CARTO: Category and Joint Agnostic Reconstruction of ARTiculated Objects

Supplementary Material

Nick Heppert, Muhammad Zubair Irshad, Sergey Zakharov, Katherine Liu,
Rares Andrei Ambrus, Jeannette Bohg, Abhinav Valada, Thomas Kollar

In this supplementary material, we first describe our model architecture in more detail in Sec. S.1. Next, we detail our backward optimization procedure in Sec. S.2. In Sec. S.3, we visualize the difference between joint code spaces trained with and without our regularization and show the fitted polynomial functions, previously introduced in Sec. 3.3.3. Subsequently, we present example images from our synthetic and real dataset in Sec. S.4. Lastly, in Sec. S.5, we report extended results for our canonical reconstruction task in Sec. 4.2 and our full pipeline experiment in Sec. 4.3.

S.1. Model Architecture

We present our model architecture for the encoder in Fig. S.1 and for the decoder in Fig. S.2.

S.1.1. Encoder

Our encoder builds upon the SimNet-architecture [3]. The input is a stereo RGB image pair of size $\mathbb{R}^{960\times512\times3}$. Each image gets passed through a shared feature encoder network that outputs a low-dimensional feature map of size $\mathbb{R}^{128\times240\times16}$. This output is then fed into a cost volume which performs approximate stereo matching. Based on the result of the cost volume of size $\mathbb{R}^{128\times240\times32}$, a lightweight head predicts an auxiliary disparity map of size $\mathbb{R}^{128\times240}$. Parallel to that, we feed the left image through a separate RGB encoder that also predicts a feature map of size $\mathbb{R}^{128\times240\times32}$. This map, as well as the output of the cost volume, get concatenated and fed into a feature pyramid network which predicts three feature maps of sizes $\mathbb{R}^{128\times240\times32}$, $\mathbb{R}^{64\times120\times64}$, $\mathbb{R}^{32\times60\times64}$. Finally, using these features, each quantity described in Sec. 3.1 is predicted by its respective output head, including a segmentation mask, 3D bounding box, object pose, full resolution disparity, shape code, and joint code heads.

As in [3], although the stereo input for sim-to-real transfer has benefits for perceiving objects in harsh lighting conditions and for transparent or reflective objects, a RGB-D version could be trained as well (see Sec. S.5.3).

S.1.2. Decoder

Our decoder is split into two sub-decoders. A geometry decoder (see Sec 3.2.1) based on DeepSDF [7] and a joint decoder (see Sec. 3.2.2). We detail both the architecture in the subsequent paragraphs.

The geometry decoder is a deep multi-layer perceptron consisting of four layers. The first layer takes a shape code $\boldsymbol{z}_\text{s}$ and joint code $\boldsymbol{z}_\text{j}$ as input. Before the second and last layer, we concatenate the space coordinate $\boldsymbol{x}$ for which we want to retrieve the SDF-value s with the output of the previous layer. Following the findings in [6], we again input the joint code $\boldsymbol{z}_\text{j}$ before the second last layer. For the exact feature vector dimensions see Fig. S.2. As an activation function, we use ReLU for all except the last layer which uses tanh. Exploring exact input positions for shape code $\boldsymbol{z}_\text{s}$, joint code $\boldsymbol{z}_\text{j}$, and space coordinate $\boldsymbol{x}$, could be a topic of further research.

The joint decoder only takes a joint code $\boldsymbol{z}_\text{j}$ as input and feeds it through a single layer outputting a feature vector with 64 dimensions. This vector is then used to regress the articulation state, consisting of the continuous joint state q (no activation) and the discrete joint type *jt* (Sigmoid activation).

An overview of the used loss scaling hyperparameters is given in Tab. S.1.

Scaling Variable	Value
$\delta_{\text{reg},\boldsymbol{z}_\text{j},\text{pre}}$	0.1
$\delta_{\text{jr,pre}}$	1.0
$\delta_{\text{reg},\boldsymbol{z}_\text{s}}$	0.0001
$\delta_{\text{reg},\boldsymbol{z}_\text{j}}$	0.001
δ_{rec}	1.0
δ_{jt}	0.001
δ_{q}	0.1
δ_{jr}	0.1

Table S.1. Scaling Hyperparameters for Decoder Training.

S.2. Backward Optimization

The goal of the backward optimization is to retrieve the shape code $\boldsymbol{z}_\text{s}^u$ and joint code $\boldsymbol{z}_\text{j}^u$ of an unknown object. The object is given through sampled SDF values, in total P. We denote the set of all P SDF-space coordinate tuples as $\boldsymbol{S}^u = \{(\boldsymbol{x}_p, s_p)_p \forall p \in P\}$ for this unknown object. The

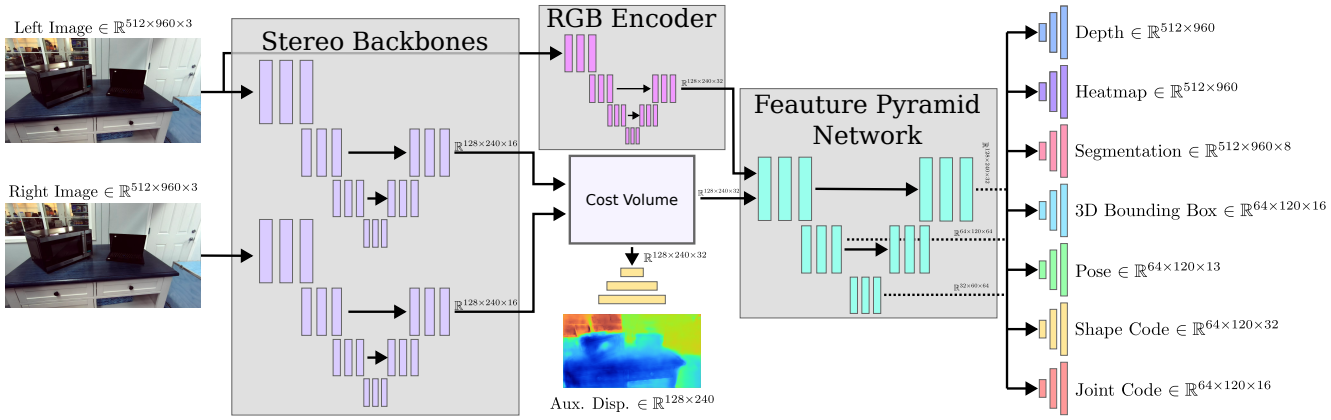


Figure S.1. Encoder Architecture based on [3]

