MaRINeR: Enhancing Novel Views by Matching Rendered Images with Nearby References

Master Thesis

Author(s):
Bösiger, Lukas

Publication date:
2024

Permanent link:
https://doi.org/10.3929/ethz-b-000665218

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision and Geometry Lab
Prof. Dr. Marc Pollefeys

Master Thesis

MaRINeR: Enhancing Novel Views by Matching Rendered Images with Nearby References

Autumn Term 2024

Supervised by:
Dr. Zuria Bauer
Dr. Mihai-Alexandru Dusmanu

Author:
Lukas Bösiger

Contents

Preface

Before you lies the master thesis *MaRINeR: Enhancing Novel Views by Matching Rendered Images with Nearby References*. It has been written from August 2023 to March 2024 to complete my journey of the masters program at the Eidgenössische Technische Hochschule Zürich.
My interest in Computer Vision and Graphics began at the end of my bachelor studies and since then I had the chance to be part of many amazing projects. In the course of my studies I made animated humans appear in virtual reality, detected and visualized Jet Stream core lines, built my own ray-tracing based renderer and even developed a fun co-op game with friends to escape from an ancient Egyptian god. While I had the opportunity to work with modern mixed reality devices and large clusters I also learned that the best ideas begin often with paper and cardboard prototypes. Despite many courses related to machine learning, before this thesis I never had the chance to work on an extensive deep learning project. This thesis has provided me with the opportunity to gain valuable insights into the most recent research in the field, as well as modern engineering methods used in practice
I would like to thank my supervisors, Dr. Zuria Bauer and Dr. Mihai-Alexandru Dusmanu, for their guidance, support and great ideas while working on this thesis. I also want to thank Prof. Dr. Marc Pollefeys and the Computer Vision and Geometry Group and for providing me with this opportunity and all the resources necessary to complete this project. Finally, I also want to thank my family and friends for being there for me.

Abstract

Rendering realistic images from 3D reconstructions is an essential task of many Computer Vision and Robotics pipelines, notably for mixed-reality applications as well as for training autonomous agents in simulated environments. However, the quality of novel views heavily depends on the source reconstruction which is often imperfect due to noisy or missing geometry and appearance. Inspired by the recent success of reference-based super-resolution networks, we propose MaRINeR, a refinement method that leverages information of a nearby mapping image to improve the rendering of a target viewpoint. We first establish matches between the raw rendered image of the scene geometry from the target viewpoint and the nearby reference based on deep features, followed by hierarchical detail transfer. We show improved renderings in quantitative metrics and qualitative examples from both explicit and implicit scene representations. We further employ our method on the downstream tasks of pseudo-ground-truth validation, synthetic data enhancement and detail recovery for renderings of reduced 3D reconstructions.

Acronyms and Abbreviations

GT Ground Truth 2

low-res. low resolution 2

high-res. high resolution 2

MEM Matching and Extraction Module 6

RefSR Reference-based Super-Resolution 2

ST Style Transfer 4

SAM Spatial Adaptation Module 6

PSNR Peak Signal Noise Ratio 16

SSIM Structural Similarity Index Measure 16

AR Augmented Reality 5

NeRF Neural Radiance Field 1

LPIPS Learned Perceptual Image Patch Similarity 5

ERQA Edge-Restoration Quality Assessment 27

DRAM Dual Residual Aggregation Module 6

Chapter 1

Introduction

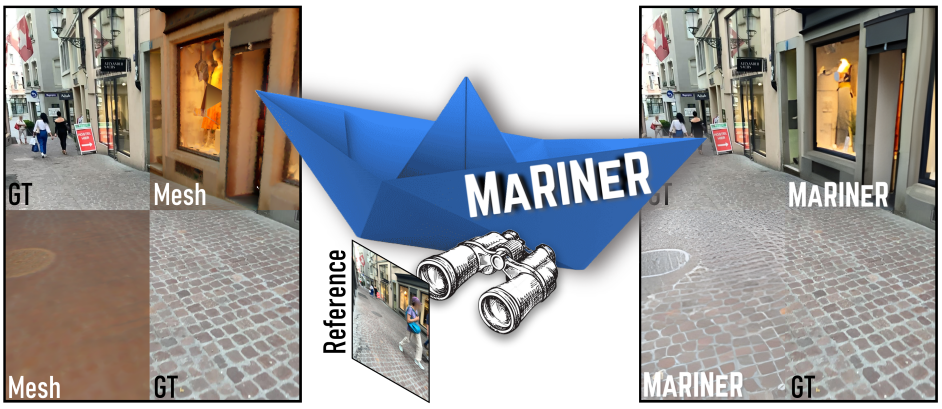


Figure 1.1: We introduce **MaRINeR**: a pipeline taking as input a novel-view obtained from a 3D reconstruction exhibiting geometric and / or appearance artifacts and inaccuracies as well as a nearby reference used during the reconstruction process, and outputting an enhanced version of the novel-view through feature matching and transfer.

One of the fundamental problems of computer vision and robotics is reconstructing the environment from sensorial data such as color or depth cameras or LiDAR scanners. These pipeline produce a computer-friendly representation of the space which can be either explicit (e.g., point-clouds, meshes), implicit (e.g, occupancy nets [1, 2], Neural Radiance Field (NeRF) [3], or hybrid (e.g., Gaussian splats [4]) which serve as starting point for many subsequent tasks, notably novel-view synthesis, environment understanding, planning, and navigation. All existing methods have limitations: point-clouds are highly dependent on the sensor quality [5], often contain artifacts due to moving objects [6], and are not suitable for occlusion checking [7] or pattern rendering. Meshing algorithms often create both appearance and geometric artifacts and inconsistencies while connecting the vertices and coloring / texturing the polygons [8, 9, 10, 11, 12, 13]. More modern implicit methods show exemplary rendering performance but often require very densely sampled frames or even depth maps which are not always available [14]. Furthermore, these methods also need extensive per-scene training. The performance decreases drastically as the frame-rate and the input modalities are reduced. Any artifacts or inconsistencies produced by the reconstruction pipeline can lead to significant impact in downstream tasks.

To address these limitations we propose **MaRINeR**, a post-processing step for novel rendered views by ***Ma**tching the **R**endered **I**mages with **Ne**arby **R**eferences*. To this end, we make further use of input images to the reconstruction process as reference data. Our task is strongly connected with Reference-based Super-Resolution (RefSR) since similar to renderings from low-quality or noisy 3D reconstructions, a naively up-scaled version of a low resolution (low-res.) image lacks details. RefSR methods use details present in a closely related high resolution (high-res.) reference image to help super-resolve the low-res.. image. We notice that the methods used to match between low-res. and high-res. image domains for information transfer and fusion can more generally be used to transfer details from a reference to a related image of any nature. However, the classically used CUFED5 [15] dataset is not suitable for our task of novel view enhancement. We therefore generate new training and test datasets building upon the recently released LaMAR [16] dataset.
The enhanced novel views show promising results for different downstream tasks. First, our method quantitatively and qualitatively narrows the gap between renderings and real images, including these of implicit representations. As a by-product, this improves the quality of data obtained when using digital twins for training reinforcement agents, following the recent advances in egocentric human synthetic data generation [17]. Second, many recent datasets (12 Scenes [18], RIO10 [19], LaMAR [16]) designed pseudo-ground-truth pipelines to automatically and accurately register trajectories from various devices, notably in the context of mixed reality experiences. One common way to validate the accuracy of these pipelines is through qualitative checks between the aligned images and an associated rendering from a 3D reconstruction. Narrowing the render-to-real gap by removing the artifacts and improving the textural accuracy and realism opens the door to automating this process by taking advantage of existing geometric methods to estimate the accuracy of the Ground Truth (GT). Third, for any downstream application, decisions have to be made to reduce the size of 3D representations in order to efficiently process them at the cost of reduced detail accuracy and realism. Our method is capable of recovering the lost details and realism of such 3D reconstructions.
To summarize, our contributions are:

- We introduce **MaRINeR** which, to the best of our knowledge, is the first method enhancing novel views by using a close-by reference image that is applicable for renderings from a wide range of 3D reconstruction pipelines.
- An extensive evaluation of the proposed method is performed, providing not only a qualitative and quantitative analysis of the method but also an overview of the robustness of **MaRINeR** to different datasets, temporal conditions and temporal changes in scenes, while discussing limitations.
- We showcase the excellent performance of our model in several applications: elimination of manual checks in pseudo-GT pipelines, improvement of synthetic AR trajectories, and enhancing the output of neural renderings.

Chapter 2

Related Work

We provide an overview of related research fields which also incorporate information from a reference image into a target image, notably reference-based image super-resolution and style transfer.

2.1 Reference-based image Super-Resolution

The goal of RefSR is to recover high-res. from low-res. images by transferring missing details from high-res. reference images. The methods usually work by aligning and fusing features extracted from low-res. and reference images. While early work uses hand crafted features [20], more recent works use either pre-trained features [15, 21] or train the feature extraction end-to-end with the task [22, 23, 24, 25, 26, 27, 28, 29]. The alignment of the reference and low-res. features proposes a challenge because of the resolution difference. Some methods use implicit alignment: CrossNet [23] estimates the optical flow between reference and low-res. images. Because optical flow fails at capturing long distance correspondences, SSEN [25] utilizes deformable convolutions which ensure a large receptive field. Other work uses explicit alignment by feature or patch matching. SRNTT [15] uses feature similarity and transfers textures from the reference images at different scales. To reduce the computational complexity, MASA-SR [28] proposes a coarse-to-fine correspondence matching module. C^2-Matching [27] introduces knowledge distillation and contrastive learning methods to improve the matching between low-res. and reference despite the resolution gap. WTRN [21] uses wavelets to separate high and low frequency parts of the images, which helps to more transfer more visually plausible texture patterns. DATSR [30] uses Swin-Transformers [31] to replace the commonly used residual blocks [32], leading to more robust matches and texture transfer. HMCF [33] improves the matching between low-res. and reference of similar objects with different texture by using high-to-low-level feature matching and complementary information fusion. RRSGAN [29] uses generative adversarial networks and deformable convolutions. FRFSR [34] notes that the commonly used perceptual and adversarial loss have an adverse effect on texture transfer and reconstruction. As a solution, they propose the use of a texture reuse framework. RRSR [35] uses a reciprocal learning strategy to strengthen the training process by using the super resolution result as reference to help super-resolve a low-res. variant of the original high-res. reference. CMRSR [36] notes that due to the gap between inputs and reference, the super resolution image often yields distortions and ghosting artifacts and they propose a contrastive attention-guided multi-level feature registration module to mitigate those. There are also methods that use multiple references as input such as CIMR-SR [37], AMRSR [38], AMSA [39] or LMR [40]. We notice that many of the ideas

to align reference and low-res. features are not limited to align images with resolution differences but can be used more broadly to also align rendered images to real images.

2.2 Style Transfer

Artistic Style Transfer (ST) methods transfer the style of a style image to a content image. A subcategory are the universal ST methods [41, 42, 43], which transfer any style to the content image. This can be done by separating content and style information in the images. AdaIN [42] transfers channel-wise mean and variance feature statistics. WCT [41] uses whitening and coloring transformations, where the whitening transformation can remove the style of the content image and the coloring transformation can incorporate the style of the style image. However the separation of content and style is challenging and some content can be corrupted. ArtFlow [43] calls this issue content leak and introduces a reversible neural flow-based network to avoid it. StyTr2 [44] uses transformers to extract and maintain global image information, which then help with the content leak problem. For universal ST, the style images usually have little content in common with the content image. The results look like an artistic version of the content image which is however far from being realistic. Semantic ST methods [45, 46] work with style images that contain similar objects as the content image. The goal is to build semantic correspondences between similar objects and map the style region only to the semantically similar content regions [47]. NNST [45] matches VGG [48] features between content and style and replaces the content features with the nearest style features. MST [46] uses graph cuts for matching between content and style features. While those methods work well at transferring the semantic correspondences, they can introduce distortions and don't produce photo-realistic images. Photo-realistic ST methods aim at transferring the style of the color distribution while preserving the structures of the content image [47]. WCT2 [49] adds a wavelet based correction to the whitening and coloring transforms of WCT [41]. This helps to preserve the structural and statistical properties of the VGG features during stylization. The result is a more photo-realistic image without distortions. However, photo-realistic ST assumes that the content image is already photo-realistic. If this image contains artifacts, then those are also carried over to the stylized image.

Chapter 3

Method Development

The task of novel-view enhancement using a reference image is a research field that has no direct prior work. Related fields like RefSR and ST exists but their tasks have different requirements. For example, we aim to enhance renderings of 3D reconstructions that may contain appearance and geometric artifacts. RefSR assumes the artifacts come only from the resolution difference and ST does not address the possibility of wrong content in the content image. These different requirements are reflected by both datasets and architectures used, which are not optimal to enhance novel-view renderings. In this chapter we describe how we take an existing RefSR method MASA-SR [28], analyze its architecture and with the help of several visualizations adapt it to the novel-view enhancement task. Because commonly used RefSR datasets are not suitable for our purpose, we use the newly released Augmented Reality (AR) dataset LaMAR [16] to create training and test data.

