CAMM: Building Category-Agnostic and Animatable 3D Models from Monocular Videos Supplementary Material

Tianshu Kuai Akash Karthikeyan Yash Kant Ashkan Mirzaei Igor Gilitschenski
University of Toronto

A. Summary

In Section B, we present additional details on the method. Section C provides additional details on the loss functions and optimization process. In Section D, we show more experimental results on each dataset. In Section E, we show examples of learned anchors and their corresponding associations. Section F provides additional discussions on the proposed two-stage optimization pipeline. Section G and Section H include the discussions on our method's limitations and potential negative impact. In Section I, we show examples of failure cases. In Section J, we provide additional details on our iiwa dataset. Finally, we provide a list of the important variables used in the paper along with their state space and descriptions in Section K.

B. Method Details

B.1. Canonical representation

We use the same canonical appearance, shape and feature representations as in BANMo [18], where the color $\mathbf{c} \in \mathbb{R}^3$, the Signed Distance Function (SDF) value $\mathbf{v} \in \mathbb{R}$, and the canonical feature $\boldsymbol{\phi} \in \mathbb{R}^{16}$ of a point $\mathbf{x} \in \mathbb{R}^3$ in the canonical space are given by separate Multi-Layer Perception (MLP) Networks $\mathcal{F}_{\mathbf{C}}$, $\mathcal{F}_{\mathbf{S}}$, and $\mathcal{F}_{\boldsymbol{\phi}}$, respectively:

$$\begin{aligned}\mathbf{c} &= \mathcal{F}_{\mathbf{C}}(\mathbf{x}, \mathbf{d}, \boldsymbol{\psi}_l)\\ \mathbf{v} &= \mathcal{F}_{\mathbf{S}}(\mathbf{x})\\ \boldsymbol{\phi} &= \mathcal{F}_{\boldsymbol{\phi}}(\mathbf{x}),\end{aligned} \tag{1}$$

where $\mathbf{d} \in \mathbb{R}^2$ is the viewing direction and $\boldsymbol{\psi}_l \in \mathbb{R}^{64}$ is a learnable latent code that captures the environment illumination conditions following [9]. Similar to [13, 19], we compute the density $\boldsymbol{\sigma} \in \mathbb{R}$ in the canonical space as:

$$\boldsymbol{\sigma} = \Psi_\beta(\mathbf{v}), \tag{2}$$

where Ψ_β is cumulative of a Laplacian distribution with zero mean and learnable scale β, and $\mathbf{v}$ is the value of the SDF at the point $\mathbf{x}$.

To perform volumetric rendering for the pixel of interest $\mathbf{x}_I^t \in \mathbb{R}^2$ at time t, we first sample N points along the camera ray corresponding to the pixel in deformed space. As our implicit representations of the shape and appearance are defined in canonical space, we first transform the sampled points back to the canonical space [11, 18] using the learned backward kinematics. Let the i-th point along the canonical space camera ray be $\mathbf{x}_i$, and its corresponding color and density queried from the MLP networks be $\mathbf{c}_i$ and $\boldsymbol{\sigma}_i$, we then follow the volumetric rendering process in NeRF [10] to get the rendered color at the pixel of interest $\mathbf{c}_r$ as:

$$\begin{aligned}\mathbf{c}_r &= \sum_{i=1}^{N} T_i(1 - \exp(-\boldsymbol{\sigma}_i \delta_i))\mathbf{c}_i\\ T_i &= \exp\left(-\sum_{j=1}^{i-1} \boldsymbol{\sigma}_j \delta_j\right),\end{aligned} \tag{3}$$

where T_i is the accumulated transmittance between the camera center to the i-th sampled point, δ_i is the interval between consecutive sampled points.

B.2. Recovering proper kinematic chain

In this section, we provide additional details on how to recover a proper and connected kinematic chain after all the joints in the canonical space are transformed into the deformed space using unconstrained transformations.