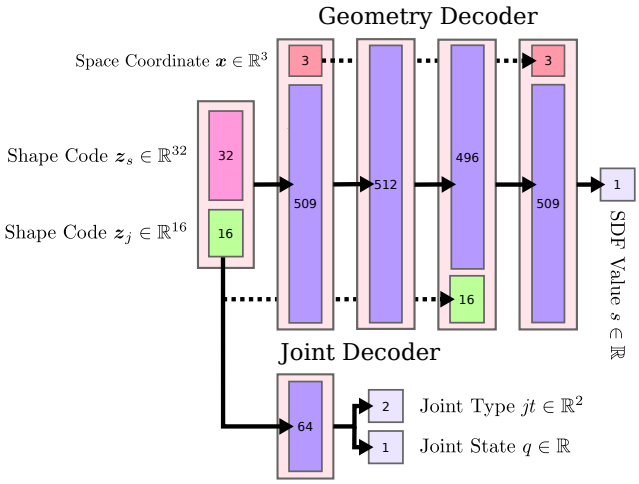


Figure S.2. Decoder Architecture. The numbers indicate the size of the respective feature vector. Each arrow represents a layer of a multi-layer perceptron. For the geometry decoder, except for the first layer, the input to a layer always has a size of 512. The output dimensions vary depending on auxiliary inputs.

problem can then be formalized as

$$\boldsymbol{z}_{\mathrm{s}}^{u}, \boldsymbol{z}_{\mathrm{j}}^{u} = \tag{S.1}$$
$$\arg\min_{\boldsymbol{z}_{\mathrm{s}}^{u}, \boldsymbol{z}_{\mathrm{j}}^{u}} \frac{1}{|\boldsymbol{S}^{u}|} \sum_{(\boldsymbol{x}_p, s_p)\in\boldsymbol{S}^{u}} \left|\phi_{\mathrm{geom}}(\boldsymbol{z}_{\mathrm{s}}^{u}, \boldsymbol{z}_{\mathrm{j}}^{u}, \boldsymbol{x}_p) - s_p\right|,$$

a minimization of the distance between the given SDF values and the one predicted by our geometry decoder (see Sec. 3.2.1).

At the beginning of the optimization, we randomly sample a set of 16 random shape codes from a zero-mean Gaussian distribution with a variance of 0.5 as well as a set of 16 corresponding joint codes. For the joint codes, we do not sample but rather take the mean from all final joint codes of the training set after training $\boldsymbol{z}_{\mathrm{j}}^{n} \forall n \in N$. We split the joint codes, where one half is using the mean of all prismatic training joint codes and the other half uses the mean of all revolute training joint codes. To guide the optimization through our latent joint code space, we propose a projection of the space as well as bounding the joint code variables.

SVD Projection: To facilitate optimization along significant axes we will construct a projection based on the singular value decomposition of our training joint codes. To that end, we stack all training joint codes

$$\boldsymbol{Z}_{\mathrm{j}} = \begin{bmatrix} \vdots \\ \boldsymbol{z}_{\mathrm{j}}^{n\mathrm{T}} \\ \vdots \end{bmatrix} \in \mathbb{R}^{N\times D_j} \tag{S.2}$$

and do a singular value decomposition

$$\boldsymbol{Z}_{\mathrm{j}} - \bar{\boldsymbol{Z}}_{\mathrm{j}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}, \tag{S.3}$$
$$\boldsymbol{U} \in \mathbb{R}^{N\times N}, \boldsymbol{\Sigma} \in \mathbb{R}^{N\times D_j}, \boldsymbol{V}^{\mathrm{T}} \in \mathbb{R}^{D_j\times D_j}. \tag{S.4}$$

We then use

$$\hat{\boldsymbol{z}}_{\mathrm{j}}{}^{u} = \left(\boldsymbol{z}_{\mathrm{j}}^{u} - \bar{\boldsymbol{Z}}_{\mathrm{j}}\right)\boldsymbol{V} \tag{S.5}$$

to *project* any joint code $\boldsymbol{z}_{\mathrm{j}}^{u}$ and

$$\boldsymbol{z}_{\mathrm{j}}^{u} = \hat{\boldsymbol{z}}_{\mathrm{j}}{}^{u}\boldsymbol{V}^{\mathrm{T}} + \bar{\boldsymbol{Z}}_{\mathrm{j}} \tag{S.6}$$

to *reproject* a joint code $\hat{\boldsymbol{z}}_{\mathrm{j}}{}^{u}$.

We carry out the optimization from Eq. (S.1) in our projected space and thus, initially we project our joint codes using Eq. (S.5). As well as in each optimization step, before inputting the joint code in our geometry decoder, we first reproject it using Eq. (S.6). In initial testing, we found that this projection-reprojection step greatly helps navigate the high-dimensional space in which our joint codes reside in.

