

A. Supplementary material

A.1. Implementation details

Dataset preparation. For CUB [52], we use the segmentation masks and poses from CMR [27] estimated using structure-from-motion. These poses adopt a weak-perspective camera model, which we keep as-is (our neural renderer implementation supports multiple projection models). Similarly, for P3D Cars [59], we use poses from CMR but obtain the segmentation masks using Mask R-CNN [21] as was done in previous work. Additionally, we upgrade its camera projection model to a full perspective model by freezing the rotations and re-estimating all the other parameters using the procedure described in the next paragraph. For ImageNet, we estimate poses from scratch using the same procedure and predict segmentation masks using PointRend [32]. Additionally, we augment all real datasets mentioned so far with horizontal flips. We do not augment synthetic datasets (CARLA [11] and ShapeNet [7]).

Pose estimation and parameterization. While most neural renderers adopt *camera-to-world* view matrices (plus focal length), this representation is not necessarily the best for optimization purposes. Among various issues, we mention the necessity to enforce orthogonality constraints in the rotation matrix, a dependency between rotation and translation (which can be solved by switching to a *world-to-camera* representation), and an entanglement between translation and focal length (ideally, as depth increases, the learning rate for the translation needs to be amplified in order to keep a linear behavior in projective space). Therefore, for all our steps where pose optimization is involved (namely, the initial data preparation and the hybrid inversion step), we adopt a custom pose parameterization that tackles these issues and is easier to optimize, while being fully differentiable and convertible to view matrices for use in neural renderers. We also make sure that our neural renderer is fully differentiable w.r.t. the pose, even when coarse-fine importance sampling is used and view-dependent effects are enabled, as we found that existing implementations present gradient detachments in some nodes of the pipeline.

Our pose representation can be regarded as an augmentation of a weak-perspective camera model, and describes a *world-to-camera* transformation parameterized by a rotation $\mathbf{q} \in \mathbb{R}^4$ (a unit quaternion), a screen-space scale $s \in \mathbb{R}$, a screen-space translation $\mathbf{t}_2 \in \mathbb{R}^2$, and a perspective distortion factor $z_0 \in \mathbb{R}$. At runtime, we derive the focal length $f = 1 + \exp(z_0)$ and the 3D translation $\mathbf{t}_3 = [\mathbf{t}_2/s; f/s]$. Such a parameterization is equivalent to a full-perspective model, but results in more “linear” optimization dynamics.

For the datasets where we estimate the poses ourselves (ImageNet and, partially, P3D Cars), we use the template-based pose estimation technique described in [44] with our pose parameterization.

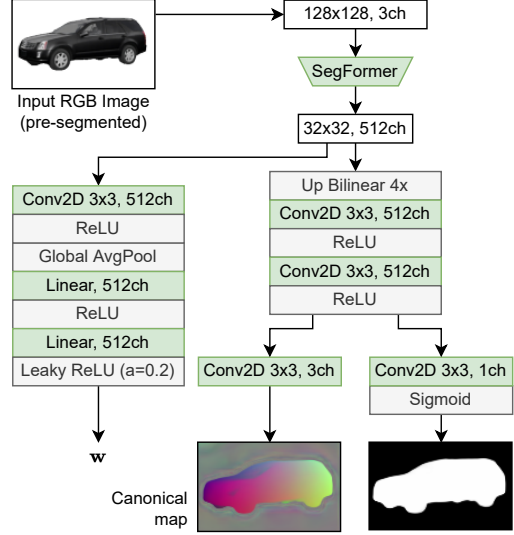


Figure 10. Architecture of the encoder used for bootstrapping the latent code and pose estimation. Note that we include a Leaky ReLU activation in the final layer of the latent code regressor, which mimics the behavior of the mapping network.

Unconditional generator. We train the unconditional generator for 300k iterations, except for CUB and ImageNet elephants, where we use 200k. Similarly, we adopt R1 regularization on all datasets with $\gamma = 5$, except on elephants where we use $\gamma = 10$. Optimizer, learning rate, and batch size are the same as in [5]. We found it beneficial for stability to warm up the learning rate of both the generator and discriminator, starting from 1/10th of the specified value and linearly increasing it over 2000 iterations.

