BANMo: Building Animatable 3D Neural Models from Many Causal Videos

2022.02.07

Byungjun Kim

Problem to solve

- Reconstructing 3D deformable objects from casually collected videos
  - deformable: non-rigid object with articulation
  - casually collected: no need for camera parameters
  - video's': exploiting all available videos

source: https://banmo-www.github.io/

Overall Pipeline

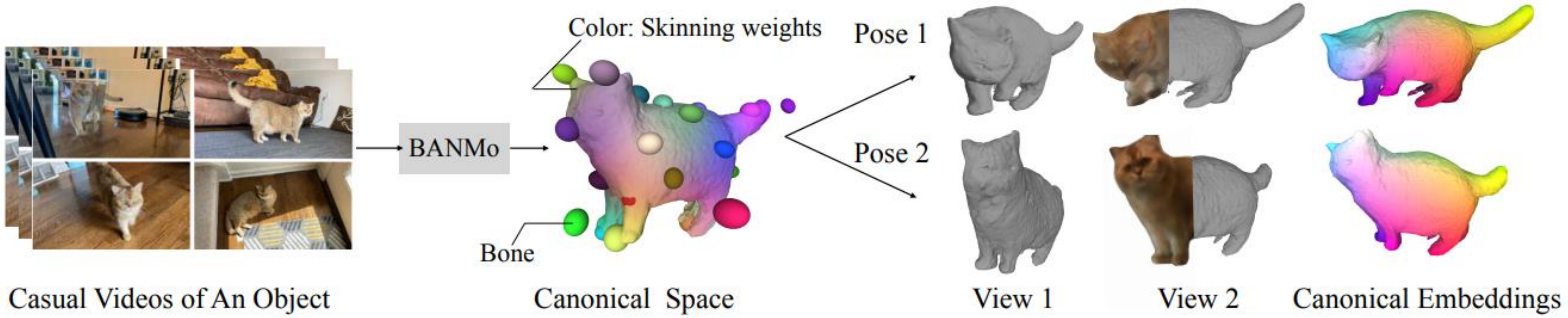
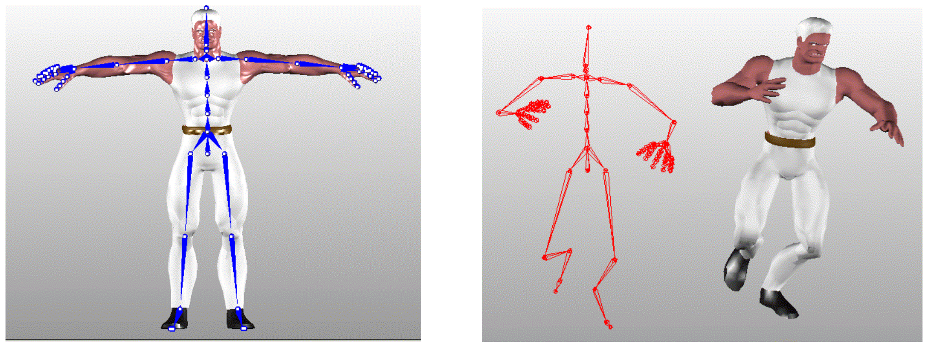


Figure 1. Given multiple casual videos capturing a deformable object, BANMo reconstructs an animatable 3D model, including an implicit canonical 3D shape, appearance, skinning weights, and time-varying articulations, without pre-defined shape templates or registered cameras. **Left**: Input videos; **Middle**: 3D shape, bones, and skinning weights (visualized as surface colors) in the canonical space; **Right**: Posed reconstruction at each time instance with color and canonical embeddings (correspondences are shown as the same colors).

Preliminaries: Skinning (in Computer Graphics)

- Method to embed a skeleton into a mesh
- Bind skin vertices to bones
  - If we animate skeleton, skin will naturally move with it.
- Attach a vertex to many bones at once
  - At a joint, skin is deformed according to a **weighted combination** of bones.
  - The weights are called **skinning weights**.



https://www.okino.com/conv/skinning.htm

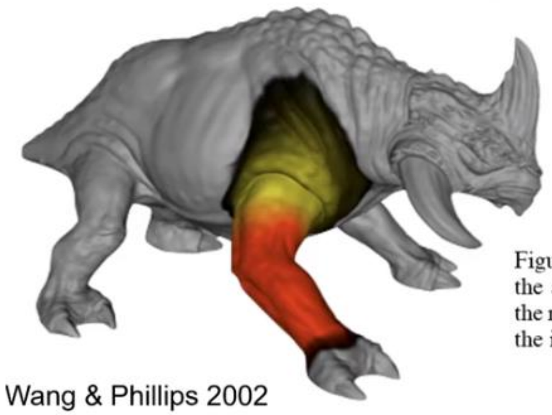
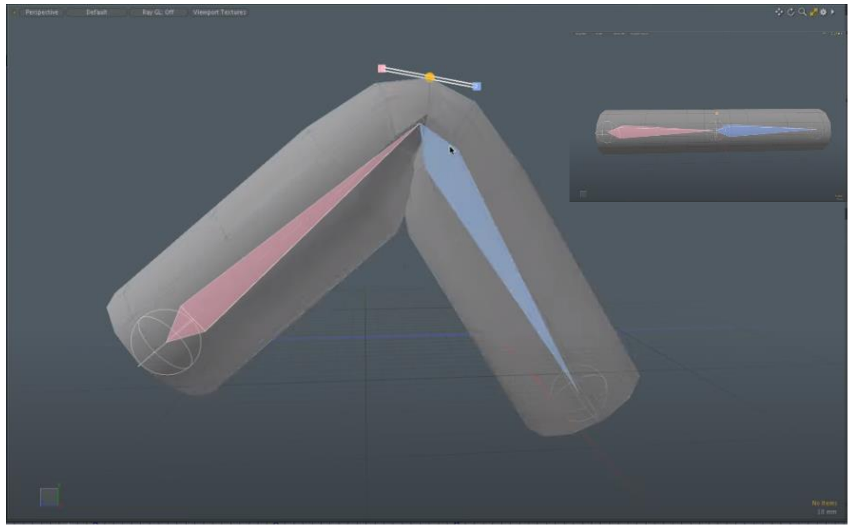


Figure 4: Color coded influences maps from two bones on the animal creature. The yellow area for the upper leg and the red area for the lower leg. The darker the area, the smaller the influence.

Wang & Phillips 2002



Linear Blend Skinning

https://youtu.be/QDXG4wNzkOE

Overall Pipeline

- Train **BANMo**(**B**uilder of **A**nimatable 3D **N**eural **M**odels) with casual videos
- BANMO gives models in Canonical Space with bones and skinning weights.
- Get posed reconstruction at each time instance.

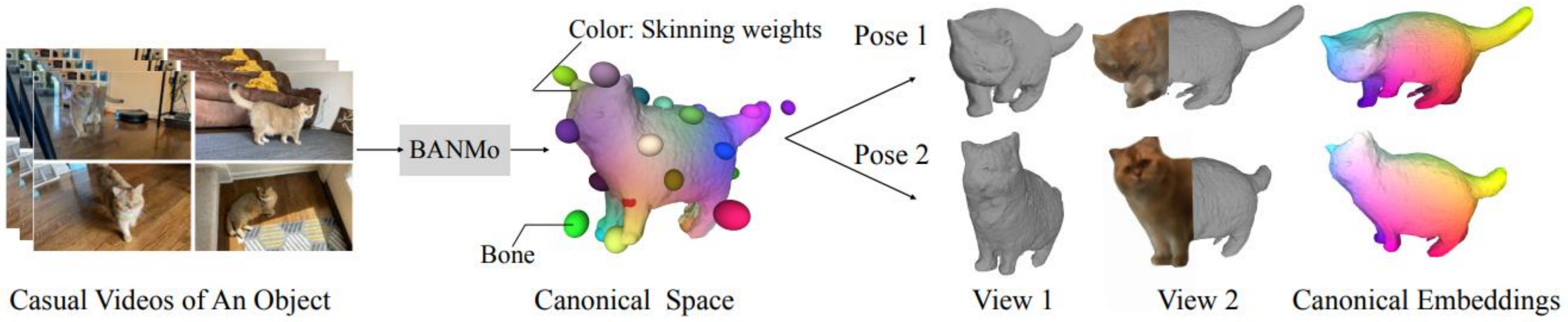


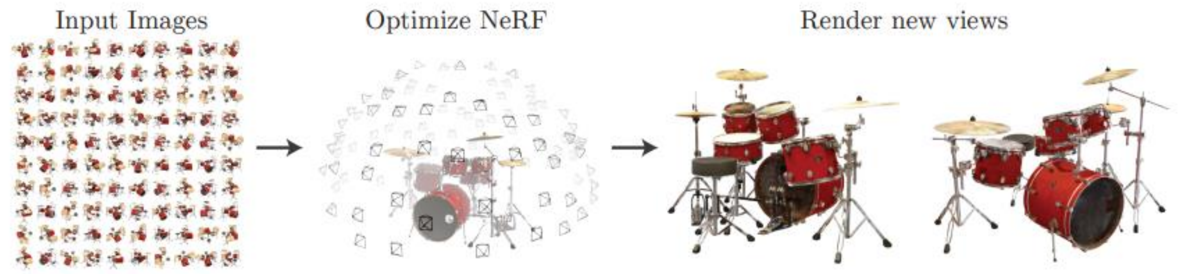
Figure 1. Given multiple casual videos capturing a deformable object, BANMo reconstructs an animatable 3D model, including an implicit canonical 3D shape, appearance, skinning weights, and time-varying articulations, without pre-defined shape templates or registered cameras. **Left**: Input videos; **Middle**: 3D shape, bones, and skinning weights (visualized as surface colors) in the canonical space; **Right**: Posed reconstruction at each time instance with color and canonical embeddings (correspondences are shown as the same colors).

