TU Delft

Delft University of Technology

Document Version
Final published version

Citation (APA)
Uzolas, L. (2026). *Efficient 3D Deformation Learning with Neural Representations*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:7e5f1c01-4102-429e-92ea-886e5cff3378

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

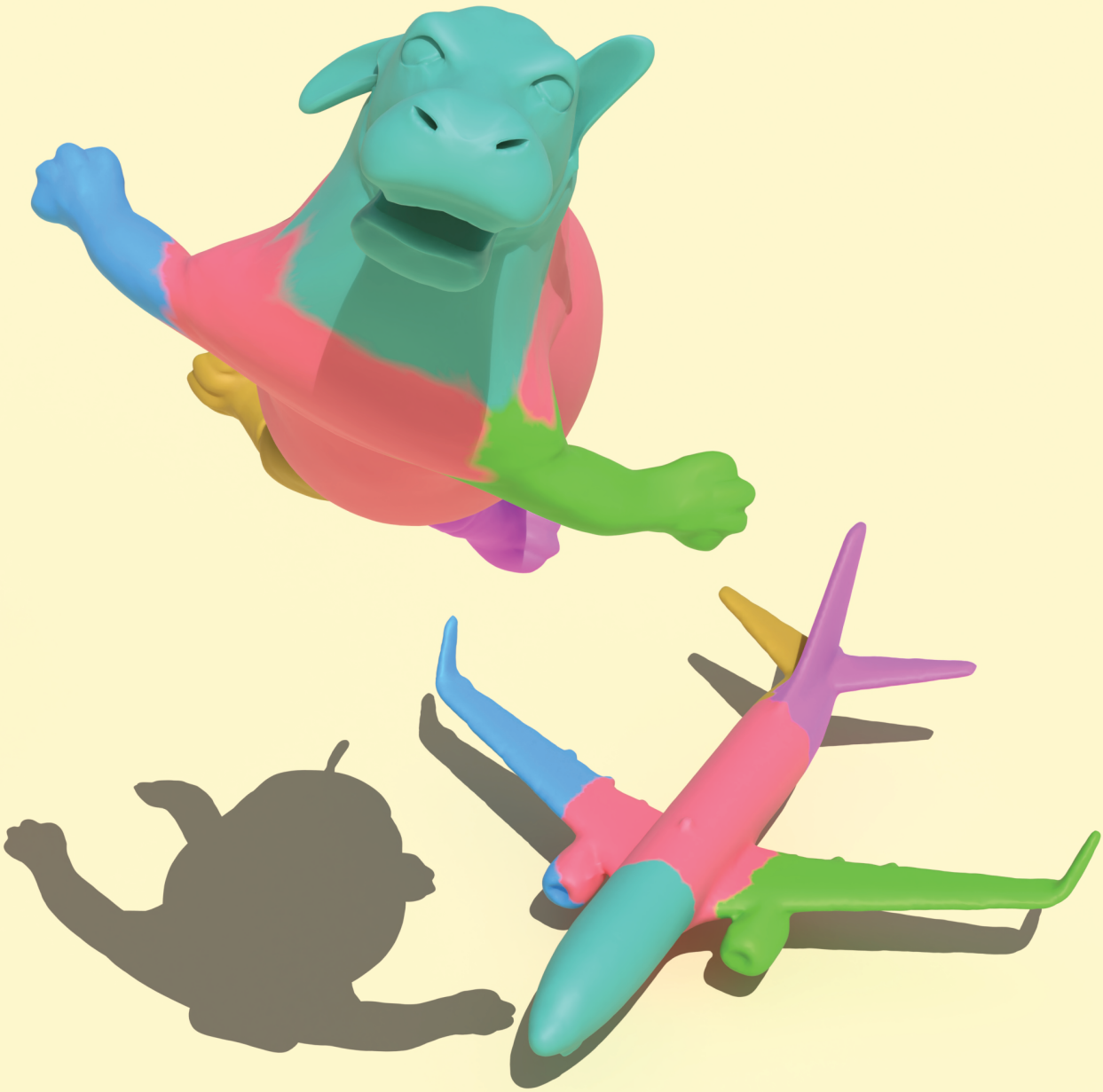
Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

Efficient 3D Deformation Learning with Neural Representations



Lukas Uzolas

Efficient 3D Deformation Learning with Neural Representations

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus,
Prof.dr.ir. H. Bijl,
chair of the Board for Doctorates
to be defended publicly on
Wednesday, 16th of September 2026 at 17:30.

by

Lukas UZOLAS

This dissertation has been approved by the (co)promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof.dr. E. Eisemann,	Delft University of Technology (NL), promotor
Dr. K.A. Hildebrandt,	Delft University of Technology (NL), promotor
Dr. P. Kellnhofer,	Delft University of Technology (NL), copromotor

Independent members:

Prof.dr.ir. W.P. Brinkman	Delft University of Technology (NL)
Prof.dr. M. Nießner	Technical University of Munich (Germany)
Prof.dr. Z. Lähner	University of Bonn (Germany)
Dr. Z. Yumak	Utrecht University (NL)
Prof.dr. O.E. Scharenborg	Delft University of Technology (NL), reserve member





Keywords: 3D Animation, Mesh Deformations, Neural Representations, Semantic Features, Neural Rendering

Printed by Ridderprint
Cover by Lukas Uzolas, incorporating meshes from:
DeformingThings4D [Li et al. 2021], *SMAL* [Zuffi et al. 2017], *Shapenet* [Chang et al. 2015], *SMPL* [Loper et al. 2015]
Copyright © 2026 by L. Uzolas, and Chapter 3 © 2025 IEEE

An electronic version of this dissertation is available at
https://repository.tudelft.nl/.

Summary

Being able to deform and manipulate 3D representations brings static objects to life in movies, video games, and other digital media. Recent advances in neural 3D representations have had a great impact on the field. However, animating and deforming them remains challenging, particularly due to the scarcity of training data in the 3D domain. This dissertation addresses these challenges along two directions: animating neural 3D representations and leveraging neural features from pre-trained 2D foundation models for 3D downstream tasks.

Chapter 2 introduces a method for reconstructing an animatable 3D neural radiance field from multi-view video. While previous methods reconstruct dynamic neural radiance fields via ray bending, we forward-warp an explicit sparse point cloud using a kinematic model, thereby avoiding projection ambiguities. We first initialize a sparse neural feature point cloud and an overparameterized kinematic skeleton using the medial axis transformation. The skeleton drives the point cloud deformation via linear blend skinning, after which the result is rendered via volumetric rendering, jointly optimizing the feature point cloud and deformation parameters. After training, the kinematic skeleton can be pruned, enabling easy manual animation of the reconstructed representation.

Chapter 3 explores how to animate 3D meshes from text prompts in a zero-shot setting, without training any deformation models. To this end, we embed 3D meshes into the latent feature space of pre-trained video diffusion models by re-projecting semantic neural features onto the mesh surface. We then optimize the mesh deformation to match a generated latent video, which is more robust than fitting to RGB outputs. In contrast to related work, our method produces plausible animation sequences in minutes, demonstrating the power of these features for 3D downstream tasks.

Chapter 4 further investigates how to improve semantic neural features from 2D foundation models for 3D applications that rely on establishing correspondences between meshes. We address the issue that these features struggle to differentiate between instances of the same semantic class, such as “left hand” versus “right hand”. We learn an embedding space that disambiguates such instances using the geodesic distance on the mesh surface, while preserving the original semantic information. The method is purely self-supervised, requires only a small number of unpaired training meshes, and we show that the resulting surface-aware features significantly improve performance on several 3D tasks.

Chapter 5 investigates a rotation-equivariant formulation of Neural Jacobian Fields, a method for learning detail-preserving mesh deformations in the gradient domain. In the previous chapter, we showed that semantic features can condition a Neural Jacobian Field to learn generalized deformations across shape categories. However, Neural Jacobian Fields are inherently tied to the orientation of training shapes in ambient space, requiring inference shapes to be aligned to the training data. We address this limitation by introducing a rotation-equivariant formulation that produces consistent results independent of input orientation, while generalizing across diverse shapes.

Samenvatting

De mogelijkheid om 3D-weergaven te vervormen en te manipuleren brengt statische objecten tot leven in films, videogames en andere digitale media. Recente ontwikkelingen op het gebied van neurale 3D-representaties hebben een grote impact gehad op het vakgebied. Het animeren en vervormen ervan blijft echter een uitdaging, met name vanwege de schaarste aan trainingsdata in het 3D-domein. Dit proefschrift pakt deze uitdagingen aan vanuit twee invalshoeken: het animeren van neurale 3D-representaties en het benutten van neurale kenmerken uit vooraf getrainde 2D-basismodellen voor 3D-downstreamtaken.

Hoofdstuk 2 introduceert een methode voor het reconstrueren van een animeerbaar 3D-Neural Radiance Fields op basis van video-opnames vanuit meerdere gezichtspunten. Terwijl eerdere methoden dynamische *Neural Radiance Fields** reconstrueren via straalbuiging, passen wij een voorwaartse vervorming toe op een expliciete, schaarse *Point Cloud* met behulp van een kinematisch model, waardoor projectie-dubbelzinnigheden worden vermeden. We initialiseren eerst een sparse neurale kenmerk *Point Cloud* en een overgeparametriseerd kinematisch skelet met behulp van de *Medial Axis Transform*. Het skelet stuurt de vervorming van de *Point Cloud* aan via lineaire blend-skinning, waarna het resultaat wordt weergegeven via volumetrische rendering, waarbij de kenmerk *Point Cloud* en de vervormingsparameters gezamenlijk worden geoptimaliseerd. Na training kan het kinematische skelet worden vereenvoudigt, waardoor eenvoudige handmatige animatie van de gereconstrueerde weergave mogelijk wordt.

Hoofdstuk 3 onderzoekt hoe 3D-meshes kunnen worden geanimeerd op basis van tekstprompts in een zero-shot-omgeving, zonder dat er vervormingsmodellen hoeven te worden getraind. Daartoe integreren we 3D-meshes in de latente kenmerkruimte van vooraf getrainde videodiffusiemodellen door semantische neurale kenmerken opnieuw op het oppervlak van de mesh te projecteren. Vervolgens optimaliseren we de meshvervorming zodat deze overeenkomt met een gegenereerde latente video, wat robuuster is dan het aanpassen aan RGB-uitvoer. In tegenstelling tot verwant werk produceert onze methode binnen enkele minuten aannemelijk animatiesequenties, waarmee de kracht van deze kenmerken voor 3D-downstreamtaken wordt aangetoond.

Hoofdstuk 4 onderzoekt verder hoe semantische neurale kenmerken uit 2D-basis modellen kunnen worden verbeterd voor 3D-toepassingen die afhankelijk zijn van het vaststellen van overeenkomsten tussen meshes. We lossen het probleem op waarbij deze kenmerken geen onderscheid kunnen maken tussen instanties van dezelfde semantische klasse, zoals “linkerhand” versus “rechterhand”. We leren een inbeddingsruimte die dergelijke instanties onderscheidt met behulp van de geodetische afstand op het mesh-oppervlak, terwijl de oorspronkelijke semantische informatie behouden blijft. De methode is puur zelf-gesuperviseerd, vereist slechts een klein aantal ongepaarde trainingsmeshes, en we tonen aan dat de resulterende oppervlakbewuste kenmerken de prestaties bij verschillende 3D-taken aanzienlijk verbeteren.

Hoofdstuk 5 onderzoekt een rotatie-equivariante formulering van *Neural Jacobian Fields*, een methode voor het leren van detailbehoudende mesh-vervormingen in het gradiënt-

* Engelse termen die gebruikelijk zijn binnen dit vakgebied worden cursief weergegeven.

domein. In het vorige hoofdstuk hebben we aangetoond dat semantische kenmerken een Neural Jacobian Field kunnen sturen om gegeneraliseerde vervormingen te leren over verschillende vormcategorieën heen. *Neural Jacobian Fields* zijn echter inherent gebonden aan de oriëntatie van trainingsvormen in de omringende ruimte, waardoor inferentievormen moeten worden uitgelijnd met de trainingsgegevens. We pakken deze beperking aan door een rotatie-equivariante formulering te introduceren die consistente resultaten oplevert, onafhankelijk van de invoeroriëntatie, terwijl er wordt gegeneraliseerd over diverse vormen.

Contents

Contents **8**

1 Introduction **1**
1.1 Kinematic Neural Point Clouds . . . 2
1.2 Animating 3D Representations with 2D Foundation Models . . . 2
1.3 Disambiguating Semantic Features . . . 3
1.4 Rotation-Equivariant Neural Jacobian Fields . . . 3

2 Template-free Articulated Neural Point Clouds for Reposable View Synthesis **4**
2.1 Introduction . . . 5
2.2 Related Work . . . 6
2.3 Preliminaries . . . 7
2.4 Method . . . 8
2.4.1 Canonical Feature Point Cloud . . . 9
2.4.2 Kinematic Model . . . 9
2.4.3 Skeleton simplification . . . 10
2.4.4 Dynamic Point Rendering . . . 10
2.4.5 Losses . . . 11
2.4.6 An extension for local deformations . . . 12
2.5 Experiments . . . 13
2.5.1 Experiments Setup . . . 13
2.5.2 Results . . . 14
2.6 Discussion . . . 18

3 MotionDreamer: Exploring Semantic Video Diffusion features for Zero-Shot 3D Mesh Animation **20**
3.1 Introduction . . . 21
3.2 Related Work . . . 21
3.2.1 Video generation . . . 21
3.2.2 Shape and pose representations . . . 22
3.2.3 3D motion and animation . . . 22
3.3 Preliminaries . . . 23
3.3.1 Video Diffusion Models . . . 23
3.3.2 Semantic Diffusion Features . . . 23
3.4 Method . . . 24
3.4.1 Single-View Texturing . . . 24
3.4.2 Motion Generation . . . 25
3.4.3 Motion fitting . . . 25
3.4.4 Implementations Details . . . 26
3.5 Experiments . . . 27
3.5.1 User Study . . . 27
3.5.2 Pose Optimization . . . 29
3.5.3 Ablations . . . 30

3.6 Discussion . . . 32

4 Surface-Aware Distilled 3D Semantic Features **33**
4.1 Introduction . . . 34
4.2 Related Works . . . 34
4.2.1 Image-based features for 3D shapes . . . 35
4.2.2 Contrastive Learning . . . 35
4.2.3 Shape correspondences . . . 36
4.3 Preliminaries . . . 36
4.3.1 Reprojection of 2D Features . . . 36
4.4 Method . . . 37
4.4.1 Setup . . . 37
4.4.2 Separating Front Paw from Back Paw . . . 38
4.4.3 Implementation . . . 39
4.5 Experiments . . . 39
4.5.1 Exploration of Embedding Space . . . 40
4.5.2 One-shot Pose Transfer . . . 40
4.5.3 Skinning Weight Regression . . . 41
4.5.4 Point-to-Point Correspondence Matching . . . 41
4.5.5 Ablations . . . 43
4.6 Applications . . . 46
4.6.1 Instance-based Part Segmentation . . . 46
4.6.2 Pose Alignment . . . 47
4.6.3 Texturing . . . 47
4.7 Discussion . . . 48

5 Rotation-Equivariant Neural Jacobian Fields for Generalizable 3D Mesh Deformations **49**
5.1 Introduction . . . 50
5.2 Related works . . . 51
5.2.1 3D mesh deformations . . . 51
5.2.2 Semantic features . . . 52
5.2.3 Rotation-equivariant learning on surfaces . . . 52
5.3 Preliminaries and analysis . . . 53
5.4 Method . . . 55
5.4.1 Rotation-Equivariant Neural Jacobians . . . 55
5.4.2 Robust semantic features . . . 56
5.4.3 Learning relative pose deformations . . . 56
5.4.4 Training . . . 57
5.5 Experiments . . . 57
5.5.1 Rotation-equivariance . . . 58
5.5.2 Generalized setting . . . 58
5.5.3 Application: Inverse kinematics without rigging . . . 62
5.6 Conclusion . . . 64

6 Conclusion **66**
6.1 Challenges & Future Work . . . 66

Appendix **69**

A Appendix: Template-free Articulated Neural Point Clouds for Reposable View Synthesis **70**
A.1 Extra Results *Blender* dataset . . . 70
A.2 Extra Results *Robots* dataset . . . 72
A.3 Implementation Details . . . 74
A.3.1 Our method . . . 74
A.3.2 Validation . . . 74
A.4 Extra Results Ablations . . . 75
A.5 Notation . . . 75
A.6 Training Schedule for the *ZJU-MoCap* dataset . . . 76
A.7 Additional Animations . . . 76

B Appendix - MotionDreamer **78**
B.1 Additional Implementation Details . . . 78
B.1.1 Video Diffusion models . . . 78
B.1.2 Animation Models . . . 78
B.2 User Study . . . 80
B.2.1 Baseline methods . . . 80
B.2.2 Stimuli . . . 80
B.2.3 Instructions . . . 81
B.2.4 Detailed Results . . . 82
B.3 Pose Optimization Experiment Details . . . 86
B.3.1 Data . . . 86
B.3.2 Methodology . . . 86
B.3.3 WHAM baseline . . . 86
B.4 Additional Results . . . 87
B.4.1 Effect of Texturing and Background for Different VDMs . . . 87
B.4.2 Qualitative Comparison to Consistent4D . . . 87
B.4.3 Comparing Semantic Featuring against RGB for Optimization . . . 88
B.4.4 Ablation of Number of Vertices . . . 88
B.4.5 Results with Stable Video Diffusion . . . 88

C Appendix - Surface-Aware Distilled 3D Semantic Features **97**
C.1 Additional implementation details . . . 97
C.1.1 Diff3F . . . 97
C.1.2 Ablations . . . 97
C.2 One Shot Pose Transfer . . . 99
C.3 Pose Alignment . . . 99
C.4 Additional results . . . 100
C.4.1 Qualitative Results on SHREC'20 . . . 100
C.4.2 Qualitative Results on ShapeNet . . . 100
C.4.3 Texturing . . . 100
C.4.4 Using a Shared Encoder for All Shapes . . . 101
C.4.5 Ablating the Number of Training Shapes . . . 101

D Appendix - Rotation-Equivariant Neural Jacobian Fields for 3D Mesh Deformations **104**
D.1 Poisson equation with extrinsic gradient operator 104

Bibliography **104**

Acknowledgments **127**

Curriculum Vitae **128**

Publications **129**

1 Introduction

Movement and the perception of motion are central to how humans interact with their surroundings. Motion can have an active component, through which we engage with the physical world, and a passive component, through which we interpret it. Understanding how objects deform and move tells us about their physical properties and how they can be used: a tree sways in the wind, a door rotates around its hinges. Similarly, the movement of humans and animals can communicate their intent: a waving person is greeting us, a dog wagging its tail is happy. Modeling how to deform 3-dimensional representations is therefore at the heart of many real-world applications.

These applications have a long history in Computer Graphics, Computer Vision, and beyond, and can be used in both real-world and digital settings. In robotics and autonomous vehicle systems, it is crucial to know which objects are moving and how, in order to anticipate their behavior and act accordingly. In addition, pose estimation systems can be used to diagnose human motion or analyze wildlife behavior. On the other hand, digital media relies on animating digital representations and reconstructing real-world motion, for example, through motion capture systems. It enables immersive experiences and storytelling that can help bring digital characters come to life in video games and movies. However, several challenges present themselves when implementing such applications in practice.

One of these challenges is the choice of the 3D representation for reconstructing real-world objects. Historically, triangular meshes have established themselves as the standard in professional contexts, due to their ease of use and well-defined behavior for animation systems. However, meshes are problematic for 3D reconstruction, as they introduce discontinuities during optimization and cannot easily accommodate changes in topology. In contrast, recent advances in Neural Rendering have introduced novel representations such as Neural Radiance Fields (NeRFs) and 3D Gaussian Splatting (3DGS), which circumvent these issues while producing photo-realistic representations of real-world scenes. Yet animating such representations in a user-friendly manner is an open challenge.

A further challenge lies in improving the accessibility of 3D deformation and animation systems. Learning-based methods can significantly lower this barrier by offering low-dimensional controls for manipulating 3D representations, such as sliders (e.g., adjusting a human face from smiling to neutral), text prompts, or example-based deformations. However, training such methods requires a substantial amount of 3D data. In the 2D domain, by contrast, a surge of foundation models has been trained in a self-supervised manner on the vast amounts of image data available. Once trained, such a model can serve as a prior for many downstream tasks thanks to its generalizable internal representations. Equivalent models do not yet exist in the 3D domain due to the far smaller amount of 3D data available. There has therefore been a major push to exploit these 2D foundation models for 3D tasks such as animation.

This dissertation tackles these challenges along two directions: how to reconstruct and animate novel articulated representations, and how to leverage and improve 2D foundation model feature representations for 3D deformation tasks where training data is limited. To this end, we first learn how to reconstruct reposable 3D objects from videos

(Chapter 2). We then exploit video diffusion models to animate such 3D representations (Chapter 3). However, the internal feature representation of diffusion models can confuse semantically similar object parts, which we address in Chapter 4. We then propose learning deformations conditioned on these feature representations (Chapter 4). We find that these deformations generalize well between various shapes but require consistent coordinate frames, which we fix in Chapter 5.

1.1 Kinematic Neural Point Clouds

In the second chapter, we investigate how a hybrid, point-based Neural Radiance Field representation enables kinematic 3D reconstruction. Once reconstructed, the representation enables manual reanimation of the captured scene. Kinematic animation defines motion via a hierarchical skeleton, where rotating bone segments induce motion of a 3D representation through per-vertex skinning weights. The kinematic skeleton provides a low-dimensional way to deform a 3D representation, without the need to move each point individually.

Implicit neural representations such as NeRFs produce photo-realistic 3D reconstructions from sparse multi-view video of dynamic scenes. However, due to their implicit nature, deformations are modeled by backward warping query points within the volumetric rendering pipeline. While this produces high-quality outputs, it severely limits the ability to reanimate the scene at test time, in contrast to explicit kinematic deformations. We therefore combine forward kinematics with neural radiance fields by warping an explicit neural feature point cloud using linear blend skinning and rendering it via volumetric rendering. The proposed method then allows for manual re-animation of arbitrary kinematic objects after reconstruction.

1.2 Animating 3D Representations with 2D Foundation Models

In the third chapter, we investigate how to animate 3D representations using the motion prior of pre-trained video diffusion models (VDMs), without any training data available. The method operates on 3D meshes, though the representations could also originate from dynamic reconstruction pipelines, as in the previous chapter. While learning-based animation methods generalize well within a shape class, they typically require large 3D training datasets and fail to transfer across categories. Due to the ubiquity of training data available in the 2D domain, other methods explore how to leverage 2D priors for 3D reconstruction tasks. One such example is Score Distillation Sampling (SDS), which offers a class-agnostic alternative by distilling motion from a pre-trained VDM, but repeatedly querying the model results in optimization times of multiple hours for animations spanning only a few frames.

We propose a method that projects semantic features from a pre-trained VDM onto a 3D mesh, enabling the mesh to be fitted in the feature space rather than RGB space. This avoids the visual artifacts common in VDM outputs and enables class-agnostic mesh animation in minutes, with no additional training or class-specific templates required.

1.3 Disambiguating Semantic Features

In the fourth chapter, we address a limitation of foundation model features that arises in the context of the previous chapter: features distilled from foundation models struggle to disambiguate instances of the same semantic class such as "left hand" versus "right hand", which can lead to significant errors during pose fitting and other 3D downstream tasks. Other approaches resolve these ambiguities using annotated datasets with dense 3D correspondences between meshes. In contrast, we propose a self-supervised method that requires no annotated correspondences and learns to separate these semantic classes from only a small number of unpaired training meshes. Specifically, we construct an embedding space based on the idea that points far apart on the surface of a 3D mesh should be separated in the embedding space, even if semantically similar. Unlike prior pairwise matching approaches, this yields a joint embedding space that generalizes to unseen shapes at test time without further optimization. After training, our embedding space encoder can be used as a drop-in replacement, improving performance in a variety of tasks, such as correspondence matching and pose transfer.

1.4 Rotation-Equivariant Neural Jacobian Fields

In chapter five, we propose a rotation-equivariant reformulation of Neural Jacobian Fields for 3D mesh deformation based on semantic features. Neural Jacobian Fields achieve state-of-the-art results for pose transfer between meshes, but regress Jacobians in single global coordinate system, requiring input shapes to be registered in a global coordinate system. Furthermore, it conditions deformations on extrinsic information and geometric descriptors, limiting its generalization capacity across shapes. In the previous chapter, we showed that the surface-aware features introduced there can replace these extrinsic signals, producing strong results in one-shot pose transfer between meshes of different shapes. However, this still requires the input mesh to be rotationally aligned with the training shapes. We propose a rotation-equivariant formulation of Neural Jacobian Fields that learns to represent the Jacobians in the learned tangent frames of each mesh face. The tangent frame is obtained by aligning it with the semantic feature gradient from chapter four, allowing for learning deformations that generalize across shapes and are rotation-equivariant.

In chapter six, we summarize our findings and discuss our contributions in light of recent advances in the field, and point to venues for future work.

2 Template-free Articulated Neural Point Clouds for Reposable View Synthesis



Figure 2.1: Left: Multi-view data set showing an object moving in an articulated fashion, which is used for training. Right: Manually animated kinematic model after training. The colors show the reconstructed skinning weights, which visualize the independent components that were discovered by our method.

Dynamic Neural Radiance Fields (NeRFs) achieve remarkable visual quality when synthesizing novel views of time-evolving 3D scenes. However, the common reliance on backward deformation fields makes reanimation of the captured object poses challenging. Moreover, the state of the art dynamic models are often limited by low visual fidelity, long reconstruction time or specificity to narrow application domains. In this chapter, we present a novel method utilizing a point-based representation and Linear Blend Skinning (LBS) to jointly learn a Dynamic NeRF and an associated skeletal model from even sparse multi-view video. Our forward-warping approach achieves state-of-the-art visual fidelity when synthesizing novel views and poses while significantly reducing the necessary learning time when compared to existing work. We demonstrate the versatility of our representation on a variety of articulated objects from common datasets and obtain reposable 3D reconstructions without the need of object-specific skeletal templates.*

* This chapter is based on the paper "Template-free articulated neural point clouds for reposable view synthesis," published at NeurIPS 2023 [1].

2.1 Introduction

Synthesizing novel photo-realistic views of captured 3D scenes is important for many domains including virtual/augmented reality, video games or movie productions. In recent years, Neural Radiance Fields (NeRFs) [2] have proved their remarkable capacity to represent complex view-dependent effects, captured in photographs and videos, sparsely sampled from natural light fields [3]. Follow-up works have extended the scope to dynamic scenes [4–11], facilitating rendering of unseen views at different timestamps. Despite progress in reconstruction quality and speed [12], manipulating learned scenes remains a challenge, but would be a highly desirable feature, as it can enable downstream applications, such as the creation of avatars for virtual presence or 3D assets for games and movies.

The key challenge of reposing a NeRF is inverting the backward-warping function that maps individual observations to a shared canonical representation [4]. It is an inversion of traditional kinematic animation, such as Linear Blend Skinning (LBS) [13], where a canonical shape is forward-warped to a desired pose. Such inverse mapping often requires resolving ambiguous situations as it is difficult to guarantee bijectivity (see Fig. 2.2). A common remedy are parametric templates, typically built for narrow application domains, such as human heads and bodies [14–17], but they are difficult to generalize. Alternatively, object shapes can be retrieved from videos as ensembles of geometric parts, yet existing techniques provide limited image-synthesis fidelity [18–20]. Our work aims to combine these different lines of work and enable joint learning of the NeRF representation and its pose parameterization from sparse or dense multi-view videos. We aim for time-efficient class-agnostic view synthesis of reposable models with high image-synthesis quality without access to a template or pose annotation, which is a combination not currently covered by existing work (see Table 2.1).

Recent work by Noguchi *et al.* [21] addresses a similar problem. However, we exploit the structure-free nature of point-based NeRF representations [22], enabling direct forward warping of the canonical object to any desired pose, while maintaining the capacity and flexibility of NeRF-like rendering to capture highly detailed image features. Together with our automatically extracted and jointly-optimized LBS skeletal pose parameterization, our work allows for faster training and achieves state-of-the-art visual quality in a much shorter time. Finally, we demonstrate how to easily repose learned representations of varied objects by directly editing joint angles of the forward LBS model.

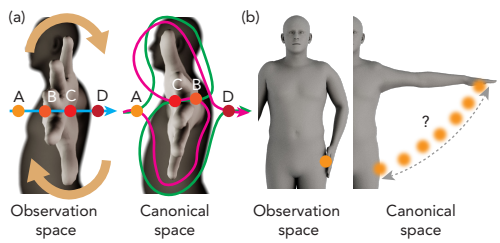


Figure 2.2: Examples of poses difficult for backward warping. (a) Ill-defined projection from an observation to the canonical space. Both ambiguous solutions (green and magenta) correctly pass through the semantically corresponding surface points B and C. (b) Projection ambiguity for points of contact between two surfaces. Note that in contrast, it is trivial to forward warp the object points from a well-chosen canonical space to any observation space.

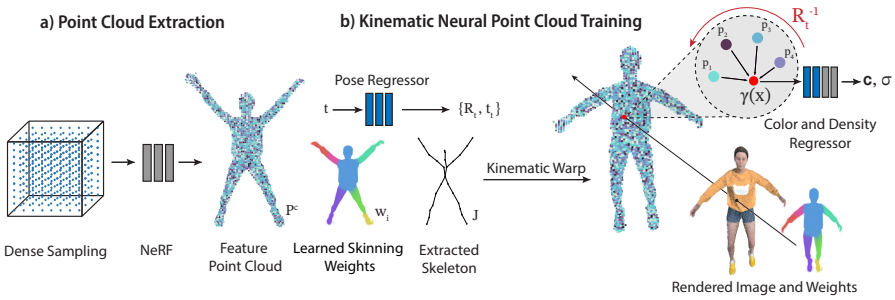


Figure 2.3: Overview of our method: a) First, we pre-train a NeRF backbone to initialize a feature point cloud P^c for a selected canonical timestamp and to extract an initial skeleton. b) During the main training stage, P^c is forward-warped using LBS consisting of learned time-invariant skinning weights $\hat{w}_i$ and time-dependent pose transformations from an MLP regressor Φ_r. The image is obtained by integration and decoding of features aggregated from points along each camera ray. In summary, we fine-tune the initial neural point features $\mathbf{f_i}$, skinning weights $\hat{w}_i$, joints J, density and color regressor Φ_d and Φ_c of the backbone. We fully train the pose regressor Φ_r and feature point decoder Φ_p from scratch. In test time, we modify the pose transformations to obtain novel poses.

To summarize, we make the following contributions: 1) We propose a novel method for learning articulated neural representations of dynamic objects using forward projection of neural point clouds. 2) We demonstrate state-of-the-art novel view synthesis for a variety of multi-view videos with training times lower than comparable methods. 3) We jointly learn a skeletal model without domain-specific priors or user annotations and demonstrate reposing of the reconstructed 3D objects.

2.2 Related Work

Neural Radiance Fields and Parameterizations Implicit neural networks have recently emerged as a powerful tool for building differentiable representations of 3D scenes [25–48] and for learning view-dependent *Neural Radiance Fields* (NeRFs) [2, 3, 49–58] enabling photo-realistic novel view synthesis. Their high computational cost has been dramatically reduced by spatial decomposition [59–61] and hybrid representations [62–66]. Recently, neural point clouds have combined expressive neural functions with the flexibility of explicit geometry [22, 67–70]. We also use neural point clouds, but unlike previous work, we jointly learn a forward kinematic model along with the associated soft part segmentation and, thus, convert fast voxel-based representations to reposable NeRFs.

Dynamic Neural Radiance Fields Equivalent to 2D video, dynamic NeRFs capture temporal changes in scenes by encoding each frame separately [71], expanding the radiance field into the temporal domain [8, 9, 72, 73], or time-dependent interpolation in a compact latent representation [7, 10, 11]. Alternatively, a single canonical NeRF can be animated by backward warping from the canonical space to individual time-varying observation spaces [4–7, 10, 12]. However, such warps rely on the bijectivity of the mapping which is difficult to guarantee for all observed poses (see Fig. 2.2). Forward mapping only requires a single well-posed canonical representation, which we exploit.

Table 2.1: A comparison of representative dynamic 3D representations learned from 2D videos. We analyze reposibility, agnosticism to object class, image synthesis fidelity and training time. Papers that demonstrate reposing are shown below the bar. "*Shape*" denotes shape-only reconstruction without texture details.

Method	Pose	Generic	Fildelity	Training
LASR [18]	No	Yes	Shape	$\leq$ 2 h
ViSER [19]	No	Yes	Shape	2–12 h
BANMO [23]	No	Yes	Low	$\geq$ 12 h
D-NeRF [4]	No	Yes	High	$\geq$ 12 h
TiNeuVox [12]	No	Yes	High	$\leq$ 2 h
CASA [24]	Yes	No	Shape	$\leq$ 2 h
LASSIE [20]	Yes	Yes	Low	$\geq$ 12 h
WIM [21]	Yes	Yes	High	$\geq$ 12 h
Ours	Yes	Yes	High	$\leq$ 2 h

Object Reposing Directly reposing high-dimensional deformation fields of dynamic NeRFs is impractical. Instead, parametric templates can sparsely represent prominent deformation modes for faces [14, 15], bodies [16, 17], hands [74], and also non-human objects, such as animals [75]. Together with skeleton-based LBS, they enable articulated neural representations of human heads [70, 76–82] or bodies [83–98], as well as modeling distributions in generative models [99–102]. However, the assumption of piece-wise rigid motion and the availability of a skeletal template restrict the applicable object classes. The former issue has been addressed by physically inspired but computationally expensive deformation models [103]. To remedy the latter issue, the skeletal model can be retrieved from a database [24], adapted from a generic graph [20, 104], fitted as a template to 2D observations [105–109] or fully learned during a 3D shape recovery [21, 110]. An external cage can also be used for re-animation of static NeRFs [111]. Alternatively, a surface embedding can be learned without a skeleton if reposing is not required [18, 19, 23].

Similarly to Noguchi *et al.* [21], we jointly learn a 3D object representation, a skeletal model, and observed poses. However, we utilize a point-based NeRF representation to benefit from its computational efficiency, ease of geometric transformation, and capacity to learn complex light fields. We share our point-based approach with a contemporaneous work *MovingParts* [112] which, however, relies on inverting the backward flow. We demonstrate the reconstruction of large transformations in the *Robots* dataset [21], not demonstrated in *MovingParts*. Furthermore, our approach does not enforce binary segmentation and, hence, can represent non-rigid part transitions (see Fig. 2.8 right).

2.3 Preliminaries

Our method builds upon available NeRF reconstruction methods as a backbone for learning a reposable dynamic representation. In principle, any NeRF-like representation is a suitable initial point for our training. In practice, we use TiNeuVox [12] in all experiments, as it enables fast reconstruction with both sparse and dense multi-view supervision in dynamic scenes. Next, we describe the core concepts of NeRF [2] and TiNeuVox [12] to provide context for our method in Sec. 2.4.

Time-Aware Neural Voxels NeRF [2] maps a point $\mathbf{p} = (x, y, z)$ and view direction $\mathbf{d} = (\theta, \phi)$ to color $\mathbf{c} = (r, g, b)$ and density σ. The mapping function often takes form

of a Multi-Layer-Perceptron (MLP) Φ, such that $(\mathbf{c}, \sigma) = \Phi(\mathbf{p}, \mathbf{d})$. The color of an image pixel $C(\mathbf{r})$ can be obtained via volume rendering along the ray $r(t) = \mathbf{o} + t\mathbf{d}$.

TiNeuVox [12] expands NeRF to dynamic scenes and significantly increases training speed by utilizing an explicit volume representation. To this extent, each point $\mathbf{p}$ at time t in observation space is backward-warped to canonical space as

$$\mathbf{p}^c = \Phi_b(\hat{\mathbf{p}}, \hat{\mathbf{t}}), \text{ where } \hat{\mathbf{p}} = \gamma(\mathbf{p}), \ \hat{\mathbf{t}} = \Phi_t(\gamma(t)), \tag{2.1}$$

Here, γ is the positional encoding [2], Φ_t a small color-decoding MLP, Φ_b a backward-warping MLP, and $\hat{\mathbf{p}}$ and $\hat{\mathbf{t}}$ the encoded point and time respectively. The canonical point $\mathbf{p}^c$ is used to sample a feature vector $\mathbf{v}_m \in \mathcal{R}^V$ from the explicit feature volume $\mathbf{V} \in \mathcal{R}^{X \times Y \times Z \times V}$ by means of trilinear interpolation $interp$ with varying stride s_m at each scale: $\mathbf{v}_m = \gamma(interp(\mathbf{p^c}, \mathbf{V}, s_m))$. The feature vectors across all scales are concatenated as $\mathbf{v} = \mathbf{v}_1 \oplus ... \oplus \mathbf{v}_m \oplus ... \oplus \mathbf{v}_M$, where $\oplus$ is the concatenation operator. Finally, the view-dependent color and density are regressed by additional MLPs Φ_f, Φ_d and Φ_c:

$$\mathbf{f} = \Phi_f(\mathbf{v}, \hat{\mathbf{p}}, \hat{\mathbf{t}}), \text{ where } \sigma = \Phi_d(\mathbf{f}), \ \mathbf{c} = \Phi_c(\mathbf{f}, \mathbf{d}). \tag{2.2}$$

The volume rendering equation [113] integrates density/color of points $\mathbf{p}_i$ sampled along the ray r:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - exp(-\sigma_i \delta_i))\mathbf{c}_i, \text{ where } T_i = exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)), \tag{2.3}$$

Here, δ_i is the distance between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. The feature volume $\mathbf{V}$ and all MLPs are optimized end-to-end and supervised by the Mean Squared Error (MSE) and the corresponding ground-truth pixel color $C(\mathbf{r})$: $\mathcal{L}_{photo} = ||\hat{C}(\mathbf{r}) - C(\mathbf{r})||_2^2$.

Note that because Φ_f (Eq. 2.2) is conditioned by the time $\hat{\mathbf{t}}$, space and time are entangled and the feature volume V does not represent a single static canonical shape. Therefore, we cannot directly learn a forward warping function to invert Φ_b as proposed by Chen *et al.* [103]. In the next section, we show that this is not necessary and we can learn a forward kinematic model directly without relying on specific properties of the backbone, which will allow us to replace it in the future.

2.4 Method

While dynamic NeRF models such as TiNeuVox [12] can reproduce motion in a dynamic scene, they do not enable reposing. Furthermore, it is impractical to post hoc reparametrize their motion representation due to the ambiguities of backward flow inversion (see Fig. 2.2).

We instead propose a completely new representation based on a neural point cloud and an explicit kinematic model for forward warping, and we use the backbone only as an initialization for our training procedure (see Fig. 2.3). First, we extract a feature point cloud from a selected canonical frame of the backbone Sec. 2.4.1. Second, we describe the underlying kinematic model of our 3D representation and its initialization Sec. 2.4.2.

Consecutively, we introduce how the 3D point cloud is warped from a canonical space to an observation space and rendered Sec. 2.4.4. Lastly, we describe our losses Sec. 2.4.5.

2.4.1 Canonical Feature Point Cloud

We first pre-train a backbone NeRF model (TiNeuVox [12]) using its default parameters. To initialize the feature point cloud, we sample the canonical density function of TiNeuVox $\sigma = \Phi_d(\mathbf{f})$ (Eq. 2.2) on a uniform coordinate grid and discard empty samples through thresholding. The grid resolution is adaptively chosen, such that $|P| \approx 10000$, similarly to other explicit NeRFs [12, 65, 114]. After thresholding, we obtain points $P = \{\mathbf{p}_i | i = 1, ..., N\}$. Furthermore, we extract a feature vector $\mathbf{f}_i$ for each point $\mathbf{p}_i$ (see Eq. 2.2).

2.4.2 Kinematic Model

We forward-warp our feature point cloud from canonical space to observation space using an LBS kinematic model to match the training images. Here, we describe how we initialize, use, and simplify our kinematic model.

Skeleton Initialization We do not rely on a class-specific template to initialize our kinematic skeleton. Instead, we extract an approximate initial skeleton tree by Medial Axis Transform (MAT) and refine it during training.

Let $M = \{\mathbf{p}_m | m = 1, ..., M\}$ be the set of medial axis points extracted by applying a MAT on P^c. We choose the root of our kinematic model $\mathbf{p}_{root}$ as the medial axis point that is closest to all other medial-axis points, i.e., $\mathbf{p}_{root} := \arg\min_i \sum_j ||\mathbf{p}_i - \mathbf{p}_j||_2, \forall \mathbf{p}_i \in M, i \neq j$.

Next, we leverage dense sampling-grid neighborhoods and define a graph G_M connecting neighboring points in M. Points disconnected from $\mathbf{p}_{root}$ are removed. We then use a heuristic to select sparse joints (nodes) as a subset of G_M and define the bones (edges) based on their connectivity. To this extent, we traverse G_M starting from $\mathbf{p}_{root}$ in a breadth-first manner and mark $\mathbf{p}_m$ as a joint $\mathbf{j}_b$ if its geodesic distance from the preceding joint exceeds a threshold $B_{length} = 10$. As a result, we obtain a set of time-invariant canonical joint positions $J = \{\mathbf{j}_b\}$ which we further optimize during training.

While this skeleton is usually over-segmented, we show in our experiments that this does not hamper the training. We propose a pruning strategy to enable easier pose manipulation afterwards (see Fig. 2.11).

Blend Skinning Weights For each point $\mathbf{p}_i$ we initialize its raw skinning weight vector $\mathbf{w}_i$ by an exponential decay function of the distance $dist$ to each bone line b_j such that $\hat{w}_{i,j} = 1/e^{dist(\mathbf{p}_i^c, b_j)}$. Before skinning, we obtain the final blend skinning weight vector $\mathbf{w}_i$ through scaling by a global learnable parameter α and applying a softmax: $w_{i,j} = e^{\hat{w}_{i,j}/\alpha} / \sum_k e^{\hat{w}_{i,k}/\alpha}$. During the training, we optimize the initial $\mathbf{w}_i$ as well as α. Accounting for the per-point weights, we define our full canonical feature point cloud as $P^c = \{(\mathbf{p}_i^c, \mathbf{f}_i, \mathbf{w}_i) | i = 1, ..., N\}$.

Point Warping We forward-warp the canonical point cloud P^c to an observation space of timestamp t via LBS [13]. The local transformation matrix $\hat{T}_b^t$ of each bone b is defined by a rotation R_b^t around its parent joint $\mathbf{j}_b$. Consequently, each point $\mathbf{p}_i^c$ is transformed by a linear combination of bone transformations as:

$$\begin{aligned} \mathbf{p}_i^t &= \bar{T}_i^t p_i^c = \sum_{b=1}^{|B|} w_{i,b} T_b^t \mathbf{p}_i^c, \text{ with } T_b^t = T_{p_b}^t \hat{T}_b^t, \text{ and} \\ \hat{T}_b^t &= \begin{bmatrix} R_b^t & \mathbf{j}_b + R_b^t \mathbf{j}_b^{-1} \\ \mathbf{0} & 1 \end{bmatrix}, \end{aligned} \tag{2.4}$$

where T_b^t is defined recursively by its parent bone p_b. $T_{p_b}^t$ of the skeleton root is identity.

