

# RUST: Latent Neural Scene Representations from Unposed Imagery

Mehdi S. M. Sajjadi\* Aravindh Mahendran Thomas Kipf  
Etienne Pot Daniel Duckworth Mario Lučić Klaus Greff  
Google Research, Brain Team

## Abstract

Inferring the structure of 3D scenes from 2D observations is a fundamental challenge in computer vision. Recently popularized approaches based on neural scene representations have achieved tremendous impact and have been applied across a variety of applications. One of the major remaining challenges in this space is training a single model which can provide latent representations which effectively generalize beyond a single scene. Scene Representation Transformer (SRT) has shown promise in this direction, but scaling it to a larger set of diverse scenes is challenging and necessitates accurately posed ground truth data. To address this problem, we propose RUST (Really Unposed Scene representation Transformer), a pose-free approach to novel view synthesis trained on RGB images alone. Our main insight is that one can train a Pose Encoder that peeks at the target image and learns a latent pose embedding which is used by the decoder for view synthesis. We perform an empirical investigation into the learned latent pose structure and show that it allows meaningful test-time camera transformations and accurate explicit pose readouts. Perhaps surprisingly, RUST achieves similar quality as methods which have access to perfect camera pose, thereby unlocking the potential for large-scale training of amortized neural scene representations.

## 1. Introduction

Implicit neural representations have shown remarkable ability in capturing the 3D structure of complex real-world scenes while circumventing many of the downsides of mesh based, point cloud based, and voxel grid based representations [24]. Apart from visually pleasing novel view syn-

\*Correspondence: [rust@msajjadi.com](mailto:rust@msajjadi.com). Project page: [rust-paper.github.io](https://rust-paper.github.io). Contributions: MS: Conception, model design, implementation lead, analysis, infrastructure, experiments, writing, project lead. AM: Conception, model design, implementation, code reviews, experiments-MSN, analysis-MSN, writing. TK: RUST explicit pose estimation, model figures, scoping, advising, writing. EP: GNeRF experiments, Street View dataset, interactive visualization for embedding inspection. DD: COLMAP experiments, evaluation. ML: Team buy-in, scoping, writing. KG: Conception, early MSN analysis, analysis-SV, data-collection and analysis, dataset generation, visualizations, scoping, advising, writing.

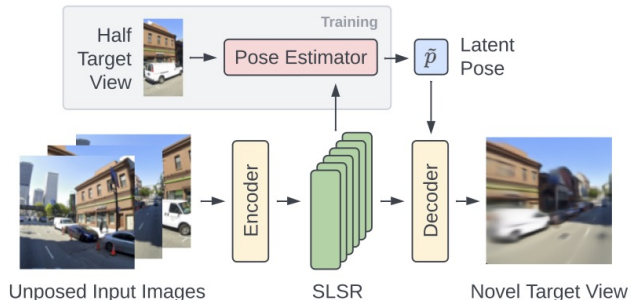


Figure 1. **Model overview** – RUST produces 3D-centric scene representations through novel view synthesis purely from RGB images without requiring any camera poses. For training, a novel *Pose Estimator* module glimpses at the target view and passes a low-dimensional latent pose feature to the decoder.

thesis [14, 16], such representations have shown potential for semantics [25], object decomposition [21], and physics simulation [2] which makes them promising candidates for applications in augmented reality and robotics. However, to be useful in such applications, they need to (1) provide meaningful representations when conditioned on a very limited number of views, (2) have low latency for real-time rendering, and (3) produce scene representations that facilitate generalization of knowledge to novel views and scenes.

The recently proposed Scene Representation Transformer (SRT) [22] exhibits most of these properties. It achieves state-of-the-art novel view synthesis in the regime when only a handful of posed input views are available, and produces representations that are well-suited for later segmentation at both semantic [22] and instance level [21]. A major challenge in scaling methods such as SRT, however, is the difficulty in obtaining accurately posed real world data which precludes the training of the models. We posit that it should be possible to train *truly* pose-free models from RGB images alone without requiring any ground truth pose information.

We propose RUST (*Really Unposed Scene representation Transformer*), a novel method for neural scene representation learning through novel view synthesis that does not require pose information: neither for training, nor for inference; neither for input views, nor for target views. While at first

glance it may seem unlikely that such a model could be trained, our key insight is that a sneak-peek at the target view at training can be used to infer an *implicit latent pose*, thereby allowing the rendering of the correct view. We find that our model not only learns meaningful, controllable latent pose spaces, but the quality of its novel views are even comparable to the quality of posed methods.

Our key contributions are as follows:

- We propose RUST, a novel method that learns latent 3D scene representations through novel view synthesis on very complex synthetic and real datasets without any pose information.
- Our model strongly outperforms prior methods in settings with noisy camera poses while matching their performance when accurate pose information is available to the base-lines.
- We provide an investigation into the structure of the learned latent pose spaces and demonstrate that meaningful camera transformations naturally emerge in a fully unsupervised fashion.
- Finally, we demonstrate that the representations learned by the model allow explicit pose readout and dense semantic segmentation.

## 2. Method

The model pipeline is shown in Fig. 1. A data point consists of an unordered set of  $N$  input views  $x = \{x_i \in \mathbb{R}^{H \times W \times 3}\}$  of a scene. Unlike SRT [22], they only consist of RGB images since RUST does not use explicit poses. Given these input views, the training objective is to predict a novel target view  $y \in \mathbb{R}^{H \times W \times 3}$  of the same scene.

To this end, the input views  $x$  are first encoded using a combination of a CNN and a transformer, resulting in the Set-Latent Scene Representation (SLSR)  $S$  which captures the contents of the scene. The target view  $y$  is then rendered by a transformer-based decoder that attends into the SLSR  $S$  to retrieve relevant information about the scene for novel view synthesis. In addition, the decoder must be conditioned on some form of a query that identifies the desired view. Existing methods, including SRT [22], often use the explicit relative camera pose  $p$  between one of the input views and the target view for this purpose. This imbues an explicit notion of 3D space into the model, which introduces a burdensome requirement for accurate camera poses, especially for training such models. RUST resolves this fundamental limitation by learning its own notion of implicit poses.

**Implicit poses.** Instead of querying the decoder with explicit poses, we allow the model to learn its own implicit space of camera poses through a learned *Pose Estimator* module. For training, the Pose Estimator sees parts of the target view  $y$  and the SLSR  $S$  and extracts a low-dimensional

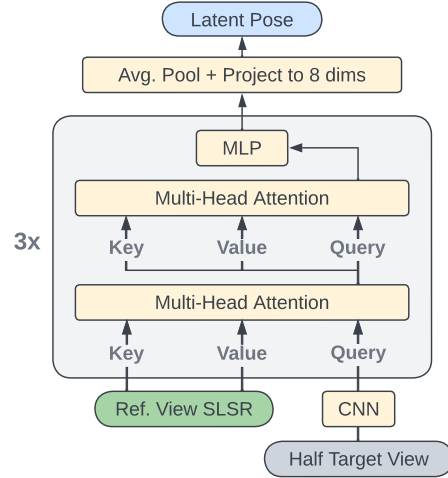


Figure 2. **Pose estimator model** – A randomly chosen half of the target image is encoded into a latent pose. The model mainly consists of a CNN encoder followed by alternating cross- and self-attention layers. The final output is projected to 8 dimensions to encourage an easily controllable latent representation.

*latent pose* feature  $\tilde{p}$ . The decoder transformer then uses  $\tilde{p}$  as a query to cross-attend into the SLSR to ultimately render the full novel view  $\tilde{y}$ . This form of self supervision allows the model to be trained with standard reconstruction losses without requiring any pose information. At test time, latent poses can be computed on the input views and subsequently modified for novel view synthesis, see Sec. 4.2.

### 2.1. Model components

**Input view encoder.** The encoder consists of a convolutional neural network (CNN) followed by a transformer. Each input image  $x_i$  is encoded independently by the shared CNN which consists of 3 downsampling blocks, each of which halves the image height and width. As a result, each spatial output feature corresponds to an  $8 \times 8$  patch in the original input image.

We add the same learned position embeddings to all spatial feature maps to mark their spatial 2D position in the images. We further add another learned embedding only to the features of the first input image  $x_1$  to allow the model to distinguish them from the others. This is relevant for the Pose Estimator module, as explained below. Finally, we flatten all spatial feature maps and combine them across input views into a single set of tokens. The encoder transformer then performs self-attention on this set of tokens, thereby exchanging information between the patch features. This results in the SLSR  $S$  which captures the scene content as a bag of tokens.

