Novel-View-Synthesis and Scene-Editing from a single RGB Image

Lukas Höllein, Felix Tristram, Erkin Türköz, Matthias Nießner
Technical University Munich
{lukas.hoellein, felix.tristram, erkin.turkoz, niessner}@tum.de

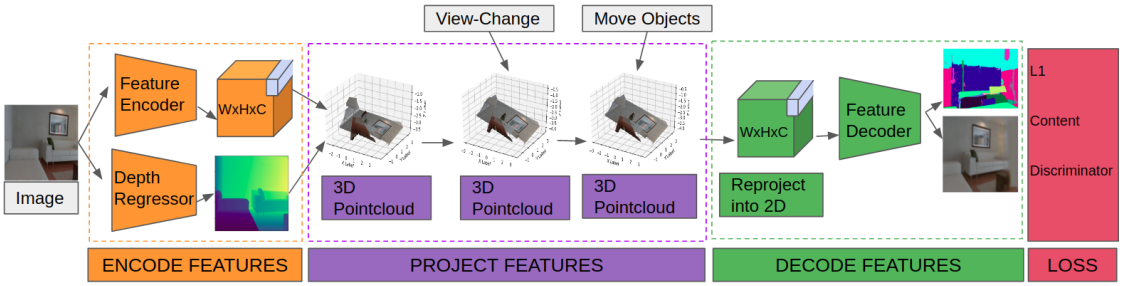

Figure 1: Our pipeline for novel view synthesis and scene editing. First, we encode each pixel into a feature vector and estimate its depth to construct a point cloud. Similar to [11] we then apply a change in view-point. After that we move all points of a desired object by a rigid-body transformation. Similar to [11] we then use a differentiable renderer and a feature decoder network to produce an RGB and a segmentation image from a novel viewpoint and with a moved object.

Abstract

Recent work on novel view synthesis [11] is capable of producing high-quality views from a single RGB image. We build up on this by additionally performing scene editing by moving single objects through a scene with an arbitrary rigid-body transformation. We introduce a neural network that can generate realistic looking images with moved objects and a different viewpoint from a single RGB image. We show state-of-the-art results on novel view synthesis and demonstrate the principle functionality of scene editing.

1. Introduction

Novel view synthesis (nvs) is the process of synthesizing an unseen viewpoint of a scene from one to multiple images. There are two paradigms for this problem: View synthesis from a single image [11] or from multiple images [1, 7].

In this work we combine two different tasks. First, we do novel view synthesis from a single RGB image with a similar pipeline as proposed by Wiles et. al. [11]. Second, we perform scene editing from a single RGB image, allowing us to move an object arbitrarily through the scene. For that, we propose a semantic segmentation loss during training.

We achieve scene editing without having access to ground-truth RGB images, that show moved objects at their new location. Such data is usually not available as real-world RGB images and could only be rendered from a modified 3D reconstruction. Instead, we propose to compare a predicted and ground-truth segmentation image that show the object at its new location. This allows us to still train our network on datasets of real-world RGB images.

2. Related Work

View-Synthesis. There are different approaches to view-synthesis that are currently pursued by the community, with the main difference being the ability to generalize to different scenes or not. One recent work that generalizes is SynSin [11], which uses a differentiable point cloud renderer from PyTorch3D [8] and a decoder network. Other works use semantics and multi-plane images to perform view-synthesis in an image-to-image fashion [4, 5]. The scene-specific works are mostly focusing on reconstructing a scene as accurately as possible from multiple views and rendering novel views from that reconstruction [1, 7].

Scene-Editing. Current object level scene-editing techniques are mostly limited to image-to-image networks. Pix2Pix [6] can be conditioned on semantic images and generate RGB images from that. Moving objects can then be done by modifying the semantic input and generating a new output image. The problem with this approach is that it only outputs any image that satisfies the semantic image, instead of having the possibility to create a specific scene.

3. Method

Our network architecture performs novel view synthesis and scene editing simultaneously from a single RGB image. Figure 1 shows our pipeline.

