

## A Center and Rotation Estimation

To estimate the animals orientation at each time point  $t$ , we modeled its shape as a Gaussian distribution  $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$  by matching the 3D moments of a shape-carved voxel grid. We then extracted the principal axis by computing the eigenvector corresponding to the largest eigenvalue of  $\Sigma_t$ . This eigenvector is normalized to unit norm and called  $\mathbf{v}_t$ . Because an eigenvector can flip sign from one timepoint to the next, we introduced a sign-consistency procedure to best ensure a smooth progression of the eigenvectors. Specifically, after computing  $\mathbf{v}_{t+1}$  at time  $t + 1$ , we used a Wasserstein-2 optimal transport map to track a reference point from the distribution at time  $t$  to that at time  $t + 1$ . We then compared the distance of this transported point to the two possible orientations ( $\boldsymbol{\mu}_{t+1} \pm \mathbf{v}_{t+1}$ ) and chose the orientation that preserved the local consistency (i.e., whichever was closer to the transported point).

Finally, we enforced a global orientation consistency by making use of the following heuristic: animals move on average in the direction they face, and not the opposite direction. We operationalize this heuristic by checking if the cumulative displacement of the mean positions  $\boldsymbol{\mu}_t$  from start to end correlated positively with the sequence of principal axes  $\{\mathbf{v}_t\}$ . If the overall dot product was negative, we flipped all axes to align with the general direction of motion. This procedure yielded a temporally consistent set of principal axes  $\{\mathbf{v}_t\}$  that reliably tracked the animal’s primary orientation over time.

## B Shape Carving Details

**Voxel Occupancy** We first start with a  $112 \times 112 \times 112$  spatial grid with equal-sized edges. The  $z$ -axis is aligned with the third axis of the grid, while the first axis of the grid is aligned with the estimated azimuthal heading direction of the animal (see Appendix A). For each point in the grid and each camera, we determine whether its projection onto the image plane of the camera corresponds with a masked (animal) or unmasked (background) point. Voxels that correspond with masked regions in at least  $N$  cameras are considered occupied.

**Voxel Colors** To estimate the color of each voxel in the reconstructed volume, we first determined its visibility from multiple camera viewpoints using a ray-casting procedure. For each camera, the voxels were sorted by their distance from the camera center, ensuring that the closest voxels were processed first. These voxels were then projected into the image plane using a standard pinhole camera model. To track occlusions, a depth buffer was maintained at each pixel location. If a voxel mapped to a pixel that already contained a closer voxel, it was marked as occluded. This process produced a visibility mask indicating whether each voxel was directly observable from each camera.

We then computed each voxel’s color by sampling the corresponding pixel values from the camera images. All voxels, whether fully visible or occluded, were projected into the camera views, and their colors were retrieved from the respective images. To integrate these sampled colors into a single representative value per voxel, we applied a weighted averaging scheme. Fully visible voxels contributed with a weight of one, while occluded voxels were assigned a reduced weight (0.25) to reflect the uncertainty introduced by occlusion. The final voxel color was obtained by normalizing and summing these weighted contributions across all cameras, allowing us to account for occlusions while still incorporating partial information from less reliable viewpoints.

### Determining Volume Dimensions

To save on RAM and GPU memory during training, we truncated our original  $112 \times 112 \times 112$  volumes based on voxel usage in the animal’s training frames. We first calculated a sum over all the voxel occupancies and then manually set thresholds and determined volume slice indices to balance voxel use and memory considerations. The following table shows the final dimensions of the volumes for each animal.

Mouse	$96 \times 80 \times 64$
Finch	$96 \times 64 \times 80$
Rat	$96 \times 80 \times 64$

Table 3: Volume dimensions ( $d_x \times d_y \times d_z$ ) for each animal.

**Final Steps** Our volumes consist of 4 channels: one binary occupancy channel and 3 color channels. To produce our final volumes, we produce two volumes independently, one with an occupancy

threshold of  $C$ , where  $C$  is the number of cameras, and one with the threshold  $C - 1$ . We then average the two volumes together. Note that the  $C - 1$  threshold produces a coarser visual hull, with generally more occupied voxels. Initial tests showed that fine body parts such as mouse tails were often not represented in the shape-carved volume with just a threshold at  $C$ , especially when masks from different views disagree on the boundaries of the animal. We found that adding the  $C - 1$  threshold and averaging resulted a much better coverage of fine body parts.