Bound Joint Code Variables: On top of the previously described projection procedure, we ensure that the joint code variable is close to final joint codes from the training

Algorithm 1 Backward Optimization: The goal is to retrieve shape and joint code for an unknown shape $\boldsymbol{S}^u$ with n hypotheses in parallel. Here, for clarity, we show the optimization for a single $i \in [1, \dots, n]$. Eventually, from all returned code pairs the pair having the lowest distance error (see Eq. (S.1)) is returned. This procedure can be efficiently parallelized on a GPU.

procedure BACKOPTIMSINGLE($\boldsymbol{S}^u, \boldsymbol{Z}_\text{j}, i$)
  project($\bullet$), reproject($\bullet$) $\leftarrow$ SVD($\boldsymbol{Z}_\text{j}$) ▷ Retrieve project (see Eq. (S.5)) and reproject (see Eq. (S.6)) function
  $\boldsymbol{z}_\text{s}^i \leftarrow \mathcal{N}(\mathbf{0}, \Sigma)$ ▷ Initialize shape codes with $\mathbf{0} \in \mathbb{R}^{D_s}, \Sigma = \text{diag}(0.5) \in \mathbb{R}^{D_s \times D_s}$
  $\boldsymbol{z}_\text{j}^i \leftarrow \begin{cases} \bar{\boldsymbol{Z}}_\text{j}^\text{prismatic} & i \mod 2 = 0 \\ \bar{\boldsymbol{Z}}_\text{j}^\text{revolute} & i \mod 2 = 1 \end{cases}$ ▷ Initialize joint codes
  $\hat{\boldsymbol{z}}_\text{j}{}^i \leftarrow \text{project}(\boldsymbol{z}_\text{j}^i)$ ▷ Project joint codes
  for *step* $\in [1, \dots, 800]$ **do**
    $$loss^i \leftarrow \frac{1}{|\boldsymbol{S}^u|} \sum_{(\boldsymbol{x}_p, s_p) \in \boldsymbol{S}^u} \left| \phi_\text{geom}(\boldsymbol{z}_\text{s}^i, \text{reproject}(\hat{\boldsymbol{z}}_\text{j}{}^i), \boldsymbol{x}_p) - s_p \right| + 5 \cdot 10^{-3} ||\boldsymbol{z}_\text{s}^i|| + 10^{-2} \min(||\text{reproject}(\hat{\boldsymbol{z}}_\text{j}{}^i) - \boldsymbol{Z}_\text{j}||)$$
    ▷ Sum distance loss and regularization terms
    if *step* ≤ 600 **then**
      $\boldsymbol{z}_\text{s}^i, \hat{\boldsymbol{z}}_\text{j}{}^i \leftarrow \text{ADAM}(loss^i)$ ▷ Update shape and joint code
    else if $600 <$ *step* ≤ 700 **then**
      $\boldsymbol{z}_\text{s}^i \leftarrow \text{ADAM}(loss^i)$ ▷ Update shape code
    else
      $\hat{\boldsymbol{z}}_\text{j}{}^i \leftarrow \text{ADAM}(loss^i)$ ▷ Update joint code
    end if
  end for
  return $\boldsymbol{z}_\text{s}^i, \text{reproject}(\hat{\boldsymbol{z}}_\text{j}{}^i)$
end procedure

examples $\boldsymbol{Z}_\text{j}$ through minimizing the minimum distance to any joint code in the training set:

$$\min(||\boldsymbol{z}_\text{j}^u - \boldsymbol{Z}_\text{j}||), \tag{S.7}$$

where $||\bullet||$ is the row-wise Euclidean norm and $\min(\bullet)$ is a differentiable operator returning the minimum of a vector. An outline of our full optimization is presented in Algorithm 1.

S.3. Learned Joint Code Space

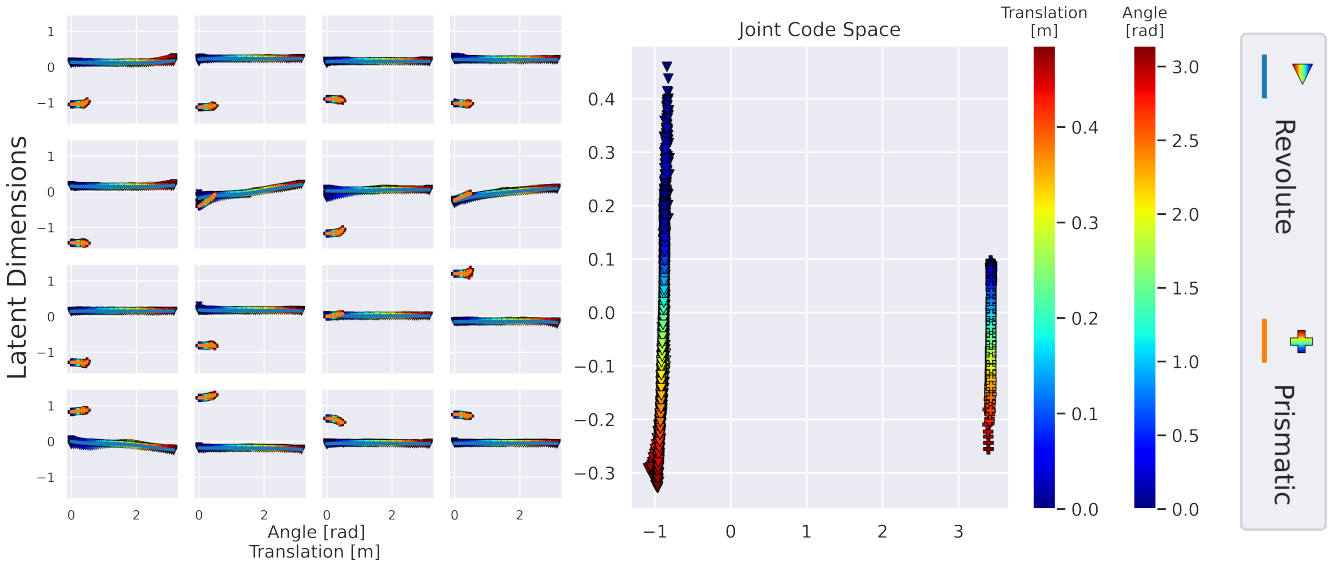
In this section, we visualize and compare the resulting learned latent joint space using our in Sec. 3.3.1 introduced regularization against naively training it. In Fig. S.3, we visualize the learned joint codes of the training results for *CARTO* and *CARTO-No-Enf* from Sec. 4.2. When comparing both visualizations, we can explain the worse performance of *CARTO-No-Enf* in Tab. S.3.