3.1 Choosing a RefSR method

We notice that the property of transferring information from a high-res. to a low-res. image is not limited to the low-res. and high-res. domain and can also be applied to transfer information from a reference image to renderings of any 3D reconstruction pipeline. For this we search for a RefSR method that can be built upon. The method we choose is MASA-SR [28] because the architecture of the model is relatively small resulting in fewer trainable parameters then newer state-of-the-art methods such as FRFSR [34]. Despite its small size, it still beats more recent methods considering Learned Perceptual Image Patch Similarity (LPIPS) [34] and gives visually pleasing results. Furthermore, the source code is publicly available and uses recent software packages, giving the project a good start without having to work with outdated library versions. Some alternative models are very large in the number of trainable parameters [30, 15] and therefore require a lot of time and resources to train. Many models do not include the full implementation code [50, 35, 51, 52, 33, 34] and others have the assumption of a low-res. input tightly coupled to the architecture making it difficult to adapt the model later to a differently shaped input [27]. Alternatively, one could also attempt to adapt an existing ST method. There is however one fundamental difference in the way those methods are trained. RefSR methods train fully supervised having a precise GT available. For ST methods on the other hand this is not the case. The GT is usually unknown and one possible way of training those models is using a cycle loss [45] making sure the features of the generated image are closely related to the ones of the style image. Because the enhanced rendering should optimally look like a real image, it is possible in our case to acquire

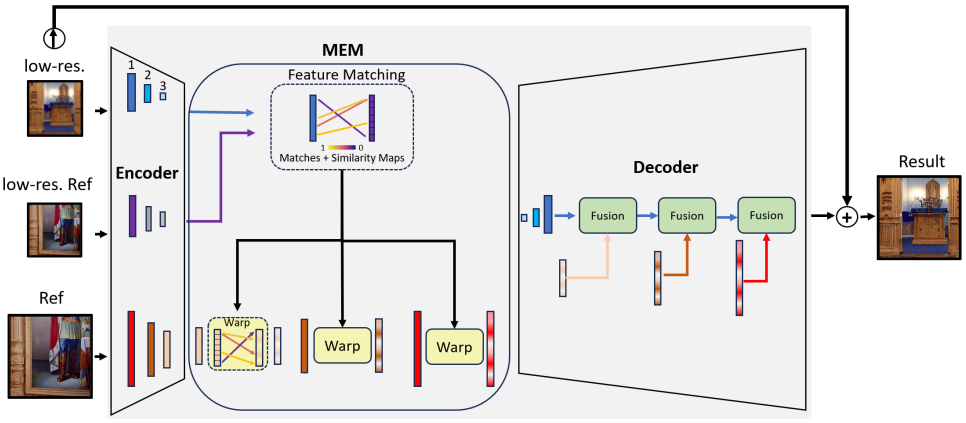


Figure 3.1: MASA-SR [28] pipeline. The RefSR method MASA-SR takes a low-res. image and a high-res. reference as input and produces a super resolved version of the low-res. image.

GT data by taking device captured images.

3.2 MASA-SR

Fig. 3.1 shows the pipeline of the MASA-SR [28] method. The method takes a low-res. image and a high-res. reference as input. Because of the resolution difference, the model down-samples the reference image resulting in a third input, the low-res. reference. All those are then run through an end-to-end trained encoder, resulting in feature maps of three different levels where level 2 and 3 each halve the spatial resolution of the features. The features of level 1 are then used for feature matching, finding similar content in both low-res. image and low-res. reference. This step is performed by the Matching and Extraction Module (MEM). For performance reasons, those matches are established hierarchically: First on a coarse block level and then on a fine patch level withing the matching blocks. Using those matches, the high-res. reference features are warped, moving the reference features to the location of the matching low-res. features. This is done patch-wise on all three levels of the reference features, where the patch size is chosen relative to the spatial resolution of the feature level. At the end, the decoder fuses the low-res. and warped high-res. features together keeping the content from the low-res. image while adding missing details from the warped reference. For fusion, the decoder uses Spatial Adaptation Module (SAM)s and Dual Residual Aggregation Module (DRAM)s where SAM maps the color distribution of the reference to that of the low-res. image, ensuring the color distribution and illumination of the low-res. image remains unchanged. DRAMs are an improved way of merging features, with the goal to enhance the details of both inputs. As a last step, an up-scaled version of the low-res. input is added to the output of the decoder, giving the pipeline a further signal to stay content-wise close to the low-res. input image. Because much of this pipeline can be reused for our task, in the following we go into more details regarding architecture specific blocks such as the encoder, the feature matching of the MEM and the fusion modules with the SAMs and DRAMs.

3.2.1 Encoder and feature matching

The encoder consists of three connected levels, where each level consists of residual blocks [32] and a convolution that halves the spatial resolution of the features. The

result are following dense feature tensors for each of the input images:

$$\begin{aligned} \{\mathcal{F}_1^R \in \mathbb{R}^{H\times W\times F}, \mathcal{F}_2^R \in \mathbb{R}^{H/2\times W/2\times F}, \mathcal{F}_3^R \in \mathbb{R}^{H/4\times W/4\times F}\} \\ \{\mathcal{F}_1^{R\downarrow} \in \mathbb{R}^{H/4\times W/4\times F}, \mathcal{F}_2^{R\downarrow} \in \mathbb{R}^{H/8\times W/8\times F}, \mathcal{F}_3^{R\downarrow} \in \mathbb{R}^{H/16\times W/16\times F}\} \\ \{\mathcal{F}_1^I \in \mathbb{R}^{H/4\times W/4\times F}, \mathcal{F}_2^I \in \mathbb{R}^{H/8\times W/8\times F}, \mathcal{F}_3^I \in \mathbb{R}^{H/16\times W/16\times F}\} \end{aligned} \tag{3.1}$$

for the reference (R), the low-res. reference ($R\downarrow$) and the low-res. image (I). For brevity of the notation we assume the spatial resolution of the reference and the up-scaled low-res. image to be the same. In practice those can be different.

Matching and Extraction Module (MEM)

For feature matching and alignment, MASA-SR [28] uses MEM. The goal is to find feature matches between the low-res. and down-scaled reference image. Those can then be used to build an aligned version of the reference. The best matching images locations are extracted from the reference and warped to their matching location in the low-res. image. Those aligned features can subsequently be fused with fusion modules. Fig. 3.1 shows an overview over the module. The matching is performed on the level 1 features of the low-res. image and the down-scaled reference $\mathcal{F}_1^I$ and $\mathcal{F}_1^{R\downarrow}$. The MEM performs patch matching first on coarse blocks and then densely within those blocks. The result of the matching is an index map $\mathcal{D} \in \mathbb{N}_0^{H/4\times W/4}$ and a similarity map $\mathcal{S} \in \mathbb{R}^{H/4\times W/4}$. The similarity between two normalized patches $p_i^I \subset \mathcal{F}_1^I$ and $p_j^{R\downarrow} \subset \mathcal{F}_1^{R\downarrow}$ is calculated using the cosine similarity $r_{i,j} = (p_i^I)^T p_j^{R\downarrow}$. The index map is defined as $\mathcal{D}_i = \text{argmax}_j r_{i,j}$ and the similarity map as $\mathcal{S}_i = \max_j r_{i,j}$. Those two maps are used to warp the reference features $\mathcal{F}_s^R$ at all three levels $s \in \{1, 2, 3\}$, where patches with size relative to the spatial resolution of the features are cropped for the different levels. The index map $\mathcal{D}$ is used to create a list of rearranged reference patches $\mathcal{L}_i^s = \text{crop}(p_{\mathcal{D}_i}^R, s)$, where $\text{crop}(p, s)$ crops the patch p relative to the spatial feature resolution of the level s. The rearranged reference features are obtained as follows: $\mathcal{F}_s^{R\to I} = \text{fold}(\mathcal{L}^s) \odot \mathcal{S}^{\uparrow_s}$, where $\odot$ denotes the element-wise multiplication and $\uparrow$ bi-linear interpolation. $\text{fold}(\cdot)$ is the operation used to combine the patches where overlapping values are averaged. This thus yields:

$$\{\mathcal{F}_1^{R\to I} \in \mathbb{R}^{H\times W\times F_1}, \mathcal{F}_2^{R\to I} \in \mathbb{R}^{H/2\times W/2\times F_2}, \mathcal{F}_3^{R\to I} \in \mathbb{R}^{H/4\times W/4\times F_3}\} \tag{3.2}$$

Multiplying the features with the similarity map has the effect that matches with low feature similarity get less weight and have little influence on the super resolved image.

3.2.2 Feature fusion

Fig. 3.2 shows the architecture of the blocks used by MASA-SR [28]. The fusion module is responsible for merging low-res. $\mathcal{F}^{lr}$ and high-res. $\mathcal{F}^{hr}$ features, where the $\mathcal{F}^{lr}$ features have halve of the spatial resolution of $\mathcal{F}^{hr}$. $\mathcal{F}^{lr}$ comes directly from the encoder or from the output of a previous fusion block. $\mathcal{F}^{hr}$ comes from the warped reference features. The fusion module uses SAM and DRAM to merge the features and residual blocks [32] to process them.

Spatial Adaptation Module (SAM)

Based on the observation that in many situations the low-res. and high-res. reference image have similar content and texture but different color and illumination distributions, the features may not merge optimally in the fusion process [28]. To address

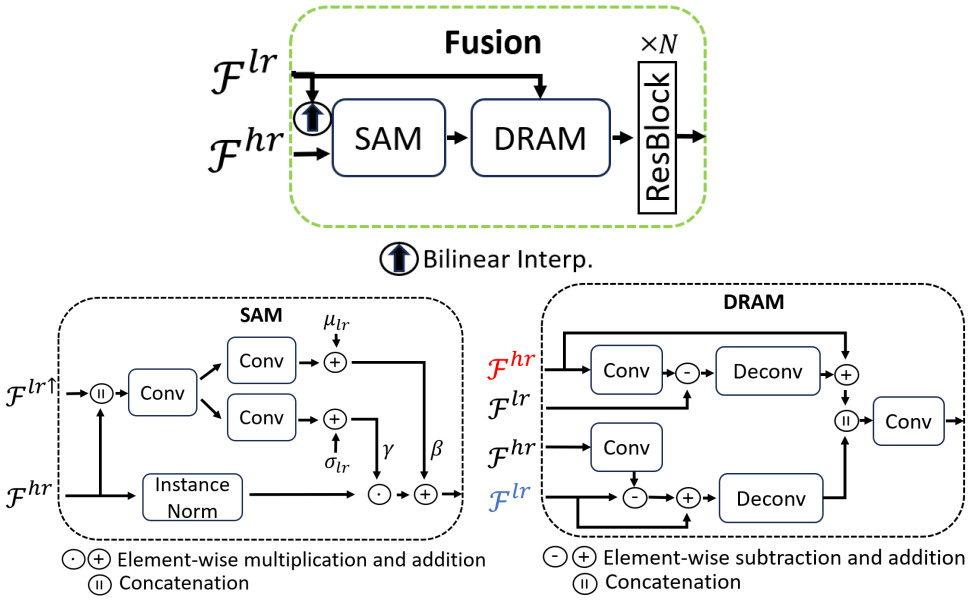


Figure 3.2: MASA-SR [28] building blocks. The feature fusion part of the pipeline consists of SAM, DRAM and fusion modules. The DRAM has two branches: The low resolution and high resolution branches responsible to refine the details of their respective features.

this issue, MASA-SR [28] introduces SAM. The structure is illustrated in Fig. 3.2. The idea is to remap the distribution of $\mathcal{F}^{hr}$ features to that of the up-scaled $\mathcal{F}^{lr\uparrow}$ features. To this end, the module learns two spatial adaption parameters γ and β using convolutions, where the convolutions take the concatenated $\mathcal{F}^{hr}$ and $\mathcal{F}^{lr\uparrow}$ features as input. Because the difference between the features varies based on the spatial location, the mean μ_{lr} and standard deviation σ_{lr} of $\mathcal{F}^{lr\uparrow}$ are added to the learned parameters before updating the instance normalized [53] $\mathcal{F}^{hr}$ features:

$$\begin{aligned} \beta &= \text{conv}(\mathcal{F}^{hr}||\mathcal{F}^{lr\uparrow}) + \mu_{lr} \\ \gamma &= \text{conv}(\mathcal{F}^{hr}||\mathcal{F}^{lr\uparrow}) + \sigma_{lr} \\ \mathcal{F}^{hr} &= \mathcal{F}^{hr} \cdot \gamma + \beta \end{aligned} \tag{3.3}$$

where $||$ stands for concatenation of the features and the other operations are performed element-wise.

Dual Residual Aggregation Module (DRAM)

The DRAM is used to merge two feature tensors of different resolutions: $\mathcal{F}^{hr}$ and $\mathcal{F}^{lr}$. A naive merging approach would be to concatenate the features and feed the into a merging convolution layer, which leads to non-optimal results because of the resolution difference [28]. MASA-SR proposes an alternative approach that consists of two branches: the low-res. and high-res. branch. Both branches aim to refine the high frequency details of their input features before merging them using concatenation and a convolution. Fig. 3.2 shows the architecture of the module. Convolutions are used to reduce the spatial resolution of the high-res. features to that of the low-res. features and deconvolutions to increase the resolution back to the high resolution. Before the final merging convolution, the features of both branches

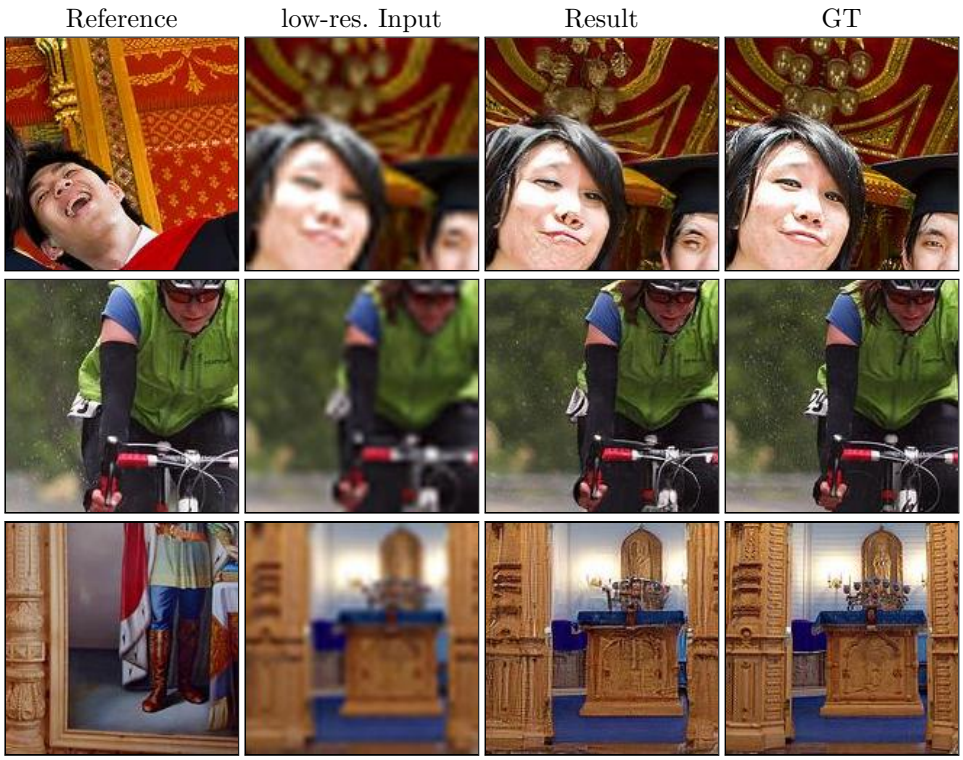


Figure 3.3: MASA-SR [28] trained on CUFED5 [15]. Successful reproduction of the MASA-SR results as described by their work.

are of similar spatial resolution, resulting in the following expression decribing the output $\mathcal{O}$:

$$\begin{aligned} \mathcal{F}^{hr}_{branch} &= \mathcal{F}^{hr} + \text{deconv}(\text{conv}(\mathcal{F}^{hr}) - \mathcal{F}^{lr}) \\ \mathcal{F}^{lr}_{branch} &= \text{deconv}(\mathcal{F}^{lr} + (\mathcal{F}^{lr} - \text{conv}(\mathcal{F}^{hr}))) \\ \mathcal{O} &= \text{conv}(\mathcal{F}^{lr}_{branch} || \mathcal{F}^{hr}_{branch}) \end{aligned} \tag{3.4}$$

where $||$ stands for concatenation of the features and the other operations are performed element-wise.