As in [5], we condition the discriminator on the pose, but we nonetheless observe that all our models converge even without such conditioning, although this sometimes leads to surface artifacts (such as objects appearing concave). On all datasets except ShapeNet, we enable adaptive discriminator augmentation (ADA) [28], which reduces discriminator overfitting by enhancing its input images with differentiable augmentations. We only adopt geometric transformations (scale, translation, rotation). However, we observe that the implementation of ADA in [5] is not 3D-aware, as the augmentations are only carried out in image space, while the discriminator is conditioned on the original camera pose, leading to artifacts. We implemented a 3D-aware version of ADA where both the image and camera pose are augmented with the same transformation, and the discriminator is conditioned on the augmented pose.

SDF details. As for the SDF representation [65], the volume density is modulated by two learnable parameters $\alpha, \beta > 0$. Unlike [65], which ties $\alpha = \beta$ (according to Equation 1), we found it helpful for convergence to learn them separately. We initialize $\alpha = 1$ and $\beta = 0.1$, and

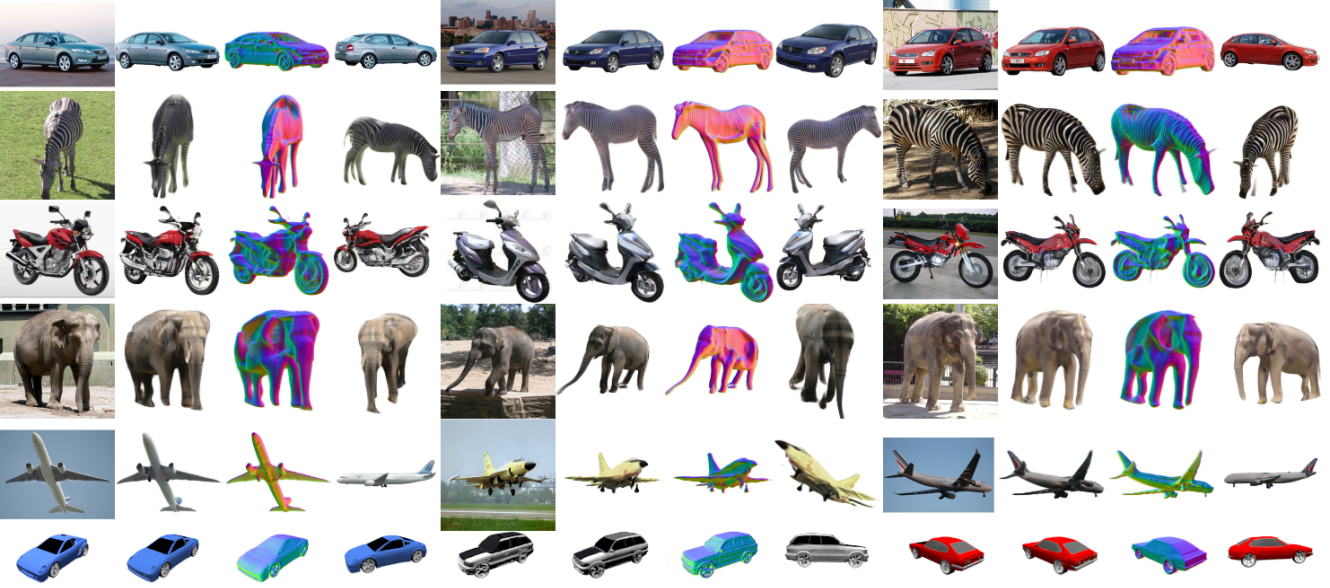


Figure 11. Additional qualitative results produced by our method on ImageNet (top rows) and ShapeNet Cars (last row).

clamp their lower bound to 10^{-3} for stability during training. Before training the unconditional model, we initialize the SDF to a unit sphere through optimization. We pre-train the model for 1000 iterations using the following loss:

$$\mathcal{L}_{\text{SDF}} = \mathbb{E}_{\mathbf{x}} \left[(d(\mathbf{x}) - (\|\mathbf{x}\| - 1))^2 \right]. \quad (3)$$

A visualization of the SDF can be seen in Fig. 13 (right). For the Eikonal loss, we use a weight of 0.1 as recommended by [65].