CARTO trained without regularization struggles to correctly align the spaces such that joint codes belonging to the same articulation state, independent of the object, are close and show a low variance. The decoder rather learns to represent the final geometry of the articulated object jointly through both codes, the shape code $\boldsymbol{z}_\text{s}$ and joint code $\boldsymbol{z}_\text{s}$ instead of disentangling one from the other. One could argue that this case is similar to not splitting the codes. Compared to that, when using our proposed regularization, we learn a cleaner disentanglement between the shape and the articulation state of the object. Joint codes of similar articulation states in the training set are arranged closer in the latent joint embedding and thus, the variance in the y-direction across all plots on the left side of Fig. S.3 (a) is much lower when compared to training without our regularization in Fig. S.3 (b). Moreover, two distinct clusters are visible (prismatic and revolute) whereas without CARTOs regularization different joint types overlap.

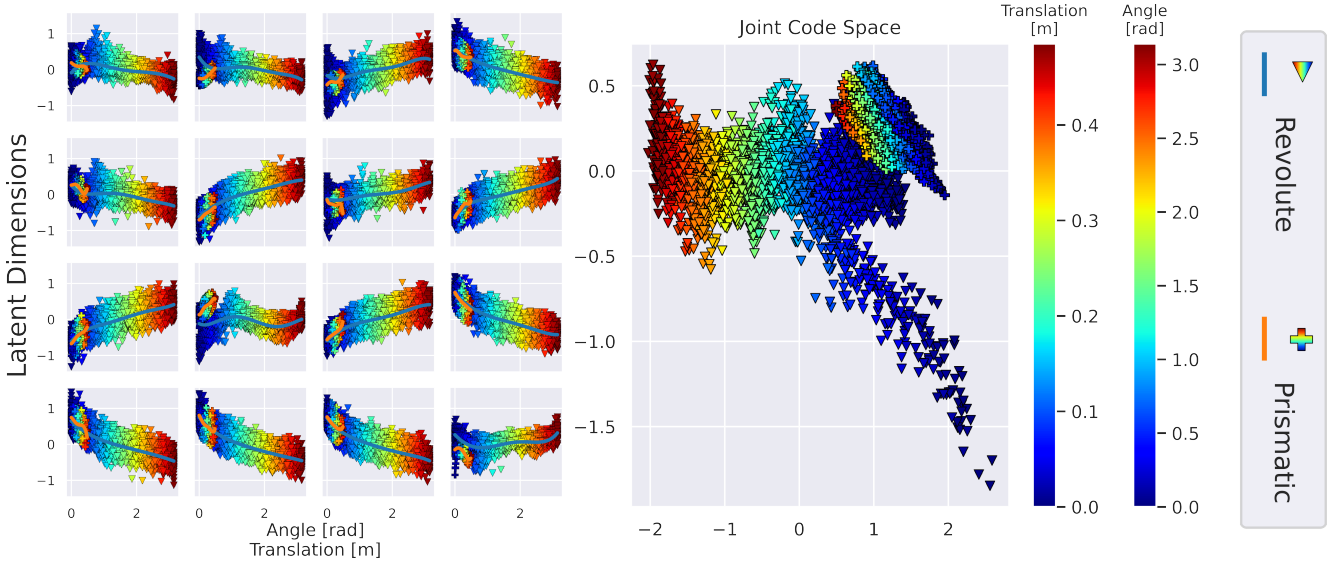
S.4. Datasets

Fig. S.4 presents exemplary images of our procedural generated kitchen dataset. In total, we collected roughly 100k images for training and 20k images for testing. Due to the long run-time of our A-SDF baseline, we only evaluated on the first 2000 samples in which according to our ground truth objects are present. See Fig. S.5 for exemplary images from our collected real-world dataset.

In Tab. S.2 we compare our dataset against the RBO [4] and the BMVC [5] dataset.



(a) Using CARTOs Regularization.



(b) Only norm Regularization.

Figure S.3. Comparison of Learned Joint Code Space. We compare the learned embedding of training joint codes when using our proposed regularization (a) against naively just regularizing the norm (b). An articulation state is expressed two-fold. First, by its form to represent the joint type. Here upside-down triangles stand for revolute and cross for prismatic joint types. Second, the form is colored by its joint state according to the scale shown on the right. In the left figure, each plot represents one component of the joint code $\boldsymbol{z}_j \in \mathbb{R}^{16}$. In the i-th plot, we plot the i-th component of all training joint codes on the y-axis against their associated known joint state on the x-axis. Additionally, we overlay the in Sec. 3.3.3 explained polynomial functions. In the right figure, we show a two-dimensional projection based on singular value decomposition of all training joint codes.

Dataset	Categories available as 3D Models *(All)*	Instances per Category	Total Unique Scenes	Input Modality	Realism	Corresponding Synthetic Images
RBO [4]	4 *(14)*	1	**385**	RGB-D	Lab + Human Occlusion	No
BMVC [5]	2 *(3)*	1	8	RGB-D	**Real Environments**	No
Ours	**7** *(7)*	**2***	9	**Stereo-RGB + RGB-D**	**Real Environments**	**Yes**

Table S.2. Full Pipeline Dataset Comparison. While RBO [4] and BMVC [5] contain additional categories, these categories lack corresponding 3D meshes in public datasets such as PartNetMobility [8]. * *one washing machine instance*