We express rotations $R_b^t \in SO(3)$ using the Rodrigues' formula, where $\mathbf{r} = \mathbf{r}/||\mathbf{r}||$ is the axis of rotation. However, we learn the rotation angle θ directly as an additional parameters because we find it leads to a better pose regression than using $\theta = ||\mathbf{r}||$. The time-dependent rotations $\mathbf{r}_b, \theta_b$ for each bone b are regressed by an MLP: $\Phi_r(\gamma(t)) = \{\mathbf{r}_1^t, \theta_1^t, ..., \mathbf{r}_b^t, \theta_b^t, ..., \mathbf{r}_B^t, \theta_B^t, \mathbf{r}_r^t, \theta_r^t, \mathbf{t}_r^t\}$, where $\mathbf{r}_r^t$, θ_r^t and $\mathbf{t}_r^t$ are the time-dependent root rotation and translation.

2.4.3 Skeleton simplification

The initial skeleton's over-segmentation does not hamper pose reconstruction during training, but we optionally simplify the skeleton after training to ease pose editing (see Fig. 2.11). We prune or merge bones based on the size of rotation angles θ_b^t produced by the transformation regressor Φ_r. Joints, which do not exhibit a rotational change from the rest pose above the threshold of t_r deg in more than 5% of the observed timestamps, are marked static. We then merge the bones of such joints and their corresponding weights $\mathbf{w}$.

More specifcally, after the static joints are identified, they can be removed and the skinning weights corresponding to their children and parent bones can be merged. Here, the parent bone refers to the skeletal tree edge starting in the given joint and pointing towards the root while the children bones are the edges in the leaf direction. The merge only happens for two specific configurations: First, for a static joint, we merge the skinning weights of the children with their parent. Second, we merge the weights of sibling bones if their motion does not differ based on the 5%-heuristic.

To simplify the kinematic skeleton, we further prune bones where possible. A kinematic chain that has multiple consecutive static joints connected by bones will be reduced to the longest possible bone, removing unnecessary joints. However, a joint that functions as a center of rotation for non-static children joints is never pruned. Lastly, static end effector joints are removed. An example of the procedure can be seen Fig. 2.4.

2.4.4 Dynamic Point Rendering

We adopt the point cloud rendering from [22] but extend it to explicitly model rotational invariance of the radiance field. For each sampling point x, we find up to $N = 8$ nearest observation feature points $\mathbf{p}_i^t$ within a radius of 0.01 and roto-translate them into their

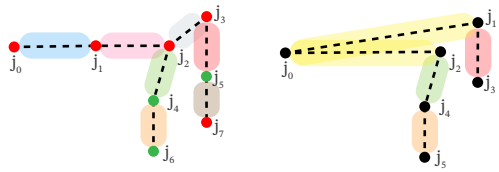


Figure 2.4: Visualization of kinematic model simplification. a) The trained kinematic mode: The nodes visualize whether the joint is static (red) or not (green). Each bone is associated with a blend-skinning weight (rounded colored rectangles), and j_0 is the root node. b) The simplified kinematic model: Based on the static joints, bones have been pruned and weights have been merged where possible. Weights of bones (j_0, j_1), (j_1, j_2), and (j_2, j_3) have been merged into the root weight (yellow), as all of the joints were marked as static. Furthermore, joints j_1 and j_7 could be removed without harming the underlying kinematic model because they do not have an effect on point cloud deformation. The root node is never pruned.

canonical frames. This enables the feature embedding MLP Φ_p to learn spatial relations in a pose-invariant coordinate frame as:

$$\mathbf{f}_{i,x}^t = \Phi_p(\mathbf{f}_i, x_{\mathbf{p}_i}^t), \text{ with } x_{\mathbf{p}_i}^t = \gamma(R_i^{t^{-1}}(x - \mathbf{p}_i^t)), \tag{2.5}$$

where, R_i^t is the rotation component $\bar{T}_i^t$ (Eq. 2.4) for point $\mathbf{p}_i^t$ and $\gamma(.)$ is the positional encoding [2]. The neighboring embeddings $\mathbf{f}_{i,x}^t$ are aggregated by inverse distance weighting, which produces the final feature input for Φ_d and Φ_c (see Eq. 2.2) and the consequent volume rendering (Eq. 2.3):

$$\mathbf{f}_x^t = \sum_i^N \frac{d_i}{\sum_j^{|B|} d_j} f_{i,x}^t, \text{ where } d_i = ||p_i^t - x||^{-1}. \tag{2.6}$$

The density and color are finally obtained via $\sigma = \Phi_d(\mathbf{f}_x^t)$, and $\mathbf{c} = \Phi_c(\mathbf{f}_x^t, \mathbf{d})$.

2.4.5 Losses

Next to the photometric loss $\mathcal{L}_{photo}$ (Sec. 2.3), we utilize a 2D chamfer loss to penalize the difference between the point cloud projected into a training view $I(P^t)$ and the corresponding 2D ground truth object mask M^t: $\mathcal{L}_{mask} := \mathcal{L}_{chamf}(I(P^t), M^t)$. The chamfer loss is defined as:

$$\begin{aligned}\mathcal{L}_{chamf}(P^1, P^2) = &\frac{1}{|P^1|}\sum_i^{|P^1|} \min_j ||\mathbf{p}_i^1 - \mathbf{p}_j^2||_2^2 \\ &+ \frac{1}{|P^2|}\sum_j^{|P^2|} \min_i ||\mathbf{p}_i^1 - \mathbf{p}_j^2||_2^2.\end{aligned} \tag{2.7}$$

To prevent the skeleton from diverging too much from the medial axis M, we further minimize the chamfer loss between M and the joints J (see Sec. 2.4.2): $\mathcal{L}_{skel} := \mathcal{L}_{chamf}(M, J)$. In addition, we minimize the transformation angles and the root translation: $\mathcal{L}_{tranf} = (\sum_i |\theta_i^t|) + |\mathbf{t}_r^t|$. Local rigidity is further enforced upon points after deformation via

as-rigid-as-possible regularization:

$$\mathcal{L}_{arap} = \sum_{i}^{|P^t|} \sum_{j}^{N(p_i)} |||p_i^c - p_j^c||_2^2 - ||p_i^t - p_j^t||_2^2|. \tag{2.8}$$

Lastly, we apply two regularizers on the blend skinning weights. To encourage smoothness, we penalize divergence of skinning weights in the rendering neighborhood N:

$$\mathcal{L}_{smooth} = \sum_{i}^{|P^t|} \sum_{j \in N(p_i)} |w_i - w_j|. \tag{2.9}$$

And finally, to encourage sparsity, we apply:

$$\mathcal{L}_{sparse} = -\sum_{i}^{|P^c|} \sum_{j}^{B} w_{i,j} \log(w_{i,j}) + (1 - w_{i,j}) \log(1 - w_{i,j}). \tag{2.10}$$

In total, our training loss is $\mathcal{L} = \omega_0 \mathcal{L}_{photo} + \omega_1 \mathcal{L}_{mask} + \omega_2 \mathcal{L}_{skel} + \omega_3 \mathcal{L}_{tranf} + \omega_4 \mathcal{L}_{smooth} + \omega_5 \mathcal{L}_{sparse} + \omega_6 \mathcal{L}_{ARAP}$ where $\omega = \{200, 0.02, 1, 0.1, 10, 0.2, 0.005\}$ in our experiments.

2.4.6 An extension for local deformations

Our full model is based on skeletal articulation with Linear Blend Skinning and as such it can well model large locally rigid deformations. While well suited for many synthetic objects, this alone does not accurately model local deformations of fabrics in the clothes of humans subjects in the *ZJU-MoCap* dataset. Despite this, our full method can perform meaningful articulation of the human subjects as shown in Sec. 2.5.2. To further

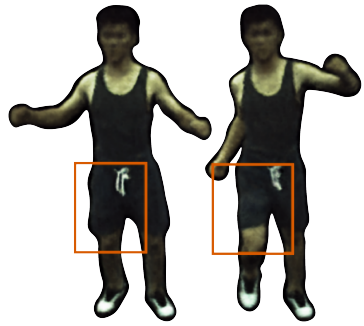


Figure 2.5: Example of clothes deformations when conditioning Φ_p on the pose embedding in Ours^{pose}.

improve reconstruction of small surface details, we propose to extend our full method and additionally condition the feature regressor Φ_p by a pose embedding $pe^t \in \mathbb{R}^{64}$ such that $\mathbf{f}_{i,x}^t = \Phi_p(\mathbf{f}_i, \underline{pe^t}, x_{\mathbf{p}_i}^t)$ (see Eq. 5 from the main paper). Here, pe^t is regressed by another MLP (4 layers, $0.5 \times |\gamma(\mathbf{J}^c)|$ hidden dimensions) from the canonical joint positions $\mathbf{J^c}$ and the joint positions $\mathbf{J^t}$ observed at time t as $pe^t = \Phi_{pe}(\gamma(\mathbf{J}^c - \mathbf{J}^t))$. This allows the method to learn additional local geometry deformations and color changes specific to the current pose (see Fig. 2.5). Note, that the poses are not an input to our

method as they are already jointly learned in our original full model. We observe, that the learned deformations are locally constrained by the fixed neighborhood search radius in Eq. 6. Furthermore, we detach the gradient flow from $\mathbf{J}^t$ such that the skeletal poses are not optimized to fit the local deformations.

2.5 Experiments

Here, we evaluate our work, which obtains state-of-the-art view-synthesis quality with a lower training cost than other articulated methods. We also demonstrate class-agnostic reposing capability and we evaluate contribution of method components in an ablation study. Video examples can be seen on the project website.

2.5.1 Experiments Setup

Datasets We chose three multi-view video datasets that are commonly used for the evaluation of dynamic multi-view synthesis methods and pose articulation. First, the *Robots* dataset [21] features multi-view synthetic videos of 8 topologically varied robots, making it well suited for testing pose articulation performance (see Fig. 2.7). We use 18 views for training and 2 for evaluation. Second, the *Blender* dataset [4] is a sparse multi-view synthetic dataset with 5 humanoid and 2 other articulated objects[†]. Its visual quality benchmarks are used to test the fidelity of image detail reconstruction (see Fig. 2.8 right). We use the original training-test split. Third, *ZJU-MoCap* dataset [83] is a common test suite for dynamic human reconstruction and we evaluate 5 of its multi-view sequences with the same 6 training views as used in *Watch-It-Move* [21]. Finally, we evaluate image quality using peak signal-to-noise ratio (PSNR), structural similarity (SSIM) [115], and learned perceptual image patch similarity (LPIPS) [116] image metrics.

Implementation details We pre-train the TiNeuVox [12] backbone using the authors' implementation and an additional distortion loss [117], as implemented in [118]. Our method is implemented in Pytorch and we train each scene using the Adam optimizer for 160k (*Blender* and *Robots*) or 320k (*ZJU-MoCap*) iterations with a batch size of 8192 rays, sampled randomly from multiple views and a single timestamp. We choose the canonical timestamp by visual inspection, and gradually increase the number of observed timestamps during training. We adjust ray sampling and scheduling for the *ZJU-MoCap* dataset (see the Supplement). All experiments were done on a single Nvidia GPU RTX 3090Ti. See the supplementary materials for details. For more details, see the Appendix.

Baselines We compare our method to state-of-the-art non-articulated and articulated methods for high-fidelity multi-view video synthesis (see Table 2.1 for an overview). *D-NeRF* [4] extends NeRF to the temporal domain by backward warping a static canonical NeRF. *TiNeuVox* [12] improves performance of *D-NeRF* using voxel grids. *Hexplane* [73] and *K-Planes* [72] decompose to the space-time volume to several hyper-planes. Finally, *Watch-It-Move* [21] (WIM) jointly learns a surface representation and LBS model for

[†] We do not use the multi-component *Bouncing balls* scene.

articulation. Note, that because we aim at time-efficient learning, training of WIM was stopped after 11 hours (i.e., 80k of the original 400k optimization steps).

2.5.2 Results



Figure 2.6: Qualitative comparison displaying two held-out views-frames of scenes from the *Robots* rendered by WIM [21] and our method after 2 and 10 hours of training, and the PSNR scores.

Novel view synthesis In Fig. 2.7 and Table 2.2 we provide quantitative results for the *Robots* view synthesis without the skeleton simplification, and qualitative results can be seen in Fig. 2.6. Additional visual examples are available in the appendix.

After the initial pre-training, our method quickly surpasses the image quality of WIM [21]. The mean image scores obtained for our method after 2 hours of training (incl. pre-training) are higher than those that WIM achieves after 10 hours. We attribute this to NeRF's capability to approximate even complex shapes, which are difficult to fit using the signed distance function utilized by WIM. Nevertheless, for visually simple objects with long nested pose transformation chains, this capacity might encourage false explanations of the articulation and cause artifacts, as visible in Fig. 2.8 left. However, this is not a common issue.

We illustrate it in the *Blender*, where WIM struggles to represent fine details (see Fig. 2.8 right), while our method achieves image scores close to those of non-reposable baselines (see Table 2.3 and the supplement).

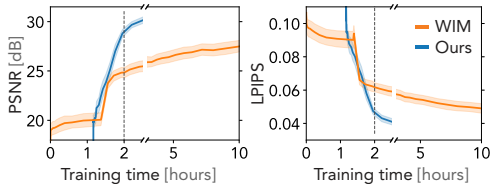


Figure 2.7: Quality of unseen view synthesis during training with 95% confidence intervals in the *Robots* dataset [21]. The initial plateau of WIM [21] matches the 10k warm-up steps used by the authors before training with all data. Our onset time corresponds to the 70 minutes required for pre-training of the backbone. Training of our method was terminated after 2.5 hours.

Table 2.2: Perfomance of our method compared to WIM [21] on the *Robots* dataset [21] after 2 and after convergence.

Time	Method	PSNR↑	SSIM↑	LPIPS↓
2 h	WIM [21]	25.05	0.952	0.059
	Ours	**28.81**	**0.968**	**0.047**
10 h	WIM [21]	27.61	0.964	0.046
	Ours	**30.19**	**0.973**	**0.041**



Figure 2.8: While our model well reproduces details and non-rigidity of the *Dinosaur* (right), it can fail for complex motions combining long chains of rotations with texture cue ambiguity (left).

Furthermore, Fig. 2.9 shows that our image quality is not highly dependent on the pre-training phase. We observe that mere 100 iterations of *TiNeuVox* pre-training provide an initialization sufficient for achieving PSNR scores over 30 dB by our method. However, while fine geometric details are still recovered, the coarse initial shape limits the skeleton complexity and, therefore, we opt for 20k pre-training steps in all other experiments.

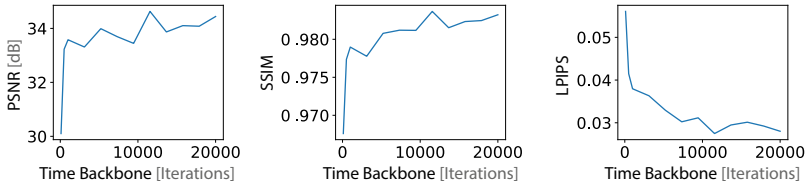


Figure 2.9: Effect of the backbone initialization pre-training steps on the final result of our method when trained on the *Jumping Jacks* scene from the *Blender* dataset. Our final choice of 20k iterations corresponds to approximately 70 minutes of real time.

Table 2.3: Quality of unseen view synthesis for the *Blender* dataset [4]. Results of D-NeRF and TiNeuVox-B reprinted from Fang *et al.* [12]. *Without the "Bouncing balls" scene.

Method	Reposable	PSNR↑	SSIM↑	LPIPS↓	Training Time
D-NeRF [4]	No	30.50	0.95	0.07	20 hours
TiNeuVox-B [12]	No	32.67	0.97	0.04	28 mins
Hexplane [73]	No	31.04	0.97	0.04	11.5 mins
K-Plane hybrid [72]	No	31.61	0.97	-	52min
WIM* [21]	**Yes**	23.81	0.91	0.10	11 hours
Ours*	**Yes**	29.10	0.94	0.06	2.5 hours

Reposing In Fig. 2.11, we visualize the learned blend-skinning weights $\mathbf{w}_i$ and skeletons with and without the skeleton simplification. The algorithm is able to substantially reduce the skeleton complexity, while largely preserving semantic articulations observed in the training data. However, it is not able to remove all unnecessary skeletal branches when complex geometry is present (see Fig. 2.11 right). In Fig. 2.12 and Fig. 2.10, we

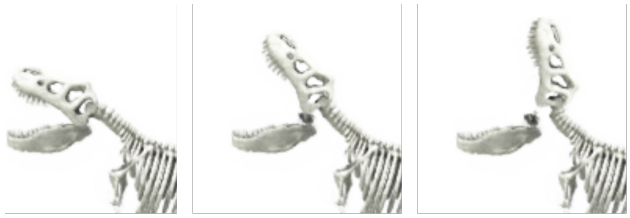


Figure 2.10: Manually defined jaw over-extension animation for the *T-Rex* from the *Blender* dataset.

show that such post-processed skeletons allow for animating of novel poses by smoothly interpolating between user-defined poses.

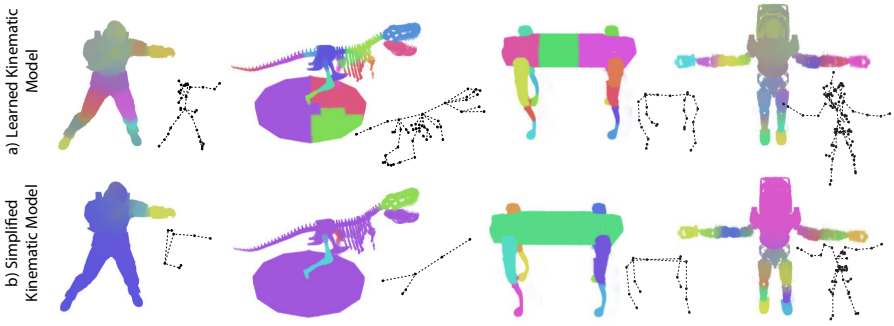


Figure 2.11: Learned LBS weights and skeleton: a) After training. b) After additional post-processing (weight merging and skeleton pruning, see Sec. 2.4.3).

Loss ablation Our experiments suggest the regularization does not improve the image quality, but it improves the quality of the kinematic model, which is important for our main goal of reposing (see Fig. 2.13). Specifically, $\mathcal{L}_{ARAP}$ avoids non-rigid distortions (a), $\mathcal{L}_{smooth}$ leads to a more semantically meaningful part segmentation (b), $\mathcal{L}_{skel}$ encourages functional placement of the skeleton joints inside the object volume even for thin parts (c), and $\mathcal{L}_{tranf}$ leads to a reduction of necessary object parts after simplification (62 components are removed instead of 49 for *Atlas*). More details are provided in the Supplement.

Backbone

	PSNR↑	SSIM↑	LPIPS↓
Full	30.19	0.973	0.041
Random	29.97	0.971	0.045

Table 2.4: Results with *TiNeuVox* initialization (full method) and with a random initialization of the features $\mathbf{f}_i$ and the regressors $\mathbf{\Phi}_d$ and $\mathbf{\Phi}_c$ in the *Robots* dataset.

	PSNR↑	SSIM↑	LPIPS↓
Full	29.10	0.94	0.06
No F.t.	29.16	0.94	0.05

Table 2.5: Results with (full method) and without fine-tuning of the feature points $\mathbf{f}_i$ in the *Blender* dataset.

The *TiNeuVox* backbone allows us to transfer parameters learned during the pre-training and initialize the features $\mathbf{f}_i$ and the regressors $\mathbf{\Phi}_d$ and $\mathbf{\Phi}_c$. Interestingly, our experiment shows that such off-the-shelf features $\mathbf{f}_i$ often do not need any further fine-tuning (see

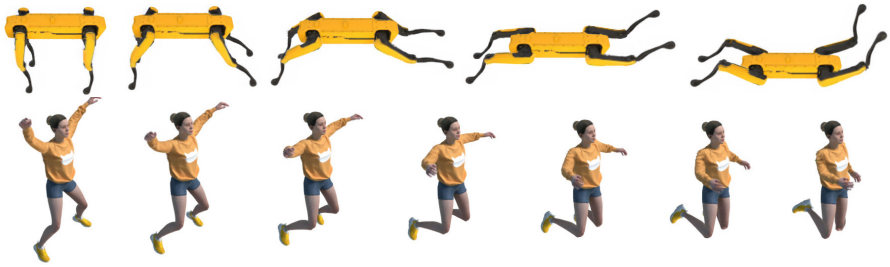


Figure 2.12: Reposing using the simplified skeleton. Interpolation from canonical to manually defined novel pose.



Figure 2.13: Ablation examples. a) $\mathcal{L}_{ARAP}$ enforces rigidity after pruning (see upper pipes), b) $\mathcal{L}_{smooth}$ results in better part-segmentation, c) $\mathcal{L}_{skel}$ enforces the joints to stay inside the shape.

Table 2.5). Despite this, our method is agnostic to the backbone choice by design and the end-to-end training procedure of the entire model is essential for this. We demonstrate it by training with a random initialization. This way, only the point positions $\mathbf{p}_i^c$ obtainable by any 3D reconstruction method are needed. Table 2.4 and Table 2.5 shows that this leads to only a negligible drop in reconstruction quality in the *Robots* dataset. Here, the weight of $\mathcal{L}_{skel}$ ($w_2 = 1$) was adjusted to prevent drift of the skeleton.

Table 2.6: Quantitative results on the *ZJU-MoCap* dataset. Ours: Our full method. Ours^{pose}: Ours with an additional pose-conditioned feature embedding network Φ_p. Ours^{SMPL}: Ours^{pose} with ground truth SMPL [16] skeleton.

Method	PSNR↑	SSIM↑	LPIPS↓
WIM [21]	31.08	0.963	**0.053**
Ours	29.60	0.958	0.063
Ours^{pose}	**32.05**	**0.967**	0.056
Ours^{SMPL}	32.01	0.967	0.056

ZJU-MoCap In Fig. 2.14 and Table 2.6, we compare our method to WIM in the *ZJU-MoCap* dataset. We observe that both methods are able to recover the 3D shape and the skeletal motion despite the difficulty of an accurate fine texture detail reconstruction. This can partly be attributed to imperfections in camera calibrations (see Supplement F of [97]) and partly by soft deformations of clothes which are modeled neither method. Notably, with a small modification our method learns to partially compensate for this. In Ours^{pose}, we condition the feature embedding network Φ_p by the skeletal poses jointly learned from scratch by our method. This improves the image quality by modeling the residual deformations. Finally, Ours^{SMPL} shows that replacing our skeleton in Ours^{pose} with a ground truth from an annotated SMPL template [16] does not affect the performance. This validates our skeleton initialization based on Medial Axis Transform.



Figure 2.14: Qualitative results on the *ZJU-MoCap* dataset, comparing our method without pose conditioning against WIM [21].

2.6 Discussion

Limitations and Future Work We demonstrate fast learning of articulated NeRFs for state-of-the-art view synthesis and straightforward skeletal reposing for objects with piece-wise rigid pose transformations. While the LBS allows for fitting partially non-rigid deformations (see Fig. 2.8 right), representing fully non-rigid, topologically varying, and multi-component objects would benefit from a higher-dimensional parameterization. Chen *et al.* [103] aim in that direction, but an intuitive and cost-effective reposing still favors skeleton-based techniques such as ours.

The performance of our method depends on the quality of the backbone reconstruction. While TiNeuVox [12] supports reconstruction from even sparse data, a different backbone could offer a more robust starting geometry and improve the skeleton initialization. Although our approach works well even for highly structured shapes and thin structures (see Fig. 2.8 right), we observe incorrect joint placement for objects with long chains of highly nonlinear geometrical transformations (see Fig. 2.8 left). Moreover, our approach successfully reduces the number of extraneous bones but it is not able to completely eliminate all superfluous skeletal elements (see Fig. 2.11). Our method is restricted to the kinematic motion space exhibited in the training sequences. While extrapolation is possible to some extent, the proposed method is not able to generalize to arbitrary unseen poses. Finally, we focus on image synthesis for user-defined poses rather than a direct fitting of unseen skeletal poses from images or motion capture. An inverse fitting of skeletal poses to novel 2D observations or transfer of poses from one object to another remain opportunities for future research.

Conclusion We presented a method for fast learning of articulated models for high-quality novel view synthesis of captured 3D objects. Our forward-warped neural point clouds avoid the pitfalls of backward warping and elegantly integrate a skeleton-based LBS. As a result, our method merges straightforward reposing even of strongly animated inputs, as present in the *Robots* dataset, with high visual fidelity of NeRF rendering. Our work is a significant step towards low-cost acquisition of animatable 3D objects for games, movies, and education.

Ethical Considerations Our method renders novel views and poses of 3D objects including human bodies. However, we do not focus on this class and we note that many human-specific methods exist (see Sec. 2.2). Nevertheless, we acknowledge that our

method could potentially be used to produce fake images of people and that research is needed to understand the risks and their mitigation.

3 MotionDreamer: Exploring Semantic Video Diffusion features for Zero-Shot 3D Mesh Animation

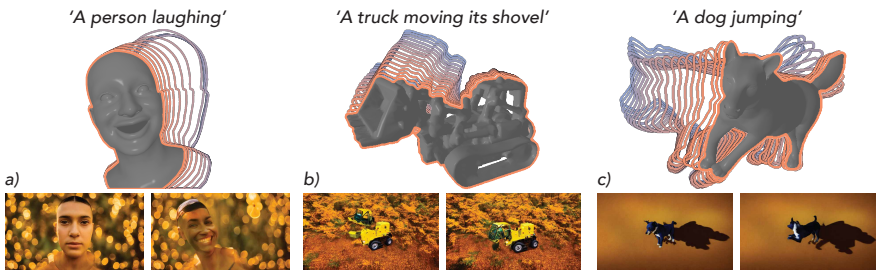


Figure 3.1: Our Zero-shot 3D mesh animations. From top to bottom: The desired motion description, the resulting animated mesh with motion contours, the driving video from a pre-trained video diffusion model. Notice robustness of our method to the temporal identity shift (a) and to the geometric distortions (b). Diverse shapes are supported through a range of animation models including a) FLAME [15], b) Neural Jacobian Fields [119] and c) SMAL [75].

Animation techniques bring digital 3D worlds and characters to life. However, manual animation is tedious and automated techniques are often specialized to narrow shape classes. In our work, we propose a technique for automatic re-animation of arbitrary 3D shapes based on a motion prior extracted from a video diffusion model. Unlike existing 4D generation methods, we focus solely on the motion, and we leverage an explicit mesh-based representation compatible with existing computer-graphics pipelines. Furthermore, our utilization of diffusion features enhances accuracy of our motion fitting. We analyze efficacy of these features for animation fitting and we experimentally validate our approach for two different diffusion models and four animation models. Finally, we demonstrate that our time-efficient zero-shot method achieves a superior performance re-animating a diverse set of 3D shapes when compared to existing techniques in a user study.*

* This chapter is based on the paper "MotionDreamer: Exploring Semantic Video Diffusion features for Zero-Shot 3D Mesh Animation," published at 3DV 2025. © 2025 IEEE. Reprinted, with permission, from 2025 International Conference on 3D Vision (3DV) [120].

3.1 Introduction

Animation is an important component of video games, simulators, and movies. It makes otherwise rigid environments come to life and is often a result of a tedious motion-data capture coupled to skilled manual editing [121]. However, this does not scale well for applications involving large virtual worlds with thousands of individual entities or for individual objects that are difficult to motion capture due to their physical size or real-world inaccessibility. For this reason, we propose an end-to-end generative method that re-animates static 3D objects using a pre-trained Video Diffusion Model [122–126] (VDM) without any additional training (Fig. 3.1).

We build on the remarkable success of Diffusion models [127, 128]. Beyond producing nearly photo-realistic 2D images [129–132], diffusion was also adapted for 3D [133, 134] and 4D shape synthesis [135–143]. However, the associated methods suffer from either a high optimization cost and low diversity [144] of the mode-seeking Stochastic Distillation Sampling [133] (SDS), or, as we show in our paper, they are susceptible to the visual artifacts in RGB outputs of existing VDMs. Furthermore, our method generates a unique animation as a sequence of object poses in a matter of minutes rather than hours common for end-to-end 4D generative methods. This is a feature crucial for processing of larger sample sets with subsequent filtering based on subjective preferences. Therefore, we position our approach into a category distinct from end-to-end 4D generation.

Instead of iterative SDS, we leverage the surprising versatility of semantic features extracted from diffusion models for down-stream tasks such as one-shot segmentation [145] or semantic feature matching [146, 147], which we adapt for motion fitting. We rely on a classical surface mesh representation in combination with diverse animation models [15, 16, 75, 119] to obtain animated 3D shapes that are fast to render, compatible with existing rendering frameworks and versatile across object classes.

In summary, we present the following contributions: 1. We introduce a novel zero-shot generative method for 3D mesh animation based on rendering in the semantic feature space of pre-trained VDMs. 2. We analyze effectiveness of VDM features for pose estimation to validate our method and design choices. 3. We evaluate two different VDMs and four animation models and demonstrate a preference of our 3D animations to existing generative approaches in a user study.

3.2 Related Work

Our method exploits VDMs to create novel animations of 3D objects. Here we discuss relevant work on video generation and existing approaches for 3D shape representation and animation.

3.2.1 Video generation

Generative visual models have advanced rapidly from Variational Auto-Encoders [148], Normalizing Flows [149, 150] and Generative Adversarial Models [151] to Diffusion Models [127, 128] and Continuous Normalizing Flows [152] achieving a nearly photorealistic

image synthesis [129–132] as well as state-of-the-art video synthesis [122–126]. Surprisingly, the features learned by the U-Net [153] of many diffusion models exhibit semantic properties useful for down-stream tasks such as segmentation [145] and feature matching [146, 154, 155]. Consequently, we analyze utility of two such models [123, 124] for our motion fitting, while we leave opportunities presented by recent large VDMs [156, 157] utilizing Visual Transformers [158] as an avenue for future research.

3.2.2 Shape and pose representations

There exist many ways for representing 3D shapes from classical explicit representations including point-clouds, voxels or surface meshes favored in real-time applications, to implicit neural shape representations [2, 3, 26] enabling photorealistic 3D scene reconstruction. In the middle, 3D Gaussians [159] have been shown to combine advantages of both at an increased storage cost. In this chapter we focus on surface meshes for their fast rendering, efficient storage and wide application support.

While animation of object poses can be encoded as a sequence of static representations [160], specialized representations ease editing for both arbitrary and class-specific shapes. In the first category, deformation fields offer maximal flexibility for dense volumetric optimization [4], Neural Jacobian Fields (NJFs) [119] offer space-time continuity and smoothness for surface optimization and external cages reduce the control space for easier editing [111]. In the second category, low-dimensional templates support manual animation and motion capture by combining Linear Blend Skinning [13] and Blend Shapes [161, 162] for specific classes of shapes such as faces [14, 15], bodies [16, 17], hands [74], or even animals [75]. Our method is agnostic to the choice of an animation model, which we test on high-dimensional NJFs [119] and on low-dimensional templates [15, 16, 75].

3.2.3 3D motion and animation

Capture Motion, most often for humans, can be directly captured [163] using sparse inertial sensors [164] or dense visual observations [165] either with tracking markers [166] or without them [167]. For a monocular video, we can estimate 2D poses [168–170] and uplift them to 3D [171–175] thanks to data priors [17, 176] based on large motion datasets [177–179]. However, the specific training for each class limits generalization. In contrast, recent advances in neural rendering [2, 3, 159] enabled class-agnostic 4D reposable reconstructions [1, 21, 110]. Our method is similarly based on class-agnostic differentiable pose optimization but differently from a direct image supervision, we exploit diffusion features of a monocular video rather than multi-view observations.

Generation Learned priors can also be used for text-conditioned motion synthesis [180]. However, this is in practice limited to human domain [181–184] where annotated 3D motion datasets exist [185, 186] or to other skeletal shapes [187] if at least 2D annotations are available. Alternatively, image and video generative models enabled class-agnostic joint shape and motion 4D generation [135–143] is usually based on Stochastic Distillation Sampling (SDS) [133] which, however, narrows the sampled distribution [144] due it its mode-seeking behavior. Closest to us, Ren et al. [137] extract motion from a full

video input. Our method shares the idea of extracting motion from a video model but thanks to utilizing the feature space it produces more natural motion with fewer visual artifacts. Furthermore, we do not use 3D uplifting methods requiring background masks such as Zero-1-to-3 [188]. Additionally, by focusing on motion alone we achieve faster sampling. Finally, both captured or generated motion can be transferred from one shape to another [189], either based on morphological similarity [190, 191] or data-driven domain matching [192, 193]. We experimentally show that our method is preferable when neither of the two conditions can be satisfied.

3.3 Preliminaries

Our method exploits internal representation of VDMs. Here, we provide a brief summary of these models and semantic information encoded in their internal features.

3.3.1 Video Diffusion Models

VDMs are a type of a generative model producing video sequences by gradual denoising [127, 128] of a Gaussian-noise image sequence $\mathbf{z} \in \mathbb{R}^{L \times H \times W \times D_{\text{lat}}}$, where L is the frame count, H, W the spatial dimensions, and D_{lat} is 3 for RGB models or the latent feature dimension for Latent Diffusion [132]. The forward diffusion process $q(\mathbf{z}_t|\mathbf{z}_0, t)$ gradually transports $\mathbf{z_0} \equiv \mathbf{z}$ to the Gaussian-noise prior over T steps such that $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This is used to learn a denoising function $f_\theta(\mathbf{z}_t, t, \mathbf{c})$ as a θ-parameterized network approximating the reverse process $p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t, t, \mathbf{c})$. A commonly used ϵ-prediction training procedure minimizes an objective $\sum_{t,\mathbf{c},\mathbf{z},\epsilon} \left\|\epsilon - f_\theta(\mathbf{z}_t, t, \mathbf{c})\right\|_2^2$ across data and noise samples $\mathbf{z} \sim p_{\text{data}}$ and $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Finally, sampling the noise prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and reversing the diffusion yields video generation. The conditioning vector $\mathbf{c} \in \mathbb{R}^N$ is often a text embedding, image embedding or both, and it steers the process, often with a classifier-free guidance [194].

3.3.2 Semantic Diffusion Features

Intermediate activations of image diffusion networks have been shown to encode semantic features and provide robust correspondences across image samples [146, 154, 155]. We adopt the methodology of Tang et al. [146], where f_θ is parameterized by a U-Net. The semantic feature maps $\mathbf{A}_u \in \mathbb{R}^{H_u \times W_u \times A_u}$ are extracted from the intermediate activations of a U-Net layer u with height, width and feature size H_u, W_u, and A_u.

Given a pair of images with feature maps $\mathbf{A}_u, \mathbf{B}_u$ and a chosen spatial location $\phi^A \in \mathbb{R}^2$ in the first image, we find a semantically corresponding spatial location $\phi^B \in \mathbb{R}^2$ in the other image as $\phi^B = \arg\max_{\phi^B} \kappa(\mathbf{A}_u[\phi^A], \mathbf{B}_u[\phi^B])$, where

$$\kappa(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^{\mathrm{T}}\mathbf{b}}{||\mathbf{a}||_2||\mathbf{b}||_2} \tag{3.1}$$

is a cosine similarity $\kappa : \mathbb{R}^{A_u} \times \mathbb{R}^{A_u} \to \mathbb{R}$ and $\mathbf{x}[\phi]$ denotes spatial sampling of a map $\mathbf{x}$ at location ϕ, which we implement as a bilinear interpolation. For video, we treat each

frame as an image with its own feature map, and we optimize semantic correspondences of reposed meshes using Eq. 3.1.

3.4 Method

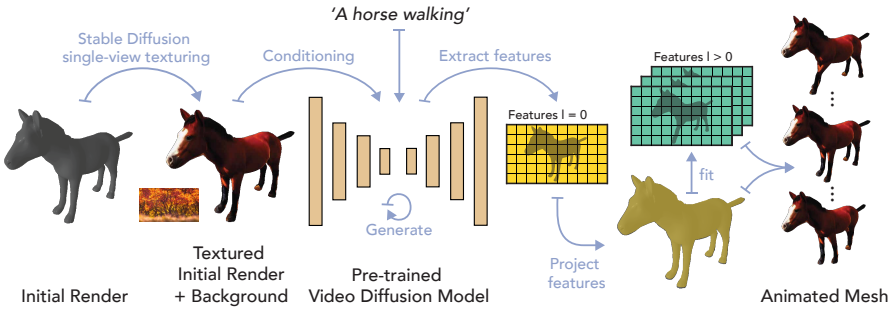


Figure 3.2: A diagram of our method. First, we automatically texture the input mesh $\mathcal{M}$ to reduce the domain gap to the VDM prior (Sec. 3.4.1). Second, we condition the VDM by a rendered image $\mathbf{I}_{rgb}$ to produce a video with motion and to extract features $\hat{\mathbf{A}}$ for all L frames from its internal U-Net (Sec. 3.4.2). Finally, we reproject the input frame features $\hat{\mathbf{A}}^0$ on the mesh surface and we optimize mesh animation parameters $\mathbf{p}$ to match the reposed mesh features to the video (Sec. 3.4.3).

Our methods accepts an unseen 2-manifold 3D mesh in an arbitrary pose and uses a pre-trained VDM to generate a temporal sequence of animation parameters (see Fig. 3.2 for an overview). We first describe our method for a general VDM and animation model before discussing specific realizations in Sec. 3.4.4.

Definitions We define the input mesh $\mathcal{M}$ as a tuple of N vertices and M triangular faces $\mathcal{M} := (\{\mathbf{u}_n \in \mathbb{R}^3 | n = 0, ..., N-1\}, \{\mathbf{f}_m \in \mathbb{N}^3 | m = 0, ..., M-1\})$. Next, we define $\tau : (\mathcal{M}, \mathbf{p}) \rightarrow \mathcal{M}'$ as a function transforming vertices to produce a mesh $\mathcal{M}' := (\{\mathbf{u}'_n\}, \{\mathbf{f}_m\})$ with a novel pose described by animation parameters $\mathbf{p} \in \mathbb{R}^P$. We refer to $\mathbf{p}_{\text{init}}$ as the input pose where $\tau(\mathcal{M}, \mathbf{p}_{\text{init}}) \equiv \mathcal{M}$ and, without a loss of generality, we assume it matches the first frame. Finally, $r_{rgb} : (\mathcal{M}, \mathcal{C}, \mathcal{T}, \mathbf{B}) \rightarrow \mathbf{I}_{rgb}$ is a rendering function producing an RGB image $\mathbf{I}_{rgb} \in \mathbb{R}^{H \times W \times 3}$ of the mesh $\mathcal{M}$ for a manually defined canonical camera $\mathcal{C}$, surface texture $\mathcal{T}$, and a background image $\mathbf{B} \in \mathbb{R}^{H \times W \times 3}$.

3.4.1 Single-View Texturing

While the visual datasets used to train existing VDMs are very large, they favor natural looking textured images with backgrounds (see Appendix B.4.1 for examples). We reduce the domain gap for our rendered image by automatically generating an RGB texture $\mathcal{T}$ and a semantically fitting background image $\mathbf{B}$. First, we render a depth map and a foreground mask ψ for a single fixed viewpoint of $\mathcal{M}$. Next, we style-transfer the depth map using a pre-trained ControlNet diffusion model [195] conditioned by a user-provided textual description to obtain a textured RGB image $\mathbf{S}$. Then, we crop the foreground texture $\mathcal{T} = \textit{unproject}(\mathbf{S} \odot \psi)$ and apply it to the mesh $\mathcal{M}_0$ using projective texturing [196]. Importantly, we do not strive for a complete texture of the entire mesh, but merely for

a stylization of the single-view VDM input image. Finally, we obtain the background image $\mathbf{B}$ by inpainting the remainder of $\mathbf{S}$ outside of the foreground bounding box using Stable Diffusion XL [197]. See Appendix B.2.2 for prompt details.

3.4.2 Motion Generation

The motion produced by our method originates from a VDM conditioned by our rendered mesh image $\mathbf{I}^0_{rgb} = r_{rgb}(\mathcal{M}^0, \mathcal{C}, \mathcal{T}, \mathbf{B})$ and an embedding of the intended motion text description. We sample the generator in a multi-step diffusion process over T steps denoted as $t \in [0, ..., T-1]$ with scheduling details specific to each VDM. Because the temporally incoherent visual artifacts in RGB video outputs make motion tracking difficult (see Fig. 3.1), we extract semantically meaningful U-Net features $\mathbf{A}^t_u$ at time step $t = \hat{t}$ and U-Net layer $u = \hat{u}$ as explained in Sec. 3.3.2, and we show that this improves the fitting accuracy. We motivate our choice of $\hat{t}$ and $\hat{u}$ in Sec. 3.5.3, and will omit the suffixes from now on for brevity, such that $\hat{\mathbf{A}} \in \mathbb{R}^{L \times \hat{H} \times \hat{W} \times \hat{A}} \equiv \mathbf{A}^{\hat{t}}_{\hat{u}}$ and $\hat{\mathbf{A}}^l$ selects the video frame l of L. We further assume $\hat{\mathbf{A}}^0$ corresponds to the input image $\mathbf{I}^0_{rgb}$ (see Appendix B.1.1 for a discussion).

3.4.3 Motion fitting

Given the known correspondence of the mesh $\mathcal{M}$, initial pose $\mathbf{p}_{\text{init}}$, image $\mathbf{I}^0_{rgb}$ and features $\hat{\mathbf{A}}^0$ for the input frame $l = 0$, we aim to recover all animation parameters $\mathbf{p}^l$ for $l \in [0, ..., L-1]$. We achieve this by optimizing $\mathbf{p}$ to match reprojections of the input $\hat{\mathbf{A}}^0$ to $\hat{\mathbf{A}}^l$ extracted from the video. To this goal, we first reproject $\hat{\mathbf{A}}^0$ to new poses $\mathbf{p}^l$ and optimize these poses using a gradient descent.