View-dependent effects. We also incorporate the ability to efficiently model view-dependent effects. The view direction of each pixel (which is constant across depth, and can therefore be computed only once per pixel) is processed through a small feed-forward network to produce a 32-dimensional vector, and summed with another 32-dimensional vector coming from the triplanar decoder. The result is then processed through a Leaky ReLU activation and another linear layer to produce the final output. This late fusion strategy ensures that memory consumption is minimal. We enable this feature only on CARLA, as ShapeNet does not have any specular reflections and the other datasets are too small to properly disentangle appearance and specular effects.

Bootstrapping and pose estimation. For the encoder architecture, we adopt SegFormer B5 [60], a recently-proposed transformer-based backbone for semantic segmentation. The output feature map from the backbone is connected to two heads: a fully-connected one that regresses the latent code \mathbf{w} and a convolutional one that regresses the canonical map and the associated segmentation mask. The detailed architecture is shown in Fig. 10. We initialize the backbone using ImageNet weights and train

the model end-to-end for 120k iterations with a batch size of 32 samples, using Adam optimizer. We adopt an initial learning rate of $6e-5$, which we decay to $6e-6$ after 60k iterations. As for the losses, we use a simple mean squared error (MSE) loss for the latent code ($\mathcal{L}_{\text{latent}}$), an L1 loss for the segmentation mask ($\mathcal{L}_{\text{mask}}$), and a masked L2 loss (with square root) for the canonical map (\mathcal{L}_{map}), *i.e.* a rotation-invariant version of the L1 loss, for better robustness to artifacts coming from the generator:

$$\mathcal{L}_{\text{latent}} = \|\hat{\mathbf{w}} - \mathbf{w}\|^2, \quad (4)$$

$$\mathcal{L}_{\text{mask}} = \frac{1}{WH} \sum_{i=1}^W \sum_{j=1}^H |\hat{m}_{i,j} - m_{i,j}|, \quad (5)$$

$$\mathcal{L}_{\text{map}} = \frac{1}{WH} \sum_{i=1}^W \sum_{j=1}^H m_{i,j} \|\hat{\mathbf{p}}_{i,j} - \mathbf{p}_{i,j}\|, \quad (6)$$

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{latent}} + \mathcal{L}_{\text{mask}} + \mathcal{L}_{\text{map}} \quad (7)$$

where $\hat{\mathbf{w}}$ is the predicted latent code, $\hat{\mathbf{p}}_{i,j}$ is the predicted canonical map at the i, j image-space coordinates (a 3D vector for each position), \mathbf{p} is the ground-truth one, \hat{m} is the predicted mask, and m is the ground-truth mask. This contrasts NOCS [53], which frames the task as a classification problem using quantized coordinates. For inference, the regressed canonical map is thresholded using the predicted mask, converted to a dense point cloud, and used as the input for SQPnP [51], a fast PnP solver that retrieves a global optimum (we use the implementation in OpenCV [3]). Since SQPnP – as most PnP methods – requires the focal length to be pre-determined, we select 10 representative focal lengths from the training set (one for each 10th per-

centile), run the algorithm for each, and select the solution with the lowest reprojection error.