## C Visual Embedding Details

The feature extractor used in the visual embedding is a pre-trained ResNet 18, trained on the ImageNet dataset. We pass rendered images through all but the last layer of the network, producing a 512-dimensional vector per image. 32 of these vectors are used in a spherical routine to produce the norms of 16 norms of spherical harmonic coefficients, independently for each feature dimension. The resulting coefficients are then flattened to 8192-dimensional vectors.

To produce our visual embeddings, we reduce the feature dimension in two steps. First, we perform PCA on dataset of feature vectors, reducing the feature vectors to 2000 dimension. Then we collect the estimated azimuthal angle for each pose and concatenate the sine and cosine components of this angle together to be used as concomitant data in the adversarial PCA routine [9]. Adversarial PCA was then performed to reduce the feature dimensionality to 50 from 2000. The regularization strength parameter  $\mu$  was set to the smallest integer power of 10 such that the adversarial PCA reconstructions of the sine and cosine components produced an average  $R^2$  value less than 0.05, indicating the sines and cosines are not readily linearly decodable from the 50-dimensional features. We call these features the *visual embedding*.

To predict egocentric 3D keypoints from visual embeddings (Figure 5, left), we used a consecutive 80/20 train/test split, fit a 5-nearest neighbor regressor to the training data, and report a uniform average of the  $R^2$  scores over the 3 spatial dimensions on the test set. We also predicted egocentric 3D keypoints from the 8192-dimensional visual features, which are processed to produce the 50-dimensional visual embeddings to test how much usable information is lost in this process. We used ridge regression with 5-fold cross validation on the training set to select a model and then report a uniform average of the  $R^2$  scores over the 3 spatial dimensions on the test set. Figure 6 shows moderately better performance using the visual features than the visual embedding to predict egocentric 3D keypoints.

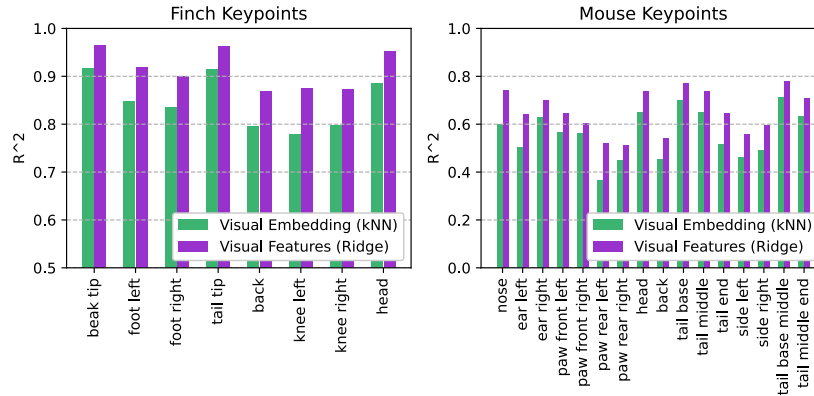


Figure 6: **Predicting egocentric 3D keypoints from visual embedding ( $d = 50$ ) and visual features ( $d = 8192$ ).** We observe good predictive ability as measured by  $R^2$  values using both visual embeddings (green) and visual features (purple) to predict held-out finch (left) and mouse (right) keypoints. Note the different vertical axes in the two subplots.

To classify behavior given egocentric 3D keypoints and visual embeddings, we used a random 60/40 train/test split using all available frames. We used 5-fold cross validation to determine the L2 regularization strength on the train set, targeting a balanced accuracy metric. Accuracies for each class are then reported on the test set. Mouse behavior was classified using one 4-way prediction

	<u>Shape Carving</u>	<u>U-Nets</u>	<u>MLP</u>	<u>Shift &amp; Rotate</u>	<u>Splatting</u>	<u>Train FPS</u>	<u>Inference FPS</u>
<b>Finch</b>	7.0 ms	4.6 ms	14.3 ms	6.0 ms	0.2 ms	7.0	7.4
<b>Mouse</b>	7.0 ms	4.7 ms	13.6 ms	3.5 ms	0.3 ms	5.8	6.5
<b>Rat</b>	7.1 ms	5.1 ms	14.3 ms	2.6 ms	0.3 ms	9.7	13.1

Table 4: Median times for the 5 stages of the *Pose Splatter* forward pass and overall frame rates for training and inference.

	<u>Before Training</u>	<u>After Training</u>
<b>Finch</b>	13.9k	13.8k
<b>Mouse</b>	8.4k	8.5k
<b>Rat</b>	4.8k	4.8k

Table 5: Average number of Gaussians rendered before training (at initialization) and after training.

943 (Walk vs. Head Up vs. Still vs. Groom) and finch behavior was classified using two two-way  
 944 predictions (Head Left vs. Head Right and Tail Down vs. Tail Up).

