BANMo: Building Animatable 3D Neural Models from Many Casual Videos:
SUPPLEMENTARY MATERIALS

A. Notations

We refer readers to a list of notations in Tab. 5 and a list of learnable parameters in Tab. 6. We compare with Nerfies and ViSER and summarize the differences in Tab. 1.

Table 1. **Difference between Nerfies, ViSER, and BANMo.**

Method	shape	deformation	registration
Nerfies	implicit	dense SE(3)	photometric
ViSER	mesh	control points	self-supervised feature
BANMo	implicit	control points	pre-trained feature

B. Method details

B.1. Root Pose Initialization

As discussed in Sec. 3.4, to make optimization robust, we train a image CNN (denoted as PoseNet) to initialize root body transforms $\mathbf{G}^t$ that aligns the camera space of time t to the canonical space of CSE, as shown in Fig. 1.

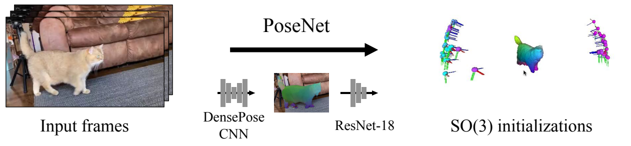

Figure 1. **Inference pipeline of PoseNet.** To initialize the optimization, we train a CNN PoseNet to predict root poses given a single image. PoseNet uses a DensePose-CNN to extract pixel features and decodes the pixel features into root pose predictions with a ResNet-18. We visualize the initial root poses on the right. Cyan color represents earlier time stamps and magenta color represent later timestamps.

Preliminary DensePose CSE [1, 2] trains pixel embeddings ψ_I and surface feature embeddings ψ for humans and quadruped animals using 2D keypoing annotations. It represents surface embeddings by a canonical surface with N vertices and vertex features $\psi \in \mathbb{R}^{N\times 16}$. A SMPL mesh is used for humans, and a sheep mesh is used for quadraped animals. The embeddings are trained such that given an pixel feature, a 3D point on the canonical surface can be uniquely located via feature matching.

Naive PnP solution Given 2D-3D correspondences provided by CSE, one way to solve for $\mathbf{G}^t$ is to use perspective-n-points (PnP) algorithm assuming objects are rigid. However, the PnP solution suffers from catastrophic failures due to the non-rigidity of the object, which motivates our PoseNet solution. By training a feed-forward network with data augmentations, our PoseNet solution produces fewer gross errors than the naive PnP solution.

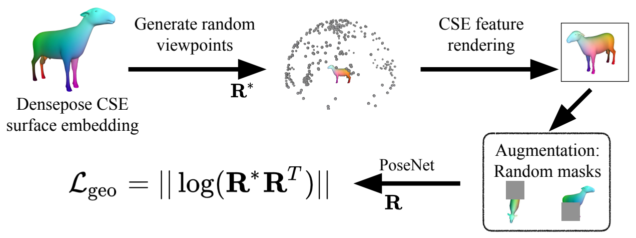
Generate random viewpoints
$\mathbf{R}^*$
CSE feature rendering
Densepose CSE surface embedding
Augmentation: Random masks
PoseNet
$\mathbf{R}$
$\mathcal{L}_{\text{geo}} = ||\log(\mathbf{R}^*\mathbf{R}^T)||$

Figure 2. **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.

Synthetic dataset genetarion. We train separate PoseNet, one for human, and one for quadruped animals. The training pipeline is shown in Fig. 2. Specifically, we render surface features as feature images $\psi_{\text{rnd}} \in \mathbb{R}^{112\times112\times16}$ given viewpoints $\mathbf{G}^* = (\mathbf{R}^*, \mathbf{T}^*)$ randomly generated on a unit sphere facing the origin. We apply occlusion augmentations [5] that randomly mask out a rectangular region in the rendered feature image and replace with mean values of the corresponding feature channels. The random occlusion augmentation forces the network to be robust to outlier inputs, and empirically helps network to make robust predictions in presence of occlusions and in case of out-of-distribution appearance.