GAN inversion. For the hybrid inversion step, we use Adam optimizer [31] with a base learning rate of 0.02, and optionally amplify the learning rate of the latent code \mathbf{w} by a gain factor (as reported in the individual experiments). We additionally set $\beta_2 = 0.95$ for a faster reaction. We do not dynamically adjust the hyperparameters throughout the procedure. For the pose, we use our previously-described parameterization as this results in better optimization dynamics. We optimize the following objective:

$$\min_{\mathbf{w}, \mathbf{q}, \mathbf{s}, \mathbf{t}_2, \mathbf{z}_0} \frac{1}{K} \sum_{k=1}^K \text{LPIPS}(\mathbf{c}_{\text{pred}}^{[k]}, \mathbf{c}_{\text{gt}}^{[k]}), \quad (8)$$

where \mathbf{c}_{pred} represents the predicted image, \mathbf{c}_{gt} is the ground-truth one, k is the augmentation index (we use $K = 16$ augmentations), and the LPIPS operator [70] describes the distance between the VGG embeddings of the two images. After each iteration, we reproject the pose parameters onto the valid set of constraints (\mathbf{q} unit length).

Evaluation details. For the evaluation on real datasets (P3D, CUB, ImageNet), we follow the protocol of [44, 45] and evaluate the FID on an empty background (value = 0, *i.e.* gray). All scores are evaluated at 128×128 , using antialiased resampling (also referred to as *area* interpolation) if resizing is needed, as the FID is sensitive to this aspect. For the approaches we compare to, we compute their FID under the same settings by modifying their public implementations.

A.2. Additional results

A.2.1 Ablation experiments

Encoder-based architecture. For our next experiment, we build an encoder-based variant of our architecture and

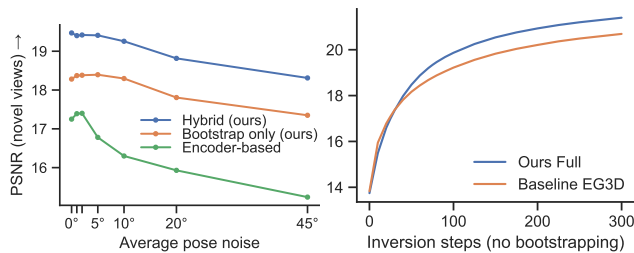


Figure 12. Additional ablations on ShapeNet Chairs, where we evaluate the PSNR on novel views from the test set. **Left:** comparison of our hybrid inversion approach (and initial bootstrapping without refinement) to an encoder-based baseline under simulated pose perturbations. **Right:** inversion on a vanilla EG3D backbone *vs* our proposed architecture. Since the goal of this experiment is to evaluate only the impact of the unconditional generator, we start from an average latent code (*i.e.* no bootstrapping) and use the ground-truth pose.

switch to a conditional GAN setting. Instead of using a mapping network to map \mathbf{z} to \mathbf{w} , we learn a convolutional encoder that takes a 2D image as input and directly predicts \mathbf{w} . We also experiment with various supervision strategies, including a dual discriminator (an unconditional one that discriminates random views plus a conditional one that discriminates the input view), and a single unconditional discriminator with an L1 or MSE loss to fit the input image, as in Pix2NeRF [4]. Based on early experiments, we found that the dual discriminator approach yields the best results (as it does not require balancing the losses), and we use this strategy throughout our ablations. Moreover, for a fair comparison, we use the same backbone for the conditional (encoder-based) and unconditional (inversion-based) experiments.

Encoder- vs inversion-based baselines. In Fig. 12 (left), we compare our hybrid inversion approach to the aforementioned encoder-based baseline. We conduct this experiment on ShapeNet Chairs, where exact ground-truth poses are known. In this setting, we randomly perturb individual poses by injecting noise at different levels (from 0° to 45°) without altering the overall pose distribution, and study which approach is more robust to inaccurate poses as noise increases. Importantly, for a fair comparison to encoder methods (which are feed-forward), we also include a *bootstrap only* baseline where we do not refine the initial guess of our solution. We immediately observe that our *bootstrap only* baseline achieves a higher PSNR compared to the encoder-based approach. Furthermore, the performance of our approach decreases gracefully as noise increases, whereas with the encoder-based method we can observe a sharp degradation as early as 5°. Based on these findings, we conclude that inversion-based approaches are more appropriate for real datasets where poses are potentially inaccurate, as these methods rely on the overall pose distribution as opposed to the correctness of individual poses.