**Pose estimator.** The model architecture of the *Pose Estimator* is shown in Fig. 2. A randomly chosen half of the target

view  $y$  (*i.e.*, either the left or the right half of the image) is first embedded into a set of tokens using a CNN similar to the input view encoder. A transformer then alternates between cross-attending from the target view tokens into a specific subset  $\tilde{S} \subset S$  of the SLSR and self-attending between the target view tokens. The intuition behind cross-attending into  $\tilde{S}$  is that the latent pose  $\tilde{p}$  should be *relative* to the scene. We allow the Pose Estimator to only attend into SLSR tokens belonging to the (arbitrarily chosen) first input view after empirical findings that this leads to better-structured latent pose spaces. It is important to note that  $\tilde{S}$  contains information about *all* input views due to the self-attention in the preceding encoder transformer.

Finally, we apply global mean pooling on the transformer’s output and linearly project it down to an 8-dimensional latent pose  $\tilde{p}$ . It is important to note that we call  $\tilde{p}$  the estimated “*pose*” since it primarily serves the purpose of informing the decoder of the target camera pose. However, we do not enforce any explicit constraints on the latent, instead allowing the model to freely choose the structure of the latent poses. Nevertheless, we find that the model learns to model meaningful camera poses, see Sec. 4.2.

**Decoder transformer.** Similar to SRT, each pixel is decoded independently by a decoder transformer that cross-attends from a query into the SLSR  $S$ , thereby aggregating relevant information from the scene representation to establish the appearance of the novel view point. We initialize the query by concatenating the latent pose  $\tilde{p}$  with the spatial 2D position of the pixel in the target image and passing it through a small query MLP. The output of the decoder is the single RGB value of the target pixel.

## 2.2. Training procedure

For each data point during training, we use 5 input views and 3 novel target views. The training objective is to render the target views by minimizing the mean-squared error between the predicted output and the target view:  $\|\tilde{y} - y\|_2^2$ . The entire model is trained end-to-end using the Adam optimizer [9]. In practice, we found the model to perform better when gradients flowing to and through the Pose Estimator module are scaled down by 0.2 which is inspired by spatial transformer networks [7].

## 3. Related works

**Neural rendering.** The field of neural rendering is vast and the introduction of Neural Radiance Fields (NeRF) [16] has led to a surge of recent follow-up works. NeRF optimizes an MLP to map 3D positions of the scene to radiance and density values, which are used through a differentiable volumetric rendering equation to reconstruct the provided posed images of the scene. We refer the reader to surveys on neural rendering [24] and NeRF [4] for recent overviews.

**NeRF without pose.** Most methods based on NeRF assume the availability of perfect camera pose. There exist a number of works extending NeRF for pose estimation or to no-pose settings. INeRF [28] uses pre-trained NeRFs to estimate the camera pose of novel views of the same scene. However, posed imagery is required for each scene in order to obtain the original NeRF models. NeRF-- [27] jointly optimizes the NeRF MLP along with the camera poses for the images. BARF [13] adds a progressive position encoding scheme for improved gradients. Both methods only succeed for forward-facing datasets while failing for more complex camera distributions unless noisy initial poses are given. GNeRF [15] works for scenes with more complex camera distributions, however, the prior distribution must be known for sampling. VMRF [30] extends this to settings where the prior distribution is unknown.

All methods above require a comparably large number of images of the same scene, since NeRF tends to fail with few observations even with perfect pose [29]. While methods exist that optimize NeRFs from fewer observations [8, 18], they have limitations in terms of scene complexity [22] and to the best of our knowledge, no NeRF-based method has been demonstrated to work with few unposed images.

**Latent 3D scene representations.** In the right setup, NeRF produces high-quality novel views, though it does so without providing tangible scene representations that could be readily used for downstream tasks. The line of research focusing on latent 3D scene representations includes NeRF-VAE [12] which uses a variational autoencoder [10] to learn a generative model of NeRF’s for synthetic scenes, and GQN [3] which adds per-image latent representations to compute a global scene representation that is used by a recurrent latent variable model for novel view synthesis. A recent extension [19] of GQN performs pose estimation for novel target views by optimizing, at inference time, the posterior probability over poses as estimated by the generative model. All the methods above require ground truth poses for training and inference. GIRAFFE [17] requires no poses for training, but a prior distribution of camera views for the specific dataset. It uses a purely generative mechanism, without the ability to render novel views for given scenes.

**Posed and “unposed” SRT.** The current most scalable latent method for neural scene representations is the Scene Representation Transformer (SRT) [22]. It uses a transformer-based encoder-decoder architecture for novel view synthesis, thereby scaling to much more complex scenes than prior work. Its scene representation, the SLSR, has been shown to be useful for 3D-centric supervised and unsupervised semantic downstream tasks [21, 22].

While SRT requires posed imagery, the “unposed” variant UpSRT has been proposed as well in the original work [22]. Similar to RUST, it does not require posed *input* views. How-

Method	Pose	PSNR	Ablation	PSNR
SRT [22]	$p_x, p_y$	23.31	Right-half PE	23.88
SRT <sup>†</sup>	$p_x, p_y$	24.40	Stop grad.	23.16
SRT <sup>†</sup>	$\hat{p}_x, p_y$	23.81	No SLSR	20.83
UpSRT <sup>†</sup>	$\cancel{p}_x, p_y$	23.03	No self-attn	22.97
SRT <sup>†</sup>	$\hat{p}_x, \hat{p}_y$	18.65	3-dim. $\tilde{p}$	20.40
UpSRT <sup>†</sup>	$\cancel{p}_x, \hat{p}_y$	18.64	64-dim. $\tilde{p}$	23.40
RUST	$\cancel{p}_x, \cancel{p}_y$	23.49	768-dim. $\tilde{p}$	23.11

Table 1. **Quantitative results on MSN – Left:** Comparison with prior work in various settings: perfect ( $p_x, p_y$ ), noisy ( $\hat{p}_x, \hat{p}_y$ ) and lack of ( $\cancel{p}_x, \cancel{p}_y$ ) input and target poses. We report SRT both as proposed [22], and with our improved architecture (SRT<sup>†</sup>, UpSRT<sup>†</sup>). Despite requiring no poses, RUST matches the performance of SRT and UpSRT<sup>†</sup> while strongly outperforming all methods when target pose is noisy  $\hat{p}_y$ . **Right:** Model ablations, see Sec. 4.1.1.

ever, training an UpSRT model requires exact pose information for the *target* views to specify to the decoder which exact view of the scene to render. This restricts the application of UpSRT to settings where accurate pose information is available for the entire training dataset. We analyze further implications of this in Sec. 4.1.

**Using targets as inputs for training.** UViM [11] encodes targets such as panoptic segmentation maps into short latent codes during training to allow a single model architecture to perform well across several tasks. A bottleneck is enforced on the so-called *guiding codes* through discretization. Recent work on inferring protein conformations and poses [20] trains a VAE model in order to encode Cryo-EM images into latent distributions over poses and protein conformations. Unlike RUST, their method requires a base scene representation that is fixed a priori and uses an explicit differentiable rendered and explicit geometry in the architecture.

## 4. Experiments

We begin our experiments on the MultiShapeNet (MSN) dataset (version from [21]), a very challenging test bed for 3D-centric neural scene representation learning [22]. It consists of synthetically generated 3D scenes [6] that contain 16-32 ShapeNet objects [1] scattered in random orientations, and with photo-realistic backgrounds. Importantly, the set of ShapeNet objects is split for the training and test datasets, meaning that all objects encountered in the test scenes are not only in novel arrangements and orientations, but the model has never seen them in the training dataset. Camera positions are sampled from a half-sphere with varying distances to the scene. In all experiments, we follow prior work [22] by using 5 input views, and all quantitative metrics are computed on the right halves of the target views, since the left halves are used for pose estimation in RUST.

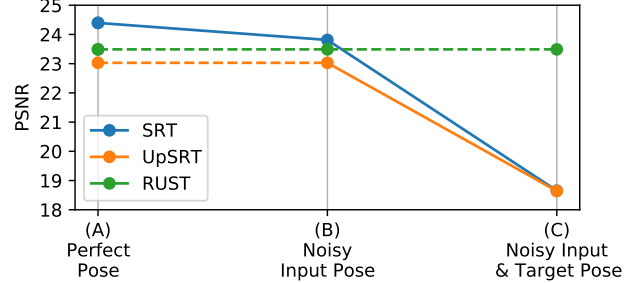


Figure 3. **Robustness to camera noise** – Sajjadi et al. [22] evaluate SRT and UpSRT on (A) perfect pose, and (B) noisy *input* pose. In the more realistic setting (C) where *input & target* pose is noisy, both methods fail as they rely on accurate target camera pose for training. RUST needs no pose, so its performance is constant.