Loss and inference. We use the geodesic distance between the ground-truth and predicted rotations as a loss to update PoseNet,

$$\mathcal{L}_{\text{geo}} = ||\log(\mathbf{R}^*\mathbf{R}^T)||, \quad \mathbf{R} = \text{PoseNet}(\psi_{\text{rnd}}), \tag{1}$$

where we find learning to predict rotation is sufficient for initializing the root body pose. In practice, we set the initial object-to-camera translation to be a constant $\mathbf{T} = (0,0,3)^T$. We run pose CNN on each test video frame to obtain the initial root poses $\mathbf{G}_0^t = \big(\mathbf{R}, \mathbf{T}\big)$, and compute a delta root pose with the root pose MLP:

$$\mathbf{G}^t = \mathbf{MLP}_{\mathbf{G}}(\omega_r^t)\mathbf{G}_0^t. \tag{2}$$

B.2. Active sampling over (x, y, t)

Inspired by iMAP [6], our ray sampling strategy follows an easy-to-hard curriculum. At the early iterations, we randomly sample a batch of N^p pixels for volume rendering and compute reconstruction losses. At the same time, we optimize a compact 5-layer MLP function to represent the uncertainty over the image coordinate and frame index: $\hat{\mathbf{U}}(x, y, t) = \mathbf{MLP}_{\mathbf{U}}(x, y, t)$. The uncertainty MLP is optimized by comparing against the color reconstruction errors in the current forward step:

$$\mathcal{L}_{\mathbf{U}} = \sum_{\mathbf{x},t} \left\| \mathcal{L}_{\text{rgb}}(\mathbf{x}^t) - \hat{\mathbf{U}}(\mathbf{x}^t) \right\|. \tag{3}$$

Note that the gradient from $\mathcal{L}_{\mathbf{U}}$ to $\mathcal{L}_{\text{rgb}}(\mathbf{x}^t)$ is stopped such that $\mathcal{L}_{\mathbf{U}}$ does not generate gradients to parameters besides $\mathbf{MLP}_{\mathbf{U}}$. After 40% of the optimization steps, we replace half of the samples with *active* samples from pixels with high uncertainties. To do so, we randomly sample $N^{a'} = 24576$ pixels, and evaluate their uncertainties by passing their coordinates and frame indices to $\mathbf{MLP}_{\mathbf{U}}$. Active samples dramatically improves reconstruction fidelity, as shown in Fig. 6.

B.3. Optimization details

Canonical 3D grid. As mentioned in Sec 3.3, we define a canonical 3D grid $\mathbf{V}^* \in \mathbb{R}^{20\times20\times20}$ to compute the matching costs between pixels and canonical space locations. The canonical grid is centered at the origin and axis-aligned with bounds $[x_{\min}, x_{\max}]$, $[y_{\min}, y_{\max}]$, and $[z_{\min}, z_{\max}]$. The bounds are initialized as loose bounds and are refined during optimization. For every 200 iterations, we update the bounds of the canonical volume as an approximate bound of the object surface. To do so, we run marching cubes on a 64^3 grid to extract a surface mesh and then set L as the axis-aligned (x, y, z) bounds of the extracted surface.

Near-far planes. To generate samples for volume rendering, we dynamically compute the depth of near-far planes (d_n^t, d_f^t) of frame t at each iterations of the optimization. To do so, we compute the projected depth of the canonical surface points $d_i^t = (\Pi^t \mathbf{G}^t \mathbf{X}_i^*)_2$. The near plane is set as $d_n^t = \min(d_i) - \epsilon_L$ and the far plane is set as $d_f^t = \max(d_i) + \epsilon_L$, where $\epsilon_L = 0.2\big(\max(d_i) - \min(d_i)\big)$. To avoid the compute overhead, we approximate the surface with an axis-aligned bounding box with 8 points.

Hyper-parameters. We use 1cycle learning rate scheduler, which warms-up with a low learning rate to the maximum, and anneals the learning rate to a final learning rate. We apply $lr_{init} = 2e-5$, $lr_{max} = 5e-4$, $lr_{final} = 1e-4$.

Table 2. Table of hyper-parameters.