Impact of the backbone. In Fig. 12 (right), we assess the impact of the backbone on the final reconstruction result. We leave out some of our contributions (bootstrapping, pose

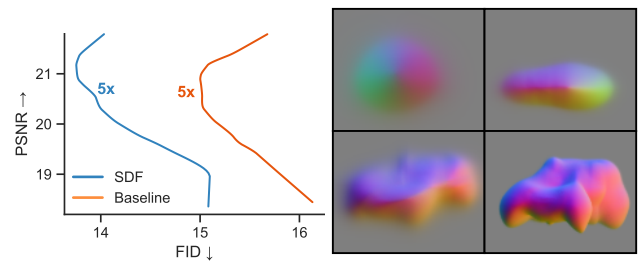


Figure 13. **Left:** impact of the SDF representation on the inversion dynamics (at gain 5x). As in Fig. 6, the analysis is carried out on our larger P3D Cars test set. **Right:** spherical initialization of the SDF and its evolution as training progresses.

estimation, and hybrid inversion) and purely focus on our proposed backbone components (SDF, color mapping, optimized path length regularization, as well as the minor improvements over [5]). To this end, we conduct a vanilla inversion experiments on ShapeNet Chairs, using ground-truth poses and starting from the “average” latent code in \mathcal{W} . We compare our full backbone to a vanilla EG3D, and observe an advantage when adopting our proposed changes.

SDF representation. Similar to Fig. 6, we conduct an analysis of the inversion dynamics with and without our proposed SDF representation (Fig. 13). We find that the optimization dynamics are similar, but the SDF baseline gets an FID boost owing to a better unconditional generator, in addition to the other practical benefits (*e.g.* ability to easily extract surface, normals, and mesh).

Color distribution disentanglement. To visually motivate our color mapping approach, we show examples of color disentanglement in Fig. 14. When our color mapping network is used, the object identity is fully disentangled from its color distribution. By contrast, attempting the same on a vanilla EG3D architecture is unsuccessful, even when adopting techniques such as *style mixing*.

Pose estimation. Pose prediction is a useful feature for AR applications and real-world datasets, where ground-truth poses are imprecise or not available. As such, it is not meant to improve performance, as it actually makes the learning task harder. Nonetheless, it is interesting to evaluate its effect on quantitative metrics. In Table 4, we conduct an ablation experiment on a dataset of real images (P3D Cars) where we turn off pose prediction and use ground-truth poses from the dataset (which are imprecise). As expected, we find that qualitative metrics (FID) are mostly unaffected, whereas the IoU is degraded when switching to poses from the dataset, regardless of whether inversion is used. This confirms that, on real datasets such as P3D, individual poses are inaccurate, but our generative framework is robust to them as it relies on the overall pose distribution.

| | | Pascal3D+ Cars | |
|------------------------|------------|----------------|------------------|
| | | IoU \uparrow | FID \downarrow |
| Ours (predicted poses) | ($N=0$) | 0.883 | 75.90 (15.08) |
| Ours (dataset poses) | ($N=0$) | 0.803 | 73.20 (16.39) |
| Ours (predicted poses) | ($N=30$) | 0.920 | 73.53 (14.36) |
| Ours (dataset poses) | ($N=30$) | 0.802 | 72.22 (15.10) |

Table 4. Ablation experiment on pose prediction (P3D Cars dataset). The first two rows are purely feed-forward-based, while the remaining are inversion-based. In parentheses, we also report the FID on our larger test set from ImageNet.



Figure 14. Disentanglement of the color distribution from the object identity. (a) In the baseline network without color mapping (EG3D), we attempt to achieve disentanglement via *style mixing*, *i.e.* we split the latent code w into two sections (before and after the 8th layer) and mix it between the two object instances. Although this leads to some variation in color, we find that disentanglement is not properly achieved. (b) With our color mapping technique, color and object identity are fully disentangled. When two different latent codes are combined, it is possible to “borrow” the color distribution from another image in a realistic way.