Given the canonical kinematic chain as $\mathbf{P} = \{\mathbf{p}_i \mid i = 1, \ldots, n_j\}$, and the initial optimization stage's learned unconstrained anchor transformations $\hat{\mathbf{T}}^t$ from the canonical space to the deformed space, we can then apply $\hat{\mathbf{T}}^t$ to transform the canonical kinematic chain joints $\mathbf{P}$ to the deformed space as:

$$\hat{\mathbf{P}}^t = \mathbf{C}^t \sum_{i=1}^{n_a} (\mathbf{w}_{a_i} \hat{\mathbf{T}}_i^t)\mathbf{P}, \tag{4}$$

where $\hat{\mathbf{P}}^t = \{\hat{\mathbf{p}}_i \mid i = 1, \ldots, n_p\}$ is the set of unconstrained kinematic chain joints in the deformed space at time t. These unconstrained joints cannot form a proper kinematic chain that preserves the hierarchical order of connections and consistent kinematic chain length, because the unconstrained transformations $\hat{\mathbf{T}}^t$ are learned without any knowledge of the kinematic chain. To recover the proper kinematic chain, we follow Algorithm 1 to obtain the revised location of the joints in hierarchical order denoted by $\tilde{\mathbf{P}} = \{\tilde{\mathbf{p}}_i \mid i = 1, \ldots, n_j\}$. Given the joints' revised locations, we can infer the anchors' revised transformations $\tilde{\mathbf{T}}^t$ by enforcing pre-defined associations between anchors and the kinematic chain links in the deformed space.

Algorithm 1 Recovering proper kinematic chain

```
for $j = 1 : n_j$ in hierarchical order do
    for $\mathbf{p}_k \in \mathtt{Children}(\mathbf{p}_j)$ do
        $\boldsymbol{\mu} \leftarrow \frac{\|\mathbf{p}_k - \mathbf{p}_j\|_2}{\|\hat{\mathbf{p}}_k - \hat{\mathbf{p}}_j\|_2}$
        $\tilde{\mathbf{p}}_k = \tilde{\mathbf{p}}_j + \boldsymbol{\mu}(\hat{\mathbf{p}}_k - \hat{\mathbf{p}}_j)$
        $\mathbf{t}_k \leftarrow \tilde{\mathbf{p}}_k - \hat{\mathbf{p}}_k$
        for $\mathbf{p}_d \in \mathtt{Descendants}(\mathbf{p}_k)$ do
            $\hat{\mathbf{p}}_d \leftarrow \hat{\mathbf{p}}_d + \mathbf{t}_k$
        end for
    end for
    $j \leftarrow j + 1$
end for
```

B.3. Additive residuals to kinematic chain

We introduce a learnable additive residual term for each kinematic chain link during the kinematic chain optimization stage. Given the canonical kinematic chain as $\mathbf{P} = \{\mathbf{p}_i \mid i = 1, \ldots, n_j\}$ and its corresponding links as $\mathbf{L} = \{\boldsymbol{\ell}_{jk} = (\mathbf{p}_j, \mathbf{p}_k)\}^{(n_j - 1)}$, let $\mathbf{R} = \{\mathbf{r}_i \mid i = 1, \ldots, (n_j - 1)\}$ be the set of learnable residuals. We clip the residuals to avoid drastic changes on the kinematic chain initialization for stable training as:

$$\tilde{\mathbf{R}} = \boldsymbol{\gamma} \tanh(\mathbf{R}), \tag{5}$$

where $\tilde{\mathbf{R}} = \{\tilde{\mathbf{r}}_i \mid i = 1, \ldots, (n_j - 1)\}$ are the set of clipped residuals and $\boldsymbol{\gamma}$ is a hyper-parameter that ensures the change of each kinematic chain link's length is within $(-\boldsymbol{\gamma}, \boldsymbol{\gamma})$.

We update the canonical kinematic chain with the latest residuals $\tilde{\mathbf{R}}$ in each iteration to obtain the updated kinematic chain $\tilde{\mathbf{P}} = \{\tilde{\mathbf{p}}_i \mid i = 1, \ldots, n_j\}$ following Algorithm 2. Specifically, we start from the root joint of the kinematic chain and update each link's length by changing the corresponding joint's position in a hierarchical manner to ensure that we do not break the kinematic chain. After the residual update, we compute the new set of association parameters for each anchor with respect to the updated kinematic chain for kinematic chain driven deformations.

C. Additional Implementation Details

In this section, we provide additional details on the loss functions and optimizations.

