

Fig. 18. Illustration of the NeuralSRNF network architecture. **(a)** NeuralSRNF recovers the neural surface f for a given SRNF map q. **(b)** The detailed network architecture of NeuralSRNF. Given a point s, its corresponding SRNF value $q(s)$, and DinoV2 encoded latent code $\mathbf{x}$, we map it to the point on the surface p. **(c)** A close-up view of the Feature-wise Linear Modulation (FiLM) layer.

VI. Supplementary Material

This Supplementary Material provides additional details, results, and evaluations. Section A discusses further the limitations and future directions of our approach. Section A provides details on the datasets used in this paper. Section A provides the implementation details of our network architecture, Section A shows the spherical parameterization results, Section A shows the ablation results of the proposed NeuralSRNF, and Section C presents additional results of the proposed framework.

Appendix

Spatial registration. Our approach requires spatially-registered surfaces, which can be obtained using existing registration techniques such as functional maps or elastic shape analysis methods. In the future, we plan to explore mechanisms for training the proposed NeuralSRNF using unregistered 3D shapes *i.e.,* 3D shapes that are not in correspondence.

Class agnostic. NeuralSRNF needs to be trained for every class of shapes separately. The existing network architecture fails to generalize the inversion of SRNFs for multi-class shapes. As a future direction, we plan to explore network architecture designs that can reconstruct the 3D surface for any given SRNF maps, in a class-agnostic manner.

Higher genus shapes. Our approach operates on spherically parameterized surfaces, limiting the inversion framework to closed genus-0 3D shapes. This already covers a large number of common classes of objects such as humans, animals, and even anatomical organs. NeuralSRNF extends in a straightforward manner to open genus-0 surfaces by parameterizing them using an open domain, such as a disk. However, extending the parameterization to higher genus requires complex parameterization, such as a volumetric domain. We plan to explore this as a future direction.

We have trained and evaluated the proposed NeuralSRNF on:

- MPI FAUST [4] is a dataset of high-resolution 3D scans of humans in different poses. It consists of 10 subjects with 10 poses.
- MPI Dynamic FAUST [5] is a dataset of high-resolution 4D scans of humans in motion. It consists of 10 subjects with $40,000$ aligned meshes, performing actions such as jumping jacks, punching, and running, among others.
- MPI CoMA [36] is a dataset of high-resolution 4D scans of human faces performing extreme facial expressions. It consists of 12 subject faces with over $20,000$ aligned meshes, performing actions such as high smile, lips back, and bareteeth, among others.
- Animal3D [43] is a dataset of high quality 3D animals parameterized using the SMAL model [47]. It contains 40 mammal species with over $3,000$ registered meshes recorded in diverse 3D shapes and poses. The five main categories are dog, tiger, cow, horse, and hippo.

We spherically parameterize each mesh using Kurtek *et al.* [19]'s implementation of the approach of Hoppe and Praun [35].

Network design. The proposed network architecture of NeuralSRNF follows the design of NeRF [28] and π-GAN [8]. It consists of eight residual blocks each containing a FiLM layer and two linear layers with 1024 nodes. The FiLM layer, which stands for Feature-wise Linear Modulation, also consists of two linear layers that produce the scaling vector and the shifting vector *i.e.,* $FiLM(\mathbf{x}) = \gamma \odot \mathbf{x} + \beta$. Here, the linear layers of γ and β have 1024 nodes, and $\mathbf{x}$ is the encoded latent code of the SRNF map q generated using DinoV2. We employ the small model of DinoV2 with registers, which requires a normalized q of size $224 \times 224 \times 3$. In addition, we apply input sampling to sample 256 points on sphere s and select their corresponding SRNF values $q(s)$ and apply positional encoding to them. We use 10 Fourier frequency bands for positional encoding.

Forward pass. A single forward pass of NeuralSRNF can be described as $F_\Theta : (s_i, q_j(s_i); \mathbf{x}_j) \rightarrow f_j(s_i)$. Here F_Θ is the proposed network, j is the batch size, and i are the sampled points. The input batch can be formulated from q, which has dimensions $u \times v \times 3$, where u and v are the spherical coordinates of $s(u, v) \in \mathbb{S}^2$. The $\mathbf{x}_j$ are the latent code generated from q_j using DinoV2. Note that DinoV2 requires q_j to have values within the range of $[0, 1]$ with a resolution of $224 \times 224 \times 3$. Now, we use NeuralSRNF to map the inputs s_i, $q_j(s_i)$, and $\mathbf{x}_j$ to the points on the surfaces



Fig. 19. Close-up views of a spherically parameterized 3D human at $2048 \times 2048 \times 3$.

Hyperparameters	Value
Epochs	1000
Batch size	8
Number of sample points per batch	256
Normal loss weights (λ)	0.2
Optimizer	AdamW.
Learning rate	10^{-4}
Weight decay	10^{-2}
Cosine annealing (eta min)	10^{-6}
Residual blocks	8
Positional Encoding	10 frequency bands.

TABLE IV
HYPERPARAMETER SETTING FOR THE TRAINING OF THE PROPOSED NEURALSRNF NETWORK ARCHITECTURE.

Spherical Parameterization	Mean	Std	Median
3D Animal	0.0092	0.0105	0.0059
3D Human	0.0028	0.0024	0.0022

TABLE V
PARAMETERIZATION ERROR FOR 3D HUMANS AND 3D ANIMALS.

$f_j(s_i)$.

Training. We use AdamW as an optimizer with a weight decay of $1e^{-2}$ and a learning rate of $1e^{-4}$. We apply CosineAnnealing to gradually reduce the learning rate from $1e^{-4}$ to $1e^{-6}$. We define 1000 epochs with a batch size of 8 and sample 256 points for a given SRNF map q. For 3D human datasets such as DFAUST [5] and CAPE [26], we segment the constituent body parts such as faces, hands, and legs and randomly sample points within each regions. See Table IV for detailed hyperparameter settings.

In this section, we perform the quantitative and qualitative evaluation of the spherical parameterization. In this paper, we use the Kurtek *et al.* [19] implementation of the Hoppe and Praun approach [35] for spherical parameterization. Table V shows the sensitivity analysis of the spherical parameterization and reports the mean, median, and standard deviation error computed over 5000 parameterized human shapes and 5000 parameterized animal shapes. As shown in the table, the parameterization error is very low, indicating that the surface quality is highly accurate.

Figure 19 shows a high-resolution 3D human spherically parameterized at a high resolution of $2048 \times 2048 \times 3$, demonstrating that spherically parameterized surfaces can capture fine-grained surface details.

In this section, we perform the qualitative and quantitative ablation of the network architecture. We consider the 3D human dataset of DFAUST for this study and evaluate the effects of input sampling, latent codes, and FiLM layer.

Sampling	Mean	Std	Median
Segmented	$\mathbf{0.043 \times 10^{-3}}$	$\mathbf{0.034 \times 10^{-3}}$	$\mathbf{0.036 \times 10^{-3}}$
Random	0.125×10^{-3}	0.127×10^{-3}	0.092×10^{-3}

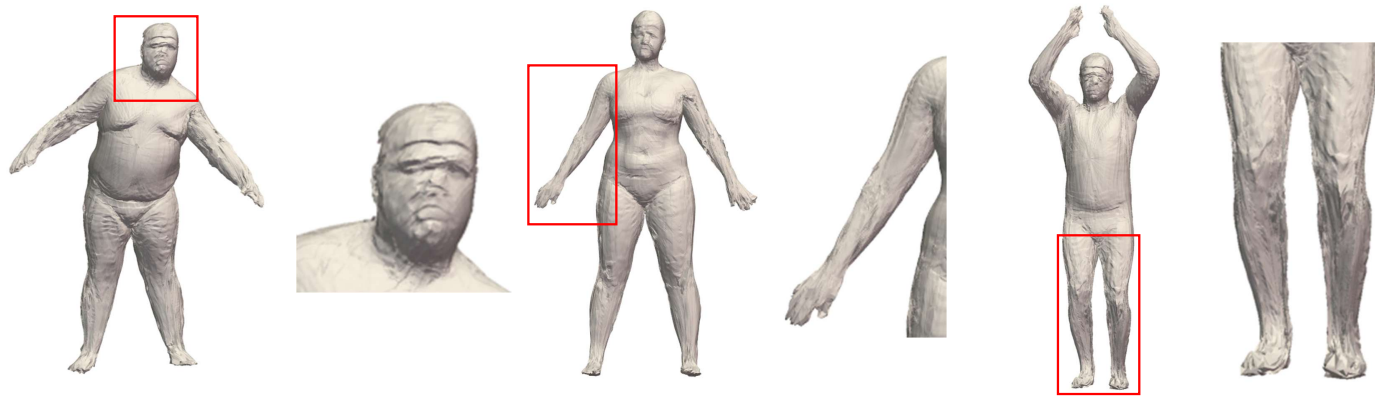
TABLE VI
COMPARISON OF THE EFFECT OF INPUT SAMPLING ON THE NETWORK ARCHITECTURE. THE EVALUATION IS PERFORMED ON 100 TEST SAMPLES FROM THE DFAUST DATASET, USING RANDOM SAMPLING ON THE ENTIRE HUMAN BODY AND ON SEGMENTED BODY PARTS, SUCH AS THE FACE, HANDS, AND LEGS. WE MEASURE THE MEAN SQUARE DISTANCE AND REPORT THE RECONSTRUCTION ERROR.

A. Input sampling

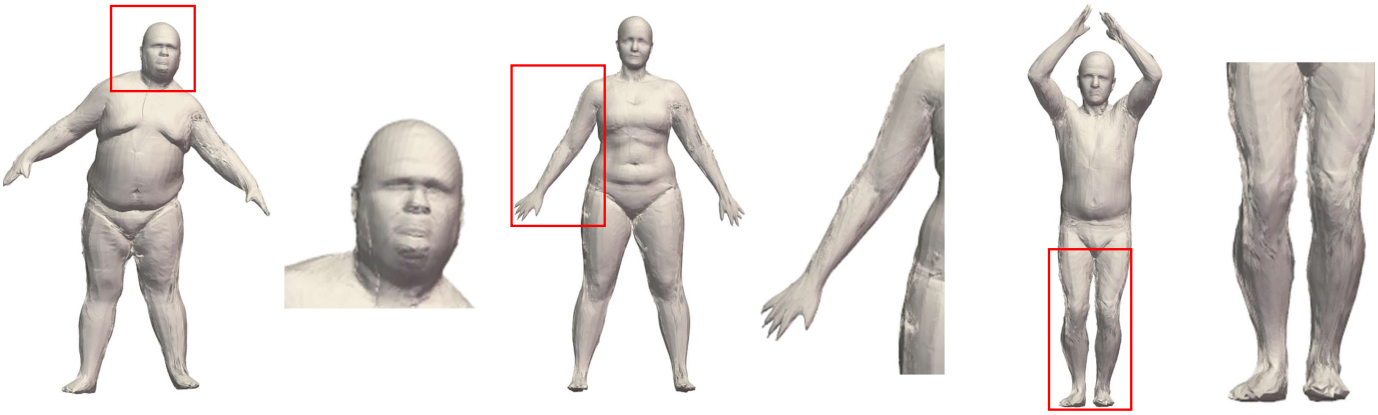
During training, instead of randomly sampling training points uniformly across the entire surface, we segmented shapes into semantic parts (*e.g.,* body parts in the case of human shapes), and sample within the parts to ensure that each semantic part is well represented in the training data. In this experiment, we evaluate the effects of random sampling and segmented sampling on the quality of the recovered neural surfaces. We trained two NeuralSRNF networks on the DFAUST dataset with the same hyperparameters. For the first network, we sample 256 points randomly across the entire body surface. Whereas, for the second network, we segment the 3D human body into its constituent parts, such as faces, arms, legs, and body, and then randomly sample points within each of these regions.

Figure 20 shows the effect of random sampling and segmented sampling. As one can clearly see that the surfaces in Figure 20-**(a)**, recovered using the first network, have artifacts and are unable to represent important body parts, especially small ones, such as faces and hands. Whereas, Figure 20-**(b)**, which uses segmented sampling, is able to recover the details on important body parts.

Table VI shows the quantitative results of input sampling with the segmented approach and the random sampling approach. The results demonstrate that randomly sampling points on each region of the segmented body part is able to generate more accurate results with an average reconstruction error of 0.043×10^{-3} compared to 0.125×10^{-3} when applying random sampling on the whole human body.