A.2.2 Qualitative results

Additional qualitative results. Following the format in the main text, we report extra qualitative results for all datasets in Fig. 11 (novel results on ImageNet and ShapeNet Cars), Fig. 15 (ShapeNet Chairs & CARLA, and comparison to Pix2NeRF [4]), and Fig. 19 (CUB and P3D Cars, including comparison to prior work).

Conversion to triangle mesh. Our adoption of an SDF representation allows us to easily extract a triangle mesh from a generated object (Fig. 16). We first quantize the SDF to a fixed-size grid, and then extract its 0-level set (*i.e.* zero-crossings) via marching cubes [37], obtaining a set of vertices and triangles. Finally, we sample colors from the radiance field by querying the network at the locations specified by the vertex positions.

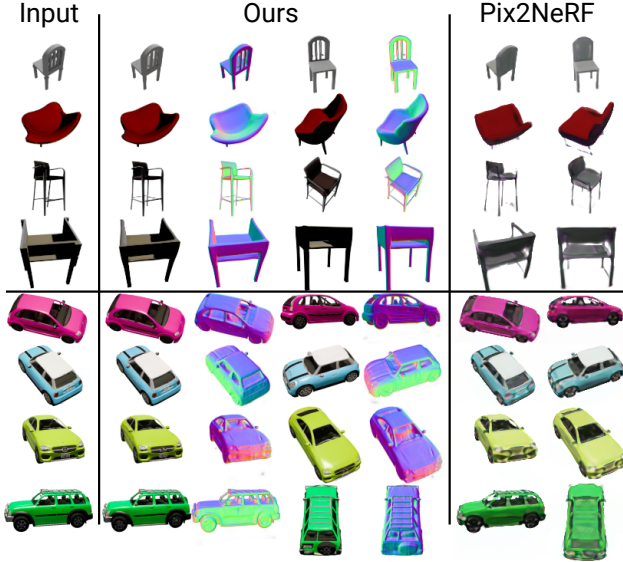


Figure 15. Additional qualitative results on synthetic datasets (test set of ShapeNet Chairs & CARLA) and side-by-side comparison to Pix2NeRF [4] on input and random views at 128×128 .

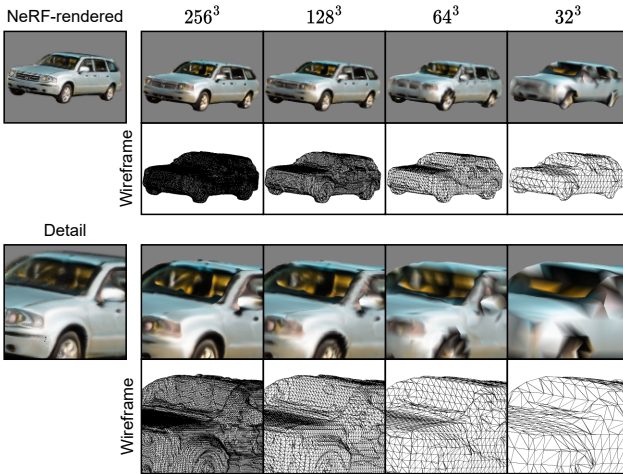


Figure 16. Extraction of a colored triangle mesh from an SDF. On the left, we show a car rendered using a neural renderer as well as zoomed viewpoint. On the right, we show the corresponding triangle mesh (including wireframe visualization) at different quantization steps.

Demo video. As part of the supplementary material, we include a video that shows additional examples of reconstructions. We break down each result into (i) prediction of the canonical map, (ii) initial pose estimation and bootstrapping of the latent code, (iii) refinement via hybrid inversion, and (iv) 360° animation of the reconstructed object and its surface. All samples are random (no selection is done) and failure cases are shown as well.

A.2.3 Limitations and failure cases

In this section, we highlight the most common failure modes of our method and attempt to categorize them in common patterns.