Algorithm 2 Kinematic chain update with residuals

```
for $j = 1 : n_j$ in hierarchical order do
    for $\mathbf{p}_k \in \mathtt{Children}(\mathbf{p}_j)$ do
        $\boldsymbol{\mu} \leftarrow \frac{\|\mathbf{p}_k - \mathbf{p}_j\|_2 + \tilde{\mathbf{r}}_k}{\|\mathbf{p}_k - \mathbf{p}_j\|_2}$
        $\tilde{\mathbf{p}}_k = \tilde{\mathbf{p}}_j + \boldsymbol{\mu}(\mathbf{p}_k - \mathbf{p}_j)$
        $\mathbf{t}_k \leftarrow \tilde{\mathbf{p}}_k - \mathbf{p}_k$
        for $\mathbf{p}_d \in \mathtt{Descendants}(\mathbf{p}_k)$ do
            $\mathbf{p}_d \leftarrow \mathbf{p}_d + \mathbf{t}_k$
        end for
    end for
    $j \leftarrow j + 1$
end for
```

C.1. Loss functions

Similar to [18], we impose reconstruction losses on the 2D observations, including 2D images, foreground masks, and optical flow. We follow the volumetric rendering process described in B.1 to obtain predicted color and foreground mask values. To render optical flow, we follow [18] to transform canonical space points to consecutive frames' deformed spaces, and use the camera projection model to find their corresponding positions on the 2D image. We then compute their difference as the optical flow at the point of interest. The reconstruction losses are formulated in the same way as in [10, 16–18, 20], where we compute the difference between the rendered and the actual observations:

$$\mathcal{L}_{\text{recon}} = \sum_{\mathbf{x}_I} \|\mathbf{o}_r - \mathbf{o}_{gt}\|^2, \tag{6}$$

where $\mathbf{o}_r$ and $\mathbf{o}_{gt}$ are the pairs of rendered and ground-truth 2D observations (2D images, foreground masks, optical flow) at pixels of interest $\mathbf{x}_I^t \in \mathbb{R}^2$ at time t.

We apply soft argmax descriptor matching [2, 8, 17, 18] at the pixel of interest $\mathbf{x}_I^t$, to find the most probable corresponding surface point $\mathbf{x}_m \in \mathbb{R}^3$ in the canonical space by matching learned canonical feature embeddings at the object's surface with the pre-trained DINO ViT [1] features. Let $\mathbf{x}_c \in \mathbb{R}^3$ be the point that corresponds to the pixel of interest $\mathbf{x}_I^t$ based on backward kinematics. To supervise the canonical feature embedding, we follow [18] to minimize the difference between these two points' positions as:

$$\mathcal{L}_{\text{3d-match}} = \sum_{\mathbf{x}_I} \|\mathbf{x}_m - \mathbf{x}_c\|_2^2. \tag{7}$$

Let π^t be the camera project model at time t, we transform $\mathbf{x}_m$ from the canonical space to the deformed space using learned forward kinematics, and use π^t to project it to image coordinates. We follow [5,17,18] to enforce the consistency at the image coordinate level by minimizing the difference between the obtained point and $\mathbf{x}_I^t$:

$$\mathcal{L}_{\text{2d-match}} = \sum_{\mathbf{x}_I} \left\| \pi^t \left(\mathbf{C}^t \sum_{i=1}^{n_a} (\mathbf{w}_{a_i} \mathbf{T}_i^t) \mathbf{x}_m \right) - \mathbf{x}_I^t \right\|_2^2 . \quad (8)$$

In addition, we follow [6, 18] to encourage consistency in terms of learned forward kinematics and backward kinematics. Specifically, we transform each point $\mathbf{x}_i^t$ along the sampled camera ray in the deformed space to the canonical space using backward kinematics, and then transform each point back to the deformed space. We encourage the resulting point $\mathbf{x}_i^{t'}$ to be at the same position as $\mathbf{x}_i^t$:

$$\mathcal{L}_{\text{transform}} = \sum_i T_i \left\| \mathbf{x}_i^{t'} - \mathbf{x}_i^t \right\|_2^2 , \quad (9)$$

where $\mathbf{x}_i^t$ is the original sampled point along the camera ray, $\mathbf{x}_i^{t'}$ is the sampled point that undergoes a backward and a forward transform, and T_i the accumulated transmittance between the camera center to the i-th sampled point, which we use as the weight for individual sampled points.