3.3 MASA-SR experiments

We take a close look at MASA-SR [28] and perform various experiments to understand if we can reproduce the results and if the architecture manages to over-fit to a single image. This experiments are performed to make sure the architecture performs in the way as it is described and not, for example, ignores the reference image.

3.3.1 Reproducing results

As a first step, we train MASA-SR [28] on the CUFED5 [15] dataset, which is typically used to train RefSR methods. CUFED5 consists of 11871 training image pairs where each pair contains a high-res. image and a corresponding reference at a resolution of 160 x 160. The low-res. input image is obtained by down-scaling the high-res. image by a factor of 4. The goal is then to super-resolve the low-res.

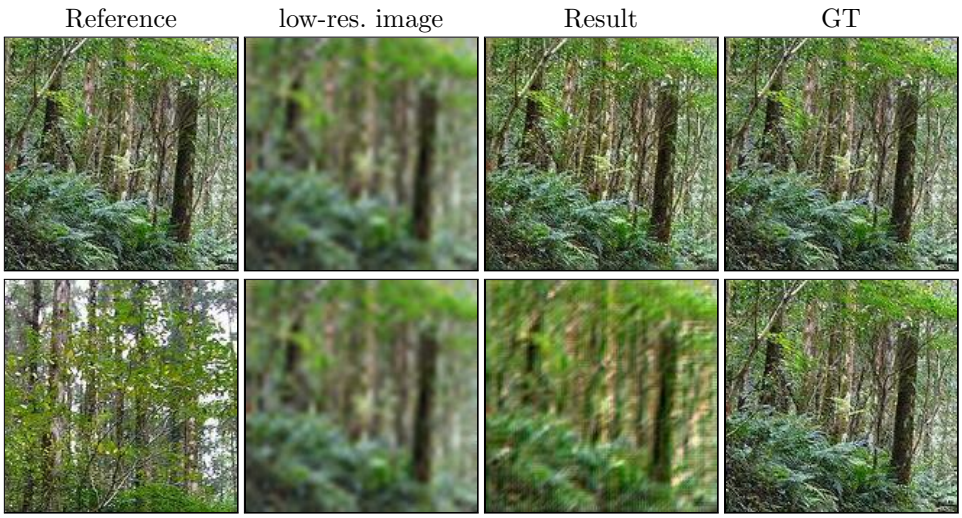


Figure 3.4: MASA-SR [28] over-fitting experiment. In the first experiment the reference is similar to the GT and we train with only one image to ensure the network can learn to predict identity. To make sure the network makes use of the reference details we train it with one image and a different reference.

image back to the high resolution. The CUFED5 [15] test set consists of 126 image pairs where each high-res. image comes with 5 references. Fig. 3.3 shows that we are successfully able to reproduce the results as they are described in their work.

3.3.2 Over-fitting to a single image

To validate the model architecture, we train MASA-SR only on one image. This image is used as low-res., high-res. and reference image. The goal is to make sure the model can over-fit to this one image simulating the simplest scenario where the model does not have to generalize. Fig. 3.4 shows the results of the experiment, where the GT is successfully recovered. To verify that the model takes the reference into account and not only enhances the input while ignoring the reference, we run a second experiment. In this experiment we train again on only one training image that is used as low-res. and high-res. input. However, this time the reference is a different image. Fig. 3.4 shows that the model did not manage to perfectly reconstruct the GT. This means that the model is not ignoring the reference and the quality of the result is related to the quality of the reference image.

3.4 Dataset

The goal of our project is to enhance renderings of novel views created by 3D reconstructions. We aim to do this by building on top of a RefSR method. Currently these methods train on CUFED5 [15] data which does not reflect the goal of our task since, as Fig. 3.3 shows, images of this dataset use a low-res. version of the GT as input and not a 3D reconstruction rendering. We therefore utilize data from LaMAR [16] to build new training and test sets. I the following we will elaborate on the process of training and test dataset creation from the LaMAR dataset.

3.4.1 LaMAR

We use the recently introduced LaMAR dataset [16] to create training and test datasets. LaMAR consists of a large-scale dataset captured using AR devices. The

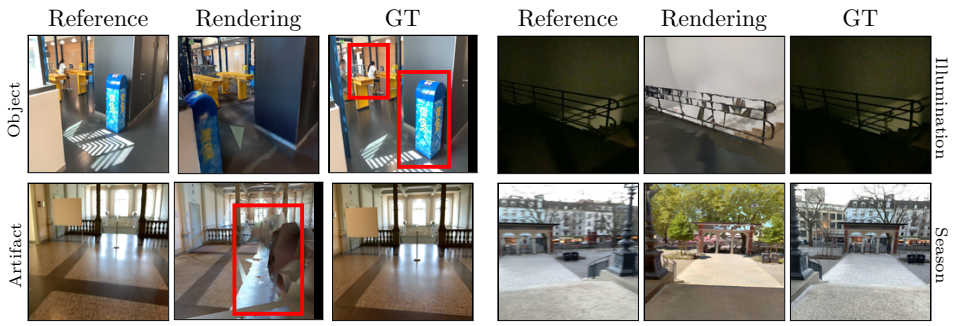


Figure 3.5: Common dataset challenges. There can be different objects present between rendering and GT, some of which can be artifacts. The illumination can also be different because of day time or seasonal changes.

dataset consists of three different scenes: CAB, LIN and HGE. CAB is a multi-floor office building, LIN is a few blocks of an old town and HGE the ground floor of a historical university building. The data was captured using head-mounted HoloLens 2 and hand held iPhone / iPad devices. The device trajectories were recorded at day and at night times and over the course of up to 1 year. What LaMAR also provides is a ground-truthing pipeline to automatically and accurately register the AR device trajectories against large-scale 3D laser scans [16]. The 3D laser scans are obtained using NavVis M6 laser scanners which use inertia-lidar SLAM to estimate for all scanned images poses relative to the beginning of the session and to create a colored 3D point cloud with grid resolution of 1cm. The point cloud is then transformed to a dense mesh using the Advancing Front algorithm [54]. We can use this 3D reconstruction to render images. The device trajectories and the mesh reconstructions are available to download. The renderings of those meshes have to be generated by ourselves. Because of the automatic GT pipeline for localization, the device trajectories are all registered inside the common 3D reconstruction. This registration gives us for every frame inside the trajectories a camera pose of the orientation and location where the image of this frame was taken. For the task of novel view enhancement, we need three images: The rendering of a novel view, the GT, showing a realistic image with the same viewpoint as the rendering, and a close by reference image. To generate those from a registered AR trajectory we do the following: Every frame inside the trajectory is captured by a device and therefore realistic. We use those frames as the GT. The camera pose that comes with the registered frame is used to create a rendering from the mesh. This is then used as the rendering input image and has the same viewpoint as the device captured image. Because the trajectory data is ordered temporally and recorded with 1 frame per second, we take the surrounding frames as reference images. Fig. 3.5 shows examples of image pairs created this way. Because the trajectories were recorded over the course of up to one year and at different day times, the content of the rendering and the GT images is not exactly the same. While the viewpoint is similar, different objects can be present in GT and rendering. There can also be seasonal changes or strong illumination changes for night time recorded trajectories. Furthermore, the rendering can have artifacts that appeared during the process of 3D reconstruction generation.

3.4.2 Training and test datasets

We create a training set from the device trajectories capture in the CAB and LIN scenes. We merge the scenes because CAB contains mostly indoor and LIN outdoor

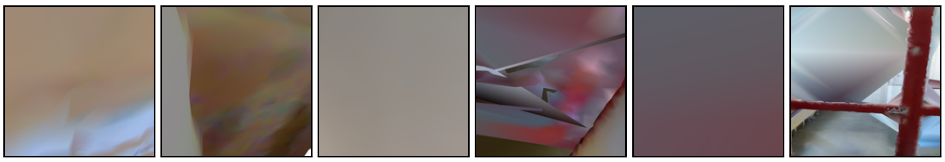


Figure 3.6: Removal of unrecognizable renderings. We filter the data to remove renderings that are mostly artifact.

recordings. Test sets are created from all three scenes: CAB, LIN and HGE. HGE is used to test generalization to novel scenes. Processing all trajectories available, the merged CAB and LIN training set contains 21350 image pairs which is roughly twice the amount of images used to train the MASA-SR [28] model. The CAB, LIN and HGE test sets consist of 329, 608 and 492 image pairs. The datasets contain different references of various levels for each rendering: a low level indicates that the reference pose is close to the GT pose (easier) and a high level indicates that the reference is further away (harder). Because RefSR methods usually train on the CUFED5 [15] dataset which consists of images with resolution 160x160, we also re-scale our dataset images to this resolution.

3.4.3 Filtering

The dataset created by LaMAR [16] contains various rendering artifacts. While it is the goal of our task to also remove rendering artifacts, some images are unrecognizable. Fig. 3.6 shows a collection of such renderings. We argue that gradients related to them would not benefit the training process of the model and therefore we filter the data first to remove such renderings. We do this by calculating a homography error in a similar way as SuperPoint [55]. Because the images are localized, we can estimate a homography between the rendering and GT. While this is no accurate way of assessing whether the localization was successful, it is enough to filter out those renderings that are mostly artifact. We estimate the homography based on SuperPoint [55] features with SuperGlue [56] matches. Ideally the homography should be identity. The homography error uses the estimated homography to remap the corners of the image. If the corners end up at their original position, the homography is close to identity and the error is close to zero. Using this method we filter out 32 % of the data. Because LaMAR provides renderings containing the depth, we also export a mask to potentially mask out pixels with invalid depth values. However, this mask is not used in the final pipeline.

3.5 MASA-SR with LaMAR data

Using the new datasets we can directly use the pre-trained MASA-SR [28] to do inference on our test data. Because the resolution of our renderings is similar to the resolution of our reference, we first downscale the rendering to be of the shape RefSR methods expect as input. To create an easier task for the network, we use the GT as reference image instead of a close-by view for evaluation only. Fig. 3.7 shows the the pipeline of this approach and Fig. 3.8 shows the results. The RefSR method almost recovers the rendering image that was down-scaled before. Details are however lost in the process and artifacts are still present. We notice that the RefSR method is not using the information from the reference image. A possible explanation of this observation can be that the feature encoder of MASA-SR is trained end-to-end on the CUFED5 [15] data. Because it has never been trained

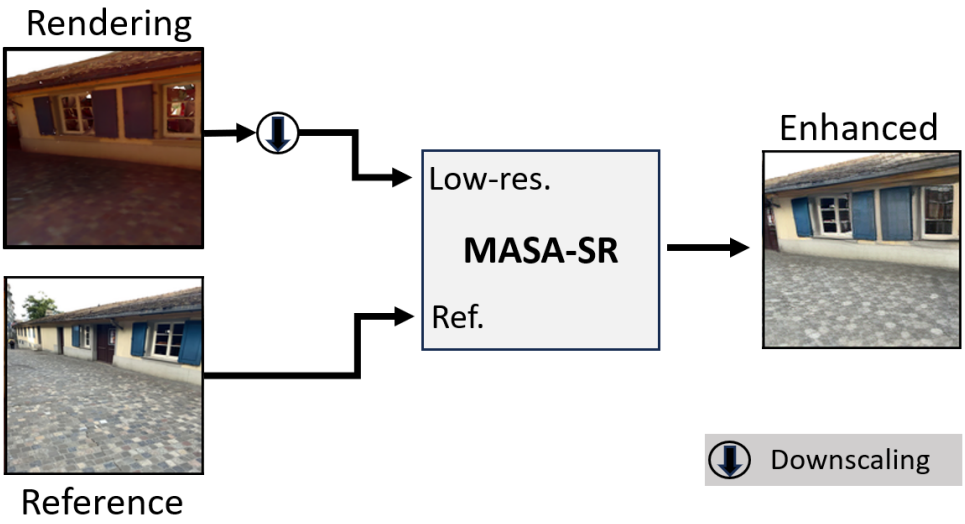


Figure 3.7: First Pipeline. Using the RefSR method MASA-SR [28] with a down-scaled LaMAR [16] rendering and a close by view as reference.

Figure 3.8: Results of MASA-SR [28] on LaMAR [16] data. Inference using the model pre-trained on *CUFED5* data leads to little enhanced realism. Training the model on *LaMAR* data successfully transfers some information from the reference but does not manage to remove the larger rendering artifacts.

to match between features obtained by renderings and the ones of real images, this matching is likely to fail. When RefSR methods fail to find correspondences in the reference, then they aim to enhance the details without the reference. This would

explain why the rendering could be mostly recovered but has no information from the reference. However, also other possible explanations for this observation are possible. Because we also created a training set from LaMAR [16], we train MASA-SR using the pipeline in Fig. 3.7. Doing this, we aim for the model to create the ability to match between the render and real domains. Fig. 3.8 shows the results of the training. This time we notice some information that is transferred from the reference to the rendering. Even tough some image parts look now enhanced, still details are missing or wrong new details are introduced. Also, larger artifacts do not disappear. We conclude that some architectural change are necessary. However, we still do not understand precisely why the method does not work in our favour. For example, why are the details from one part of the image successfully transferred but others are not. The first image in Fig. 3.8 shows that this happens even if the image parts are right next to each other and belong to the same object.