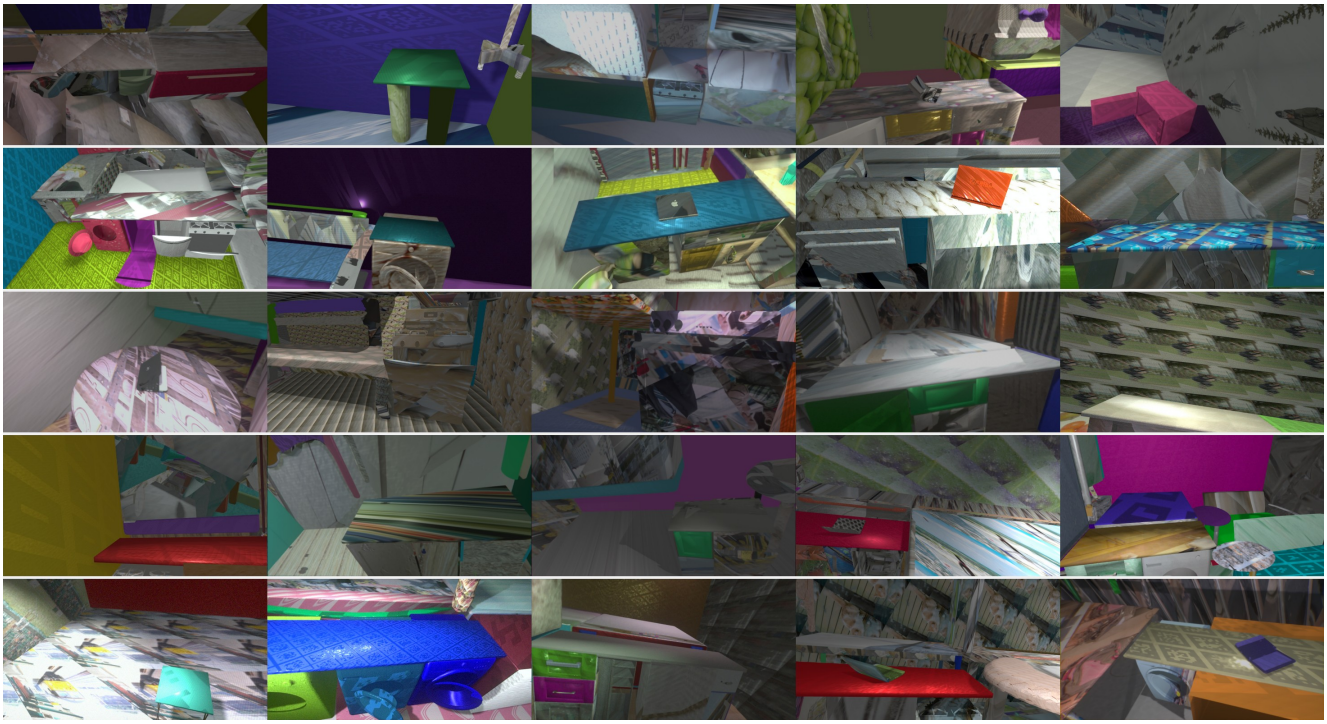


Figure S.4. Synthetic Example Images

Figure S.5. Real Example Images

Table S.3. Decoder Optimization Results. Each object is sampled in 50 different joint states for training as well as for testing. † *means the model is trained only on a single category.*

Method	Dishwasher	Laptop	Microwave	Oven	Refrigerator	Table	Washing Machine	Instance Mean	Category Mean
A-SDF [6] †	2.162	0.264	2.256	1.540	1.409	1.465	0.960	1.437	1.418
CARTO †	1.874	0.965	1.429	1.606	0.690	1.330	0.437	1.190	1.252
A-SDF [6]	**0.336**	**0.601**	**0.700**	**0.883**	1.425	1.862	**0.608**	**0.934**	**0.916**
CARTO-No-Enf	1.043	3.820	2.685	2.317	2.454	**1.676**	1.727	2.246	2.248
CARTO	0.554	1.448	0.782	2.056	**0.988**	1.688	0.830	1.192	1.181

(a) Shape Reconstructions Results. We report the bi-directional L2-Chamfer distance (CD) (↓) times 1000 between the original mesh and the reconstructed version.

Method	Dishwasher	Laptop	Microwave	Oven	Refrigerator	Table	Washing Machine	Instance Mean*	Category Mean*
A-SDF [6] †	1.616°	17.282°	11.161°	4.045°	19.254°	0.094m	16.462°	11.337°	11.636°
CARTO †	6.264°	6.818°	20.425°	8.156°	21.456°	0.081m	21.057°	12.474°	14.029°
A-SDF [6]	**3.457**°	30.740°	7.189°	**3.884**°	34.714°	0.235m	12.265°	16.139°	15.375°
CARTO-No-Enf	29.289°	37.476°	36.086°	46.648°	37.856°	**0.104**m	34.451°	35.892°	36.967°
CARTO	8.214°	**10.678**°	**6.815**°	14.136°	**23.467**°	0.141m	**8.328**°	**11.512**°	**11.940**°
A-SDF [6]	**1.000**	**0.956**	0.933	**0.99**	**0.9**	**0.988**	0.923	**0.962**	**0.957**
CARTO-No-Enf	0.604	0.748	0.727	0.600	0.700	0.496	0.710	0.646	0.655
CARTO	0.932	0.932	**0.973**	0.570	0.867	0.956	**0.970**	0.908	0.886

(b) Articulation State Prediction Results. We report the joint state error (↓) in the first set of rows for all correctly classified joints and joint type accuracy (↑) in the last set of rows. As A-SDF does not classify the joint type and CARTO trained on a single category always predicts the correct joint type, we do not report joint accuracy for those models. **The joint state error mean is only reported across the revolute categories, as there is only one prismatic category.*

Table S.4. Reconstruction and Articulation State Prediction Results when using A-SDF [6] with the Proposed Test-Time-Adaptation. **The joint state error mean is only reported across the revolute categories, as there is only one prismatic category.*

Method	Dishwasher	Laptop	Microwave	Oven	Refrigerator	Table	Washing Machine	Instance Mean*	Category Mean*
Chamfer Distance (↓)	0.101	1.035	0.529	0.451	1.383	64.097	0.332	13.339	9.704
Joint State Error (↓)	19.387°	18.675°	58.088°	25.432°	20.700°	0.552m	51.467°	29.024°	32.292°

S.5. Additional Experimental Results

In this section, we present additional metrics for our experiments. Namely, using A-SDFs proposed test time adaption as well as the Chamfer distance and joint state error for the full pipeline experiment. Also, in addition to Tab. 2, we report the more fine-grained category-level metrics in Tab. S.3.