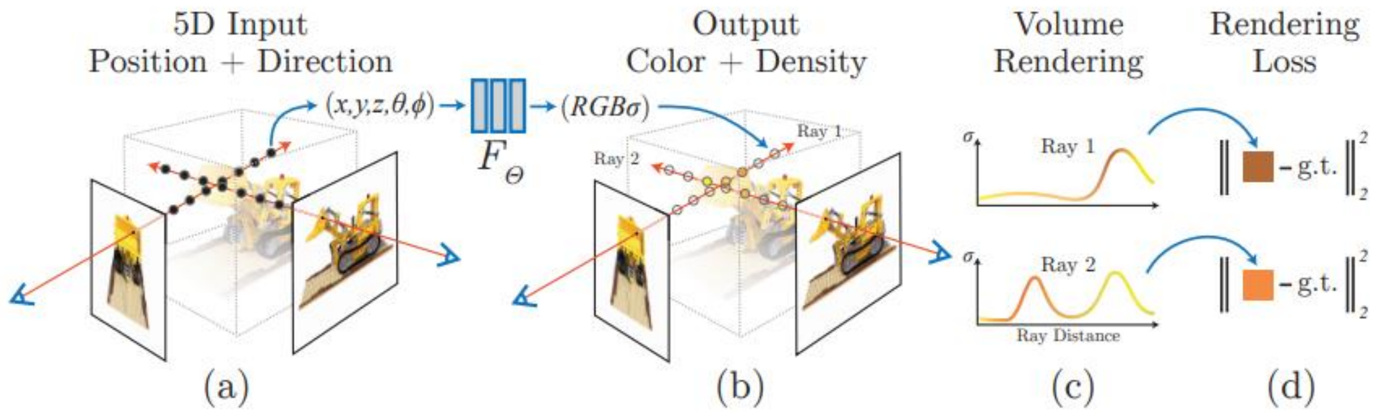
Introduction

- non-rigid 3D reconstruction in dynamic NeRF framework
- 3 core challenges
  - representation of 3D appearance and deformation in canonical space
    - neural implicit functions(color, 3D surface) & neural blend skinning(deformation)
  - mapping from canonical space to each frame
    - canonical embedding matching
  - 2D correspondences across images under view and light change, and object deformations
    - 2D optical flow

Preliminaries: NeRF (ECCV 2020)

- training MLP(continuous scene function) with images from sparse set of views
  - input: 3D point $(\mathrm{x}, \mathrm{y}, \mathrm{z})$, view direction (θ, ϕ) of samples on rays
  - output: color c, volume density σ
- render photorealistic novel view images for **static** scenes

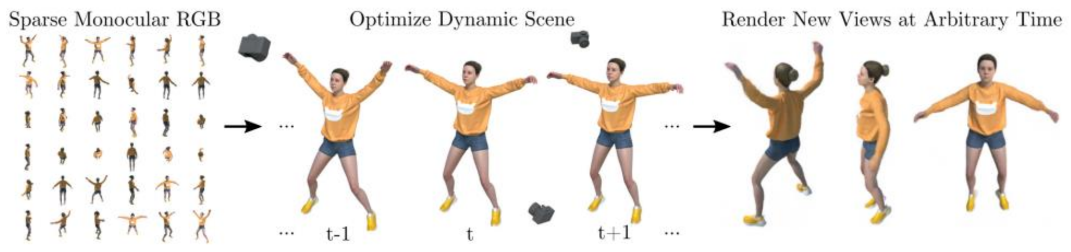




Mildenhall, Ben, et al. "Nerf: Representing scenes as neural radiance fields for view synthesis." European conference on computer vision. Springer, Cham, 2020.

Preliminaries: D-NeRF (CVPR 2021)

- Extend NeRF to a **dynamic** domain
  - Render novel view & time images under motions
- deformation network + canonical network
  - deformation network: predicts deformation of ray samples to canonical space
  - canonical network: encode the scene in canonical configuration(t=0)



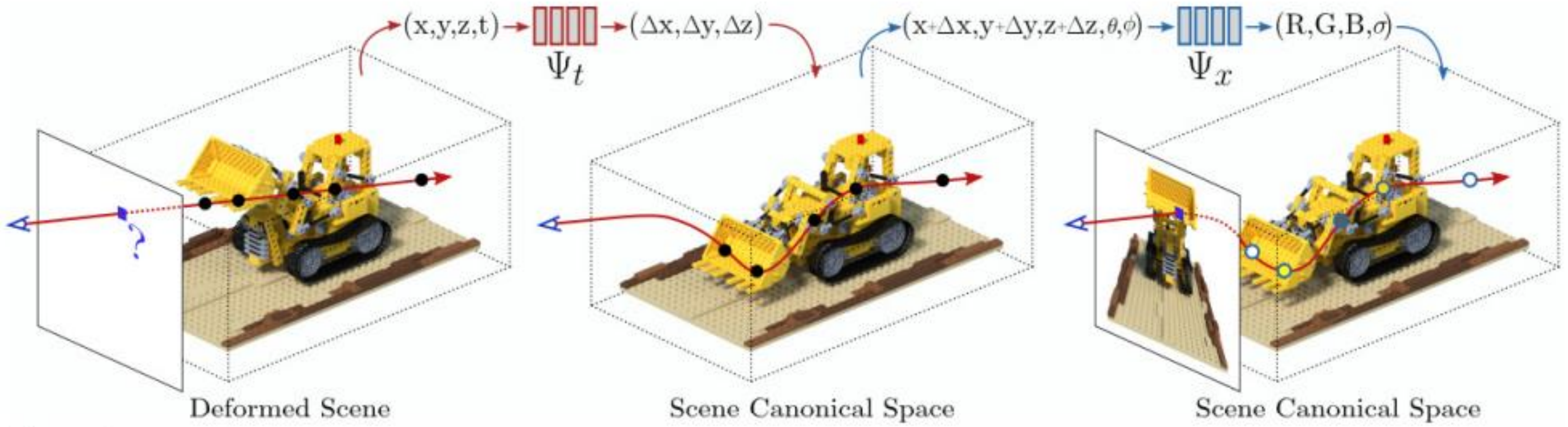
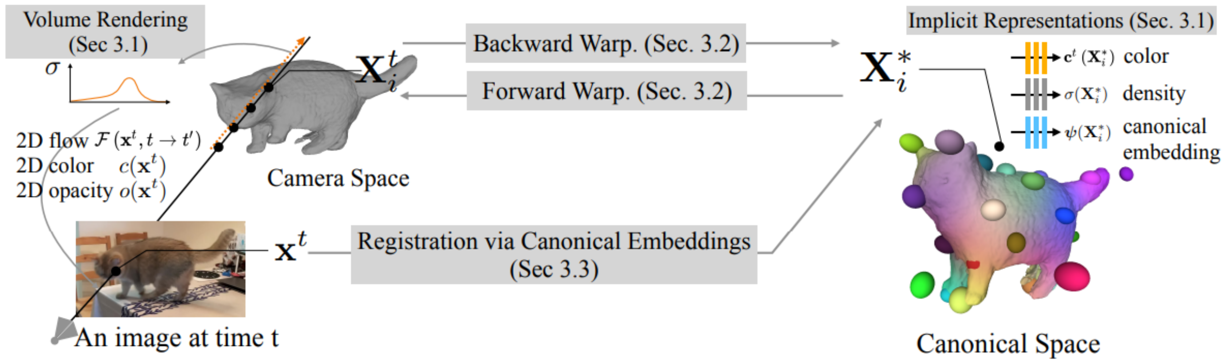


Figure 3: **D-NeRF Model**. The proposed architecture consists of two main blocks: a deformation network Ψ_t mapping all scene deformations to a common canonical configuration; and a canonical network Ψ_x regressing volume density and view-dependent RGB color from every camera ray.