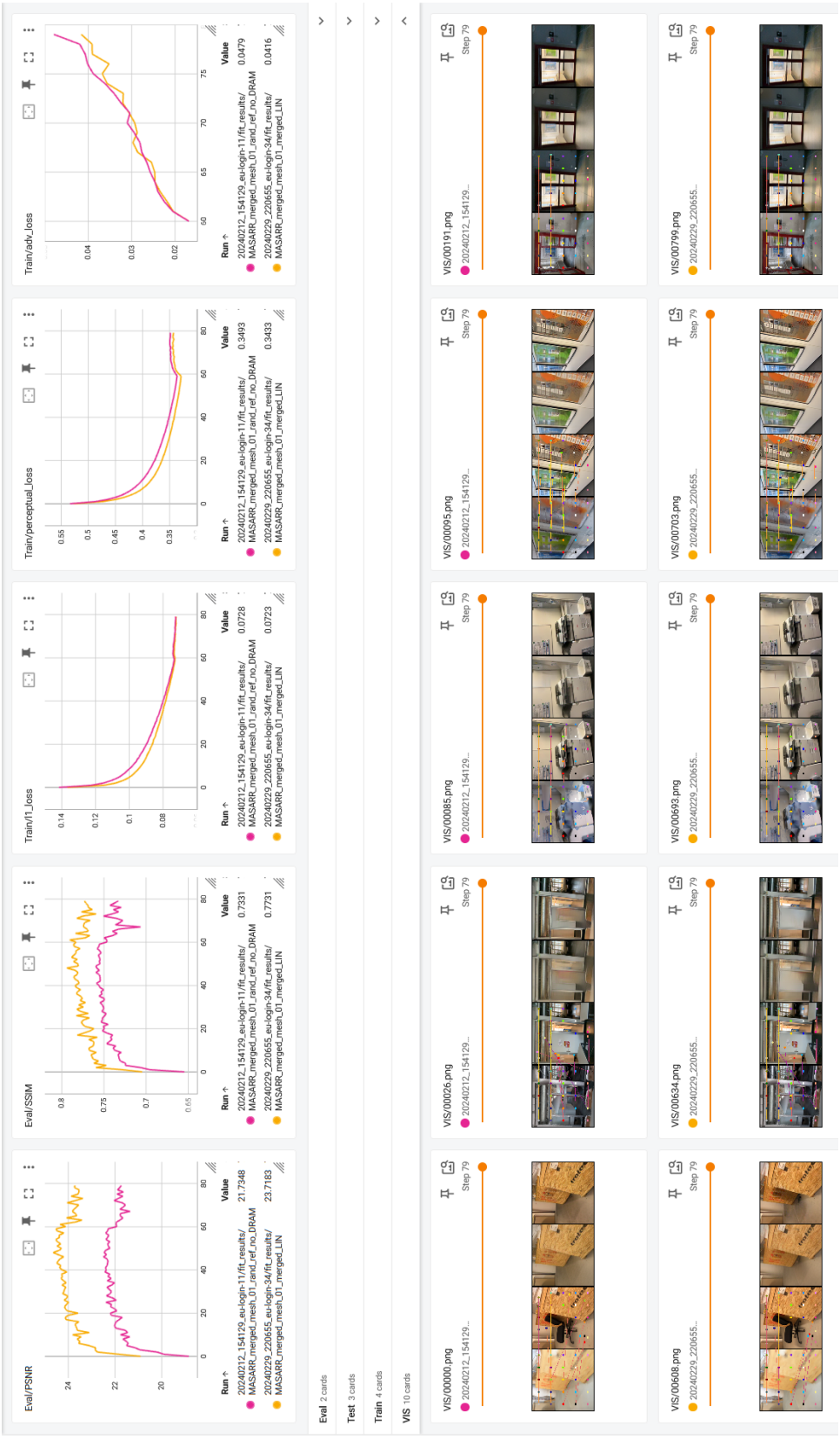


Figure 3.9: Tensorboard. Visualization of the training process plotting the losses, evaluation metrics and images on an epoch level.

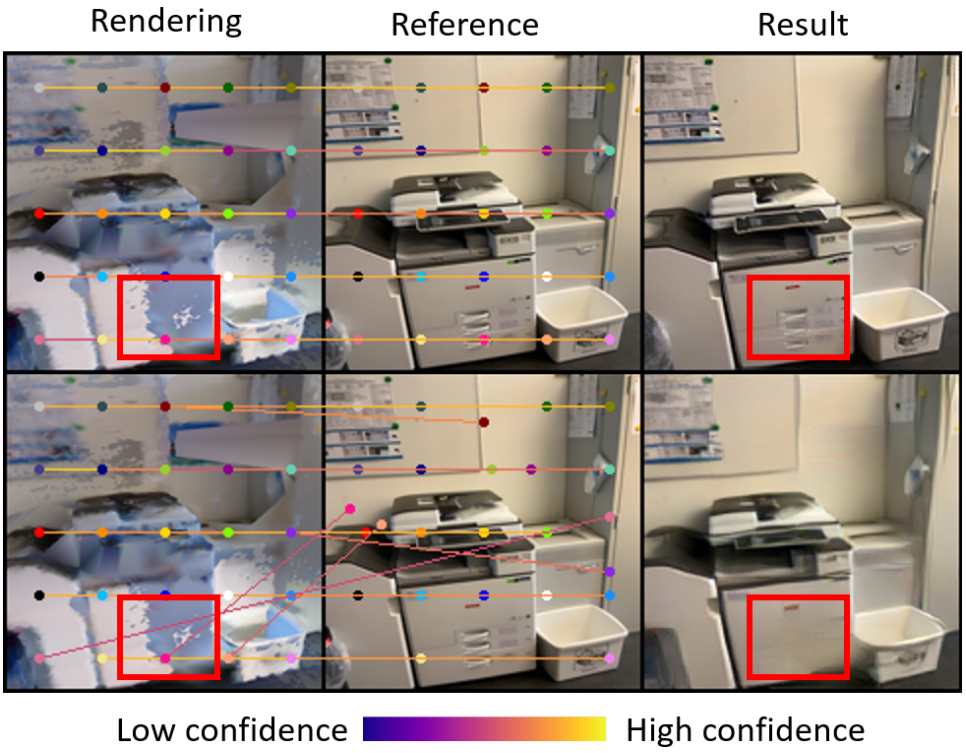


Figure 3.10: Feature matching visualization. The dots show the matches of the coarse blocks from the MEM. The lines are colored with the matching confidence, using the plasma color map. The visualization shows that missing details can be caused by block mismatches.

3.6 Visualizations

To better understand the learning process of MASA-SR [28] and the effect of its core modules, we build several visualizations. We plot the training progress using tensorboard, visualize the feature matching and compare the results using diagonal swipe animations and grid overlay plots.

3.6.1 Tensorboard

We setup a tensorboard environment to visualize the training process. Also, we fix a hand picked selection of evaluation images to compare training sessions with different models. Additionally, we log all experiments noting the scores and visual results on the evaluation images. Fig. 3.9 shows an overview over the tensorboard environment. The logged values are the reconstruction, perceptual and adversarial training losses and the Peak Signal Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) evaluation metrics.

3.6.2 Feature matching

The MEM inside the MASA-SR [28] pipeline is used to find correspondence in the reference. To visualize this process, we mark the matched locations of the coarse matching between rendering and reference. The confidence of the matches is visualized by coloring the connecting line with the plasma color-map. Blue values denote low and yellow values high confidence. Fig. 3.10 shows a relationship between

Figure 3.11: Diagonal swipe animations. To make sure the viewpoint is not changed, animations that perform diagonal swipes over the images can be inspected.

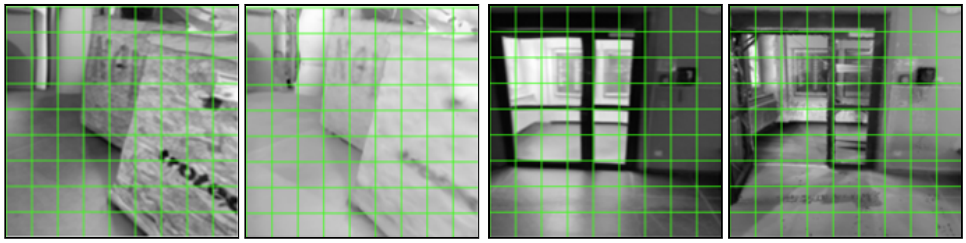


Figure 3.12: Grid overlay. Visualization to compare the result with the GT in detail.

the quality of the enhanced rendering and the matching confidence. If the matching fails, no details are transferred from the reference. This visualization enables us to better understand the effect of architectural changes to the pipeline, where we can now explain why certain changes end up in worse or better quality results. For example, this explains why for the same object some parts can be very detailed while neighboring parts lack details. If the coarse matching is not successful, then also the dense matching can have no success. This then leads to the network ignoring the reference and enhancing the image on its own.

3.6.3 Diagonal swipe & grid overlay

We want to make sure that the model keeps the viewpoint similar to the rendering. To this end, diagonal swipe Fig. 3.11 and grid overlay Fig. 3.12 visualizations are added to make sure of this with manual sanity checks.

3.7 Migration to PyTorch Lightning

With all visualizations in place, the next step is to modify the RefSR architecture. A first step is to adapt the model to take as input a high-res. rendering instead of a low-res. image. To be able to add architectural changes to MASA-SR [28], we need to be able to build several architectures within one project. Ideally, all of those can be run using config scripts. Even tough MASA-SR uses a decent codebase with recent package versions, it is designed to build RefSR models. Assumptions about the spatial resolution of the input are scattered throughout the project, making it very time consuming and error-prone to adapt it to a high-res. input. The process would also lead to a large amount of code duplication because the disentangling of the resolution assumption. Instead, we use the PyTorch Lightning [57] framework. This has the advantage that the code can be written hardware agnostic, removing all the existing shifting of tensors from and to devices. The engineering code, such as for the trainer or for multi GPU support is hidden, well maintained and

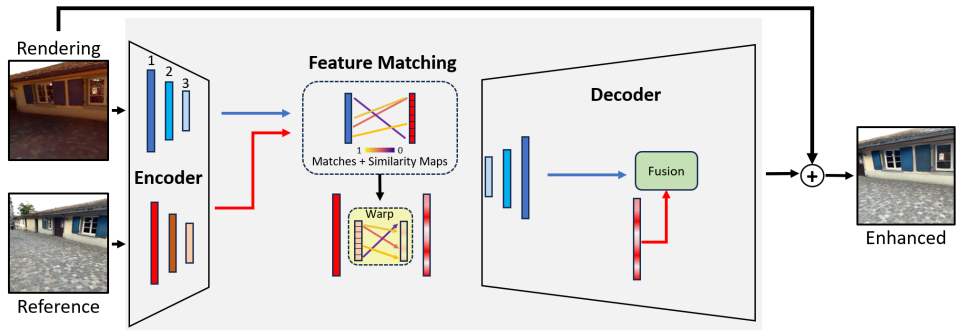


Figure 3.13: Pipeline of MASA-RE Removing all content related to low-res. inputs from MASA-SR [28]

abstracted away. This allows for focusing on the actual model development. The framework is also fully modular, which allows for dividing the MASA-SR codebase into modules and swapping them out with new ones in the model development process. While the use of such a high level framework has many advantages, it also comes with certain limitations. The fact that all engineering code is abstracted away can lead to situations where the provided abstractions are not sufficient to the current need. In such a case, those functionalities have to be overwritten or extended leading to more complicated code then when using pure PyTorch [58]. An example of such a situation is training Generative Adversarial Networks [59] that use non-standard optimizer behaviour, where first the discriminator takes some steps and then the generator. Because errors caused by such behavior are thrown within the framework, those are much more time consuming to debug then when using a custom pure PyTorch training loop. Knowing the limitations, we conclude that the advantages out-weight the shortcomings for this specific project. After the migration, our code-base is organized in high level encoder, decoder, attention modules. The assembling of the modules happens on the level of a config file, which makes it extremely fast to swap different modules or to select different hyper parameters by just changing one line in a config file and no code. This way, we have for the MASA-SR architecture one config file and for the following experiments we create others. This allows to maintain the architectures next to each other while minimizing code that deals with the different input resolutions.

3.8 MASA-RE

Using the modular code base, we iterate on the first model by removing all architecture related to low-res. inputs. We call the new model MASA-RE for Rendering Enhancement. The pipeline is shown in Fig. 3.13. Similar to MASA-SR [28], the level 1 features of rendering and reference are used for feature matching. However, only the level 1 features from the reference are warped and fused to the rendering features. The performance of the new model is worse because many blocks were removed, some of which were also important for the rendering enhancement task. To understand how the model can be improved we analyze the impact of the loss functions, data-augmentation strategies and alternative encoder and decoder modules.

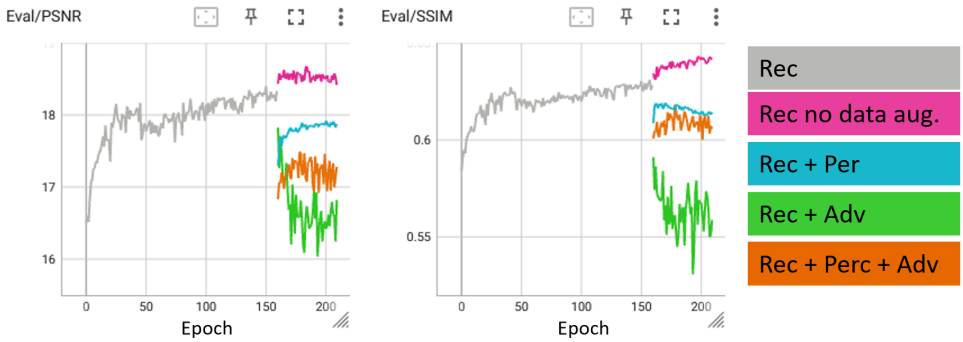


Figure 3.14: Loss Analysis Fine-tuning the model with different loss functions. The best results can be seen using the Reconstruction loss (Rec) without using other losses or data-augmentation. Adding the Perceptual loss (Per) decreases the scores and so does the Adversarial loss (Adv).