Feature Reprojection Our mesh pose fitting is based on reprojection of $\hat{\mathbf{A}}^0$ to any new pose $\mathbf{p}^l$. First, we use projective texturing to map $\hat{\mathbf{A}}^0$ to $\mathcal{M}$. We obtain per-vertex features $\{\mathbf{a}_n\}$ by mapping each mesh vertex $\mathbf{u}_n$ to the image plane of the camera $\mathcal{C}$ and sampling $\hat{\mathbf{A}}^0$ as $\mathbf{a}_n = \hat{\mathbf{A}}^0[P(\mathbf{u}_n, \mathcal{C})]$, where $P(.)$ is a world space to image plane projection function and $[.]$ is a bilinear sampler. Finally, we transform $\mathcal{M}$ to $\mathcal{M}^l = \tau(\mathcal{M}, \mathbf{p}^l)$ for a given novel pose $\mathbf{p}^l$ and we render a feature image

$$\mathbf{I}^l_{\mathbf{A}} = r_{\mathbf{A}}(\mathcal{M}^l, \mathcal{C}, \{\mathbf{a}_n\}, \mathbf{B}_{\mathbf{A}}) \tag{3.2}$$

where $r_{\mathbf{A}}$ is a rasterization function interpolating the vertex attributes $\{\mathbf{a}_n\}$ and $\mathbf{B}_{\mathbf{A}}$ is a background feature map produced by inpainting the background $\hat{\mathbf{A}}^0 \odot (1 - \psi)$ with a mean of valid features. Notice that Eq. 3.2 implies an approximate identity $\mathbf{I}^0_{\mathbf{A}} \approx \hat{\mathbf{A}}^0$, and we optimize $\mathbf{p}$ to improve this match for the full animation.

Mesh Pose Optimization We observe that direct optimization of each $\mathbf{p}^l$ independently is prone to local minima. Instead, we exploit the implicit bias of Multi-Layer-Perceptrons (MLPs) towards smooth functions, and regress $\mathbf{p}^l$ as a frame-dependent offset from an initial pose $\mathbf{p}_{\text{init}}$ such that $\mathbf{p}^l = \alpha m_\omega(\gamma(l)) + \mathbf{p}_{\text{init}}$, where $\alpha = 0.01$ is a scaling constant, γ is a frequency encoding [2], and $m(.)$ is an MLP with learnable parameters ω. We optimize

ω by gradient descent to enforce semantic correspondences between the animated mesh and the video, i.e. $\mathbf{I}^l_{feat} \approx \hat{\mathbf{A}}^l$, using the rendering loss:

$$\mathcal{L}_r = 1 - \frac{1}{L\hat{H}\hat{W}} \sum_{l=0}^{L-1} \sum_{i \in \Omega_{\mathbf{A}}} \kappa(\mathbf{I}^l_{feat}[i], \hat{\mathbf{A}}^l[i]), \tag{3.3}$$

where $\kappa()$ is the cosine similarity (Eq. 3.1), $\Omega_{\mathbf{A}}$ is the spatial domain of $\hat{\mathbf{A}}$ and $[i]$ a spatial sampler.

Regularization losses First, our monocular video provides only a limited supervision for motion-in-depth. We discourage the optimization from explaining spatial deformation artifacts in the input video via motion-in-depth by per-vertex regularization loss

$$\mathcal{L}_d = \frac{1}{LN} \sum_{l=0}^{L-1} \sum_{n=0}^{N-1} ||(\bar{d}^0 - d^0_n) - (\bar{d}^l - d^l_n)||_1, \tag{3.4}$$

where d^l_n is the projected depth of vertex u_n in frame l, and $\bar{d}^l = 1/N \sum_{n=0}^{N-1} d^l_n$. Second, we enforce temporal smoothness beyond the MLP's implicit bias to further reduce jitter using the smoothness loss $\mathcal{L}_s = 1/((L-1)N) \sum_{l=0}^{L-2} ||\mathbf{p}^l - \mathbf{p}^{l+1}||_1$. Lastly, we penalize propagation of local spatial distortions from video by suppressing large deformations using the fidelity loss $\mathcal{L}_f = 1/(LN) \sum_{l=0}^{L-1} ||\mathbf{p}^l||_1$. Consequently, our complete optimization objective is $\mathcal{L} = w_r\mathcal{L}_r + w_d\mathcal{L}_d + w_s\mathcal{L}_s + w_f\mathcal{L}_f$ with $w_r = 5$, $w_d = 0.01$, $w_s = 0.1$, $w_f = 0.01$.

3.4.4 Implementations Details

We implement our method in PyTorch [198] with PyTorch3D [199] mesh rasterizer. We optimize the pose fitting for 1 000 iterations. We initially optimize only $\mathbf{p}^0$ and linearly increase the number of optimized frames from 1 to L between iterations 0 and 500. We use a constant learning rate of 0.0005 and Adam optimizer [200] in Pytorch [198]. Our MLP m consists of 6 layers, each with a hidden dimension size of 256, and we scale the final output by a constant $\alpha = 0.01$. We apply a frequency encoding [2] for the input l: $\gamma(.) = (l, sin(2^0\pi l), cos(2^0\pi l), ..., sin(2^{k-1}\pi l), cos(2^{k-1}\pi l))$ with $k = 6$. For each input shape $\mathcal{M}$, we define the canonical camera $\mathcal{C}$ manually.

Animation Models We experiment with four animation models for poses $\mathbf{p}$. For domain specific shapes, we use low-dimensional articulated models SMPL [16] (for humans), SMAL [75] (animals) and FLAME [15] (faces), where $\mathbf{p}^l$ are the joint angles and the other shape parameters are fixed. For other meshes, we use Neural Jacobian Fields (NJF) [119], which encodes the pose $\mathbf{p}^l$ by surface Jacobians, in combination with a single global translation and rotation - see Appendix B.1.2 for details and for an additional rigidity regularizer $\mathcal{L}_j$ applied for NJF.

VDMs We evaluate 2 VDMs: VideoComposer [123] (VC) and DynamiCrafter [124] (DC) with $\hat{\mathbf{A}}$ resolution of $(160, 88)$ and $(128, 72)$ respectively (1/8 of their outputs). We use their recommended inference schedulers with $T = 50$ steps. Our assumption of

$\hat{\mathbf{A}}^0 \sim \mathbf{I}^0_{rgb}$ is satisfied by design for VC, and we present a solution for DC in Appendix B.1.1. We empirically find VC performs better for images with the background $\mathbf{B}$, while DC performs well even with a uniform white background. Additionally, we assessed another VDM, Stable Video Diffusion [122], but we discarded due to its low motion quality (see Appendix B.4.1, B.4.5).

3.5 Experiments

We compare our zero-shot motion generation to other methods in a user study. Further, we quantitatively evaluate our pose fitting algorithm on a synthetic human motion dataset and measure the contribution of the individual components in an ablation study.

3.5.1 User Study

We compare our method to two other approaches for zero-shot 3D motion synthesis. First, we compare to DG4D [137] as an end-to-end shape-and-motion generative method based on image and video diffusion. Second, in absence of a class-agnotic method, we compare to a human motion diffusion model (MDM) [184] combined with motion retargeting (MT) [190]. We provide an additional qualitative comparison to a contemporary end-to-end generative method Consistent4D [142] in Appendix B.4.2. We run our method with both VC and DC backbones and use the same generated videos as inputs for DG4D (see Appendix B.2.1 for details). We use 10 meshes and a total of 10 prompts combined to obtain 2 human stimuli (using the SMPL mesh), 2 horses (SMAL), 2 faces (FLAME) and 4 other stimuli each with a unique mesh (NJF). See Fig. 3.1, Fig. 3.3, and the supplemental video for visual examples, Appendix B.2.2 for a complete list.

12 participants aged 24–41, naïve to the purpose of the experiment and with a normal or corrected-to-normal vision, participated in a $\sim$ 20 min low-risk IRB approved study, after signing an informed consent without any compensation. 16 frame (1 second) long video pairs from different methods were presented side-by-side in random order. Each displayed the same untextured animated shape from two viewpoints to clearly display the motion. Videos were looped until a binary answer was entered using a keyboard. The same stimuli were used for three different questions in three blocks. See Appendix B.2.3 for details.

In Fig. 3.4 (Left), we observe a statistically significant preference for our method compared to both DG4D and MDM-MT in terms of having "more natural motion", "fewer visual artifacts" and "capturing the prompt better" ($p < 0.001$, binomial test). We provide a break-down for individual shapes in Appendix B.2.4. As expected, the human-specific MDM-MT approach excels for human stimuli but fails for morphologically distinct shapes, where correspondences are difficult to establish, which results in semantically incorrect and visually distracting motion (see Fig. 3.3 "Bunny"). In contrast, the other class-agnostic model, DG4D, struggles to accurately represent the video motion sequences leading to noisy reconstructions (see Fig. 3.3 "Raptor"). Moreover, the motion optimization in DG4D (Stage 2) takes 233 ± 5 seconds on an NVIDIA RTX 3090, while our method leverages fast rasterization and performs pose optimization in only 148 ± 39 seconds. Fig. 3.1, Fig. 3.3, Appendix B.2.4, and our video show more examples.

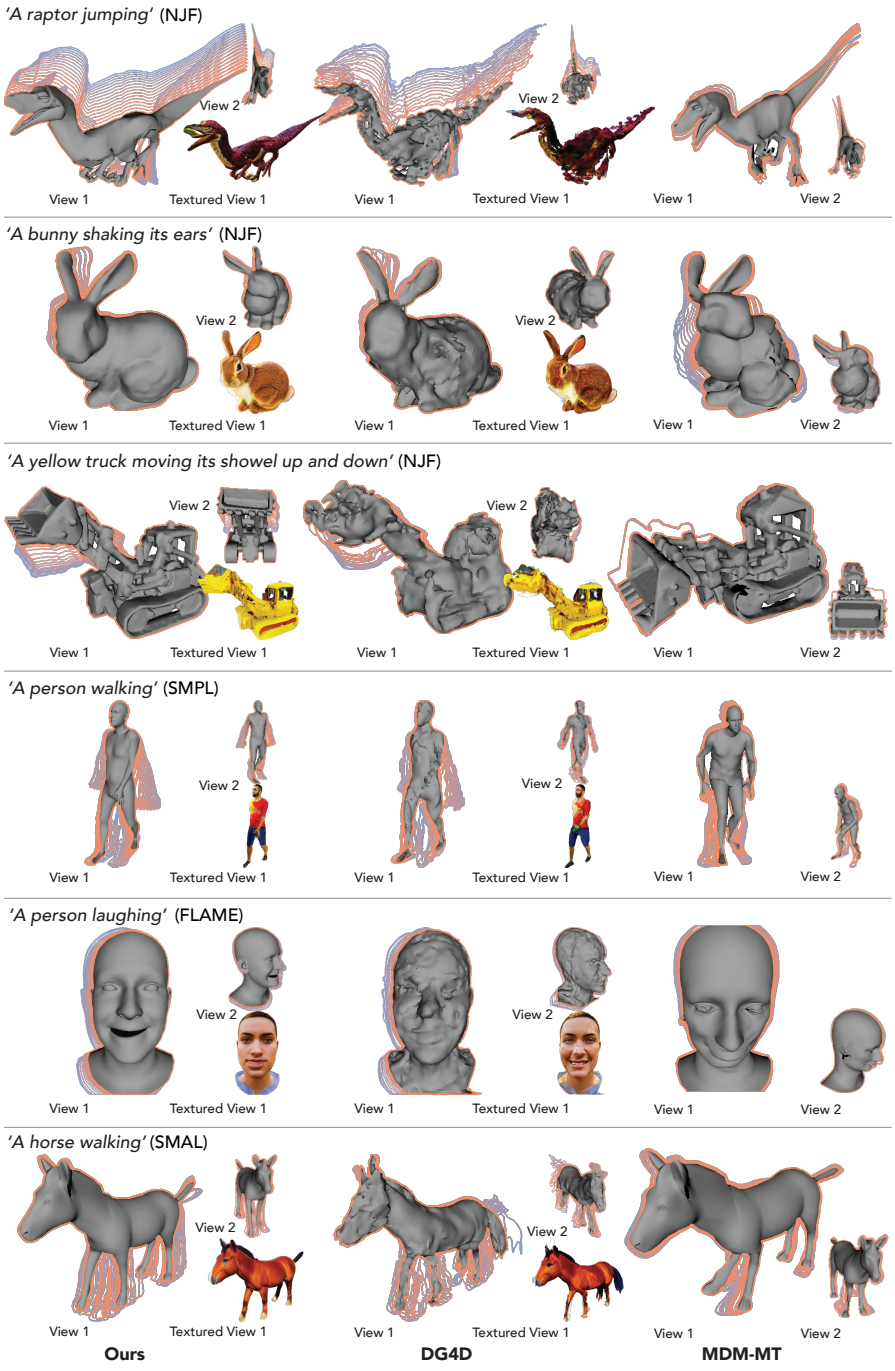


Figure 3.3: A qualitative comparison of our method to DG4D and MDM-MT for the prompts and the shapes used in our study. We display 2 untextured views of the last frame with one one additional textured image for reference. The contours convey the motion trajectory.

Input Accuracy After the main study, we asked each participant to additionally compare our textured output to the full DG4D color rendering and to the unprocessed VDM videos for overall preference (see the last bars Fig. 3.4 Left). The participants strongly prefer our method to DG4D, likely due to the more accurate geometry (Fig. 3.3). There was no effect when comparing to the VDMs, suggesting that our method closely preserves characters of the generated videos and should, therefore, benefit from future VDMs with a more accurate motion depiction.

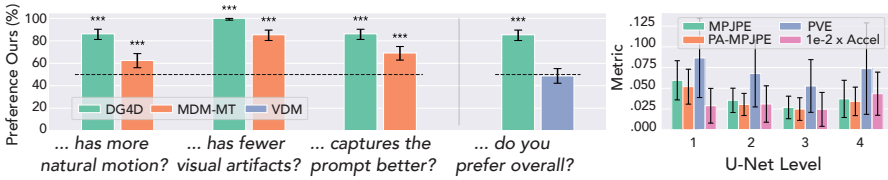


Figure 3.4: Left: Results of the user study, asking the question: *"Which video... ?"* For the first three questions we compare our method against untextured renders of DG4D and MDM-MT. For the last question we compare against the full RGB outputs of DG4D and the VDM output. *** denotes significance at $p < 0.001$ (bars show 95% confidence intervals, binomial test). Right: Pose fitting errors for $\mathbf{A}_u^{\hat{i}}$ extracted across U-Net layers u with bars showing standard deviations.

Mixed-Effects Logistic Regression In addition to the previous evaluation, we analyze the responses with a model that does not assume independence between user answers. For each question and baseline, we fit an intercept-only mixed-effects logistic regression with random intercepts for user and stimulus, testing whether the preference for our method differs from random chance (i.e., 50%). This corresponds to the R expression `glmer(preference ~ 1 + (1 | user_id) + (1 | stimulus_id), family = binomial)` using the lme4 package [201].

Compared to the results obtained from the previous test, the preferences remain significant at the $\alpha = 0.05$ level under a Wald z-test on the fitted intercepts, with one exception: No significant difference can be observed for the MDM-MT comparison on "Which video has more natural motion?" ($p = 0.086$). The VDM comparison on "Which video do you prefer overall?" remains insignificant ($p = 0.75$). Furthermore, the DG4D comparison on "Which video has fewer visual artifacts?" cannot be fit with this model, as all 240 responses are in favor of our method.

3.5.2 Pose Optimization

We observe that the limiting factor of our method is the VDM motion quality. To remove this influence, we quantitatively evaluate performance of our pose fitting component using a captured human dancing motion dataset AIST++ [202] with known poses. First, we randomly select 20 test sequences and re-render the first 20 frames from each using the available SMPL mesh to simulate a perfect VDM. Then, we use VC to extract $\hat{\mathbf{A}}$ from the rendered video following Tang et al. [146] and optimize $\mathbf{p}$ for the SMPL model (Sec. 3.4.3) before evaluating the common metrics [175]: the Mean Per Joint Position Error (MPJPE), the Procrustes-aligned MPJPE (PA-MPJPE), the Per Vertex Error (PVE), and finally the Acceleration error (Accel) for smoothness.

Table 3.1: The pose fitting performance of WHAM [175] and variants of our method for re-rendered AIST++ human body sequences [202]. Less is better for all metrics (see Sec. 3.5.2).

	MPJPE	PA-MPJPE	PVE	Accel
***Textured* (default)**				
WHAM	.059 ± .029	.042 ± .016	.075 ± .036	7.9 ± 9.0
RGB	.044 ± .051	.044 ± .042	.077 ± .059	7.5 ± 16.7
Ours (full)	**.041** ± .036	**.039** ± .035	**.063** ± .057	**5.0** ± 7.2
Untextured				
WHAM	.057 ± .028	**.039** ± .015	**.070** ± .035	7.4 ± 9.1
RGB	.146 ± .056	.126 ± .043	.203 ± .074	**3.2** ± 3.0
Ours	**.051** ± .037	.044 ± .034	.073 ± .054	4.7 ± 5.7

Table 3.2: Performance of our ablated method variants in pose fitting. Notable performance impacts highlighted in red.

	MPJPE	PA-MPJPE	PVE	Accel
no $\mathscr{L}_s$	.027 ± .016	.025 ± .016	.053 ± .036	**2.72 ± 2.19**
no $\mathscr{L}_f$	.026 ± .017	.025 ± .018	**.103 ± .044**	2.52 ± 2.28
no $\mathscr{L}_d$	.025 ± .006	.024 ± .008	.046 ± .016	2.46 ± 2.04
Full	.027 ± .016	.025 ± .016	.053 ± .036	2.50 ± 2.36

We conduct three comparisons. First, we compare our single-view texturing (*textured*, Sec. 3.4.1) to a uniform gray shading (*untextured*). Second, we compare our semantic features $\hat{\mathbf{A}}$ (*Ours*) to RGB features (*RGB*) extracted directly from the input videos. Finally, we additionally test a state-of-the-art human pose estimation method WHAM [175] as a domain-specific reference. Since our method always starts with known $\mathbf{p}_{\text{init}}$, we emulate the same for WHAM by measuring its first-frame per-joint error and transform all predictions accordingly. This empirically improves WHAM scores relative to the unprocessed outputs. Appendix B.3 provides details and alternative alignment strategies.

Results As summarized in Table 3.1, *Ours* consistently achieves better results with *textured* inputs than with *untextured* inputs, which motivates our Single-View Texturing (Sec. 3.4.1). Furthermore, *Ours (full)* with semantic features $\hat{\mathbf{A}}$ achieves lower errors than the variant with *RGB* features, which documents the utility of these features for our task. Moreover, *Ours (full)* compares favorably even to the WHAM pose estimator despite the lack of human-specific training. This might be explained by the artificial appearance of our input videos which differ from common human pose estimation datasets. We do not claim general supremacy of our method for human pose estimation. This is showcased in Fig. 3.9 (right), where our method struggles to avoid physiologically implausible poses. Finally, in Fig. 3.5 we compare both features qualitatively in our full generative method and confirm that our semantic features lead to a better motion fit with fewer artifacts. See Appendix B.4.3 for more examples.

3.5.3 Ablations

We reuse the pose optimization experiment to validate our design choices. To this end, we follow the same procedure for 6 of the same AIST++ sequences [202]. First, we analyze the choice of $\hat{u}$ (Fig. 3.4 right) and $\hat{t}$ (Fig. 3.6 left) for extraction of $\hat{\mathbf{A}}$ using PA-MPJPE. We observe the best performance for $\hat{u} = 3$, which we consequently use for both VDMs

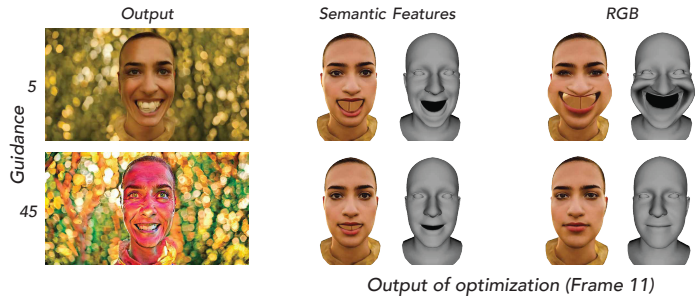


Figure 3.5: Effect of features on optimization under different guidance scales. Higher guidance scales are usually accompanied by stronger artifacts in RGB space (bottom row). The left column shows frame 11 of the VDM output, i.e. the target expression. The other columns show semantic features vs RGB used for optimization.

in our other experiments. We further find our method is not sensitive to the choice of $\hat{t}$ above $t \approx 15$. Therefore, we select $\hat{t} = 20$ for VC and $\hat{t} = 40$ for DC.

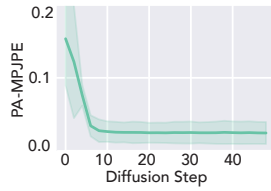


Figure 3.6: PA-MPJPE ↓ with a standard deviation range for features $\mathbf{A}_u^t$ extracted for different diffusion steps t.

Next, we ablate our regularization losses (Table 3.2). As expected, the smoothness of $\mathscr{L}_s$ reduces the Acceleration error, while $\mathscr{L}_f$ reduces shape distortions recorded by the Per-Vertex Error. In contrast, the depth regularization of $\mathscr{L}_d$ does not lead to an improvement in performance metrics, but we observe that it discourages perceptually-objectionable depth errors (Fig. 3.7 right).



Figure 3.7: Depth regularization prevents undesirable motion-in-depth explanations.

Finally, in Fig. 3.8 we ablate the mesh resolution in our method with NJF. We find that the output quality degrades gracefully and predictably with reduced vertex count which is favorable for our method's versatility. See Appendix B.4.4 for additional examples.



Figure 3.8: Effect of number of vertices on our method. Model source: Jaka Ardian 3D art / model from Indonesia.

3.6 Discussion

Limitations and Future Work Single-view motion supervision struggles to resolve motion-in-depth or occlusions, which we mitigate using regularization at a risk of overall motion reduction (see Fig. 3.7 right). We acknowledge this as a limitation and a motivation for further research which could offer an improvement through multi-view supervision at the cost of additional training data [187]. We demonstrate a zero-shot method supporting a range of animation models, but we acknowledge that the high degree-of-freedom in NJF permits undesired distortions (see Fig. 3.9a). These could be potentially remedied though a static shape supervision inspired by 3D generative models [134] with a possible diversity reduction stemming from SDS [144]. On top of this, the motion produced by the current VDMs might not adhere to the prompt or might contain physically impossible transitions (see Fig. 3.9b). To counter this, the fast run-time of our method could be combined with a suitable rejection heuristic. Furthermore, we expect to benefit from future VDM improvements [156, 157]. This will also allow for generating longer sequences, necessitating memory off-loading, which is currently absent in our implementation. Finally, an interesting future direction is to close the loop and constrain the VDM generation with our simultaneously optimized 3D animation model in order to prevent any spatiotemporal distortions from emerging.

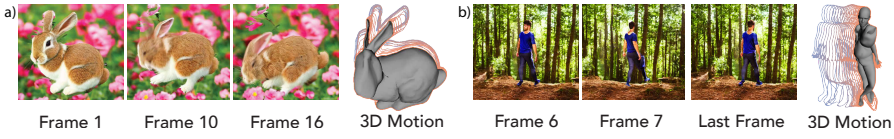


Figure 3.9: Failure cases showing frames of the VC VDM output and our fitted motion. (a) The VDM produces fast motion accompanied by ear disappearance that our model explains as an undesired head deformation. (b) The VDM suddenly flips body orientation by 180 degrees which confuses our tracking and leads to self-intersections.

Conclusion We presented a novel generative method for zero-shot 3D animation. Despite its limitations stemming from the single-view supervision, we demonstrated that it produces visually preferable motions across diverse unseen 3D shapes at computation cost lower than end-to-end 4D generative methods. We see our method as a capable tool for analysis of motion spaces in VDMs, and for affordable re-animation of otherwise static 3D assets in large-scale virtual environments.

Ethical Considerations Our method produces novel poses for 3D objects including human bodies and faces, but we do not focus on realistic appearance modeling. The biases in backbone VDMs can influence our method and are a priority research interest to the community.

Surface-Aware Distilled 3D Semantic Features

4

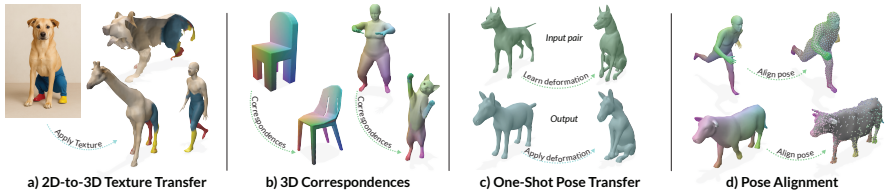


Figure 4.1: We introduce a *surface-aware* feature embedding space based on 2D pre-trained foundational models. In contrast to related works, our embedding space separates instances of the same semantic class (e.g. right vs. left instances for "hand/paw") which facilitates many downstream applications: a) Texturing of even incomplete 3D shapes based on a 2D image (the input image was produced by ChatGPT), b) 3D correspondences between non-isometric shapes, c) Re-posing of meshes based on a single source and target pair, d) Pose alignment of 3D meshes with dense and sparse point correspondences.

Many 3D tasks such as pose alignment, animation, motion transfer, and 3D reconstruction rely on establishing correspondences between 3D shapes. This challenge has recently been approached by pairwise matching of semantic features from pre-trained vision models. However, despite their power, these features struggle to differentiate instances of the same semantic class such as "left hand" versus "right hand" which leads to substantial mapping errors. To solve this, we learn a surface-aware embedding space that is robust to these ambiguities while facilitating shared mapping for an entire family of 3D shapes. Importantly, our approach is self-supervised and requires only a small number of unpaired training meshes to infer features for new possibly imperfect 3D shapes at test time. We achieve this by introducing a contrastive loss that preserves the semantic content of the features distilled from foundational models while disambiguating features located far apart on the shape's surface. We observe superior performance in correspondence matching benchmarks and enable downstream applications including 2D-to-3D and 3D-to-3D texture transfer, in-part segmentation, pose alignment, and motion transfer in low-data regimes. Unlike previous pairwise approaches, our solution constructs a joint embedding space, where both seen and unseen 3D shapes are implicitly aligned without further optimization.*

* This chapter is based on the paper "Surface-Aware Distilled 3D Semantic Features," published at Siggraph Asia 2025 [203].

4.1 Introduction

Mapping 3D shapes into a shared space guaranteeing mutual correspondences is important for many applications, including 3D registration, pose alignment, motion transfer, as well as static and dynamic 3D reconstruction. Historically, geometric descriptors have been used to determine matches between pairs of 3D shapes under isometric deformations, but they struggle with non-isometric deformations [204–206]. In contrast, neural features, stemming from pre-trained 2D vision models, have recently achieved great success in identifying correspondences between pairs of vastly different shapes [146, 154, 207, 208], such as mapping from cats to lions. In this chapter, we make another important step by moving from one-to-one pairwise shape correspondence matching to a joint embedding space establishing many-to-many shape correspondences.

Despite their inter-class robustness, neural features often struggle to disambiguate between instances of the same class like "left hand" and "right hand" (see Fig. 4.3). Such mismatches can lead to substantial errors in downstream applications (see Sec. 4.6). Recent research has demonstrated that these features contain global pose information and that disambiguation is possible in a 2D scenario [209]. However, achieving the same effect on distilled 3D features is not trivial, especially in a low-data regime, which is prevalent in 3D, where data acquisition and labeling is difficult.

Our work improves 3D neural features distilled from pre-trained 2D vision models by embedding them into a space disambiguating intraclass instances. We achieve this without large annotated datasets using a self-supervised learning scheme guided by in-shape geodesic distances without the need for shape pairs. Training with a limited number of 3D meshes, our method produces space of *surface-aware features* establishing multi-faceted correspondences for diverse new shapes without any further fine-tuning. In quantitative and qualitative comparisons to prior work, we demonstrate superior suitability of these features to serve as robust descriptors for matching and as building blocks for solving other tasks. Since geodesic distances are not used during inference, our method has only a minimal overhead from its shallow neural encoder and its point-wise nature makes it robust to varying mesh complexity or shape incompleteness. Finally, the encoder preserves compatibility with per-pixel image features, and hence, also naturally establishes robust 2D-to-3D mappings.

In summary, we make the following contributions: 1. We introduce a novel contrastive loss for self-supervised distillation of 3D features. 2. We quantitatively demonstrate the effectiveness of our surface-aware features in pose transfer, correspondence matching and skinning weight regression. 3. We showcase versatility of our approach in additional downstream applications including pose alignment, instance-based part segmentation and 2D-to-3D or 3D-to-3D texture transfer. 4. We show the versatility of our features when matching many-to-many shapes of not only humanoids and animals but also other classes.

4.2 Related Works

Our method utilizes contrastive learning to embed semantic features from foundational models to a space enabling robust n-to-n 3D shape matching. In this section, we discuss prior work in these three areas.

4.2.1 Image-based features for 3D shapes

Image-based features emerge in large visual models for 2D image tasks. Self-supervised features from Vision Transformers, such as DINO-ViT [210] and DINOv2 [211], locally encode semantic information useful for segmentation [210] or image-to-image correspondence matching [212]. SD-DINO [213] adds complementary features from the diffusion-based image synthesis model Stable Diffusion [132]. Lifting these features to 3D has enabled the self-supervised construction of canonical surface maps [214], transfer of appearance between 3D shapes [215], 3D animation [120], keypoint detection [208] or matching of surface correspondences [147, 216, 217]. However, despite their semantic versatility, disambiguating between intraclass instances, such as left and right hands, remains challenging but possible, as shown in a recent 2D image study [209]. This motivates our 3D shape descriptors for resolving in- stance ambiguity.

Prior work tackled this ambiguity by mapping shapes to a spherical template [218], which is difficult for complex shapes including humans. Alternatively, Liu et al. [219] recovered non-isometric correspondences from a 2D semantic flow learned from vision features. Instead, we adapt Diff3F features [147] in 3D space and resolve the ambiguity through contrastive learning enforcing geodesic distances.

Geodesic distances have previously supported point cloud analysis [220] and more recently, NIE [221] and the concurrent work DV-Matcher [217] similarly utilize geodesic distances for feature disambiguation. Yet, the previously mentioned methods rely on aligned mesh pairs or a learned alignment, while our method learns purely from intrinsic properties of individual shapes. This eases adaptation to less common classes beyond humanoids and animals (Fig. 4.8) and cross-class mappings (Fig. 4.9).

Beyond vision-only models, multimodal large language models have recently been effective in image and 3D shape analysis including keypoint labeling [222] and shape co-segmentation [223]. In our work, we focus on vision-only models because of their simplicity, but we consider a model combination a promising research direction.

4.2.2 Contrastive Learning

Contrastive learning embeds similar samples close to each other while keeping dissimilar samples apart. This can be achieved directly by minimizing and maximizing embedding distances for positive and negative pair samples, respectively [224–227] or indirectly, such as by optimizing performance in an auto-regressive task [228]. Training pairs can be obtained by data augmentation [229], from memory banks [230, 231], or by clustering [232, 233]. Learning with cross-domain labels yields joint embeddings, as demonstrated by CLIP [234] for text and images. Contrastive learning was applied to learn end-to-end pose transfer from multiple unregistered meshes of the same identity in different poses [235]. We design our contrastive loss to disambiguate intra-class instances guided by a geodesic metric, while learning from intrinsic properties of individual meshes rather than same-identity shape pairs.

4.2.3 Shape correspondences

Point-to-Point Classical shape registration methods directly minimize global [236] or local [237] inter-shape distances making them susceptible to local minima [238]. This motivates the design of more informative local geometric descriptors [204–206]. These can alternatively be learned [239, 240] from voxelized patches [241–243] or from point clouds [244–248]. The learning can be supervised by labels [239] or achieved without them [244, 249–251]. Our method falls into the latter category, as our contrastive loss motivates our encoder to separate instances by approximating geodesic distances [252] without training data labels. This is conceptually similar to previous methods for near-isometric shape deformations [253–255]. However, we distinctly do not measure geodesic distortions between shape pairs, and therefore we do not limit our method to isometric deformations, and we do not compute any geodesics during inference. Instead, we only use the geodesics to disambiguate information already available in the image-based features, which is critical for our results.

The correspondences can be recovered from descriptors by a matching [256], directly regressed [257, 258] or established on parametric templates [249, 259]. Here, we focus on the descriptors themselves, and we show several different application scenarios in Sec. 4.6.

Surface mapping Functional Maps (FMs) [260] allow for matching on a surface. FMs are real-valued surface functions in the space of Laplace-Bertrami eigenfunctions, supporting linear transformations between shapes. Constrained to match surface descriptors for each shape [204–206] they allow extracting point-wise correspondences [260, 261]. These functions can also be learned [262] often with little or no supervision [254, 263–266]. Extrinsic alignment can support nonisotropic deformations [267, 268]. In this chapter, we focus on improving features for direct point-to-point matching in the spatial domain, but we later demonstrate a combination of our features with FM.

4.3 Preliminaries

We build upon methods that aggregate features from pre-trained 2D vision models on 3D meshes [147, 208, 216, 217]. In this section, we give a brief overview on these methods.

4.3.1 Reprojection of 2D Features

We represent a 3D shape as a triangular mesh with a tuple of N vertices and M triangular faces, that is, $\mathcal{M} := (\{\mathbf{p}_n \in \mathbb{R}^3 | n = 1, ..., N\}, \{\mathbf{t}_m \in \mathbb{N}^3 | m = 1, ..., M\})$. The rendering function $R_{rgb} : (\mathcal{M}, \mathcal{C}) \rightarrow \mathbf{I}_{rgb}$ projects $\mathcal{M}$ into a camera $\mathcal{C}$ and outputs an image $\mathbf{I}_{rgb} \in \mathbb{R}^{H \times W \times 3}$, with height H and width W. Optionally, texturing is possible in $R_{rgb}(.)$ or as a ControlNet [195] post-processing. The image is then passed to a pre-trained vision model [132, 210, 211, 213] to obtain dense semantic feature maps $\mathbf{F} \in \mathbb{R}^{h \times w \times f}$ with h, w, f as two spatial and one feature dimension. Finally, per-vertex features $\mathbf{f}_n \in \mathbb{R}^f$ are obtained by projective texture mapping of $\mathbf{F}$ onto $\mathcal{M}$. To cover the whole surface, features are aggregated across multiple cameras, resulting in a set of features

$\mathcal{F}_{\mathcal{M}} := \{\mathbf{f}_n \in \mathbb{R}^f | n = 1, ..., N\}$. Throughout this chapter, we refer to $\mathcal{F}_{\mathcal{M}}$ as the *base features* on which our method is built. The exact choice of $\mathcal{F}_{\mathcal{M}}$ is orthogonal to our contribution but must encode semantic information. To this extent, we use Diff3F [147] in this chapter.

Correspondence Matching Features $\mathcal{F}_{\mathcal{M}}$ have been shown to encode strong semantic information useful for correspondence matching [146, 147]. In the simplest case, the feature $\mathbf{f}_n \in \mathcal{F}_{\mathcal{T}}$ of a target mesh $\mathcal{T}$ that best matches the feature $\mathbf{f}_m \in \mathcal{F}_{\mathcal{S}}$ of a source mesh $\mathcal{S}$ is determined by maximizing the cosine similarity $\phi : \mathbb{R}^f \times \mathbb{R}^f \rightarrow \mathbb{R}$:

$$\phi(\mathbf{f}_i, \mathbf{f}_j) = \frac{\mathbf{f}_i^T \mathbf{f}_j}{\|\mathbf{f}_i\|_2 \|\mathbf{f}_j\|_2}, \tag{4.1}$$

such that $\tau(\mathbf{p}_m) = \arg\max_{\mathbf{p}_n} \phi(\mathbf{p}_n \rightarrow \mathbf{f}_n, \mathbf{p}_m \rightarrow \mathbf{f}_m)$ is the best matching point. However, the features $\mathcal{F}_{\mathcal{M}}$ do not differentiate between semantic instances well (see Fig. 4.7) which we address by learning robust *surface-aware features* $\mathcal{S}_{\mathcal{M}}$.

4.4 Method

Our goal is to learn an embedding resolving instance ambiguities of the *base features* $\mathcal{F}_{\mathcal{M}}$ and obtain *surface-aware features* $\mathcal{S}_{\mathcal{M}}$ (see Fig. 4.2). We achieve this by training a point-based feature auto-encoder with a limited set of training meshes and our contrastive loss for self-supervision. At test time, we can produce *surface-aware features* for novel unseen shapes without additional fine-tuning.

4.4.1 Setup

Our method requires a potentially small set of training meshes $\mathbf{M}_t = \{\mathcal{M}_i | i = 1, ..., K\}$, each associated with *base features* $\mathcal{F}_{\mathcal{M}}$ obtained following Sec. 4.3 and normalized by a Euclidean norm such that $\forall \mathbf{f}_n \in \mathcal{F}_{\mathcal{M}}, \|\mathbf{f}_n\|_2 := 1$.

Unlike other approaches [221, 248], we do not require extrinsic canonical mesh alignment during training, because we rely only on intrinsic properties of individual meshes. Similar considerations were previously made for Functional maps [260]. Moreover, we inherit the rotation invariance of the *base features* demonstrated by Dutt et al. [147, Supplement "Robustness to Rotation"], although we observe a performance degradation if meshes are upside down and we avoid this in our inputs (see Appendix C.1.1).

Furthermore, our encoder is point-based and does not rely on shape completeness or a consistent topology. Both of these design choices favor generalization under transformations ranging from coordinate swap to shape reposing, and memory-unconstrained batch-based processing of even large shapes.

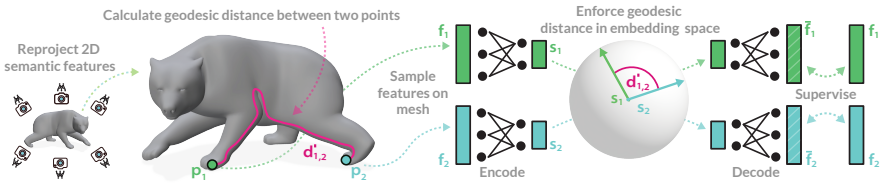


Figure 4.2: Overview of our method. We feed images of a 3D shape rendered from multiple viewpoints to a pre-trained 2D vision model and extract features that are then projected back onto surface points $\mathbf{p}_i$ and aggregated into per-point features $\mathbf{f}_i$ (Sec. 4.3). Next, we pointwise embed the *base features* $\mathbf{f}_i$ into our *surface-aware features* $\mathbf{s}_i$ residing in a lower-dimensional space learned using our contrastive loss preserving geodesic distances $d_{i,j}$ and a reconstruction loss matching decoded features $\mathbf{f}_i$ to $\mathbf{f}_i$ (Sec. 4.4). The *surface-aware features* $\mathbf{s}_i$ serve as robust descriptors for correspondence matching (Sec. 4.5) and base blocks for many down-stream applications (Sec. 4.6).

4.4.2 Separating Front Paw from Back Paw

Our embedding aims to separate multiple instances of the same class that are difficult to directly disambiguate in $\mathcal{F}_\mathcal{M}$. For example, consider the two surface points, $\mathbf{p}_1$ and $\mathbf{p}_2$, on the bear's paws in Fig. 4.2. The prevalent semantic significance of the "paw" concept hinders the separability of the corresponding *base features* $\mathbf{f}_1$ and $\mathbf{f}_2$. Fig. 4.3 illustrates this for human arms and Diff3F features [147]. To solve this, we train a point-wise feature autoencoder, producing our *surface-aware features* $\mathcal{S}_\mathcal{M} \subset \mathbb{R}^S$ in its embedding space. We motivate the feasibility of separation by the prior observations that vision features additionally carry information about the global pose [209]. We postulate that this enables our model to distinguish between part instances when guided by their intrinsic distance. We adopt the geodesic distance $d_{1,2}$ between $\mathbf{p}_1$ and $\mathbf{p}_2$ for this purpose. Following contrastive learning, we sample point pairs on a single shape to enforce $\phi(\mathbf{s}_1, \mathbf{s}_2) \approx d_{1,2}$ for $\mathbf{s}_n \in \mathcal{S}_\mathcal{M}$. We validate our choice of hyperspherical embedding space against Euclidean space in Sec. 4.5.5.

Model We train a *base feature* encoder $\mathscr{E}(.)$, such that, following a normalization, $\mathbf{s}_n = \mathscr{E}(\mathbf{f}_n)/\|\mathscr{E}(\mathbf{f}_n)\|_2$ is a *surface-aware feature* $\mathbf{s}_n \in \mathbb{R}^S$ in a hypersphere embedding. During training, we randomly sample an unpaired training mesh $\mathcal{M} \in \mathbf{M}_t$ with *base features* $\mathcal{F}_\mathcal{M}$, which we encode pointwise to obtain $\mathcal{S}_\mathcal{M}$.

In each training iteration, we use furthest-point sampling to choose a random subset of A anchor points $\mathbf{p}_a$ among the mesh vertices $\mathbf{p}_i \in \mathcal{M}$ and compute geodesic distances $d_{n,a}$ for each pair of a mesh and anchor point. We additionally rescale $d_{n,a}$ to a maximum of one, such that $d'_{n,a} := d_{n,a}/\max_{n,a}(d_{n,a})$, which removes the dependency on the scale of the mesh. We find this robust and leading to features later generalizing across morphologically equivalent shapes with different proportions (see an elephant vs. a giraffe in Fig. 4.8).

Subsequently, our contrastive loss preserves the rescaled geodesic distances in the embedding space:

$$\mathscr{L}_c = \frac{1}{NA} \sum_{n}^{N} \sum_{a}^{A} \left| d'_{n,a} - \left(\frac{1 - \phi(\mathbf{s}_n, \mathbf{s}_a)}{2} \right) \right|. \tag{4.2}$$

This loss operates in a hyperspherical embedding space and utilizes cosine similarity mapped to the $[0, 1]$ range. Hereby, $\mathscr{L}_c$ penalizes features close in the embedding space

but distant on the shape surface and vice versa.

Furthermore, we found it beneficial for the preservation of semantic information to train a feature decoder $\mathbf{f}_n = \mathscr{D}(\mathbf{s}_n)/\|\mathscr{D}(\mathbf{s}_n)\|_2$ in an autoencoder fashion. To this extent, we utilize a reconstruction loss:

$$\mathscr{L}_r = \frac{1}{N}\sum_n 1 - \phi(\mathbf{f}_n, \mathbf{f}_n). \tag{4.3}$$

We train both the encoder and the decoder end-to-end with the combined loss $\mathscr{L} = w_r\mathscr{L}_r + w_c\mathscr{L}_c$, where a choice $w_r = w_c = 1$ works well in our tests. We do not observe an increase in performance with a higher w_c.

Note that our training procedure, in contrast to related works [217, 248, 250], does not require target and source shape pairs.

4.4.3 Implementation

During preprocessing, we rasterize our triangular meshes and precompute *base features* for all vertices following Sec. 4.3. We implement our autoencoder in PyTorch2 [269] and use the Polyscope renderer [270] for visualizations. The encoder $\mathscr{E}$ is a Multilayer Perceptron (MLP) consisting of three blocks, where each block has two linear layers, SiLU activation [271], and layer normalization [272]. The first layer in each block employs a skip connection [273], while the second reduces the dimensionality by a factor of two. With Diff3F [147] as *base features*, $\mathscr{E}$ reduces feature dimensionality from $f = 2048$ to $s = 256$. The decoder $\mathscr{D}$ is a mirrored copy of the encoder. We train our model on NVIDIA RTX 3090 for 50k iterations with the AdamW optimizer [274] and a learning rate of 0.0001 which takes $\approx$ 2 hours.