Pumarola, Albert, et al. "D-nerf: Neural radiance fields for dynamic scenes." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

Method : Overview

- Optimize 3D model with 2D cues from video
  - Learn implicit representations of an object in canonical space
  - Forward warp points in canonical space to points in (time-specific) camera space to applying pinhole camera projection. => **optical flow reconstruction loss**
  - Backward warp samples in camera space to canonical space to query color and density. => **color reconstruction loss, silhouette reconstruction loss**
  - Other losses: **feature registration losses, 3D cycle-consistency loss**



Volume Rendering (Sec 3.1)
σ
2D flow $\mathcal{F}(\mathbf{x}^t, t \to t')$
2D color $c(\mathbf{x}^t)$
2D opacity $o(\mathbf{x}^t)$
$\mathbf{X}_i^t$
Camera Space
$\mathbf{x}^t$
An image at time t
Backward Warp. (Sec. 3.2)
Forward Warp. (Sec. 3.2)
Registration via Canonical Embeddings (Sec 3.3)
Implicit Representations (Sec. 3.1)
$\mathbf{X}_i^*$
$\mathbf{c}^t(\mathbf{X}_i^*)$ color
$\sigma(\mathbf{X}_i^*)$ density
$\psi(\mathbf{X}_i^*)$ canonical embedding
Canonical Space

Figure 2. **Method overview.** BANMo optimizes a set of shape and deformation parameters (Sec. 3.1) that describe the video observations in pixel colors, silhouettes, optical flow, and higher-order features descriptors, based on a differentiable volume rendering framework. BANMo uses a neural blend skinning model (Sec. 3.2) to transform 3D points between the camera space and the canonical space, enabling handling large deformations. To register pixels across videos, BANMo jointly optimizes an implicit canonical embedding (CE) (Sec. 3.3).

Method: Canonical Shape Model

- model the shape, appearance in a canonical space with NeRF based method
    - input: 3D point X^*, view direction v^t, learnable environment code ω_e^t
    - output: color c^t, density σ, learned 3D canonical embedding φ

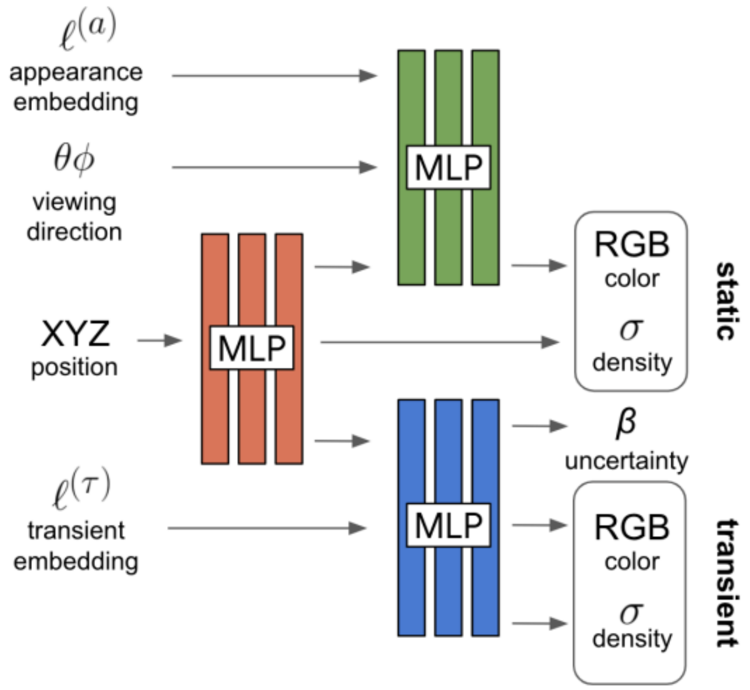
$$\mathbf{c}^t = \mathbf{MLP}_{\mathbf{c}}(\mathbf{X}^*, \mathbf{v}^t, \boldsymbol{\omega}_e^t), \tag{1}$$

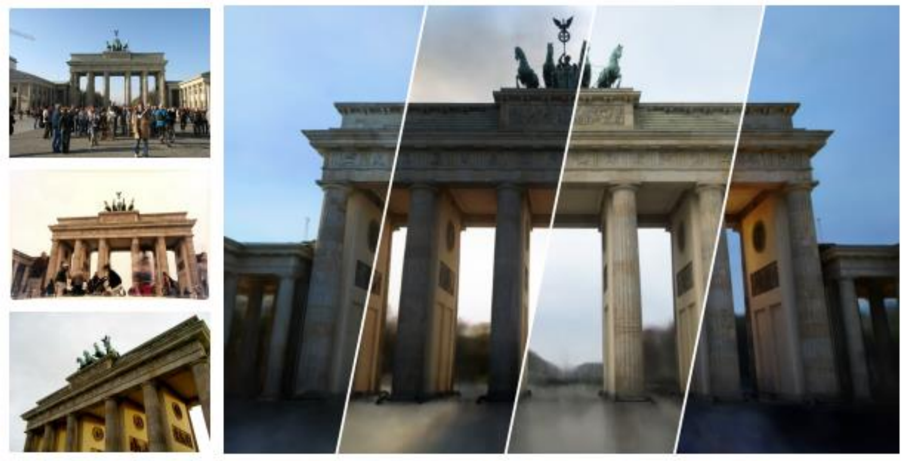
$$\sigma = \Gamma_\beta(\mathbf{MLP}_{\mathrm{SDF}}(\mathbf{X}^*)), \tag{2}$$

$$\boldsymbol{\psi} = \mathbf{MLP}_{\boldsymbol{\psi}}(\mathbf{X}^*). \tag{3}$$

Preliminaries: NeRF in the wild (CVPR 2021 oral)

- NeRF with images under variable illuminations and transient occluders
- Latent Appearance Modeling
  - For each image, assign an **appearance embedding vector** $l^{(a)}$
  - It explains away image-specific photometric and environmental variations





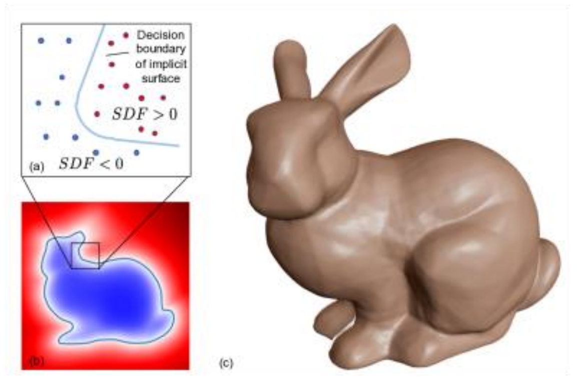
(a) Photos (b) Renderings

Figure 1: Given only an internet photo collection (a), our method is able to render novel views with variable illumination (b). Photos by Flickr users dbowie78, vasnic64, punch / CC BY.

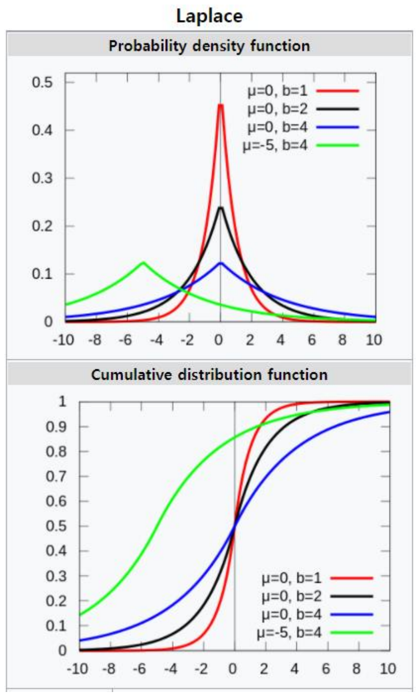
Martin-Brualla, Ricardo, et al. "Nerf in the wild: Neural radiance fields for unconstrained photo collections." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

Method: Canonical Shape Model(cont'd)

- model the shape, appearance in a canonical space with NeRF based method
  - input: 3D point X^*, view direction v^t, learnable environment code ω_e^t
  - output: color c^t, density σ, learned 3D canonical embedding φ
- Predict SDF(Signed Distance Function) first, instead of direct prediction of σ
  - SDF provides a principled way of extracting surface as the zero level-set
  - Γ_β converts the SDF to a density σ
    - Γ_β is the CDF of Laplacian distribution with zero mean and β scale.
    - β controls the solidness of objects, approaching zero for solid objects.



$$\mathbf{c}^t = \mathbf{MLP_c}(\mathbf{X}^*, \mathbf{v}^t, \boldsymbol{\omega}_e^t), \qquad (1)$$
$$\sigma = \Gamma_\beta(\mathbf{MLP}_{\mathbf{SDF}}(\mathbf{X}^*)), \qquad (2)$$
$$\psi = \mathbf{MLP}_\psi(\mathbf{X}^*). \qquad (3)$$