### 4.1. Novel view synthesis

The core contribution of RUST is its ability to model 3D scenes without posed imagery. While camera poses in synthetic environments such as MSN are perfectly accurate, practical applications must often rely on noisy sensor data or inaccurate estimated poses. We therefore begin our investigations by measuring the quality of synthesized novel views in new scenes under various assumptions on the accuracy and availability of camera pose information during training. To simulate the real world, we follow prior work [13, 22] and perturb the training camera poses with additive noise for a relatively mild amount of  $\sigma = 0.1$ . We visualize the effect of this noise in Fig. 14 (appendix) to provide context for its scale. This leads to three possible settings for both input poses and target poses: perfect ( $p_x, p_y$ ), noisy ( $\hat{p}_x, \hat{p}_y$ ) and lack of ( $\cancel{p}_x, \cancel{p}_y$ ) input and target cameras poses, respectively. Note that for the posed baselines, we always use perfect target poses for evaluation, *i.e.* the target poses are only perturbed during training.

Tab. 1 (left) shows our quantitative evaluation on the MSN dataset. We first compare SRT as proposed [22] with our improved architecture SRT<sup>†</sup>, both using perfect pose. We observe that our modifications to the architecture lead to significant improvements in PSNR (+1.09 db). Continuing with the improved architecture, perturbing the input views leads to a loss in 0.59 db (SRT<sup>†</sup>,  $\hat{p}_x, p_y$ ) while the lack thereof leads to a drop of 1.37 db (UpSRT<sup>†</sup>,  $\cancel{p}_x, p_y$ ).

Crucially, the most realistic setting where *all* poses are equally noisy leads to a dramatic decline in performance for prior work (SRT<sup>†</sup> and UpSRT<sup>†</sup>,  $\hat{p}_y$ ). RUST meanwhile outperforms the baselines by +4.84 db in this setting without any poses, while matching the performance of SRT [22] with perfect pose. We highlight the difference between only perturbing input poses (following prior work [22]) or all camera poses (more realistic setting) in Fig. 3. It is evident from the plot that RUST is the only model that is applicable when camera poses are noisy or even unavailable.



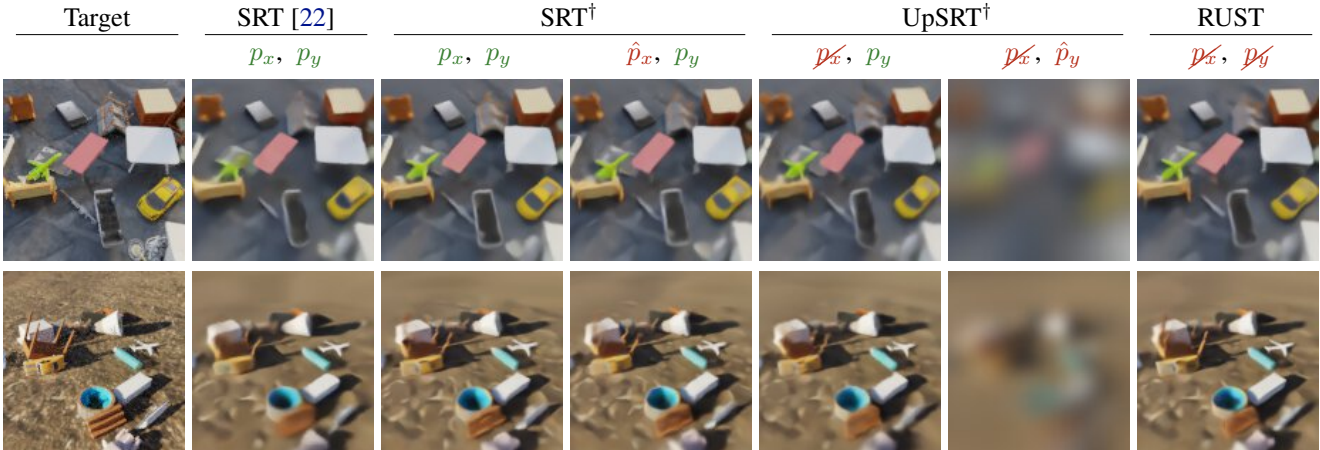


Figure 4. **Qualitative results on MSN** – We compare RUST to SRT [22] as proposed, and with the respective models using our improved architecture (SRT<sup>†</sup>, UpSRT<sup>†</sup>). Different models support perfect ( $p_x, p_y$ ), noisy ( $\hat{p}_x, \hat{p}_y$ ), or a lack of ( ~~$p_x, p_y$~~ ,  ~~$\hat{p}_x, \hat{p}_y$~~ ) input and target poses, respectively. SRT<sup>†</sup> shows a mild drop in quality for perturbed input poses and achieves a similar reconstruction quality as UpSRT<sup>†</sup> with perfect target poses. In the realistic setting where the target poses are also perturbed ( $\hat{p}_y$ ), both methods fail entirely to render sharp images (only UpSRT<sup>†</sup> shown here, see appendix for SRT<sup>†</sup>). RUST, without any camera poses, produces similar-quality renders as the baselines with perfect pose information, and even visibly outperforms the original SRT model [22] with perfect pose information.

Fig. 4 shows qualitative results for a selection of these models. Notably, UpSRT<sup>†</sup> produces extremely blurry views of the scene when the target poses are mildly inaccurate, while RUST produces similar-looking results as UpSRT<sup>†</sup> with perfect target poses. Further qualitative results including videos are provided in the supplementary material.

For the remainder of the experimental section, we only compare RUST to the stronger SRT<sup>†</sup> and UpSRT<sup>†</sup> baselines and refer to these as SRT and UpSRT for ease of notation.

#### 4.1.1 Ablations

We investigate the effect of a selection of our design choices for RUST. The results are summarized in Tab. 1 (right). Metrics are computed only on the right half of each target image, since the left half is used for pose estimation by RUST. While we see a lot of evidence that the model is encoding a form of camera pose in the latent space (see Sec. 4.2), it is still possible that the model uses parts of the latent pose feature to encode *content* information about the target view. We therefore now evaluate the exact same RUST model (with identical weights), but now use the right halves of the target images in the Pose Estimator. We observe that this scheme only outperforms the left-half encoding scheme by +0.39 db, thereby showing that the latent pose primarily serves as a proxy for the camera position rather than directly informing the model about the target view content.

In our default model, we allow gradients to flow from the latent pose back into the encoder, though they are effectively scaled down by a factor of 0.2 (see Sec. 2.2). Cutting the gradients fully implies that the the Pose Estimator cannot

directly affect the encoder anymore through the SLSR during training. This variant leads to a drop of 0.33 db in PSNR.

As described in Sec. 2.1, the Pose Estimator cross-attends into parts of the SLSR to allow it to anchor the target poses relative to the input views. Removing this cross-attention module makes it much harder for the model to estimate the target pose and pass it to the decoder. The performance therefore drops by a significant 2.66 db in PSNR. Removing the self-attention between the target image tokens from the Pose Estimator module has a less dramatic effect of 0.52 db.

Finally, we investigate different choices for the size of the latent pose feature  $\tilde{p}$ . We found empirically that smaller sizes such as 3 dimensions would lead to significantly worse results (-3.09 db in PSNR). This is likely only a result of worse training dynamics, as the pose space in the MSN dataset can in theory be fully described with 3 degrees of freedom. Using significantly larger latent pose sizes (64, 768) on the other hand leads to a less controllable latent pose space, while rendering quality remains comparable.

## 4.2. Latent pose investigations

In order to analyze the structure of the learned latent pose space, we use principal component analysis (PCA). Specifically, we collect latent poses  $\tilde{p}$  for three target views per scene from 4k test scenes of the MSN dataset and inspect the major PCA components of the resulting 8-dimensional pose distribution. We visualize the first three PCA components in Fig. 5 (left) and color-code the points such that hue encodes rotation relative to the first input view’s camera and intensity encodes camera height. We find that the points

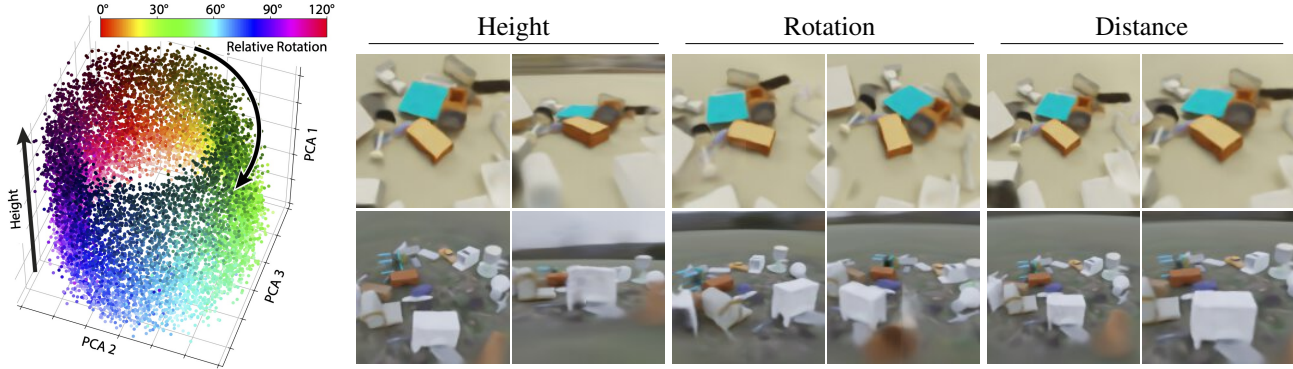


Figure 5. **Latent pose investigations on MSN** – **Left**: The first three PCA components of the latent poses across many scenes roughly constitute a cylinder. Points are coloured such that intensity represents camera height and hue represents their rotation around the scene’s z-axis relative to the first input camera, modulo  $120^\circ$ . That means a single rotation around the scene maps to three rotations around this circle, with the  $0^\circ$  position mapping to the first input camera. We found that components 5 and 6 roughly capture one full 360 degree rotation, resolving the ambiguity. We elaborate on this in Appendix A.2. **Right**: Intuitive camera movement induced by traversals in PCA space.