We choose an exponential moving average [275] of the model with the lowest validation loss, without the need for any correspondence labels. Geodesic distances for training are calculated on the fly with the heat method [276] implemented in Geometry Central [277]. No geodesics are required during inference and the computational cost is determined by the Diff3F baseline with a only a negligible overhead from our shallow encoder $\mathscr{E}$. For a shape with 10k vertices, this is less than 5 milliseconds on top of $\approx$ 4 minutes from Diff3F. Moreover, downstream tasks cost benefits from the smaller feature dimensionality. Functional maps are calculated with the base algorithm [260], provided by the Diff3F implementation [147].

4.5 Experiments

Here, we first motivate the benefits of our *surface-aware feature* embedding space by visualizing its distribution. Next, we evaluate their effectiveness in tasks with quantitative benchmarks including pose transfer, skinning weight regression and 3D correspondence matching. Finally, we analyze the impact of our design choices in an ablation experiment.

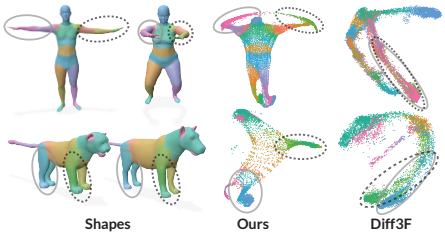


Figure 4.3: Two shapes (left) and a PCA-based 2D projections of their aggregated Diff3F *base features* and our *surface-aware features* (right). Notice the separation of limbs in our result compared to Diff3F. Our features originate from the same encoder for both shapes. The animal legs appear merged along the sagittal plane due limitations of the PCA projection, but they remain disambiguated in our feature space as demonstrated in Fig. 4.9.

Training We train a single autoencoder on a joint dataset consisting of 49 animal samples from the *SMAL* dataset [75] and 49 humans from the *SURREAL* dataset [249]. We choose 2 samples from each dataset for validation. We use this single shared model without any additional optimization for all experiments, unless stated otherwise.

4.5.1 Exploration of Embedding Space

To illustrate the effect of our contrastive loss on feature separation, we compare the 2D projections of the Diff3F [147] *base features* with our *surface-aware features*.

Setup We create two dataset $SMPL^{eval}$ and $SMAL^{eval}$ unseen during training. The former consists of 50 randomly-sampled *SMPL* [278] shapes and poses from *AMASS* [179], while the latter consists of 50 randomly-sampled *SMAL* [75, 279] shapes in canonical poses. For each sample, we obtain the *base features* and *surface-aware features* as described in Sec. 4.3 and Sec. 4.4

Embedding We project Diff3F features aggregated from $SMPL^{eval}$ to two dimensions using principal component analysis (PCA). In Fig. 4.3, we visualize the projection for two selected shapes from the same dataset. We repeat this with our *surface-aware features*. To avoid bias, we derive the visualized colors from the true SMPL [278] skinning weights $\mathbf{w}_n \in \mathbb{R}^B$ for both methods, where B is the skinning weight dimension. We repeat this process for $SMAL^{eval}$. In Fig. 4.3, our method yields an interpretable embedding that separates the leg and hand instances for animals and humans despite not having access to extrinsic (x, y, z) point positions. This validates the suitability of our features for downstream tasks and highlights the limitations of the Diff3F *base features*.

4.5.2 One-shot Pose Transfer

We evaluate the performance of the *surface-aware features* in a one-shot re-posing task for arbitrary 2-manifold meshes. We use our features to fit a Neural Jacobian Field (NJF) [119] between poses of two input shapes and then apply it to re-pose a new target shape as described in Appendix C.2.

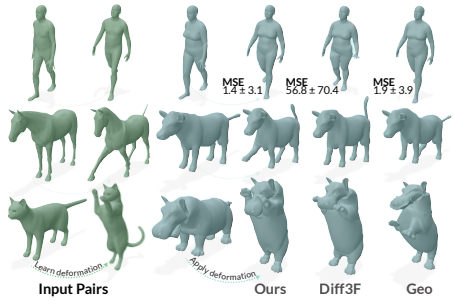


Figure 4.4: One-shot pose transfer using our features, Diff3F features, or Geometric descriptors. MSE $\times 10^{-4}$ is reported for human shapes.

We sample 5 input shape pairs from $SMPL^{eval}$ by choosing one challenging pose as a target and one random shape as an initial pose. The remaining shape samples serve as test inputs for pose transfer with known ground truth. We report MSE for 240 such test pairs (see upper row, Fig. 4.4). Next, we repeat the same procedure with the *base features* and with the Geometric descriptors (Geo) [119], consisting of the face centroid, face normals, and a Wave-Kernel Signature [206]. Finally, we provide additional qualitative results for transfer of animal poses from *TOSCA* to $SMAL^{eval}$ in Fig. 4.4.

We observe that our *surface-aware features* outperform the *base features* both quantitatively and qualitatively by correctly distinguishing individual posed limbs. The Geometric descriptors also perform well for humans, but they struggle with larger input and output shape differences in animals. Our features perform well in both cases.

4.5.3 Skinning Weight Regression

We train a simple regressor to predict skinning weights of a kinematic model, based on a single training sample.

Our pointwise regressor $\mathcal{W}_{\mathbf{s}}(\mathbf{s}_n)$ consists of a linear layer and a Softmax activation and regresses skinning weights $\bar{\mathbf{w}}_n$ from our *surface-aware features* or, in case of $\mathcal{W}_{\mathbf{f}}(\mathbf{f}_n)$ and $\mathcal{W}_{\mathbf{g}}(\mathbf{g}_n)$, from the *base features* or the Geometric descriptors, respectively. We train all models fivefold supervised with the Mean Squared Error (MSE) and true weights separately on the $SMPL^{eval}$ and dsmalours datasets, and we report test MSE for the remaining unseen samples in the source datasets (see Fig. 4.5). Our features achieve lower errors and exhibit better robustness to instance ambiguities than the two alternatives. It is worth noting that the relatively lower dimensionality of the Geometric descriptors and *base features* affects the number of regressor parameters, impact of which is not studied in this experiment. However, our conclusions also hold for a 2-layer MLP regressor with an equal hidden dimensionality of 106 which matches the dimensionality of the Geometric descriptors.

4.5.4 Point-to-Point Correspondence Matching

Our features can be easily integrated into correspondence matching pipelines. Therefore, we replicate the evaluation setup of Diff3F [147] and assess our *surface-aware features* in a



	$SMPL^{eval}$	$SMAL^{eval}$
Ours	**.0014±.0006**	**.0048±.0008**
Diff3F	.0058±.0024	.0086±.0028
Geo	.0210±.0031	.0287±.0017

Figure 4.5: Mean Squared Error of skinning weight regression (↓ is better) and its distribution across the SMPL mesh surface.

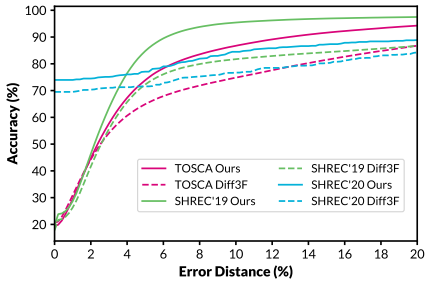


Figure 4.6: Correspondence accuracy (↑ is better) at different error distance thresholds for our own and Diff3F features.

correspondence matching task on human and animal shapes.

Data For testing we use re-meshed versions of humans from *SHREC'19* [264, 280] and animals from both *SHREC'20* [281] and the animal-only subset of *TOSCA* [282].

Baselines We compare our method against the unsupervised image-based Diff3F method [147], which also provides our *base features*, and against 3DCODED [249], DPC [250] and SE-OrNet [248], which have been trained on thousands of samples, while our method is trained on less than 100 samples.

Metrics We report commonly used point correspondence metrics for 1024 points per mesh [147, 248–250][†]. The correspondence error measures a distance between the computed correspondence point $\tau(\mathbf{p}_n)$ (see Sec. 4.3) and the ground-truth correspondence point $\mathbf{t}_n^{gt}$: $err = \frac{1}{n}\sum_{\mathbf{p}_n \in \mathcal{S}} \|\tau(\mathbf{p}_n) - \mathbf{t}_n^{gt}\|_2^2$. The accuracy is the fraction of points with an error below a threshold $\epsilon \in [0, 1]$: $acc(\epsilon) = \frac{1}{n}\sum_{\mathbf{p}_n \in \mathcal{S}} \mathbb{I}(\|\tau(\mathbf{p}_n) - t_n^{gt}\|_2 < \epsilon g)$, where g is the maximal Euclidean distance in the target shape and $\mathbb{I}(.)$ is the indicator function.

Results We provide quantitative results in Table 4.1 and qualitative comparisons in Fig. 4.7. We find that our method achieves the lowest error on *SHREC'19* and *TOSCA*, despite being trained on fewer samples than the supervised baselines. Furthermore,

[†] We use the provided code and validate that we follow the same experimental procedure and metric definitions.

Table 4.1: Comparison of our 3D correspondence matching to prior works 3DC (3D-CODED [249]), DPC [250], SEN (SE-OrNet [248]), and Diff3F [147]. †) Numbers originate from [147], *) Experiments were replicated, x) Omitted due to non-manifold meshes, + FM) Semantic features combined with Functional Maps. Accuracy is for the commonly used $\epsilon = 1\,\%$. The per-column best results are bold and the second-to-best results are underlined.

		SHREC'19	TOSCA	SHREC'20
3DC†	err ↓	8.10	19.20	-
	acc ↑	2.10	0.50	-
DPC†	err ↓	6.26	<u>3.74</u>	<u>2.13</u>
	acc ↑	17.40	30.79	31.08
SEN†	err ↓	4.56	4.32	**1.00**
	acc ↑	21.41	**33.25**	31.70
Diff3F*	err ↓	1.69±1.44	4.51±5.48	5.34±10.22
	acc ↑	<u>26.25±9.30</u>	<u>31.00±15.73</u>	<u>69.50±24.99</u>
Diff3F + FM*	err ↓	1.51±1.65	x	4.44±7.87
	acc ↑	21.71±7.12	x	58.03±25.94
Ours	err ↓	<u>0.43±0.76</u>	**1.65±2.15**	3.89±8.90
	acc ↑	**28.78±9.30**	29.35±14.53	**73.97±26.47**
Ours + FM	err ↓	**0.24±0.64**	x	3.54±7.59
	acc ↑	24.83±6.80	x	63.61±24.34

we outperform the Diff3F *base features* on both *SHREC* datasets in terms of accuracy. While Diff3F achieves a higher accuracy at 1% threshold in *TOSCA*, Fig. 4.6 shows that the accuracy of our model is higher for thresholds above $\approx 2\%$. This suggests that our method excels in the removal of outliers that can be caused by mismatched components. Additionally, adapting our features to Functional maps [260] (*+ FM*) distributes the error towards a lower mean at the cost of accuracy, and maintains the beneficial comparison to Diff3F+FM.

Fig. 4.7 shows that Diff3F struggles to separate intraclass instances such as left and right legs. In contrast, the results confirm the effectiveness of our contrastive loss in mitigating this issue. We observe the same behavior for *SHREC'20* in Fig. 4.8 (*top*), which contains highly diverse animal shapes. Furthermore, our method generally produces visually smoother results (see Appendix C.4.1 and the supplementary videos on the project website).

Other Shapes Our method is applicable beyond humanoid and animal shapes, which we show by training two additional encoders for a subset of 50 chairs and 50 airplanes from ShapeNet [283]. Here, we uniformly resample the mesh vertices for a better surface coverage [284].

For unseen shapes in Fig. 4.8, our *surface-aware features* again better distinguish same-class instances such as chair legs and airplane wings, supporting a wider applicability of our methodology. More examples are shown in Appendix C.4.2 and the supplementary videos.

4.5.5 Ablations

We motivate our design choices by ablation on various parts of our method in Table 4.2 following the setup of Sec. 4.5.4.

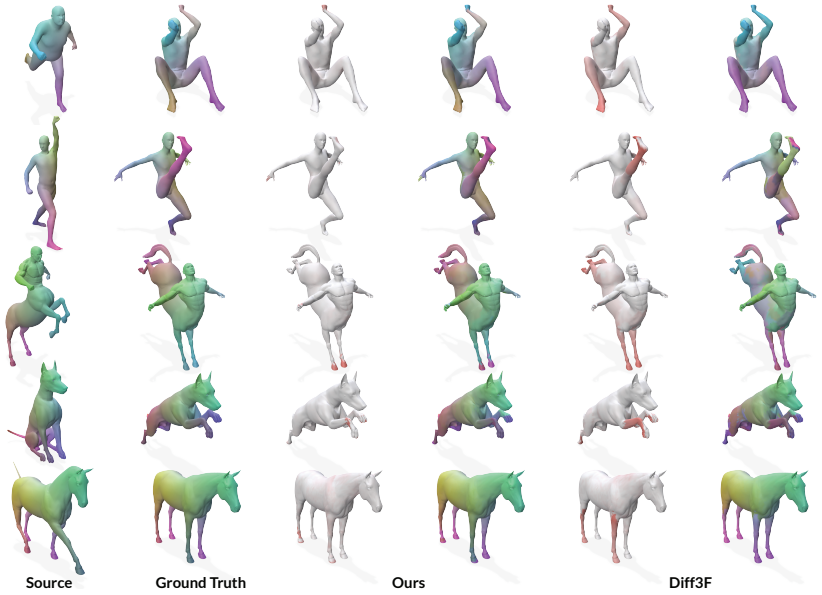


Figure 4.7: Qualitative comparison on the *SHREC'19* and *TOSCA* datasets with dense true correspondence labels provided by their authors. We show the source and target meshes with their ground truth correspondence labels (the two left-most columns) in comparison to correspondences computed using our *surface-aware features* (the forth column) and Diff3F *base features* (the right-most column). We further highlight the correspondence error on the mesh surface (the third and the fifth column). The error colormap is normalized per sample by the maximal error over both methods to keep the error scale comparable across columns but not across rows. Our *surface-aware features* notably improve separation of the limb instances.

Choice of Angular Space We demonstrate the effectiveness of our hyperspherical embedding by replacing our contrastive loss $\mathcal{L}_c$ (Eq. 4.4.2) with three different options inspired by related work (see Appendix C.1.2). First, the Relative Geodesic Loss (*RGL*) [221] optimizes relative distances in a Euclidean embedding. Similarly, the Naive Geodesic Loss (*NGL*) minimizes absolute distances. Finally, the Geometrical Similarity Loss (*GSL*) [217] enforces similarity of feature and surface distances in a local neighborhood. We remove feature and geodesic normalization wherever absolute magnitude needs to be learned. In Table 4.2 (top), we observe that, except for correspondence accuracy for TOSCA, our contrastive loss $\mathcal{L}_c$ outperforms all of the alternatives in the correspondence matching task.

Contrastive and Reconstruction Loss In Table 4.2 (bottom), we individually assess our two losses. We see that the performance with only the reconstruction loss $\mathcal{L}_r$ is close to Diff3F. This indicates that the gain in performance does not originate predominantly from a smaller embedding space or from access to training data. Similarly, the contrastive loss $\mathcal{L}_c$ alone results in an accuracy drop compared to our full model. This justifies our autoencoder approach with both losses playing an import role. Ablations on the number of anchors can be found inAppendix C.1.2.

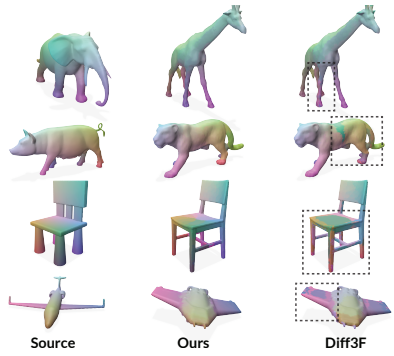


Figure 4.8: Qualitative comparison of correspondence matching on *TOSCA* and ShapeNet [283] (dense ground truth labels not available). Source shape (left) matched to target (right) using our and Diff3F features.

Table 4.2: Ablation on our method. *Above the bar:* Ablation on alternative losses inspired by related work [217, 221] compared to the unmodified Diff3F features. *Below the bar:* Our full method compared to its reduced variant omitting losses $\mathcal{L}_c$ or $\mathcal{L}_r$. The per-column best results are bold and the second-to-best results are underlined.

		SHREC'19	TOSCA	SHREC'20
RGL	err ↓	0.80±1.08	2.74±2.53	5.29±9.85
	acc ↑	20.16±10.03	16.53±10.41	55.23±21.93
NGL	err ↓	0.54±0.90	2.11±2.02	4.86±9.49
	acc ↑	18.84±9.59	18.86±11.50	58.97±23.12
GSL	err ↓	1.72±1.45	4.17±5.21	4.34±9.23
	acc ↑	26.89±9.07	29.77±14.92	73.39±26.31
Diff3F	err ↓	1.69±1.44	4.51±5.48	5.34±10.22
	acc ↑	26.25±9.30	**31.00±15.73**	69.50±24.99
only $\mathcal{L}_r$	err ↓	1.65±1.44	4.70±5.64	4.87±9.38
	acc ↑	26.53±9.19	30.27±15.17	72.94±26.21
only $\mathcal{L}_c$	err ↓	**0.38±0.61**	1.67±2.29	4.30±9.31
	acc ↑	26.21±8.78	25.58±13.88	70.08±25.17
Ours	err ↓	0.43±0.76	**1.65±2.15**	**3.89±8.90**
	acc ↑	**28.78±9.30**	29.35±14.53	**73.97±26.47**

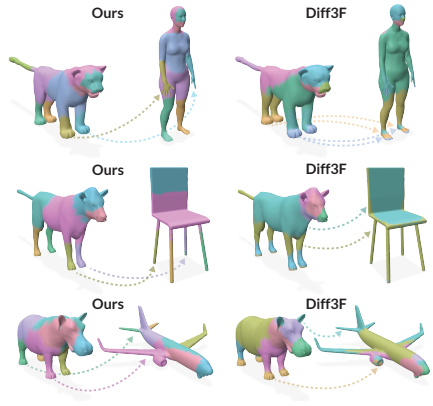


Figure 4.9: In the top row, 10 k-means cluster centers from the big cat were used to segment the human. In the bottom two rows, 8 k-means cluster centers from the animals were used to segment the chairs and airplanes with a shared encoder. Unlike Diff3F, our method successfully separates all limbs for a plausible mapping from animal limbs to human limbs, chair legs, or airplane wings.

4.6 Applications

We present additional downstream tasks that benefit from our *surface-aware features* learned in Sec. 4.5.

4.6.1 Instance-based Part Segmentation

Following the prior work [147], we segment a target shape by clustering features around centroids from K-means clustering of source-shape features. In the top row of Fig. 4.9, we demonstrate a transfer from a big cat to a human and see that, unlike the Diff3F features, our *surface-aware features* disambiguate the limbs. In the bottom two rows, we repeat this experiment with a shared encoder trained on human, animals, and a subset of *ShapeNet* (see Appendix C.4.4) where a true mapping cannot be defined but our method finds reasonable analogies between the classes.

In Fig. 4.10, we repeat this with centroids obtained jointly from all $SMPL^{eval}$ and *TOSCA* samples. In contrast to Diff3F, our method successfully matches features across diverse shapes, which demonstrates our embedding's capability of many-to-many shape matching without any additional pairwise optimization. Finally, we show similar results for chairs and airplanes in Fig. 4.11.



Figure 4.10: Results when clustering features across all samples in $SMPL^{eval}$ and *TOSCA*. Our method implicitly aligns semantically-related regions (shown as the same colors) across diverse 3D shapes in a self-supervised manner (the top row). Diff3F produces inconsistent labeling across different shape categories as well as lack of separability between instanced components such as individual limbs (the bottom row).



Figure 4.11: Results when clustering the *surface-aware features* across chairs (top row) and airplanes (bottom row) from ShapeNet [283]. Note, that while we use a single shared encoder for all humanoid and animal shapes, we train a separate encoder for each ShapeNet class due to the large domain gap.

4.6.2 Pose Alignment

Our *surface-aware features* are also useful for pose alignment of a kinematic model to another 3D shape. To this end, we establish point correspondences between shape pairs as in Sec. 4.5.4 and optimize the kinematic pose parameters to minimize point-to-point distances (see Appendix C.3).

In Fig. 4.12, we align $SMPL^{eval}$ to *SHREC'19*, and $SMAL^{eval}$ to DeformingThings4D [285] animals. Benefiting from the robust instance separation, our method produces poses closer to the targets for both dense and sparse correspondences. See our video for a 3D shape animation obtained by aligning to a target shape sequence.

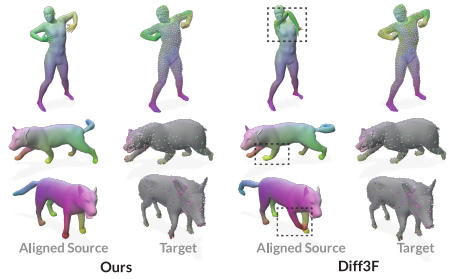


Figure 4.12: Pose alignment of a source shape (color) into the pose of a target (gray). The boxes highlight challenging areas handled well by our method. For humans, we densely fit all the target vertices, while for animals, we only fit 5 % of the vertices as highlighted.

4.6.3 Texturing

Since the *base features* are obtained from image models (see Sec. 4.3) and our pointwise encoder can process points sampled from a mesh as easily as pixels sampled from an image, we can establish correspondences between a 2D image and a 3D mesh. We demonstrate this by texturing 3D meshes from a masked target image and individually assign each vertex a color from the image pixel that maximizes the mutual feature similarity (Eq. 4.3.1) (see Fig. 4.13 and Appendix C.4.3). We observe that our features produce a more coherent mapping leading to a better preservation of the source appearance when compared to the Diff3F *base features*. In Fig. 4.14, we further show that textures can also be effectively transferred between two 3D shapes using a combination our *surface-aware features* with Functional Maps like in Sec. 4.5.4.

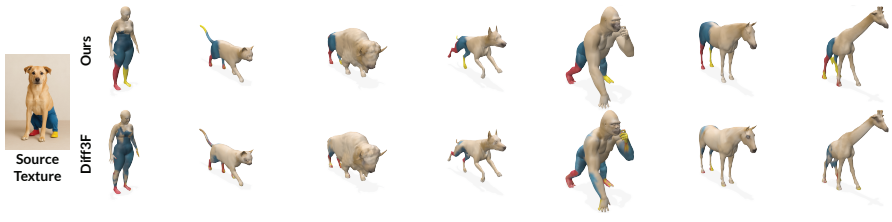


Figure 4.13: Texturing of 3D meshes from $SMPL^{eval}$ and *TOSCA*, based on a 2D image generated with ChatGPT. The appearance is transferred by establishing correspondence between the image features and 3D mesh features. In contrast to Diff3F, our *surface-aware features* represent the input image more faithfully.



Figure 4.14: Texturing of 3D meshes based on a source 3D mesh. The appearance is transferred based on correspondences established by combining our *surface-aware features* with Functional Maps. The data originate from $SMPL^{eval}$, SMPLitex [286], and DeformingThings4D [285].

4.7 Discussion

Limitations and Future Work Our method inherits limitations connected to the extraction of the *base features*. Specifically, the extraction of Diff3F features [147] takes several minutes per mesh and its vision model is sensitive to rendering artifacts or upside-down mesh orientations. We expect that advances in rendering of point representations increase the applicability across representations [159]. Furthermore, our method cannot establish a consistent partitioning for objects that are both geometrically and semantically isotropic (e.g., a round table). Hence, while our embedding separates human legs following the body's notion of front and rear, it cannot do so for table legs. However, this is not an issue for applications such as shape morphing [287]. Lastly, our method relies on consistency of geodesic distances between semantically distinct parts, and therefore it will be affected by geodesic shortcuts for partially blended parts in noisy 3D reconstructions (e.g., touching hands).

Beyond 3D alignment, our methodology could inspire 3D-to-2D pose estimation [105, 288], articulated 3D reconstruction [1, 20], automated rigging [289] or 2D-to-3D uplifting [133, 188], where our features could support more view-consistent representations. Finally, an interesting topic for future research is the development of foundational features using massive datasets, such as Objaverse [290].

Conclusion We have introduced novel *surface-aware features* for 3D shape matching that disambiguate intra-class instances among semantic features derived from pre-trained 2D vision models. Our descriptors have proven effective in distinguishing instances of the same semantic class and they generalize even when trained on a limited number of 3D shapes. Furthermore, our contrastive loss facilitates easy integration in future unsupervised methods which reduces data labeling effort. Consequently, our method is a promising building block toward adapting pre-trained 2D models to 3D tasks.

Rotation-Equivariant Neural Jacobian Fields for Generalizable 3D Mesh Deformations

5

Neural Jacobian Fields are state-of-the-art neural representations for deformations on 3D meshes. A fundamental limitation of the original formulation of NJF and follow-up works is that the learned Jacobians are not rotation-equivariant. This limits generalization to rotated shapes as well as shapes with greatly diverging geometry. A simple solution could be to use rotation-invariant input features; however, our analysis shows that the rotation-dependence of NJF lies in the representation of the Jacobians themselves, not just the input features. We propose a rotation-equivariant formulation of Neural Jacobian Fields by representing Jacobians in local tangent frames of triangle faces. The tangent frames are optimized end-to-end in the gradient space of semantic surface features, distilled from 2D image foundation models. Such semantic features tend to be robust to rotations and large changes in geometry, and therefore the learned tangent frames and Jacobians generalize to transformations in SO(3) and even to substantial changes in geometry (e.g., dog to human). We evaluate our method on human and animal pose deformation datasets, where we achieve competitive results while being equivariant to rotation and robust to input shape. Our method removes the requirement to align 3D meshes during training and inference and enables further applications of task-specific, semantics-based tangent frames. *

* This chapter is based on the preprint "Rotation-Equivariant Neural Jacobian Fields for Generalizable 3D Mesh Deformations." (2026) by Lukas Uzolas, Ruben Wiersma, Olga Sorkine-Hornung, and Petr Kellnhofer.

5.1 Introduction

Shape deformation is crucial for turning static meshes into animated, moving characters. Since per-vertex adjustments require a lot of manual work, deformations are typically applied through a lower-dimensional parametrization. Handle points, for example, let users control complex shape deformations with only a few manual interventions [13, 291, 292]. These approaches typically require manually crafted deformation structures, such as rigs or cages, which are non-trivial to create. Moreover, producing realistic deformations through manual editing requires considerable domain expertise, posing a substantial barrier to producing realistic 3D deformations. Data-driven methods, on the other hand, could learn such a mapping between user-facing control and the shape deformation directly from examples. For instance, one could learn the mapping from a generic skeleton to humanoid- and animal meshes. The training dataset would consist of meshes deformed by explicit skeletons, so that the connection between the skeleton and deformed mesh is given. At inference time, however, the model can be applied to new meshes without setting up skinning weights or even a rough placement of an explicit skeleton. This can significantly reduce the barrier to animating 3D shapes.

Neural Jacobian Fields (NJFs) [119] are a popular building block in learning-based methods for deformation [293–295]. NJFs represent deformations in the gradient domain using a neural network and achieve state-of-the-art performance for applying deformations to various shapes. A core strength is that NJFs are agnostic to triangulation. In our work, we seek to strengthen this generalizability to significant changes in *geometry*, so that one can learn deformations on one set of shapes and generalize to unseen shapes, even of different classes. This requires two core properties: equivariance to transformations of the input (rotation, translation, scale) and input features that are robust to changes in geometry. Jacobians are already equivariant to translation and scaling; a translation or scaling of the input will result in a corresponding transformation of the output.

An important limitation is that NJFs are not *rotation*-equivariant, even when using rotation-invariant input features or learning input features with a rotation-invariant network, such as PoissonNet [295]. This means that the input shape must be registered to a canonical coordinate system and cannot diverge too much from the training shape. We find that data augmentation with random rotations at training time does not solve this problem (Fig. 5.7). To illustrate the issue, we overfit an NJF on a single deformation for a single mesh using per-vertex random noise as the only input feature (Fig. 5.1). For the rotation used in training (0°, top row), NJF matches the target shape. In contrast, when the input is rotated (90°, bottom row), the output fails to match the target. In general, any non-zero rotation results in error (vertex error plot, right). Note that the input feature has no influence on this result: the per-vertex noise is constant for each rotation. The lacking rotation-equivariance in NJF is caused by how Jacobians are represented. Jacobians map tangent vectors (2D, intrinsic) to the deformed vector in ambient space (3D, extrinsic). If the shape is rotated in ambient space, the deformed vectors should rotate with it. NJF does not account for this, resulting in the error observed in Fig. 5.1.

Our approach, Rotation-equivariant NJF (Re-NJF) solves this: Re-NJF decouples the predicted deformation from ambient space by expressing Jacobians in local tangent frames defined per face. This ensures that the deformations change according to rotations of the input shape. The use of local frames means that our method is robust to localized transformations, e.g., rotating an arm around the shoulder joint. We propose to learn

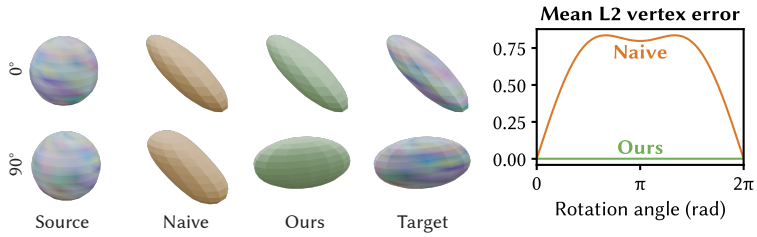


Figure 5.1: Comparison of deformation error between our and the naive method. We train a Neural Jacobian Field to deform the source mesh to the target mesh at an angle of 0°, and then apply the learned deformation at the 90° rotated source mesh. The naive method produces high errors when the mesh is rotated.

tangent frames for deformation from the gradients of an input feature, inspired by approaches for intrinsic geometric deep learning [295–297]. The resulting tangent frames are intrinsic, smooth and consistent across diverse geometries. Re-NJF can be used as a drop-in replacement for NJF pipelines using general rotation-invariant features, such as WKS or HKS [204, 298].

A second step toward geometry-agnostic NJFs is the use of semantic input features. Instead of using geometric descriptors as the input feature, we leverage 2D image features from foundation models, inspired by recent work [147]. Because these features encode semantic information, they are robust to variations in shape geometry and orientation. When used as a conditioning signal for NJF, they significantly improve deformation transfer between strongly differing shapes [203], addressing both the limitation of poor cross-shape generalization and forming a robust signal to learn tangent frames.

To summarize, we make the following contributions:

- We define a rotation-equivariant formulation of Neural Jacobian Fields.
- We propose an intrinsic approach for learning tangent frames that can be trained end-to-end.
- We implement a full pipeline for rotation-equivariant NJFs using semantic features and use PoissonNet to process them for robustness.

We empirically demonstrate strong generalization under arbitrary rotations and across diverse shape categories, including humanoid and animal meshes.

5.2 Related works

We build upon learning mesh deformations on surfaces conditioned on semantic features obtained from 2D foundation models. In this section, we discuss related works on surface deformations, semantic features and rotation-equivariant processing.

5.2.1 3D mesh deformations

Traditional approaches for deformation enable free-form 3D mesh deformation through user-defined manipulation handles, including cages [291], keypoints [292], bone rotations within skeletal animation pipelines [13], blend shape interpolation weights [161, 162], and

physics-based simulation [299]. However, these methods typically require significant manual effort, such as character rigging, and can produce unnatural deformations.

Data-driven methods aim to reduce this burden by learning deformations from examples. One line of work focuses on learned template models that decouple pose and shape from training data for specific shape categories, such as human bodies [16], hands [74], and animals [75]. Another line of work addresses deformation transfer, where a source mesh is deformed to match a target, either through skeletal deformations [193, 300] or free-form mesh deformations [119, 301, 302]. Among the latter, Neural Jacobian Fields (NJFs) [119] achieve state-of-the-art, detail-preserving results by modeling deformations in the gradient domain through learned ambient-space Jacobians. NJF has since been adopted for various pose and motion transfer tasks, facilitating the transfer of deformation between broad shape categories [293, 294, 303]. However, NJF is not rotation-equivariant, which can hinder its generalization, a limitation that we address in this chapter.

5.2.2 Semantic features

Image foundation models trained on large-scale image datasets exhibit strong internal semantic representations. Examples include the DINO family [210, 211, 304], a series of self-supervised vision transformers, and Stable Diffusion [132], a diffusion model trained for image generation. These models produce semantically meaningful dense features that have proven effective for a range of 2D tasks, including image segmentation, classification, and correspondence estimation [209, 211, 212, 304].

The 3D domain currently lacks equivalent foundation models that produce domain-native semantic features. Therefore, several works have explored lifting 2D features to 3D. This can be achieved by rendering images of the 3D representations, extracting 2D image features, and re-projecting them onto the 3D surface [147, 208]. These aggregated features have been utilized for keypoint correspondences [147, 203, 208, 214, 305], dense surface correspondences [147, 216], 3D appearance transfer [215], embedding language into 3D representations [306], and 3D deformation transfer [203]. The latter example demonstrates effectiveness of surface-aware features for conditioning pose transfer based on NJF. However, the prior work required alignment of training and inference shapes, which is a limitation addressed by our contribution.

5.2.3 Rotation-equivariant learning on surfaces

The past decade has seen many efforts to generalize deep learning from images and text to 3D surfaces and graphs, termed geometric deep learning [307, 308]. Rotation-equivariance is a core concern here, because the geometry of a shape is often represented in an extrinsic coordinate system, which may be rotated arbitrarily, and core building blocks for neural networks, such as convolution, require a local coordinate system that can be rotated around the normal arbitrarily. Because we use semantic features as input, we do not target rotation-equivariance to input coordinates. Instead, our focus is on a localized, rotation-equivariant Jacobian operator to encode deformation. Still, we share several characteristics with related work.

PointNet [309], a pioneering work in geometric deep learning, addresses rotations by aligning the input using a transform-network, inspired by spatial transformer networks

for images [310]. We also learn transformation matrices; however, our transformations are localized (per-triangle) and are aligned to feature *gradients*. That means our solution is robust to non-rigid deformations of the input shape. In our ablations, we compare to a global, rigid alignment. Other approaches adjust the convolution kernels in CNNs to ensure global rotation-equivariance [311, 312] and tangent-space rotation-equivariance [313–316]. We point readers to the survey by Bronstein et al. [308] for more details. The latter approaches typically apply convolution kernels through a mapping from the tangent plane to the surface. An alternative is to construct coordinate-free convolution-like blocks from geometric operators, such as the (graph) Laplacian and gradient operator [295–297, 317]. We take inspiration from these approaches, by aligning local coordinate systems to feature gradients. In addition, we process semantic features with PoissonNet [295], which is robust to arbitrary choices of global- and local coordinate systems, given rotation-invariant input features.

5.3 Preliminaries and analysis

3D Representation. We assume as input a triangular surface mesh $\mathscr{M} := \{\mathbf{V}, \mathbf{F}\}$, where $\mathbf{V} \in \mathbb{R}^{V\times 3}$ represents the vertex locations and $\mathbf{F} \in \mathbb{N}^{F\times 3}$ provides the vertex indices for F triangles. Each face is associated with two tangent vectors $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}^3$ orthonormal to each other and to the face normal $\mathbf{n}$. Together, these vectors form a local basis $\mathbf{B}_i := [\mathbf{x}_i, \mathbf{y}_i, \mathbf{n}_i]$. The rotation of the tangent vectors around the normal is ambiguous; we discuss how these are set in the method section.

The discrete intrinsic gradient operator $\mathbf{G} \in \mathbb{R}^{2F\times V}$ is computed from $\mathscr{M}$ and outputs the gradient vector in the per-face tangent space spanned by $[\mathbf{x}_i, \mathbf{y}_i]$. We also use an extrinsic gradient operator $\mathbf{G}_{\text{ext}} \in \mathbb{R}^{3F\times V}$, which can be constructed from $\mathbf{G}$ by representing the tangent-space gradient in ambient space coordinates.

Mesh Deformation and Jacobians. The Jacobian captures the deformation of a surface as a map from the 2-dimensional tangent space to the 3-dimensional ambient space. For a triangle mesh, the Jacobian is constant on each triangle. Given the vertex positions of a deformed mesh, we can determine the Jacobian $\mathbf{J}_i \in \mathbb{R}^{3\times 2}$ for triangle i from the transformation of the triangle edges

$$\mathbf{J}_i\,[\mathbf{x}_i, \mathbf{y}_i]^T[\mathbf{v}_k - \mathbf{v}_j,\ \mathbf{v}_l - \mathbf{v}_j] = [\mathbf{v}'_k - \mathbf{v}'_j,\ \mathbf{v}'_l - \mathbf{v}'_j], \tag{5.1}$$

$$\mathbf{J}_i\ = [\mathbf{v}'_k - \mathbf{v}'_j,\ \mathbf{v}'_l - \mathbf{v}'_j]\left([\mathbf{x}_i, \mathbf{y}_i]^T[\mathbf{v}_k - \mathbf{v}_j,\ \mathbf{v}_l - \mathbf{v}_j]\right)^{-1} \tag{5.2}$$

where j, k, l are the vertex indices of triangle i, and $\mathbf{v}_i$ are vertex coordinates in column format.

In the continuous setting, one can find the surface M' whose Jacobian is closest to a target Jacobian J' in a least-squares sense by solving the Poisson equation

$$\Delta M' = \text{div} J'. \tag{5.3}$$

The discrete equivalent of this equation for a triangle mesh is constructed from the cotangent Laplacian $\mathbf{L} \in \mathbb{R}^{V\times V}$, the discrete intrinsic gradient operator $\mathbf{G} \in \mathbb{R}^{2F\times V}$, the

triangle area matrix $\mathbf{A} \in \mathbb{R}^{2F \times 2F}$, and a Jacobian matrix $\mathbf{J}' \in \mathbb{R}^{3 \times 2F}$ that stacks the target Jacobian $\mathbf{J}'_i$ of all faces $i = \{1 \dots F\}$ horizontally

$$\mathbf{L}\mathbf{V}' = \mathbf{G}^\top \mathbf{A}\, \mathbf{J}'^\top. \tag{5.4}$$

For more details on linear deformation models, we refer the reader to the survey by Botsch and Sorkine [318].

The core insight of NJF is to encode the Jacobian over the surface as a smooth, continuous field represented as a neural network. The neural field d_θ takes a feature per face $\mathbf{c}_i$ as input and outputs a 3×2 target Jacobian

$$\mathbf{J}'_i = d_\theta(\mathbf{c}_i). \tag{5.5}$$

NJF [119] chooses a combination of input features per task, for example, concatenating the barycenters and normals of each face. By moving away from a per-face Jacobian to a continuous field, the deformation can generalize to different triangulations and, depending on the input feature $\mathbf{c}$, to other shapes. The deformation can be controlled at inference time by further concatenating a global shape encoding, such as bone-rotation angles or a text prompt to $\mathbf{c}$.

Instead of regressing the Jacobians directly, follow-up works [293, 295] use a *relative* formulation, where only the difference to the gradient of the input shape is learned:

$$\mathbf{J}'_i = \mathbf{P}_i(\mathbf{G}\mathbf{V}) + d_\theta(\mathbf{c}_i), \tag{5.6}$$

where $\mathbf{P}_i \in \mathbb{R}^{3 \times 2F}$ is the row selection matrix, selecting the pairs of rows belonging to the i'th face. A benefit of this formulation is that, when $d_\theta(\mathbf{c}_i) = \mathbf{0}$, the original vertex positions $\mathbf{V}$ are recovered when solving the Poisson equation (5.4).

Problem analysis The issue that arises in both the direct and relative setting is that the Jacobian maps to coordinates in ambient space. This means that the mapping $d_\theta(\mathbf{c}_i)$ is inherently tied to the configuration of the input- and output shape in ambient space, regardless of the input features $\mathbf{c}_i$. The central question of the current work is: how should we design an NJF to be equivariant (i.e., a transformation to the input of the NJF results in an equivalent transformation of the output) to global and per-part rotations of the shape?

Input features NJF [119] sets the input features $\mathbf{c}_i$ to a concatenation of extrinsic positions and normals or intrinsic wave kernel signatures [298]. Maesumi et al. learn the input features to an NJF using PoissonNet [295], where the input to PoissonNet can be a combination of point positions and normals or wave kernel signatures. The entire pipeline is trained end-to-end on a dataset of source and target shapes, so that the PoissonNet descriptors can be more robust to shape variations. However, the approach can still be sensitive to linear transformations of the input shape if positions and normals are used as input. Note that, even though PoissonNet can produce rotation-invariant features, such features do not solve the rotation-equivariance problem.

5.4 Method

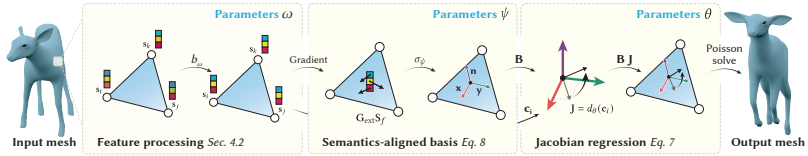


Figure 5.2: Overview of our method. As input, we receive a mesh where each vertex is associated with a semantic feature **s**, which are passed to b_ω for preprocessing. We first construct the tangent frame **B** by aligning the vector **x** with the learned feature gradient. We then learn how to deform each face, by regressing a Jacobian conditioned on $\mathbf{c}_i$. The Jacobians are then rotated to ambient space and used for obtaining the final vertex positions via the Poisson solve.

We first propose a rotation-equivariant formulation of Neural Jacobian Fields and its parameterization. Secondly, we discuss the semantic features used to condition deformation and tangent frame regression. Thirdly, we explain the deformation conditioning scheme. Lastly, we describe the training procedure.

5.4.1 Rotation-Equivariant Neural Jacobians

To learn rotation-equivariant Neural Jacobian Fields, we decouple the Jacobian from ambient space by representing Jacobians, d_θ, in a local tangent frame $\mathbf{B}_i \in \mathbb{R}^{3\times3}$. The ambient-space per-face Jacobian is then given by

$$\mathbf{J}'_i = \mathbf{B}_i d_\theta(\mathbf{c}_i), \tag{5.7}$$

where $\mathbf{B}_i := [\mathbf{x}_i, \mathbf{y}_i, \mathbf{n}_i]$ [†]. The tangent vectors $\mathbf{x}_i, \mathbf{y}_i$ are determined up to a rotation around the normal vector. Typically, $\mathbf{x}_i$ is set to an arbitrary direction, such as the first edge of the corresponding triangle. This would tie the tangent frame to the exact triangulation, hindering generalizability. Another option would be to align $\mathbf{x}_i$ and $\mathbf{y}_i$ to the (smoothed) principal curvature directions of the surface. While this may work for similar source- and target shapes, this would prohibit generalizability to target shapes with different geometry as demonstrated in our experiments.