## 945 D Training and Network Details

946 Training was run with a single Nvidia RTX A4000 GPU along with 32 CPU cores used for data  
 947 fetching. Please note that our model uses only about 2.5GB of GPU memory (VRAM), thanks to its  
 948 simple architecture. Training runs varied from between 2 and 12 hours, depending on the length of  
 949 the dataset. We estimate that the experiments presented in this paper required 6 days of compute time  
 950 to complete.

951 The learning rate was fixed at  $10^{-4}$  for all experiments. The number of epochs were chosen so to  
 952 minimize validation set loss, and ranged from 40 to 75 epochs across all experiments.

953 In experiments with fewer than 6 camera views, we found that this center and rotation estimation  
 954 procedure produces less reliable results. While it would be useful to develop a more robust procedure  
 955 for camera systems with fewer cameras, this was not a primary aim of our work. Therefore, we used  
 956 the centers and orientations inferred with all 6 cameras for all experiments.

957 Tables 4 and 5 show timing information and the number of Gaussians rendered for all three datasets.  
 958 Additional details may be found in the supplemental code.

959 Meanwhile, our additional training details for the comparison baselines were as follows. For the  
 960 per-scene optimization methods (3DGS [24], FSGS [71], and GaussianObject [61]), the original  
 961 pipelines initialize optimization with a point cloud created from "Structure-from-Motion Revisited  
 962 (COLMAP)." In typical neuroscience-oriented animal-behavior studies, however, one may work with  
 963 hundreds of thousands of video frames; running COLMAP on every frame would be prohibitively  
 964 slow. We therefore assume no COLMAP ground-truth point cloud is available. Instead, we follow  
 965 GaussianObject’s COLMAP-free protocol: we seed optimization with a point cloud created from  
 966 "DUST3R: Geometric 3D Vision Made Easy (DUST3R)," feeding the calibrated poses directly into  
 967 DUST3R (because the camera intrinsics and extrinsics are known) so that its reconstruction is as clean  
 968 as possible. The GaussianObject paper shows that this DUST3R initialization maintains state-of-the-art  
 969 performance, a result that our experiments confirm. All remaining hyperparameters mirror the “best”  
 970 settings recommended in each method’s official repository. Additionally, per-scene optimization  
 971 methods require varying amounts of time depending on the approach, ranging from a few minutes  
 972 (3DGS, FSGS) to about an hour (GaussianObject) per scene. As such, optimizing tens of thousands  
 973 of frames is impractical. Therefore, we randomly sampled 150 images from the test set and conducted  
 974 experiments using this subset. For the feed-forward baselines, PixelSplat [10] and MVSplat [11],  
 975 we likewise adopt the authors’ recommended hyperparameters. Finally, the single-view animal  
 976 reconstruction models MagicPony [59] and 3D Fauna [31] are trained with their prescribed data  
 977 preprocessing pipelines and hyperparameters; for these methods, we present qualitative results  
 978 only, because their required resizing and cropping distort the camera parameters, so a quantitative  
 979 comparison against *Pose Splatter* that use the exact camera parameters would be inherently unfair.

## E Additional Renderings

The following figure shows randomly sampled test set renderings for the three 6-camera models **from observed views**.

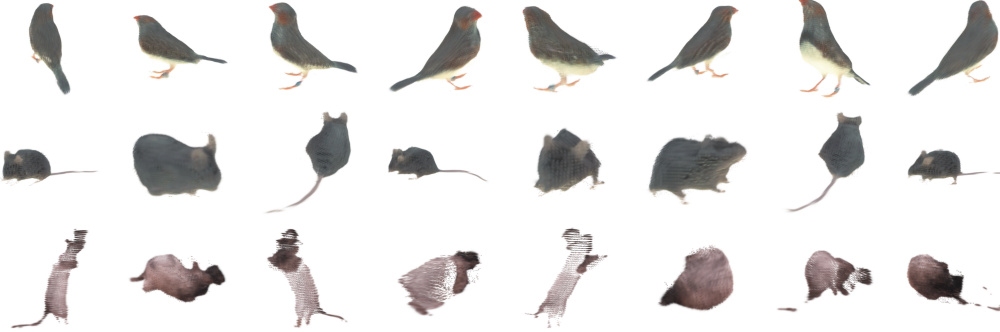


Figure 7: Representative renderings from observed views.

The following figure shows randomly sampled test set rendering for the three 6-camera models **from unobserved views**.