S.5.1. Canonical Reconstruction Task: A-SDF TTA

In our experiments (see Sec. 4.2), the proposed test time adaptation (TTA) [6] did not prove to be stable. We report the results in Tab. S.4. While for some object instances TTA reduces the Chamfer distance, for the table category TTA does not prove to be robust. Additionally, the joint state error increases substantially. Both behaviors are reasonable when reflecting on the proposed TTA. When jointly optimizing the input shape code, the joint state, and network weights, the entire network will overfit to the single given geometry. Thus, it is easier to achieve a lower Chamfer distance. Whereas, the joint state variable becomes unbound from other examples and can be optimized freely, losing its meaning and therefore, potentially resulting in a high joint state error.

The proposed TTA is still promising and with further investigation into how to mitigate the aforementioned problems, it can prove to be an ideal tool for reconstructing (articulated) objects in the wild [2].

S.5.2. Extended Metrics for Full Pipeline

In addition to the tabular values reported in Tab. 3a, we present the respective mAP curve in Fig. S.6. The results highlight even more that an optimization-based two-stage approach suffers from its partial input. The two counter objects, laptops and microwaves, which are free-standing and thus much more points for reconstruction are available get reconstructed much better compared to other objects. On the other hand, for these objects, the predicted rotation is much worse. This can be rooted in the fact that for all other objects, we can learn a strong prior of the rotation being roughly camera facing, whereas, for laptops and microwaves the range of the possible rotation is much higher as they are placed freely on top of the counter.

While in Tab. 3a and previously we only discussed the overall 3D IoU and pose error, which gives a holistic evaluation of the full pipeline, we additionally report object-centric L2-Chamfer distances (similar to [1, 2]) multiplied by 10^3, as well as the joint state error in Fig. S.7. Since this is an object-centric evaluation and should not evaluate the detection quality, we are very forgiving in selecting our detection matches. For each scene, we calculate our spatial 2D detections and retrieve the ground-truth spatial 2D detections from the heatmap, we then match the predicted and ground truth detections by solving a linear sum assignment problem, ignoring unmatched detections (either ground-truth or predicted). For each matched detection we then reconstruct the object as before using our geometry decoder and retrieve the joint through our joint decoder. We then calculate the Chamfer distance between the predicted points and the ground-truth points and compare the joint states.

In this experiment, we observe the same trend as for 3D IoU. One major difference is that *A-SDF-GT* reconstructs laptops more accurately compared to *A-SDF* and *CARTO* which can be attributed to laptops having the least occlusion (either through self-occlusion or other objects).

S.5.3. CARTO RGB-D Version

In addition to the proposed stereo-RGB input version of *CARTO*, we also evaluated and tested an RGB-D version *CARTO-D*. We report quantitative results on the same synthetic dataset in Tab. S.5 as well as compare the detections qualitatively in Fig. S.8.

Quantitatively, the *CARTO-D* performs slightly better compared to our proposed stereo RGB version. This is to be expected given that CARTO needs to learn the notion of depth first whereas *CARTO-D* does not. Contrary to this observation, in our real world experiments, we do not get a single meaningful detection using the RGB-D input version (see Fig. S.8). Thus, overall, we decided for the proposed stereo version of CARTO.

Table S.5. Full Scene Reconstructions Results with RGB-D Input.

Method	IOU25 ↑	IOU50 ↑	10°10cm ↑	20°30cm ↑
CARTO	64.0	31.5	**28.7**	76.6
CARTO-D	**67.8**	**38.2**	27.0	**84.7**

References

[1] Muhammad Zubair Irshad, Thomas Kollar, Michael Laskey, Kevin Stone, and Zsolt Kira. Centersnap: Single-shot multi-object 3d shape reconstruction and categorical 6d pose and size estimation. In *Proc. IEEE Int. Conf. on Rob. and Auto.*, pages 10632–10640, 2022. 7

[2] Muhammad Zubair Irshad, Sergey Zakharov, Rares Ambrus, Thomas Kollar, Zsolt Kira, and Adrien Gaidon. Shapo: Implicit representations for multi-object shape, appearance, and pose optimization. In *Proc. Springer Eur. Conf. Comput. Vis.*, pages 275–292, 2022. 7

[3] Thomas Kollar, Michael Laskey, Kevin Stone, Brijen Thananjeyan, and Mark Tjersland. Simnet: Enabling robust unknown object manipulation from pure synthetic data via stereo. In *Proc. Conf. on Rob. Learn.*, pages 938–948, 2021. 1, 2

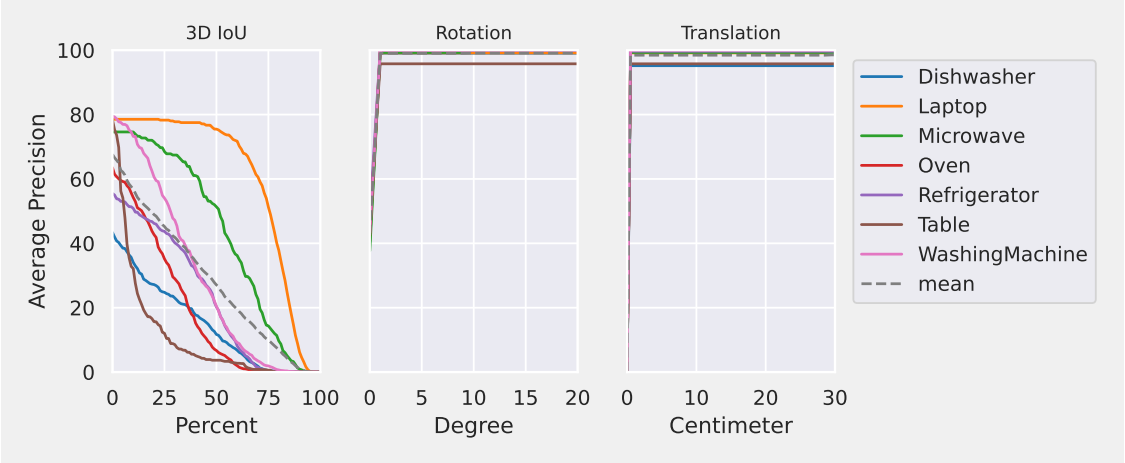
[4] Roberto Martín-Martín, Clemens Eppner, and Oliver Brock. The RBO Dataset of Articulated Objects and Interactions, June 2018. 3, 4