Inspired by work on coordinate-free geometric deep learning [296, 297], we align the tangent basis to a learned function σ_ϕ of the gradient of semantic features $\mathbf{S} \in \mathbb{R}^{V\times S}$, which outputs a single 3-vector per face

$$\mathbf{x}_i = [\sigma_\phi(\mathbf{G}_{\text{ext}}\mathbf{S})]_i, \tag{5.8}$$

$$\mathbf{y}_i = \mathbf{n}_i \times \mathbf{x}_i. \tag{5.9}$$

We will detail the input features in the following section. The simplest choice for σ_ϕ with gradient vectors $\mathbf{G}_{\text{ext}}\mathbf{S}$ is their linear combination expressed as

$$\sigma^{\text{lin}}_\phi(\mathbf{G}_{\text{ext}}\mathbf{S}) = \mathbf{G}_{\text{ext}}\mathbf{S}\phi, \tag{5.10}$$

where ϕ is an S-dimensional vector weighing the most informative semantic features, similarly to the first principal component vector in Principal Component Analysis (PCA).

[†] Note that the same basis is assumed for the intrinsic gradient operator in (5.4).

In our experiments, we also test non-linear combinations of the gradient vectors (using vector non-linearities [295, 296]) and semantic features and find that these make little difference. This is likely because **S** is already processed using PoissonNet (see the following section).

Why local bases? An alternative to a per-face basis is to find a global coordinate basis for the entire shape and align the mesh to this basis before applying NJF; however, this would limit the applicability of NJF to global rigid rotations. In practice, deformations might be applied locally, such as rotating an arm or leg. Learning a local basis makes our formulation robust to even such local deformations and rotations (For an example, see Fig. 5.5).

5.4.2 Robust semantic features

As input to both (5.8) and the NJF, we use the semantic features **S** computed using Surface-Aware Semantic Features [203] which are robust to geometric variation across shape classes and largely rotation-invariant. This allows for a mapping between similar categories (e.g., horse ↔ dog) as well as distant categories (e.g., horse ↔ human). A per-face feature is computed as the average of its vertex features:

$$\mathbf{c}_i = \frac{1}{3} \sum_{j \in \mathbf{f}_i} \mathbf{s}_j. \tag{5.11}$$

Using the semantic feature distilled from image-based feature extractors 'as-is' could result in issues. Due to projective occlusions, some surface regions may not receive any features. In addition, a high-frequency variation of extracted features makes the tangent bases sensitive to noise. A known solution is to compute the bases on a smoothed input function or compute a smoothed vector field [319, 320]. To preclude a manually-set smoothing factor and to allow additional non-linear filtering, we pass the semantic features through a PoissonNet, b_ω with trained parameters ω, before computing (5.8) and (5.6), which also handles the required smoothing.

5.4.3 Learning relative pose deformations

A possible application includes training NJF on a dataset of mesh deformations, each paired with $\mathbf{p}_i$ as a their global pose encoding. At inference, the model can be used to deform unseen meshes for a new pose $\mathbf{p}$, without requiring an explicit skeleton or parametrization. Typically, the pose $\mathbf{p}$ is encoded relative to some 'canonical' pose, such as a T-pose. This limits the trained model to inputs in the same canonical pose. Alternatively, one can learn to encode the pose explicitly [119] or implicitly [295], and let a network learn how to deform *any* input mesh to an *absolute* target pose. However, training a generalizable pose encoder is challenging on its own, and typically requires extrinsic information to encode how body parts are arranged in Euclidean space. Given our localized rotation-equivariant formulation, it is now possible to apply our method to any starting configuration without training a pose encoder. To demonstrate the benefit of this new capability, we condition deformations on *relative* target poses. This makes it possible, for example, to rotate the arm on a mesh in any input pose by a given rotation.

In our experiments, we derive relative pose vectors, $\mathbf{p}$, from the kinematic tree of two template models, namely SMPL [16] and SMAL [75]. As these models differ in their number of bones and connectivity, we denote them as $\mathbf{p}^h$ and $\mathbf{p}^a$ for *human* and *animal*, respectively. Where each vector encodes the relative rotation of each bone, omitting the global rotation component. The relative pose vectors are jointly embedded with the semantic features via an end-to-end learned function. As $\mathbf{p}^h$ and $\mathbf{p}^a$ differ in size, we parameterize them as two separate, small MLPs:

$$\mathbf{s}_j = p^h_\gamma(\mathbf{p}^h, \mathbf{s}_j), \text{ or} \tag{5.12}$$

$$\mathbf{s}_j = p^a_\lambda(\mathbf{p}^a, \mathbf{s}_j). \tag{5.13}$$

These feature vectors are then averaged for each face (5.7) and used as input to the deformation network (5.11). After training, this formulation allows choosing either pose representation to drive the deformation of humanoid or animal meshes alike, as the semantic features implicitly align corresponding parts across categories. Moreover, adopting relative deformations eliminates the need for a canonical pose shared between the two templates and opens a path towards more categories in the future.

5.4.4 Training

We train d_θ, σ_ϕ, b_ω, p^a_λ, and p^h_γ end-to-end by learning how to deform random source mesh $\mathcal{M}_s$ such that it aligns with target mesh $\mathcal{M}_t$, given the ground-truth relative pose embedding. Remarkably, we find it sufficient to indirectly supervise σ_ϕ with commonly used task-specific deformation losses to produce a smooth, generalizing basis across shapes.

To this end, we predict Jacobians $\mathbf{J}'_i$ (see (5.7)), solve for the vertex positions (see (5.4)), and iteratively update the network parameters via gradient descent by minimizing the following two losses proposed in NJF [119]. First,

$$\mathcal{L}_v = \sum_i |m^v_i| \|\mathbf{v}^{gt}_i - \mathbf{v}'_i\|^2, \tag{5.14}$$

where $\mathbf{v}^{gt}_i$ is the ground-truth vertex, $\mathbf{v}'_i$ is the predicted vertex, and m^v_i is the vertex mass of vertex i. Second,

$$\mathcal{L}_j = \sum_i |m^f_i| \|\mathbf{J}^{gt}_i - \mathbf{J}'_i\|^2, \tag{5.15}$$

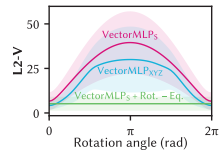
where m^f_i is the face area, and $\mathbf{J}^{gt}_i$ is the ground-truth Jacobian. The final training loss is given by $\mathcal{L} = \mathcal{L}_v + \mathcal{L}_j$.

5.5 Experiments

In our experiments, we first validate the core rotation-equivariance property of our approach and compare this to prior work. Then, we demonstrate its effectiveness in a generalized setup, training on a dataset with strong shape variety and ablate our design decisions. Lastly, we show how pre-trained generalized deformation model enables better performance in a downstream optimization task.

	With Rotation			No Rotation		
	L2-V	L2-J	L2-N	L2-V	L2-J	L2-N
MLP_S + Rot.-Eq.	5.36±3.94	4.62±1.89	14.59±6.72	5.36±3.94	4.62±1.89	14.59±6.72
MLP_S	22.53±14.07	9.38±3.73	32.26±15.45	5.74±3.60	4.84±1.91	13.36±6.03
MLP_{XYZ}	19.65±15.04	8.36±3.98	30.45±17.75	4.43±2.39	4.03±1.65	11.28±5.17
$VectorMLP_S$ + Rot.-Eq.	**5.18±3.92**	**4.25±1.77**	**13.48±6.23**	5.18±3.92	4.25±1.77	13.48±6.23
$VectorMLP_S$	26.17±16.86	9.86±4.66	35.01±17.04	6.60±4.94	4.46±1.96	14.97±7.51
$VectorMLP_{XYZ}$	20.02±15.41	7.87±4.09	31.49±18.63	**4.27±2.39**	**2.98±1.29**	**9.87±4.33**

(a) Quantitative results on the test set. 'With Rotation' reports the results when the metrics are averaged across 60 rotational steps around the Y-axis, while 'No Rotation' only reports the results at training angle 0°. Our extension allows for consistent results across all angles.



(b) L2-V error across different evaluation angles. The baseline's error increases with increasing angles, while ours remains constant.

Figure 5.3: Quantitative results on the test case, showcasing the rotation-equivariance of our method. Comparing our method as a drop-in replacement within deformation head of related works: the MLP head from NJF on xyz coordinates [119], the VectorMLP from PoissonNet on xyz coordinates [295] and both heads on semantic coordinates [203].

5.5.1 Rotation-equivariance

We first validate the rotation-equivariance property of our method in an isolated setting, where input features and changes in geometry do not affect our method. To this end, we overfit an NJF on a single source-target ellipsoidal mesh pair $\mathcal{M}_s - \mathcal{M}_t$ and apply it to the same source mesh augmented by ambient-space rotations. We expect NJF to perform well for the training source mesh but to fail for rotated meshes. In contrast, our method should yield constant performance across all rotations.

In this simplified setting, we parameterize σ_ϕ as a linear transformation, and d_θ as an MLP. To exclude their impact, we sample a fixed set of feature vectors from a random Gaussian distribution $\mathbf{s}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This is possible here, as we do not generalize between shapes. We follow Sec. 5.4.4, to train our Re-NJF, but omit b_ω and the pose conditioning, as it is not necessary in this setup. We also train a *naive* version corresponding to the original NJF formulation without a pose encoder.

At test time, we evaluate both models on $\mathcal{M}_s$ rotated around the Y-axis in constant increments. Fig. 5.1 confirms our expectations: *our* method produces the same output at all angles, resulting in a constant error. This is in contrast to the *naive* method, which, even in this simplified setting, faithfully reproduces the target only at the training angle of 0°, thus fails to generalize to ambient space transformations.

5.5.2 Generalized setting

In this section, we demonstrate the effectiveness of Re-NJF on an end-to-end, generalized task: learning relative pose deformations for arbitrary humanoid and animal meshes. We aim to show that, by leveraging semantic features, we can learn generalizable deformations between the two shape categories. Furthermore, because our method is rotation-equivariant, we do not need to align or canonicalize the meshes during training or inference.

Data We construct a mesh dataset consisting of humanoid and animal shapes. For humans, we randomly sample body shapes from the STAR body model [321] and use poses from the AMASS dataset [179]. For animals, we randomly sample body

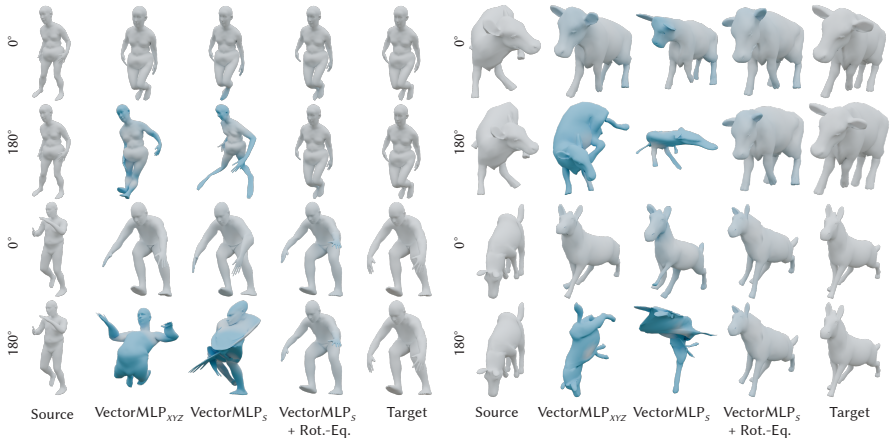


Figure 5.4: Qualitative comparison of our method in comparison with the baselines. We aim to deform the *source* mesh to the *target* mesh, where errors are highlighted in blue. While all methods produce reasonable results at a training angle of 0°, only our method produces consistent results when the mesh is rotated 180° before the deformation is applied. Note that we rotate the mesh back after deformation for visualization purposes.

shapes from the SMAL animal model [75, 322] and sample poses from Animal3D [323]. Representative examples can be seen in Fig. 5.4. As the animal pose dataset is several orders of magnitude smaller than the human pose dataset, we devise a different sampling strategy for animal poses. We fit a Gaussian distribution to the bone angles in the dataset and then randomly sample angles from it instead. Each dataset entry consists of a pair of source and target meshes, corresponding to the same *shape* parameters, β, but differing in their *pose* parameters, $\mathbf{p}$ (see Sec. 5.4.3). A mesh is then obtained via $\mathcal{M}^a = \text{SMAL}(\mathbf{p}^a, \beta^a)$, and likewise for the humans $\mathcal{M}^h = \text{STAR}(\mathbf{p}^h, \beta^h)$, where the superscript denotes either animal or human.

Lastly, we extract semantic features for 100 human- and 100 animal training meshes, forming two feature sets $\mathcal{S}^a = \{\mathbf{S}_i^a\}_{i=1}^{100}$ and $\mathcal{S}^h = \{\mathbf{S}_i^h\}_{i=1}^{100}$. Each element in either feature set has the same size because the mesh topology is the same across shape categories. In addition, we construct separate validation and test sets for the features and poses used for quantitative and qualitative evaluation. For qualitative evaluation, we further test on unseen animal shapes from DT4D [285] and TOSCA [282], as well as human shapes from CAPE [324, 325], which contain high-frequency deformations on the surface due to clothing and other humanoid characters [295].

Baselines and Implementation We benchmark our method by using it as a drop-in replacement of the deformation heads used for mesh deformation in PoissonNet [295], NJFs [119], and SAF3D [203]. In the case of PoissonNet, d_θ is parameterized as a modulated *Vector MLP*. While NJF's and SAF3D's parameterization of d_θ consists of a regular *MLP* with fully connected layers. Note that PoissonNet regresses restricted Jacobians $\mathbf{J}_i' \in \mathbb{R}^{2\times3}$, while NJF uses unrestricted Jacobians in their implementation, i.e., $\mathbf{J}_i' \in \mathbb{R}^{3\times3}$. When extending PoissonNet to use our proposed method and when using NJF, we solve the linear system (see (5.4)) with unrestricted Jacobians. More details are given in the Appendix D.1.

We further evaluate the performance difference between the deformation heads when

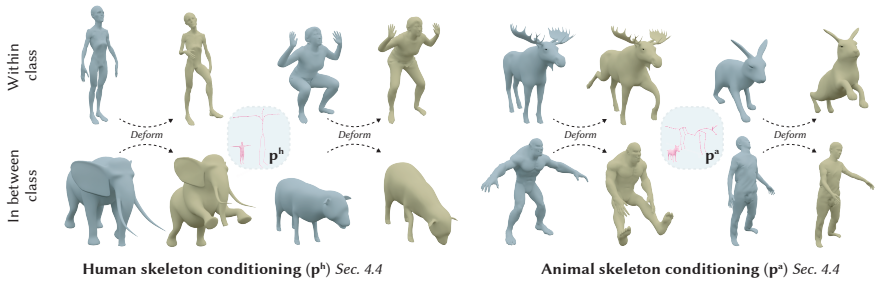


Figure 5.5: Results obtained when conditioning the deformation network on manually defined bone rotations, applied to unseen input meshes. We can choose the human kinematic model ($\mathbf{p}^h$) or the animal one ($\mathbf{p}^a$) as *conditioning* signal (shown in the middle of each quadruplet), irrespective of the input shape. The top row shows examples where the conditioning and input mesh are within the same class, i.e. $\mathbf{p}^h$ is used to deform humanoid characters, while the bottom rows shows examples across classes. Our method can realistically deform animal meshes, even with the human conditioning signal, and vice versa.

using the vertex positions (XYZ) as input, compared to the semantic features denoted as S. Parameterizations of σ_ϕ are evaluated and discussed in the ablations.

We preprocess the semantic features with b_ω, which is parameterized as three Poisson blocks with a hidden dimension size of 128, and each deformation head contains 5 layers of size 128. We train all networks end-to-end with a batch size of 16, where each batch consists entirely of either human or animal meshes, making use of the AdamW optimizer [274]. Due to the pose imbalance, we further sample animal meshes with only a 30% probability. We find that the models stabilize with respect to their validation metrics after around 120.000 iterations, and therefore choose the model for final evaluation at 150.000 training iterations.

Metrics We report three metrics [119] that measure the difference between predictions and ground truth. First, **L2-V**, which measures the average L2 norm of the vertices scaled by 10^2. Second, **L2-J** measures the average L2 norm of the Jacobians scaled by 10. Lastly, **L2-N** is the average L2 norm of face angles between prediction and target in degrees. Metrics are reported on 2000 test pairs, sampled from the test set, which consists of unseen poses and semantic features.

Test results. Quantitative results are provided in Fig. 5.3b and Table 5.3a. In contrast to the baselines, our method is the only one that produces consistent results across global rotations around the Y-axis. The baselines produce high errors if the mesh orientation deviates from the training orientation at inference time. Therefore, our method produces the best results when averaging across all angles (see "With Rotation" in Table 5.3a). Qualitative results further confirm these findings in Fig. 5.4. The baselines produce unnatural deformations, once the input mesh is rotated around the y-axis.

We find that the Vector MLP deformation head generally outperforms the MLP, and that it achieves the best results across all angles when combined with our method. We therefore use the VectorMLP$_S$ + Rot.-Eq. model for all consecutive experiments. We observe a slight degradation in performance when conditioning the deformation network on semantic features (S) rather than world-space coordinates (XYZ). On the other hand, the semantic features enable cross-category deformation, as we demonstrate later.

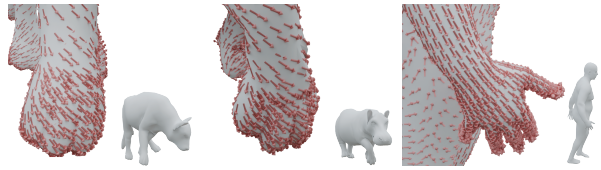


Figure 5.6: Visualization of learned $\mathbf{x}_i$ vectors for multiple shapes. The learned vectors are consistent across shapes, allowing for generalized deformation transfer.

Smooth basis. In Fig. 5.6 we visualize $\mathbf{x}_i$ across three shapes, which is used to construct our basis (see (5.8)). Firstly, we observe that σ_ϕ produces smooth vector fields on the surface. Secondly, when paying attention to the front right legs of the cow and the hippo, and the right hand of the human, the basis vectors point in a similar direction over the paw/hand on each shape. This allows our method to learn generalized deformation between diverse meshes.

Generalized deformation results. In Fig. 5.5 we present reposed shapes from datasets unseen during training, demonstrating that our method generalizes to unseen and unrigged meshes at test time. To this end, we condition the network on manually defined bone rotations, represented by $\mathbf{p}^h$ and $\mathbf{p}^a$. In the top row, we demonstrate how the relative deformations are applied from a source pair (blue) to a target pair (yellow) within the same category. Note that due to our formulation and relative pose conditioning, deformations can be freely applied relative to the input pose, producing realistic articulated deformations of meshes without a rig. Due to semantic alignment, relative pose deformations can also be applied between different shape categories. In particular, we can deform humanoid meshes realistically with the SMAL kinematic model $\mathbf{p}^a$, as well as animal meshes with the SMPL kinematic model $\mathbf{p}^h$ as input (see bottom row of Fig. 5.5).

Due to our method's generalizability, we can further condition the deformation network on motion sequences stemming from the AMASS test set. We apply such deformations to animal and humanoid characters in Fig. 5.10, without aligning the characters globally with the training set. The produced motion sequences are smooth and natural, even for unseen shapes.

Training orientation While we have shown that Re-NJF produces rotation-equivariant results at inference time, we further demonstrate its benefits at training time. For this purpose, we sample random global rotations during training, and apply the same rotation to each training pair, $\mathcal{M}_s$ and $\mathcal{M}_t$. In Fig. 5.7, we show that only our method exhibits convergence behavior identical to training without random rotations, whereas the other baseline methods converge considerably slower or fail to converge at all.

Ablations Ablation result can be found in Table 5.1, where we report test results on the within-class test sets. Generalization performance is shown in Fig. 5.8, where visual results for in-between-class performance are shown. We first evaluate the performance when replacing the semantic features sets with wave kernel signatures sets (*WKS*) [206]. We find that this produces strong rotation-equivariant results within class but struggles

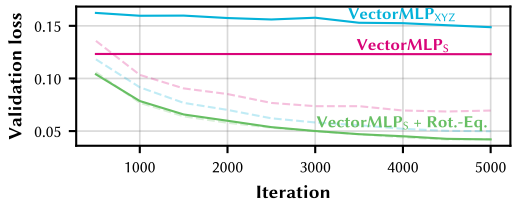


Figure 5.7: Effect of rotating the training and validation set on each method on the validation loss. The dashed line shows the convergence behavior without rotating the meshes. Only our method demonstrates the same convergence behavior under both conditions.

to generalize across categories. Showing that our method produces rotation-equivariant results for other types of rotation-invariant input features, but semantic features are needed for strong generalization.

Next, we evaluate the three different basis function parameterizations. Firstly, we use a global coordinate basis obtained by calculating the principal axes of the shape, denoted as *global*. Secondly, we set $\mathbf{x}_i$ to be the first component of the principal component direction (see *PC*). Thirdly, in *Face*, the basis is aligned consistently with one of the triangle's edges. Each of these approaches produces worse results than our proposed method and fails to generalize, except for *global*. However, its generalization performance is poor because the basis changes with input shape, leading to severe deformation artifacts.

Lastly, we demonstrate that removing the feature preprocessing with b_ω, as well as using the absolute pose as input (see *Absolute*), increases the error substantially (see *no* b_ω).

Table 5.1: Quantitative ablation results for our method within the same class. Results are averaged across all angles, equivalent to 'With Rotation' in Table 5.3a. In the first section, we compare different input features to σ_ϕ, as well as using the absolute pose as input. In the middle section, different parameterizations of the basis function are ablated. The last section shows results when omitting b_ω.

Variant	L2-V	L2-J	L2-N
Ours	**5.18±3.92**	**4.25±1.77**	**13.48±6.23**
INPUT FEATURES AND POSE			
+ WKS	5.70±3.68	4.48±2.17	14.41±7.74
+ absolute pose	6.70±4.38	4.87±1.94	16.29±6.73
COORDINATE FRAME			
+ Global	11.86±8.63	7.31±2.65	26.03±11.21
+ PC	9.29±5.48	6.63±2.51	20.92±9.43
+ Face	8.18±4.95	6.66±2.41	20.68±8.81
FEATURE PROCESSING			
+ No b_ω	8.69±4.70	6.15±2.16	21.54±8.58

5.5.3 Application: Inverse kinematics without rigging

We demonstrate an application of a trained network outside direct control of the pose parameters. The network represents some space of expected deformations. This can be used in an optimization for inverse kinematics. In inverse kinematics, finding the configuration of a kinematic skeleton is treated as an optimization problem. Instead of specifying the relative bone rotations directly (as in Sec. 5.5.2), the bone rotations are

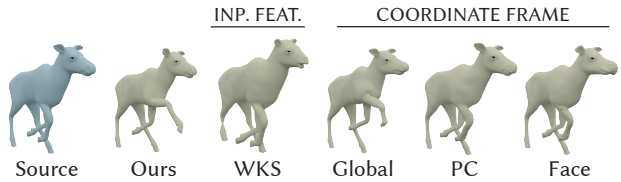


Figure 5.8: Qualitative ablation results of our method in between class. For each ablation, we apply a manual deformation to the source mesh where we rotate the collar bone 90° in $\mathbf{p}^h$.

optimized to match some target constraint. One can, for example, specify the hand of a person to be in a specific location in Euclidean space and iteratively optimize the bone configuration to fulfill this requirement.

While our deformation network does not utilize a hierarchical kinematic tree, it has been trained on data originating from one. Therefore, we can optimize the implicit bone rotations passed to the network, $\mathbf{p}^h$ and $\mathbf{p}^a$, to match a target configuration, rather than optimizing the vertex positions or per-face Jacobians directly. This allows us to deform meshes in a manner similar to inverse kinematics, even without a given rig for the input mesh.

Optimization Let $v_i : \mathbf{p} \rightarrow \mathbb{R}^3$ be the function that maps a pose vector to a vertex position i (See (5.13), (5.7), (5.6)), omitting the semantic features. We aim to find the pose vector that minimizes the following term:

$$E_{ik} = \min_{\mathbf{p}} \sum_i ||\mathbf{v}_i^{tgt} - \mathbf{v}_i(\mathbf{p})||^2, \tag{5.16}$$

where $\mathbf{v}_j^{tgt}$ is the user-specified target position of the i'th vertex. We choose these vertices by defining the position of a source and target sphere, where all vertices within the source sphere are subject to (5.16). We optimize the pose vectors for 500 iterations with the AdamW optimizer with a learning rate of 0.005.

Baselines We compare our method against VectorMLP$_{XYZ}$ to demonstrate the benefit of rotation-equivariance in this setup. Note that input meshes are purposefully rotated to highlight the shortcomings of VectorMLP$_{XYZ}$. We further compare against a baseline, where we directly optimize the relative per-face Jacobian $d_\theta(\mathbf{c}_i)$, instead of the pose vectors. As is common, we add an ARAP [292] regularizer, $\lambda_e E_{ARAP}$, to encourage the deformations to be rigid, where λ_e is the weighting term. We implement E_{ARAP} penalizing the deviation of the singular values of $\mathbf{J}_i'$ from 1 [326], and set $\lambda_e = 1000$. This baseline is referred to as the ARAP-only baseline. We further find that our method and VectorMLP$_{XYZ}$ benefit from ARAP regularization as well. Due to the already constrained optimization space, a smaller weighting factor of $\lambda_e = 1$ suffices.

Results Results are shown in Fig. 5.9, where the blue sphere shows the source position and the green one shows the target position. While all methods succeed in placing the vertices at the correct target positions, ours produces the best visual results. Our method produces a realistic bend at the elbow to estimate the hand's position (top row)

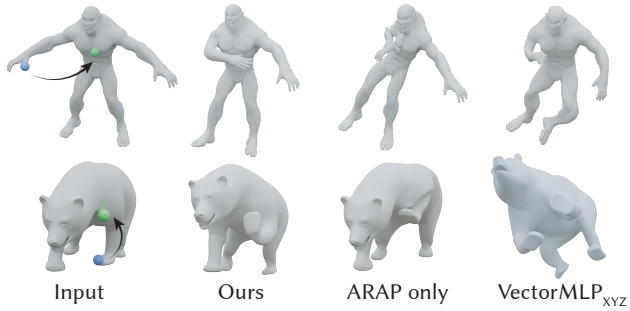


Figure 5.9: Results of our pseudo-inverse kinematics pipeline. The blue sphere marks the vertices that should move to the green target sphere. Optimizing the pose embedding rather than the Jacobians directly results in articulated deformations, even though the meshes are not rigged.

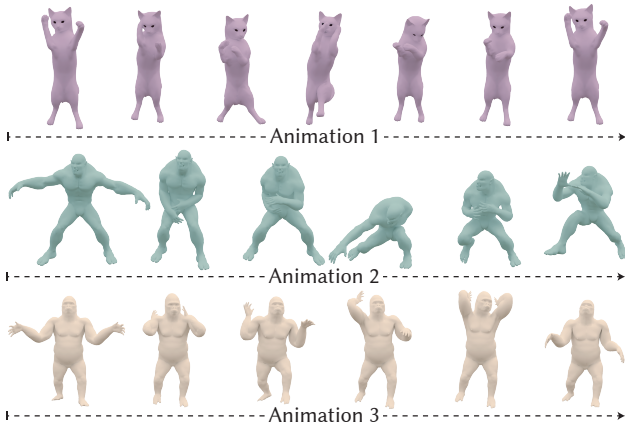


Figure 5.10: Three different animation sequences, parameterized by SMPL bone rotations, from the AMASS test set [179] applied to meshes not seen during training.

and deforms the bear's leg and torso in a natural manner (bottom row). In contrast, VectorMLP$_{XYZ}$ produces spurious deformations of other limbs, and ARAP-only is not able to preserve the volume of the limbs (top row). This demonstrates that Re-NJF could help with restricting deformations during other optimization tasks and equivariance is of the essence.

5.6 Conclusion

We presented Re-NJF, a rotation-equivariant formulation of Neural Jacobian Fields. We achieve rotation-equivariance by expressing Jacobians local tangent frames, which are aligned with semantic features. Our formulation is independent of the input and output orientations and initial deformations of meshes and therefore allows us to apply deformations at inference time without aligning them with the orientation or configuration of the training dataset samples. Because the local tangent frames are aligned to semantic feature gradients, our approach generalizes across widely varying shapes and facilitates deformation transfer between shape categories such as humans

and animals.

Limitations and Future Work Our full pipeline uses semantic features based on 3D mesh renders. We find that meshes with strong occlusions and intersections, such as those in the MOYO dataset [327] used in PoissonNet [295], yield poor semantic features. Future work could explore how to effectively inpaint the semantic features in a surface-aware manner. Furthermore, extracting semantic features [147] remains a bottleneck, as it takes several minutes per mesh. We have focused on relative pose transfer due to the difficulty of encoding human and animal poses with only intrinsic information. Therefore, our method could benefit from a rotation-invariant pose encoder that does not use world-space coordinates, which could help it learn in an absolute pose transfer setup. Lastly, our method inherits the limitations of NJF and PoissonNet, for example, that deformations are currently supported only on meshes consisting of a single component.

6 Conclusion

In this dissertation, we explore 3D deformation methods based on neural representations, either using neural rendering or neural features from pre-trained models for deformation tasks. We start by investigating how a kinematic 3D model could be reconstructed solely from multi-view video. Because of its kinematic nature, we are able to manually define bone angles and animate each model after reconstruction. However, doing this manually is cumbersome, and without the necessary expertise, the resulting animations tend to look unrealistic. Consequently, we investigate how 2D video diffusion priors can be utilized more efficiently for 3D model animation. While we are able to produce results quickly, we find that the semantic features used to track 3D deformation can be erroneous and ambiguous. One of these ambiguities stems from the fact that the internal feature representation struggles to differentiate between instances of the same semantic class. We address this issue with our surface-aware semantic features, which show surprising capabilities for deformation transfer. These features allow us to learn Neural Jacobian Fields on, for example, only horses, yet still generalize to shapes such as dogs, cows, and hippos. Finally, while the surface-aware semantic features are largely rotation-invariant, the underlying deformation method, Neural Jacobian Fields, is not rotation-equivariant, making it dependent on the orientation of the training meshes. We therefore propose a rotation-equivariant formulation of Neural Jacobian Fields that allows for the deformation of meshes in any global or per-part orientation while preserving the generalizability of the semantic features.

6.1 Challenges & Future Work

The field of Visual Computing is undergoing rapid change: 3D Gaussian Splatting is replacing Neural Radiance Fields, and the usage of 2D foundation model features is becoming ubiquitous. We therefore discuss which challenges have already been addressed and which remain for future work.

In the second Chapter, we explore how to render and deform explicit neural points. Due to the explicit geometry, the point cloud can be easily forward-warped using linear blend skinning. However, obtaining 2D images from these neural point clouds is slow due to volumetric rendering. 3D Gaussian Splatting now dominates 3D reconstruction, due to its fast rasterization-based rendering, replacing Neural Radiance Fields as the state-of-the-art method. When adopting 3D Gaussians for dynamic scenes, each Gaussian can be easily warped in space due to its explicit nature, similar to our method. This leads to significant progress in producing dynamic and photo-realistic 3D representations [328]. Unsurprisingly, follow-up work therefore explores how to combine skeletal discovery with dynamic Gaussian point clouds [329]. At the same time, generalizable models for object rigging [330] are now available, capable of producing skeleton and skinning weights for 3D assets in a regressive manner. In contrast, our skeletons are created ad hoc using a heuristic based on the geometry of the input point cloud. An interesting challenge is therefore to investigate how such models, which encode kinematic information from

large 3D datasets, could be utilized during the dynamic reconstruction of arbitrary articulated objects.

In a similar vein, we introduce a rotation-equivariant deformation network in Chapter Five that uses semantic features derived from 2D image models. Because rotation equivariance is defined per face, our model can apply learned articulated deformations to shapes in any orientation. A natural extension would be to integrate our deformation model in dynamic 3D reconstruction tasks, since the underlying features are derived from images to begin with. This would significantly reduce the degrees of freedom during reconstruction, which is severely ill-posed. To further extend this idea, investigating how to train Re-NJF in a generative manner could enable it to be used as a template-free deformation prior. This would require learning how to encode the poses of input meshes so that the model can learn the conditional distribution over what each pose could deform to. However, Re-NJF currently struggles to encode such poses due to the lack of extrinsic information. Overcoming this challenge would then allow one to restrict the reconstructed motion to the pose distribution induced by a dataset. VPoser [17], for example, can be used as such a prior for inverse problems*, albeit for human bodies only. As noted in the previous paragraph, integrating a pre-trained rigging model during a dynamic reconstruction task could improve performance, and a template-free deformation prior could further help constrain deformations beyond what has been explored in the field.

The application of our proposed methods, specifically Re-NJF combined with surface-aware semantic features, extends beyond Visual Computing. For example, finding species-consistent keypoints and estimating poses is of interest to animal behavioral analysis [331]. Since our methods generalize across species, they could improve pose-estimation tools used in this field and, going further, enable modeling joint deformations over time rather than 2D keypoint trajectories. Robotics is another natural target: interacting with kinematically deforming objects such as humans is an active research area [332], where robotic learning pipelines often rely on specific object templates. The combination of cross-category generalization and rotation-equivariance offered by our method could enable manipulation of unfamiliar articulated objects observed in arbitrary poses.

Chapters three, four, and five explore how to use and improve 2D semantic features from foundation models for 3D tasks. Related work [305] is starting to address the issue of left-right ambiguities for 3D representations as well. However, extracting these semantic features is still a post-processing step that can break in the presence of strong self-occlusions or intersections. Given that these features produce exceptional results across a variety of downstream tasks, addressing these remaining limitations is of great interest for advancing the field. One could investigate how to account for occluded regions by incorporating surface information into the feature distillation process.

On the other hand, PartField [333] trains a generalizable 3D part-segmentation network that operates inherently in 3D. It produces part segmentation results similar to those obtained by clustering surface-aware features on meshes, although their method uses extrinsic information. While not a foundation model per se, and with its training dataset still under a million samples, it shows a promising direction for future work. One can therefore wonder how and if such 3D and 2D feature models could be effectively combined. 3D-native approaches and 2D feature distillation methods have complementary strengths and weaknesses, and combining them could address each other's limitations. Training

* https://github.com/nghorbani/human_body_prior

a foundation model that treats both domains equally has the prospect of producing powerful, generalizable features that could benefit applications in either domain.

In summary, this dissertation contributes methods for 3D deformation learning based on neural representations. While challenges remain in how to utilize deformation priors during dynamic 3D reconstruction and in bridging the gap between the 3D and 2D semantic representations, our contributions and the rapid progress in foundation models provide a strong basis for addressing them. We are optimistic that the convergence of 3D and 2D representations will enable the development of powerful tools for manipulating and animating 3D representations efficiently, as well as tools beyond the Visual Computing domain.

Appendix

A Appendix: Template-free Articulated Neural Point Clouds for Reposable View Synthesis

A.1 Extra Results *Blender* dataset

Table A.1, and Fig. A.1 show the quantitative and qualitative per-scene results respectively in the *Blender* dataset [4]. While the non-reposable methods often achieve excellent results, our method is a clear improvement with respect to the other reposable method, WIM [21], while simultaneously reducing training time*.

Table A.1: Quantitative Results Per-Scene on *Blender* dataset. We only highlight best and second-best values for PSNR as the precision reported of SSIM and LPIPS in [12] is not enough for fair comparison in many cases.

	Jumping Jacks			**Mutant**			**Hook**			**T-Rex**		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
D-Nerf [4]	32.80	0.98	0.03	<u>31.29</u>	0.97	0.02	29.25	0.96	0.11	31.75	0.97	0.03
TiNeuVox [12]	34.23	0.98	0.03	**33.61**	0.98	0.03	**31.45**	0.97	0.05	<u>32.70</u>	0.98	0.03
HexPlane [73]	31.65	0.97	0.04	33.79	0.98	0.03	28.71	0.96	0.05	30.67	0.98	0.03
Tensor4D [334]	<u>34.43</u>	0.98	0.03	×	×	×	×	×	×	×	×	×
WIM [21]	29.77	0.97	0.04	25.80	0.95	0.06	25.33	0.94	0.06	26.19	0.94	0.08
Ours	**34.50**	0.98	0.03	28.56	0.96	0.03	<u>30.24</u>	0.97	0.05	**32.85**	0.98	0.02
	Stand Up			**Hell Warrior**			**Lego**			**Bouncing Ball**		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
D-Nerf[4]	<u>32.79</u>	0.98	0.02	25.02	0.95	0.06	<u>21.64</u>	0.83	0.16	38.93	0.98	0.10
TiNeuVox [12]	**35.43**	0.99	0.02	**28.17**	0.97	0.07	**25.02**	0.92	0.07	**40.73**	0.99	0.04
HexPlane [73]	34.36	0.98	0.02	24.24	0.94	0.07	25.22	0.94	0.04	39.69	0.99	0.03
Tensor4D [334]	36.32	0.98	0.02	×	×	×	26.71	0.95	0.003	×	×	×
WIM [21]	27.46	0.96	0.04	16.71	0.87	0.14	15.41	0.73	0.25	×	×	×
Ours	31.93	0.97	0.02	<u>27.53</u>	0.96	0.06	17.91	0.76	0.14	×	×	×

* Training times are shown in the main paper.



Figure A.1: Additional qualitative per-scene results of our method for the *Blender* dataset. We compare WIM [101], with our method and the ground truth. The displayed results correspond to the final state of the training procedures as described in the paper.

A.2 Extra Results *Robots* dataset

Table A.2, and Fig. A.2 show the quantitative and qualitative per-scene results respectively for the *Robots* dataset [21]. Consistent with the findings in the main paper, we see that WIM produces overly smooth results while our method can capture more details in a shorter amount of training time. However, it struggles to accurately recover the true poses for long kinematic chains (Table A.2, *Iiwa* and *Pandas*).

Table A.2: Quantitative Results Per-Scene in the *Robots* dataset.

	Atlas			**Baxter**			**Cassie**			**Iiwa**		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
WIM[21]	23.99	0.94	0.07	22.70	0.95	0.06	30.20	0.97	0.04	**31.58**	**0.98**	**0.03**
Ours	**28.71**	**0.97**	**0.04**	**28.60**	**0.97**	**0.04**	**31.84**	**0.98**	**0.04**	29.74	0.98	0.04
	Nao			**Pandas**			**Spot**					
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR ↑	SSIM ↑	LPIPS↓	PSNR ↑	SSIM↑	LPIPS↓			
WIM[21]	26.85	0.95	0.06	**32.31**	**0.98**	**0.03**	26.30	0.97	0.04			
Ours	**29.50**	**0.96**	**0.05**	30.56	0.97	0.05	**32.40**	**0.98**	**0.02**			



Figure A.2: Additional qualitative per-scene results of our method in the *Robots* dataset. We compare WIM [101], with our method and the ground truth. The displayed results correspond to the final state of the training procedures as described in the paper.

A.3 Implementation Details

A.3.1 Our method

For training, we utilize PyTorch and the Adam optimizer. We pre-train TiNeuVox for 20k iterations on the synthetic *Blender* data set, and 40k iterations on the WIM data set, and add distortion loss regularization [117] as implemented in [118]. All hyperparameters are adopted from [12].

The neural point clouds are trained for 160k iterations. In each iteration, we sample 8192 rays randomly from multiple views at time step t. On each ray, we sample multiple points p_i as a function of the voxel size of the pre-trained NeRF volume, as specified in [12]. For each sampling point, we query a maximum of 8 neighborhood points $N(p_i)$ within a radius of 0.01 using the KeOps framework [335].

For the kinematic skeleton, we use a bone length of $B_{length} = 10$ for all experiments and a point cloud density threshold of 0.05. We apply the MAT on the binary volume that we retrieve after thresholding the density volume. Assuming a single object, we remove small holes from the volume and extract the medial axis on the biggest blob via the method proposed in [336] and use the scikit-image library [337].

For the mask loss, we project the warped point cloud to five views for the *Robots* dataset and to a single view for *Blender*, as only a single view per timestamp is available. Furthermore, we subsample the point cloud and ground truth mask to 3 000 points.

We fine-tune neural point features $\mathbf{f_i}$, skinning weights $\mathbf{w}_i$, joints J, density and color regressor Φ_d and Φ_c and train the pose regressor Φ_r and feature point decoder Φ_p from scratch. The MLP of Φ_r has 4 layers of size 128, except for the first layer which has a size of $128 + dim(\gamma(\mathbf{x})) = 191$, where γ is the positional encoding. For all MLPs and the weights, we use a learning rate of 0.0001, except for Φ_r which uses a learning rate of 0.001. We further set the learning rate of α and J to 0.00001. We further utilize learning decay, as in [12, 65] which decays the learning rate by 0.1 after 80 000 iterations. We train on a single Nvidia GPU RTX 3090.

A.3.2 Validation

We used the author's code[†] to reproduce the results of Watch-It-Move [21].

For the *Robots* datasets we adapted the author's configuration files. First, we modified the selection of training and testing views to match the split used in our experiments. That way we held out images from the 10th and 20th camera for testing and provided the remaining 18 cameras for training. Second, to enable early pose fitting and a meaningful computation of progressive learning metrics (see Fig. 4 in the paper), we disabled the gradual scheduler of the training frames. We kept the author's initialization phase with only the first 10 timestamps accessible for the first 10 000 iterations but we provided the entire training set afterwards. This corresponds to the moment past the 1-hour mark in the paper Fig. 4 where the metrics start to improve as the model gets a chance to fit poses of the entire sequence. Without this modification, the network would need another 70 000 steps to access this information. Note that we have not observed a meaningful

[†] https://github.com/NVlabs/watch-it-move

difference in the quality of the final trained model as a result of this modification. Finally, we reduced the batch size from 16 to 8 views to fit into the 24 GB VRAM of our Nvidia RTX 3090 GPU.

We applied the same training strategy and settings when training with the *Blender* dataset and used the author's pre-trained snapshots for the *ZJU-MoCap* dataset.

A.4 Extra Results Ablations

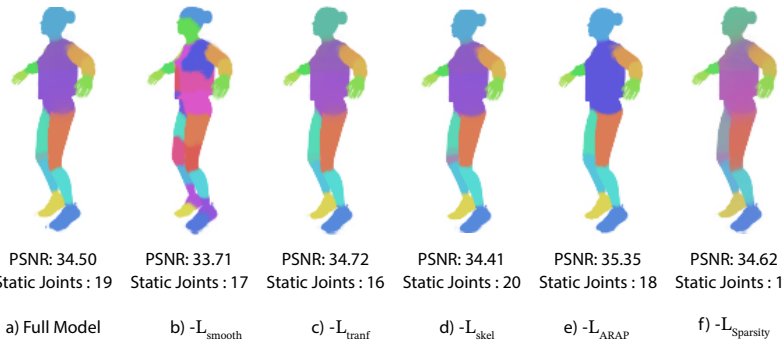


Figure A.3: Additional ablation results on *Jumping Jacks* scene from the *Blender* dataset. The skinning weights and PSNR values are displayed prior to the simplification step. The static joint counts refer to the number of joints that can be removed in our simplification step to ease the reposing task. See Sec. 2.4.3. b) - f) show the show the results without the denoted regularization term.