form a cylinder whose axis is aligned with the first principal component. This axis correlates strongly with camera height with a Pearson’s  $r = 0.94$ . The qualitative effect of moving along this dimension is shown in Fig. 5 (*Height*). Similarly, rotating around the cylinder causes the camera to rotate around the scene in 3D as shown in Fig. 5 (*Rotation*), although only a third of the rotation is completed by a full traversal along the cylinder. Note that rotations in the latent space are encoded relative to the position of the first input camera, likely because the Pose Estimator only consumes the SLSR tokens  $\tilde{S}$  corresponding to the first input view.

The cylinder-shaped latent pose distribution is sensible, considering that the camera poses in the MSN dataset are distributed along a dome and always point at the scene origin. At the pole of this dome is hence a discontinuity as the camera flips when crossing that point. The cylinder is therefore akin to this dome that is opened up at its pole to eliminate the discontinuity. The remaining PCA components are visualized in Appendix A.2. The fourth component captures the distance of the camera from the scene center ( $r = 0.94$ ). We show traversals along this axis in Fig. 5 (*Distance*). We further find that components 5 and 6, capture a 360 degree *absolute* rotation in scene coordinates. This surprising finding is further investigated in Appendix A.2.

It is notable that the model has learned this meaningful structure in the latent pose space without any form of camera pose supervision. Furthermore, this shows that test-time camera control is feasible directly in the latent space.

#### 4.2.1 Explicit pose estimation

To validate to what extent information about the camera pose is retained in the small latent pose  $\tilde{p}$  of RUST, we perform an explicit pose estimation (EPE) experiment. As there is no canonical frame of reference, we choose to predict *relative*

Method	# Views	MSE	$R^2$ (%)	Success (%)
RUST EPE	7	0.08	99.9	[100]
COLMAP	10	0.00	100.0	4.2
COLMAP	80	0.07	99.7	29.5
COLMAP	160	0.38	99.1	58.9
GNeRF	12	29.39	46.7	[100]
GNeRF	150	9.24	83.1	[100]
GNeRF-FG	150	4.05	92.7	[100]

Table 2. **Explicit pose estimation on MSN** – RUST EPE recovers relative camera poses nearly perfectly from the SLSR (5 input views) and the pair of latent target poses. COLMAP [23] requires a much larger number of images, and still has a significantly lower success rate for registration. Similarly, GNeRF [15] requires many views of the scene, and fails to estimate accurate poses even when the background pixels are removed from the data (GNeRF-FG).

poses between two separate target views. Since all cameras on the MSN dataset are pointed at the center of the scene, we only estimate the relative camera *position*.

Our EPE module is trained on top of a (frozen) pre-trained RUST model. It takes the SLSR  $S$  and the latent pose features  $\tilde{p}_1$  and  $\tilde{p}_2$  for the two target views and follows the design of the RUST decoder: the concatenation of the two latent poses acts as the query for a transformer which cross-attends into the SLSR  $S$ . We found that this cross-attention step is necessary as the latent pose vectors  $\tilde{p}$  partially carry pose information that is relative to the SLSR (*e.g.*, rotation, see Sec. 4.2). Finally, the result is passed through an MLP which is tasked to predict the explicit relative pose between the target views. We train the model using the L2 loss.

Quantitative results in terms of mean-squared error (MSE)

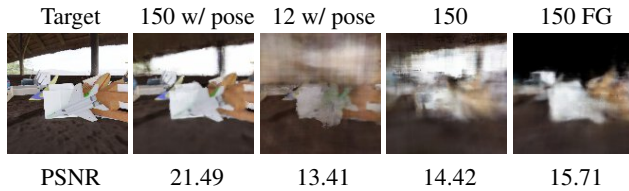


Figure 6. **Qualitative results for GNeRF [15]** – When trained with many posed views (150), results have high fidelity. Reducing the number of images to 12 leads to worse reconstruction and strong artifacts. When no pose is available, the model produces worse quality. Removing the backgrounds from the training dataset helps improve reconstruction accuracy. PSNR computed on 50 views.

and  $R^2$  scores over 95 MSN test scenes are shown in Tab. 2. EPE recovers relative camera positions nearly perfectly on novel test scenes ( $R^2 = 99.9\%$ ) using only the SLSR (derived from 5 input views) and the pair of latent pose vectors.

**Photogrammetry.** To demonstrate the difficulty of the task, we apply COLMAP [23] to 95 newly generated MSN test scenes with up to 160 views (otherwise using the same parameters as the MSN test set). COLMAP uses correspondences between detected keypoints to estimate camera poses. While RUST EPE successfully recovers camera positions from just 7 total views, COLMAP struggles when only given access to a small number of views (*e.g.*,  $< 10$ ) and needs up to 160 views to successfully register most views and estimate their camera poses. As COLMAP may pose only a subset of the provided images, we report metrics for a predetermined, arbitrarily chosen camera pair per scene. If COLMAP fails to pose either of these cameras, we consider pose estimation to have failed and exclude this camera pair from evaluation. We find an optimal rigid transformation that maps estimated camera positions to ground-truth positions before evaluating MSE and  $R^2$  to be able to compare with RUST EPE.

**NeRF-based methods.** We evaluate GNeRF [15] as a strong representative for unsupervised NeRF-based pose estimation approaches on a single MSN scene using the implementation provided by the authors. GNeRF assumes knowledge of the camera intrinsics and that all cameras are pointed at the origin, *i.e.* only the 3-dimensional camera positions are estimated. Further, a prior for the camera pose initializations is given to the model. Using only 12 views, GNeRF fails to capture the scene. When using 150 views to train GNeRF, it still performs significantly worse than RUST EPE on explicit pose estimation, demonstrating the benefit of learning a generalizable pose representation *across* scenes, see Tab. 2. We show qualitative and quantitative results for novel view synthesis in Fig. 6. Training on perfect poses significantly improves NVS performance for GNeRF, showing that even with 150 views, GNeRF fails to accurately register the camera poses. As GNeRF expects the scene to be



Figure 7. **Qualitative results on SV** – Comparison of RUST with prior work using accurate camera pose. RUST outperforms our improved UpSRT variant, while producing similar quality as the fully posed improved SRT model. We further train a dense semantic segmentation decoder on top of the frozen RUST scene representation, showing that it retains semantic information about the scene.

contained within a given bounding box, we also try training it only on the foreground (FG) objects in the scene, dropping all background pixels during optimization. We empirically find that this only leads to slightly improved pose accuracy.

### 4.3. Application to real-world dataset

To answer the question whether RUST can learn to represent the 3D structure of complex real-world scenes, we apply it to the Street View (SV) dataset. We have received access to SV through private communication with the authors [5]. It contains 5M dynamic street scenes of San Francisco with moving objects, changes in exposure and white balance, and highly challenging camera positions, often with minimal visual overlap. Following Sajjadi et al. [22], we train and test the model using 5 randomly selected input views for each scene.

**Novel view synthesis.** We first evaluate the novel view synthesis performance of RUST, and again find that its performance of 22.50 db in PSNR is comparable to that of SRT (22.72) and that it falls in-between that of our improved SRT (23.63) and UpSRT (21.25). Qualitatively, we find that novel views generated by RUST correctly capture the 3D structure of the scenes and, remarkably, the model seems to be more robust than UpSRT, see Fig. 7. This happens especially often when the reference view is far away from the target view.