(a) Random sampling.



(b) Segmented sampling.

Fig. 20. We show the effect of training a NeuralSRNF with **(a)** random sampling and **(b)** segmented sampling. We show the close-up views of constituent body parts. Observe that the surfaces reconstructed with **(b)** have more detailed body parts.

B. Importance of the DinoV2 latent code

We train a NeuralSRNF without DinoV2 latent codes. Figure 21-**(a)** illustrates the network architecture used in this experiment, Figure 21-**(b)** highlights some examples of the inversion of SRNF using the architecture of Figure 21-**(a)**. As one can see, the network architecture is unable to recover the surface f for a given SRNF q. This experiment shows the importance of a global latent code $\mathbf{x}$.

Instead of using learned DinoV2 latent codes, one can learn the latent code in an auto-decoding fashion following DeepSDF [33]. Such method, however, can be very slow at runtime, since the code needs to be optimized at inference time. This would make the approach computationally inefficient and impractical for the inversion of SRNF maps.

C. Importance of the FiLM layer

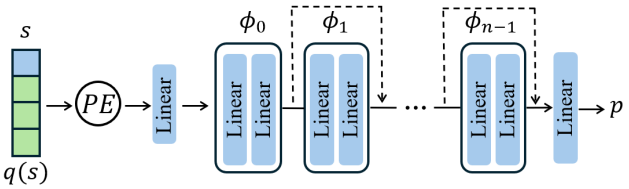
Figure 22 shows network architecture without the FiLM layer. It follows early fusion and concatenates the DinoV2 features and SRNF value after applying a linear projection. For the network architecture with FiLM layer; see Figure 18. We trained both network architectures with DinoV2 latent codes and quantitatively and qualitatively evaluated the importance of the FiLM layer.

Conditioning	**Mean**	**Std**	**Median**
FiLM	$\mathbf{0.043 \times 10^{-3}}$	$\mathbf{0.040 \times 10^{-3}}$	$\mathbf{0.034 \times 10^{-3}}$
Concatenation	0.118×10^{-3}	0.107×10^{-3}	0.078×10^{-3}

TABLE VII

COMPARISON OF CONDITIONING OF THE NETWORK ARCHITECTURE WITH FILM LAYER AND WITH FEATURE CONCATENATION. THE EVALUATION IS PERFORMED ON 30 TEST SAMPLES OF THE DFAUST DATASET WITH EXTREME POSES. WE MEASURE THE MEAN SQUARE DISTANCE AND REPORT THE RECONSTRUCTION ERROR FOR BOTH CONDITIONING APPROACHES.

Figure 23 shows the qualitative results of the conditioning on latent codes using feature concatenation or a FiLM layer. Observe that the surfaces reconstructed using FiLM-based conditioning in Figure 23-**(b)** are able to accurately represent the correct poses. Whereas, Figure 23-**(a)**, which uses feature concatenation, is unable to accurately represent the correct poses.



(a) Network architecture without DinoV2 latent codes).



(b) Surfaces recovered from their SRNF maps using NeuralSRNF trained without a latent code.

Fig. 21. We show the network architecture design without the DinoV2 latent codes in **(a)**. In **(b)**, we show the recovered neural surfaces for a given q.

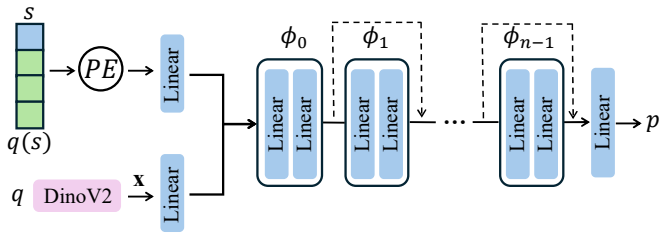


Fig. 22. We show the fusion-based conditional network architecture design tested during the design of NeuralSRNF.

Table VII shows the quantitative results of conditioning with FiLM layer and without. This demonstrates that the FiLM layer is able to generate more accurate extreme poses with an average reconstruction error of 0.043×10^{-3} compared to 0.118×10^{-3} when dropping the FiLM layer.

This section provides additional results of our framework that could not fit within the page limit of the main manuscript. We also show the main manuscript figures in high resolution.

D. SRNF Inversion

Figure 24 shows the qualitative results of the SRNF inversion on 3D faces, humans, and animals. This figure demonstrates additional results of Figure 3 presented in the main manuscript. Here, we display **(a)** the ground truth 3D surface, **(b)** the inversion computed using the numerical inversion approach of Laga *et al.* [21], and **(c)** the inversion computed using the proposed NeuralSRNF. We show the reconstruction errors for Figure 24-**(b)** and Figure 24-**(c)** plotted as a heat map. Observe that the proposed NeuralSRNF is able to accurately represent complex surfaces with a reconstruction error that is less than 0.1.

Figure 25 shows additional SRNF inversion results of the proposed NeuralSRNF for 3D animals, faces, and humans. Similar to Figure 24, we show the reconstruction accuracy for each 3D surface computed as the pointwise error between the original surface and the surface obtained by SRNF inversion. As one can see that the NeuralSRNF is able to accurately reconstruct complex 3D surfaces with an error that is less than 0.1.

E. Geodesics

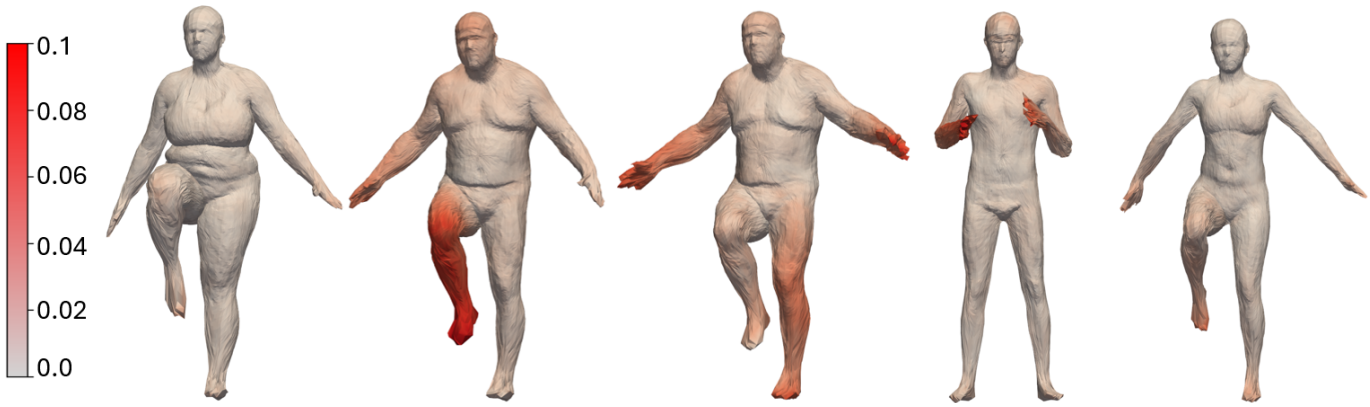
Figures 27 and 27 show in high resolution the qualitative results of Figures 6 and 8 in the main manuscript. Both figures show **(a)** the linear path, **(b)** the geodesic computed using numerical SRNF [21], **(c)** the geodesic computed using implicit neural surface deformation [37], **(d)** the geodesic computed using 4Deform [38], and **(e)** the geodesic computed using the proposed NeuralSRNF. As one can see, the geodesics computed using the proposed NeuralSRNF are more natural and do not exhibit unnatural shrinkage, distortions, or artifacts as observed in other methods. Similar to this, Figure 28 shows an additional example of geodesics computed on 3D humans.

Figures 29 and 30 show additional geodesic comparisons on 3D humans and Figures 31, 32, and 33, show it on 3D animals. For all these figures, we compare the proposed NeuralSRNF with **(a)** the linear path and **(b)** the numerical SRNF [21]. Observe that the movement of human hands and the movement of animal heads look natural in the geodesics computed using NeuralSRNF.

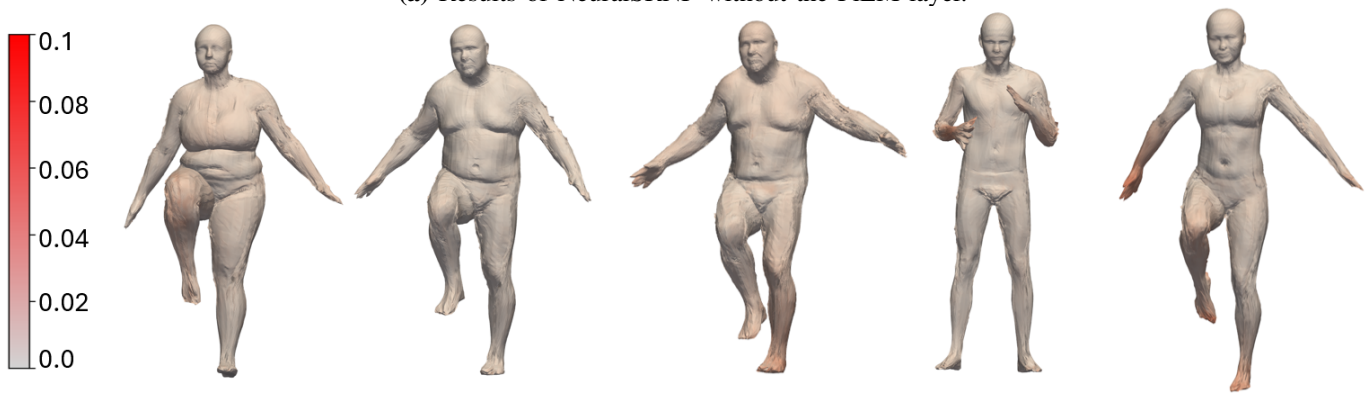
Figure 34 follows 4Deform [38] and performs the interpolation between key frames. The authors of 4Deform [38] have evaluated their approach on similar looking key frames *i.e.,* interpolating the sequence that has the same shape with sparse temporally consistent intermediate frames. However, in this example, we have evaluated the 4Deform [38] and the proposed NeuralSRNF on extreme poses with different 3D shapes. We show the mean shape between eight different 3D shape pairs. As we can see, the interpolation of 4Deform exhibits missing surfaces with unnatural shrinkage, whereas the interpolation using the proposed NeuralSRNF is able to generate natural poses between pairs.

F. Deformation transfer

Figure 35 shows an example of deformation transfer from the **(a)** source deformations *i.e.,* $f_1 \rightarrow h_1$ that deform f_1 onto the surface h_1, we transfer this deformation computed in the SRNF space $Q(v) = Q(h_1) - Q(f_1)$ to the target surface $Q(f_2)$ resulting in a new surface $Q(h_2)$. As one can see, in Figure 35-**(c)** that the deformation transferred in the SRNF space, when inverted to the original space $\mathcal{F}$ of surfaces using the proposed NeuralSRNF, looks natural and similar to h_1 without unnatural shrinkage. Figures 36 and 37 show additional examples of deformation transfer of Figure 10 in the main manuscript. Here, we apply the linear interpolation and extrapolation computed on the source deformation as $Q(v)$ to f_2 using the deformation transfer approach described in the main manuscript.



(a) Results of NeuralSRNF without the FiLM layer.



(b) Results of NeuralSRNF with the FiLM layer.

Fig. 23. We show the contribution of the FiLM layer as a conditioning layer of NeuralSRNF. The networks were trained using the same hyperparameter setting. **(a)** SRNF inversion results without the FiLM layer. **(b)** SRNF inversion results with the FiLM layer. The pointwise error between the groundtruth surface and reconstructed surface is color-coded. Observe that NeuralSRNF with the FiLM layer is able to accurately recover the correct pose of the surface.

G. Summary statistics and 3D shape generation

Figures 38 and 39 show summary statistics and shape generation, for 3D humans, computed in the SRNF space and inverted to the original space using the proposed NeuralSRNF. As one can see in Figure 39-**(b)**, 3D shapes generated in the SRNF space and inverted to the original space using the proposed NeuralSRNF look natural, whereas Figure 39-**(a)** has unnatural artifacts such as missing and shrunk limbs.

Figure 40 shows the same results as of Figure 12 but from a different viewing angle.