Shape artifacts. In some cases, we observe that reconstructed shapes present some artifacts, even though the appearance of the object looks correct when rendered. For instance, in animals, features such as the beaks of the birds are sometimes bifurcated (Fig. 17, top). We attribute this issue to a lack of density in some areas of the pose distribution of the dataset, *e.g.* most birds are observed from the side, but rarely from the front. On some small datasets such as zebras and elephants – which comprise only 1.7k and 1.4k images respectively – we also observe concavities (see Fig. 9 in the main text), and in the specific case of zebras, a failure to disentangle the stripes from the shape. We expect these entanglement issues to improve with larger datasets.

Pose estimation errors. More rarely, failures are caused by inaccurate pose estimation. If the initial estimated pose is too far from the true one, the optimizer can get stuck in a local minima and cause the reconstructed surface to become distorted (Fig. 17, bottom). We also notice that pose estimation is more challenging on examples with “extreme” postures, *e.g.* open wings in birds and head flexion in zebras, but also observe that many of these cases are handled correctly (Fig. 18). Most likely, the failure cases are due to these poses being underrepresented in the unconditional generator, and are not due to a limitation of the pose estimation framework itself.



Figure 17. Most common modes of failure on CUB Birds and P3D Cars. On CUB (first two rows), we sometimes observe a “split beak” phenomenon. On P3D (last two rows), in the rare instances the pose is estimated imprecisely, the optimizer can get stuck in a local minima and lead to a distorted reconstruction.

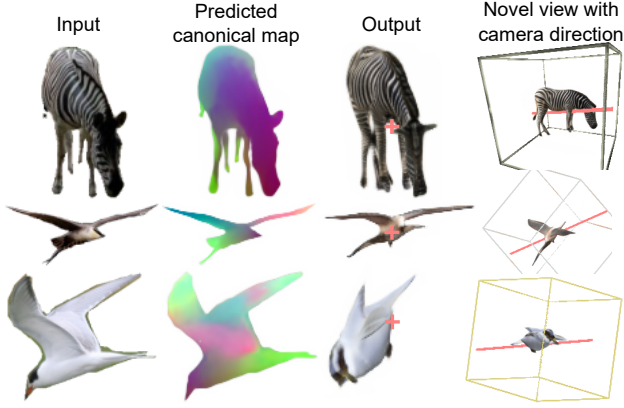


Figure 18. Our pose estimation approach can correctly handle “extreme” postures (top and middle), although some failure cases are still possible (bottom).

Incomplete inversion. For some hard examples, the encoder might return an initial solution which is far from the optimum, which in turn needs to be optimized for longer to correctly match the input image. Since we adopt a fixed schedule (*i.e.* the same number of optimization steps for all images), some images may only be partially inverted. As part of future work, this issue could be mitigated by using an adaptive optimization schedule that varies for each sample.

A.3. Negative results

Throughout the development of our method, we experimented with various techniques drawing inspiration from the literature on GANs and representation learning. To guide further research in this area, we provide a list of ideas we explored but did not work out as expected.

- We initially experimented with various NeRF representations, including MLP-based, voxel-based, and triplanar-based. We eventually settled with the triplanar representation of [5] since it was as expressive as the other ones but more efficient.
- Our initial attempts at solving the reconstruction task used an encoder-based approach with multiple discriminators. While this was appropriate for synthetic data, we quickly found out that it was not robust on real datasets with imprecise poses, which is also one of the main motivating factors for our approach. Based on the intuition that the issue might have been caused by the limited expressivity of the encoder, we tried to replace the encoder with an embedding layer that learned a different latent code for each instance (similar to [26, 47], but using a GAN framework with multiple discriminators), essentially decoupling the impact of the encoder architecture from that of the learning framework. In this setting, we

found that the issue was not resolved, which prompted us to explore other ideas.

- Before settling with our hybrid inversion framework, we also explored techniques from the representation learning literature, such as bidirectional GANs (BiGAN) [10]. In this setting, the encoder is not connected to the generator and the learning signal comes from a joint discriminator. Our expectation was that this setting would make the approach less reliant on precise poses, but we found that the reconstructions did not mirror the input images closely enough.

