

Robot See Robot Do: Part-Centric Feature Fields for Visual Imitation of Articulated Objects

Supplementary Material

Anonymous Author(s)

Affiliation

Address

email

1 A Appendix

2 A.1 Implementation Details

3 **Part-Centric Feature Fields** Our implementation is built on Nerfstudio’s Splatfacto model, taking
4 advantage of the same splitting and culling logic. We represent DINOv2 ViT-B/14 features by
5 taking the PCA across all input image features to compress them to 64 dimensions, then assign
6 every gaussian a learnable 32-dimension vector. These vectors can be rasterized with the exact same
7 rendering equations as RGB, using the N-D rasterization implementation from the gsplat library.
8 After rasterization, pixel values are passed through a 4-layer, 64-wide MLP to output the final 64-
9 dimension features. The outputs are supervised with a simple MSE loss against the image features.
10 We additionally apply a nearest-neighbors total-variation loss, which at each step minimizes the
11 standard deviation of a gaussian with its 3 neighbors, encouraging feature embeddings to be spatially
12 smooth. To refine camera poses from their potentially noisy initialization from Polycam, we enable
13 camera optimization from view matrix gradients propagated from RGB rasterization. We train for
14 6000 steps, which takes about 3 minutes on an RTX 4090 GPU.

15 **Tracking** During loss calculation we weight the three optimization objectives with $\lambda_{ARAP} =$
16 0.2 , $\lambda_{MONO} = 0.5$, $\lambda_{DINO} = 1$ before summing. Adam’s learning rate is decreased from 0.005
17 to 0.0005 over the course of 50 steps each frame with an exponential decay. During tracking we
18 sample 30,000 random pairs within the object mask to use with the sparse depth loss, where the
19 object mask is defined by pixels with rendered alpha values over 0.9 (mostly opaque).

20 **Grasp and Motion Planning** As described in main text’s section 4.3, part contact selection out-
21 puts a ranked list of candidate object parts to interact with, from human hand detection. Then, the
22 planner attempts to find the first set of parts where the motion is executable. For bimanual tasks
23 the list is composed of length-two tuples $[(p_1, p_2), \dots]$, one part for each hand, and we exhaustively
24 check over both arms i.e., left arm to p_1 and right to p_2 , and vice versa. We first optimize for the
25 robot motion following the pose of the desired object using a motion planner implemented in JAX,
26 optimizing for smooth joint positions given a set of 6D robot gripper poses, as cuRobo currently
27 does not provide a waypoint-based trajectory optimization. Then, for the successful trajectories, we
28 use curobo to plan collision-free trajectories to the pre-grasp and grasp pose for each part.

29 A.2 Experiment Details

30 **Robot Trials** Please see the supplemental video for example executions of these motions on the
31 robot, as well as failure case videos.

32 The experiment motions for each objects are as follows:

- 33 1. Red Box: Closing the box, by lowering the lid

- 34 2. Nerf Gun: Sliding back the firing mechanism of the gun
- 35 3. Scissors: Closing, then opening the scissors
- 36 4. Sunglasses: Folding back the left leg of the sunglasses
- 37 5. Bear: Waving the right arm of the bear

38 **Tracking Evaluation** 3D pose for part trajectories is manually annotated for keyframes by visu-
39 alizing the dense RGB-pointcloud obtained from the depth camera in a 3D viewer, then manually
40 moving the rendered gaussian splat of the object part to align with this pointcloud.