Figures 41 and 42 show the computed summary statistics of animals in the SRNF space and inverted using numerical SRNF [21]. Figure 43 and 44, on the other hand, show the summary statistics computed in the original space $\mathcal{F}$ of surfaces. Observe that the surfaces computed in the SRNF space and inverted using NeuralSRNF look accurate and detailed.

Figure 45 shows 24 generated shapes in the space of SRNF and inverted using the proposed NeuralSRNF.

H. Application to 4D surfaces

Figure 46 shows the high resolution 4D geodesic comparison of Figure 14 from the main paper. Note that we only show the geodesic sequences; we do not show the high-resolution results of spatiotemporal registration due to space constraints.

Figure 47 shows the 4D sequence comparison of the proposed NeuralSRNF with the linear interpolation approach. Here, we show the additional results of Figure 15 of the main paper and compare it with the linear approach.

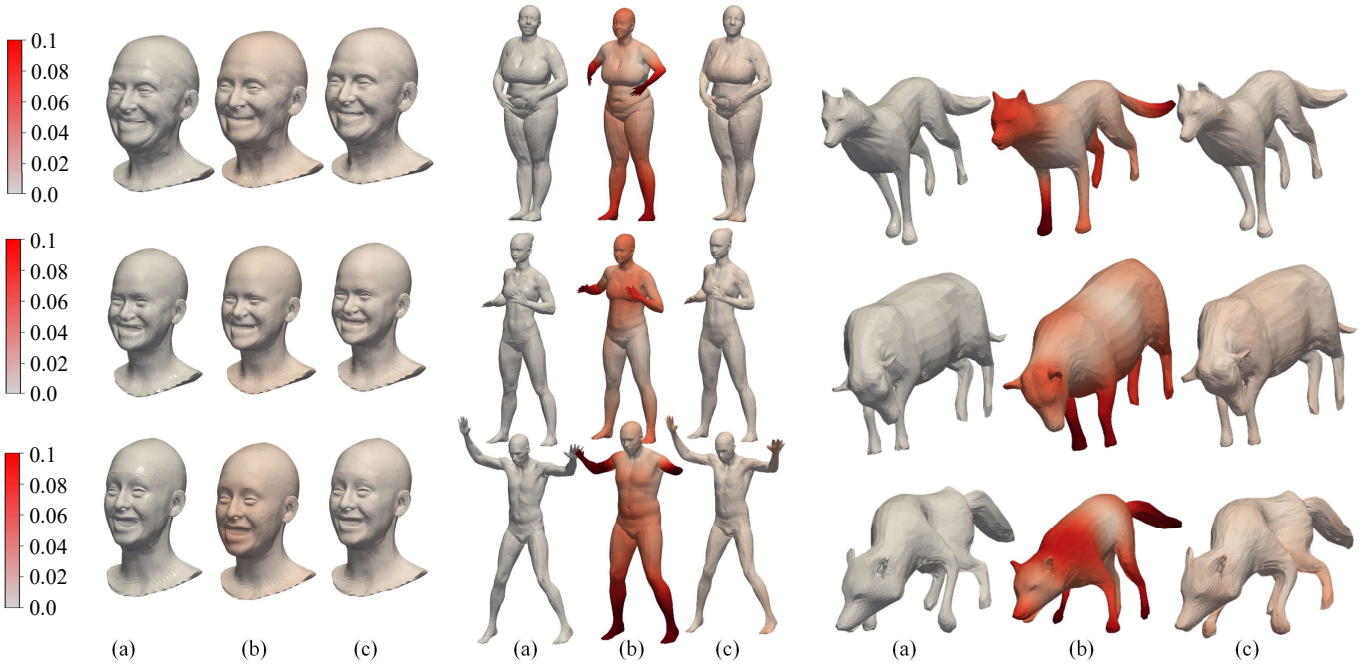


Fig. 24. We highlight the additional qualitative results of SRNF inversion for faces, humans, and animals. Here, (a) represents the ground truth surface, (b) represents the numerical inversion of SRNF [21], and (c) represents our NeuralSRNF method. We show the surfaces for (b) and (c) with the error plotted as a heatmap. As we can see, (c) is able to accurately represent the surfaces. The pointwise error is between 0 and 0.1.

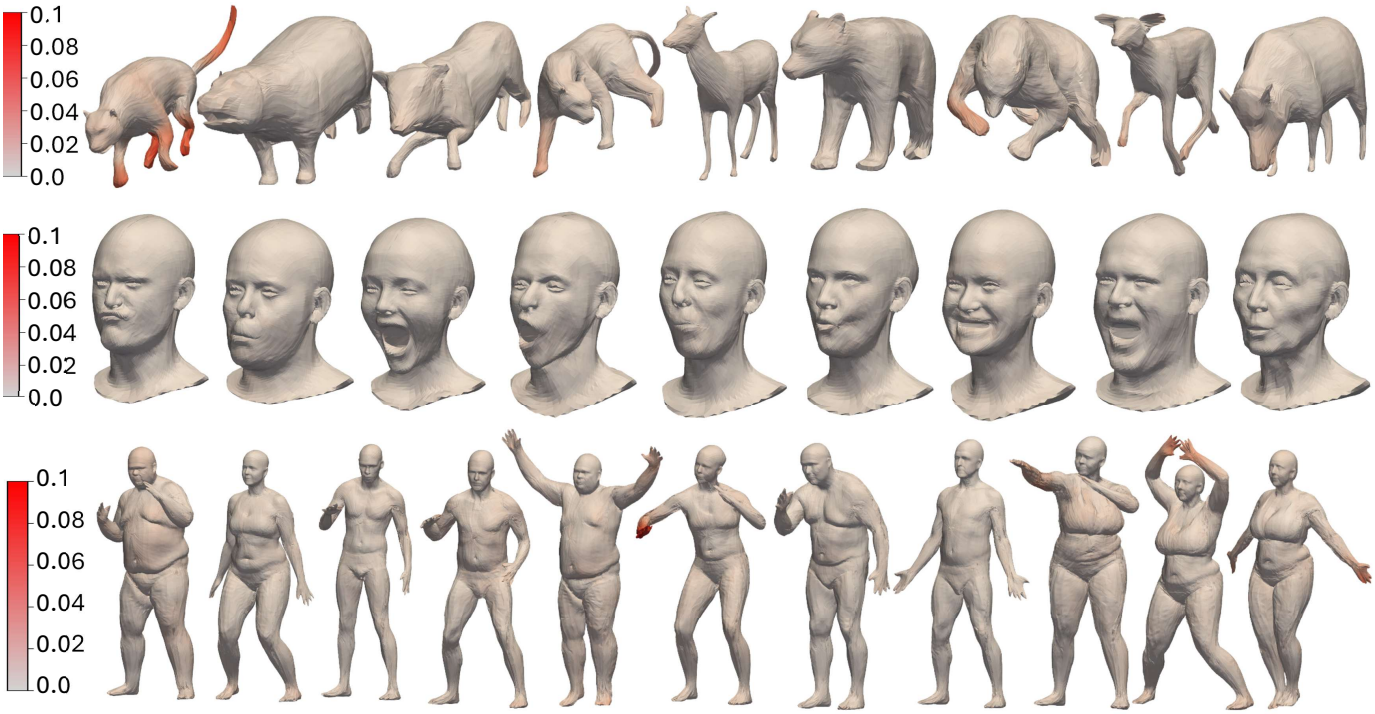
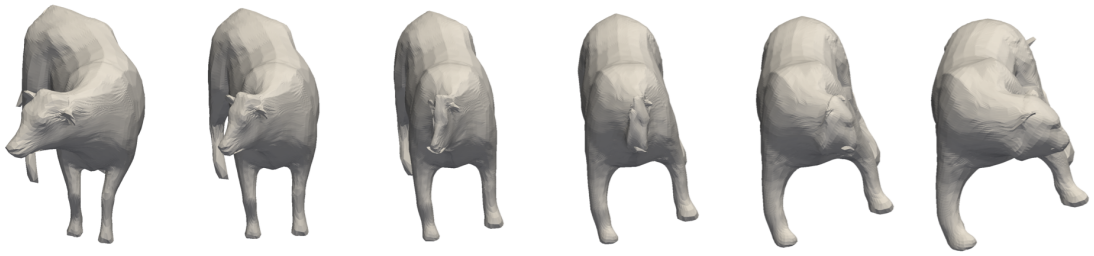
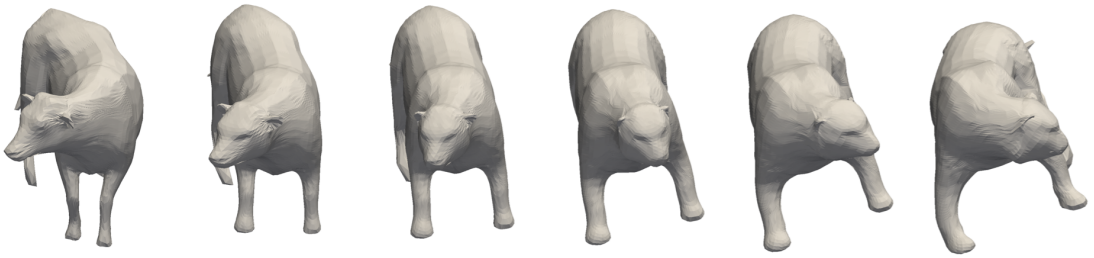


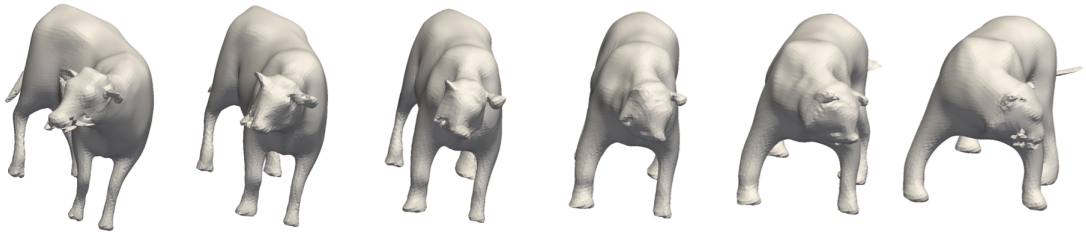
Fig. 25. Additional qualitative results of the proposed NeuralSRNF inversion on animals, faces, and humans. We show the error as heatmap following Figure 3 approach.



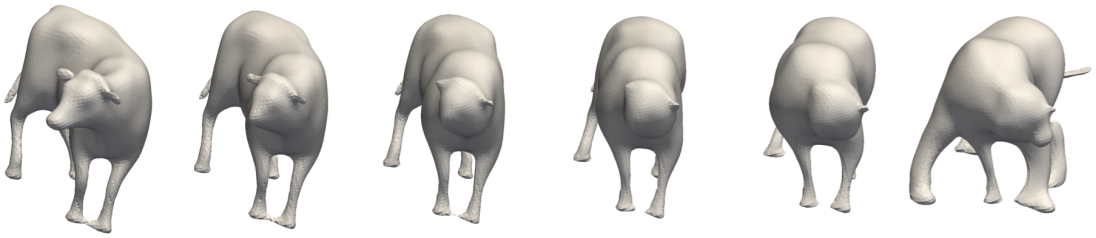
(a) Linear path.



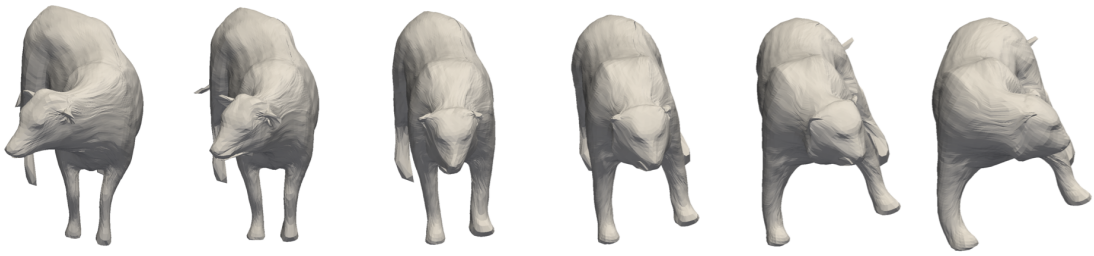
(b) Geodesic using numerical SRNF [21] (118 secs).