Extra ablation results for the *Jumping Jacks* scene from the *Blender* dataset can be seen in Fig. A.3. We observe that without $\mathcal{L}_{smooth}$ (Fig. A.3b), the skinning weights are less consolidated and wrong skinning weights are produced (compared to the full model in Fig. A.3a). This limits the image quality represented by the PSNR score. Next, without $\mathcal{L}_{tranf}$ (Fig. A.3c) joint rotations are less sparse and, therefore, we detect fewer static joints for removal during the skeleton simplification (see Sec. 2.4.3). The presented static joint counts were measured for a simplification threshold of 20 degrees). Next, $\mathcal{L}_{skel}$ (Fig. A.3d) does not have a major effect in this scene. However, for scenes with thin structures, omission of $\mathcal{L}_{skel}$ allows the skeleton to drift outside of the geometry. This results in bad reconstruction (see Fig. 8c in the paper). Next, utility of $\mathcal{L}_{ARAP}$ depends on the object class. For strictly articulated shapes (see the robot in Fig. 8a in the paper), it enforces part rigidity and avoids unrealistic deformations. However, its contribution is less obvious for partially soft shapes such as humans (see Fig. A.3e). Despite this, we used the same loss weights for all our results without notable issues. Finally, omission of $\mathcal{L}_{sparse}$ decreases separation of the part labels (note the reduced label color saturation in Fig. A.3f). This suggests that this term reduces entanglement between points and joints which indirectly leads to a higher static joint count for skeleton simplification.

A.5 Notation

An overview of our notation and trainable parameters can be found in Table A.3.

Table A.3: Overview of our notation.

Symbol	Description	Trainable	Initialization
$\mathbf{p}_i$	3D Point from canonical point cloud	Fixed	Pre-trained model
$\mathbf{f}_i$	Feature vector associated each $\mathbf{p}_i$	Fine-tuned	Pre-trained model
$\hat{\mathbf{w}}_i$	Blend skinning vector **before** softmax	Fine-tuned	Inv. bone-to-point dist.
α	global scalar which scales $\hat{\mathbf{w}}_i$	Trained from scratch	0.1
Φ_c	Color regressor	Fine-tuned	Pre-trained model
Φ_d	Density regressor	Fine-tuned	Pre-trained model
Φ_p	Feature decoder	Trained from scratch	Random
Φ_r	Pose regressor	Trained from scratch	Random
Φ_{pe}	Pose embedding network	Trained from scratch	Random

A.6 Training Schedule for the *ZJU-MoCap* dataset

We make two adjustments while training in the *ZJU-MoCap* dataset that both aim to compensate for a bias towards observation of early timestamps due to our incremental training data scheduler. We deem this important since the real captured human subjects do not exhibit all motion modalities uniformly through the entire sequences. This is different from the synthetic *Robots* and *Blender* datasets where motion is simplistic, exaggerated and evenly distributed throughout the sequences.

First, we sample the training timestamps with importance sampling to increase probability of later timestamps and compensate for their shorter overall accessibility during the course of training. We define an importance of a timestamp as the inverse of the total sample count so far. Second, we only enable the sparsity regularization after 160K training iterations when the scheduler converges and all timestamps become accessible. This avoids loss of kinematic joints by early merging when not all motion modalities were observed yet. Neither change leads to a computational overhead.

A.7 Additional Animations

We demonstrate animation capabilities of our method by creating two animation sequences based on manually defined joint rotations. First, we create a walking sequence of the *Spot* robot from the *Robots* dataset in Fig. A.4. Second, we animate a scene from *ZJU-MoCap* from two different views Fig. A.5.

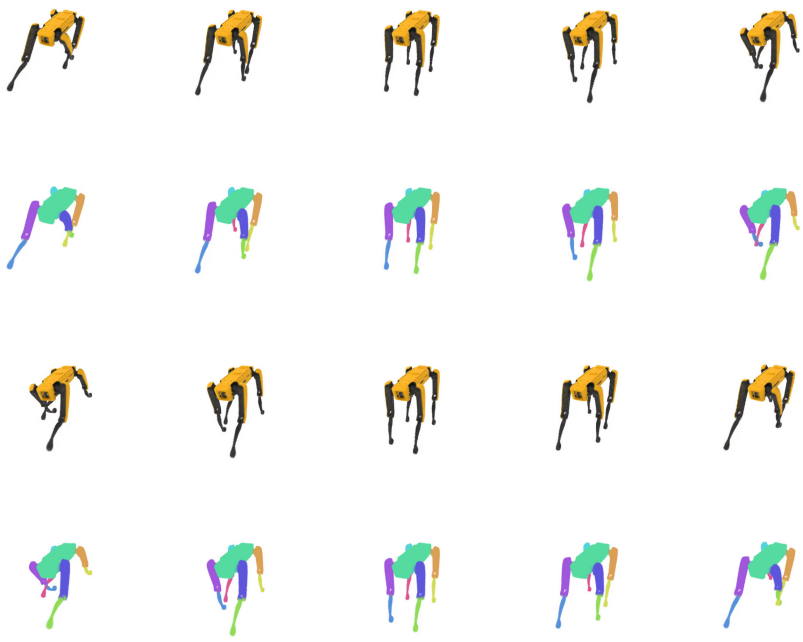


Figure A.4: Manually defined walking animation for the *Spot* from the *Robots* dataset.

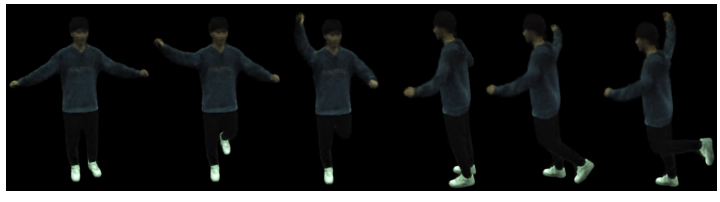


Figure A.5: Manually defined animation for scene *384* from the *ZJU-MoCap* dataset from two different views.

B Appendix - MotionDreamer

B.1 Additional Implementation Details

B.1.1 Video Diffusion models

We use the official implementations for VideoComposer [123] (VC), DynamiCrafter [124] (DC), as well as for Stable Video Diffusion [122] (SVD), where the latter two are accessed through the Diffusers library [338]. We adopt the same hyperparameters for all wherever possible. We set the classifier-free guidance [194] to 6 and we generate 16 frames with an assumed framerate of 16 fps. We use the recommended schedulers with $T = 50$ inference steps. We discuss the choice of conditioning images for each model and the omission of SVD from our experiments in Appendix B.4.1. Finally, we observe that VC provides faster inference than DC, which is why we adopt it for our quantitative experiments that require large number of optimizations.

Matching rendered image to VDM features for DC Unlike VC and SVD, Dynamicrafter does not enforce the input image to be the first frame of the output video, which is an assumption of our method. This is because all VDM frames are initialized with the same embedded input image: $\mathcal{E}(\mathbf{x}) = \mathbf{z}^0 = \mathbf{z}^1 = ... = \mathbf{z}^L$ and then they drift during the inference. We observe that this drift is minimized for the output frame matching the input image and hence we explicitly detect the frame l^* where features change the least between the inference steps t:

$$l^* = \arg\max_l \sum_t \frac{\kappa(\mathbf{A}_t^l, \mathbf{A}_{t-1}^l) - \mu_{\mathbf{A}_{\kappa,t}}}{\sigma_{\mathbf{A}_{\kappa,t}}}, \tag{B.1}$$

where $\mu_{\mathbf{A}_{\kappa,t}}$ and $\sigma_{\mathbf{A}_{\kappa,t}}$ are the mean and standard deviation of the cosine similarities of activations $\mathbf{A}_t$ at step t. Finally, l^* can be used as the frame index for feature reprojection.

B.1.2 Animation Models

We experiment with four different animation models.

SMPL Skinned Multi-Person Linear [16] is a skinned mesh-based human model that supports various body shapes and human poses. Vertices are deformed based on forward kinematics and linear blend skinning: $\mathbf{u}_i^l = \sum_b w_{b,i} \mathbf{T}_b^l \mathbf{u}_i^{\text{init}}$, where $\mathbf{T}_b^l \in \mathbb{R}^{4\times4}$ is the roto-translation of bone b at time step l and $w_{b,i}$ the skinning weight determining how strongly vertex $\mathbf{u}_i$ is attached to b. $\mathbf{T}_b^l$ is defined recursively by its parent bone transformation according to a kinematic hierarchy.

SMAL SMAL [75] is another skinned model that can represent various quadrupedal animals, namely lions, cats, dogs, horses, cows and hippos. It follows the sample approach of forward kinematic and linear blend skinning for reposing as SMPL. We make use of the SMALify [322] implementation in our work.

FLAME FLAME [15] also adopts the SMPL formulation but expands it by articulation of the jaw, and the eyes. It utilizes blend-shapes to model facial expression offsets for all vertices in the mesh: $\mathcal{U}_{exp} = \sum_{n}^{|\vec{\psi}|} \vec{\psi}_n \mathbf{E}_n$, where $\vec{\psi}_n$ denotes the n'th expression coefficient, $\mathcal{E} = [\mathbf{E}_n, ..., \mathbf{E}_{|\vec{\psi}|}] \in \mathbb{R}^{3N \times |\vec{\psi}|}$ is the orthonormal expression basis, and $\mathcal{U}_{exp}$ contains the vertex expression offset for each $\mathbf{u}_n$. We further find it beneficial to scale the expression coefficient $\vec{\psi}_n$ by a factor of 5 in FLAME

Note that we keep the shape parameters fixed for SMPL, SMAL, and FLAME. Please refer to the corresponding work for more details.

Neural Jacobian Fields (NJF) Our method also supports arbitrary meshes that are neither rigged nor have blendshapes. To animate these types of meshes we make use of NJF [119]. In NJF, the deformation is obtained by indirectly optimizing the per-triangle Jacobians $J_i \in \mathbb{R}^{3\times3}$ for each face f_i, instead of directly regressing the displacement for each vertex. To retrieve the deformation map $\Phi*$, a Poisson problem is solved: $\Phi* = \min_{\Phi} \sum_{f_i} |f_i| \parallel \nabla_i(\Phi) - J_i \parallel_2^2$, where $\nabla_i(\Phi)$ is the Jacobian of Φ at triangle f_i and $|f_i|$ represents the area of the triangle. We follow the implementation of Gao et al. [339], and initialize the Jacobians with identity matrices. Besides the Jacobians, we additionally optimize root rotation, center of rotation, and a global translation vector. We also makes use of the Jacobian regularization [339] to avoid diverging too far from the initial geometry. Consequently, we expand our full optimization objective with an additional term

$$\mathcal{L}_j = 1/(2M) \sum_i (\parallel J_i - I \parallel_2) + \parallel J_i - I \parallel_1),$$

where M is the number of triangle faces. Therefore, for NJF, we minimize $\mathcal{L}' = \mathcal{L} + w_j \mathcal{L}_j$, where $w_j = 0.5$.

Our requirements for the inputs mesh are entirely dependent on the animation model. For NJF, we assume a mesh with a single connected component. For multi-component meshes, we adopt the preprocessing from Wang et al. [340] and transform the mesh representation into an SDF and resample the mesh based from this SDF. We additionally decimate faces through Quadric edge collapse [341] to reach 8 000 vertices. In practice, we observe that this procedure is robust even for meshes that are not perfectly watertight nor 2-manifold.

For all animation models, we scale the global translation vector $\mathbf{t}$ by 0.1.

B.2 User Study

B.2.1 Baseline methods

DG4D We use the original implementation provided by Ren et al. [137] but adapt two hyperparameters such that the model can be trained with only 24 GB of VRAM. Namely, we reduce the batch size from 14 to 8 and and the number of views per step (*n_views*) from 4 to 2. In its original setup, DG4D automatically removes the background of the input image before passing it to a VDM with a tool *Rembg**. However, as we show in Appendix B.4.1, VC produces better results for input images with background. Therefore, for VC, we remove the background after the video generation instead by applying Rembg to each video frame. When using DC, the pipeline of DG4D is unaffected.

MDM-MT We observe a lack of class-agnostic end-to-end pure motion generators. Therefore, we combine a human-specific motion generator with a general motion transfer method while accepting that the performance of such solution will depend on morphological and semantic proximity of the source and target shape class. To this goal, we first use a pre-trained author's implementation of the text-conditioned motion diffusion by Tevet et al. [184] to generate a unique 2D skeletal human motion sequence for each example in our study. We adapt the motion text prompts used for our method (Table B.2) to the human domain using a template "a person is [ACTION]" e.g., "a horse is walking" → "a person is walking". Next, we use a 2D-to-3D human body pose uplifting method adapted from the code of Zuo et al. [342] to obtain sequence of SMPL [16] meshes. Finally, we follow the procedure and code of Liao et al. [190] to retarget the SMPL animations to our target meshes. We aply this step consistently even for the SMPL target mesh. Finally, we render the first 16 frames of the resulting mesh sequences in the same way as for our own method.

B.2.2 Stimuli

In our study we utilize 10 different shape-prompt pairs (2 SMAL, 2 FLAME, 2 SMPL, and 4 Neural Jacobian Field combinations) and combine them each with 2 different VDMs resulting in 20 unique videos for each evaluated method. We compare pairwise to 2 methods (DG4D and MDM-MT) for the first 3 questions, and we similarly compare to 2 methods (DG4D and VDM) for the last additional question. In total this produces $3 \times 2 \times 20 + 1 \times 2 \times 20 = 160$ study trials.

Meshes We extract the surface models for SMPL [16], FLAME [75] and SMAL [15] from their official implementations. For SMPL, we opt to lower the arms to 45 degrees from the original T-pose, while we use the default "zero" pose parameters for others. For NJF [119], we use the 4 open assets listed in Table B.1.

VDM Target Motion Prompts Table B.2 lists VDM prompts used to generate the motion sequences for the stimuli in our study.

* https://github.com/danielgatis/rembg

Table B.1: Mesh assets used to evaluate our method with NJF.

Shape	Author	License	URL
Bunny	Stanford	Stanford Public	http://graphics.stanford.edu/data/3Dscanrep/
Lego truck	Mildenhall et al. [2]	MIT License	https://github.com/bmild/nerf
Raptor	Gatzegar	TurboSquid Standard	https://www.turbosquid.com/3d-models/raptor-dinosaur-model-1538088
Palm tree	mr_zaza	TurboSquid Standard	https://www.turbosquid.com/3d-models/3d-tropic-palm-tree-model-2090490

Table B.2: Prompts used to generate stimuli in our study.

Shape	Prompt
SMPL	*"A person jumping up"*
SMPL	*"A person walking forward"*
Horse (SMAL)	*"A horse walking"*
Horse (SMAL)	*"A horse jumping"*
FLAME	*"A person laughing"*
FLAME	*"A person being very angry"*
Bunny	*"A bunny shaking its ears"*
Lego truck	*"A yellow truck moving its shovel up and down"*
Raptor	*"A raptor jumping"*
Palm tree	*"A palm tree swaying in the wind"*

Single-View Texturing Prompts Table B.3 shows the positive and negative prompts used for Single-view Texturing as an input for the ControlNet diffusion model [195] for each shape in our experiments and the prompt for the Stable Diffusion XL [197] background inpainting.

Table B.3: Prompts used for our Single-View Texturing and background inpainting.

	Prompt	**Negative Prompt**
SMPL	*"A photo of a clothed person wearing pants and tshirt in front of a <background>, photorealistic, 4k, DLSR"*	*"grey, gray, monochrome, distorted, disfigured, naked, nude"*
FLAME	*"A portrait photo a face in front of a <background>, photorealistic, 4k, DLSR, bokeh"*	*"grey, gray, monochrome, distorted, disfigured, render, teeth, hat"*
SMAL	*"A photo of a <animal> in front of a <background>, photorealistic, 4k, DLSR"*	*"grey, gray, monochrome, distorted, disfigured, render"*
Others	*"A photo of a <object> in front of a <background>, photorealistic, 4k, DLSR"*	*"grey, gray, monochrome, distorted, disfigured, render"*
Inpainting	*"Background image of a <background>"*	*"Person, face, animal, object"*

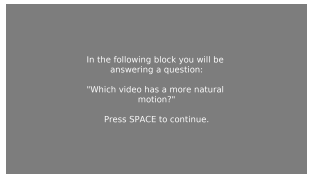
B.2.3 Instructions

Fig. B.1 shows the instructions as presented to each participant before the start of the study. Fig. B.2, Fig. B.3, Fig. B.4, Fig. B.5 show screenshots of our study interface for each of the four distinct questions (3 questions in the main part and one additional question). The questions were presented in four blocks sequentially always in the same order. There was an instruction screen displaying the next question shown at the beginning of each block. The order of blocks was fixed but the order and layout of the trials was randomized for each participant.

Study Information
The study will take approximately 15 minutes.
You can drop out of the study whenever you want.
You may voice concerns or ask questions throughout the study.
You may take breaks.

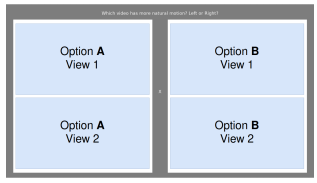
Study Procedure
You will go through 4 different blocks, each block associated with one question.

Before the start of each block, the question will be written on the screen and you will have to press SPACE to initiate each block. Example:



The questions are:
- Which video has more natural motion?
- Which video has fewer visual artifacts?
- Which video captures the prompt better?
- Which video do you prefer overall?

After initiating a block, you will see two videos of rendered 3D objects. One on the left and one on the right. Block one, two and three show two views of the scene. Example:



You will have to indicate your preference given the question of the block. For block one, two and three, the question will also be displayed at the top of the screen.

You can choose your preference by pressing the RIGHT or LEFT arrow key.

Figure B.1: Study instructions that were read out and explained to our participants before the study.

B.2.4 Detailed Results

Here, we present a break-down of the results from our user study separately for the human stimuli (Fig. B.6), where the human-specific MDM-MT baseline performs well and for the remaining stimuli (Fig. B.7), where our class-agnostic method dominates. We also offer a detailed breakdown in Table B.4.

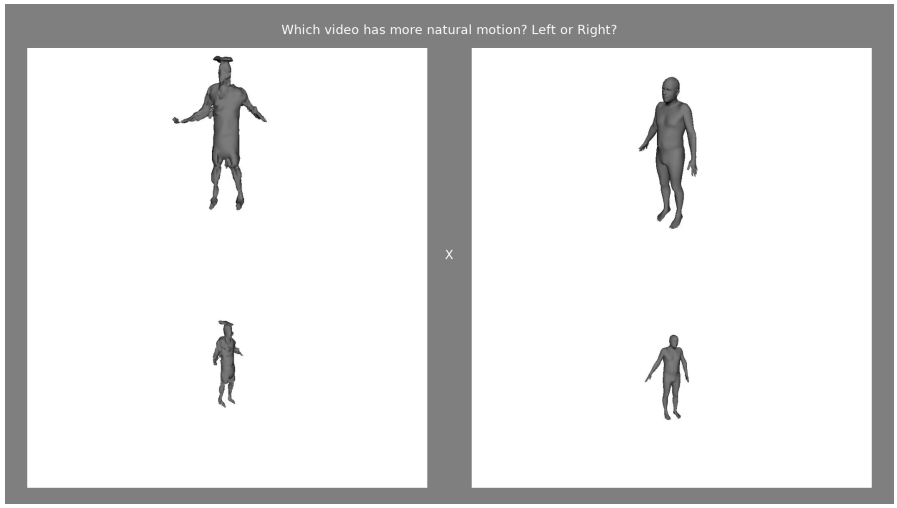


Figure B.2: A screenshot of a trial for the 1st question in our user study.

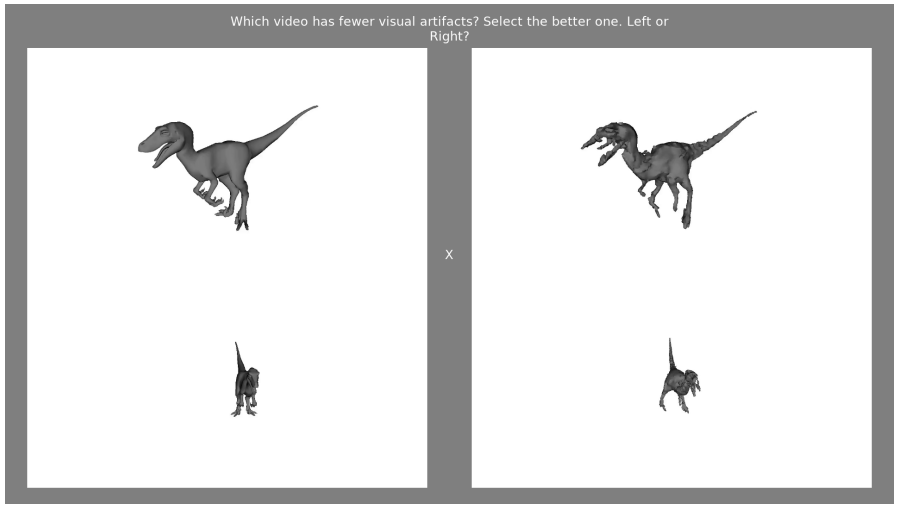


Figure B.3: A screenshot of a trial for the 2nd question in our user study.

Table B.4: Breakdown of our study results showing a relative preference of our method in %. Q1: Which video has more natural motion? Q2: Which video has fewer visual artifacts? Q3: Which video captures the prompt better? Q4: Which video do you prefer overall?

	Q1		Q2		Q3		Q4	
	DG4D	MDM-MT	DG4D	MDM-MT	DG4D	MDM-MT	DG4D	VDM
SMPL	91.7	39.6	100.0	47.9	91.7	89.6	87.5	70.8
SMAL	54.2	64.6	100.0	100.0	77.1	64.6	85.4	31.3
FLAME	97.9	97.9	100.0	100.0	93.8	95.8	62.5	29.2
Others	93.8	55.2	100.0	89.6	84.8	47.9	95.8	56.3
All	86.25	62.5	100.0	85.4	86.3	69.2	85.4	48.8

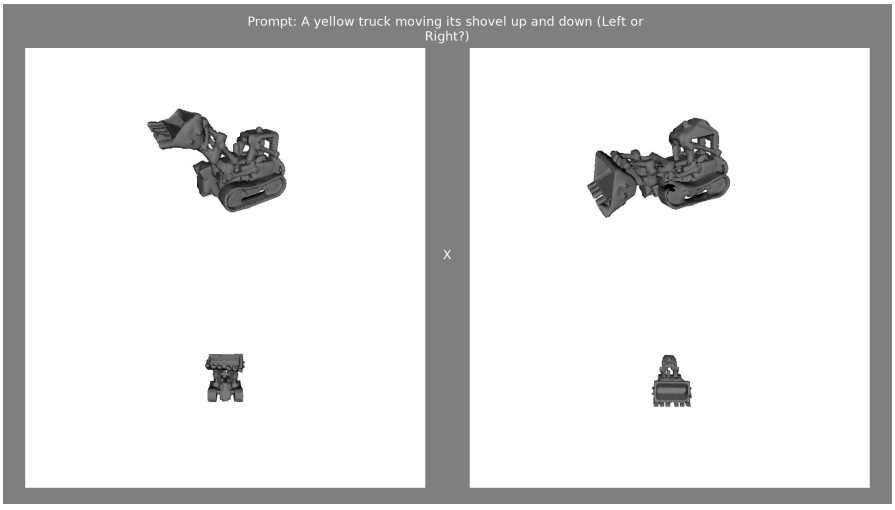


Figure B.4: A screenshot of a trial for the 3rd question in our user study.

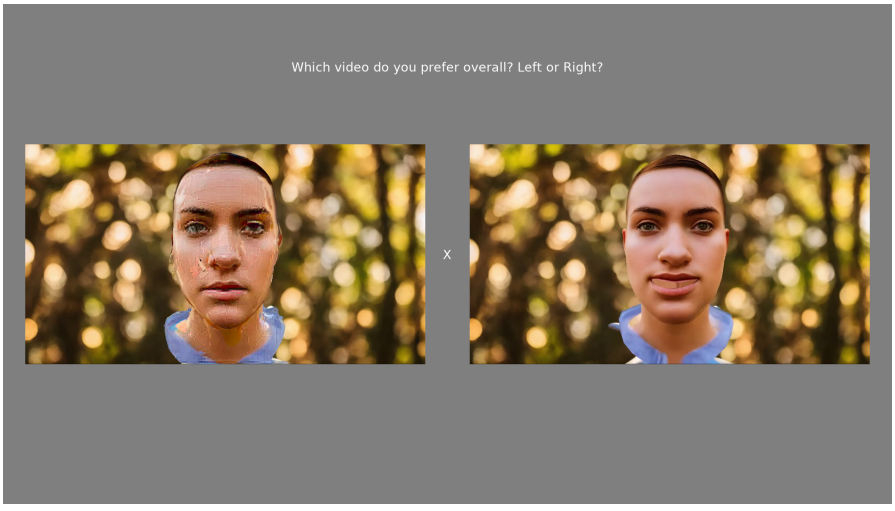


Figure B.5: A screenshot of a trial for the additional 4th question in our user study.

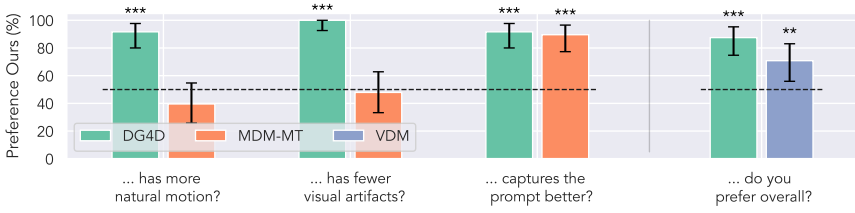


Figure B.6: Study results for *SMPL scenes only*.

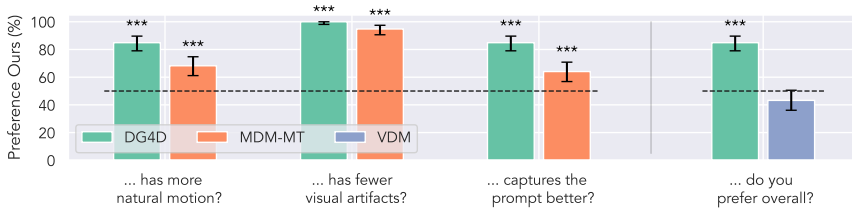


Figure B.7: Study result *without SMPL scenes*.



Figure B.8: Example of rendered AIST++ scenes. On the top: The untextured models. On the bottom: Models preprocessed by our single-view texturing.

B.3 Pose Optimization Experiment Details

B.3.1 Data

We select the first 20 frames from 20 randomly selected human dancing motion sequences in the AIST++ dataset [202]. Since our goal is not to reproduce the original camera poses, we use a single fixed camera $\mathscr{C}$ and position the first-frame SMPL mesh into the center of its viewport. Then we render the rest of the SMPL sequence with a fixed camera. An example can bee seen in Fig. B.8.

B.3.2 Methodology

We follow Tang et al. [146] to extract semantic features $\hat{\mathbf{A}}$ from our rendered videos. First, we add noise corresponding to a diffusion inference step t to the encoded the rendered video $\mathbf{x}$: $\mathbf{z}_t = \alpha_t Enc(\mathbf{x}) + \sigma_t \epsilon$, where $Enc()$ is a latent encoder for Latent Diffusion Models [132] or identity for RGB models. Then, we use $\mathbf{z}_t$ as an input to the VDM denoiser f_θ and obtain $\hat{\mathbf{A}}$ as the U-Net activations in the same manner as in our main method (Sec. 4.4).

Note that we utilize MSE loss when using RGB features for optimization, as this results in better performance compared to the cosine distance.

B.3.3 WHAM baseline

To offer a fair comparison, we evaluate four different alignment strategies for WHAM [175], because our method starts with the known pose $\mathbf{p}_{\text{init}}$. Results for either strategy can be found in Table B.5. In strategy *align*, we find the rotation and translation to align the wham output with the ground truth:

$$\hat{R} = R^{0^T}_{wham} R^0_{gt}, \text{ and } \hat{T} = diag(t)^{0^{-1}}_{wham} diag(t)^0_{gt}. \tag{B.2}$$

The transformations are then applied to the consecutive frames l: $\tilde{R}^l_{wham} = \hat{R}^l R^l_{wham}$, and $\tilde{T}^l_{wham} = \hat{T}^l T^l_{wham}$, where $\tilde{R}$ and $\tilde{T}$ are the new aligned root rotation and translation.

In $copy$, we copy ground truth root rotations and translations, i.e., we set $\tilde{R}^l_{wham} := R^l_{gt}$ and $\tilde{T}^l_{wham} := T^l_{gt}$. In *full align*, we transform not only root rotations, like in Eq. B.2, but every bone rotation. Lastly, in *copy&align*, we copy all root rotations and translation vectors from the ground truth and also transform the bone rotations as in *full align*. Note that the WHAM prediction is in full correspondence at $l = 0$ in *full align* and *copy&align*. We find that *copy&align* performs the best for WHAM and, therefore, adopt this alignment strategy in Sec. 3.5.2.

Table B.5: Evaluating different alignment strategies for WHAM.

	MPJPE	PA-MPJPE	PVE	Accel
***Textured* (default)**				
WHAM_{align}	.092 ± .038	.057 ± .015	.125 ± .047	8.0 ± 9.2
WHAM_{copy}	.092 ± .038	.057 ± .015	.112 ± .046	8.0± 9.2
$\text{WHAM}_{full\ align}$	**.059 ± .029**	**.042 ± .016**	.090 ± .039	**7.9 ± 9.0**
$\text{WHAM}_{copy\&align}$	**.059 ± .029**	**.042 ± .016**	**.075 ± .036**	**7.9 ± 9.0**
Untextured				
WHAM_{align}	.091 ± .037	.054 ± .014	.122 ± .043	7.4 ± 9.1
WHAM_{copy}	.091 ± .037	.054 ± .014	.109 ± .044	7.4 ± 9.1
$\text{WHAM}_{full\ align}$	**.057 ± .028**	**.039 ± .015**	.086 ± .038	**7.4 ± 9.1**
$\text{WHAM}_{copy\&align}$	**.057 ± .028**	**.039 ± .015**	**.070 ± .035**	**7.4 ± 9.1**

B.4 Additional Results

B.4.1 Effect of Texturing and Background for Different VDMs

In Fig. B.9 and Fig. B.10, we compare different image input variants for the three different considered VDMs: VideoComposer [123] (VC), DynamiCrafter [124] (DC), and Stable Video Diffusion [122] (SVD). We observe that VC struggles to produce coherent output for images without background images, as the object often either disappears (Fig. B.9 top) or gets distorted (Fig. B.10 top). DC exhibits resilience to this problem and performs well both with and without a background image. Therefore, we opted to use images without background for DC, since it makes the videos more similar to the typical inputs of the DG4D baseline [137]. Finally, the publicly accessible SVD model is conditioned by image only without any text prompt input. We observe that the motion produced by SVD for our image inputs often results in a global camera motion with no object motion. This is particularly prominent if no background is used. For this reason, we excluded SVD from our other experiments.

B.4.2 Qualitative Comparison to Consistent4D

We further compare our method to Consistent4D [142]. Since this is a computationally significantly more expensive method (50 minutes in its low VRAM setup versus less than 3 minutes for ours), which ened-to-end produces both the shape and the animation, we consider it a separate category from method which is better suited for quick motion prototyping and iterative animation development. This is why we did not include Consistent4D in our user study and instead provide a general discussion and a qualitative comparison here. A similar method DG4D was included in our study instead.

As seen in Fig. B.11 and Fig. B.12, our method generally produces more plausible motion given the underlying geometry in a faster manner. The evaluation of Consistent4D take on average 32 minutes per object in its low VRAM setup, while our motion fitting takes on average under 3 minutes on an NVIDIA RTX 3090. Note that we exclude the time it takes to generate the driving videos in both cases. Consistent4D's slower runtime can be explained by its adoption of SDS which necessitates encoding the rendered RGB image into the latent space repeatedly. In contrast, our method remains in the semantic feature space.

Consistent4D utilizes a K-Plane [72] for their 4D representation which allows for a higher flexibility when modeling geometry distortions produced by the VDM. One such example can be seen in the raptor sequence in Fig. B.11. Here, the raptor's right and left legs rapidly alternate between the foreground and background, creating a sense of motion, but disrupting the raptor's underlying topology. While the ability to fit such non-physical effects can be beneficial in some scenarios, our explicit representation is more robust to fitting to VDM artifacts and thus reduces the consequent visually implausible transformations.

Furthermore, the K-plane representation entangles shape and motion and hence it cannot be easily integrated into common computer-graphics pipelines. This is in contrast to our method which deforms an explicit canonical mesh, where the time-dependent vertex deformations can be easily exported.

Lastly, Consistent4D (as well as DG4D) adopts an image-to-3D model Zero-1-to-3 [188] for 3D-Uplifting. However, Zero-1-to-3 requires input images without background which contradicts our observations that VDMs benefit from context in the background for better results (see in Appendix B.4.1). To tackle this, Consistent4D creates masks in an automated fashion for videos with background which can, however, introduce additional errors. In comparison, our method does not rely on any such masks.

B.4.3 Comparing Semantic Featuring against RGB for Optimization

Here, we complement our Pose estimation experiment from the main paper and compare the RGB and semantic features end-to-end in our full pipeline We find that modeling temporal deformations with NJF in combination with RGB features results in unstable optimization. Therefore, we only showcase the kinematic models in Fig. B.13. Similarly to pose fitting, semantic features lead to superior results.

B.4.4 Ablation of Number of Vertices

Fig. B.14 and Fig. B.15 show additional results when varying the number of vertices in our method with NJF. We find that the output quality degrades gracefully and predictably, when scaling down from 4000 to 500 vertices. Notice that the VDM output slightly differs, due to the variations in the conditioning input image when varying the number of vertices.

B.4.5 Results with Stable Video Diffusion

As reported in Appendix B.4.1, SVD produces camera motion rather than object motion. For completeness sake, we show that our method produces plausible results in Fig. B.16 when fitting to the SVD semantic features.

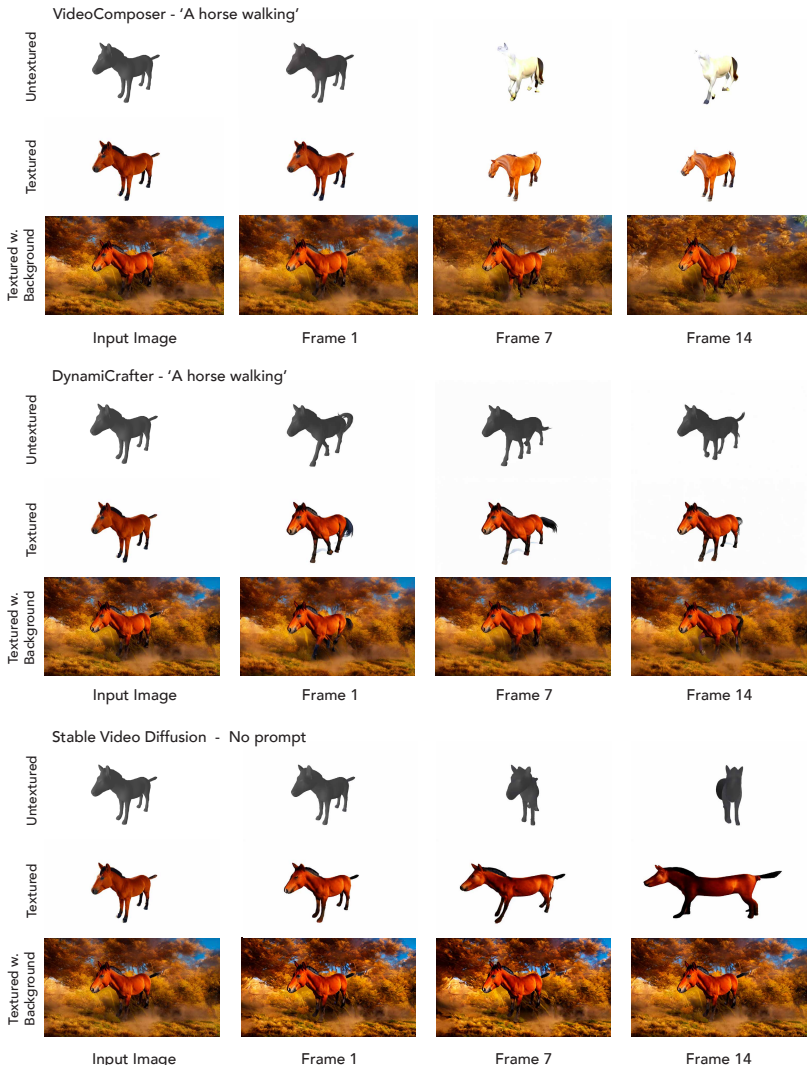


Figure B.9: Comparison of 3 output video frames (columns 2–4) for 3 VDMs considered for our experiments given the same *Horse* 3D mesh (1st column) rendered (from top to bottom) as an untextured shaded image, single-view textured image and a single-view textured image with a synthesized background **B** (Sec. 3.4.1 for details of the texturing process).

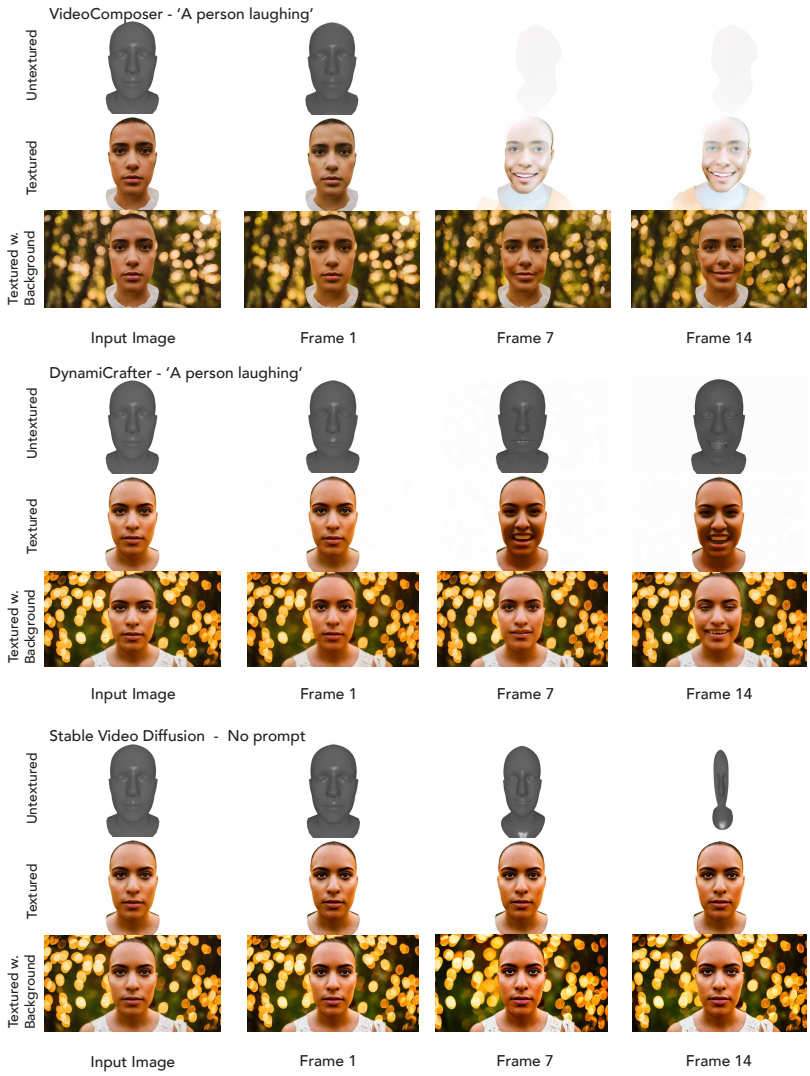


Figure B.10: Comparison of 3 output video frames (columns 2–4) for 3 VDMs considered for our experiments given the same FLAME 3D mesh (1st column) rendered (from top to bottom) as an untextured shaded image, single-view textured image and a single-view textured image with a synthesized background **B** (Sec. 3.4.1 for details of the texturing process).

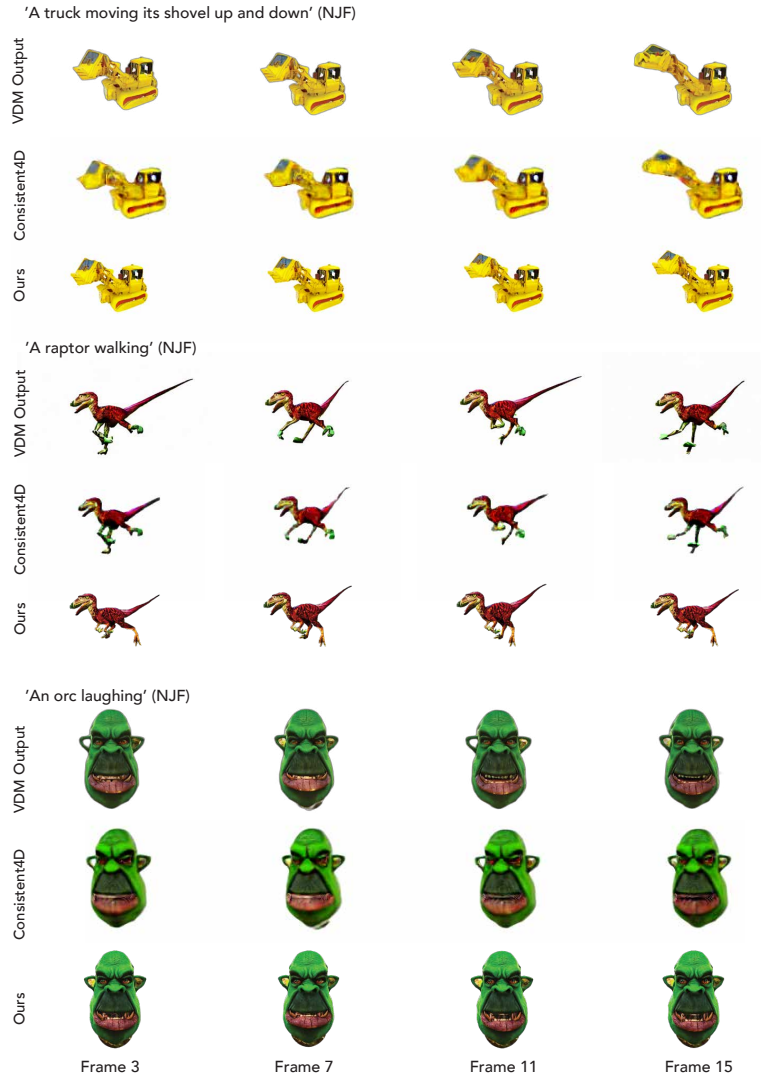


Figure B.11: Comparing Consistent4D against our method.