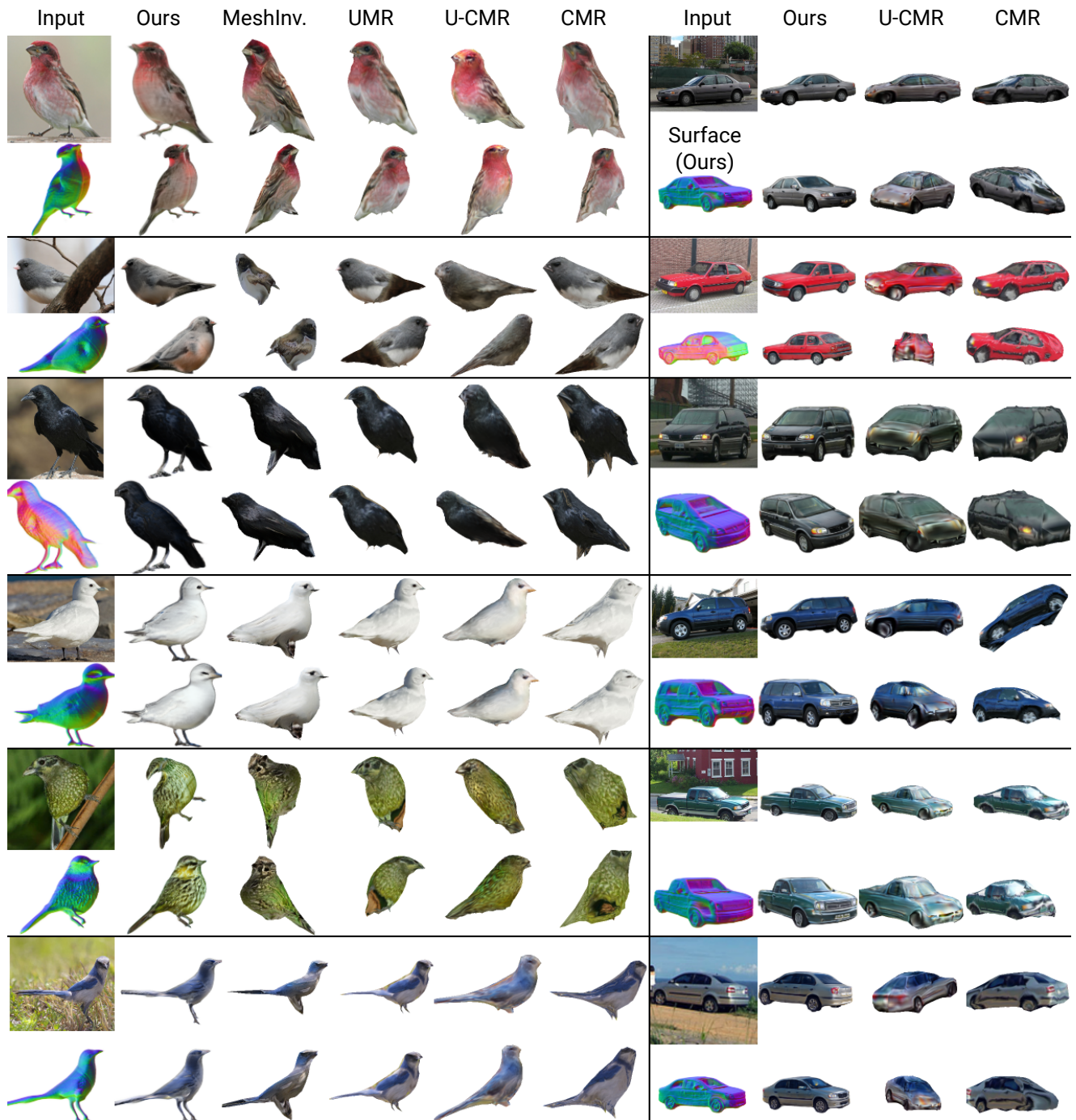


Figure 19. Additional qualitative results and side-by-side comparison on the test set of CUB (left) and Pascal3D+ Cars (right), at 128×128 . The first row of each sample is rendered from the input viewpoint, whereas the second row illustrates a random view.