(c) Geodesic using [37] (authors reported a time of 20 mins, time taken on our machine is 2 hours).



(d) Geodesic using 4Deform [38] (authors reported a time of 7 mins, time taken on our machine is 1 hour).

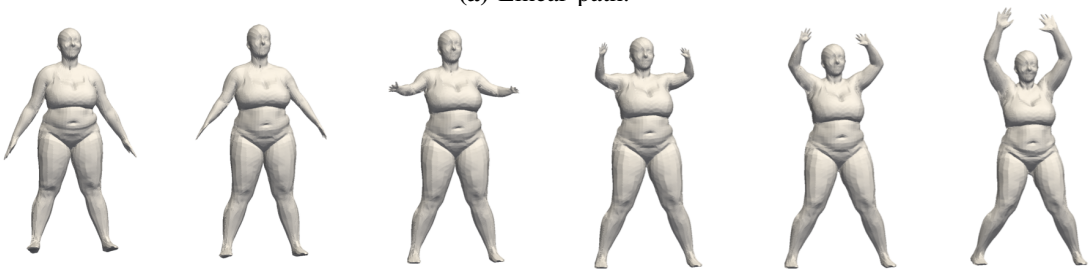


(e) Geodesic using our NeuralSRNF inversion (2 seconds).

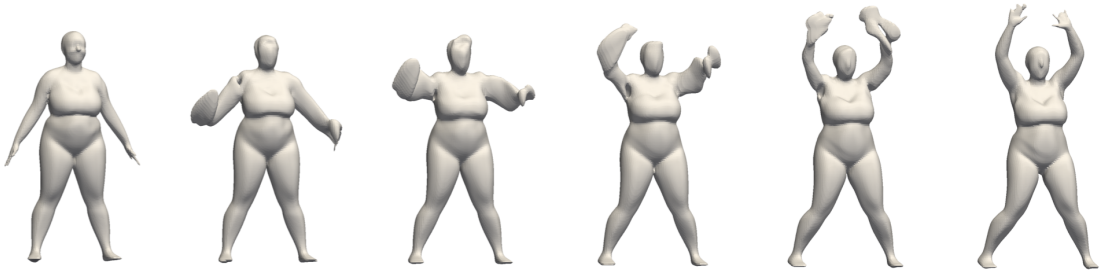
Fig. 26. Comparison of the geodesic computed using the proposed NeuralSRNF with state-of-the-art methods. Observe that the intermediate surfaces of our approach looks more accurate and natural than other methods. More importantly, our approach has more than 50 times speedup compare to numerical SRNF [21].



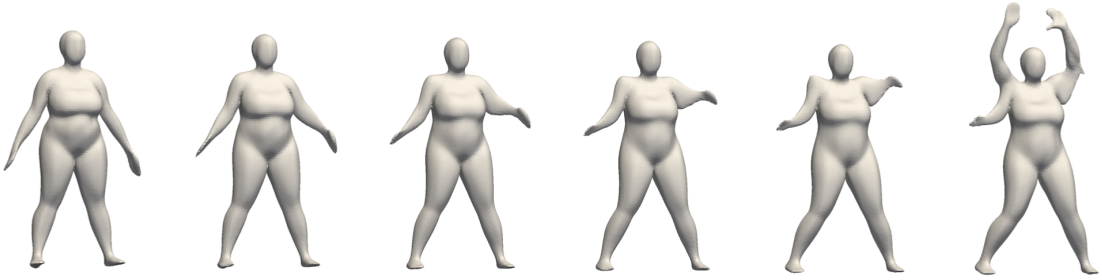
(a) Linear path.



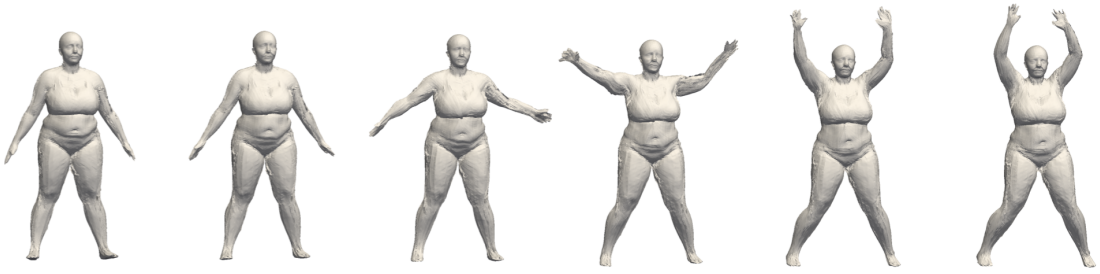
(b) Geodesic using numerical SRNF [21] (818 secs).



(c) Geodesic using [37] (authors reported a time of 20 mins, time taken on our machine is 2 hours).

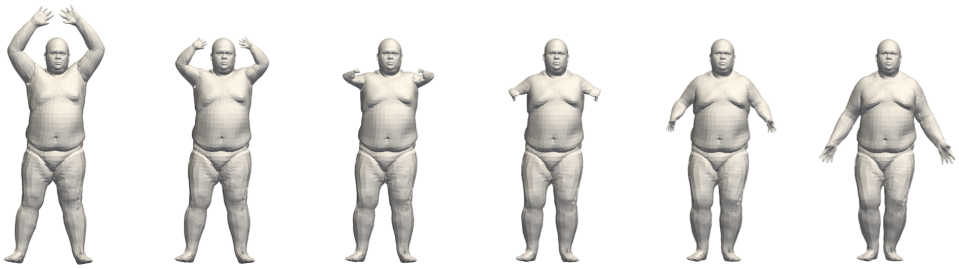


(d) Geodesic using 4Deform [38] (authors reported a time of 7 mins, time taken on our machine is 1 hour).

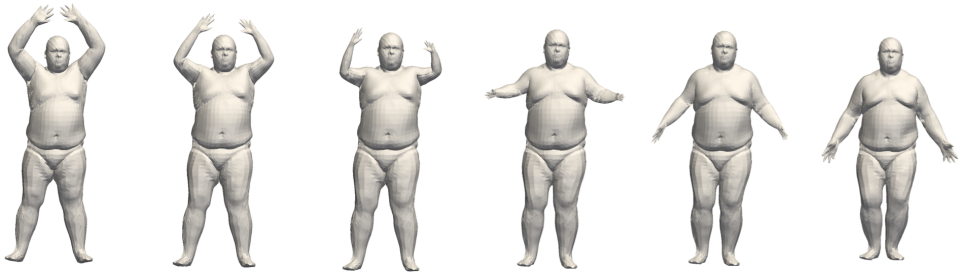


(e) Geodesic using our NeuralSRNF inversion (3 secs).

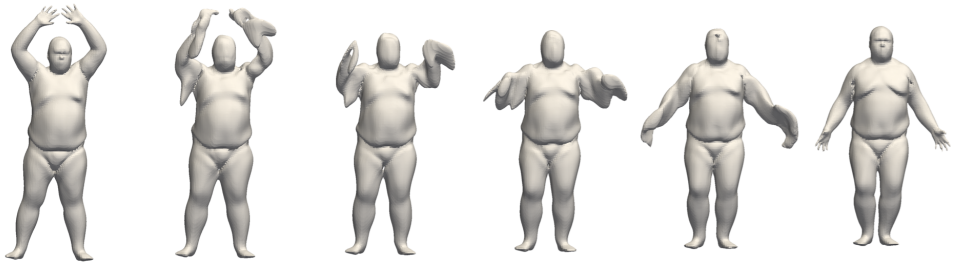
Fig. 27. Comparison of the geodesic computed using the proposed NeuralSRNF with state-of-the-art methods. Observe that the intermediate surfaces of our approach looks more accurate and natural than other methods.



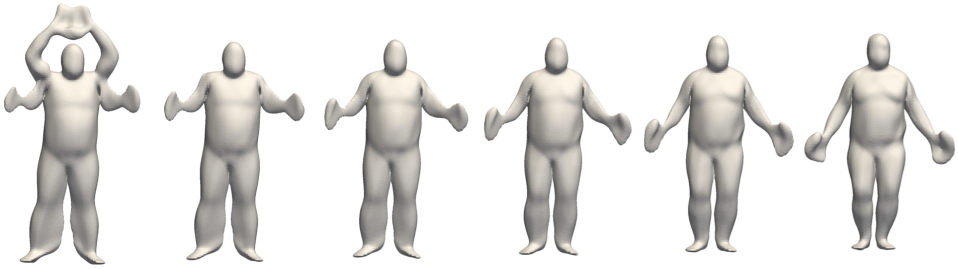
(a) Linear path.



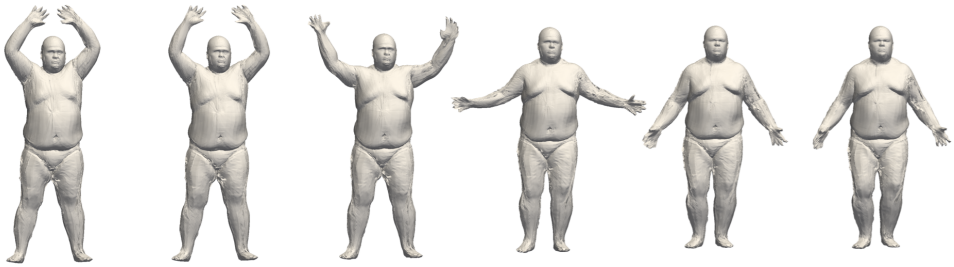
(b) Geodesic using numerical SRNF [21] (818 secs).



(c) Geodesic using [37] (authors reported a time of 20 mins, time taken on our machine is 2 hours).

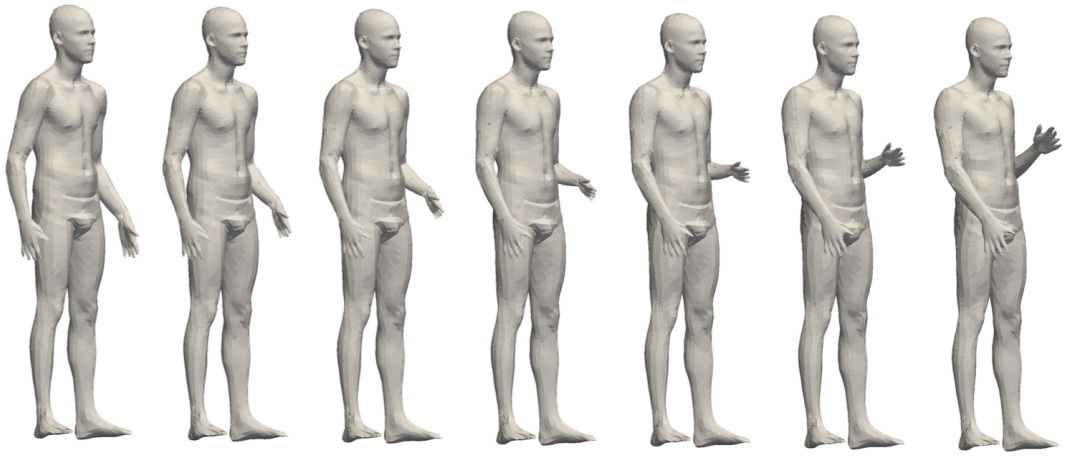


(d) Geodesic using 4Deform [38] (authors reported a time of 7 mins, time taken on our machine is 1 hour).

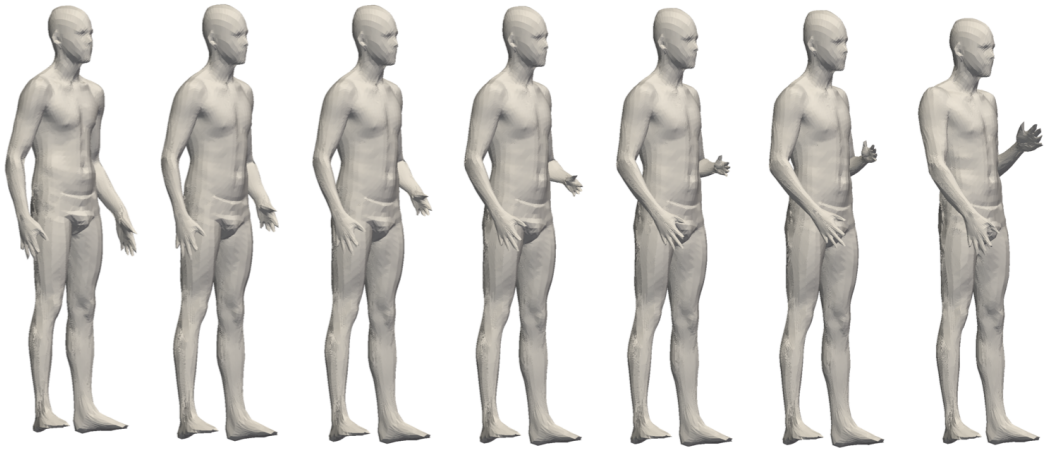


(e) Geodesic using our NeuralSRNF inversion (4 secs).