[5] Frank Michel, Alexander Krull, Eric Brachmann, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Pose estimation of kinematic chain instances via object coordinate regression. In *Proc. Brit. Mach. Vis. Conf.*, pages 181–1, 2015. 3, 4

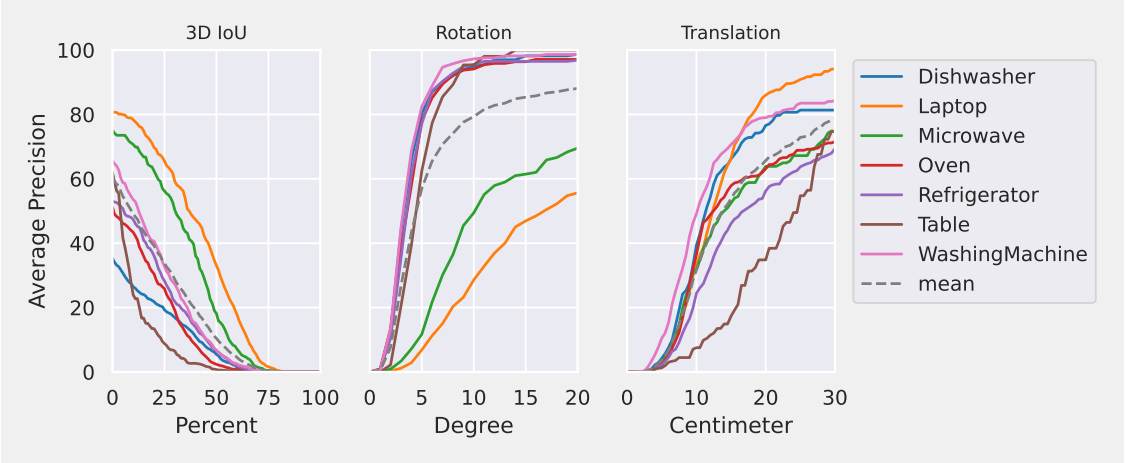
[6] Jiteng Mu, Weichao Qiu, Adam Kortylewski, Alan Yuille, Nuno Vasconcelos, and Xiaolong Wang. A-SDF: Learning disentangled signed distance functions for articulated shape representation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 13001–13011, 2021. 1, 6, 7

[7] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 165–174, 2019. 1

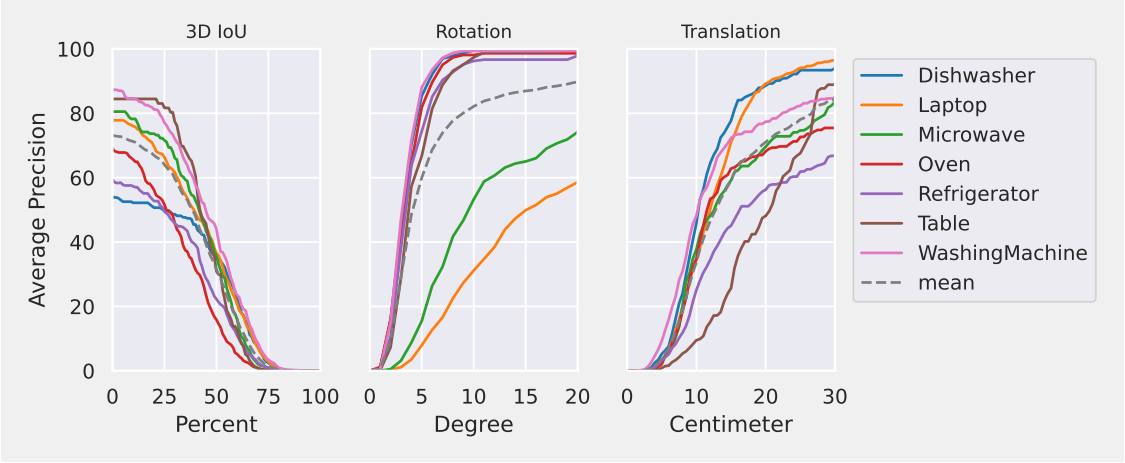
[8] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11097–11107, June 2020. 4