During the kinematic chain optimization stage, we also introduce regularization on the anchors and learned transformations by minimizing the differences between the revised anchors based on the kinematic chain and the unconstrained anchors from learned unconstrained anchor transformations $\hat{\mathbf{T}}_i^t$ predicted by $\mathcal{F}_{\mathbf{A}}$:

$$\mathcal{L}_{\text{anchors}} = \sum_{i=1}^{n_a} \left\| \tilde{\mathbf{a}}_i - \hat{\mathbf{T}}_i^t \mathbf{a}_i \right\|_2^2 , \quad (10)$$

where $\tilde{\mathbf{a}}_i$ is the revised anchor based on kinematic chain constraints and $\hat{\mathbf{T}}_i^t$ is the unconstrained rigid transformation from the canonical space to the deformed space given by the MLP network $\mathcal{F}_{\mathbf{A}}$ for the canonical space anchor $\mathbf{a}_i$.

C.2. Optimizations

Our model is trained using the Adam [3] optimizer with weight decay on a single RTX 3090 GPU. In the initial optimization stage, we use an initial learning rate of 0.0005 and update it using one-cycle policy [12] and cosine annealing [7]. Inspired by [18], we introduce a separate MLP network after the initial optimization stage that takes in a 3D point, and outputs a skinning weight residual with respect to each anchor that adds to the Euclidean distance based skinning weights before normalization, to obtain smoother deformed surface. All the MLP networks and learnable parameters are optimized jointly during training. For the iiwa dataset, we do not update the kinematic chain link lengths with additive residuals because iiwa is considered a rigid robot arm without deformable surface, and its kinematic chain initialization is generally a single chain that already captures its articulation modes.

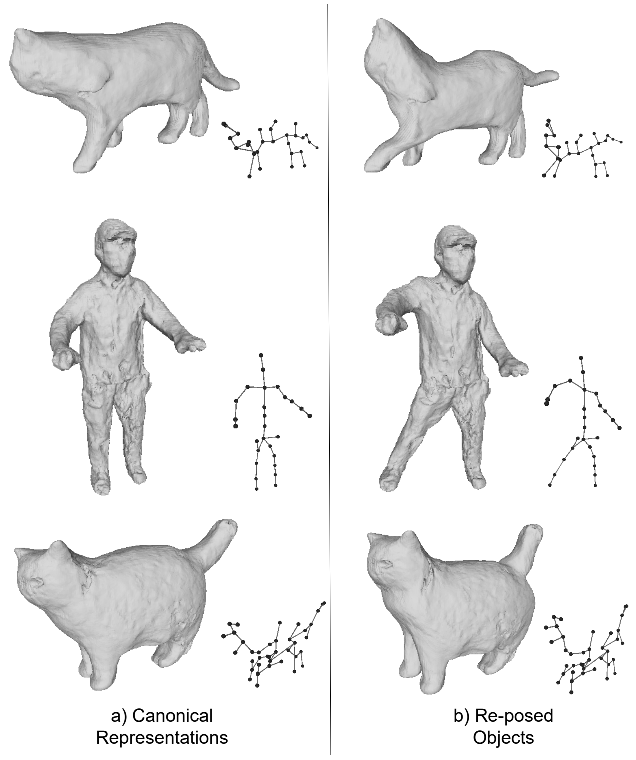

Figure 1. **Pose manipulation examples**. In a) we show the learned canonical representations of the objects for in-the-wild datasets. We perform pose manipulations using optimized kinematic chain and show the reposed kinematic chain and meshes in b).

D. Additional Results

D.1. More qualitative results

We test on in-the-wild monocular captures of various deformable and articulated objects [18]. We use off-the-shelf pre-trained models to obtain foreground masks [4], initial root pose estimation [18] and optical flow [15] for each frame of the in-the-wild dataset videos. We show manual pose manipulation results in Figure 1, and 3D surface reconstruction results in Figure 2 for in-the-wild datasets.

We also include video examples of manual pose manipulations by re-posing the kinematic chain, and the 3D reconstruction results for poses in training videos for each dataset. Please refer to the videos in the supplementary material for better visualization.