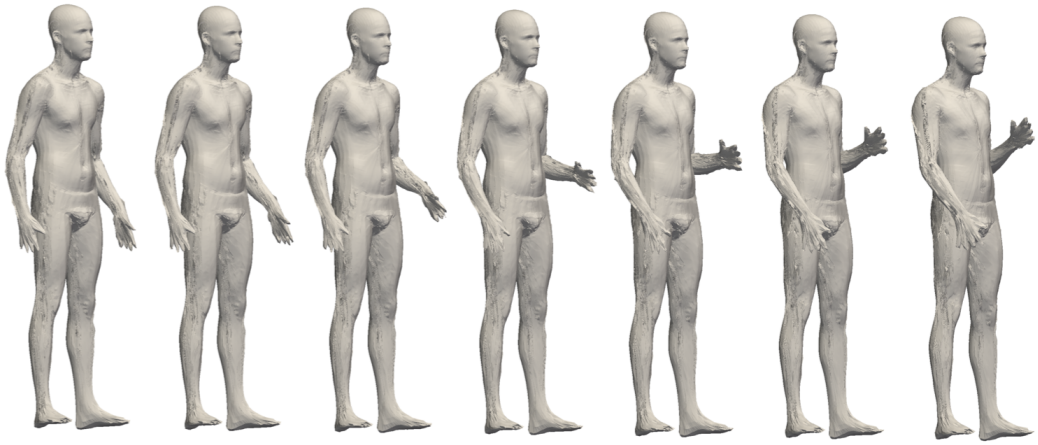
Fig. 28. Comparison of the geodesic computed using the proposed NeuralSRNF with state-of-the-art methods. Observe that the intermediate surfaces of our approach looks more accurate and natural than other methods. More importantly, our approach is computationally more efficient and compute geodesic in 4 secs.



(a) Linear path.

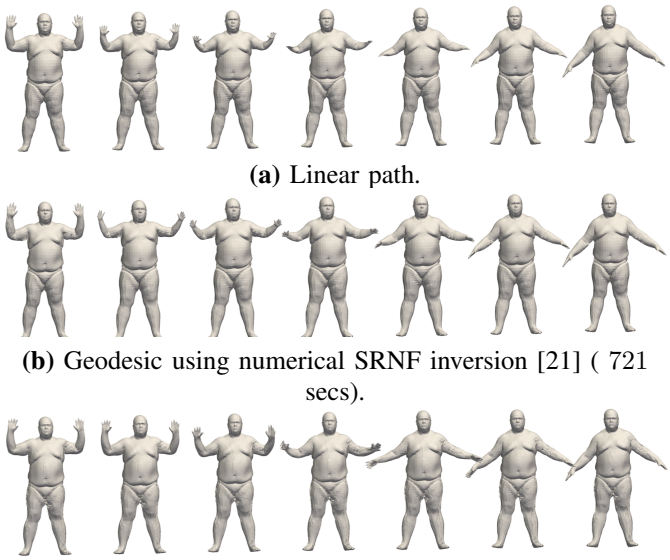


(b) Geodesic of numerical SRNF inversion [21] (600 secs).



(c) Geodesic of the proposed NeuralSRNF (4 secs).

Fig. 29. Geodesic comparison of our method with linear and numerical SRNF methods [21]. Here **(c)** shows our geodesic path, **(b)** shows the geodesic obtained using numerical SRNF, and **(a)** shows the linear path.

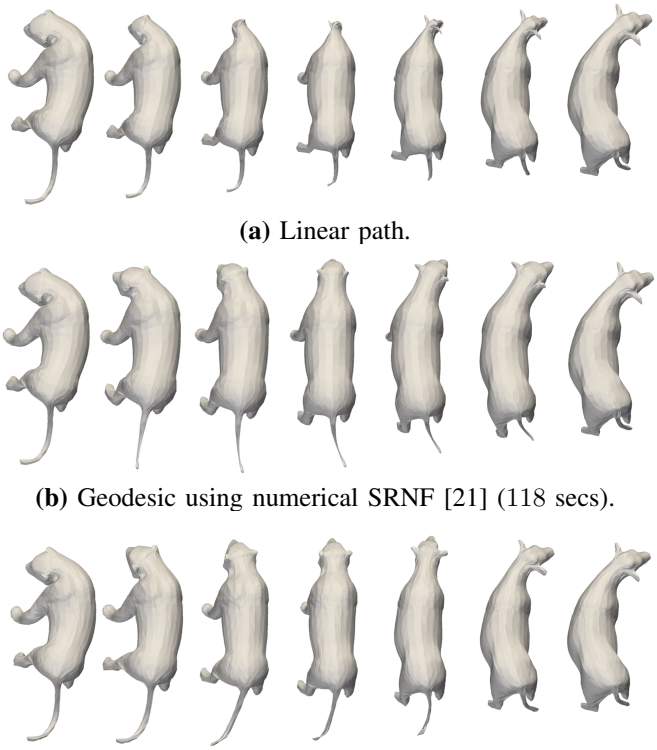


(a) Linear path.

(b) Geodesic using numerical SRNF inversion [21] (721 secs).

(c) Geodesic using the proposed NeuralSRNF (6 secs).

Fig. 30. Geodesics comparison of the proposed SRNF inversion with linear and numerical SRNF [21].

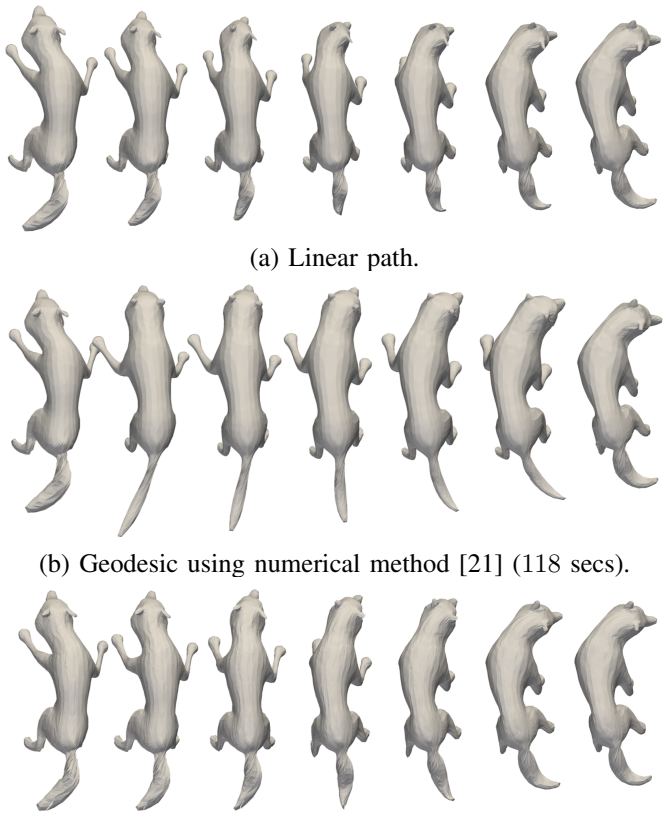


(a) Linear path.

(b) Geodesic using numerical SRNF [21] (118 secs).

(c) Geodesic using our NeuralSRNF inversion (2 secs).

Fig. 31. Geodesic results of Figure 6 but shown from a different viewing angle.

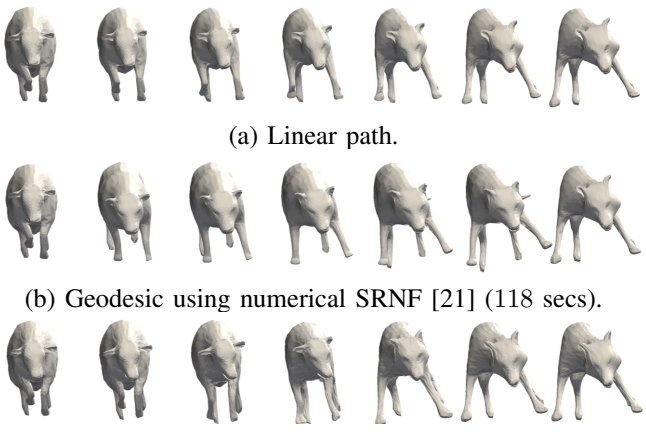


(a) Linear path.

(b) Geodesic using numerical method [21] (118 secs).

(c) Geodesic using our NeuralSRNF (2 secs).

Fig. 32. We compare the geodesics between different methods from the top view. Notice that the head movement in **(c)** looks more natural compared to **(a)** linear and **(b)** numerical.

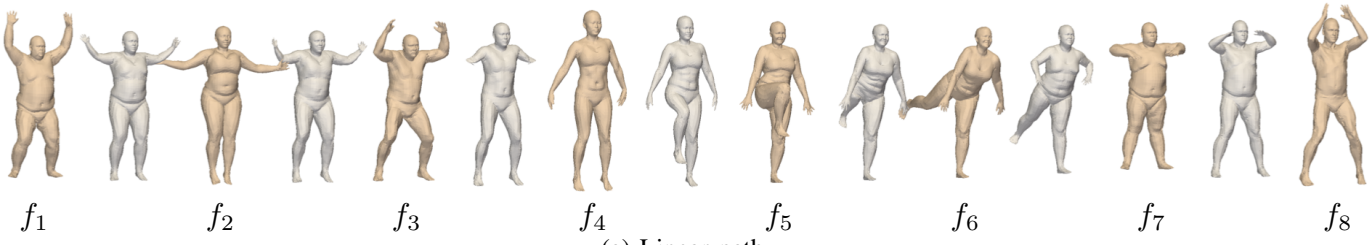


(a) Linear path.

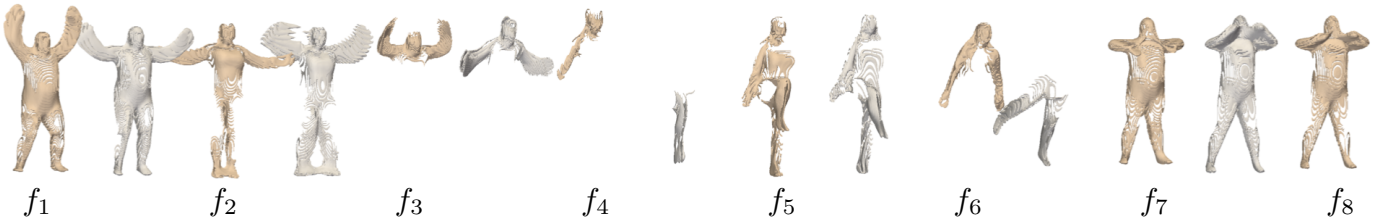
(b) Geodesic using numerical SRNF [21] (118 secs).

(c) Geodesic using the proposed NeuralSRNF (2 secs).

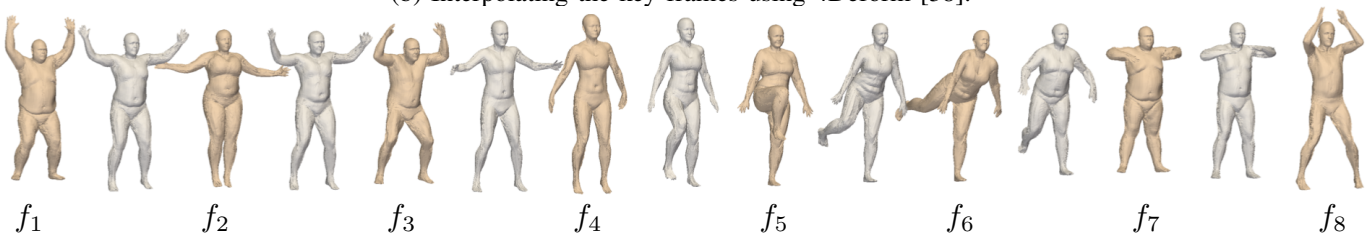
Fig. 33. Geodesics comparison of the proposed SRNF inversion with linear and numerical SRNF [21].