Method: warping model

- forward, backward warping function between canonical and camera space
  - forward warping function $W^{t,\rightarrow}: \mathrm{X}^{*} \rightarrow \mathrm{X}^{\mathrm{t}}$ maps canonical location to camera space location
  - backward warping function $W^{t,\leftarrow}: \mathrm{X}^{\mathrm{t}} \rightarrow \mathrm{X}^{*}$ for inverse mapping
- Apply **bone** transformation, then apply **root body** transformation
  - X^{*} can be considered as points in the "rest" pose
  - $\mathrm{J}^{\mathrm{t},\rightarrow}$, $\mathrm{J}^{\mathrm{t},\leftarrow}$ are weighted averages of B(=25) rigid transformations ΔJ_b^t (next slide)
  - $\mathrm{G}^{\mathrm{t}} \in SE(3)$ is a root body transformation of the object from canonical to time t, which is regressed from MLP given **root pose** latent code at time t $\boldsymbol{\omega_r^t}$

$$\mathbf{G}^t = \mathbf{MLP}_{\mathbf{G}}(\boldsymbol{\omega}_r^t), \quad \mathbf{J}_b^t = \mathbf{MLP}_{\mathbf{J}}(\boldsymbol{\omega}_b^t) \qquad (10)$$

$$\mathbf{X}^t = \mathcal{W}^{t,\rightarrow}(\mathbf{X}^*) = \mathbf{G}^t\left(\mathbf{J}^{t,\rightarrow}\mathbf{X}^*\right), \qquad (7)$$

$$\mathbf{X}^* = \mathcal{W}^{t,\leftarrow}(\mathbf{X}^t) = \mathbf{J}^{t,\leftarrow}\left((\mathbf{G}^t)^{-1}\mathbf{X}^t\right), \qquad (8)$$

Method: warping model – bone transform

- Recall skinning!
  - To animate the object, move "bones", then "skin" is deformed according to skinning weights
- skinning weights $\mathbf{W}^{\mathbf{t},\rightarrow}$, $\mathbf{W}^{\mathbf{t},\leftarrow}$
  - skinning weight $\mathbf{W}$ assigns $\mathbf{X}$ to $\mathbf{B}$ bones given **body pose** latent code $\boldsymbol{\omega}_{\mathbf{b}}$
  - each bone transformation J_b^t, J_b^*(rest pose) are regressed from MLP given $\boldsymbol{\omega}_{\mathbf{b}}$
  - $\boldsymbol{\Delta J_b^t}$ maps points in world coordinates to the local bone coordinates in rest pose by $\boldsymbol{J_b^{*-1}}$, then apply rigid transformation of the bone at time t by $\boldsymbol{J_b^t}$.

$$\mathbf{W}^{t,\rightarrow} = \boldsymbol{S}(\boldsymbol{X}^*, \boldsymbol{w}_b^*), \mathbf{W}^{t,\leftarrow} = \boldsymbol{S}(\boldsymbol{X}^t, \boldsymbol{\omega}_b^t)$$

$$\mathbf{G}^t = \mathrm{MLP}_{\mathbf{G}}(\omega_r^t), \quad \boxed{\mathbf{J}_b^t = \mathrm{MLP}_{\mathbf{J}}(\omega_b^t) \qquad (10)}$$

$$\mathbf{J}^{t,\rightarrow} = \sum_{b=1}^{B} \mathbf{W}_b^{t,\rightarrow} \Delta\mathbf{J}_b^t, \quad \mathbf{J}^{t,\leftarrow} = \sum_{b=1}^{B} \mathbf{W}_b^{t,\leftarrow} (\Delta\mathbf{J}_b^t)^{-1}.$$

$$\boldsymbol{\Delta J_b^t} = \boldsymbol{J_b^t J_b^{*-1}}$$

Method: warping model – bone transform

- skinning weight function $S : (\boldsymbol{X}, \boldsymbol{\omega_b}) \rightarrow \mathbf{W} \in \boldsymbol{R^B}$
- **coarse to fine** manner
  - coarse: Mahalanobis distance between $\boldsymbol{X}$ and 3D Gaussian ellipsoids(=bone)
  - fine: regress delta skinning weights with MLP

$$\mathbf{W} = \mathcal{S}(\mathbf{X}, \omega_b) = \sigma_{\text{softmax}}\big(\mathbf{W}_\sigma + \mathbf{W}_\Delta\big). \qquad (12)$$

$$\mathbf{W}_\sigma = (\mathbf{X} - \mathbf{C}_b)^T \mathbf{Q}_b (\mathbf{X} - \mathbf{C}_b),$$

$$\mathbf{W}_\Delta = MLP_\Delta(X, \omega_b)$$

$$\mathbf{Q}_b = \mathbf{V}_b^T \boldsymbol{\Lambda}_b^0 \mathbf{V}_b$$

$$\left(\mathbf{V}_b | \mathbf{C}_b\right) = \mathbf{J}_b \left(\mathbf{V}_b^0 | \mathbf{C}_b^0\right)$$

Volume rendering based on backward warp

- Use warped 3D ray for volume rendering as in Dynamic NeRF framework.
  - **Backward warp** the ray at time t, then query color and density from canonical space
- Based on volume rendering, **3 reconstruction losses** are made.
  - **color, silhouette, 2D flow**
- The **color** c and opacity(=**silhouette**) o are given by

$$\mathbf{c}(\mathbf{x}^t) = \sum_{i=1}^{N} \tau_i \mathbf{c}^t \left(\mathcal{W}^{t,\leftarrow}\left(\mathbf{X}_i^t\right)\right), \quad o(\mathbf{x}^t) = \sum_{i=1}^{N} \tau_i,$$

N : #samples on the ray

$\tau_i = \prod_{j=1}^{i-1} p_j (1 - p_i) \longleftarrow$ **probability the object is visible to camera**

$p_i = \exp(-\sigma_i \delta_i) \longleftarrow$ **local opacity**

$\sigma_i = \sigma\left(\mathcal{W}^{t,\leftarrow}\left(\mathbf{X}_i^t\right)\right) \longleftarrow$ **differential probability that ray terminates**

Volume rendering based on backward warp

- expected 3D location X^*
  - τ_i can be considered as the probability of object existence at the sample location.

$$\mathbf{X}^*(\mathbf{x}^t) = \sum_{i=1}^{N} \tau_i \left(\mathcal{W}^{t,\leftarrow}(\mathbf{X}_i^t)\right), \qquad (4)$$

  - $\mathrm{X}^*(x^t)$ can be mapped to another time t' by **forward warping** $W^{t',\rightarrow}$, then projected to pixel
    - $\Pi^{t'}$ is the projection matrix of a pinhole camera model.

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

$$\mathbf{x}^{t'} = \Pi^{t'}\left(\mathcal{W}^{t,\rightarrow}(\mathbf{X}^*(\mathbf{x}^t))\right), \qquad (5)$$

$$\mathbf{P} = \underbrace{\begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsic parameters}} \underbrace{\left[\begin{array}{ccc|c} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_3 \end{array}\right]}_{\text{extrinsic parameters}}$$

- With projected pixels of two different frames, we can compute a **2D flow**.

$$\mathcal{F}\left(\mathbf{x}^t, t \rightarrow t'\right) = \mathbf{x}^{t'} - \mathbf{x}^t. \qquad (6)$$

Registration via Canonical Embeddings

- BANMo uses embeddings in canonical space for matching pixels in different t
- the embedding of a point X^* is computed by MLP_φ
  - MLP_φ is optimized to output embedding that matches to embedding of corresponding pixel.
- the embedding of a pixel x_t is computed by CNN
  - CNN is initialized with CSE(next slide).

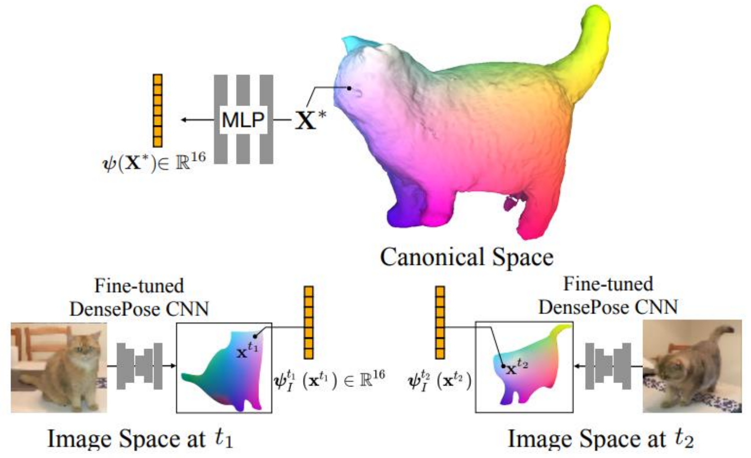


Figure 3. **Canonical Embeddings.** We jointly optimize an implicit function to produce canonical embeddings from 3D canonical points that match to the 2D DensePose CSE embeddings [28].