D.2. More quantitative results

We report quantitative 3D surface reconstruction results for iiwa (Table 1), Eagle (Table 2), AMA-swing (Table 3), and AMA-samba (Table 4) evaluated in 3D Chamfer Distances in centimeters and F-scores at distance thresholds of 1%, 2%, and 5%. Note that for the AMA dataset, we train

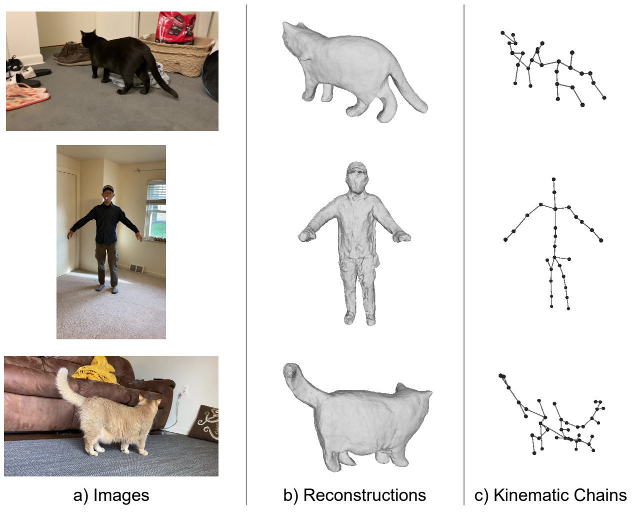

Figure 2. **3D surface reconstruction results.** In a) we show some RGB images from the in-the-wild datasets. In b) we show the reconstructed 3D meshes and in c) we show the corresponding kinematic chains for the same frame.

Table 1. 3D surface reconstruction results on **iiwa** dataset evaluated in 3D Chamfer Distances (↓) and F-scores (↑) at distance thresholds of 1%, 2%, and 5% averaged across all frames.

Method	CD	F@1%	F@2%	F@5%
5 anchors	7.40	26.76	51.16	79.84
10 anchors	5.52	28.69	56.53	85.61
20 anchors	5.95	28.56	55.17	83.51
with DensePose feat.	5.64	28.85	55.67	85.79
w/o feat.	7.26	25.16	47.33	73.82
w/o kine. chain opt.	5.69	26.90	52.44	81.81

Table 2. 3D surface reconstruction results on **Eagle** dataset evaluated in 3D Chamfer Distances (↓) and F-scores (↑) at distance thresholds of 1%, 2%, and 5% averaged across all frames.

Method	CD	F@1%	F@2%	F@5%
12 anchors	4.51	43.20	81.86	98.88
25 anchors	4.21	43.41	83.38	99.22
50 anchors	4.38	42.21	82.52	98.91
with DensePose feat.	4.31	43.07	82.45	99.11
w/o feat.	4.31	43.51	82.81	98.98
w/o kine. chain opt.	7.44	25.30	57.87	94.40

each model using all the sequences (both swing and samba), and report results on swing and samba separately.

E. Anchors and Associations

The anchors are optimized to model the modes of the deformations, and the skinning weights are optimized w.r.t. anchors. By enforcing deterministic associations between the anchors and the kinematic chain, the transformations of anchors can be directly inferred from user-defined kinematic chain transformations. Therefore, we can make good use of optimized anchors and their corresponding skinning weights, instead of optimizing the skinning weights w.r.t. kinematic chain joints from scratch. In Figure 3, we show the learned deformation anchors and their associations to the kinematic chain links for each dataset. The anchors move along with the kinematic chain links to keep the association parameters constant at all times to enable re-posings directly driven by the kinematic chain.

Note that our proposed kinematic chain driven deformations formulation is capable of modeling twists of a kinematic chain link about its axial direction even though our kinematic chain is defined as connected line segments. An additional rotation matrix that represents the rotation about the axial axis of the link is pre-multiplied to rotate the anchor around its associated link for twist effects.

Table 3. 3D surface reconstruction results on **AMA-swing** evaluated in 3D Chamfer Distances (↓) and F-scores (↑) at distance thresholds of 1%, 2%, and 5% averaged across all frames.

Method	CD	F@1%	F@2%	F@5%
17 anchors	12.41	24.37	45.69	78.10
35 anchors	9.69	29.17	53.29	85.20
70 anchors	12.49	25.62	46.72	77.45
with DensePose feat.	8.96	33.68	58.46	86.49
w/o feat.	9.88	27.60	52.15	84.42
w/o kine. chain opt.	11.13	26.54	48.88	80.69

Table 4. 3D surface reconstruction results on **AMA-samba** evaluated in 3D Chamfer Distances (↓) and F-scores (↑) at distance thresholds of 1%, 2%, and 5% averaged across all frames.