3.1. Novel View Synthesis

We adapt the pipeline for novel view synthesis as proposed by Wiles et. al. [11]. We learn to synthesize novel views from a single RGB image without any 3D supervision, except for the change in viewpoint which is expressed as a rigid-body transformation. The predicted RGB image, as explained in figure 1, gets compared to the ground-truth RGB image at the novel viewpoint. We use a combination of $L1$ loss, a perceptual loss and a discriminator loss for training: $L_{nvs} = \lambda_1 * L_{l1} + \lambda_2 * L_{perceptual} + \lambda_3 * L_D$

3.2. Scene Editing

We perform scene editing by applying a rigid-body transformation to all points in the 3D point cloud that correspond to the object that shall be moved. We identify these points by masking all pixels of an object in the 2D input image. During training, we use a segmentation of the input to calculate the mask. During inference, we can specify any mask.

We use a sequential decoder network that first predicts the RGB image out of the rendered features and then a segmentation image from this RGB image. We apply the novel view synthesis loss L_{nvs} only to the image areas unaffected by the movement, because we do not use ground-truth RGB images that show the movement (see chapter 1). Instead, the scene editing is trained with an $L1$ segmentation loss on the predicted segmentation image. The ground-truth segmentation image contains the correct position of the object after movement. With that, we use the predicted segmentation image as a proxy to force the network to learn the correct movements. Since the segmentation image is derived from the RGB image, it also needs to show the correct movement. The discriminator loss L_D enforces a realistic inpainting of the region from which the object was moved.

3.3. Dataset

We use the Matterport3D dataset [2] to train our network for both novel view synthesis and scene editing. Like SynSin [11] we use the Habitat framework [9] to extract image pairs with slightly different viewpoints. We use the same scene level train/val/test split as SynSin.

For scene editing, we move an object and generate RGB and segmentation images showing the movement from different viewpoints with a custom pipeline (see figure 2). As this is a manual and slow process, we generated only 29 different object movements with 50 images each resulting in a total of 1450 images so far. Furthermore, the data contains a lot of black background because we could only load the mesh of a single room instead of the whole house due to memory limitations.

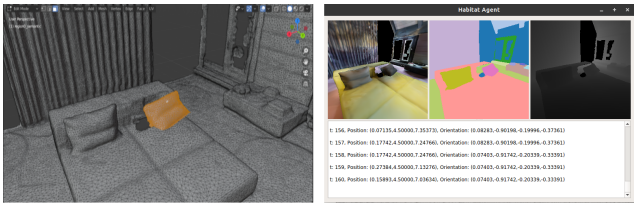

Figure 2: Data generation for modified scenes: (1) Load the region in Blender [3], perform the modification manually (here: move a cushion), export modified mesh. (2) Load the modified mesh in Habitat [9], take RGB, segmentation and depth samples along a manually specified camera trajectory.

4. Results

We show results on novel view synthesis, comparing them to the SynSin results [11], and on scene editing, comparing them to a Pix2Pix2HD [10] baseline.

4.1. Quantitative Results

We compare peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) between our models and the baselines.

As shown in table 1, we reach similar values in PSNR and SSIM scores as the SynSin model [11] for novel view synthesis. All models are trained without ground-truth depth for 50k iterations on the same train/val split as in [11]. Our model that was only trained on novel view synthesis (only nvs) has nearly the same values. Differences can be explained by less hyperparameter tuning and computing resources. The second model (only nvs + segmentation) additionally produces segmentation images. Our third model (nvs + scene editing) is fine-tuned on scene editing. The second and third model have lower results, because they also perform other tasks.

Method	PSNR↑	SSIM↑
SynSin	20.91	0.71
Ours (only nvs)	19.4	0.68
Ours (only nvs + segmentation)	18.8	0.63
Ours (nvs + scene-editing)	17.43	0.50

Table 1: Comparison between our networks and SynSin [11] on the task of novel view synthesis.

In table 2 we show results on scene editing and compare them to a Pix2PixHD baseline [10] that was trained to predict RGB images from segmentation images. We show results from different models (pre-trained on nvs), the first is fine-tuned only on scene editing without novel view syn-

thesis and uses ground-truth depth instead of estimating it. The results get worse when doing novel view synthesis and scene editing at the same time (second model) and when training without ground-truth depth (third model).