**Latent pose space.** In SV scenes, the camera positions mainly vary along the single forward-dimension of the street while horizontal rotations mainly move in discrete steps of  $60^\circ$  due to the spatial configuration of the six fixed cameras. This distribution of camera poses is very different from MSN, and we should thus expect the learned pose space to differ significantly as well. Similar to before, we compute a PCA decomposition on target views of 4k test-scenes and visualize the first three components in Fig. 8. Surprisingly,



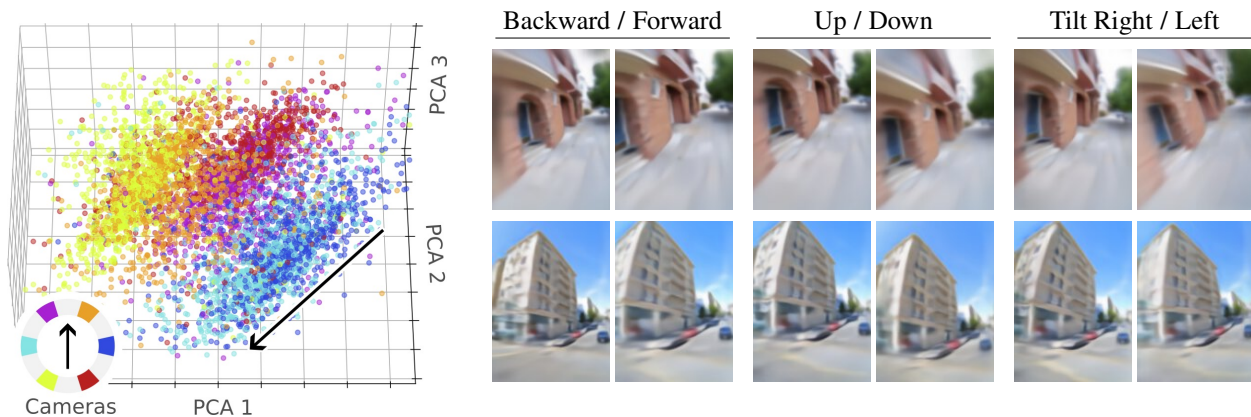


Figure 8. **Latent pose investigations on SV** – **Left:** Scatterplot showing the first three PCA components of the latent pose  $\tilde{p}$  distribution for 12k target views from 4k scenes. The color of each point indicates by which of the 6 cameras the image was taken, as shown on the color wheel. The first principal axis of these ellipsoids, marked by the long arrow, corresponds to movement along the street. **Right:** We found that specific linear directions in the latent space correspond to meaningful camera motions. Videos are provided in the supplementary material.

we find only three (rather than six) ellipsoid clusters that each correspond to an opposing pair of cameras. This suggests that the model has learned to represent rotation in an absolute (scene-centric) coordinate frame with the direction of the street as a reference, and that opposing cameras are less well-distinguished due to the streets’ approximate mirror symmetries, see Fig. 10 (appendix) for more details. We further find that movement along the street corresponds well to the long axis of these ellipsoids, which depending on the camera can correspond to either a forward/backward or a sideways motion in image space.

**Traversals.** In Fig. 8, we show linear latent pose traversals that correspond to movement along the street, pitch and tilt of the camera. We find that these dimensions are mapped into the pose space predictably, contiguously and smoothly. The correct parallax effect observed in the forward movement in particular demonstrates that the model has correctly estimated the depth of the scene. These results indicate that RUST can learn to capture the 3D structure of complex real-world scenes from a handful of images, without access to any pose information. The learned latent pose space covers the training distribution of camera poses and enables traversals and novel view synthesis within distribution.

## 5. Limitations

RUST shares most strengths and limitations with other latent methods, especially with the SRT model [22] that it is based on. Resolution and quality of synthesized views do not reach the performance of NeRF [16], though it is important to note that NeRF does not yield latent scene representations, and it generally requires a large number of similar views of the scene. As has been demonstrated [22], even special-

ized NeRF variants for sparse input view settings [8] fail to produce meaningful results on the challenging SV dataset, despite having access to accurate pose information.

As shown in Secs. 4.2 and 4.3, RUST learns to effectively model the camera pose distribution of the datasets it is trained on. While this is desirable in many applications, it can limit generalization of the model to new poses that lie outside of the original distribution. For example, we may not find latent poses for which RUST would render views that do not point at the center of the scene in MSN, or represent movement orthogonal to the street in SV. It is notable that all learned methods suffer from such limitations to varying degrees, including light field parametrizations [22] and even volumetric models, especially when given few input images [18]. Nonetheless, more explicit methods tend to generalize better to out-of-distribution poses due to hard-coded assumptions such as explicit camera parametrizations or the volumetric rendering equation. Finally, we empirically found RUST to show higher variance in terms of PSNR compared to SRT. Over independent training runs with 3 random seeds, we observed a standard error of 0.19.

## 6. Conclusion

We propose RUST, a novel method for latent neural scene representation learning that can be trained without any explicit pose information. RUST matches the quality of prior methods that require pose information while scaling to very complex real-world scenes. We further demonstrate that the model learns meaningful latent pose spaces that afford smooth 3D motion within the distribution of camera poses in the training data. We believe that this method is a major milestone towards applying implicit neural scene representations to large uncurated datasets.



## References

- [1] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012*, 2015. 4
- [2] Danny Driess, Zhiao Huang, Yunzhu Li, Russ Tedrake, and Marc Toussaint. Learning Multi-Object dynamics with compositional neural radiance fields. *CoRL*, 2022. 1
- [3] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 2018. 3
- [4] Kyle Gao, Yina Gao, Hongjie He, Denning Lu, Linlin Xu, and Jonathan Li. NeRF: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379*, 2022. 3
- [5] Google. Street view, 2007. URL [www.google.com/streetview/](http://www.google.com/streetview/). 7, 18
- [6] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanaprasam, Florian Golemo, Charles Herrmann, et al. Kubric: A Scalable Dataset Generator. In *CVPR*, 2022. 4
- [7] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NeurIPS*, 2015. 3
- [8] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting NeRF on a Diet: Semantically Consistent Few-Shot View Synthesis. In *ICCV*, 2021. 3, 8
- [9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 3
- [10] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014. 3
- [11] Alexander Kolesnikov, André Susano Pinto, Lucas Beyer, Xiaohua Zhai, Jeremiah Harmsen, and Neil Houlsby. UViM: A Unified Modeling Approach for Vision with Learned Guiding Codes. In *NeurIPS*, 2022. 4
- [12] Adam Kosior, Heiko Strathmann, Daniel Zoran, Pol Moreno, Rosalia Schneider, Sona Mokrá, and Danilo Rezende. NeRF-VAE: A Geometry Aware 3D Scene Generative Model. In *ICML*, 2021. 3
- [13] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. BARF: Bundle-Adjusting Neural Radiance Fields. In *ICCV*, 2021. 3, 4
- [14] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021. 1
- [15] Quan Meng. GNeRF: GAN-Based Neural Radiance Field Without Posed Camera. In *ICCV*, 2021. 3, 6, 7, 12, 16
- [16] Ben Mildenhall, Pratul Srinivasan, Matthew Tancik, Jonathan Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*, 2020. 1, 3, 8
- [17] Michael Niemeyer and Andreas Geiger. GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields. In *CVPR*, 2021. 3
- [18] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Reg-NeRF: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, 2022. 3, 8
- [19] Dan Rosenbaum, Frederic Besse, Fabio Viola, Danilo Jimenez Rezende, and S M Ali Eslami. Learning models for visual 3d localization with implicit mapping. *NeurIPS workshop on Bayesian Deep Learning*, 2018. 3
- [20] Dan Rosenbaum, Marta Garnelo, Michal Zielinski, Charlie Beattie, Ellen Clancy, Andrea Huber, Pushmeet Kohli, Andrew W. Senior, John Jumper, Carl Doersch, et al. Inferring a continuous distribution of atom coordinates from cryo-EM images using VAEs. *NeurIPS workshop on Machine Learning in Structural Biology*, 2021. 4
- [21] Mehdi S. M. Sajjadi, Daniel Duckworth, Aravindh Mahendran, Sjoerd van Steenkiste, Filip Pavetic, Mario Lucic, Leonidas J. Guibas, Klaus Greff, and Thomas Kipf. Object Scene Representation Transformer. In *NeurIPS*, 2022. 1, 3, 4, 11
- [22] Mehdi S. M. Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lucic, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene Representation Transformer: Geometry-Free Novel View Synthesis Through Set-Latent Scene Representations. In *CVPR*, 2022. 1, 2, 3, 4, 5, 7, 8, 11, 13, 17, 18
- [23] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *CVPR*, 2016. 6, 7, 12
- [24] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. *Computer Graphics Forum*, 41(2), 2022. 1, 3
- [25] Suhani Vora, Noha Radwan, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi SM Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. NeSF: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *TMLR*, 2022. 1
- [26] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. In *IEEE TIP*, 2004. 13

- [27] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF--: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021. 3
- [28] Lin Yen-Chen, Pete Florence, Jonathan Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. iNeRF: Inverting Neural Radiance Fields for Pose Estimation. In *IROS*, 2021. 3
- [29] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural Radiance Fields from One or Few Images. In *CVPR*, 2021. 3
- [30] Jiahui Zhang, Fangneng Zhan, Rongliang Wu, Yingchen Yu, Wenqing Zhang, Bai Song, Xiaoqin Zhang, and Shijian Lu. VMRF: View matching neural radiance fields. In *ACM MM*, 2022. 3
- [31] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 13

## A. Appendix

**Acknowledgements.** We thank Urs Bergmann for helpful feedback on the manuscript and Henning Meyer and Thomas Unterthiner for fruitful discussions.