Preliminaries: Continuous Surface Embeddings

- universal networks for learning dense correspondences within and across different object categories(animals)

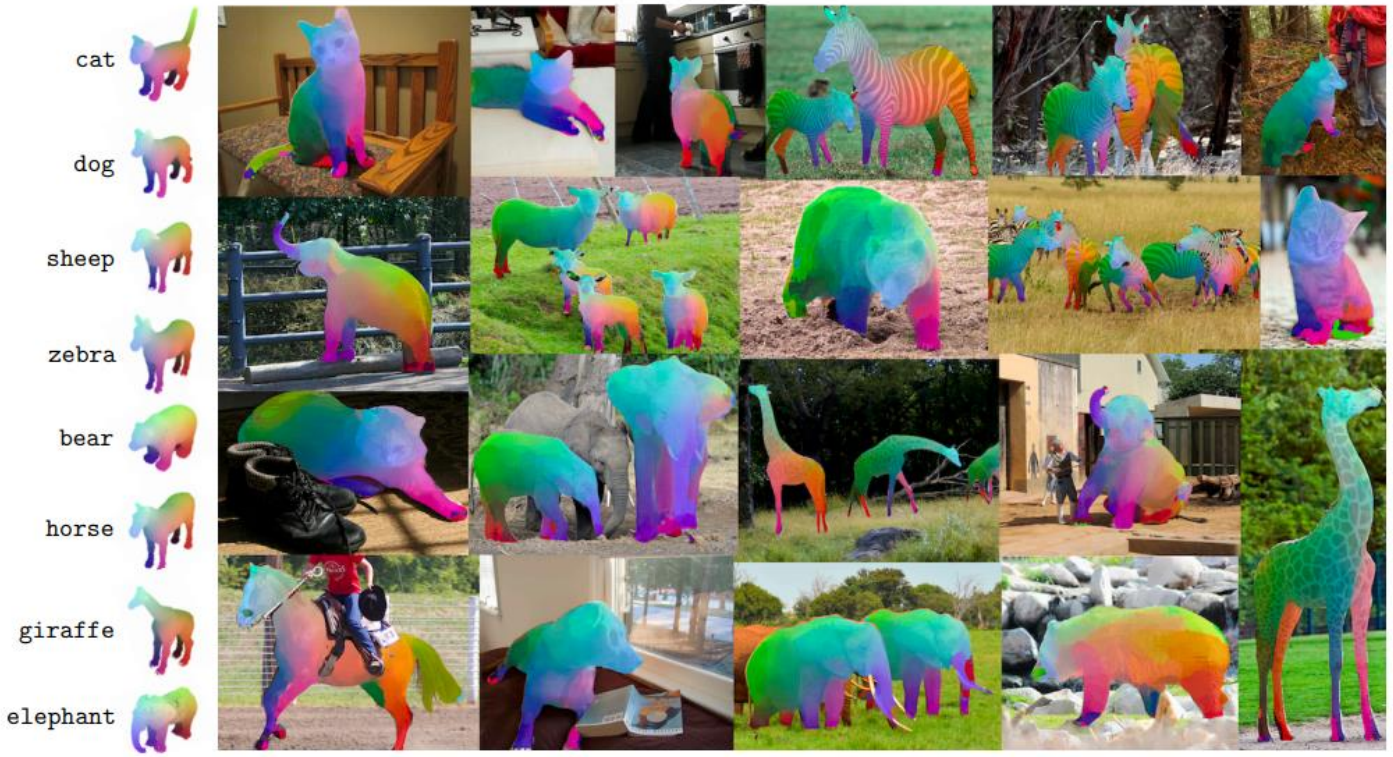


Figure 4: **Qualitative results on the DensePose-LVIS dataset** (single predictor for all classes).

Neverova, Natalia, et al. "Continuous surface embeddings." Advances in Neural Information Processing Systems 33 (2020): 17258-17270.

Registration via Canonical Embeddings

- Find matching 3D point $\hat{\mathrm{X}}^*$ to a 2D pixel x_t by **softmax descriptor matching**

$$\hat{\mathbf{X}}^*(\mathbf{x}^t) = \sum_{\mathbf{X} \in \mathbf{V}^*} \tilde{\mathbf{s}}^t(\mathbf{x}^t, \mathbf{X})\mathbf{X}, \qquad (13)$$

$$\hat{s}^t(x^t, X) = \sigma_{softmax}(\alpha_s \langle \varphi_I^t(x^t), \varphi(X^*) \rangle)$$

α_s : learnable scaling parameter
$\langle \quad \rangle$: cosine similiarity

- $\hat{\mathrm{X}}^*(x^t)$ is used for **registration loss**.

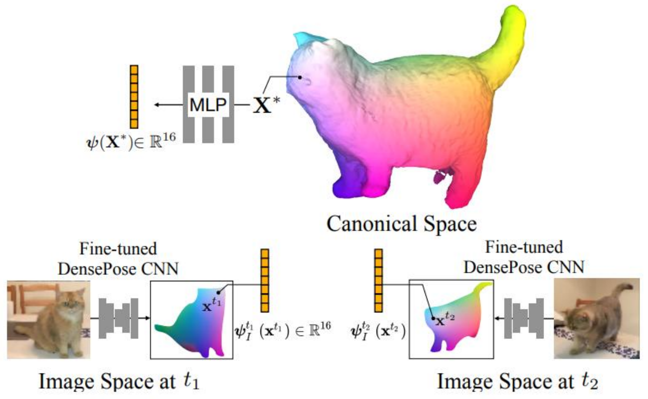


Figure 3. **Canonical Embeddings.** We jointly optimize an implicit function to produce canonical embeddings from 3D canonical points that match to the 2D DensePose CSE embeddings [28].

Loss

- Overall losses

$$\mathcal{L} = \underbrace{\left(\mathcal{L}_{\text{sil}} + \mathcal{L}_{\text{rgb}} + \mathcal{L}_{\text{OF}}\right)}_{\text{reconstruction losses}} + \underbrace{\left(\mathcal{L}_{\text{match}} + \mathcal{L}_{\text{2D-cyc}}\right)}_{\text{feature registration losses}} + \mathcal{L}_{\text{3D-cyc}}.$$

Loss: Reconstruction loss

- 3 reconstruction loss: **color, silhouette, 2D flow**
  - GT of silhouette and 2D flow are computed by off-the-shelf networks
    : PointRend, VCN-Robust

$$\mathcal{L}_{\text{rgb}} = \sum_{\mathbf{x}^t} \left\| \mathbf{c}(\mathbf{x}^t) - \hat{\mathbf{c}}(\mathbf{x}^t) \right\|^2, \; \mathcal{L}_{\text{sil}} = \sum_{\mathbf{x}^t} \left\| \mathbf{o}(\mathbf{x}^t) - \hat{\mathbf{s}}(\mathbf{x}^t) \right\|^2,$$

$$\mathcal{L}_{\text{OF}} = \sum_{\mathbf{x}^t, (t,t')} \left\| \mathcal{F}\left(\mathbf{x}^t, t \to t'\right) - \hat{\mathcal{F}}\left(\mathbf{x}^t, t \to t'\right) \right\|^2, \quad (14)$$

Loss: registration loss (1)

- enforces 3D point prediction $\hat{X}^*(x^t)$ from canonical embedding to match the prediction from backward warping

$$\mathcal{L}_{\text{match}} = \sum_{\mathbf{x}^t} \left\| \mathbf{X}^*(\mathbf{x}^t) - \hat{\mathbf{X}}^*(\mathbf{x}^t) \right\|_2^2 , \qquad (15)$$

$$\mathbf{X}^*(\mathbf{x}^t) = \sum_{i=1}^{N} \tau_i \left(\mathcal{W}^{t,\leftarrow}(\mathbf{X}_i^t) \right) , \qquad (4)$$

$$\hat{\mathbf{X}}^*(\mathbf{x}^t) = \sum_{\mathbf{X} \in \mathbf{V}^*} \tilde{\mathbf{s}}^t(\mathbf{x}^t, \mathbf{X})\mathbf{X}, \qquad (13)$$

Loss: registration loss (2)

- **2D cycle-consistency loss**
  - forces the projection of forward warping of $\hat{X}^*(x^t)$ to go back to x^t.

$$\mathcal{L}_{\text{2D-cyc}} = \sum_{\mathbf{x}^t} \left\| \Pi^t \left(\mathcal{W}^{t,\rightarrow}(\hat{\mathbf{X}}^*(\mathbf{x}^t)) \right) - \mathbf{x}^t \right\|_2^2 . \qquad (16)$$

$$\hat{\mathbf{X}}^*(\mathbf{x}^t) = \sum_{\mathbf{X} \in \mathbf{V}^*} \tilde{\mathbf{s}}^t(\mathbf{x}^t, \mathbf{X})\mathbf{X}, \qquad (13)$$

Loss: 3D cycle-consistency loss

- **3D cycle-consistency loss**
  - regularize forward, backward warping function $W^{t,\rightarrow}$, $W^{t,\leftarrow}$
  - backward warp a sampled 3D point in the camera space X_i^t into canonical space, then forward warp to camera space again.

$$\mathcal{L}_{\text{3D-cyc}} = \sum_i \tau_i \left\| \mathcal{W}^{t,\rightarrow}\left(\mathcal{W}^{t,\leftarrow}(\mathbf{X}_i^t, t), t\right) - \mathbf{X}_i^t \right\|_2^2, \quad (17)$$

$$\mathbf{X}^t = \mathcal{W}^{t,\rightarrow}(\mathbf{X}^*) = \mathbf{G}^t\left(\mathbf{J}^{t,\rightarrow}\mathbf{X}^*\right), \quad (7)$$

$$\mathbf{X}^* = \mathcal{W}^{t,\leftarrow}(\mathbf{X}^t) = \mathbf{J}^{t,\leftarrow}\left((\mathbf{G}^t)^{-1}\mathbf{X}^t\right), \quad (8)$$