Method	PSNR↑	SSIM↑
Pix2PixHD [10]	6.92	0.11
Ours (scene-editing + no nvs + gt depth)	22.23	0.80
Ours (scene-editing + nvs + gt depth)	17.21	0.63
Ours (scene-editing + nvs + no gt depth)	14.49	0.51

Table 2: Comparison on scene-editing between the Pix2PixHD baseline [10] and our model variants.

4.2. Qualitative Results

Here we present some visual samples for novel view synthesis and scene editing. Please zoom in for a better visual comparison. For animated results, please see our website [1].

In figure 3 we show a comparison of our model (only nvs) and SynSin [11] on novel view synthesis. The inpainting of previously unseen regions with geometrically accurate features is achieved by both methods.

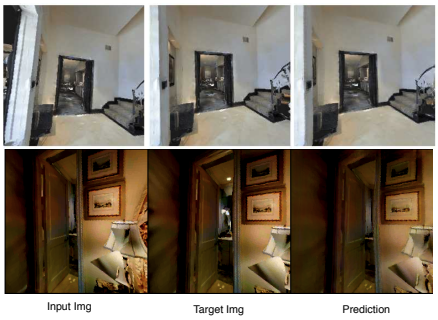

Figure 3: Results on novel view synthesis. From top to bottom: SynSin results [11], our results (only nvs).

In figure 4 we show results on scene editing from our model (scene-editing + no nvs + gt depth) on the training set of our scene editing dataset. Since we only have 1000 samples, we are overfitting to the training images. Nevertheless, the movements in figure 4 use novel, unseen rigid-body transformations for scene editing. Thus, we are able to generalize to new movements for the same objects that the network was trained on.

In figure 5 we show results on scene editing from our model (scene-editing + no nvs + gt depth) and the Pix2PixHD baseline [10] on the validation set of our scene editing dataset. While the baseline just produces an RGB image, that roughly fits the segmentation image, our model learns to move the objects in the same scene. However, the movement does not look as good as in figure 4, due to the low number of samples in our scene editing dataset.

[1] https://github.com/lukasHoel/novel-view-synthesis/tree/master/examples

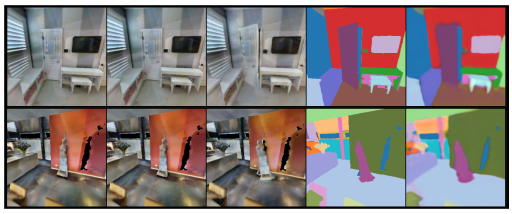

Figure 4: Results on scene editing. From left to right: Input RGB image, gt output RGB image without movement, predicted RGB image, gt segmentation image with movement, predicted segmentation image.

Figure 5: Results on scene editing. From left to right: input RGB image, ground-truth segmentation with moved object, Pix2PixHD [10] prediction from segmentation image, our prediction from RGB image (scene-editing + no nvs + gt depth).

The inpainting in figures 4 and 5 in the areas from which the object was moved away from remains black most of the time. This is because our scene editing dataset contains a lot of black regions in it naturally and thus the discriminator L_D is not penalizing our network for not inpainting the areas. For data that does not contain such regions, the inpainting is working, as shown in figure 3.

5. Conclusion

We present a novel approach for combining novel view synthesis with scene editing and show that we match state-of-the-art results on novel view synthesis [11]. Our results on scene editing look promising, but lack two features: (1) generalization of the movement to unseen scenes and (2) inpainting of regions from which the objects are moved away. Both missing features can be explained by our scene editing dataset which is very small and contains a lot of black background. In the future we want to generate more data of higher quality and hope to solve these two issues. Furthermore, we want to train and test our network on real RGB images from another dataset than Matterport3D [2].

References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics, 2019.

[2] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments, 2017.

[3] Blender Online Community. Blender - a 3d modelling and rendering package, 2018.

[4] Tewodros Habtegebrial, Varun Jampani, Orazio Gallo, and Didier Stricker. Generative view synthesis: From single-view semantics to novel-view images, 2020.

[5] Hsin-Ping Huang, Hung-Yu Tseng, Hsin-Ying Lee, and Jia-Bin Huang. Semantic view synthesis. In *European Conference on Computer Vision (ECCV)*, 2020.

[6] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2016.

[7] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[8] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

[9] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research, 2019.

[10] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans, 2017.

[11] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image, 2019.