3.8.1 Loss analysis

We analyze the impact of the different loss functions on the result of the model, assessing whether they are suitable for the task of rendering enhancement. MASA-SR [28] uses three loss functions: The reconstruction, perceptual and adversarial loss. The loss is calculated between the high-res. Ground Truth image $\mathbf{I}_{GT}$ and the Super Resolved output image $\mathbf{I}_{SR}$.

$$\mathcal{L} = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{per}}\mathcal{L}_{\text{per}} + \lambda_{\text{adv}}\mathcal{L}_{\text{adv}} \tag{3.5}$$

$$\mathcal{L}_{\text{rec}} = \|\mathbf{I}_{\text{GT}} - \mathbf{I}_{\text{SR}}\|_1 \tag{3.6}$$

$$\mathcal{L}_{\text{per}} = \|\phi_i(\mathbf{I}_{\text{GT}}) - \phi_i(\mathbf{I}_{\text{SR}})\|_2 \tag{3.7}$$

$$\mathcal{L}_{\text{disc}} = -\mathbb{E}_{\mathbf{I}_{\text{GT}}}[\log(D(\mathbf{I}_{\text{GT}}, \mathbf{I}_{\text{SR}}))] - \mathbb{E}_{\mathbf{I}_{\text{SR}}}[\log(1 - D(\mathbf{I}_{\text{SR}}, \mathbf{I}_{\text{GT}}))] \tag{3.8}$$

$$\mathcal{L}_{\text{adv}} = -\mathbb{E}_{\mathbf{I}_{\text{GT}}}[\log(1 - D(\mathbf{I}_{\text{GT}}, \mathbf{I}_{\text{SR}})) - \mathbb{E}_{\mathbf{I}_{\text{SR}}}[\log(D(\mathbf{I}_{\text{SR}}, \mathbf{I}_{\text{GT}}))] \tag{3.9}$$

The reconstruction loss calculates the ℓ_1 norm between the enhanced rendering and GT. The loss ensures that the result is on a per pixel basis close to the GT. The perceptual loss uses the i-th layer of VGG19 [48] features ϕ_i to guide the images to be semantically similar. MASA-SR uses the layer `conv5_4`. The adversarial loss improves the visual quality of the image by training a discriminator. Fig. 3.14 shows the result of fine-tuning the MASA-RE model using different loss configurations. We notice that both the perceptual and adversarial loss have a negative impact on the scores. While for the adversarial loss this can be explained because the loss aims for visual quality instead of reconstruction accuracy, the perceptual loss should not have this effect, hinting that an adaption of the loss can have a beneficial impact on the training process. With a perceptual loss that does not decrease the scores, the training process could also be adapted to train with the perceptual loss from the beginning and fine-tune only with the adversarial loss. When looking at the loss plots, we notice high fluctuations. At the beginning of the learning process this is expected, but near the end when the loss converges, the changes should be more subtle. The reason for this behavior is the learning rate scheduler which is configured to only converge after 500 epochs. Changing this to the current number of epochs leads to smaller changes in the fine-tuning process. The result is an image with higher reconstruction accuracy and more subtle impact of the adversarial loss.

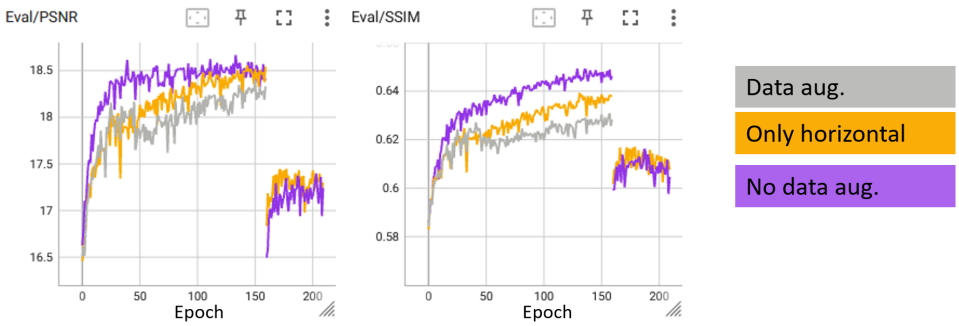


Figure 3.15: Analysis of the data-augmentation strategies. The model without data-augmentation performs the best, hinting that the rotations and flips, even if they are only performed horizontally, are not beneficial for the task of rendering enhancement.

3.8.2 Data-augmentation analysis

The data-augmentation strategy used by MASA-SR [28] focuses randomly rotating and flipping the images as well as on random contrast and brightness changes. Those flips can be both horizontally and vertically, leading to the image being upside down. We train the model three times: once using all data-augmentations, once without any and once where we only remove the vertical flips. Fig. 3.15 shows that the model trained without data-augmentation performs the best. We conclude that while the data-augmentations are beneficial for the RefSR task, for the task of novel view enhancement we need to find different, more task specific strategies.

3.8.3 Architecture experiments

We experiment with different architectural changes to improve the model. In this section we give an overview over the performed experiments. Because from Fig. 3.10 we learned that the matching of the features has a substantial impact on the final result, our idea is to make this matching more robust by using different types of features. Fig. 3.16 shows an overview over the architectures of different encoders we tried.

Deeper Feature Channels

Some RefSR methods [15, 21] use pre-trained features instead of training them end-to-end like MASA-SR [28]. Those features can leverage that they are trained on a much larger dataset and generalize better. Typically, VGG features are used for this purpose. The MASA-RE model supports only feature channels of dimension 64 for all encoder levels. We adapt both encoder, feature matcher and decoder to support features with 64, 128 and 256 channels making it compatible with common pre-trained feature extractors. We also train the model end-to-end on those deeper features which leads to similar results as with the original encoder that has fixed 64 feature channels.

VGG Encoder

Using the compatibility with deeper feature channels, we swap the encoder against an encoder using pre-trained VGG16 [48] features. We see the effect of those features already at the beginning of the training. While the end-to-end trained features starting from randomly initialized weights fail to establish any matches, the VGG16

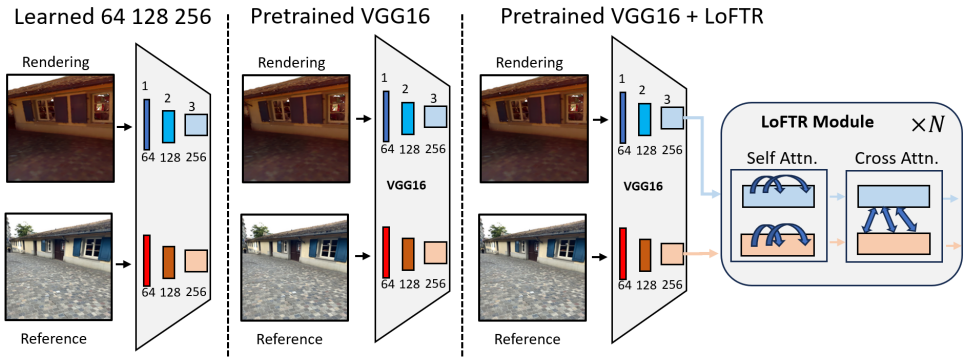


Figure 3.16: Architecture of alternative encoders. We experiment with different encoders: an end-to-end trained encoder with larger feature channels, an encoder using pre-trained VGG16 features and an encoder that adds an additional LoFTR [60] module consisting of alternating self and cross attention layers to fine-tune the features for improved matching.

features successfully establish many correspondences. Despite this early advantage, only after couple epochs the matches of the end-end trained model are already better. The VGG16 features are frozen and do not adapt to the current task. We conclude that we need to fine-tune or process the features such that they can specialize on the rendering enhancement task.

Transformers and LoFTR

There is a whole research field focused on feature matching. Because feature matching is also in essence what we do in the first part of the MEM module, the idea comes up to use modules of a modern feature matching pipeline to improve our model. LoFTR [60] is a detector free local feature matcher that uses transformers. Specifically, we use the LoFTR module which consists of alternating self and cross attention layers [61] shown in Fig. 3.16. We experiment with fixed, learnable and rotational [62] position encodings. Now the matches also change in the training process for the pre-trained features. We not that the performance is worse than for the end-to-end trained model while the training time significantly increased.

DINOv2 Features

DINOv2 [63] features could be an alternative to the use of VGG features because they are trained without supervision on a large dataset of 142 million images and show promising results on downstream tasks. Training on those features shows an interesting effect: The decoder converges to a trivial state and just returns an image full of zeros. We experiment with different decoders and also try to over-fit to just one image. Unfortunately all experiments end with the same result: a black image. This leaves us to explore further options to improve the architecture.

Iterative training

Having tried several alternative encoders, the result is still the same. Especially in the presence of larger artifacts, the matching is likely to fail which leads to a lack of detail in the affected image regions. We found a solution for this by implementing an iterative process. Instead of applying our model once, we apply it over several iterations. This has the effect that while in a first iteration the matching might

fail, the model can still remove the artifacts and enhance the image. As a result, subsequent iterations have better chances for finding successful matches.

Matching on level 3 features

In the MASA-RE model shown in Fig. 3.13, we use the level 1 features for matching similar to MASA-SR [28]. This comes from the resolution difference of the RefSR inputs, where the level 3 features are a scale factor of 16 times smaller in the spatial resolution than the high-res. reference features. For our model, the inputs are of similar spatial resolution. We therefore change the model to use deeper level 3 features, which exhibit increased robustness. Having matches on the level 3 features, opens also the door to warp and fuse the level 3, 2 and 1 reference features in a similar way as MASA-SR. We indeed observe increased performance for the matching process and on the overall visual result quality.

Adaption to different resolutions

The current model works best on 160 x 160 images. For other resolutions the matching process is not successful. We experiment with several options to increase the supported resolution. For one, in the current architecture the block and patch sizes are fixed. We adapt the architecture such that those can be chosen dynamically based on the input size. We change to a deeper encoder and fine-tune the network on higher resolution images. This requires a lot of computational time where the training of our model increases from 24 hours to over one week. We detect the bottleneck in the MEM module, but even after one week, the matching shows no improvement. Furthermore, the discriminator trained for the adversarial loss is specifically targeted to images with resolution of 160 and would have to be adapted for fine-tuning using this loss. We suspect that switching to a more involved matching pipeline and eventually replacing the MEM might allow the model to also predict higher resolution images. In the meantime, we use an alternative approach for higher resolution images, where we first down-scale the images such that the shorter edge matches the resolution of 160, do inference with our model and upscale the result using the super-resolution method Real-ERSGAN [64] to recover the original resolution. While it works and some results look good, this approach is far from optimal because both our method and Real-ERSGAN introduce artifacts which are amplified in the process. Also Real-ERSGAN only scales the images up to a factor of times 4. Everything higher is bilinear interpolated and lacks details.

3.9 MaRINeR

Having experimented with many possible architecture changes, loss functions and data-augmentation strategies we create a new architecture, **MaRINeR**, incorporating the conclusions of those experiments. The new pipeline includes a different version of the perceptual loss, matches on level 3 features and fuses warped reference features of all levels. To further increase the robustness, the result is refined in an iterative fashion. In the training process we use two new data-augmentation strategies specifically target for the task of novel-view enhancement.

Chapter 4

Method

The **MaRINeR** pipeline, illustrated in Fig. 4.1, takes as input a rendering with noisy appearance and geometry as well as a nearby reference and outputs an enhanced version of the rendering by transferring relevant information from the reference. The pipeline starts by densely extracting features at multiple levels from both input images with a shared convolutional encoder. Next, the deepest features extracted from the rendering are matched to those of the reference to retrieve similar content. These matches are then used to warp the reference features at different levels. The warped reference features are fused with those of the rendering in the decoder. Given the severe artifacts sometimes present in novel views, we employ an iterative refinement approach that repeats the process by replacing the input rendering with the enhanced output of the previous iteration. We start from the MASA-SR RefSR [28] pipeline and implement several changes in architecture, loss function as well as data augmentation to make it amenable to the novel task of reference-based rendering enhancement.

4.1 Encoder

As mentioned above, we use a shared convolutional encoder to extract features at multiple levels from both the rendered image I and the reference R, for simplicity assumed both of size $H \times W$. We use three levels, each halving the resolution of the previous one, yielding two sets of dense tensors: $\{\mathcal{F}_1^I \in \mathbb{R}^{H \times W \times F_1}, \mathcal{F}_2^I \in$

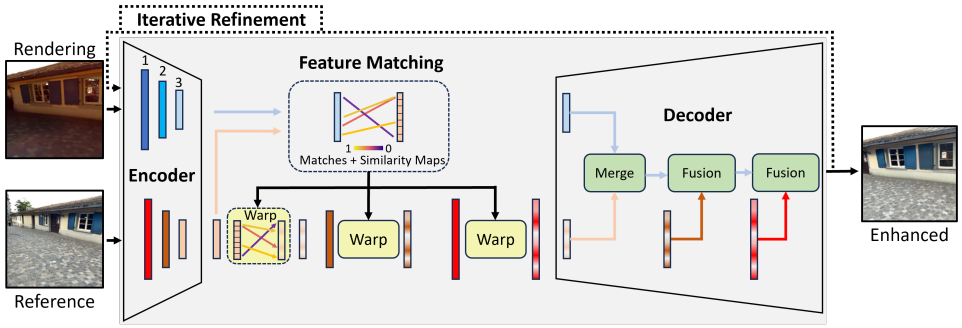


Figure 4.1: MaRINeR architecture. The learned features of the encoder are used to for correspondence matching and warping of the reference features. They are fused with the rendering features to create a enhanced rendering, which is iteratively refined.

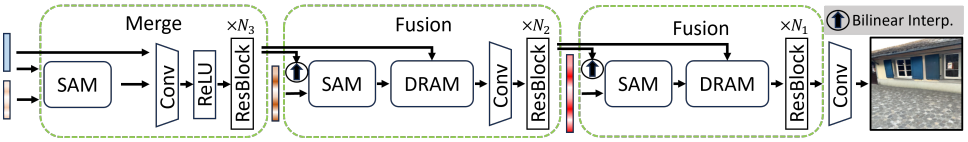


Figure 4.2: Architecture of the decoder. We fuse the rendering and warped reference features using SAM [28], DRAM [28] and residual blocks [32].

