = PCA_X$$



PCA - Normalization

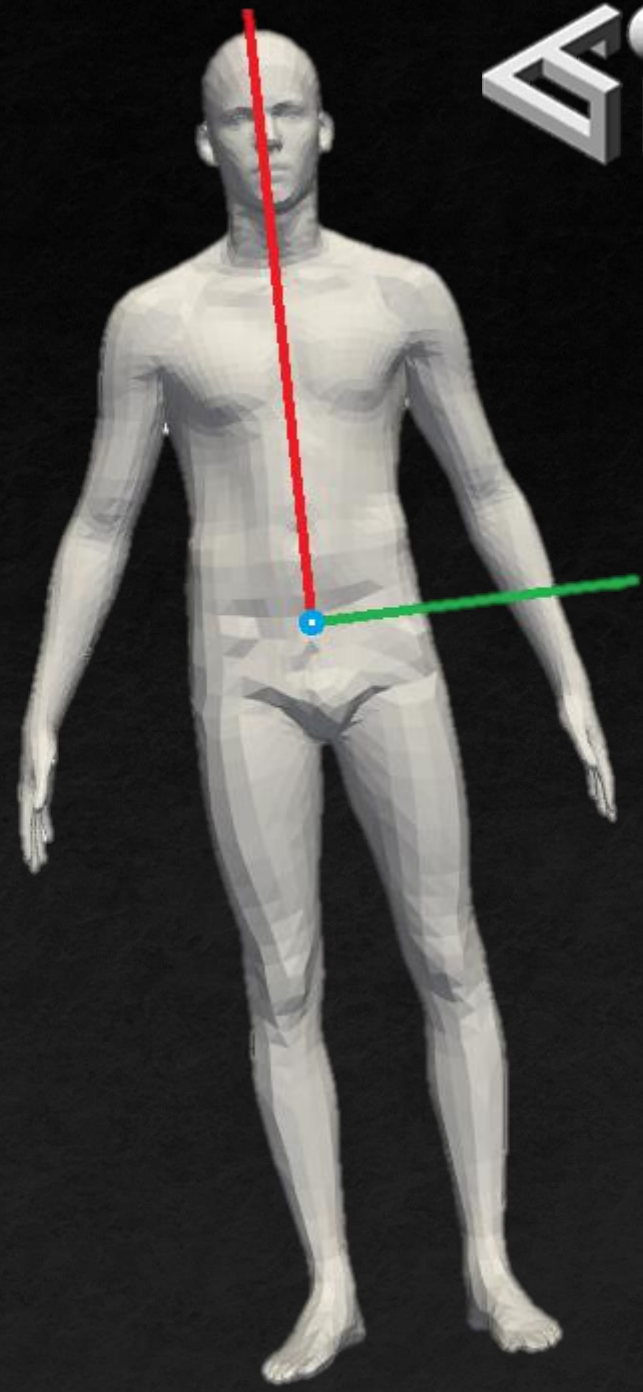
- Using the standard PCA algorithm to embed our pointcloud into $\mathbb{R}^3$, the embedding becomes a simple rotation and translation transform.

- We proved that the embedding is rotation invariant, additionally we observe that because of the nature of PCA embeddings, for most poses, humanoids are aligned naturally by PCA.

- The solution is not unique and is determined up to a flip of each coordinate independently, meaning a multiplication with:

$$diag\{\pm 1, \pm 1, \pm 1\}$$

- Note: if two eigenvalue were equal, a degree of freedom representing planar rotation would be added, but we showed this not to be the case, and we argue that it is numerically unlikely.



Results w\ Rotation

	Shape classification (↑)			Pose Classification (↓)
Algorithm \ Test	DFaust – 1 layer head	DFaust – 2-layer head	Faust – 2-layer head	Faust – 2-layer head
Baseline classifier	82%	82%	30%	70%
SN w/o moments	75.1%	74.6%	25%	33%
SN w/ moments	77%	80%	50%	36.6%
PCA baseline	12%	11.6%	30%	23%
PCA SN w/o moments	88.22%	86.34%	65%	10%
PCA SN w/ moments	91.48%	92%	55%	18%
GDD	12%	92%	25%	10%

- We see that using the PCA normalization we achieve results that are superior to the SOTA in most metrics.
- Notably we do that in a lower complexity, and thus allowing for computations on larger pointclouds.

Conclusions

- We propose a training framework which produces SOTA results.
- Assuming aligned data, our framework needs no preprocessing and surpasses the previous SOTA by a large margin.
- When removing the above assumption, we add a pre-processing step, which helps the model surpass the SOTA.