(a) A-SDF-GT

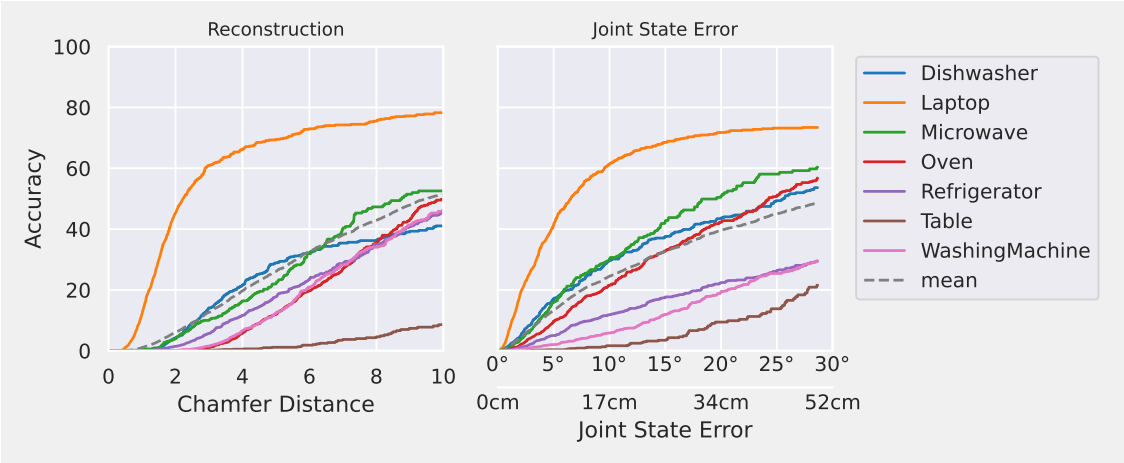


(b) A-SDF

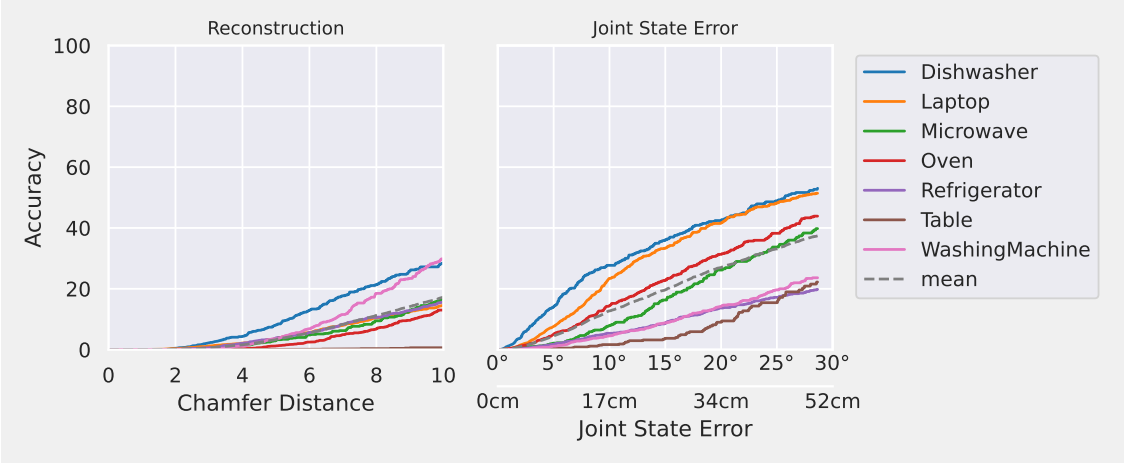


(c) CARTO

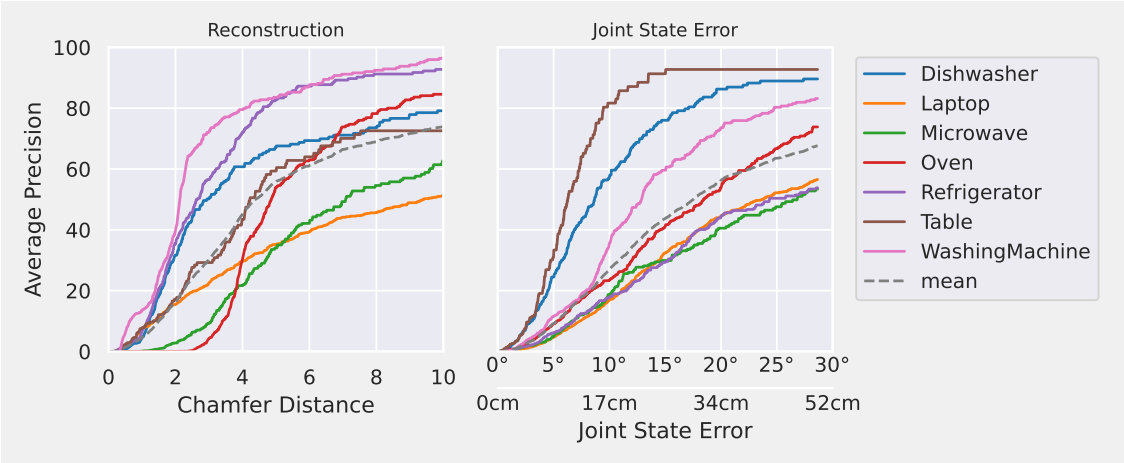
Figure S.6. Detailed metrics for the experiment presented in Sec. 4.3. We report the average precision for the 3D IoU and the pose prediction for each category in our test set as well as the mean over all instances. It can be observed that overall the mean 3D IoU is lower for *CARTO* compared to *A-SDF-GT* and *A-SDF*. For *A-SDF-GT* the laptop and microwave category stands out as they are mostly placed on counters and thus they are less occluded than other objects. As expected, the poses predicted by *A-SDF* and *CARTO* are similar as they both use the same pose map predicted by our encoder.



(a) A-SDF-GT



(b) A-SDF



(c) CARTO

Figure S.7. Additional metrics for the experiment presented in Sec. 4.3. In addition to the metrics already reported in Sec. 4.3, we report the more fine-grained object-centric Chamfer distance as well as the joint state prediction error. Both metrics show a similar trend as the more coarse 3D IoU. One can observe though, that for the laptop category *A-SDF-GT* performs significantly better than all other categories. Compared to that *A-SDF*, which uses a predicted segmentation masks, does not show this special behavior for laptops. As laptops are small and the segmentation mask is very thin, this gap in performance highlights potential failure cases of an optimization-based method due to imperfect segmentation masks.

Figure S.8. Stereo-RGB Image (Left) vs. RGB-D Image (Right) Input. The first row shows a successful detection and reconstruction of CARTO in an office kitchen environment. Second row shows a reconstruction of a cabinet. Eventhough, CARTO has never seen objects from this category it highlight its generalization beyond the trained categories. The third and fourth row show two failure cases of either no detection at all (third row) or a misdetection of a laptop on the kitchen counter (fourth row). CARTO with RGB-D input is not able to reconstruct any objects.