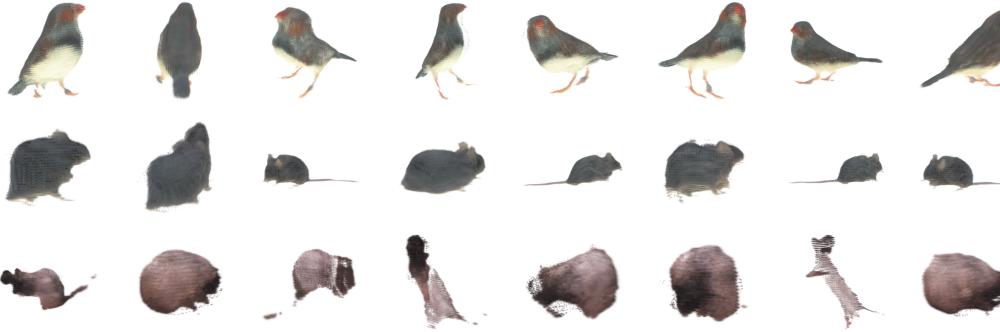


Figure 8: Representative renderings from unobserved views.

Despite the de-voxelization step in *Pose Splatter*, we observed regular grid-like patterns in many renderings. For the mouse and finch datasets we trained models with twice the spatial resolution in the voxelization step, resulting in 8 times as many voxels. As shown in Figure 9, we see a similar visual quality but no apparent regular grid patterns.

## F Quality of 3D Keypoints

After training a SLEAP model [44] to predict 2D keypoints from images, we perform a robust triangulation across views using the known camera parameters to estimate 3D keypoints. To assess the quality of these 3D keypoints, we calculated the distribution of reprojection errors (in pixels) for both the mouse and finch datasets. As seen in Figure 10, we find higher quality 3D keypoints for the finch dataset. We believe this results from the higher quality 2D keypoints for this dataset. This may also explain the better performance predicting 3D keypoints from visual embeddings on the finch dataset (Figure 5, left). Representative examples of SLEAP-predicted 2D keypoints and reprojected 3D keypoints are shown in Figure 11.

## G Additional Comparisons

We also report how the feed-forward baselines behave as the number of context views changes (see Figure 12). According to the papers and authors’ code repositories, both PixelSplat [10] and

## Regular Models Large Models

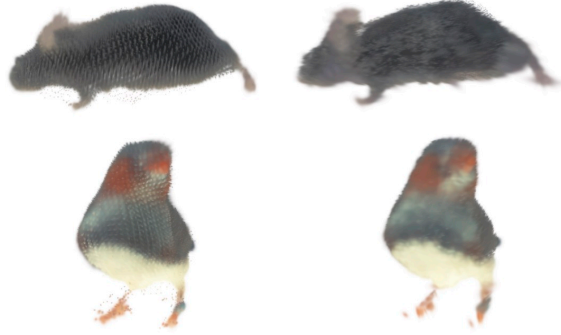


Figure 9: **Visual comparison between regular and large shape carving volumes.** The renderings on the left come from models using a “standard” volume size of  $96 \times 80 \times 64$  for the mouse and  $96 \times 64 \times 80$  for the finch (the volume sizes used in the rest of the paper). The renderings on the right come from models that use a larger volume size of  $192 \times 160 \times 128$  for the mouse and  $192 \times 128 \times 160$  for the finch. Note that the regular grid artifacts seen on the left are not visible on the right. Best viewed zoomed in.

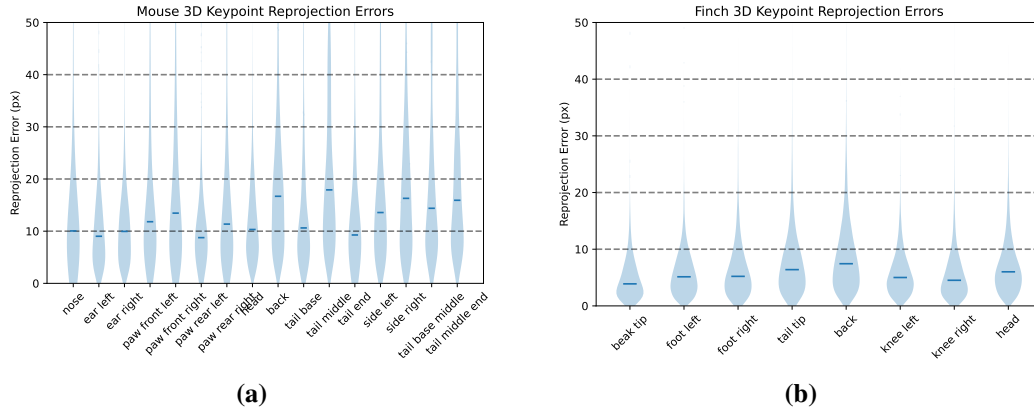


Figure 10: Summary of per-keypoint reprojection errors for **a)** mouse and **b)** finch datasets. Violin plots for each keypoint show the distribution of errors between SLEAP-predicted 2D keypoints and the 2D projections of per-frame robustly triangulated 3D keypoints. Units are Euclidean distance in pixels, for full  $1536 \times 2048$  images. Median errors are marked.