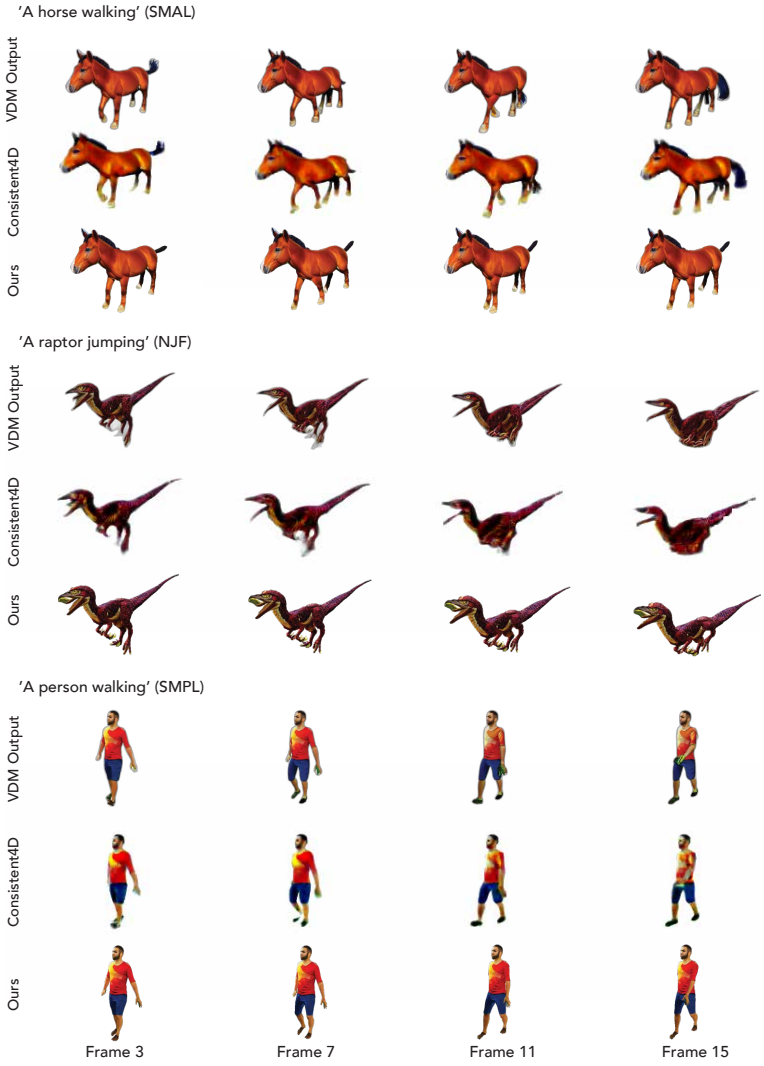


Figure B.12: Comparing Consistent4D against our method.

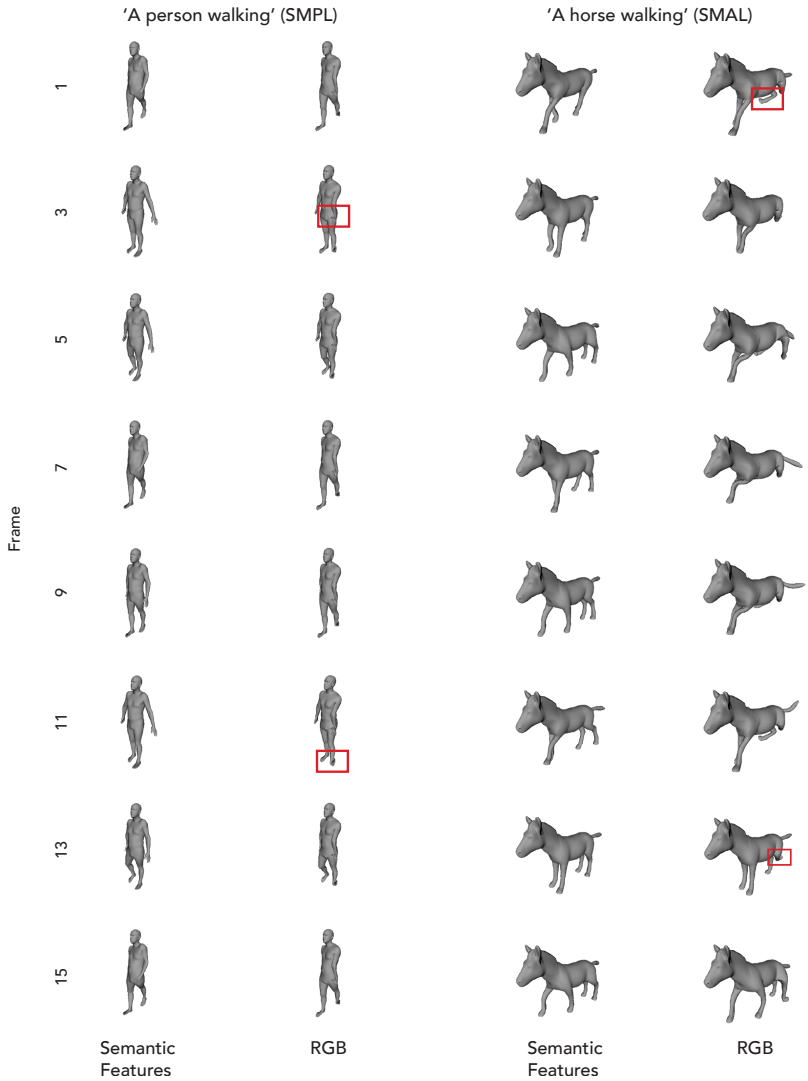


Figure B.13: Comparison of semantic features against RGB used for pose optimization. Each column shows frames for a single sequence. Note the red boxes, highlighting errors in the the pose optimization when utilizing RGB: 1) In case of SMPL (human), the hand gets stuck in front of the torso, as the RGB features do not distinguish the body from the hand. 2) In case of SMAL (horse), the limbs of the horse assume less realistic articulation with RGB features.

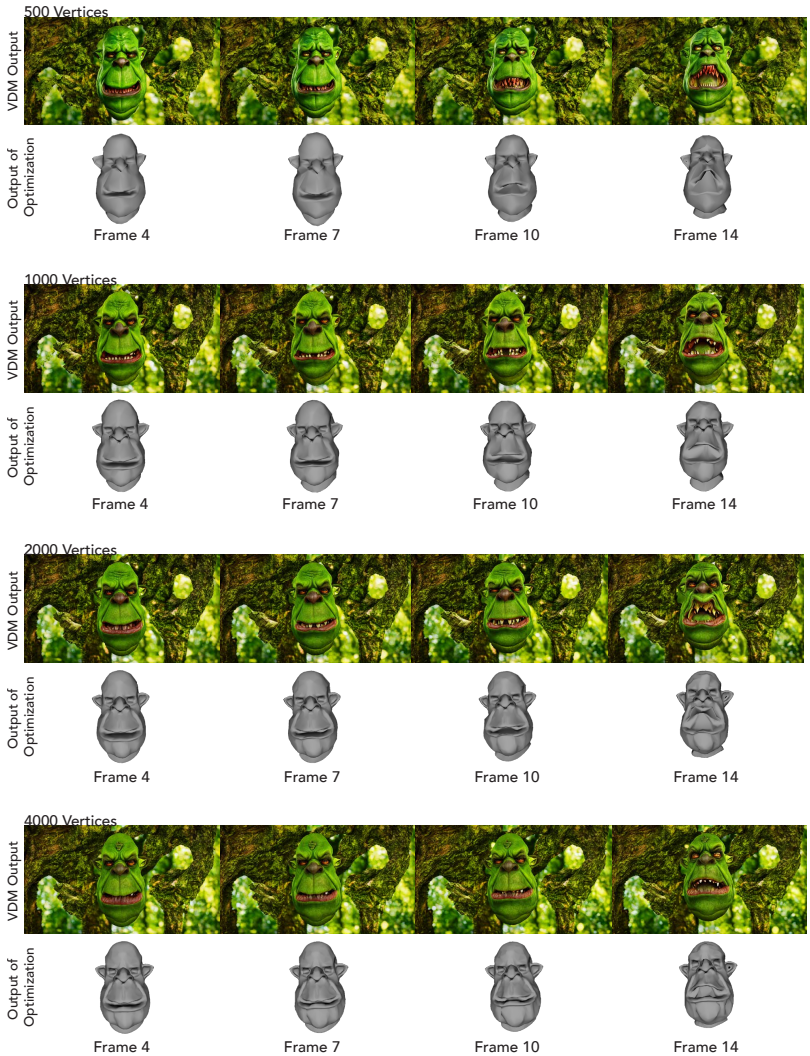


Figure B.14: Effect of vertex number on our method when adopting NJF for deformations with VC as VDM backbone. Each row shows the output of our method with an increasing number of vertices. Prompt: 'An orc laughing.'

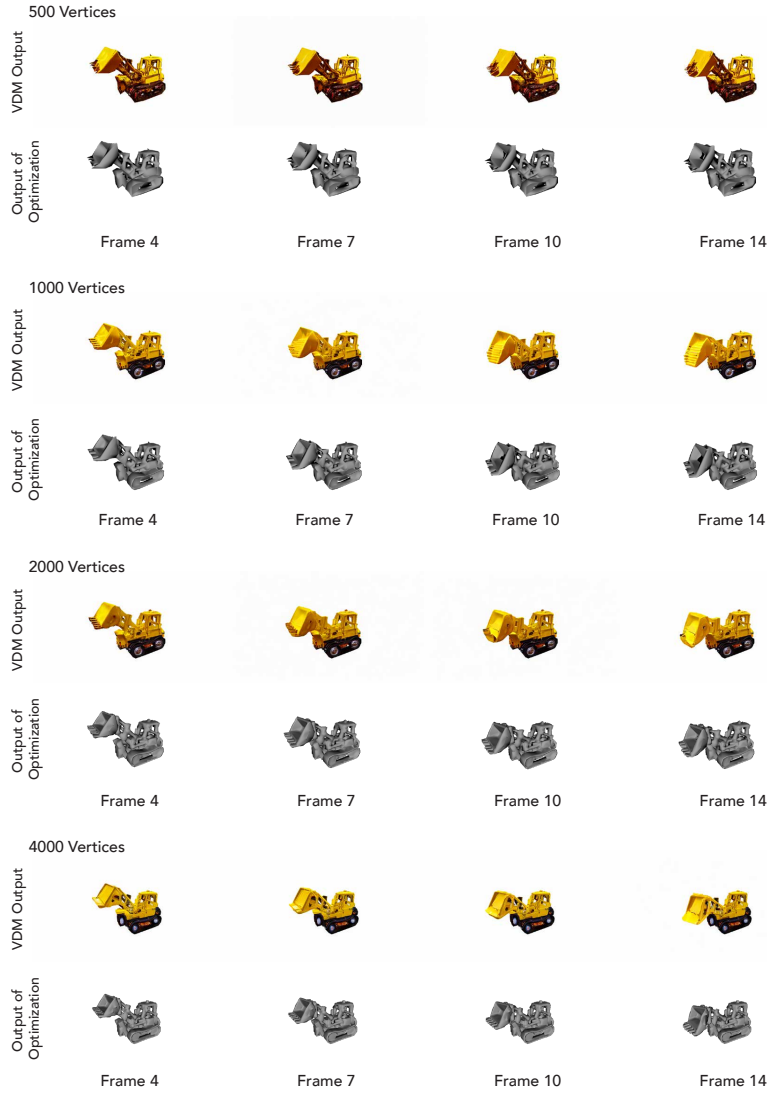


Figure B.15: Effect of vertex number on our method when adopting NJF for deformations with DC as VDM backbone. Each row shows the output of our method with an increasing number of vertices. Prompt: 'A truck moving its shovel up and down.'

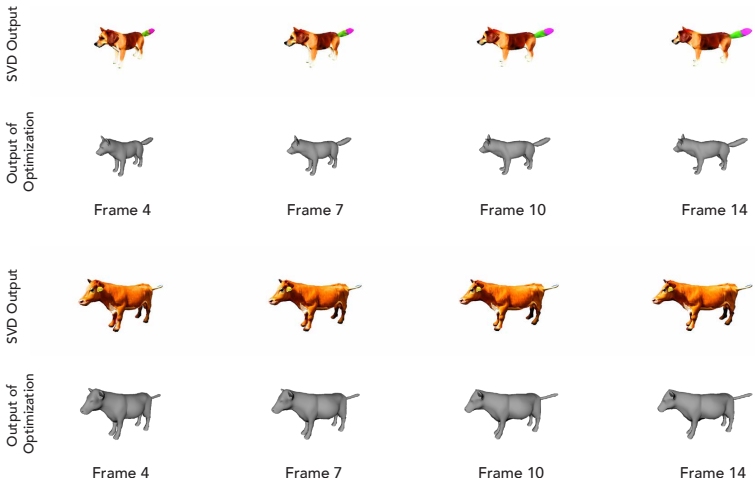


Figure B.16: Results when motion fitting using the SVD semantic features. While our method is well suited to work even with SVD features, SVD videos tend to focus on a global camera rotation rather than actual object motion. Consequently, the fitted object motion is often minimal or uninteresting. As a result we omitted SVD in our further studies in favor of other VDMs.

Appendix - Surface-Aware Distilled 3D Semantic Features

C

C.1 Additional implementation details

Here, we provide additional details to reproduce our experiments.

C.1.1 Diff3F

We use the authors' code to compute the Diff3F features [147] for our *base features* and as a baseline method for our comparisons.

Since we observed that the camera poses used for sampling in the *TOSCA* dataset are biased towards a specific up-direction, we modified the code to flip the coordinate system for only this dataset. Doing so yields $\approx 10\%$ increase in correspondence accuracy in *TOSCA* for both our method and the Diff3F baseline, when compared to the numbers reported in the Diff3F paper [147]. An alternative solution could be a modification of the camera sampling algorithm itself.

C.1.2 Ablations

Number of Anchors We train our method for different anchor counts A with a constant two-hour training budget per model. In Fig. C.1, we observe a low sensitivity to the anchor count. Due to repeated random sampling over the course of training, even $A = 1$ outperforms the correspondence error of Diff3F. Ultimately, we opt for $A = 100$ in all our experiments, as it balances computation cost and matching performance well. Note that neither the parameter A nor any computation of geodesic distances in generally are used during inference.

Losses Utilizing the geodesic distance as a supervision signal for embeddings has been explored in related works [217, 221]. However, in our ablations, we show that we achieve superior results with our formulation. In this section, we discuss the key differences.

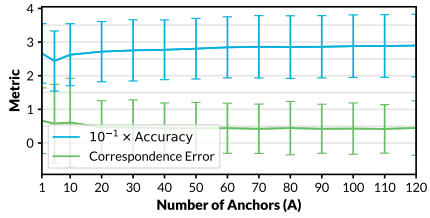


Figure C.1: Ablation on the anchor count A on *SHREC'19*. In terms of the correspondence error, our method improves upon Diff3F already starting from $A = 1$.

Relative Geodesic Loss (RGL) Based on two points, v_p and v_q, the Relative Geodesic Loss [221] minimizes the difference between the geodesic distance d^S and the *Euclidean* embedding distance d^E of those two points, relative to the surface distance:

$$L_{RGL} = \sum_i \sum_{(p,q)\in S_i} \frac{|d_i^E(v_p, v_q) - d^S(v_p, v_q)|^2}{d^S(v_p, v_q)^2}. \tag{C.1}$$

The normalization term is introduced to prioritize local distance preservation. We do not utilize this normalization term, because the *base features* struggle to disambiguate samples that are far away on the surface but close in feature space, such as "left hand" and "right hand".

***Naive* Geodesic Loss (NGL)** *NGL* is discussed by Jiang et al. [221] but not used for training, as the authors state that it might hamper local distance preservation. Indeed, in our ablations, it achieves worse results in terms of correspondence accuracy when compared to *RGL*. It is identical to *RGL* but it omits the normalization term:

$$L_{NGL} = \sum_i \sum_{(p,q)\in S_i \in [n_i]^2} |d_i^E(v_p, v_q) - d^S(v_p, v_q)|^2 \tag{C.2}$$

While not actually utilized in their work, the *NGL* formulation is the closest of the three to our own formulation. However, our choice of a hyperspherical rather than Euclidean embedding space in combination with our autoencoder setup achieves notably better results.

Geometric Similarity Loss (GSL) A concurrent work proposes to maximize the *local angular similarity* between geodesic distances and *Euclidean* feature vectors for a set of neighbors for each point [217]. Crucially, the similarity is only maximized for a set of k point pairs nearest in the embedding space. The loss is a cosine metric between a vector of geodesic distances $\mathbf{m}_i \in \mathbb{R}^k$ and a vector of *Euclidean* embedding distances $\mathbf{d}_i \in \mathbb{R}^k$:

$$\mathscr{L}_{\text{GSL}} = \frac{1}{N} \sum_{i=1}^{N} \left(1 - \frac{\mathbf{d}_i \cdot \mathbf{m}_i}{\|\mathbf{d}_i\| \|\mathbf{m}_i\|}\right). \tag{C.3}$$

This restricts *GSL* supervision to a fixed neighborhood size and potentially limits disambiguation of features that are close in feature space but not among the k nearest neighbors. In contrast, our method follows a global approach by sampling anchors based on a furthest point sampling.

Conclusion Our method differs from recent and concurrent works utilizing vision-based features for 3D shape matching in three main aspects: 1) we follow a global approach when enforcing distances in the embedding space; 2) our embedding space is hyperspherical and it only encodes angular information; 3) in the context of the whole pipeline, we solely rely on intrinsic properties.

C.2 One Shot Pose Transfer

We train an MLP to model the deformation between the paired input source mesh $\mathcal{M}^{src}_{train}$ and the output target mesh $\mathcal{M}^{tgt}_{train}$ obtained from $SMPL^{eval}$ and thus not used for training of our features. The paired training meshes share the same identity β but they differ in poses θ such that

$$\mathcal{M}^{src}_{train} := SMPL(\beta_{src}, \theta_{src})$$
$$\mathcal{M}^{tgt}_{train} := SMPL(\beta_{src}, \theta_{tgt}).$$

We train an MLP $\mathcal{M}^{tgt}_{train} = \mathcal{F}(\mathcal{M}^{src}_{train}, \mathcal{S}^{src}_{\mathcal{M}})$ to produce the target pose mesh $\mathcal{M}^{tgt}_{train}$ conditioned on the source mesh $\mathcal{M}^{src}_{train}$ and its *surface-aware features* $\mathcal{S}^{src}_{\mathcal{M}}$. These features are produced by our pre-trained general encoder $\mathcal{E}(.)$ from Sec. 4.5 without any further fine-tuning. Alternatively, other features are used for comparisons.

During training, $\mathcal{F}(.)$ learns a *per-face* residual function [293] to model a Neural Jacobian Field [119] while being supervised by MSE between the predicted re-posed mesh and the target $\mathcal{M}^{tgt}_{train}$.

Crucially, $\mathcal{F}(.)$ is defined on a per-face basis, which means that the input meshes used in test time do not need to have the same connectivity as the training mesh pair. In quantitative comparisons, we apply the learned mapping $\mathcal{F}(.)$ to input meshes of unseen identities $\mathcal{M}^{src}_{test} := SMPL(\beta_{test}, \theta_{src})$ and compare against their ground truths $\mathcal{M}^{tgt}_{test} := SMPL(\beta_{test}, \theta_{tgt})$. For other shape classes, we provide qualitative comparisons.

C.3 Pose Alignment

We establish correspondences between two input shapes based on the feature cosine similarity $\phi(.)$ (Sec. 4.3), such that each point $\mathbf{x}^{\mathcal{S}}_i$ in the source shape is assigned a target point $\mathbf{x}^{\mathcal{T}}_i$ in the target shape. Next, we align the source to the target by minimizing the following L1 loss:

$$\mathcal{L}_{point} = \frac{1}{N}\sum_{i}^{N} ||\mathbf{x}^{\mathcal{S}}_i - \mathbf{x}^{\mathcal{T}}_i||_1. \tag{C.4}$$

For the first half of the optimization steps, we only optimize the root rotation $\mathbf{R}$, the root translation $\mathbf{t}$, and the scale s, which roughly rigidly aligns the meshes. In the second half, we additionally optimize the rotation $\mathbf{R}_b$ of each kinematic bone b. The parameters are optimized based on a gradient-descent for 4000 iterations, which takes approximately 30 seconds for a static pose.

Furthermore, we found it beneficial to use an as-rigid-as-possible regularization [292], which penalizes the deviation between the initial edge lengths of the mesh δ^{init}_e and the current edge length δ_e for each edge e:

$$\mathcal{L}_{arap} = \frac{1}{E}\sum_{e}^{E} |\delta^{init}_e - \delta_e|. \tag{C.5}$$

When fitting an animation as a pose sequence, we optimize the pose parameters for each time step t. Furthermore, we apply pointwise temporal smoothing for neighboring frames:

$$\mathcal{L}_{smooth} = \frac{1}{N(T-1)} \sum_{t}^{T-1} \sum_{i}^{N} ||\mathbf{x}_{i,t}^{S} - \mathbf{x}_{i,t+1}^{S}||_2^2. \tag{C.6}$$

The final loss is $\mathcal{L}_{pose} = w_p\mathcal{L}_{point} + w_a\mathcal{L}_{arap} + w_s\mathcal{L}_{smooth}$ with $w_p = w_a = w_s = 1$ for animations and $w_s = 0$ otherwise.

C.4 Additional results

C.4.1 Qualitative Results on SHREC'20

Fig. C.2 presents additional results for the *SHREC'20* dataset. As the dataset only provides ≈ 50 correspondences for each shape pair, we display the predicted correspondences without dense ground-truth labels. However, we find that our features generally produce smoother correspondences (e.g., bottom left) and a better separation of legs (e.g., the second to last row on the right).

C.4.2 Qualitative Results on ShapeNet

In Fig. C.3, we show additional results for chairs and airplanes from ShapeNet [283]. Since no dense ground truth labels are available, we show the predicted correspondences alone. We find that our *surface-aware features* achieve results better than the Diff3F baseline when separating the chair legs (see the top left row) and the airplane wings (see the top right row).

C.4.3 Texturing

We provide additional examples of 2D-to-3D texturing based on our own features in Fig. C.4.

Table C.1: Correspondence metrics measured for human and animal test shapes. The *Specialized encoder* was trained on a join set of humans and animals following our point-to-point correspondence experiment procedure in Sec. 4.5.4, and thus, the values match Table 4.1 in the paper (see *Ours*). *All-shape encoder* was trained on a larger more generalized training set covering humans and animals but also chairs and airplanes.

Encoder		SHREC'19	TOSCA	SHREC'20
Specialized	err ↓	**0.43±0.76**	1.65±2.15	**23.89±8.90**
	acc ↑	**28.78±9.30**	**29.35±14.53**	**73.97±26.47**
All-shape	err ↓	0.56±1.03	**1.62±2.08**	4.37±9.47
	acc ↑	27.53±9.47	27.45±14.56	70.33±24.86

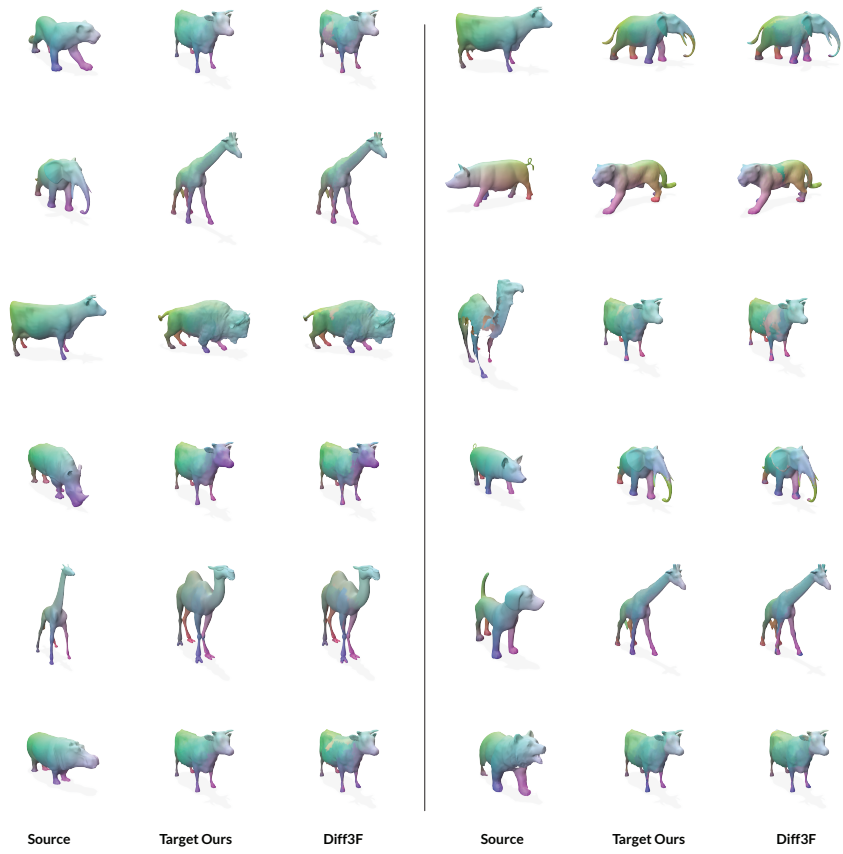


Figure C.2: Qualitative results on *SHREC'20*. Our features result on average in a more accurate smooth mapping between different limbs.

C.4.4 Using a Shared Encoder for All Shapes

In Sec. 4.5, we train a shared encoder for (*SURREAL*) and animals (*SMAL*) shapes as well as separate encoders for chair and airplanes (*ShapeNet*). Here, we follow the same procedure and train a new single shared encoder on a union of all these shapes and test it on human and animal shapes as in the paper Table 4.1 to assess further generalization of our approach. The correspondence metrics in Table C.1 show that the all-shape encoder generally slightly under-performs the specialized encoders but it still improves upon the baselines (see Table 4.1 in the paper). *TOSCA* is an exception, as the larger combined train set marginally reduces the error.

C.4.5 Ablating the Number of Training Shapes

In Table C.2, we explore how the number of training shapes affects the correspondence error with the same fixed validation set and training policy as in the main experiments. We vary the number of training samples while retaining a constant animal-to-human shape ratio. We find that just 2 training samples already decrease the error when compared to

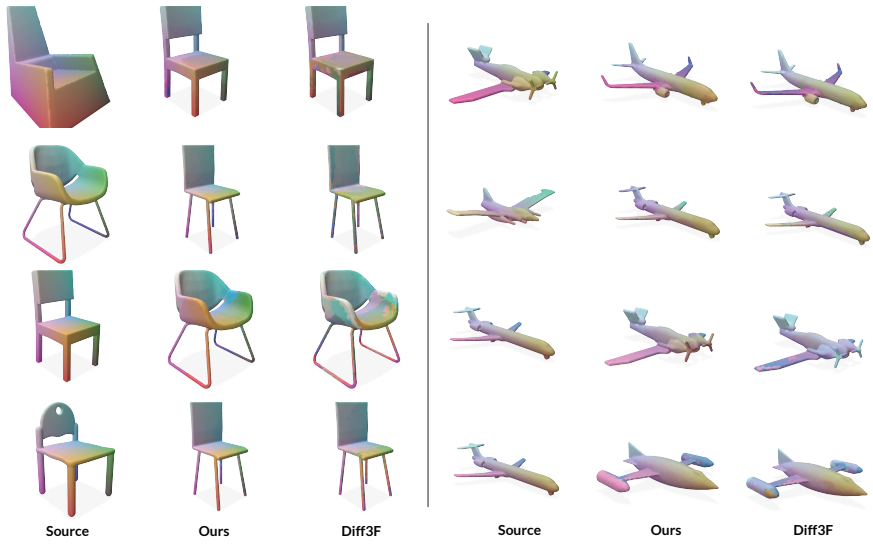


Figure C.3: Qualitative results on ShapeNet [283]. Our features result on average in a more accurate smooth mapping between chair legs and wings.

Table C.2: Training set size (columns) vs. correspondence error.

	Diff3f	2	10	50	94
SHREC'19	1.69±1.44	1.32±1.22	1.31±1.31	0.48±0.85	0.43±0.76
TOSCA	4.51±5.48	3.75±3.50	2.60±2.74	1.84±2.47	1.65±2.15
SHREC'20	5.34±10.22	3.89±8.49	4.05±9.90	3.96±9.33	3.89±8.90

Diff3F. As expected, additional samples lead to further improvements for SHREC'19 and TOSCA. This trend is more subtle for SHREC'20.



Figure C.4: Texturing of 3D meshes from *SMPL*^{eval} and *TOSCA*, based on a 2D image generated with ChatGPT. The appearance is transferred by establishing correspondence between the image features and 3D mesh features. Our method performs well even on incomplete meshes (notice the bear in the second row).

D Appendix - Rotation-Equivariant Neural Jacobian Fields for 3D Mesh Deformations

D.1 Poisson equation with extrinsic gradient operator

In our implementation, we use an extrinsic gradient operator. Here, we provide the derivation of corresponding Poisson equation. For convenience, we set up a coordinate transform matrix that composes the local coordinate systems in a $3F \times 3F$ block-diagonal matrix.

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_i & & \\ & \ddots & \\ & & \mathbf{B}_N \end{bmatrix}, \tag{D.1}$$

where $\mathbf{B}$ is orthonormal by construction, i.e., $\mathbf{B}^\top \mathbf{B} = \mathbf{B}\mathbf{B}^\top = \mathbf{I}$.

We seek to rewrite the following Poisson equation to the ambient coordinate system

$$\mathbf{L}\mathbf{V}' = \mathbf{G}^\top \mathbf{A}\, \mathbf{J}'^\top, \tag{D.2}$$

where $\mathbf{G}$ is a $2F \times V$ matrix, $\mathbf{J}'$ a $3 \times 2F$ matrix and $\mathbf{A}$ is a $2F \times 2F$ diagonal matrix with twice-repeated triangle areas.

We can write the extrinsic gradient using a coordinate transform. The intrinsic gradient outputs tangent vectors, resulting in the $2F$ row dimension. For ease of notation, we add a row/column of zeros for the normal direction to both the gradient and Jacobian matrices, such that $\mathbf{G}$ is of size $3F \times V$, $\mathbf{J}$ is of size $3 \times 3F$ and $\mathbf{A}$ is of size $3F \times 3F$. Then, we can write the following identities

$$\mathbf{G}_{\text{ext}} = \mathbf{B}\mathbf{G}, \tag{D.3}$$

$$\mathbf{G} = \mathbf{B}^\top \mathbf{G}_{\text{ext}}. \tag{D.4}$$

These identities let us rewrite the Poisson equation in Equation D.2

$$\mathbf{L}\mathbf{V}' = \mathbf{G}_{\text{ext}}^\top \mathbf{B}\mathbf{A}\mathbf{J}'^\top. \tag{D.5}$$

Note that $\mathbf{J}'$ is a composition of per-face Jacobians computed as $\mathbf{J}'_i = \mathbf{B}_i d_\theta(\mathbf{c}_i)$. We factor out $\mathbf{B}$ by setting $\tilde{\mathbf{J}}'_i = d_\theta(\mathbf{c}_i)$ and $\mathbf{J}'_i = \mathbf{B}_i \tilde{\mathbf{J}}'_i$. Equation D.5 can then be written as

$$\mathbf{L}\mathbf{V}' = \mathbf{G}_{\text{ext}}^\top \mathbf{A}\mathbf{B}\tilde{\mathbf{J}}'^\top \mathbf{B}^\top, \tag{D.6}$$

which is the equation used in our implementation. $\tilde{\mathbf{J}}'$ and the underlying neural network d_θ operates entirely in the local coordinate frame, as it is wedged between $\mathbf{B}$ and $\mathbf{B}^\top$, resulting in the desired rotation-equivariance property.

Bibliography

[1] Lukas Uzolas, Elmar Eisemann, and Petr Kellnhofer. Template-free articulated neural point clouds for reposable view synthesis. *Advances in Neural Information Processing Systems*, 36: 31621–31637, 2023.

[2] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. volume 65, pages 99–106. ACM New York, NY, USA, 2021.

[3] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Wang Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. In *Computer Graphics Forum*, volume 41, pages 703–735. Wiley Online Library, 2022.

[4] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10318–10327, 2021.

[5] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6498–6508, 2021.

[6] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *ICCV*, 2021.

[7] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5865–5874, 2021.

[8] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *CVPR*, pages 9421–9431, 2021.

[9] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021.

[10] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph. (SIGGRAPH Asia)*, 40(6), 2021.

[11] Tianye Li, Mira Slavcheva, Michael Zollhöfer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5521–5531, June 2022.

[12] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022.

[13] John P Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 165–172, 2000.

[14] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *ACM SIGGRAPH*, page 187–194, 1999.

[15] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a model

of facial shape and expression from 4D scans. *ACM Trans. Graph. (SIGGRAPH Asia)*, 36(6): 194:1–194:17, 2017.

[16] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (SIGGRAPH Asia)*, 34(6): 248:1–248:16, 2015.

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

[18] Gengshan Yang, Deqing Sun, Varun Jampani, Daniel Vlasic, Forrester Cole, Huiwen Chang, Deva Ramanan, William T Freeman, and Ce Liu. Lasr: Learning articulated shape reconstruction from a monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15980–15989, 2021.

[19] Gengshan Yang, Deqing Sun, Varun Jampani, Daniel Vlasic, Forrester Cole, Ce Liu, and Deva Ramanan. Viser: Video-specific surface embeddings for articulated 3d shape reconstruction. *Advances in Neural Information Processing Systems*, 34:19326–19338, 2021.

[20] Chun-Han Yao, Wei-Chih Hung, Yuanzhen Li, Michael Rubinstein, Ming-Hsuan Yang, and Varun Jampani. Lassie: Learning articulated shapes from sparse image ensemble via 3d part discovery. *Advances in Neural Information Processing Systems*, 35:15296–15308, 2022.

[21] Atsuhiro Noguchi, Umar Iqbal, Jonathan Tremblay, Tatsuya Harada, and Orazio Gallo. Watch it move: Unsupervised discovery of 3d joints for re-posing of articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3677–3687, 2022.

[22] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022.

[23] Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. Banmo: Building animatable 3d neural models from many casual videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2863–2873, 2022.

[24] Yuefan Wu, Zeyuan Chen, Shaowei Liu, Zhongzheng Ren, and Shenlong Wang. Casa: Category-agnostic skeletal animal reconstruction. In *Advances in Neural Information Processing Systems*, volume 35, pages 28559–28574, 2022.

[25] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

[26] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[27] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019.

[28] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4743–4752, 2019.

[29] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5939–5948, 2019.

[30] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2565–2574, 2020.

[31] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning*, pages 3789–3799. PMLR, 2020.

[32] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6001–6010, 2020.

[33] Thomas Davies, Derek Nowrouzezahrai, and Alec Jacobson. Overfit neural networks as a compact shape representation. *arXiv preprint arXiv:2009.09808*, 2020.

[34] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *European Conference on Computer Vision*, pages 608–625. Springer, 2020.

[35] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *european conference on computer vision*, pages 523–540. Springer, 2020.

[36] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11358–11367, 2021.

[37] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in neural information processing systems*, volume 33, pages 7462–7473, 2020.

[38] Julien NP Martel, David B Lindell, Connor Z Lin, Eric R Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: adaptive coordinate networks for neural scene representation. *ACM Transactions on Graphics (TOG)*, 40(4):1–13, 2021.

[39] Petr Kellnhofer, Lars C Jebe, Andrew Jones, Ryan Spicer, Kari Pulli, and Gordon Wetzstein. Neural lumigraph rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4287–4297, 2021.

[40] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1251–1261, 2020.

[41] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33:2492–2502, 2020.

[42] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in neural information processing systems*, volume 32, 2019.

[43] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2437–2446, 2019.

[44] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2304–2314, 2019.

[45] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3d supervision. *Advances in Neural Information Processing Systems*, 32, 2019.

[46] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: learning dynamic renderable volumes from images. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[47] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3504–3515, 2020.

[48] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2019–2028, 2020.

[49] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7210–7219, 2021.

[50] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7495–7504, 2021.

[51] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[52] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H Mueller, Chakravarty R Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. 40(4):45–59, 2021.

[53] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. volume 33, pages 7537–7547, 2020.

[54] David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14556–14565, 2021.

[55] Terrance DeVries, Miguel Angel Bautista, Nitish Srivastava, Graham W Taylor, and Joshua M Susskind. Unconstrained scene generation with locally conditioned radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14304–14313, 2021.

[56] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5589–5599, 2021.

[57] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14346–14355, 2021.

[58] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. In *Computer graphics forum*, volume 39, pages 701–727. Wiley Online Library, 2020.

[59] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14335–14345, 2021.

[60] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14153–14161, 2021.

[61] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene

neural view synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8248–8258, 2022.

[62] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5875–5884, 2021.

[63] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5752–5761, 2021.

[64] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4): 1–15, 2022.

[65] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022.

[66] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020.

[67] Julian Ost, Issam Laradji, Alejandro Newell, Yuval Bahat, and Felix Heide. Neural point light fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18419–18429, 2022.

[68] Jad Abou-Chakra, Feras Dayoub, and Niko Sünderhauf. Particlenerf: A particle-based encoding for online neural radiance fields. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5975–5984, 2024.

[69] Ruofan Liang, Jiahao Zhang, Haoda Li, Chen Yang, and Nandita Vijaykumar. Spidr: Sdf-based neural point fields for illumination and deformation. *arXiv preprint arXiv:2210.08398*, 2022.

[70] Yufeng Zheng, Wang Yifan, Gordon Wetzstein, Michael J Black, and Otmar Hilliges. Pointavatar: Deformable point-based head avatars from videos. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21057–21067, 2023.

[71] Shengze Wang, Alexey Supikov, Joshua Ratcliff, Henry Fuchs, and Ronald Azuma. Inv: Towards streaming incremental neural videos. *arXiv preprint arXiv:2302.01532*, 2023.

[72] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023.

[73] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023.

[74] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Trans. Graph. (SIGGRAPH Asia)*, 36(6), November 2017.

[75] Silvia Zuffi, Angjoo Kanazawa, David W Jacobs, and Michael J Black. 3d menagerie: Modeling the 3d shape and pose of animals. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6365–6373, 2017.

[76] Guy Gafni, Justus Thies, Michael Zollhofer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8649–8658, 2021.

[77] Ziyan Wang, Timur Bagautdinov, Stephen Lombardi, Tomas Simon, Jason Saragih, Jessica Hodgins, and Michael Zollhofer. Learning compositional radiance fields of dynamic human heads. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5704–5713, 2021.

[78] Yudong Guo, Keyu Chen, Sen Liang, Yong-Jin Liu, Hujun Bao, and Juyong Zhang. Ad-nerf: Audio driven neural radiance fields for talking head synthesis. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5784–5794, 2021.

[79] Yufeng Zheng, Victoria Fernández Abrevaya, Marcel C. Bühler, Xu Chen, Michael J. Black, and Otmar Hilliges. I M Avatar: Implicit morphable head avatars from videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[80] Kacper Kania, Kwang Moo Yi, Marek Kowalski, Tomasz Trzciński, and Andrea Tagliasacchi. Conerf: Controllable neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18623–18632, 2022.

[81] Yiyu Zhuang, Hao Zhu, Xusen Sun, and Xun Cao. Mofanerf: Morphable facial neural radiance field. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*, pages 268–285. Springer, 2022.

[82] ShahRukh Athar, Zexiang Xu, Kalyan Sunkavalli, Eli Shechtman, and Zhixin Shu. Rignerf: Fully controllable neural 3d portraits. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20364–20373, 2022.

[83] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *CVPR*, 2021.

[84] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14314–14323, 2021.

[85] Shih-Yang Su, Frank Yu, Michael Zollhöfer, and Helge Rhodin. A-nerf: Articulated neural radiance fields for learning human shape, appearance, and pose. *Advances in neural information processing systems*, 34:12278–12291, 2021.

[86] Hongyi Xu, Thiemo Alldieck, and Cristian Sminchisescu. H-nerf: Neural radiance fields for rendering and temporal reconstruction of humans in motion. *Advances in Neural Information Processing Systems*, 34:14955–14966, 2021.

[87] Atsuhiro Noguchi, Xiao Sun, Stephen Lin, and Tatsuya Harada. Neural articulated radiance field. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5762–5772, 2021.

[88] Youngjoong Kwon, Dahun Kim, Duygu Ceylan, and Henry Fuchs. Neural human performer: Learning generalizable radiance fields for human performance rendering. *Advances in Neural Information Processing Systems*, 34:24741–24752, 2021.

[89] Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields for interactive free-viewpoint video. In *SIGGRAPH Asia 2022 conference papers*, pages 1–9, 2022.

[90] Tao Hu, Tao Yu, Zerong Zheng, He Zhang, Yebin Liu, and Matthias Zwicker. Hvtr: Hybrid volumetric-textural rendering for human avatars. In *2022 International Conference on 3D Vision (3DV)*, pages 197–208. IEEE, 2022.

[91] Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. Neural actor: Neural free-view synthesis of human actors with pose control. *ACM Trans. Graph.(ACM SIGGRAPH Asia)*, 2021.

[92] Tianhan Xu, Yasuhiro Fujita, and Eiichi Matsumoto. Surface-aligned neural radiance fields for controllable 3d human synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15883–15892, 2022.

[93] Fuqiang Zhao, Wei Yang, Jiakai Zhang, Pei Lin, Yingliang Zhang, Jingyi Yu, and Lan Xu. Humannerf: Efficiently generated human radiance field from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7743–7753, 2022.

[94] Chung-Yi Weng, Brian Curless, Pratul P Srinivasan, Jonathan T Barron, and Ira Kemelmacher-Shlizerman. Humannerf: Free-viewpoint rendering of moving people from monocular video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern Recognition*, pages 16210–16220, 2022.

[95] Boyi Jiang, Yang Hong, Hujun Bao, and Juyong Zhang. Selfrecon: Self reconstruction your digital avatar from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5605–5615, 2022.

[96] Xu Chen, Yufeng Zheng, Michael J Black, Otmar Hilliges, and Andreas Geiger. Snarf: Differentiable forward skinning for animating non-rigid neural implicit shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11594–11604, 2021.

[97] Ruilong Li, Julian Tanke, Minh Vo, Michael Zollhöfer, Jürgen Gall, Angjoo Kanazawa, and Christoph Lassner. Tava: Template-free animatable volumetric actors. In *European Conference on Computer Vision*, pages 419–436. Springer, 2022.