(a) Linear path.

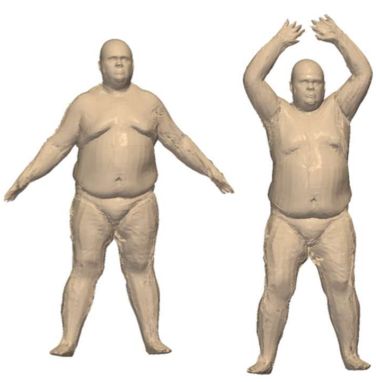


(b) Interpolating the key frames using 4Deform [38].

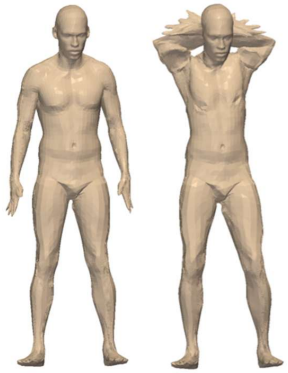


(c) Interpolating the key frames using our proposed NeuralSRNF.

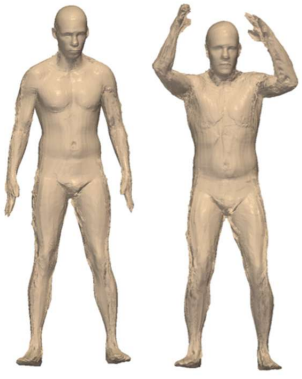
Fig. 34. We show the interpolation of key frame obtained using the proposed NeuralSRNF and compare it with linear interpolation and interpolation with 4Deform [38]. Observe that the intermediate frames generated using 4Deform [38] suffers from artifacts and have missing regions.



(a)
Source

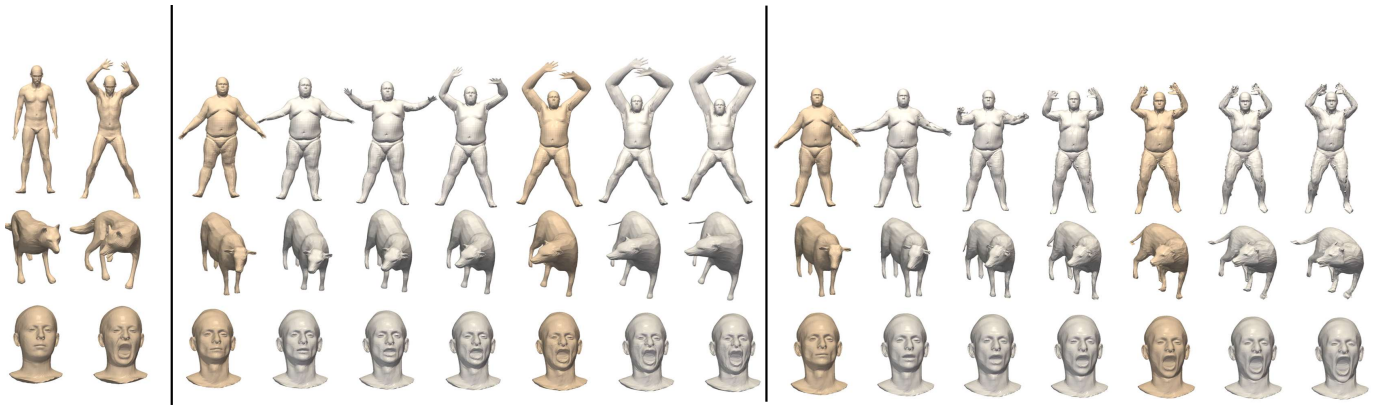


(b)
Target deformation in original space



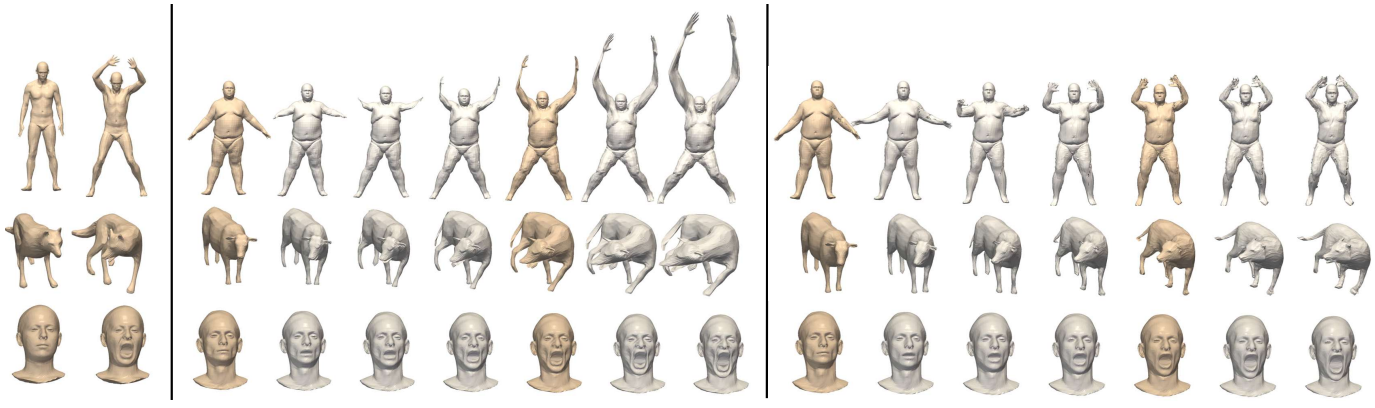
(c)
Target deformation in SRNF space

Fig. 35. Given f_1, h_1, and f_2, we compute the deformation $v = h_1 - f_1$ and apply it to f_2 in order to generate h_2. Observe that the deformation generated in **(b)** has unnatural shrinkage of the arms, while **(c)**, deformation looks natural and matches the source deformation.



(a) Source. (b) Using numerical SRNF [21]. (c) Using the proposed method.

Fig. 36. Examples of deformation transfer for human, face, and animal. **(a)** The source deformation, **(b)** deformation transfer by numerical SRNF [21] interpolation and extrapolation in the space of SRNFs, and **(c)** deformation transfer by linear interpolation and extrapolation in the space of SRNFs $\mathcal{Q}$ followed by SRNF inversion for visualization. This shows additional results of numerical SRNF [21] of Figure 10 of the main manuscript.



(a) Source. (b) Using linear deformation. (c) Using the proposed method.

Fig. 37. Examples of deformation transfer for human, face, and animal. **(a)** The source deformation, **(b)** deformation transfer by linear interpolation and extrapolation in the original space of surface, and **(c)** deformation transfer by linear interpolation and extrapolation in the space of SRNFs $\mathcal{Q}$ followed by SRNF inversion for visualization. This shows additional deformation transfer results of Figure 10 of the main manuscript.

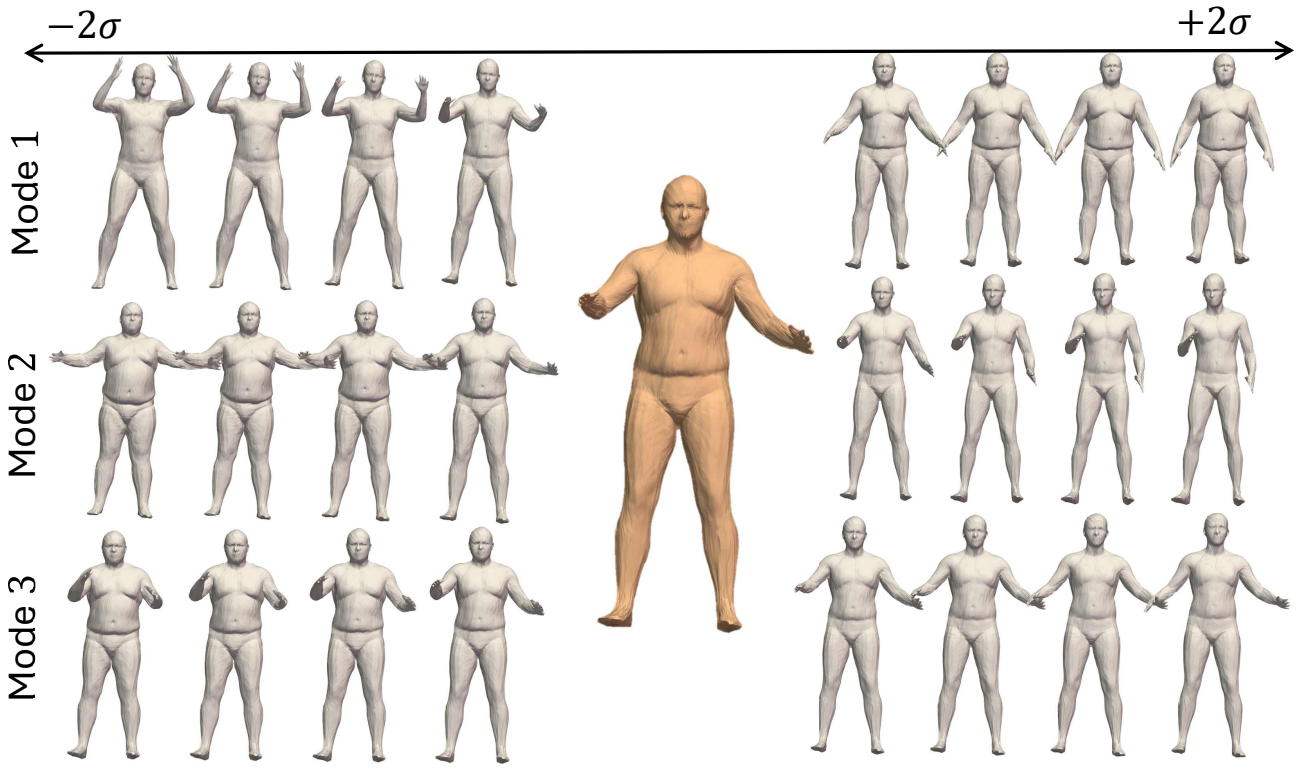
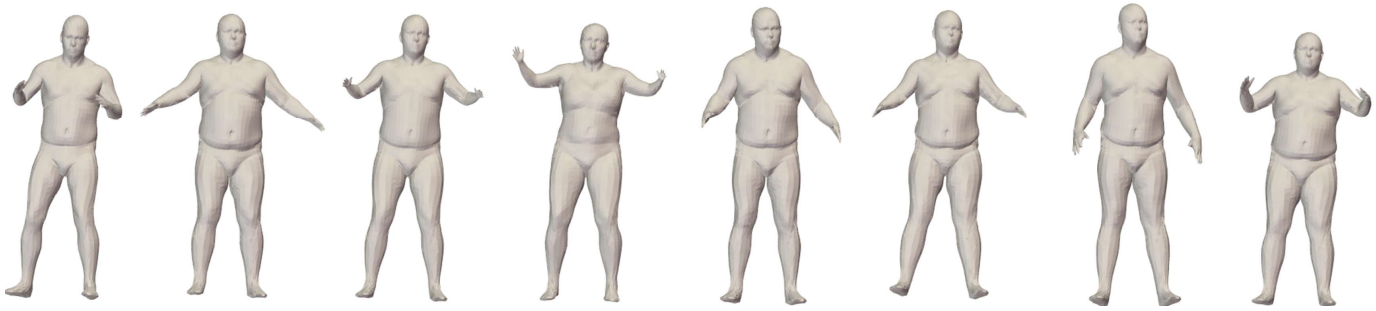
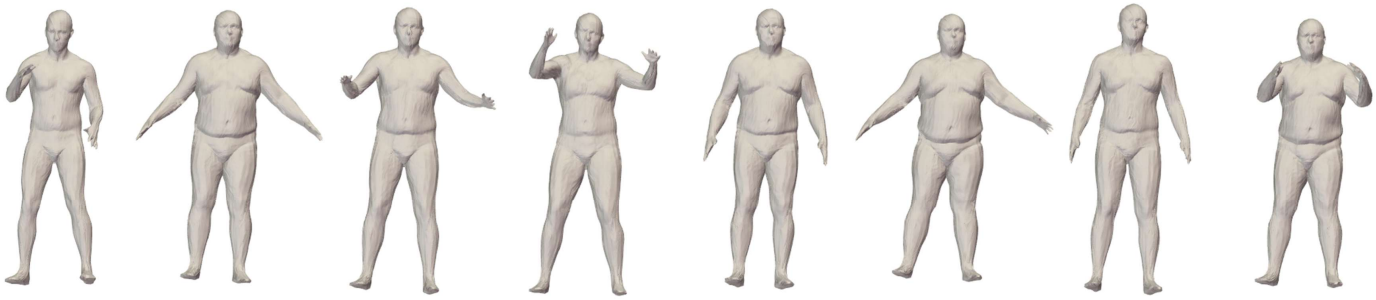


Fig. 38. Summary statistics of humans computed in the SRNF space inverted using the proposed NeuralSRNF.