$\mathbb{R}^{H/2\times W/2\times F_2}, \mathcal{F}_3^I \in \mathbb{R}^{H/4\times W/4\times F_3}\}$ and $\{\mathcal{F}_1^R, \mathcal{F}_2^R, \mathcal{F}_3^R\}$ for the rendering and the reference, respectively, where F_i is the number of channels of the output of level i. These features will next be used to find corresponding patches between the rendering and the reference in a coarse-to-fine fashion.

4.2 Feature matching

We use the MEM from MASA-SR [28] to match the deepest features of both input images, $\mathcal{F}_3^R$ and $\mathcal{F}_3^I$, using cosine similarity. The MEM performs matching first on a coarse grid with a stride and then densely within a fixed-size window around the resulting matches. This step yields a mapping m of indices from the level 3 features of the rendering to those of the reference and associated matching scores s:

$$m_{I\to R} : (x, y) \in \mathcal{F}_3^I \to \{(u, v) \in \mathcal{F}_3^R, s \in \mathbb{R}\} \ . \tag{4.1}$$

This mapping is used to warp and weight the reference features at each of the three levels i, where blocks of features with size relative to the spatial resolution of the current level are cropped and moved together resulting in warped feature maps $\{\mathcal{F}_1^{R\to I}, \mathcal{F}_2^{R\to I}, \mathcal{F}_3^{R\to I}\}$. In contrast to RefSR methods which have an input with lower resolution and thus need to perform the matching on the $\mathcal{F}_1$ features of the low-res. input and down-scaled reference [28], we use deeper features, allowing us to leverage the increased robustness to find better quality matches. Weighting the warped features based on the matching scores reduces the impact of features with low confidence matches. This enables the model to only use the reference features if they have a confident match and otherwise use the rendering features when fusing them in the decoder.

4.3 Decoder

Using the deepest features of the rendering $\mathcal{F}_3^I$ and the warped reference features $\mathcal{F}_3^{R\to I}$, $\mathcal{F}_2^{R\to I}$ and $\mathcal{F}_1^{R\to I}$, we fuse them using SAM [28], DRAM [28] and residual blocks [32], as shown in Fig. 4.2. SAM learns to remap the distribution of the reference features to the one of the rendering features. DRAM fuses features of different spatial resolution aiming to refine and aggregate the details of both branches and up-sample the low-res. features with a transposed convolution. The decoder procedure can be summarized as follows:

$$\begin{aligned}
\mathcal{O}_3 &= \mathrm{P}_3(\mathrm{SAM}(\mathcal{F}_3^I, F_3^{R\to I}) \oplus \mathcal{F}_3^I) \\
\mathcal{O}_2 &= \mathrm{P}_2(\mathrm{DRAM}(\mathrm{SAM}(\mathcal{O}_3^{\uparrow}, \mathcal{F}_2^{R\to I}), \mathcal{O}_3)) \\
\mathcal{O}_1 &= \mathrm{P}_1(\mathrm{DRAM}(\mathrm{SAM}(\mathcal{O}_2^{\uparrow}, \mathcal{F}_1^{R\to I}), \mathcal{O}_2))
\end{aligned} \tag{4.2}$$

where P_i stands for processing the features using a convolution and N_i residual blocks [32], $\oplus$ for a convolution to merge the features and $\uparrow$ for bilinear interpolation. We merge the level 3 rendering features $\mathcal{F}_3^I$ with the warped level 3 reference

features $\mathcal{F}_3^{R\to I}$ by concatenating and processing them using a convolution, leveraging that the rendering and warped reference features are of similar spatial resolution. The SAM aligns the rendering and warped reference feature distributions such that the DRAM can successfully merge the features. The output image is then created from O_1 using a convolution to reduce the feature dimension. In contrast, RefSR methods such as MASA-SR deal with features with a different spatial resolution that can not directly be merged. MASA-SR first fuses the low resolution features of level 3 to 1 together before merging the reference features. For the task of RefSR this is beneficial because the result is encouraged to be structurally similar to the input with only additional details from the reference. For the task of rendering enhancement where the rendering can contain structures that come from artifacts, merging the features of similar spatial resolution enables the model to also take structural information from the reference. This is beneficial to remove rendering artifacts or fill in missing image parts caused by gaps in the source 3D reconstruction.

4.4 Iterative refinement

Since the gap between the rendering and the reference can be large due to rendering artifacts that occlude the underlying geometry, we found it beneficial to apply the model several times in an iterative fashion. The first iteration can thus focus on removing artifacts and enhancing the image. The following iterations are then more successful in establishing correspondences and transferring the missing details to the enhanced rendering. To this end, we supervise the model after each iteration, thus obtaining a more general model that can deal with a wide variety of rendering qualities.

4.5 Loss function

Our goal is to preserve the spatial information of the rendering while removing artifacts, adding details from the reference, and producing a visually pleasing result. To this end, we combine a reconstruction loss, perceptual loss, and adversarial loss with associated weights λ_{rec}, λ_{per}, and λ_{adv}, written as:

$$\mathcal{L} = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{per}}\mathcal{L}_{\text{per}} + \lambda_{\text{adv}}\mathcal{L}_{\text{adv}} \ . \tag{4.3}$$

4.5.1 Reconstruction loss

The enhanced rendering I_{ER} should be close to the GT image taken at the same pose as the rendering by using the information present in the close-by reference. We adopt the following reconstruction loss:

$$\mathcal{L}_{\text{rec}} = \|\mathbf{I}_{\text{GT}} - \mathbf{I}_{\text{ER}}\|_1 \ , \tag{4.4}$$

where $\|\cdot\|_1$ is the ℓ_1 norm.

4.5.2 Perceptual loss

The perceptual loss is widely used by RefSR models [28, 30, 34, 27] to enhance the visual quality of the result by guiding the resulting image to be more semantically similar to the GT. This loss is formulated as:

$$\mathcal{L}_{\text{per}} = \frac{1}{3}\sum_{i=1}^{3}\|\phi_i(\mathbf{I}_{\text{GT}}) - \phi_i(\mathbf{I}_{\text{ER}})\|_2^2 \ , \tag{4.5}$$

where $\phi_i(\cdot)$ denotes the outputs of ImageNet [65]-pretrained VGG16 [48] at layers `relu1_1`, `relu2_2` and `relu3_3`. Contrary to RefSR methods, we chose to use more shallow features [66] since the domain gap between rendering and reference leads to a mismatch between the deeper features and therefore causes increased artifact generation. We show qualitative results in the ablation study Chapter 7.

4.5.3 Adversarial loss

The drawback of the perceptual loss is that it tends to generate grid like artifacts [67]. The adversarial loss [68] helps to remove those artifacts and generate visually pleasing images:

$$\mathcal{L}_{\text{disc}} = -\mathbb{E}_{\mathbf{I}_{\text{GT}}}[\log(D(\mathbf{I}_{\text{GT}}, \mathbf{I}_{\text{ER}}))] - \mathbb{E}_{\mathbf{I}_{\text{ER}}}[\log(1 - D(\mathbf{I}_{\text{ER}}, \mathbf{I}_{\text{GT}}))] \ , \tag{4.6}$$

$$\mathcal{L}_{\text{adv}} = -\mathbb{E}_{\mathbf{I}_{\text{GT}}}[\log(1 - D(\mathbf{I}_{\text{GT}}, \mathbf{I}_{\text{ER}})) - \mathbb{E}_{\mathbf{I}_{\text{ER}}}[\log(D(\mathbf{I}_{\text{ER}}, \mathbf{I}_{\text{GT}}))] \ , \tag{4.7}$$

where $\mathcal{L}_{\text{disc}}$ represents the discriminator loss and L_{adv} is the generator loss. We adopted the Relativistic GAN [69] formulation following MASA-SR [28].

Chapter 5

Experiments

5.1 Implementation details

The encoder consists of 3 levels where each level is connected to the next one and consists of 1 convolutional layer and 4 residual blocks [32]. In our experiments, we keep the number of feature channels fixed to $F_i = 64$ for all levels. We train on 160x160 images following the convention of recent RefSR methods [28, 30, 34]. In the decoder we use $N_3 = 12$, $N_2 = 8$ and $N_1 = 4$ residual blocks in the merge and fusion layers. We train our model for 60 epochs using only the reconstruction and perceptual loss and fine-tune the model for 20 epochs using additionally the adversarial loss. In our experiments, the weight coefficients λ_{rec}, λ_{per} and λ_{adv} are 1, 1 and 0.001. For training we use a NVIDIA Tesla A100 40GB GPU with a batch size of 9 for 37 hours. We use two data-augmentation strategies specifically targeted to our task. For generalization to a wide range of rendering qualities, we augment the training data with renderings from down-sampled versions of the meshes containing only 10% of the original triangles. To ensure that the model removes artifacts and enhances the rendering even if the reference image is far away or has little content in common, we pick the training reference images randomly from within a 5s temporal window in the sequence.

5.2 Evaluation metrics

Because we start from a RefSR method, we use the same metrics for evaluation, notably: Peak Signal Noise Ratio (PSNR) ↑ and Structural Similarity Index Measure (SSIM) ↑. To follow the convention [28, 30], all PSNR and SSIM results are evaluated on the Y channel of the YCbCr color space. Because PSNR and SSIM can not determine visual quality we also report the Learned Perceptual Image Patch Similarity (LPIPS) ↓ [70] version 0.1 and the Edge-Restoration Quality Assessment (ERQA) ↑ [71]. LPIPS represents the visual quality with respect to the human perception. Because PSNR and SSIM do not align with human perception when it comes to value blurry images against images with details [70], we also use ERQA that measures how well a method performs at restoring edge details. The PSNR [72] is defined as

$$\begin{aligned} PSNR(\mathbf{I}_{GT}, \mathbf{I}_{ER}) &= 10\log_{10}\left(\frac{255^2}{MSE(\mathbf{I}_{GT}, \mathbf{I}_{ER})}\right) \\ MSE(\mathbf{I}_{GT}, \mathbf{I}_{ER}) &= \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}(\mathbf{I}_{GT}(i,j) - \mathbf{I}_{ER}(i,j))^2 \end{aligned} \tag{5.1}$$

where the MSE is the mean squared error. The SSIM [73] is calculated on two equally sized windows $x \subset \mathbf{I}_{GT}$ and $y \subset \mathbf{I}_{ER}$

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{5.2}$$

where $c_1 = (0.01 \cdot 255)^2$ and $c_2 = (0.03 \cdot 255)^2$. The formula is based on three components that measure the difference between x and y in terms of luminace, constrast and structure. The ERQA [71] finds edges in $\mathbf{I}_{GT}$ and $\mathbf{I}_{ER}$ using the Canny algorithm [74]. Those edges are compared using the $F1$ score

$$precision = \frac{TP}{TP + FP}, recall = \frac{TP}{TP + FN} \tag{5.3}$$

$$F1 = 2\frac{precision \cdot recall}{precision + recall} \tag{5.4}$$

where TP (True Positive) are the number of pixels detected as edge in both $\mathbf{I}_{GT}$ and $\mathbf{I}_{ER}$. FP (False Positive) is the number of pixels detected only in $\mathbf{I}_{ER}$, FN (False Negative) are pixels only detected in $\mathbf{I}_{GT}$. To account for networks that produce small edge shifts either globally over the entire image or locally, ERQA builds in compensations to match the pixels of those edges before calculating the $F1$ score. The LPIPS [70] uses deep neural networks as feature extractor and trains a similarity predictor network based on the feature difference of the images at several layers.

$$LPIPS(\mathbf{I}_{GT}, \mathbf{I}_{ER}) = \sum_l \mathcal{G}_l \left(\frac{1}{H_l W_l} \sum_i^{H_l} \sum_j^{W_l} \| w_l \odot (\phi_l(\mathbf{I}_{GT})_{i,j} - \phi_l(\mathbf{I}_{ER})_{i,j} \|_2^2 \right) \tag{5.5}$$

where ϕ_l denotes the output of layer l of the pretrained AlexNet [75]. LPIPS uses layers `conv_1` to `conv_5`. $\mathcal{G}_l$ is the trained prediction network for layer l, $\odot$ stands for scaling the activations channel-wise by a vector w_l.

5.3 Comparison with RefSR and ST methods

We conduct quantitative and qualitative comparisons between our method and existing RefSR and ST methods. The RefSR methods we compare with are MASA-SR [28] and DATSR [30]. The ST methods are the universal method Artflow [43], the photorealistic method WCT2 [49] and the semantic method NNST [45]. Tab. 5.1 shows the results of the quantitative comparison. The metrics are calculated between the GT and the enhanced rendering. The row *Render* is the baseline and shows the scores of the not enhanced rendering and the GT. In the column $\text{CAB}_{\text{Ref.=GT}}$ the GT is used as the reference showing the methods' performance when using the optimal reference image. **MaRINeR** successfully enhances the rendering leading to a significant improvement in all metrics. It also performs better at the task than existing RefSR and ST methods. Even though HGE is a novel scene, our model performs similarly well as on the CAB and LIN scenes, demonstrating that our model exhibits a strong generalization ability. Because in our dataset different objects can be present in rendering and GT, the enhanced rendering will not necessarily exactly look like the GT. This is also reflected in the scores, which are generally lower then when comparing RefSR methods where the GT and low-res. match content-wise.

Table 5.1: Quantitative evaluation. Our model enhances the rendering in common image quality metrics. It does so using the optimal reference Ref. = GT as an upper bound or using a close-by reference. It generalizes to the unseen HGE scene and performs better than existing RefSR (RSR) and Style Transfer (ST) methods.