[98] Yihao Zhi, Shenhan Qian, Xinhao Yan, and Shenghua Gao. Dual-space nerf: Learning animatable avatars and scene lighting in separate spaces. In *2022 International Conference on 3D Vision (3DV)*, pages 1–10. IEEE, 2022.

[99] Yang Hong, Bo Peng, Haiyao Xiao, Ligang Liu, and Juyong Zhang. Headnerf: A real-time nerf-based parametric head model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20374–20384, 2022.

[100] Alexander W Bergman, P Kellnhofer, Wang Yifan, Eric R Chan, David B Lindell, and Gordon Wetzstein. Generative neural articulated radiance fields. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022): NeurIPS Proceedings*. 2022.

[101] Atsuhiro Noguchi, Xiao Sun, Stephen Lin, and Tatsuya Harada. Unsupervised learning of efficient geometry-aware neural articulated representations. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII*, pages 597–614. Springer, 2022.

[102] Suyi Jiang, Haoran Jiang, Ziyu Wang, Haimin Luo, Wenzheng Chen, and Lan Xu. Humangen: Generating human radiance fields with explicit priors. pages 12543–12554, 2023.

[103] Hsiao-yu Chen, Edith Tretschk, Tuur Stuyck, Petr Kadlecek, Ladislav Kavan, Etienne Vouga, and Christoph Lassner. Virtual elastic objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15827–15837, 2022.

[104] Shangzhe Wu, Ruining Li, Tomas Jakab, Christian Rupprecht, and Andrea Vedaldi. Magicpony: Learning articulated 3d animals in the wild. pages 8792–8802, 2023.

[105] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7122–7131, 2018.

[106] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2252–2261, 2019.

[107] Soubhik Sanyal, Timo Bolkart, Haiwen Feng, and Michael J Black. Learning to regress 3d face shape and expression from an image without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7763–7772, 2019.

[108] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 84–93, 2020.

[109] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5253–5263, 2020.

[110] Chun-Han Yao, Wei-Chih Hung, Yuanzhen Li, Michael Rubinstein, Ming-Hsuan Yang, and Varun Jampani. Hi-lassie: High-fidelity articulated shape and skeleton discovery from sparse image ensemble. pages 4853–4862, 2023.

[111] Tianhan Xu and Tatsuya Harada. Deforming radiance fields with cages. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*, pages 159–175. Springer, 2022.

[112] Kaizhi Yang, Xiaoshuai Zhang, Zhiao Huang, Xuejin Chen, Zexiang Xu, and Hao Su. Movingparts: Motion-based 3d part discovery in dynamic radiance field. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=QQ6RgKYiQq.

[113] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984.

[114] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 333–350. Springer, 2022.

[115] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[116] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 586–595, 2018.

[117] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[118] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Improved direct voxel grid optimization for radiance fields reconstruction. *arXiv preprint arXiv:2206.05085*, 2022.

[119] Noam Aigerman, Kunal Gupta, Vladimir G Kim, Siddhartha Chaudhuri, Jun Saito, and Thibault Groueix. Neural jacobian fields: learning intrinsic mappings of arbitrary meshes. *ACM Transactions on Graphics (TOG)*, 41(4):1–17, 2022.

[120] Lukas Uzolas, Elmar Eisemann, and Petr Kellnhofer. Motiondreamer: Exploring semantic video diffusion features for zero-shot 3d mesh animation. *© 2025 IEEE. Reprinted, with permission, from 2025 International Conference on 3D Vision (3DV)*, pages 893–904, 2025.

[121] Michael Gleicher, S Grassia, Zoran Popovic, S Rosenthal, and Jeffrey Thingvold. Motion editing, principles, practice and promise. In *SIGGRAPH 2000 Course Notes 26*, 2000.

[122] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[123] Xiang Wang, Hangjie Yuan, Shiwei Zhang, Dayou Chen, Jiuniu Wang, Yingya Zhang, Yujun Shen, Deli Zhao, and Jingren Zhou. Videocomposer: Compositional video synthesis with motion controllability. *Advances in Neural Information Processing Systems*, 36, 2024.

[124] Jinbo Xing, Menghan Xia, Yong Zhang, Haoxin Chen, Wangbo Yu, Hanyuan Liu, Gongye Liu, Xintao Wang, Ying Shan, and Tien-Tsin Wong. Dynamicrafter: Animating open-domain images with video diffusion priors. In *European Conference on Computer Vision*, pages 399–417. Springer, 2024.

[125] Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. In *The Twelfth International Conference on Learning Representations*, 2023.

[126] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

[127] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[128] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*.

[129] Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob Mcgrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *International Conference on Machine Learning*, pages 16784–16804. PMLR, 2022.

[130] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[131] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

[132] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[133] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2022.

[134] Hanhung Lee, Manolis Savva, and Angel X Chang. Text-to-3d shape generation. In *Computer Graphics Forum*, page e15061. Wiley Online Library, 2024.

[135] Uriel Singer, Shelly Sheynin, Adam Polyak, Oron Ashual, Iurii Makarov, Filippos Kokkinos, Naman Goyal, Andrea Vedaldi, Devi Parikh, Justin Johnson, et al. Text-to-4d dynamic scene generation. In *Proceedings of the 40th International Conference on Machine Learning*, pages 31915–31929, 2023.

[136] Huan Ling, Seung Wook Kim, Antonio Torralba, Sanja Fidler, and Karsten Kreis. Align your gaussians: Text-to-4d with dynamic 3d gaussians and composed diffusion models. pages 8576–8588, 2024.

[137] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. Dreamgaussian4d: Generative 4d gaussian splatting. *arXiv preprint arXiv:2312.17142*, 2023.

[138] Yuyang Yin, Dejia Xu, Zhangyang Wang, Yao Zhao, and Yunchao Wei. 4dgen: Grounded 4d content generation with spatial-temporal consistency. *arXiv preprint arXiv:2312.17225*, 2023.

[139] Xinzhou Wang, Yikai Wang, Junliang Ye, Fuchun Sun, Zhengyi Wang, Ling Wang, Pengkun Liu, Kai Sun, Xintong Wang, Wende Xie, et al. Animatabledreamer: Text-guided non-rigid 3d model generation and reconstruction with canonical score distillation. In *European Conference on Computer Vision*, pages 321–339. Springer, 2024.

[140] Sherwin Bahmani, Ivan Skorokhodov, Victor Rong, Gordon Wetzstein, Leonidas Guibas, Peter Wonka, Sergey Tulyakov, Jeong Joon Park, Andrea Tagliasacchi, and David B Lindell. 4d-fy: Text-to-4d generation using hybrid score distillation sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7996–8006, 2024.

[141] Yufeng Zheng, Xueting Li, Koki Nagano, Sifei Liu, Otmar Hilliges, and Shalini De Mello. A unified approach for text-and image-guided 4d scene generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7300–7309, 2024.

[142] Yanqin Jiang, Li Zhang, Jin Gao, Weiming Hu, and Yao Yao. Consistent4d: Consistent 360°

dynamic object generation from monocular video. In *The Twelfth International Conference on Learning Representations*, 2023.

[143] Yuyang Zhao, Zhiwen Yan, Enze Xie, Lanqing Hong, Zhenguo Li, and Gim Hee Lee. Animate124: Animating one image to 4d dynamic scene. *arXiv preprint arXiv:2311.14603*, 2023.

[144] Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6517–6526, 2024.

[145] Aliasghar Khani, Saeid Asgari, Aditya Sanghi, Ali Mahdavi Amiri, and Ghassan Hamarneh. Slime: Segment like me. In *The Twelfth International Conference on Learning Representations*, 2023.

[146] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. *Advances in Neural Information Processing Systems*, 36:1363–1389, 2023.

[147] Niladri Shekhar Dutt, Sanjeev Muralikrishnan, and Niloy J Mitra. Diffusion 3d features (diff3f): Decorating untextured shapes with distilled semantic features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4494–4504, 2024.

[148] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[149] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[150] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[151] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[152] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2022.

[153] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[154] Junyi Zhang, Charles Herrmann, Junhwa Hur, Luisa Polania Cabrera, Varun Jampani, Deqing Sun, and Ming-Hsuan Yang. A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence. *Advances in Neural Information Processing Systems*, 36, 2024.

[155] Grace Luo, Lisa Dunlap, Dong Huk Park, Aleksander Holynski, and Trevor Darrell. Diffusion hyperfeatures: Searching through time and space for semantic correspondence. *Advances in Neural Information Processing Systems*, 36, 2024.

[156] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. URL `https://openai.com/research/video-generation-models-as-world-simulators`.

[157] Google DeepMind. Veo, 2024. URL `https://deepmind.google/technologies/veo/`.

[158] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly,

et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[159] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023.

[160] Adrien Maglo, Guillaume Lavoué, Florent Dupont, and Céline Hudelot. 3d mesh compression: Survey, comparisons, and emerging trends. *ACM Computing Surveys (CSUR)*, 47(3):1–41, 2015.

[161] Frederick I Parke. Computer generated animation of faces. In *Proceedings of the ACM annual conference-Volume 1*, pages 451–457, 1972.

[162] Frederic Ira Parke. *A parametric model for human faces.* The University of Utah, 1974.

[163] Joseph O'rourke and Norman I Badler. Model-based image analysis of human motion using constraint propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6): 522–536, 1980.

[164] Jochen Tautges, Arno Zinke, Björn Krüger, Jan Baumann, Andreas Weber, Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Bernd Eberhardt. Motion reconstruction using sparse accelerometer data. *ACM Transactions on Graphics (ToG)*, 30(3):1–12, 2011.

[165] David Hogg. Model-based vision: a program to see a walking person. *Image and Vision computing*, 1(1):5–20, 1983.

[166] Leonid Sigal, Alexandru O Balan, and Michael J Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International journal of computer vision*, 87(1):4–27, 2010.

[167] Leonid Sigal, Sidharth Bhatia, Stefan Roth, Michael J Black, and Michael Isard. Tracking loose-limbed people. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. IEEE, 2004.

[168] Leonid Pishchulin, Arjun Jain, Mykhaylo Andriluka, Thorsten Thormählen, and Bernt Schiele. Articulated people detection and pose estimation: Reshaping the future. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3178–3185. IEEE, 2012.

[169] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, 2014.

[170] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017.

[171] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pages 561–578. Springer, 2016.

[172] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3d human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2640–2649, 2017.

[173] Sebastian Lutz, Richard Blythman, Koustav Ghosal, Matthew Moynihan, Ciaran Simms, and Aljosa Smolic. Jointformer: Single-frame lifting transformer with error prediction and refinement for 3d human pose estimation. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 1156–1163. IEEE, 2022.

[174] Siwei Zhang, Bharat Lal Bhatnagar, Yuanlu Xu, Alexander Winkler, Petr Kadlecek, Siyu Tang, and Federica Bogo. Rohm: Robust human motion reconstruction via diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14606–14617, 2024.

[175] Soyong Shin, Juyong Kim, Eni Halilaj, and Michael J Black. Wham: Reconstructing world-

grounded humans with accurate 3d motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2070–2080, 2024.

[176] Jose Luis Ponton, Haoran Yun, Andreas Aristidou, Carlos Andujar, and Nuria Pelechano. Sparseposer: Real-time full-body motion reconstruction from sparse data. *ACM Transactions on Graphics*, 43(1):1–14, 2023.

[177] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013.

[178] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3334–3342, 2015.

[179] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5442–5451, 2019.

[180] Wentao Zhu, Xiaoxuan Ma, Dongwoo Ro, Hai Ci, Jinlu Zhang, Jiaxin Shi, Feng Gao, Qi Tian, and Yizhou Wang. Human motion generation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[181] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5152–5161, 2022.

[182] Guy Tevet, Brian Gordon, Amir Hertz, Amit H Bermano, and Daniel Cohen-Or. Motionclip: Exposing human motion generation to clip space. In *European Conference on Computer Vision*, pages 358–374. Springer, 2022.

[183] Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. Motiongpt: Human motion as a foreign language. *Advances in Neural Information Processing Systems*, 36, 2024.

[184] Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. Human motion diffusion model. In *The Eleventh International Conference on Learning Representations*, 2023.

[185] Matthias Plappert, Christian Mandery, and Tamim Asfour. The KIT motion-language dataset. *Big Data*, 4(4):236–252, dec 2016. doi: 10.1089/big.2016.0028.

[186] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5152–5161, June 2022.

[187] Roy Kapon, Guy Tevet, Daniel Cohen-Or, and Amit H Bermano. Mas: Multi-view ancestral sampling for 3d motion generation using 2d diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1965–1974, 2024.

[188] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9298–9309, 2023.

[189] Michael Gleicher. Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 33–42, 1998.

[190] Zhouyingcheng Liao, Jimei Yang, Jun Saito, Gerard Pons-Moll, and Yang Zhou. Skeleton-free pose transfer for stylized 3d characters. In *European Conference on Computer Vision*, pages 640–656. Springer, 2022.

[191] Jiashun Wang, Xueting Li, Sifei Liu, Shalini De Mello, Orazio Gallo, Xiaolong Wang, and Jan Kautz. Zero-shot pose transfer for unrigged stylized 3d characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8704–8714, 2023.

[192] Helge Rhodin, James Tompkin, Kwang In Kim, Kiran Varanasi, Hans-Peter Seidel, and Christian Theobalt. Interactive motion mapping for real-time character control. In *Computer Graphics Forum*, volume 33, pages 273–282. Wiley Online Library, 2014.

[193] Tianyu Li, Jungdam Won, Alexander Clegg, Jeonghwan Kim, Akshara Rai, and Sehoon Ha. Ace: Adversarial correspondence embedding for cross morphology motion retargeting from human to nonhuman characters. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–11, 2023.

[194] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[195] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

[196] Lance Williams. Casting curved shadows on curved surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 270–274, 1978.

[197] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *The Twelfth International Conference on Learning Representations*, 2024.

[198] 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.

[199] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

[200] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[201] Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015. doi: 10.18637/jss.v067.i01.

[202] Ruilong Li, Shan Yang, David A Ross, and Angjoo Kanazawa. Ai choreographer: Music conditioned 3d dance generation with aist++. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13401–13412, 2021.

[203] Lukas Uzolas, Elmar Eisemann, and Petr Kellnhofer. Surface-aware distilled 3d semantic features. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pages 1–12, 2025.

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

[205] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part III 11*, pages 356–369. Springer, 2010.

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

[207] Grace Luo, Lisa Dunlap, Dong Huk Park, Aleksander Holynski, and Trevor Darrell. Diffusion hyperfeatures: Searching through time and space for semantic correspondence. *Advances in Neural Information Processing Systems*, 36:47500–47510, 2023.

[208] Thomas Wimmer, Peter Wonka, and Maks Ovsjanikov. Back to 3d: Few-shot 3d keypoint detection with back-projected 2d features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4154–4164, 2024.

[209] Junyi Zhang, Charles Herrmann, Junhwa Hur, Eric Chen, Varun Jampani, Deqing Sun, and

Ming-Hsuan Yang. Telling left from right: Identifying geometry-aware semantic correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3076–3085, 2024.

[210] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[211] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. Featured Certification.

[212] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *arXiv preprint arXiv:2112.05814*, 2(3):4, 2021.

[213] Junyi Zhang, Charles Herrmann, Junhwa Hur, Luisa Polania Cabrera, Varun Jampani, Deqing Sun, and Ming-Hsuan Yang. A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence. *Advances in Neural Information Processing Systems*, 36: 45533–45547, 2023.

[214] Aleksandar Shtedritski, Christian Rupprecht, and Andrea Vedaldi. Shic: Shape-image correspondences with no keypoint supervision. In *European Conference on Computer Vision*, pages 129–145. Springer, 2024.

[215] Michael Fischer, Zhengqin Li, Thu Nguyen-Phuoc, Aljaz Bozic, Zhao Dong, Carl Marshall, and Tobias Ritschel. Nerf analogies: Example-based visual attribute transfer for nerfs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4640–4650, 2024.

[216] Luca Morreale, Noam Aigerman, Vladimir G Kim, and Niloy J Mitra. Neural semantic surface maps. In *Computer Graphics Forum*, volume 43, page e15005. Wiley Online Library, 2024.

[217] Zhangquan Chen, Puhua Jiang, and Ruqi Huang. Dv-matcher: Deformation-based non-rigid point cloud matching guided by pre-trained visual features. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 27264–27274, 2025.

[218] Octave Mariotti, Oisin Mac Aodha, and Hakan Bilen. Improving semantic correspondence with viewpoint-guided spherical maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19521–19530, 2024.

[219] Haolin Liu, Xiaohang Zhan, Zizheng Yan, Zhongjin Luo, Yuxin Wen, and Xiaoguang Han. Stable-score: A stable registration-based framework for 3d shape correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 917–928, 2025.

[220] Tong He, Haibin Huang, Li Yi, Yuqian Zhou, Chihao Wu, Jue Wang, and Stefano Soatto. Geonet: Deep geodesic networks for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6888–6897, 2019.

[221] Puhua Jiang, Mingze Sun, and Ruqi Huang. Neural intrinsic embedding for non-rigid point cloud matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21835–21845, 2023.

[222] Bingchen Gong, Diego Gomez, Abdullah Hamdi, Abdelrahman Eldesokey, Ahmed Abdelreheem, Peter Wonka, and Maks Ovsjanikov. Zerokey: Point-level reasoning and zero-shot 3d keypoint detection from large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22089–22099, 2025.

[223] Ahmed Abdelreheem, Abdelrahman Eldesokey, Maks Ovsjanikov, and Peter Wonka. Zero-shot 3d shape correspondence. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–11, 2023.

[224] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

[225] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[226] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.

[227] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[228] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[229] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020.

[230] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.

[231] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[232] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018.

[233] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020.

[234] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[235] Jiaze Sun, Zhixiang Chen, and Tae-Kyun Kim. Mapconnet: Self-supervised 3d pose transfer with mesh and point contrastive learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14452–14462, 2023.

[236] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992.

[237] Benedict J Brown and Szymon Rusinkiewicz. Global non-rigid alignment of 3-d scans. *ACM Transactions on Graphics (TOG)*, 26(3):21–es, 2007.

[238] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-icp: A globally optimal solution to 3d icp point-set registration. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2241–2254, 2015.

[239] Etienne Corman, Maks Ovsjanikov, and Antonin Chambolle. Supervised descriptor learning for non-rigid shape matching. In *European conference on computer vision*, pages 283–298. Springer, 2014.

[240] Kan Guo, Dongqing Zou, and Xiaowu Chen. 3d mesh labeling via deep convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 35(1):1–12, 2015.

[241] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas

Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1802–1811, 2017.

[242] Zan Gojcic, Caifa Zhou, Jan D Wegner, and Andreas Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5545–5554, 2019.

[243] Souhaib Attaiki, Lei Li, and Maks Ovsjanikov. Generalizable local feature pre-training for deformable shape analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13650–13661, 2023.

[244] Gil Elbaz, Tamar Avraham, and Anath Fischer. 3d point cloud registration for localization using a deep neural network auto-encoder. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4631–4640, 2017.

[245] Zi Jian Yew and Gim Hee Lee. 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *Proceedings of the European conference on computer vision (ECCV)*, pages 607–623, 2018.

[246] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 195–205, 2018.

[247] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *Proceedings of the European conference on computer vision (ECCV)*, pages 602–618, 2018.

[248] Jiacheng Deng, Chuxin Wang, Jiahao Lu, Jianfeng He, Tianzhu Zhang, Jiyang Yu, and Zhe Zhang. Se-ornet: Self-ensembling orientation-aware network for unsupervised point cloud shape correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5364–5373, 2023.

[249] 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.

[250] Itai Lang, Dvir Ginzburg, Shai Avidan, and Dan Raviv. Dpc: Unsupervised deep point correspondence via cross and self construction. In *2021 International Conference on 3D Vision (3DV)*, pages 1442–1451. IEEE, 2021.

[251] Yiming Zeng, Yue Qian, Zhiyu Zhu, Junhui Hou, Hui Yuan, and Ying He. Corrnet3d: Unsupervised end-to-end learning of dense correspondence for 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6052–6061, 2021.

[252] Qianwei Xia, Juyong Zhang, Zheng Fang, Jin Li, Mingyue Zhang, Bailin Deng, and Ying He. Geodesicembedding (ge): a high-dimensional embedding approach for fast geodesic distance queries. *IEEE Transactions on Visualization and Computer Graphics*, 28(12):4930–4939, 2021.

[253] Facundo Mémoli and Guillermo Sapiro. A theoretical and computational framework for isometry invariant recognition of point cloud data. *Foundations of Computational Mathematics*, 5:313–347, 2005.

[254] Oshri Halimi, Or Litany, Emanuele Rodola, Alex M Bronstein, and Ron Kimmel. Unsupervised learning of dense shape correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4370–4379, 2019.

[255] Gil Shamai and Ron Kimmel. Geodesic distance descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6410–6418, 2017.

[256] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[257] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. Deepvcp: An end-to-end deep neural network for point cloud registration. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12–21, 2019.

[258] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3523–3532, 2019.

[259] Theo Deprelle, Thibault Groueix, Matthew Fisher, Vladimir Kim, Bryan Russell, and Mathieu Aubry. Learning elementary structures for 3d shape generation and matching. *Advances in Neural Information Processing Systems*, 32, 2019.

[260] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (ToG)*, 31(4):1–11, 2012.

[261] E Rodolá, M Moeller, D Cremers, et al. Point-wise map recovery and refinement from functional correspondence. In *VMV 2015-Vision, Modeling and Visualization*, pages 25–32. Eurographics Association, 2015.

[262] Or Litany, Tal Remez, Emanuele Rodola, Alex Bronstein, and Michael Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. In *Proceedings of the IEEE international conference on computer vision*, pages 5659–5667, 2017.

[263] Jean-Michel Roufosse, Abhishek Sharma, and Maks Ovsjanikov. Unsupervised deep learning for structured shape matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1617–1627, 2019.

[264] Nicolas Donati, Abhishek Sharma, and Maks Ovsjanikov. Deep geometric functional maps: Robust feature learning for shape correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8592–8601, 2020.

[265] Dvir Ginzburg and Dan Raviv. Cyclic functional mapping: Self-supervised correspondence between non-isometric deformable shapes. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 36–52. Springer, 2020.

[266] Mingze Sun, Shiwei Mao, Puhua Jiang, Maks Ovsjanikov, and Ruqi Huang. Spatially and spectrally consistent deep functional maps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14497–14507, 2023.

[267] Marvin Eisenberger, Zorah Lahner, and Daniel Cremers. Smooth shells: Multi-scale shape registration with functional maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12265–12274, 2020.

[268] Marvin Eisenberger, Aysim Toker, Laura Leal-Taixé, and Daniel Cremers. Deep shells: Unsupervised shape correspondence with optimal transport. *Advances in Neural information processing systems*, 33:10491–10502, 2020.

[269] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, et al. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 929–947, 2024.

[270] Nicholas Sharp et al. Polyscope, 2019. www.polyscope.run.

[271] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

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

[273] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image

recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[274] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[275] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[276] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. The heat method for distance computation. *Communications of the ACM*, 60(11):90–99, 2017.

[277] Nicholas Sharp, Keenan Crane, et al. Geometrycentral: A modern c++ library of data structures and algorithms for geometry processing. 2019.

[278] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 851–866. 2023.

[279] Benjamin Biggs, Thomas Roddick, Andrew Fitzgibbon, and Roberto Cipolla. Creatures great and SMAL: Recovering the shape and motion of animals from video. In *ACCV*, 2018.

[280] Simone Melzi, Riccardo Marin, Emanuele Rodolà, Umberto Castellani, Jing Ren, Adrien Poulenard, P Ovsjanikov, et al. Shrec'19: matching humans with different connectivity. In *Eurographics Workshop on 3D Object Retrieval*, pages 1–8. The Eurographics Association, 2019.

[281] Roberto M Dyke, Yu-Kun Lai, Paul L Rosin, Stefano Zappalà, Seana Dykes, Daoliang Guo, Kun Li, Riccardo Marin, Simone Melzi, and Jingyu Yang. Shrec'20: Shape correspondence with non-isometric deformations. *Computers & Graphics*, 92:28–43, 2020.

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

[283] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[284] Peng-Shuai Wang, Yang Liu, and Xin Tong. Dual octree graph networks for learning adaptive volumetric shape representations. *ACM Transactions on Graphics (TOG)*, 41(4):1–15, 2022.

[285] Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12706–12716, 2021.

[286] Dan Casas and Marc Comino-Trinidad. SMPLitex: A Generative Model and Dataset for 3D Human Texture Estimation from Single Image. In *British Machine Vision Conference (BMVC)*, 2023.

[287] Mingze Sun, Chen Guo, Puhua Jiang, Shiwei Mao, Yurun Chen, and Ruqi Huang. Srif: Semantic shape registration empowered by diffusion-based image morphing and flow estimation. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024.

[288] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4561–4570, 2019.

[289] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. *ACM transactions on graphics*, 39(4), 2020.

[290] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36:35799–35813, 2023.

[291] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. In *ACM SIGGRAPH*, pages 561–566. 2005.

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

[293] Sanjeev Muralikrishnan, Niladri Dutt, Siddhartha Chaudhuri, Noam Aigerman, Vladimir Kim, Matthew Fisher, and Niloy J Mitra. Temporal residual jacobians for rig-free motion transfer. In *European Conference on Computer Vision*, pages 93–109. Springer, 2024.

[294] Seungwoo Yoo, Juil Koo, Kyeongmin Yeo, and Minhyuk Sung. Neural pose representation learning for generating and transferring non-rigid object poses. *Advances in Neural Information Processing Systems*, 37, 2024.

[295] Arman Maesumi, Tanish Makadia, Thibault Groueix, Vladimir Kim, Daniel Ritchie, and Noam Aigerman. Poissonnet: A local-global approach for learning on surfaces. *ACM Transactions on Graphics (TOG)*, 44(6):1–16, 2025.

[296] Ruben Wiersma, Ahmad Nasikun, Elmar Eisemann, and Klaus Hildebrandt. Deltaconv: anisotropic operators for geometric deep learning on point clouds. *ACM Trans. Graph.*, 41(4): 105:1–105:10, 2022. doi: 10.1145/3528223.3530166.

[297] Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. Diffusionnet: Discretization agnostic learning on surfaces. *ACM Trans. Graph.*, 41(3):27:1–27:16, 2022. doi: 10.1145/3507905.

[298] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *IEEE International Conference on Computer Vision Workshops, ICCV 2011 Workshops, Barcelona, Spain, November 6-13, 2011*, pages 1626–1633. IEEE Computer Society, 2011. doi: 10.1109/ICCVW.2011.6130444. URL https://doi.org/10.1109/ICCVW.2011.6130444.

[299] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, 1998.

[300] Kfir Aberman, Peizhuo Li, Dani Lischinski, Olga Sorkine-Hornung, Daniel Cohen-Or, and Baoquan Chen. Skeleton-aware networks for deep motion retargeting. *ACM Transactions on Graphics (ToG)*, 39(4):62–1, 2020.

[301] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23(3):399–405, 2004.

[302] Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. Automatic unpaired shape deformation transfer. *ACM Transactions on Graphics (ToG)*, 37(6): 1–15, 2018.

[303] Sanjeev Muralikrishnan, Niladri Shekhar Dutt, and Niloy J Mitra. Smf: Template-free and rig-free animation transfer using kinetic codes. *ACM Transactions on Graphics (TOG)*, 44(6): 1–11, 2025.

[304] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3, 2025.

[305] Tobias Weißberg, Weikang Wang, Paul Roetzer, Nafie El Amrani, and Florian Bernard. Symmetry informative and agnostic feature disentanglement for 3d shapes. In *Thirteenth International Conference on 3D Vision*, 2026.

[306] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 19729–19739, 2023.

[307] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34 (4):18–42, 2017.

[308] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric Deep

Learning: Grids, Groups, Graphs, Geodesics, and Gauges, May 2021. URL http://arxiv.org/abs/2104.13478. arXiv:2104.13478 [cs].

[309] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[310] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial transformer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/33ceb07bf4eeb3da587e268d663aba1a-Paper.pdf.

[311] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds. *CoRR*, abs/1802.0, 2018. URL http://arxiv.org/abs/1802.08219. arXiv:1802.08219.

[312] Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral cnn. In *International conference on Machine learning*, pages 1321–1330. PMLR, 2019.

[313] Ruben Wiersma, Elmar Eisemann, and Klaus Hildebrandt. Cnns on surfaces using rotation-equivariant features. *ACM Trans. Graph.*, 39(4), August 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392437. URL https://doi.org/10.1145/3386569.3392437.

[314] Pim De Haan, Maurice Weiler, Taco Cohen, and Max Welling. Gauge equivariant mesh cnns: Anisotropic convolutions on geometric graphs. In *International Conference on Learning Representations*, 2020.

[315] Adrien Poulenard and Maks Ovsjanikov. Multi-directional geodesic neural networks via equivariant convolution. *ACM Transactions on Graphics (TOG)*, 37(6):1–14, 2018.

[316] Thomas W Mitchel, Vladimir G Kim, and Michael Kazhdan. Field convolutions for surface cnns. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10001–10011, 2021.

[317] In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[318] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230, January 2008. ISSN 1077-2626. doi: 10.1109/TVCG.2007.1054. URL https://doi.org/10.1109/TVCG.2007.1054.

[319] Daniele Panozzo, Enrico Puppo, Marco Tarini, and Olga Sorkine-Hornung. Frame fields: anisotropic and non-orthogonal cross fields. *ACM Trans. Graph.*, 33(4):134:1–134:11, 2014. doi: 10.1145/2601097.2601179. URL https://doi.org/10.1145/2601097.2601179.

[320] Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. Designing N-polyvector fields with complex polynomials. *Comput. Graph. Forum*, 33(5):1–11, 2014. doi: 10.1111/CGF.12426. URL https://doi.org/10.1111/cgf.12426.

[321] Ahmed AA Osman, Timo Bolkart, and Michael J Black. Star: Sparse trained articulated human body regressor. In *European Conference on Computer Vision*, pages 598–613. Springer, 2020.

[322] Benjamin Biggs, Oliver Boyne, James Charles, Andrew Fitzgibbon, and Roberto Cipolla. Who left the dogs out? 3d animal reconstruction with expectation maximization in the loop. In *European Conference on Computer Vision*, pages 195–211. Springer, 2020.

[323] Jiacong Xu, Yi Zhang, Jiawei Peng, Wufei Ma, Artur Jesslen, Pengliang Ji, Qixin Hu, Jiehua Zhang, Qihao Liu, Jiahao Wang, et al. Animal3d: A comprehensive dataset of 3d animal pose and shape. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9099–9109, 2023.

[324] Gerard Pons-Moll, Sergi Pujades, Sonny Hu, and Michael Black. Clothcap: Seamless 4d clothing capture and retargeting. *ACM Transactions on Graphics, (Proc. SIGGRAPH)*, 36(4), 2017. URL http://dx.doi.org/10.1145/3072959.3073711. Two first authors contributed equally.

[325] Qianli Ma, Jinlong Yang, Anurag Ranjan, Sergi Pujades, Gerard Pons-Moll, Siyu Tang, and Michael J. Black. Learning to Dress 3D People in Generative Clothing. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[326] Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. Scalable locally injective mappings. *ACM Transactions on Graphics (TOG)*, 36(4):1, 2017.

[327] Shashank Tripathi, Lea Müller, Chun-Hao P Huang, Omid Taheri, Michael J Black, and Dimitrios Tzionas. 3d human pose estimation via intuitive physics. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4713–4725, 2023.

[328] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *2024 International Conference on 3D Vision (3DV)*, pages 800–809. IEEE, 2024.

[329] Diwen Wan, Yuxiang Wang, Ruijie Lu, and Gang Zeng. Template-free articulated gaussian splatting for real-time reposable dynamic view synthesis. *Advances in Neural Information Processing Systems*, 37:62000–62023, 2024.

[330] Isabella Liu, Zhan Xu, Wang Yifan, Hao Tan, Zexiang Xu, Xiaolong Wang, Hao Su, and Zifan Shi. Riganything: Template-free autoregressive rigging for diverse 3d assets. *ACM Transactions on Graphics (TOG)*, 44(4):1–12, 2025.

[331] Shaokai Ye, Anastasiia Filippova, Jessy Lauer, Steffen Schneider, Maxime Vidal, Tian Qiu, Alexander Mathis, and Mackenzie Weygandt Mathis. Superanimal pretrained pose estimation models for behavioral analysis. *Nature communications*, 15(1):5165, 2024.

[332] Zackory Erickson, Vamsee Gangaram, Ariel Kapusta, C Karen Liu, and Charles C Kemp. Assistive gym: A physics simulation framework for assistive robotics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10169–10176. IEEE, 2020.

[333] Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, Hao Su, Sanja Fidler, Nicholas Sharp, and Jun Gao. Partfield: Learning 3d feature fields for part segmentation and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9704–9715, 2025.

[334] Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. Tensor4d: Efficient neural 4d decomposition for high-fidelity dynamic reconstruction and rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16632–16642, 2023.

[335] Jean Feydy, Joan Glaunès, Benjamin Charlier, and Michael Bronstein. Fast geometric learning with symbolic matrices. *Advances in Neural Information Processing Systems*, 33, 2020.

[336] Ta-Chih Lee, Rangasami L Kashyap, and Chong-Nam Chu. Building skeleton models via 3-d medial surface axis thinning algorithms. *CVGIP: graphical models and image processing*, 56(6): 462–478, 1994.

[337] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[338] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022.

[339] William Gao, Noam Aigerman, Thibault Groueix, Vova Kim, and Rana Hanocka. Textdeformer: Geometry manipulation using text guidance. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023.

[340] Peng-Shuai Wang, Yang Liu, and Xin Tong. Dual octree graph networks for learning adaptive volumetric shape representations. *ACM Transactions on Graphics (SIGGRAPH)*, 41(4), 2022.

[341] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997.

[342] Xinxin Zuo, Sen Wang, Jiangbin Zheng, Weiwei Yu, Minglun Gong, Ruigang Yang, and Li Cheng. Sparsefusion: Dynamic human avatar modeling from sparse rgbd images. *IEEE Transactions on Multimedia*, 23:1617–1629, 2021.

Acknowledgments

This PhD has been a long time in the making, and its biggest contributors are my parents. I was not an easy child, and nobody expected me to go far in school. I have cost my family countless nerves and caused a lot of frustrations, but I have been lucky to have parents who were able and willing to support me throughout. It enabled me to develop a genuine interest in learning and creating at my own pace, which, contrary to all expectations, eventually led to succeeding and even flourishing at school and university. For this patience, love, and continuing support, I am deeply grateful, because I know that many have not enjoyed such a privilege.

My interest in creating images with computers was sparked during my Master's degree in Image Processing and Computer Vision. I was fortunate enough to receive a scholarship to study across Europe, which allowed me to focus on my studies and meet and become friends with people from all over the world. This program had a great impact on me and my way of thinking. I am thankful for having been given that opportunity, and while the impact of such programs might be hard to quantify, it is truly tremendous and one of the greatest reasons I wanted and was able to pursue a PhD in this domain.

Coming from a Computer Vision background and being mostly familiar with image synthesis through generative models, my path into Computer Graphics was not an obvious one. I am happy to have had Elmar Eisemann and Petr Kellnhofer as my supervisors, guiding me along this path. I remember Petr's kind and detailed email to me in response to my worries about research directions before I even started my PhD. He assured me that we can pretty much do anything as long as we understand the scope of the related work, which quickly resolved my self-doubts. I was happy to learn that his approachable manner was not limited to emails alone, and I am grateful for Petr's involved guidance throughout. Elmar's enthusiasm for computer graphics and its applications is infectious. It allowed me to find new angles for how my research could be applied in the real world, which made it all the more motivating. Combined, their support fostered an environment that enabled me to explore and tackle problems I did not even know existed at the beginning of my PhD and that I did not think I was capable of tackling.

I am aware that the flexibility to explore and pursue one's research interests is not a given, and I want to thank TU Delft for creating a system that enabled me to do so. I also want to thank Klaus Hildebrandt for helping us at the final stages of this dissertation, making it possible.

I am thankful for all the members of the CGV group being part of this. Mark, for bringing the group together by hosting board game nights. Berend, for showing me that triangles are actually not that bad after all. Ruben, for being a great collaborator and teaching me about differential geometry. Jackson, for being my machine learning sparring buddy. Amir, for his strong opinions and insights, as well as VR experiences. Soumyadeep, for always being up for a laugh. Ali and Chen-Chi, for their efforts in making Visulunch happen. Benno, for helping me convince the group of the merits of neural representations. Alex and Jerry, for raising the bar for coffee in the group. Mika and Joris, for making the office a nice place to be. Baran, for our light-hearted conversations. Christoph, for exploring Shenzhen and Hong Kong with me. The staff and support team that make this group work: Ricardo, Michael, Thomas, Rafael, Ruud, and Lauretta. And of course, all the other members, former members, and other PhDs at TU Delft who were part of my PhD journey: Nicolas, Mathijs, Xiaoqi, Yang, Mrinal, Fengshi, Celine, Priyanka, Xuemei, Shaad, Nail, Xiangwei, Gabriel, Ojas, and Tianqi. I would also like to express my gratitude towards my mentors during my time at Google, and to the other interns who made my stay in Zurich really exciting and formative.

I want to thank all my friends and family for supporting me throughout the PhD. It is all too easy to get absorbed during this process. All the more important it is to have friends who can help maintain pockets of sanity outside of the PhD. Lastly, I am grateful to my girlfriend, Jingwen. Her optimism and encouragement helped me navigate and overcome the hardest parts of this journey, as well as celebrate its successes.

Curriculum Vitae

Ph.D. | **Computer Science** — May 2022 – May 2026
Delft University of Technology — *Delft, Netherlands*

M.Sc. | **Image Processing and Computer Vision** — Sep. 2019 – Sep. 2021
University of Bordeaux — *Bordeaux, France*
Autonomous University of Madrid — *Madrid, Spain*
Pázmány Péter Catholic University — *Budapest, Hungary*
Awards: Honors, Erasmus Mundus Master Scholarship

B.Sc. | **Human-Computer-Interaction** — Oct. 2015 – Jun. 2019
University of Hamburg — *Hamburg, Germany*
Awards: Distinction

Publications

- **L. Uzolas**, E. Eisemann, and P. Kellnhofer. "Template-free articulated neural point clouds for reposable view synthesis." Advances in Neural Information Processing Systems 36 (2023): 31621-31637. [1]
- **L. Uzolas**, E. Eisemann, and P. Kellnhofer. "Motiondreamer: Exploring semantic video diffusion features for zero-shot 3d mesh animation." 2025 International Conference on 3D Vision (3DV). IEEE, 2025. [120]
- **L. Uzolas**, E. Eisemann, and P. Kellnhofer. "Surface-Aware Distilled 3D Semantic Features." Proceedings of the SIGGRAPH Asia 2025 Conference Papers. 2025. [203]
- **L. Uzolas**, R. Wiersma, O. Sorkine-Hornung, and P. Kellnhofer. "Rotation-Equivariant Neural Jacobian Fields for Generalizable 3D Mesh Deformations." Preprint. 2026.

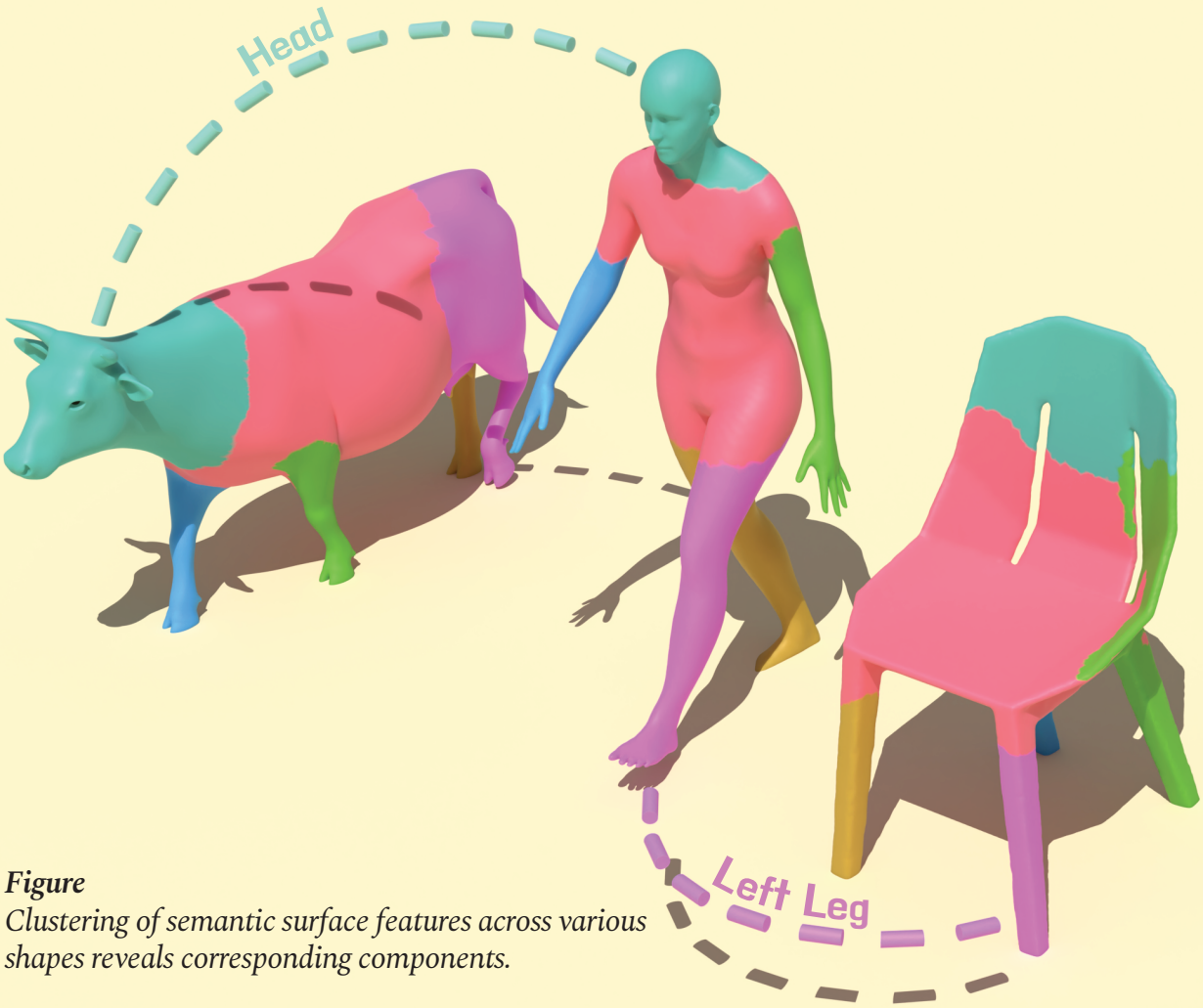


Figure
Clustering of semantic surface features across various shapes reveals corresponding components.