Name	Value	Description
B	25	Number of bones
N	128	Sampled points per ray
N^p	6144	Sampled rays per batch
(H, W)	(512,512)	Resolution of observed images

We refer readers to a complete list of hyper-parameters in Tab. 2.

Multi-stage optimization The final optimization takes three stages, where the parameters being updated and the loss functions being used are different. The first stage uses all the losses and updates all the parameters described in the paper. Typically, the first stage already produces 3D reconstructions with good shape and deformation. The goal of the stage 2 is to improve the articulations (e.g., to correctly articulate the crossing legs for `cat-pikachiu`) with coordinate gradient descent, where we turn off the reconstruction losses and only use the 2D cycle consistency loss to update the articulation parameters while keeping shape parameters fixed. Finally, stage 3 improves details of the geometry by using a larger weight for the color reconstruction loss.

Experiment details When running Nerfies on AMA and animated objects, we found using RGB reconstruction loss does not produce meaningful results possibly due to the homogeneous background color. To improve Nerfies results, we provide it with ground-truth object silhouettes, and optimize a carefully balanced RGB+silhouette loss [8].

C. Additional results

C.1. SFM root pose initialization

COLMAP [3,4] failed to converge when focused on the deformable object due to violation of rigidity, leading to very few successful registrations (18 over 811 images registered on `casual-cat`). A recent end-to-end method, DROID-SLAM [7], registered all the images but the accuracy is low compared to PoseNet, as shown in Tab. 3. We also tried SFM to estimate and compensate for the camera motion (using background as rigid anchor), but this did not help to recover the pose of the object due to its global movement w.r.t. to the background.

C.2. More ablation study

In Sec. 4.3, we presented qualitative results of diagnostics experiments. In Tab. 4, we report the results of other ablations followed by analysis.

Number and location of bones As shown in the first group of Tab. 4 and Fig. 3, using too few bones fails to re-

Table 3. **Evaluation on root pose prediction.** Mean and standard deviation of the rotation error (°) over all frames (↓). We use BANMo-optimized poses as ground-truth. Rotations are aligned to the ground-truth by a global rotation under chordal L2 distance.

Method	`c-cat`	`c-human`	`ama-human`
CSE-PoseNet	**18.6**±16.2	**12.8**±8.9	**11.8**±17.4
DROID-SLAM	65.5 ± 44.5	55.8 ± 39.2	83.6 ± 50.5

Table 4. **Results on AMA swing and samba.** 3D Chamfer distance (cm, ↓) and F-score (%, ↑) averaged over all frames.

Method	CD	F@1%	F@2%	F@5%
number-bone=4	9.88	28.1	52.4	84.1
number-bone=9	9.08	31.2	56.4	86.8
number-bone=16	**9.02**	**31.8**	**57.2**	87.2
number-bone=25	9.08	**31.8**	57.0	87.1
–w/o in-surface loss	9.14	29.9	54.8	86.7
–quad. embedding	9.70	29.8	54.2	85.4
number-bone=64	9.18	31.1	56.6	**87.5**
number-bone=100	9.11	31.4	56.7	87.3
pose error ϵ=20°	**8.75**	**30.9**	**57.0**	**88.1**
pose error ϵ=50°	8.91	29.8	56.1	**88.1**
pose error ϵ=90°	9.91	28.4	54.8	85.7
coverage=90° (2 vids)	10.61	29.3	54.3	84.1
coverage=180° (4 vids)	**8.94**	**33.0**	**59.8**	**87.9**
coverage=270° (6 vids)	9.09	29.8	56.1	87.6
active-sample=0%	9.63	29.1	53.7	85.8
active-sample=25%	**8.60**	**32.3**	**57.9**	**88.0**
active-sample=50%	9.14	29.9	54.8	86.7

cover all body parts due to over-regularization. Using more than 16 bones produces good reconstructions, but consumes more memory when computing skinning weights. Enforcing them to stay close to the surface with a sinkhorn divergence loss improves the results (Tab. 4, L16-17).