### A.1. Model details [Sec. 2]

Unless stated otherwise, model architecture and hyper parameters follow the original SRT [22]. Our improved architecture makes the following changes that are partially inspired from prior work [21]. In the encoder CNN, we drop the last block, leading to smaller  $8 \times 8$  patches compared to the  $16 \times 16$  patches as in SRT [22]. This reduces the parameter count of the CNN and makes it faster, while the subsequent encoder transformer needs to process  $4 \times$  more tokens. However, the number of parameters in the encoder transformer is not affected by this change. In the encoder transformer, we use 5 layers instead of 10. Instead of using post-normalization, we use pre-normalization. We further use full self-attention instead of cross-attending into the output of the CNN. The initial queries in the decoder pass through a linear layer that maps them to 256 dimensions.

All models (SRT [22], SRT<sup>†</sup>, UpSRT<sup>†</sup>, RUST) were always trained for 3 M steps with batch size 256 on MSN and 128 on SV. Similar to SRT [22], we note that RUST is not fully converged yet at this point, though improvements are comparably small when trained further.

**Pose Estimator.** The CNN in the Pose Estimator follows very similar structure as the CNN in the encoder, with two exceptions: we use 4 CNN blocks as in SRT [22] (*i.e.*,  $16 \times 16$  patches), and we do not add image id embeddings, as each target view is processed and rendered fully independently. The transformer follows the same structure as the encoder transformer: we use 768 features for the attention process and 1536 features in the MLPs and pre-normalization.

**Appearance encoder.** For all models on SV, we additionally include the appearance encoder from the original SRT [22]. Empirically, we have seen that leaving out the appearance encoder in RUST leads to the Position Encoder capturing variations in exposure and while balance in  $\tilde{p}$ , making the appearance encoder obsolete – but for better comparability with the baselines, we opted to include it for our model as well. This design choice also has the positive side effect that the latent pose space is disentangled from variations in appearance.

### A.2. Latent pose investigations on MSN [Sec. 4.2]

In Fig. 12 we visualize the latent pose distribution along PCA components 4, 5, and 6, and demonstrate full  $360^\circ$  camera rotations around the scene by interpolating poses. The scatter plot of poses on the left side of Fig. 12 shows camera



Figure 9. **Example backgrounds from HDRi Haven** – Note that the main light source is always slightly to the right of the center.

poses colored by their absolute ground-truth rotation. The fact that the learned poses map so well onto the ground-truth coordinate frame of the generated scenes is surprising, since the model at no point has access to those coordinates, and the scene layout is mostly rotation symmetric. It turns out that the reason for this alignment has to do with the  $360^\circ$  HDR images from [hdri-haven.com](http://hdri-haven.com) used as backgrounds and for global illumination in MSN. These images are aligned such that the strongest light source (often the sun, but some-



# Views	MSE	$R^2$ (%)	Success (%)
10	0.00	100.0	4.2
20	0.01	99.8	5.3
40	0.01	99.9	16.8
80	0.07	99.7	29.5
120	0.56	97.8	43.2
160	0.38	99.1	58.9
200	0.28	99.3	66.3

Table 3. **Explicit Pose Estimation with COLMAP** – COLMAP requires between 120 and 160 images per scene to successfully pose the designated image pair more than half of the time. The perfect  $R^2$  score in the first row is an artifact of the low success rate and our alignment procedure for evaluation.

times, windows or lamps as well) is always in roughly the same direction, see Fig. 9. The background images are not randomly rotated between scenes, so the main direction of light correlates strongly with the ground-truth coordinate frame. And, as Fig. 12 shows, it thus also correlates strongly with the coordinate frame learned by RUST.

### A.3. Explicit Pose Estimation [Sec. 4.2.1]

The explicit pose estimation (EPE) module for RUST follows the architecture of the RUST decoder: we take the RUST latent pose for a target view and a reference view of a scene, concatenate both vectors along the channel dimension, and project the result to a 768-dim vector using a learnable Dense layer. The result is used as a query for a transformer that cross-attends into the SLSR with 2 layers, 12 heads, query/key/value projection dimension of 768, and an MLP hidden dimension of 1536 for the feed-forward layer, using pre-normalization and trained without dropout.

The result is finally used to predict the 3-dim difference between the target and reference camera origins using an MLP with a single 768-dim hidden layer. We train the EPE module on top of a frozen, pre-trained RUST model using a mean-squared error loss between prediction and ground-truth camera origin difference (target origin – reference origin). The pre-trained RUST model was trained on MSN for 3M steps (i.e. the same model that is used in other experiments). We train RUST EPE for 1M steps on the MSN training set using the Adam optimizer with otherwise the same hyperparameters as RUST. We evaluate RUST EPE on the first 95 (unseen) test set scenes. Estimated camera views (relative to arbitrarily chosen reference views) are shown in Fig. 15.

**COLMAP.** As a baseline, we compare RUST EPE to COLMAP [23], a general purpose Structure-from-Motion (SfM) pipeline for estimating camera parameters. Unlike RUST, COLMAP requires high visual overlap between im-

age pairs to identify “feature matches” – pixel locations corresponding to the same point in 3D space. In preliminary experiments, we found that COLMAP is unable to reconstruct scenes with fewer than 10 images. We thus generate an evaluation set of 95 scenes, each with 200 frames, from the same distribution as the MSN dataset.

We evaluate COLMAP as follows. We begin by running COLMAP’s SfM pipeline using the first 10 / ... / 200 images of each scene, sorted lexicographically. As the coordinate system of the estimated camera origins does not align with that of the ground-truth cameras, we find the optimal rigid transformation (rotation + translation + scale) that minimizes the squared error between camera origin pairs (predicted and ground-truth). To evaluate the relative pose, we select the camera origins of two fixed, arbitrary images (`camera_1` and `camera_2` in our experiments). Estimated camera views after alignment are shown in Fig. 16.

If COLMAP failed to identify the pose of either of these images, we label scene reconstruction as unsuccessful. The remainder of evaluation follows the protocol applied for RUST EPE.

**GNeRF.** We use the official GNeRF implementation from the authors [15]. We train GNeRF on 3 different scenes of 200 images each, generated from the MSN dataset. We select 150 or 12 views views for training and 50 views for evaluation. GNeRF is trained on each scene independently.

Similar to our evaluation protocol for COLMAP, we transform the predicted camera locations into the coordinate system of the ground-truth camera origins via a rigid transformation (rotation + translation) that minimizes the squared error between camera pairs. Estimated camera poses after alignment are shown in Fig. 17.

To measure MSE and  $R^2$ , we consider the relative position difference between all camera pairs of the 50 evaluation views, resulting in a total of 2450 pairs for metric computation. We show GNeRF results on further scenes in Fig. 18.

### A.4. Quantitative and qualitative results [Secs. 4.1 and 4.3]

Table 5 summarizes the quantitative results on the Street View dataset from Sec. 4.3. Figures 19 and 20 show more qualitative results on the MSN and SV datasets, respectively.

Method	Pose	PSNR $\uparrow$	SSIM $\downarrow$	LPIPS $\downarrow$ (AlexNet)	LPIPS $\downarrow$ (VGG)
SRT [22]	$p_x, p_y$	23.31	0.690	0.290	0.364
SRT $^\dagger$	$p_x, p_y$	24.40	0.740	0.237	0.314
SRT $^\dagger$	$\hat{p}_x, p_y$	23.81	0.718	0.265	0.332
UpSRT $^\dagger$	$\cancel{p}_x, p_y$	23.03	0.683	0.300	0.362
SRT $^\dagger$	$\hat{p}_x, \hat{p}_y$	18.65	0.469	0.741	0.612
UpSRT $^\dagger$	$\cancel{p}_x, \hat{p}_y$	18.64	0.469	0.741	0.612
RUST	$\cancel{p}_x, \cancel{p}_y$	23.49	0.703	0.287	0.351

Ablation	PSNR $\uparrow$	SSIM $\downarrow$	LPIPS $\downarrow$ (AlexNet)	LPIPS $\downarrow$ (VGG)
Right-half PE	23.88	0.711	0.277	0.346
Stop grad.	23.16	0.690	0.306	0.362
No SLSR	20.83	0.581	0.476	0.475
No self-attn	22.97	0.678	0.320	0.377
3-dim. $\tilde{p}$	20.40	0.563	0.506	0.492
64-dim. $\tilde{p}$	23.40	0.699	0.289	0.355
768-dim. $\tilde{p}$	23.11	0.684	0.307	0.371

Table 4. **Full quantitative results on MSN** – This table is identical to Tab. 1, but additionally reports SSIM [26] and LPIPS [31].