	Method	$CAB_{Ref.=GT}$				CAB				LIN				HGE			
		PSNR	SSIM	ERQA	LPIPS	PSNR	SSIM	ERQA	LPIPS	PSNR	SSIM	ERQA	LPIPS	PSNR	SSIM	ERQA	LPIPS
	Render	15.60	0.559	0.564	0.380	15.60	0.559	0.564	0.380	14.39	0.529	0.549	0.392	15.84	0.575	0.619	0.364
RSR	MASA [28]	15.55	0.555	0.568	0.347	15.47	0.524	0.544	0.367	14.17	0.419	0.523	0.397	15.62	0.478	0.576	0.360
RSR	DATSR [30]	15.65	0.568	0.553	0.349	15.63	0.557	0.530	0.364	14.34	0.468	0.483	0.438	15.80	0.536	0.553	0.376
ST	Artflow [43]	16.30	0.472	0.533	0.334	15.39	0.414	0.489	0.393	17.00	0.503	0.622	0.321	16.97	0.500	0.602	0.341
ST	WCT2 [49]	16.48	0.559	0.569	0.357	16.08	0.554	0.567	0.367	17.44	0.569	0.565	0.332	17.52	0.589	0.623	0.324
ST	NNST [45]	18.52	0.643	0.620	0.303	16.33	0.559	0.566	0.370	18.53	0.568	0.661	0.315	18.48	0.591	0.646	0.315
RE	**MaRINeR**	**23.89**	**0.799**	**0.722**	**0.089**	**20.03**	**0.697**	**0.643**	**0.180**	**21.73**	**0.668**	**0.705**	**0.155**	**20.96**	**0.673**	**0.684**	**0.176**

5.4 Qualitative comparison

Fig. 5.1 shows a visual comparison with RefSR and ST methods. RefSR methods stay close to their low-res. input structure and color wise. They add details from the reference which are first transformed to the low-res. color distribution. Therefore they can not remove artifacts and the color distribution is not adapted to the one of the reference. Style-transfer methods on the other hand have a built in trade-off between content preservation and style transfer. Photo-realistic methods successfully adapt the color distribution of the reference while not changing the content of the rendering, which inadvertently also preserve the artifacts. Universal style transfer methods transfer both the color and content from the reference to the output so some artifacts can disappear. However, they do not distinguish between real content and artifacts and therefore introduce unrealistic distortions into the image. Semantic style transfer methods match between content and style and successfully transfer the style of matching objects. If no matches are found, then the methods also introduce distortions. Our model successfully transfers the colors, removes rendering artifacts while preserving the underlying content and fills in missing parts.

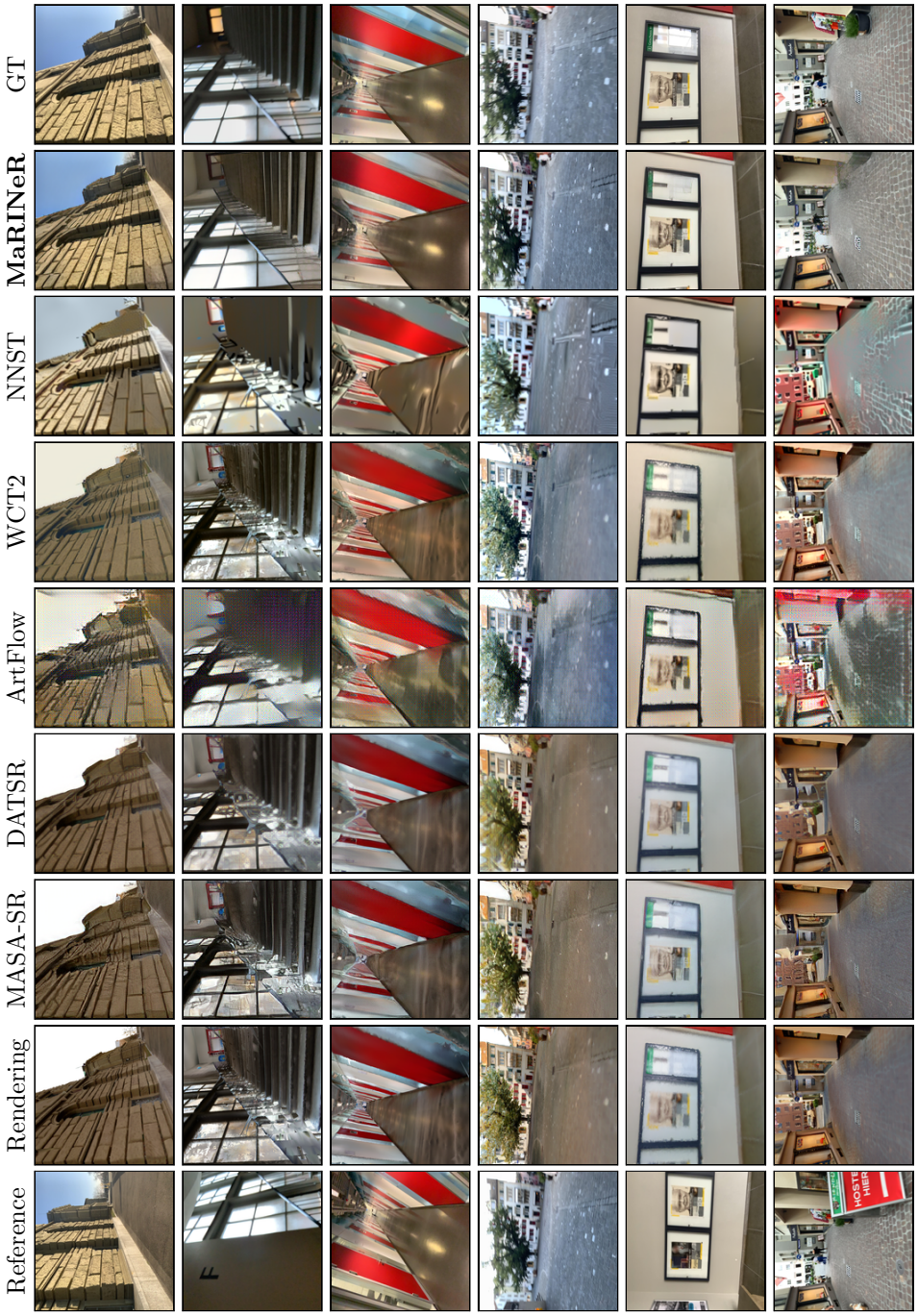


Figure 5.1: Qualitative comparison. Comparing the results of different methods MASA-SR [28], DATSR [30] (RefSR), ArtFlow [43], WCT2 [49] and NNST [45] (ST) with ours for the task of novel view enhancing.

Chapter 6

Further Analysis

To understand the capabilities of **MaRINeR**, we analyze its robustness to various challenging situations. Those include renderings of meshes with different resolution or created using different algorithms as well as renderings with large artifacts or missing image parts. In the end, we investigate if our model also works for high resolution renderings.

6.1 Rendering artifacts

We test our model on different images with challenging artifacts that cover large parts of the image. Fig. 6.1 a shows that our model manages to recover the sky, which was lost during creation of the 3D model. Fig. 6.1 b shows the models capability to recover large parts of the floor that was occluded by a floating triangle.

6.2 Mesh resolution

Fig. 6.1 e shows the robustness of the model to different mesh resolutions. We use renderings of a down-sampled mesh with 10 % size of the original mesh to make the model robust to different rendering qualities. As a result the model is able to enhance a wide range of renderings of different quality 3D reconstructions.

6.3 Device agnostic

The LaMAR [16] dataset also contains trajectories recorded with HoloLens 2 devices, which capture greyscale images. Fig. 6.1 d shows that also for greyscale renderings and references recorded with a different device our model performs similarly well.

6.4 Reference similarity

In some cases the reference image might not be similar to the rendering. Possible reasons are changed illumination or non overlapping views. Fig. 6.1 c shows that our model also handles very strong illumination changes, Fig. 6.1 f shows that even with a reference with no content in common our model still removes the rendering artifacts and enhances the image.

Figure 6.1: Robustness of MaRINeR. Our model recovers missing parts that appear due to rendering artifacts **a**, **b**. It adopts the illumination from the reference **c**, is device agnostic generalizing to gray-scale images **d**. The model enhances renderings of low triangle meshes **e** and also improves the rendering even if the reference has little content in common **f**. It can be applied to unseen scenes such as 12 Scenes [18] **g** or Aachen Day-Night [76] **h** without retraining.

Table 6.1: 12 Scenes evaluation. 12 Scenes [18] was captured with a higher frame rate which is reflected in the scores that are closer to the Ref.=GT case on LaMAR.

Method	12 Scenes			
	PSNR	SSIM	ERQA	LPIPS
Render	20.59	0.732	0.640	0.164
MaRINeR	**22.99**	**0.775**	**0.703**	**0.071**

Table 6.2: Evaluation on images of higher resolution. Renderings of higher resolutions are only marginally enhanced by **MaRINeR**. To improve the enhancement, we extend the pipeline with the super resolution method Real-ERSGAN [64]. As a result also higher resolution renderings can be enhanced.

Method	$\text{CAB512}_{\text{Ref.=GT}}$			
	PSNR	SSIM	ERQA	LPIPS
Render	15.27	0.649	0.287	0.544
MaRINeR	<u>17.91</u>	<u>0.709</u>	<u>0.398</u>	<u>0.350</u>
MaRINeR + Real-ERSGAN	**21.51**	**0.759**	**0.434**	**0.277**

6.5 Reconstruction method

Contrary to LaMAR [16] which uses a NavVis scanner running LiDAR-inertial SLAM followed by the Advancing Front [54] algorithm for meshing, we report results on the 12 Scenes [18] dataset which uses RGB-D SLAM on Kinect data and BundleFusion [77] in Tab. 6.1 and visually in Fig. 6.1 g. Furthermore, Fig. 6.1 h shows the results on the Aachen Day-Night [76] dataset which uses texture maps instead of vertex coloring to texture the 3D reconstruction.

6.6 Generalization to higher resolutions

MaRINeR is trained on 160x160 images. We evaluate the generalization to higher resolution images by enhancing 512x512 renderings of the CAB scene using the reference as GT. Tab. 6.2 shows that the enhancement of our model on 512 images is not as successful as on the 160 images. In order to use our model also for higher resolution images, we add an optional extension to the pipeline. Higher resolution images are down-scaled such that the smaller edge of the image is of size 160. Then we use **MaRINeR** to enhance the low-res. rendering. To scale the result back to the original resolution we use the super-resolution method Real-ERSGAN [64]. Fig. 6.2 shows that using this method, our model also enhances the high quality renderings.

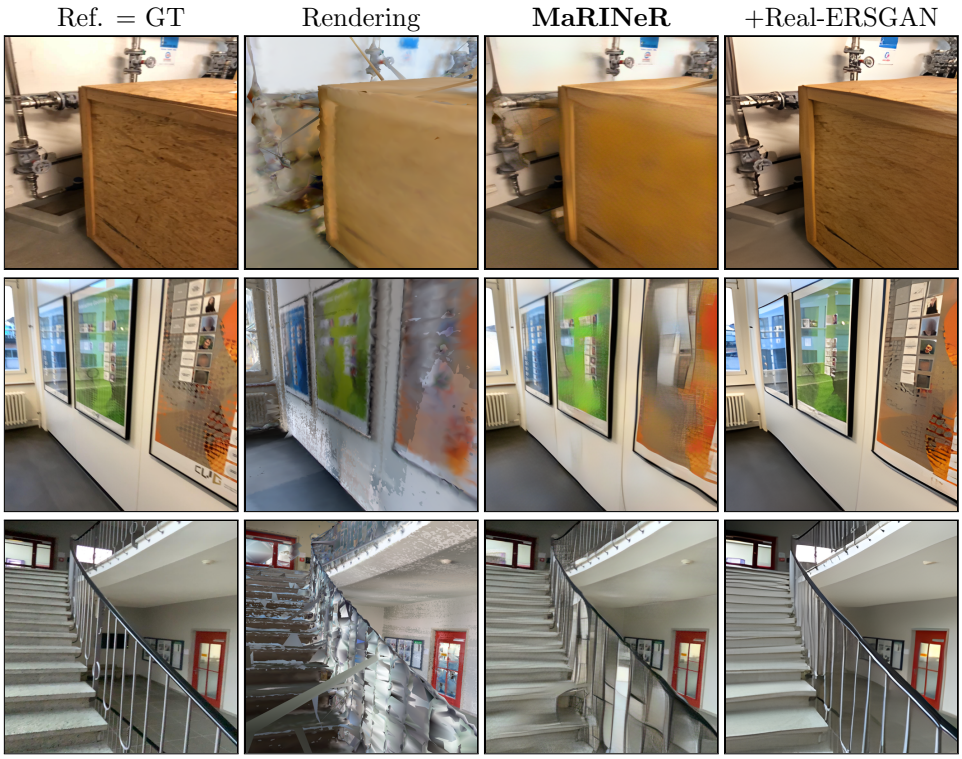


Figure 6.2: Inference on high resolution images. The **MaRINeR** model works best on resolutions in the order of 160. On images with higher resolution, here 512x512, the feature matching is not optimal. Down-scaling the renderings to 160 and predicting with our model and up-scaling again using the super resolution method Real-ERSGAN [64] allows to enhance renderings of arbitrary resolution.

Chapter 7

Ablation Study

We perform ablation studies to asses the effectiveness of our data augmentations and the loss functions for the given task. Further we compare the encoder and decoder to alternative versions confirming the respective architecture.

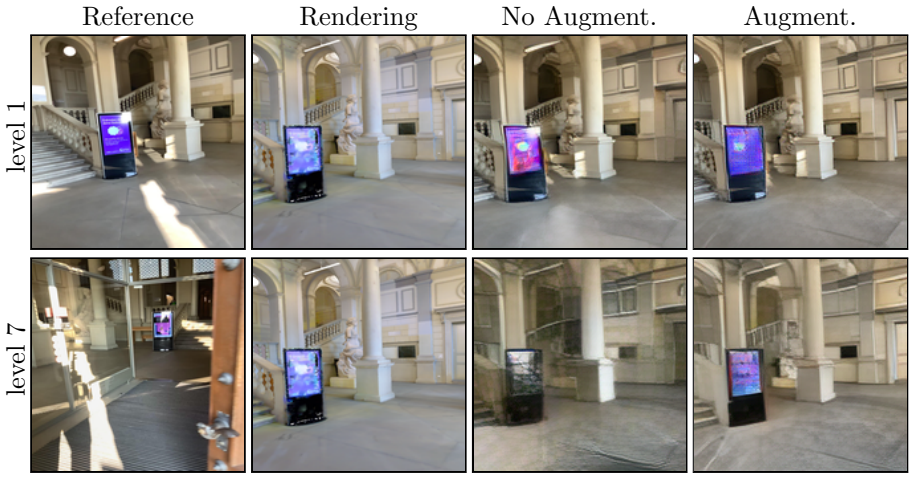


Figure 7.1: Reference level data augmentation. Impact of using random close-by images as reference instead of only the closest one. While the model performs similar for the close by references of level 1, the correspondence matching fails for further away references of level 7 leading to worse results.