Robust Optimization: Root pose initialization

- Train PoseNet to provide a rough per-frame initialization of root poses $\mathtt{G}^t$
- PoseNet takes CSE feature image as input and predict the initial root pose $\mathtt{G}_0^t$
    - PoseNet is trained with a synthetic dataset produced offline.
- BANMo only needs to compute a delta root pose via MLP

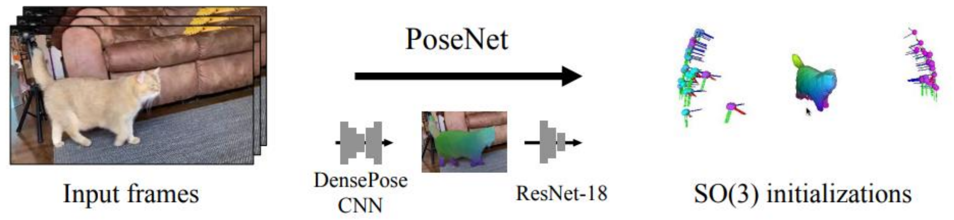


Figure 11. **Inference pipeline of PoseNet.** To initialize the optimization, we train a CNN PoseNet to predict root poses given a single image. PoseNet uses a DensePose-CNN to extract pixel features and decodes the pixel features into root pose predictions with a ResNet-18. We visualize the initial root poses on the right. Cyan color represents earlier time stamps and magenta color represent later timestamps.

$$\mathbf{G}^t = \mathbf{MLP}_{\mathbf{G}}(\boldsymbol{\omega}_r^t)\mathbf{G}_0^t. \tag{18}$$

$$\mathbf{G}_0^t = \text{PoseNet}(\boldsymbol{\psi}_I^t)$$

Robust Optimization: Root pose initialization

- Training pipeline

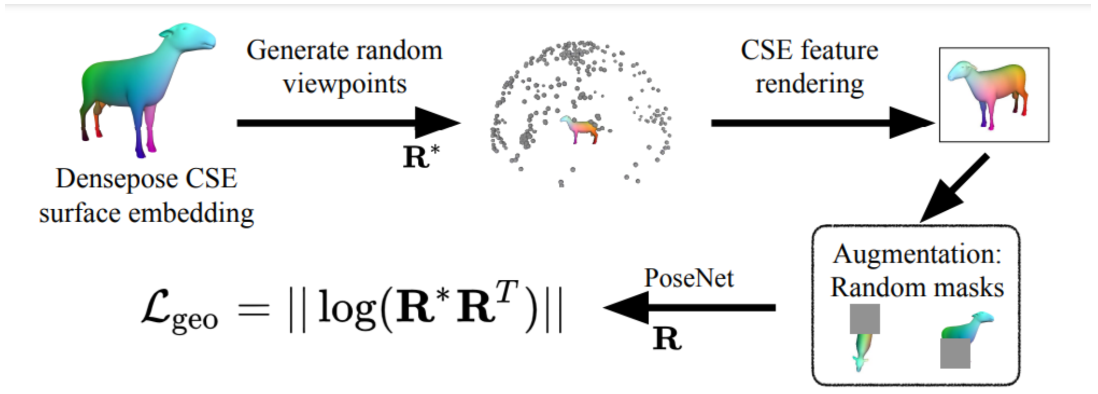

Generate random viewpoints
$\mathbf{R}^*$
CSE feature rendering
Densepose CSE surface embedding
Augmentation: Random masks
PoseNet
$\mathbf{R}$
$\mathcal{L}_{\text{geo}} = ||\log(\mathbf{R}^*\mathbf{R}^T)||$

Figure 12. **Training pipeline of PoseNet.** To train PoseNet, we use DesePose CSE surface embeddings, which is pertained on 2D annotations of human and quadruped animals. We first generate random viewpoints on a sphere that faces the origin. Then we render surface embeddings as 16-channel images. We further augment the feature images with random adversarial masks to improve the robustness to occlusions. Finally, the rotations predicted by PoseNet are compared against the ground-truth rotations with geodesic distance.

Robust Optimization: Active Sampling

- easy-to-hard curriculum sampling strategy
  - At the early iterations, randomly sample a batch of pixels for volume rendering
  - At the same time, optimize 5-layer MLP to represent the **uncertainty** over (x, y, t)
  - which means, finding time-specific pixels hard to optimize
  - After a half of the optimization steps, select additional active samples of **high uncertainty**.

Experiments : Quantitative

- Dataset
  - AMA(Articulated Mesh Animation) human dataset
    - 10 sets of multi-view videos captured by 8 synchronized camera
    - select 2 sets: swing, samba
  - Animated Objects dataset
    - download free animated 3D models from TurboSquid : eagle, human hands
    - render them from different camera trajectories
- Metric: 3D Chamfer distances
  - Chamfer distances between the GT mesh points and the estimated mesh points
  - align two shapes via ICP(Iterative Closest Point) before comparison

$$d_{\mathrm{CD}}\left(\mathbf{S}^*, \hat{\mathbf{S}}\right) = \Big(\sum_{x \in \mathbf{S}^*} \min_{y \in \hat{\mathbf{S}}} \|x-y\|_2^2\Big) + \Big(\sum_{y \in \hat{\mathbf{S}}} \min_{x \in \mathbf{S}^*} \|x-y\|_2^2\Big).$$

Experiments : Quantitative

- Compare with ViSER[1] and Nerfies[2]

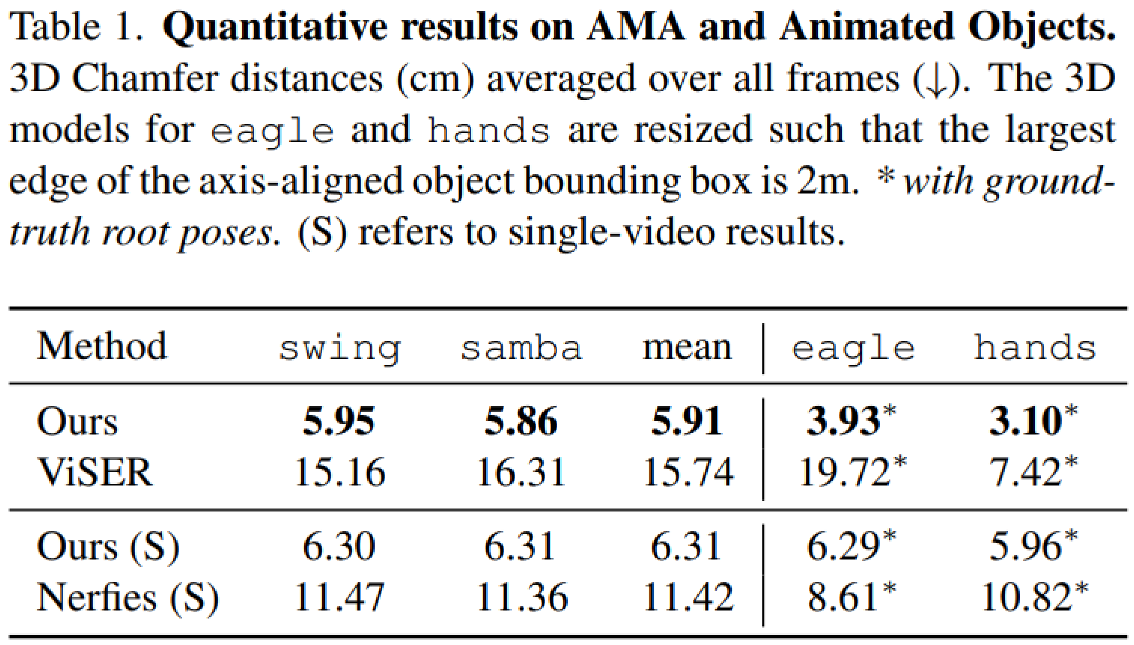


Table 1. **Quantitative results on AMA and Animated Objects.** 3D Chamfer distances (cm) averaged over all frames (↓). The 3D models for `eagle` and `hands` are resized such that the largest edge of the axis-aligned object bounding box is 2m. * *with ground-truth root poses.* (S) refers to single-video results.

Method	`swing`	`samba`	mean	`eagle`	`hands`
Ours	**5.95**	**5.86**	**5.91**	**3.93***	**3.10***
ViSER	15.16	16.31	15.74	19.72*	7.42*
Ours (S)	6.30	6.31	6.31	6.29*	5.96*
Nerfies (S)	11.47	11.36	11.42	8.61*	10.82*

[1] Yang, Gengshan, et al. "ViSER: Video-Specific Surface Embeddings for Articulated 3D Shape Reconstruction." Advances in Neural Information Processing Systems 34 (2021).
[2] Park, Keunhong, et al. "Nerfies: Deformable neural radiance fields." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021.