Method	CD	F@1%	F@2%	F@5%
17 anchors	11.76	25.51	49.09	81.52
35 anchors	9.22	29.81	54.20	86.91
70 anchors	11.72	25.74	48.72	80.69
with DensePose feat.	8.34	33.89	60.16	88.66
w/o feat.	9.24	27.06	52.88	87.37
w/o kine. chain opt.	9.80	31.86	55.34	84.69

F. Necessity of Two-Stage Optimization

Since we do not have access to any shape prior or template, building both object's shape and kinematic chain from scratch simultaneously is a highly ill-conditioned optimization problem given only a collection of monocular videos. Our proposed two-stage optimization simplifies this problem by taking advantage of the techniques in the existing template-free surface reconstruction method [18] to extract an initial estimate of the 3D shape, which allows us to use RigNet [14] to extract an initial estimate of the kine-

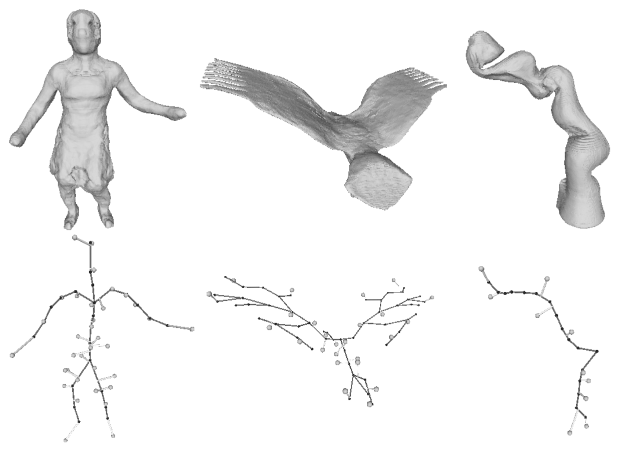
Figure 3. **Anchors and associations.** Anchors (white dots around the kinematic chain) and their corresponding associations (line segments between anchors and kinematic chain links) for the synthetic and AMA datasets.

matic chain. The two-stage scheme significantly reduces the search space for the underlying rigid structure of the object and makes the optimization much more stable.

As RigNet [14] operates on 3D mesh, it's possible to directly apply RigNet to the initial optimization stage results to obtain an animtable model driven by RigNet's predicted skinning weights. However, it would yield three problems. The first problem is that the skinning weights predicted by RigNet are based on a static mesh without any knowledge from the videos (dynamic scenes). These weights are inaccurate and not optimized for the deformation modes that occurred in the videos. The second problem is that RigNet suffers from significantly large memory consumption and runtime. The author of RigNet recommends users to only predict skinning weights for kinematic chain joints on low-resolution mesh ($\leq$ 5,000 vertices) on its code repository [14] to prevent running out of hardware's memory and extremely long runtime. Our approach does not have this bottleneck as we do not rely on the skinning weights predicted by RigNet. We build the associations between anchors and the kinematic chain to animate object meshes regardless of the resolution. The third problem is that directly using the RigNet initialization combined with the anchors would lead to poor results. As reported in the Tables in the main paper and the supplementary material, we achieve consistent improvements on all datasets with our proposed kinematic chain optimization stage.

G. Limitations

Input dependencies. In our pipeline, we leverage an off-the-shelf model [15] to extract optical flow from monocular videos, and use ground-truth foreground masks from the datasets for optimizations. For the Eagle and iiwa datasets, we use the ground-truth root poses, while for the AMA dataset, we leverage a pre-trained PoseNet [18] for root pose initializations and jointly optimize them during training. Therefore, our pipeline's performance is affected by the quality of these inputs, and it requires a generic root pose estimator to enable our pipeline to work on any object of interest because PoseNet [18] is only suitable for humans and quadruped animals.

Kinematic chain initialization. We use a category-agnostic skeleton estimator, RigNet [14], on the initial 3D mesh estimate of the object to obtain kinematic chain initialization. However, the underlying structures and articulation modes vary across different object categories, requiring users to tune a RigNet's test-time hyper-parameter that controls how dense the joints are distributed within the object's mesh. We encourage the users to run RigNet multiple times on the same initial estimate of the mesh with different test-time hyper-parameters, and select the kinematic chain that agrees with the object's underlying structure and articulation modes the most before optimizing it.