(a) Using numerical SRNF [21].



(b) Using our NeuralSRNF.

Fig. 39. 3D shape generation using **(a)** numerical SRNF [21] and **(b)** our NeuralSRNF. This shows additional shape generation of Figure 13 of the main manuscript.

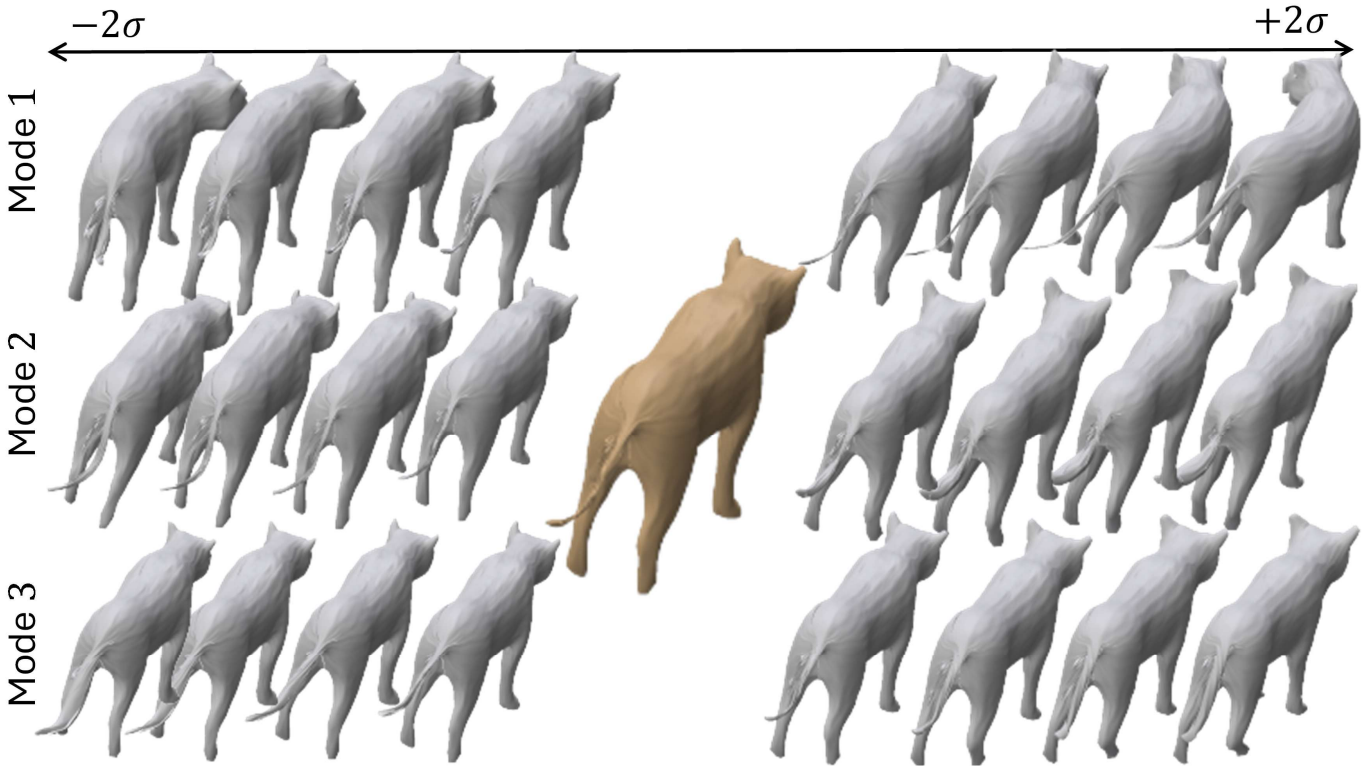


Fig. 40. View 2 of summary statistics computed in the SRNF space using PCA and inverted using the proposed SRNF method.

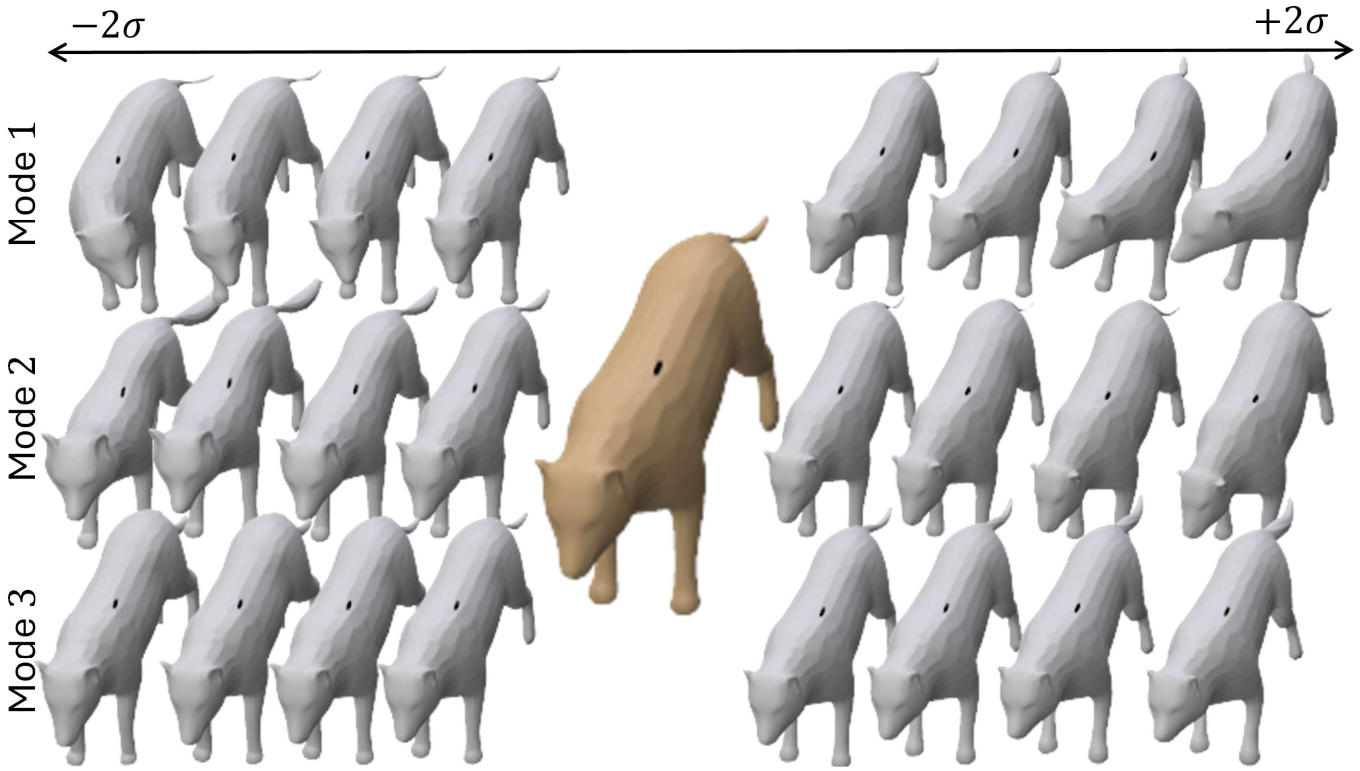


Fig. 41. View 1 of shape statistics computed in the SRNF space using Principal Component Analysis (PCA) and inverted using numerical SRNF [21].

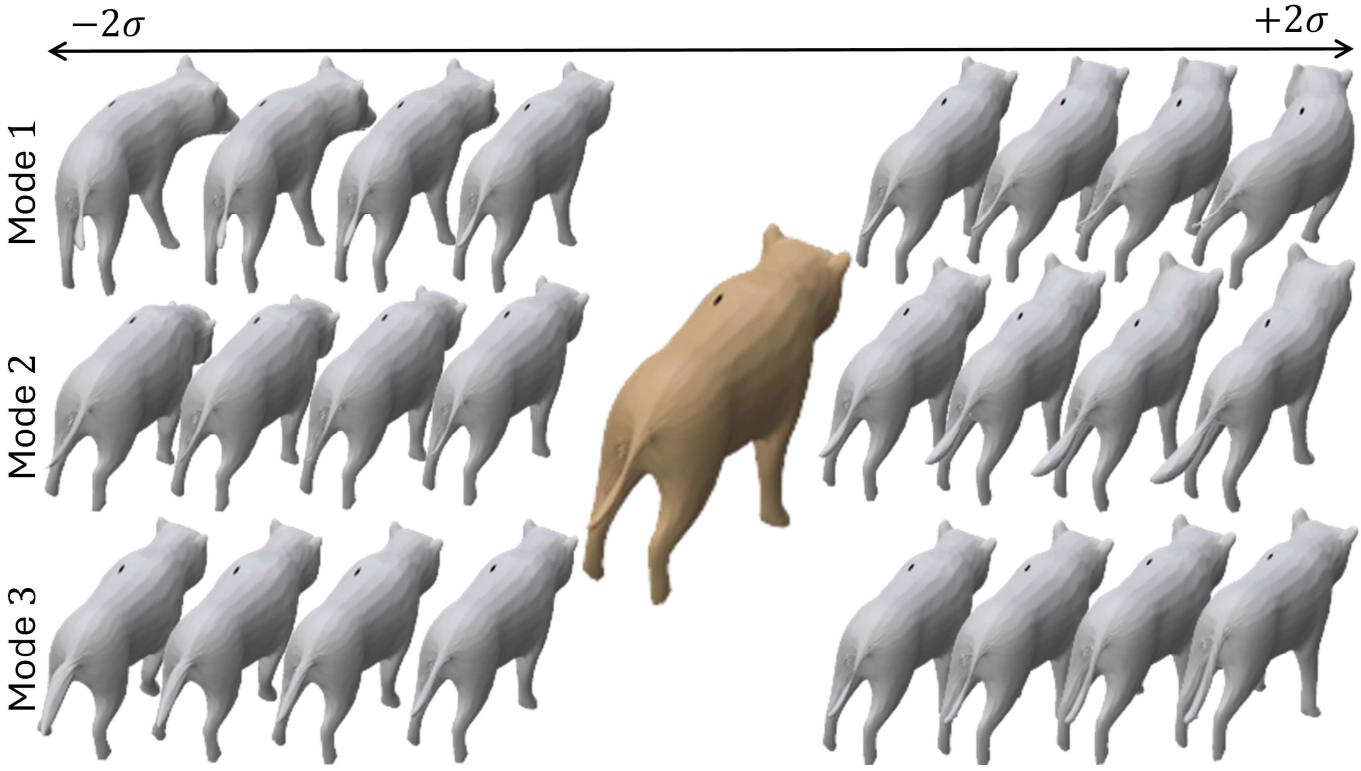


Fig. 42. View 2 of summary statistics computed using numerical SRNF inversion [21] method.