MVSplat [11] were trained with two input views, and adding more cameras does not guarantee improvement in the output. Our results confirm that increasing the number of context views does not appreciably improve the performance of either baseline.

## H Survey

To ensure transparency and enable replication of the nearest neighbor survey, with results presented in Figure 2b, we provide the survey instructions and illustrative screenshots of the user interface. The complete wording of the participant instructions is reproduced verbatim below, while Figure 13 depicts the layout of an individual survey question.

**Please read these instructions carefully before beginning the survey.**

In this study, you will judge how closely two candidate poses resemble the body posture of a reference (“query”) mouse.

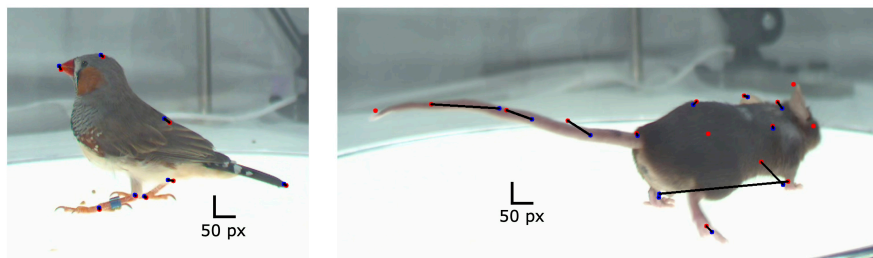


Figure 11: Frames showing SLEAP-predicted 2D keypoints (*blue*) and corresponding projected 3D keypoints (*red*). Black lines connect corresponding 2D and 3D keypoints and 50-pixel bars are shown for scale. Note the high accuracy of the triangulated finch keypoints (left) and some 2D keypoint errors for the mouse pose such as front right versus back left paw (right).

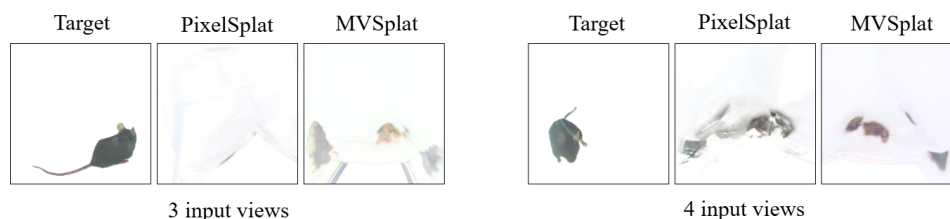


Figure 12: Results for the two feed-forward baselines with varying numbers of input views.

Each question is laid out in 3 columns: the left column shows the query pose, while the center and right columns show Option 1 and Option 2. Every column contains two synchronized camera views (top and bottom) captured at the same instant, giving complementary angles on a single pose.

Your task is to decide which option—1 or 2—better matches the query. Focus only on the configuration of body parts such as the head, torso, limbs, and tail, and disregard background, lighting, or color differences. Beneath the images are two radio buttons; click the button under the option you believe is closer to the query.

There are 40 questions in total, and there is no time limit. By continuing, you confirm that you consent to your anonymous responses being used for academic research on animal-pose representations.

Question 1

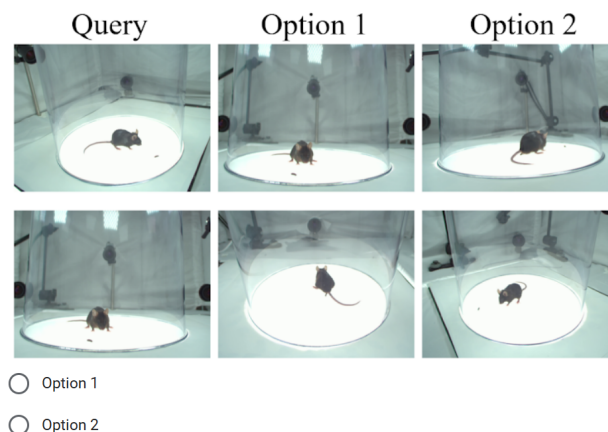


Figure 13: Example survey question