Experiments: Qualitative

- Compare with ViSER[1] and Nerfies[2]

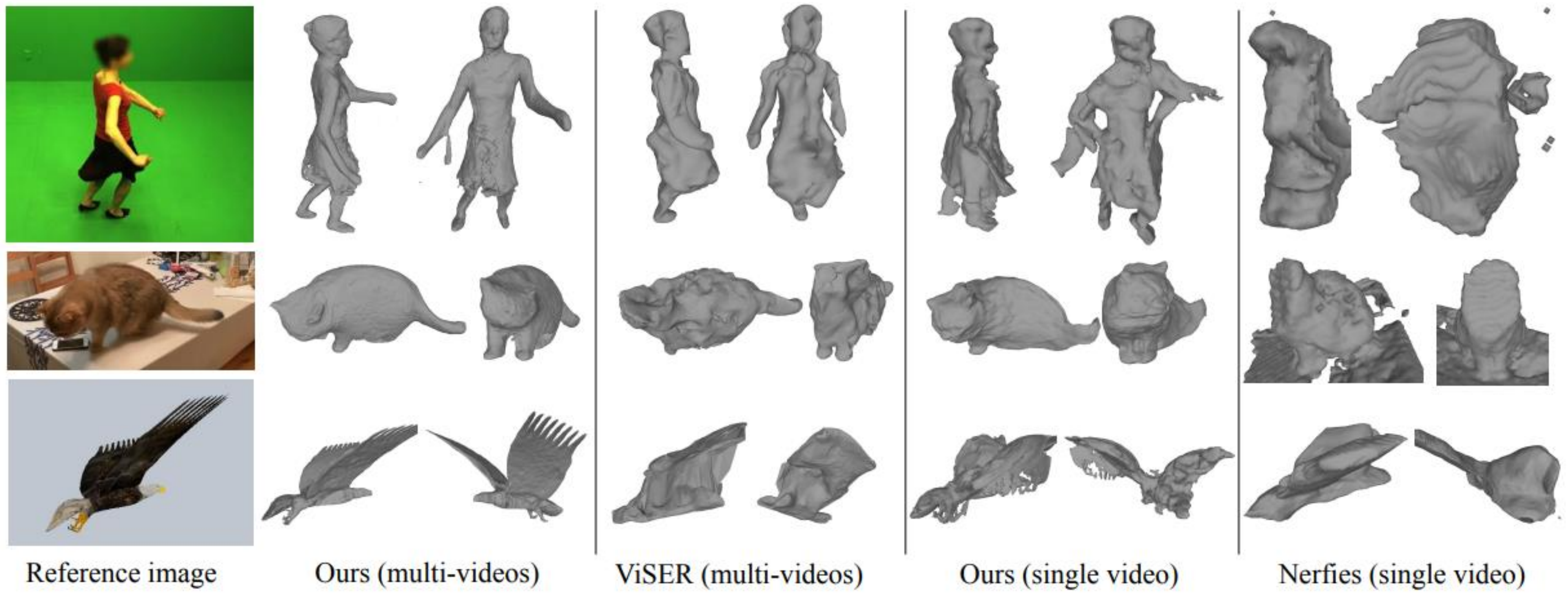


Figure 4. **Qualitative comparison of our method with prior art [31,57].** From top to bottom: AMA's `samba`, `casual-cat`, `eagle`.

Ablation Study: Root pose initialization

- the effect of **PoseNet** for root pose initialization
  - without it, the root poses collapsed to a degenerate solution after optimization

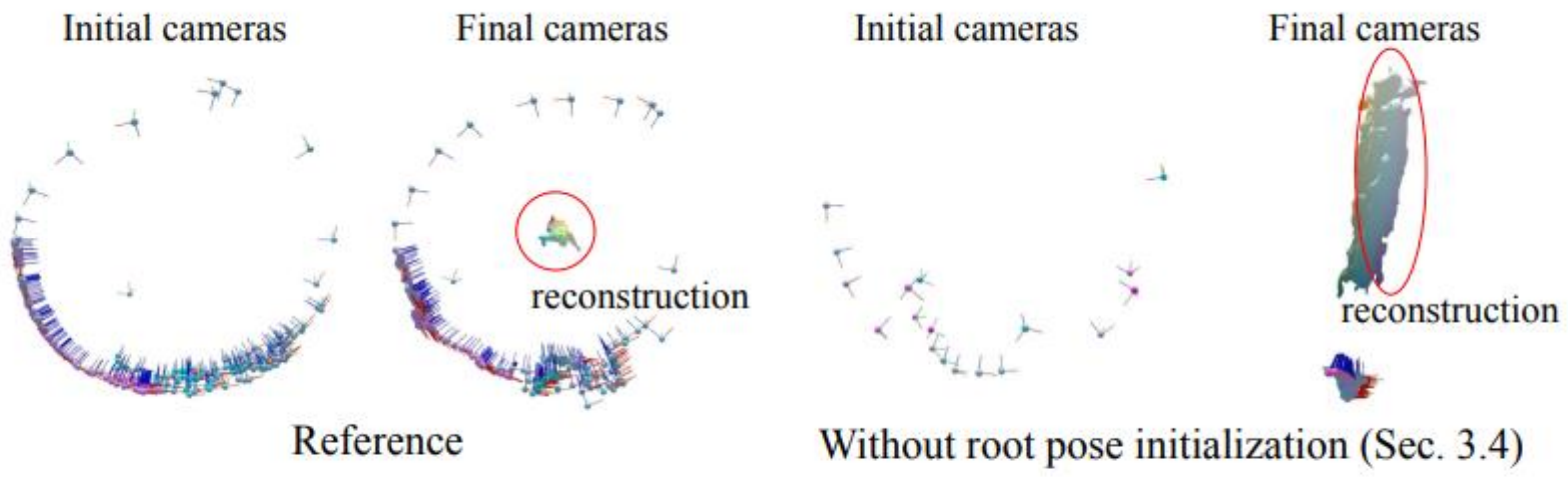


Figure 6. **Diagnostics of root pose initialization (Sec.3.4).** With randomly initialized root poses, the estimated poses (on the right) collapsed to a degenerate solution, causing reconstruction to fail.

Abltation Study: Registration

- losses for exploiting multiple videos
  - within video: 2D flow
  - across video: matching with canonical embeddings

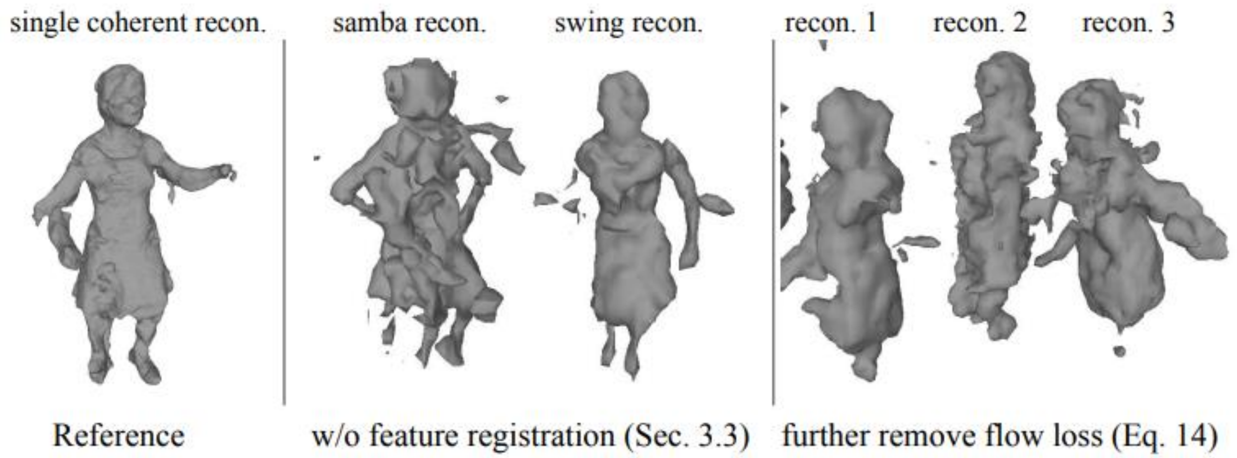


Figure 7. **Diagnostics of registration (Sec. 3.3).** Without canonical embeddings (middle) or flow loss (right), our method fails to register frames to a canonical model, creating ghosting effects.