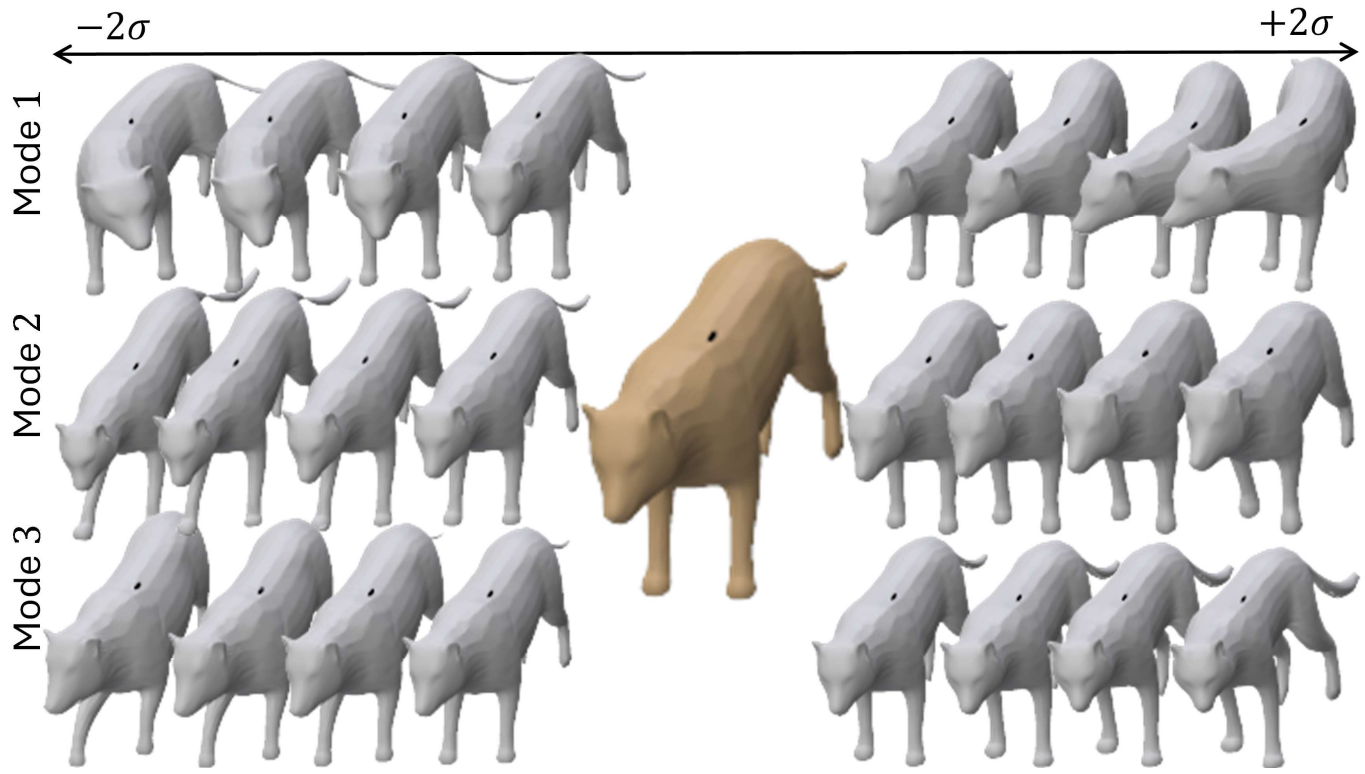


Fig. 43. View 1 of linear shape statistics computed in the original space of surfaces using Principal Component Analysis (PCA). We show the three modes of variations with the standard deviations of $\pm 2\sigma$.

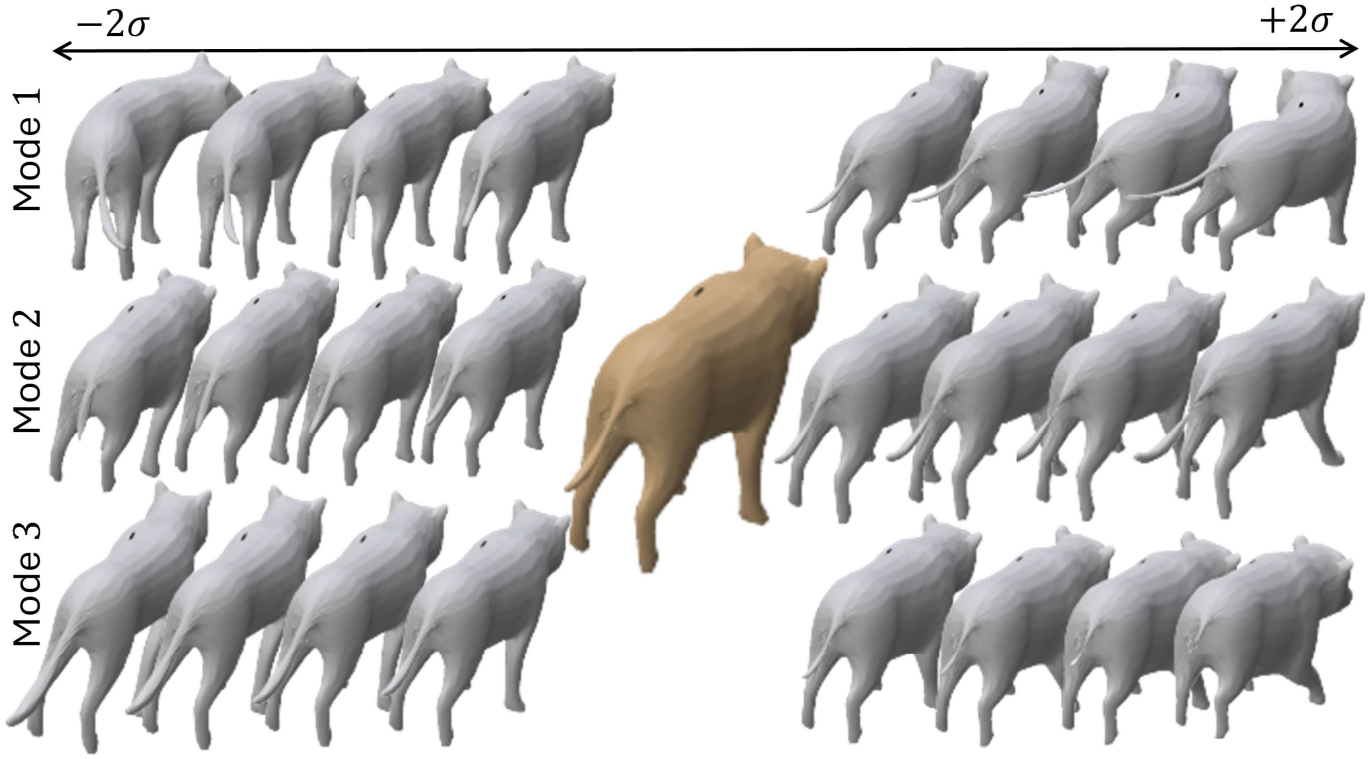


Fig. 44. View 2 of linear shape statistics computed in the original space of surfaces $\mathcal{F}$.

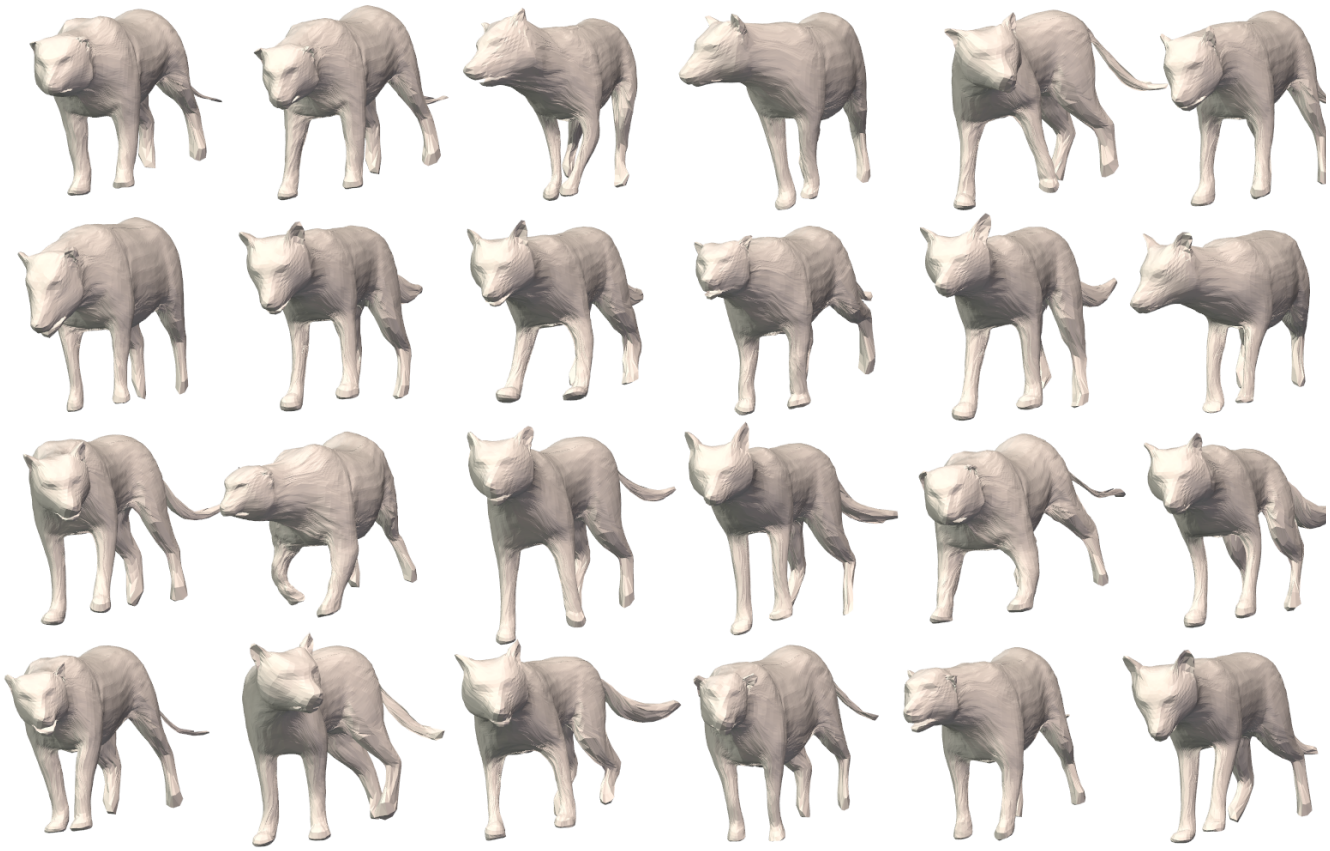


Fig. 45. Here we show 24 randomly generated shapes of 3D animals using random sampling of principal modes in the SRNF space and inverted to the surface space using the proposed NeuralSRNF inversion.



4D geodesic using NeuralSRNF before temporal registration.



4D geodesic using NeuralSRNF after temporal registration.

Fig. 46. Example of the temporal registration and geodesic generated before and after temporal alignment of source and target 4D surfaces using the proposed NeuralSRNF framework. The mean 4D surface is highlighted with a black rectangle. Note that the geodesics look more natural and are generated within 15 s, which is 180 times faster than the numerical SRNF [21]; see the Supplementary Material for additional results.



(a) Source.



(b) Target before temporal registration.



(c) Target after temporal registration.



(d) Mean (with NeuralSRNF, after temporal registration, time taken 15 s).



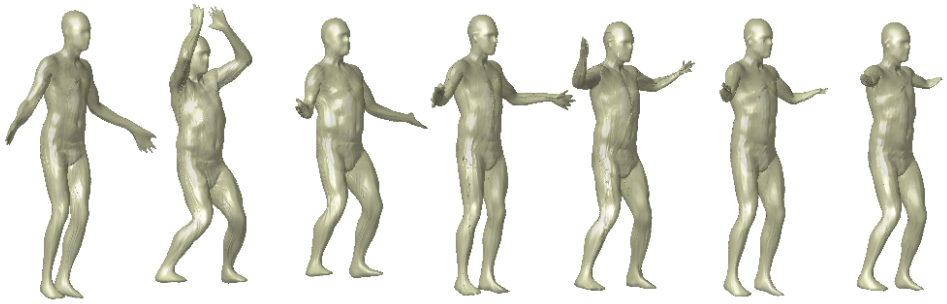
(e) Mean (with NeuralSRNF, before temporal registration, time taken 15 s).



(f) Mean (with linear interpolation, after temporal registration).



(g) Mean (with linear interpolation, before temporal registration).



(a) Source | (b) Target unregistered | (c) Target registered | (d) Ours registered | (e) Ours unregistered | (f) Linear registered | (g) Linear unregistered

Fig. 47. Comparison of geodesic quality of mean 4D surface obtained using the linear interpolation and the proposed NeuralSRNF. The bottom row highlights a 3D instance from each 4D sequence.