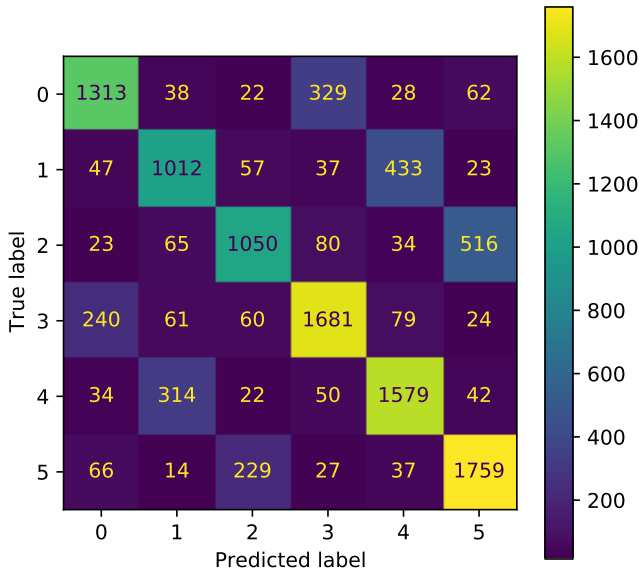


Figure 10. **Street View camera ids in latent space** – Confusion matrix between the true camera ids (6 cameras on the rig) and the predicted labels of a linear readout from the RUST latent pose features  $\tilde{p}$ . We can see that the identity is largely predicted correctly. Remarkably, the latent pose even distinguishes between opposite cameras, which is much more challenging due to symmetries in the dataset: For example, the left-back camera will produce very similar photos to the front-right camera when the car is passing the same street in the opposite direction. See Sec. 4.3 for context.

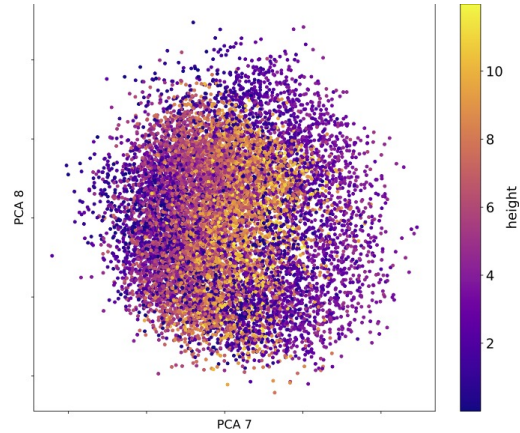


Figure 11. **Latent poses on MSN (cont.)** – The seventh and eighth PCA components of the latent poses show some (non-linear) relation with the ground truth camera height.

Model	Pose	PSNR
SRT [22]	$p_x, p_y$	22.72
SRT $^\dagger$	$p_x, p_y$	23.63
UpSRT $^\dagger$	$\cancel{p}_x, p_y$	21.25
RUST	$\cancel{p}_x, \cancel{p}_y$	22.50

Table 5. **Quantitative results on SV** – See Sec. 4.3 for context.

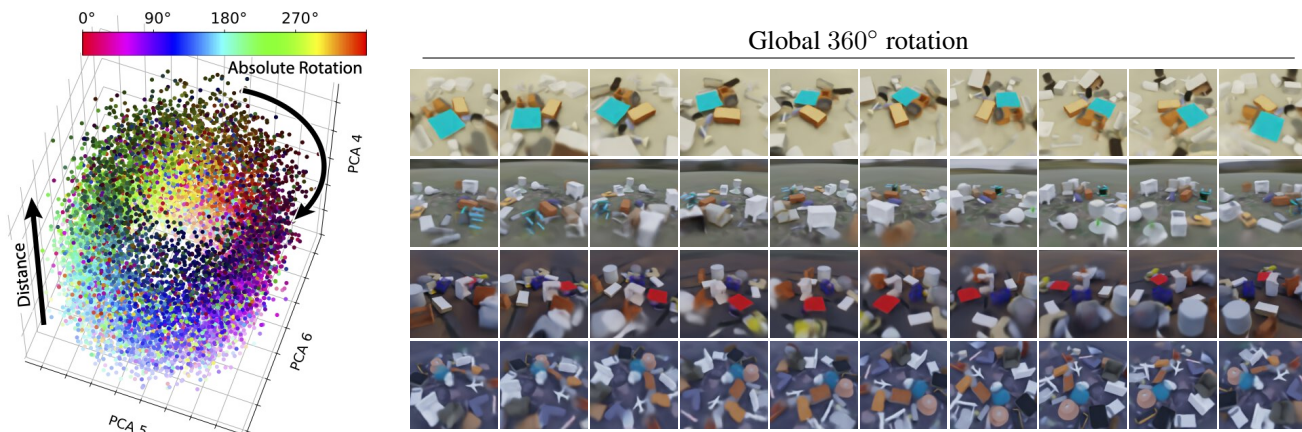


Figure 12. **Latent pose investigations on MSN (cont.)** – In the main manuscript, we investigate the space of latent poses using Principal Component Analysis (PCA). We found meaningful structure in that the first PCA component encodes height, the second and third encode relative rotation, and the fourth encodes distance from the scene center. **Left:** Here we investigate PCA components 5 and 6 which capture absolute rotation. These components, together with component 4, constitute a cylinder. Points are coloured such that intensity represents the camera’s distance from the scene center and hue represents their rotation around the scene’s z-axis in absolute world coordinates. Rotations around this cylinder’s axis are by themselves insufficient to rotate the camera. Instead, moving  $\theta^\circ$  along this cylinder and  $3\theta^\circ$  along the cylinder in PCA components 1, 2 and 3 is necessary to rotate the camera  $\theta^\circ$  around the scene’s z-axis. As shown in the **right**, one can do full  $360^\circ$  rotations using this method. Camera paths that vary distance from the scene center are shown in Fig. 5 of the main manuscript. Videos are provided in the included html file.

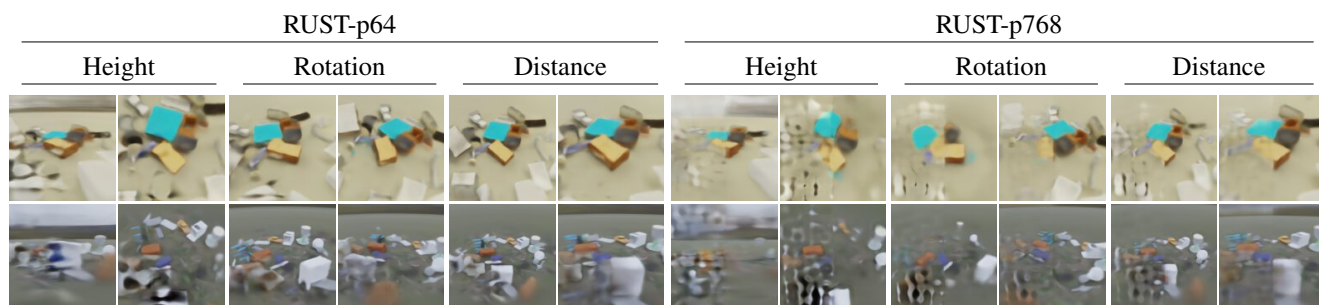


Figure 13. **Latent pose traversals for RUST with large latent poses** – Here we analyze the amount of control afforded by ablations, namely variations of RUST that use 64 or 768 dimensions for the latent pose  $\tilde{p}$ . We refer to these as RUST-p64 and RUST-p768, respectively. We find that camera motion based on PCA decomposition is only possible to a limited extent for these models. For example, we found no good way to perform full  $360^\circ$  rotations. Traversals for camera height, distance, and local rotations are feasible. However, these ablations that use higher-dimensional latent poses  $\tilde{p}$  are prone to auto-encoding the half of the target image that is input to the Pose Estimator (always the left half here). Although the right halves of these images demonstrate 3D camera movement, the left halves show wrong content with heavy artifacts. Videos are provided in the included html file.



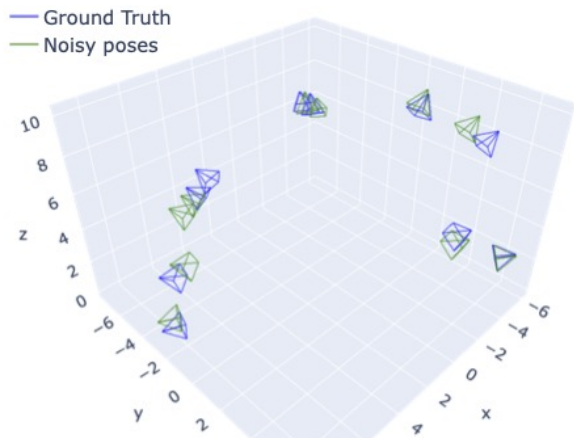


Figure 14. **Perturbed camera poses** – We visualize the synthetically perturbed camera poses on the MSN dataset with  $\sigma = 0.1$ . See Sec. 4.1 for the performance of the models on noisy cameras.