H. Potential Negative Impact

As our approach effectively builds a 3D animatable model of an object using monocular videos, it can be potentially used for malicious purposes, such as generating fake

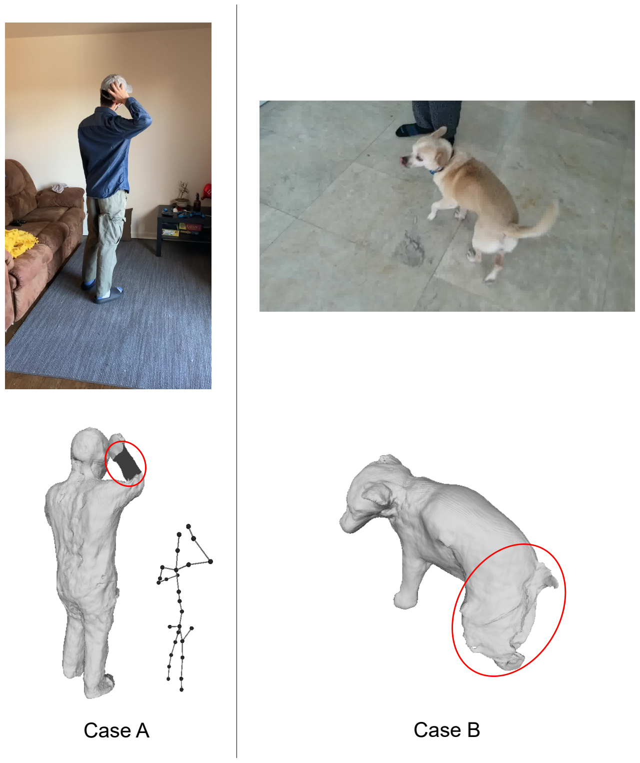

Figure 4. **Failure case examples.** In Case A, the mesh of the person's right arm collapses despite the kinematic chain being able to represent the correct pose. In Case B, the dog's tail and back legs are missing due to bad reconstruction.

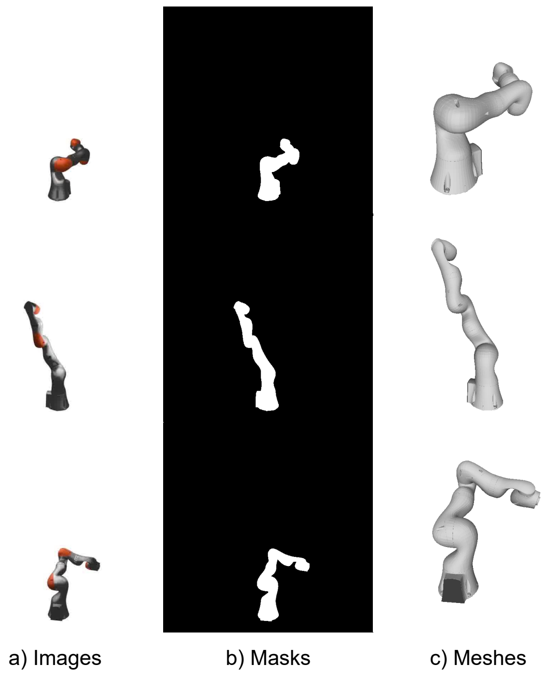

Figure 5. **Examples of our iiwa dataset.** We show examples of a) RGB images, b) corresponding foreground masks, and c) corresponding ground truth meshes for three randomly picked frames in the dataset. Note that the meshes are re-scaled for better visualization in the figure.

videos or other formats of illegal content. Since monocular videos are often easy to acquire in the real world, it is important to ensure our method is used with prior consent.

I. Failure Cases

We show some example failure cases in Figure 4. In Case A, the mesh of the person's right arm collapses at the pose shown in the figure despite the correct kinematic chain pose. This is because the deformation anchors are not optimized well to handle the desired pose. In the cases of training with fast-moving objects and large self-occlusions in the videos, the reconstruction results sometimes suffer from significant degradation, as shown in Case B. The dog's tail is moving very fast in the training video and often causes self-occlusions at the back of the dog, making the overall optimization much less stable and sometimes converge to poor results. In such cases, we recommend tuning the hyper-parameters in training or gathering more videos with 360 degrees captures of the object to make the training more stable. In general, objects that are moving slowly with 360 degrees captures are usually easier to train.