Possible Future Work

- Our work did not focus on the network architecture, future works might investigate designing better fitting architectures.
- Extending our work by adding a generator that takes in the signature and reconstructing the shape in a constant\conditioned pose.
- Learning a rotation invariant signature without the use of PCA normalization.
- Using geodesic distances as an extra input to further improve the pose invariance.

Questions

References

[1] G. Pai, A. Wetzler, and R. Kimmel, “L EARNING I NVARIANT R EPRESENTATIONS O F,” pp. 1–11, 2017.

[2] R. S. Society, “Review of the Development of Multidimensional Scaling Methods Author (s): A . Mead Source : Journal of the Royal Statistical Society . Series D (The Statistician) , 1992 , Vol . Published by : Wiley for the Royal Statistical Society Stable URL : https,” vol. 41, no. 1, pp. 27–39, 1992.

[3] J. A. Burgoyne and S. McAdams, “Non-linear scaling techniques for uncovering the perceptual dimensions of timbre,” *Int. Comput. Music Conf. ICMC 2007*, no. April 2014, pp. 73–76, 2007.

[4] J. A. Sethian, “Fast Marching Methods,” *SIAM Rev.*, vol. 41, no. 2, pp. 199–235, 1999, doi: 10.1137/S0036144598347059.

[5] K. Crane, C. Weischedel, and M. Wardetzky, “The heat method for distance computation,” *Commun. ACM*, vol. 60, no. 11, pp. 90–99, 2017, doi: 10.1145/3131280.

[6] G. Shamai, Y. Aflalo, M. Zibulevsky, and R. Kimmel, “Classical scaling revisited,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 International Conference on Computer Vision, ICCV 2015, pp. 2255–2263, 2015, doi: 10.1109/ICCV.2015.260.

[7] E. Peterfreund and M. Gavish, “Multidimensional scaling of noisy high dimensional data,” *Appl. Comput. Harmon. Anal.*, vol. 51, pp. 333–373, 2021, doi: 10.1016/j.acha.2020.11.006.

[8] S. Martin and J. P. Watson, “Non-manifold surface reconstruction from high-dimensional point cloud data,” *Comput. Geom. Theory Appl.*, vol. 44, no. 8, pp. 427–441, 2011, doi: 10.1016/j.comgeo.2011.05.002.

[9] S. M. Holland, “Non-Metric Multidimensional Scaling (MDS),” *J. Cell Biol.*, no. May, p. 8, 2008.

[10] E. Beutner and U. Kamps, “Order restricted statistical inference for scale parameters based on sequential order statistics,” *J. Stat. Plan. Inference*, vol. 139, no. 9, pp. 2963–2969, 2009, doi: 10.1016/j.jspi.2009.01.017.

[11] G. Shamai and R. Kimmel, “Geodesic distance descriptors,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 3624–3632, 2017, doi: 10.1109/CVPR.2017.386.

[12] A. van den Oord, Y. Li, and O. Vinyals, “Representation Learning with Contrastive Predictive Coding,” 2018, [Online]. Available: http://arxiv.org/abs/1807.03748.

[13] O. Halimi et al., “Towards Precise Completion of Deformable Shapes,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 12369 LNCS, pp. 359–377, 2020, doi: 10.1007/978-3-030-58586-0_22.

[14] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017, vol. 2017-Janua, pp. 77–85, 2017, doi: 10.1109/CVPR.2017.16.

[15] M. Joseph-Rivlin, A. Zvirin, and R. Kimmel, “Momenet: Flavor the moments in learning to classify shapes,” Proc. - 2019 Int. Conf. Comput. Vis. Work. ICCVW 2019, pp. 4085–4094, 2019, doi: 10.1109/ICCVW.2019.00503.

[16] J. Kang and A. K. Patterson, “Principal component analysis of mRNA levels of genes related to inflammation and fibrosis in rats treated with TNBS or glutamine,” Inflammatory Bowel Diseases, vol. 17, no. 7. pp. 1630–1631, 2011, doi: 10.1002/ibd.21544.

[17] R. Hadsell, S. Chopra, and Y. LeCun, “Dimensionality reduction by learning an invariant mapping,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2, pp. 1735–1742, 2006, doi: 10.1109/CVPR.2006.100.

[18] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 07-12-June, pp. 815–823, 2015, doi: 10.1109/CVPR.2015.7298682.

[19] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” 32nd Int. Conf. Mach. Learn. ICML 2015, vol. 1, pp. 448–456, 2015.

[20] T. Salimans and D. P. Kingma, “Weight normalization: A simple reparameterization to accelerate training of deep neural networks,” Adv. Neural Inf. Process. Syst., no. Nips, pp. 901–909, 2016.

[21] Y. Wu and K. He, “Group Normalization,” Int. J. Comput. Vis., vol. 128, no. 3, pp. 742–755, 2020, doi: 10.1007/s11263-019-01198-w.