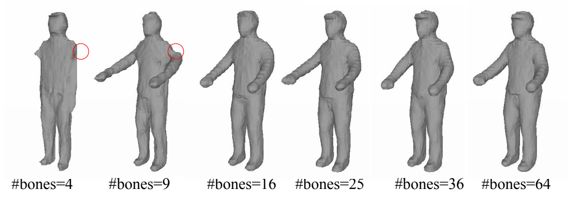

Figure 3. **Sensitivity to number of bones.**

Sensitivity to incorrect initial pose We inject different levels of Gaussian noise into the initial poses, leading to average rotation errors $\epsilon \in \{20, 50, 90\}°$. As shown in the second group of Tab. 4, BANMo is stable up to 50° rotation error.

Pre-trained embeddings Pre-trained embeddings help BANMo outperform Nerfies, but it is not crucial given good initial root poses ($\epsilon = 12.8 \pm 8.9°$). As shown in Tab. 4, using embeddings pre-trained for quadruped animals for human optimization produces slightly worse results.

How much data are needed? To reconstruct a complete shape, BANMo requires all object surface to be visible from at least one frame. Beside completeness, more videos allows to estimate better skinning weights and a more regular motion. We evaluate view coverage in the third group of Tab. 4.

Importance sampling We use active sampling to avoid sampling from uninformative frames and pixels. It consistently improves reconstruction results as shown in the last group of Tab. 4.

Bone re-initialization We qualitatively evaluate the effect of rest bone re-initialization, which re-initializes bone parameters according to the current estimation of shape. As shown in Fig. 4, without re-initializing the bones, the optimization may stuck at bad local optima and the final reconstruction may become less accurate.

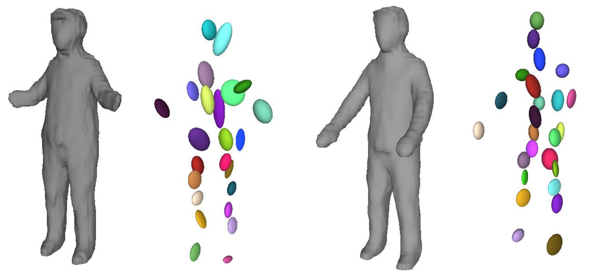

Figure 4. **Effect of bone re-initialization.** We find it important to re-initialize rest bone parameters after finding a better approximation of object geometry.

Delta skinning weights We qualitatively evaluate the effect of delta skinning weights. As shown in Fig. 5, without learning a delta skinning weights specific to each 3D point, the reconstructed shape and motion may be over-regularized by the 3D Gaussians.

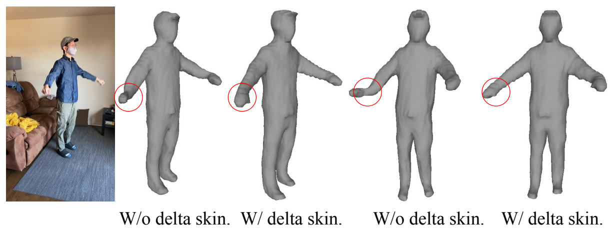

Figure 5. **Effect of delta skinning weights.** We find it important to learn a point-specific delta skinning weight function to reconstruction motions in high-quality.

Active sampling. We show the effect of active sampling on a `casual-cat` video (Fig. 6): removing it results in slower convergence and inaccurate geometry.

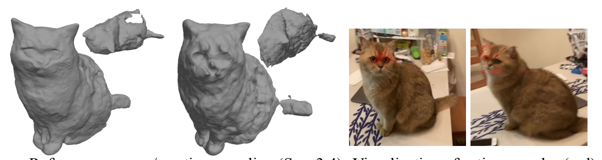
Reference
w/o active sampling (Sec. 3.4)
Visualization of active samples (red)

Figure 6. **Diagnostics of active sampling over** (x, y)**.** 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.

C.3. Qualitative results

We refer readers to our supplementary webpage for complete qualitative results.

Table 5. Table of notations.