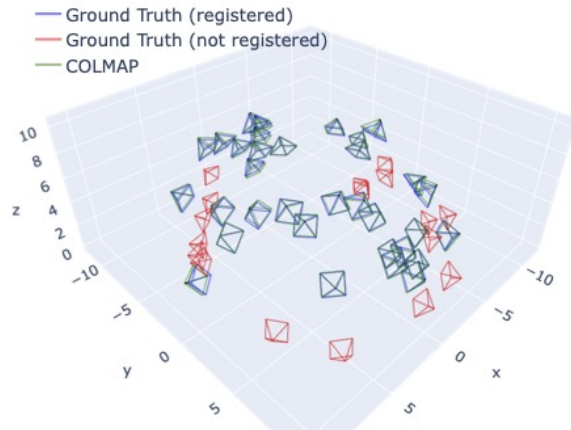


Figure 16. **COLMAP** – Camera positions recovered by COLMAP (oriented towards zero-center solely for visualization). For images with estimated camera pose, COLMAP’s estimates (green) are extremely close to the ground truth (blue). COLMAP failed to pose ground truth cameras in red.

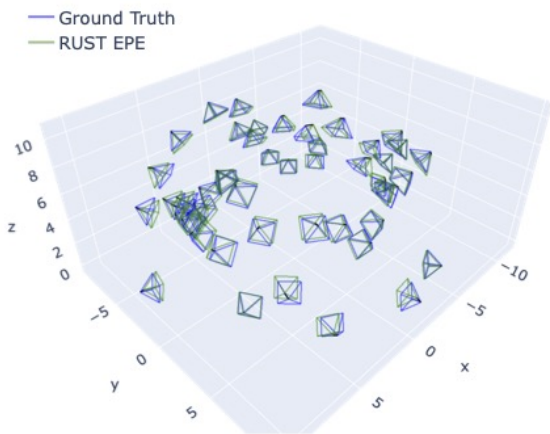


Figure 15. **RUST EPE** – Camera positions recovered by RUST EPE (oriented towards zero-center solely for visualization). RUST EPE estimates (green) are obtained relative to randomly chosen reference views, and very closely match the ground truth (blue).

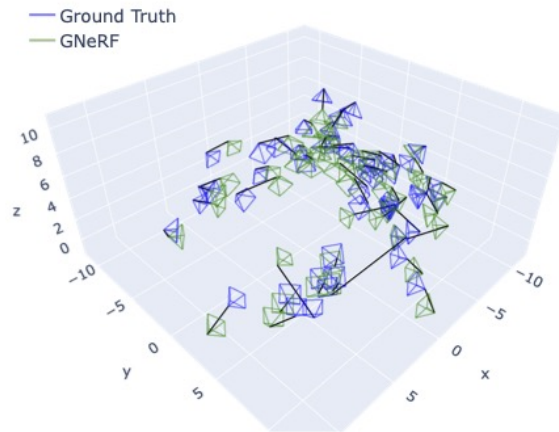


Figure 17. **GNeRF** – Camera positions recovered by GNeRF (oriented towards zero-center solely for visualization). GNeRF predictions (green) approximately recover many ground truth (blue) camera positions, but frequently make very large errors, leading to significant deviations in camera positions.

Target		Perfect poses				No poses			
Normal	FG	150 views		12 views		150 views		12 views	
		Normal	FG	Normal	FG	Normal	FG	Normal	FG
PSNR		21.49	23.25	13.38	17.91	16.14	16.04	11.45	12.74
$R^2$						85.34	89.33	49.14	53.39
MSE						8.42	6.68	28.55	26.64
PSNR		24.69	24.71	17.15	20.57	18.03	19.39	13.58	12.63
$R^2$						93.66	86.88	68.91	45.52
MSE						3.23	7.89	20.20	37.74
PSNR		26.38	23.68	18.06	20.17	21.71	19.91	17.81	16.36
$R^2$						96.72	87.19	80.70	90.58
MSE						1.25	7.27	11.76	4.72

Figure 18. **More results on GNeRF** – Qualitative and quantitative GNeRF [15] results on further scenes. We notice a large variance in terms of model performance. See also Sec. 4.2.1.



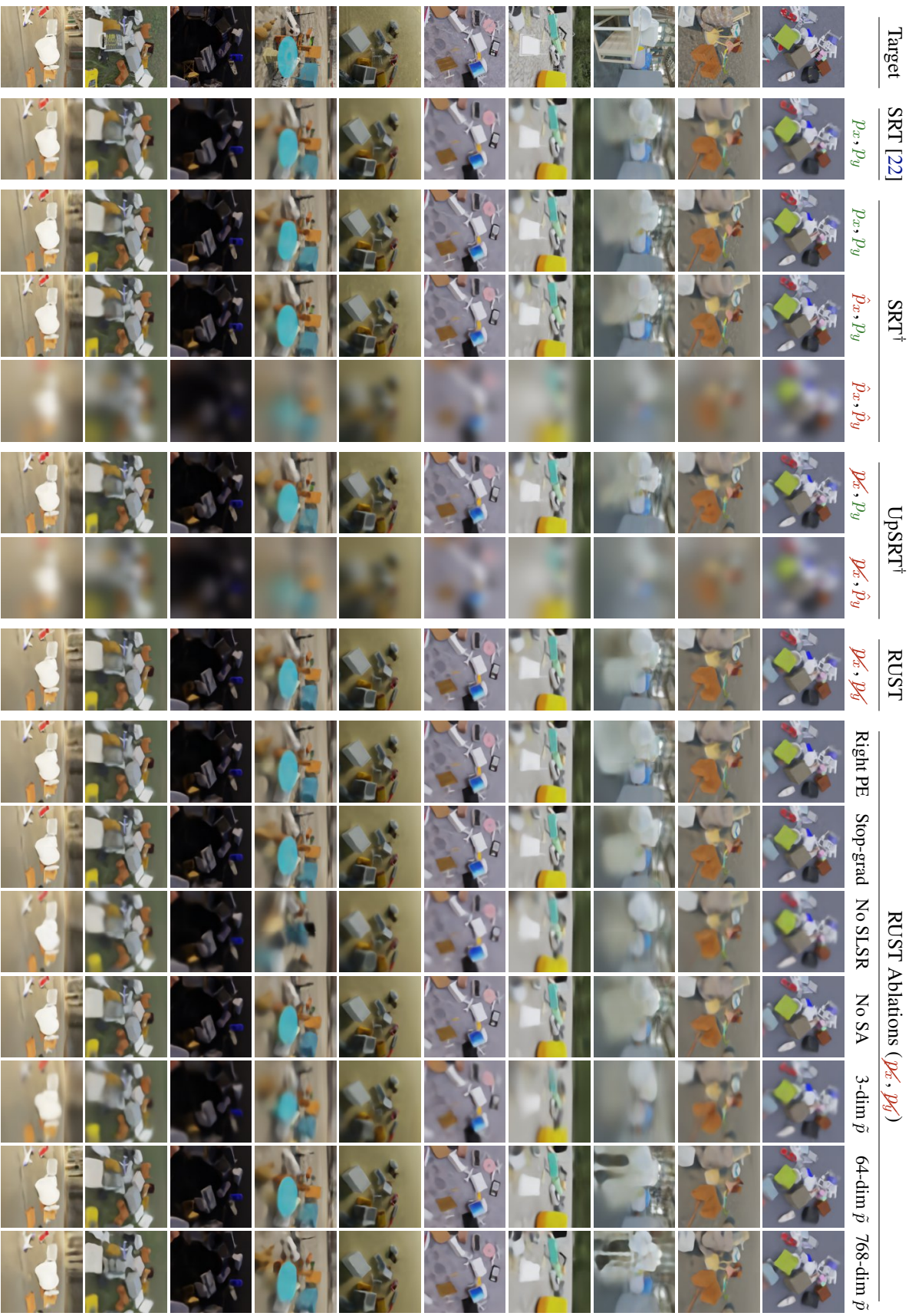


Figure 19. **Full qualitative results on MSN** – Left to right: Target image, SRT [22] as originally proposed, SRT<sup>†</sup> with our improved architecture (different camera pose accuracy settings), UpSRT<sup>†</sup> with our improved architecture (different camera pose accuracy settings), RUST. Further, we show RUST ablations (see Sec. 4.1.1): right-half Pose Embedding, stop gradient, no SLSR, no self-attn, 3-dim.  $\tilde{p}$ , 64-dim.  $\tilde{p}$ , 768-dim.  $\tilde{p}$ .





Figure 20. **Full qualitative results on SV** – Left to right: Target image, SRT [22] as originally proposed, SRT<sup>†</sup> with our improved architecture, UpSRT<sup>†</sup> with our improved architecture, RUST. Street View imagery and permission for publication have been obtained from the authors [5].