J. Additional Details on iiwa Dataset

We created a new dataset that contains four video sequences of an animated KUKA LBR iiwa robot arm. The synthetic model of the robot arm is obtained from BlenderKit. We animate the robot arm in Blender and render the videos of the robot arm's motion. Each video contains 400 frames in total. Note that for each frame in the video sequences, the camera pose, foreground mask, and 3D mesh of the robot arm are also included in the dataset for quantitative evaluation purposes. We show examples of our dataset in Figure 5.

K. Notations

In Table 5, we list the important variables used in this paper, along with their state space and descriptions.

Table 5. **Notations.** A list of the important variables used in the paper.

Symbol	**State space**	**Description**
$\mathcal{F}_{\mathbf{C}}$	MLP	MLP for color in canonical space
$\mathcal{F}_{\mathbf{S}}$	MLP	MLP for SDF in canonical space
$\mathcal{F}_{\boldsymbol{\phi}}$	MLP	MLP for canonical feature in canonical space
$\mathcal{F}_{\mathbf{A}}$	MLP	MLP for unconstrained transformations of anchors
$\mathbf{p}_i$	$\mathbb{R}^3$	i-th kinematic chain joint in canonical space
$\hat{\mathbf{p}}_i$	$\mathbb{R}^3$	i-th unconstrained kinematic chain joint
$\tilde{\mathbf{p}}_i$	$\mathbb{R}^3$	i-th revised kinematic chain joint
$\mathbf{a}_i$	$\mathbb{R}^3$	i-th deformation anchor in canonical space
ℓ_{jk}	link	Kinematic chain link between $\mathbf{p}_j$ and $\mathbf{p}_k$
$\mathbf{C}^t$	$SE(3)$	Object root pose w.r.t the canonical root pose at time t
$\mathbf{T}_i^t$	$SE(3)$	Transformation of anchor $\mathbf{a}_i$ at time t
$\hat{\mathbf{T}}_i^t$	$SE(3)$	Unconstrained transformation of anchor $\mathbf{a}_i$ at time t
$\tilde{\mathbf{T}}_i^t$	$SE(3)$	Revised transformation of anchor $\mathbf{a}_i$ at time t
$\mathbf{H}_i^t$	$SE(3)$	Forward kinematics of kinematic chain joint $\mathbf{p}_i$ at time t
$\mathbf{w}_{a_i}$	$\mathbb{R}$	Forward skinning weight w.r.t anchor $\mathbf{a}_i$
$\mathbf{w}_{a_i}^t$	$\mathbb{R}$	Backward skinning weight w.r.t anchor $\mathbf{a}_i$ at time t
$\mathbf{R}$	$\mathbb{R}^{n_j-1}$	Unclipped additive residuals for kinematic chain
$\tilde{\mathbf{R}}$	$\mathbb{R}^{n_j-1}$	Clipped additive residuals for kinematic chain
$\mathbf{x}_I^t$	$\mathbb{R}^2$	Pixel of interest at time t
$\boldsymbol{\psi}_a^t$	$\mathbb{R}^{128}$	Learnable latent code for anchor transformations at time t
$\boldsymbol{\psi}_l$	$\mathbb{R}^{64}$	Learnable latent code illumination conditions
π^t	$\mathbb{R}^{3\times 4}$	Camera projection model at time t

References

[1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 2

[2] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *ICCV*, 2017. 2

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint*, 2014. 3

[4] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *CVPR*, 2020. 3

[5] Nilesh Kulkarni, Abhinav Gupta, David F Fouhey, and Shubham Tulsiani. Articulation-aware canonical surface mapping. In *CVPR*, 2020. 2

[6] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 3

[7] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint*, 2016. 3

[8] Diogo C Luvizon, Hedi Tabia, and David Picard. Human pose regression by combining indirect part detection and contextual information. In *Computers & Graphics*, 2019. 2

[9] 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 *CVPR*, 2021. 1

[10] 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. In *ECCV*, 2020. 1, 2

[11] 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 *ICCV*, 2021. 1

[12] Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint*, 2018. 3

[13] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021. 1

[14] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. In *SIGGRAPH*, 2020. 4, 5

[15] Gengshan Yang and Deva Ramanan. Volumetric correspondence networks for optical flow. In *NeurIPS*, 2019. 3, 5

[16] 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 *CVPR*, 2021. 2

[17] 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. In *NeurIPS*, 2021. 2

[18] Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. Banmo: Building animatable 3d neural models from many casual videos. In *CVPR*, 2022. 1, 2, 3, 4, 5

[19] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *NeurIPS*, 2021. 1

[20] 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