Ablation Study: Active Sampling

- without it, slower convergence and inaccurate geometry

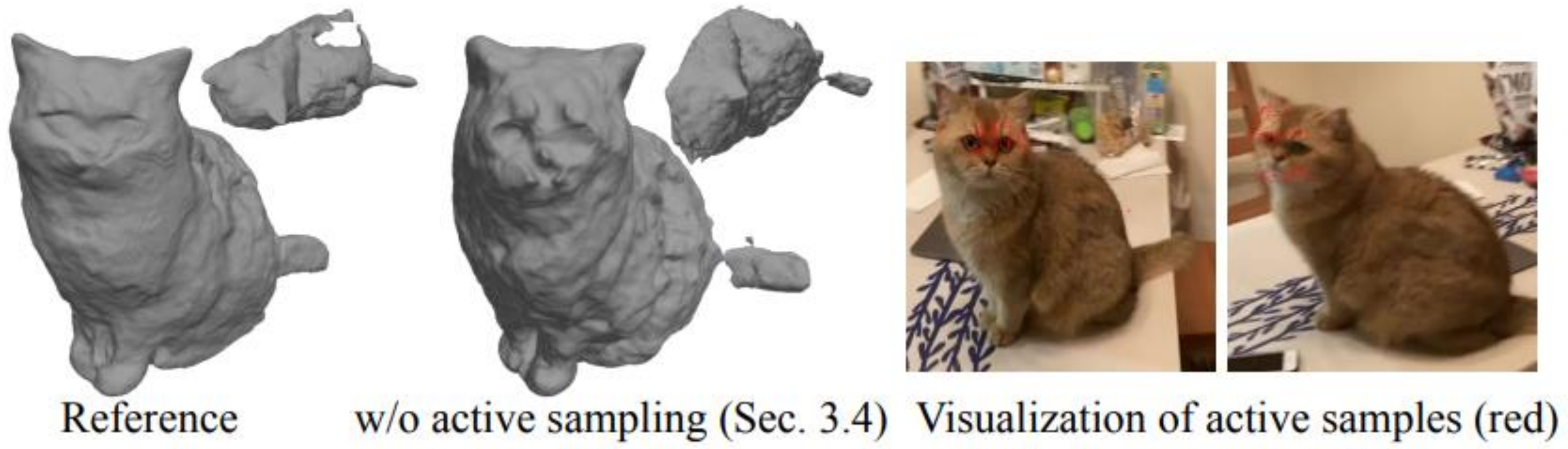

Reference
w/o active sampling (Sec. 3.4)
Visualization of active samples (red)

Figure 8. **Diagnostics of active sampling over (x, y) (Sec. 3.4).** With no active sampling, our method converges slower and misses details (such as ears and eyes). Active samples focus on face and boundaries pixels where the color reconstruction errors are higher.

Ablation Study: Deformation modeling

- neural blend skinning model
  - backward swap each samples on the ray according to skinning model
  - Nerfies used SE(3), D-NeRF & NSFF used translation

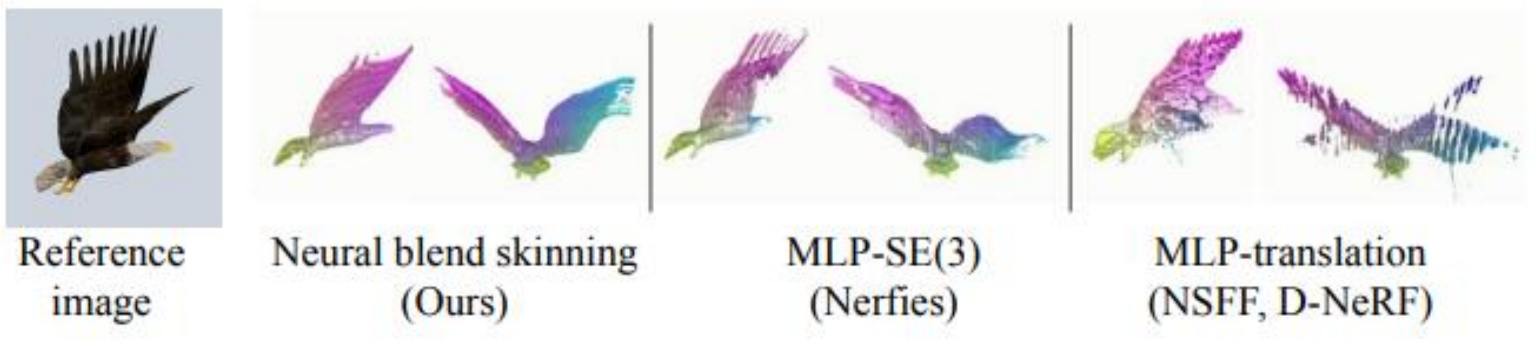


Figure 9. **Diagnostics of deformation modeling (Sec.3.2).** Replacing our neural blend skinning with MLP-SE(3) results in less regular deformation in the non-visible region. Replacing our neural blend skinning with MLP-translation as in NSFF and D-Nerf results in reconstructing ghosting wings due to significant motion.

Ablation Study: Leveraging more videos

- Reconstruction quality vs #input videos (compare with Nerfies)

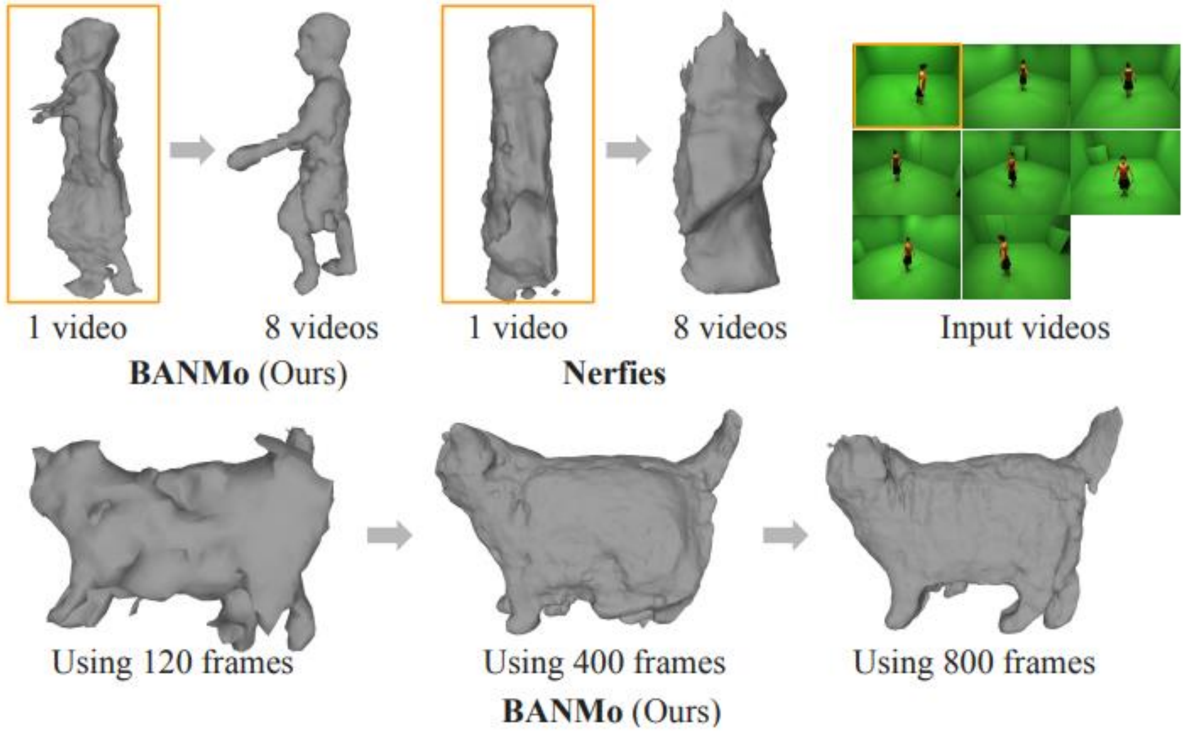


Figure 5. **Reconstruction completeness vs number of input videos and video frames.** BANMo is capable of registering more input videos if they are available, improving the reconstruction.

Ablation Study: Motion re-targeting

- re-target pretrained BANMo 3D model to new driving frame
  - freeze the shared model parameters of the cat model
  - only optimize the video-specific and frame-specific codes
    - root pose code w_r^t, body pose code w_b^t, environment lighting code w_e^t

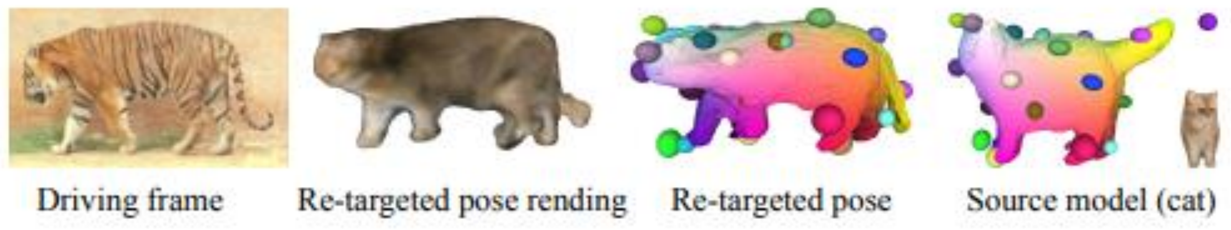


Figure 10. **Motion re-targeting from a pre-optimized cat model to a tiger.** Color coded by point locations in the canonical space.