Symbol	Description
	Index
t	Frame index, $t \in \{1, \ldots, T\}$
b	Bone index $b \in \{1, \ldots, B\}$ in neural blend skinning
i	Point index $b \in \{1, \ldots, N\}$ in volume rendering
	Points
$\mathbf{x}$	Pixel coordinate **x** = (x,y)
$\mathbf{X}^t$	3D point locations in the frame t camera coordinate
$\mathbf{X}^*$	3D point locations in the canonical coordinate
$\hat{\mathbf{X}}^*$	Matched canonical 3D point locations via canonical embedding
	Property of 3D points
$\mathbf{c} \in \mathbb{R}^3$	Color of a 3D point
$\sigma \in \mathbb{R}$	Density of a 3D point
$\boldsymbol{\psi} \in \mathbb{R}^{16}$	Canonical embedding of a 3D point
$\mathbf{W} \in \mathbb{R}^B$	Skinning weights of assigning a 3D point to B bones
	Functions on 3D points
$\mathcal{W}^{t,\leftarrow}(\mathbf{X}^t)$	Backward warping function from $\mathbf{X}^t$ to $\mathbf{X}^*$
$\mathcal{W}^{t,\rightarrow}(\mathbf{X}^*)$	Forward warping function from $\mathbf{X}^*$ to $\mathbf{X}^t$
$\mathcal{S}(\mathbf{X}, \omega_b)$	Skinning function that computes skinning weights of $\mathbf{X}$ under body pose ω_b
	Rendered and Observed Images
$\mathbf{c}/\hat{\mathbf{c}}$	Rendered/observed RGB image
$\mathbf{o}/\hat{\mathbf{s}}$	Rendered/observed object silhouette image
$\mathcal{F}/\hat{\mathcal{F}}$	Rendered/observed optical flow image

Table 6. Table of learnable parameters.

Symbol	Description
	Canonical Model Parameters
$\mathbf{MLP_c}$	Color MLP
$\mathbf{MLP}_{\text{SDF}}$	Shape MLP
$\mathbf{MLP}_{\boldsymbol{\psi}}$	Canonical embedding MLP
	Deformation Model Parameters
$\mathbf{\Lambda}^0 \in \mathbb{R}^{3\times3}$	Scale of the bones in the "zero-configuration" (diagonal matrix).
$\mathbf{V}^0 \in \mathbb{R}^{3\times3}$	Orientation of the bones in the "zero-configuration".
$\mathbf{C}^0 \in \mathbb{R}^{3}$	Center of the bones in the "zero-configuration".
$\mathbf{MLP}_{\mathbf{\Delta}}$	Delta skinning weight MLP
$\mathbf{MLP_G}$	Root pose MLP
$\mathbf{MLP_J}$	Body pose MLP
	Learnable Codes
$\boldsymbol{\omega}_b^* \in \mathbb{R}^{128}$	Body pose code for the rest pose
$\boldsymbol{\omega}_b^t \in \mathbb{R}^{128}$	Body pose code for frame t
$\boldsymbol{\omega}_r^t \in \mathbb{R}^{128}$	Root pose code for frame t
$\boldsymbol{\omega}_e^t \in \mathbb{R}^{64}$	Environment lighting code for frame t, shared across frames of the same video
	Other Learnable Parameters
$\boldsymbol{\psi}_I$	CNN pixel embedding initialized from DensePose CSE
α_s	Temperature scalar for canonical feature matching
β	Scale parameter that controls the solidness of the object surface
$\Pi \in \mathbb{R}^{3\times3}$	Intrinsic matrix of the pinhole camera model

References

[1] Natalia Neverova, David Novotny, Vasil Khalidov, Marc Szafraniec, Patrick Labatut, and Andrea Vedaldi. Continuous surface embeddings. In *NeurIPS*, 2020. 1

[2] Natalia Neverova, Artsiom Sanakoyeu, Patrick Labatut, David Novotny, and Andrea Vedaldi. Discovering relationships between object categories via universal canonical maps. In *CVPR*, 2021. 1

[3] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 2

[4] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[5] Krishna Kumar Singh and Yong Jae Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *ICCV*, 2017. 1

[6] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew Davison. iMAP: Implicit mapping and positioning in real-time. In *ICCV*, 2021. 2

[7] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[8] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *NeurIPS*, 2020. 2