7.1 Data augmentation

We first analyze the effectiveness of our data augmentation strategies. We augment the data in two ways: First we add renderings of a down-sampled mesh to the training data to make the model more robust against changes in the mesh quality and second we use a random close-by image as reference instead of the closest one. Tab. 7.1 left shows the model trained without the augmentation performs worse on meshes with different resolutions. While on the 100% mesh the PSNR/SSIM difference is 0.35 dB and 0.001, on the 10% mesh it increases to 1.12 dB and 0.024. Visually the effect can be seen in Fig. 7.2 where the model without augmentation fails to find correspondences in the rendering of the down-sampled mesh. Tab. 7.1 right shows that for the case without random reference augmentation, the performance is better when the reference is the GT. However as soon as the reference

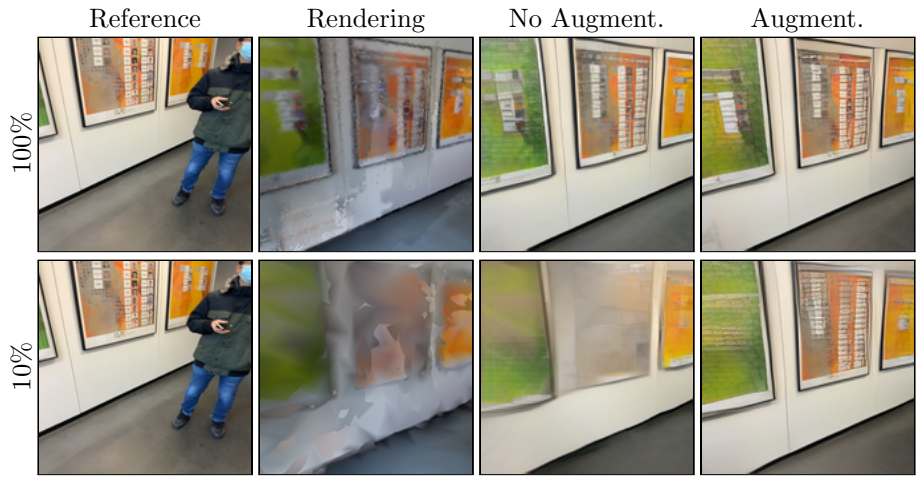


Figure 7.2: Mesh quality data augmentation Comparison of the model with and without augmenting the data with renderings from a down-sampled mesh. While for a mesh size of 100% the results are visually similar, for the mesh size of 10% the non augmented model fails to find correspondences and the result lacks the details from the reference.

Table 7.1: Robustness. Left – Mesh quality. Augmenting the data with renderings of a down-sampled mesh increases robustness to changes in the mesh resolution. **Right – Reference level.** A higher level indicates a larger temporal distance to the rendering within a sequence which generally correlates with less content in common.

Mesh size	CAB_{aug}		CAB	
	PSNR	SSIM	PSNR	SSIM
100%	**19.80**	**0.687**	19.45	0.686
75%	**19.46**	**0.680**	18.92	0.678
50%	**19.41**	**0.677**	18.83	0.673
25%	**19.33**	**0.669**	18.56	0.660
10%	**19.10**	**0.650**	17.98	0.626

Ref. Level	CAB_{aug}		CAB	
	PSNR	SSIM	PSNR	SSIM
0 = GT	22.91	0.777	**23.51**	**0.783**
1	**19.88**	**0.687**	19.85	0.681
2	**18.99**	**0.664**	18.48	0.644
5	**18.25**	**0.646**	17.13	0.607
8	**18.04**	**0.643**	16.79	0.600

level increases, the performance drops to a PSNR/SSIM difference of 1.25 dB and 0.043 compared to the augmented model. Fig. 7.1 shows the effect visually where without the augmentation, the model fails to match objects with a different scale.

7.2 Influence of the perceptual loss

Because the task of novel view enhancement is different from RefSR, we investigate the effectiveness of the commonly used perceptual loss on our task. Fig. 7.3 $\lambda_{per} = 0$ shows that without the perceptual loss, fine geometric structure like the texture of the box are not correctly transferred. Increasing the weight to $\lambda_{per} = 0.02$ and $\lambda_{per} = 0.1$, we observe increased texture details. A higher perceptual weight $\lambda_{per} = 0.5$ leads to grid like artifacts [67] which are more visible in image regions where the correspondence matching is less confident. The extreme case can be observed for $\lambda_{perMASA}$ using the same perceptual loss as MASA-SR [28]. Fig. 7.3 shows that $\lambda_{per} = 0.1$ increases the details optimally while introducing minimal artifacts which is also confirmed numerically in Fig. 7.4a.

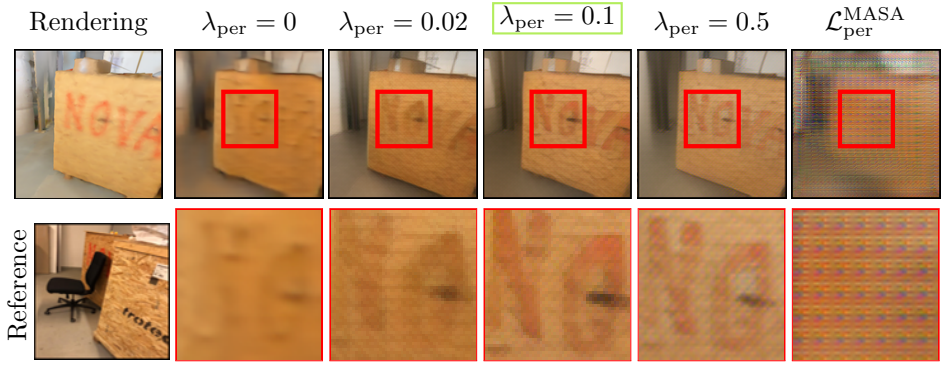
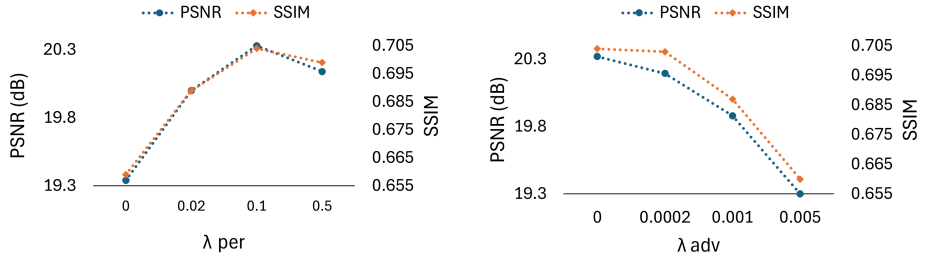


Figure 7.3: Impact of the perceptual loss. Increased weight enhances details and the visual quality but also introduces perceptual loss specific grid-like artifacts.



(a) Perceptual loss weight influence **(b)** Adversarial loss weight influence

Figure 7.4: Influence of the loss weights. The results of our experiments finding the optimal weights for a) the perceptual loss and b) the adversarial loss.

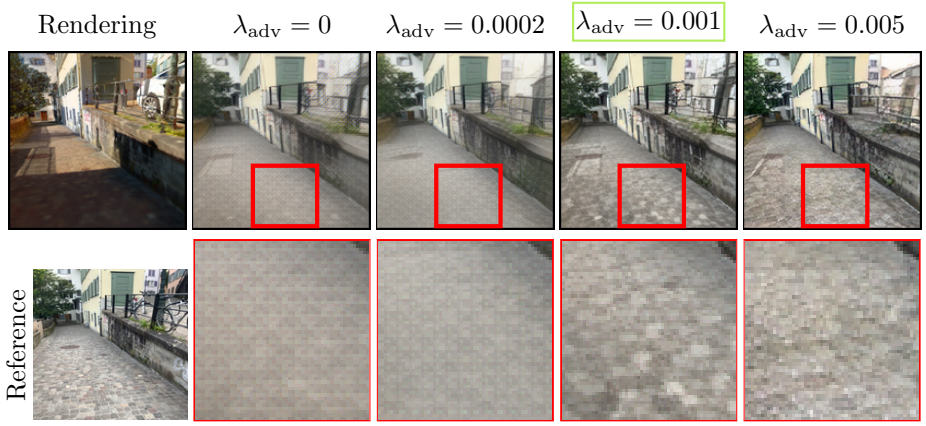
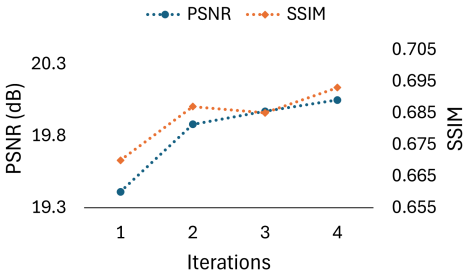


Figure 7.5: Impact of the adversarial loss. Increased weight removes the perceptual loss artifacts and keeps the underlying texture. Increasing the weight too much leads to the introduction of hallucinated details not present in the reference.

7.3 Influence of the adversarial loss

Using the perceptual loss can lead to grid-like artifacts [67]. To remove those and make the images more visually pleasing [28, 30] we use the adversarial loss. Fig. 7.5 shows the impact of different weights for the loss. $\lambda_{adv} = 0$ contains the artifacts from the perceptual loss. $\lambda_{adv} = 0.005$ removes those artifacts completely but



Iterations #	Runtime (ms)
1	49.2
2	66.3
3	88.7
4	110.5

Figure 7.6: Impact of the number of iterations. Increasing the number of iterations leads to better results. The largest improvement can be seen between 1 and 2 iterations. More iterations marginally improve the result while increasing the inference time linearly.

introduces high frequency details not present in the reference. Fig. 7.4b shows that also the scores decrease with higher adversarial loss weight. We found that with $\lambda_{adv} = 0.001$ the perceptual loss artifacts are removed while minimal new details are wrongly introduced.

7.4 Effect of iterative refinement

Fig. 7.7 and Fig. 7.6 show the effect of refining the rendering over multiple iterations. With one iteration the model can fail matching regions with large artifacts. Using two or more iterations, the first iteration can remove the artifacts such that the correspondence matching can succeed in the next iteration. We note the largest improvement when using 2 iterations. Using more iteration gives only small improvement but for each iteration the model requires more inference time.

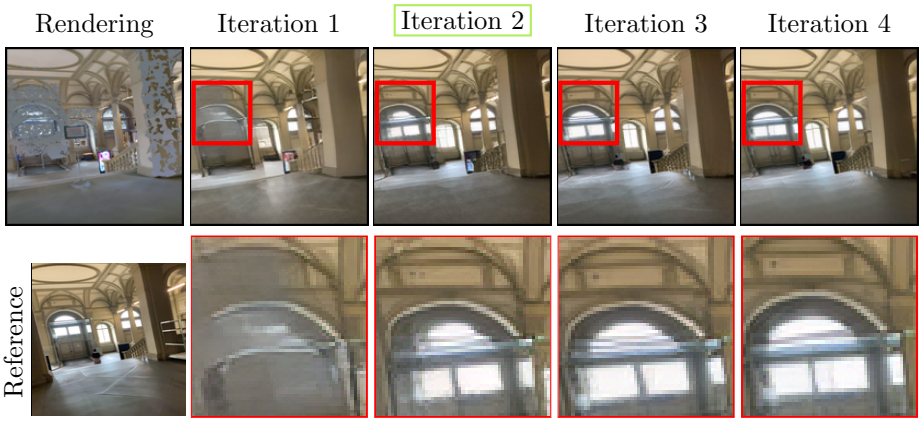


Figure 7.7: Visual improvement with iterative refinement. Refining the result over several iterations helps the correspondence matching in presence of large artifacts.

7.5 Encoder

An important part of the model performance is whether the matching between rendering and real image is successful. This matching is performed on the features

Table 7.2: Left - Encoders Results of using different encoders. **Right - Decoders** Results of replacing SAMs and DRAMs in the decoder.

Encoder	CAB	
	PSNR	SSIM
Learned 64	**19.88**	**0.687**
Learned 64 128 256	19.66	0.685
VGG	19.08	0.653
VGG + LoFTR	18.99	0.652

Decoder	CAB	
	PSNR	SSIM
SAM + DRAM	**19.88**	**0.687**
No SAM	19.73	0.685
No DRAM	19.71	0.676

from the encoder. MASA-SR [28] uses features trained end-to-end with the super resolution task which has the advantage that the features are hand tailored for the task. Another option is to use pre-trained features [15, 21]. If we use for example VGG features, we can leverage that those models were trained on a much larger dataset and the features potentially generalize better. The first encoder is trained end-to-end like ours but increases the feature dimension with each stage. The second one uses a pretrained VGG16 [48] encoder where we use the `relu1_1`, `relu2_2` and `relu3_3` features. Because VGG features were not trained to match between rendering and real images, we use a LoFTR module [60] with a rotational position encoding [62] to finetune and enhance the features in the third encoder. Tab. 7.2 left shows that our original encoder, where all stages have 64 feature channels, gives the best result.

7.6 Decoder

We show that the SAM and DRAM blocks are also applicable for the task of novel view enhancement. For this we train two models, where in the first one the decoder has no SAMs. In the second model, the DRAMs are replaced by simply concatenating the features and merging them using a convolution. Tab. 7.2 right shows that the scores are the best using both DRAMs and SAMs.

Chapter 8

Applications

Novel view enhancement can be applied to a variety of situations. We showcase the benefits of using our model for eliminating manual sanity checks, enhancing synthetic trajectories and as post-processing tool for renderings of NeRFs.

8.1 Validation of localization pseudo-ground-truth

A limitation of automatic pseudo-GT pipelines for localization is that they often require manual validation. For instance, the LaMAR [16] pipeline registers sequences of images recorded by AR devices, into a common 3D reconstruction based on a high quality LiDAR scanner. To check if the generated alignment is accurate, manual checks between renderings from mesh and input images are needed. To automate those, an option is to estimate a homography between the rendering of the scene at the estimated pose and the associated image of the input sequence. The homography should be identity if the localization of the pipeline was successful. However, estimating a homography between the rendered and real image is not accurate because of the domain gap. Using our method, we can enhance the renderings and improve the accuracy of the estimated homography. We use SuperPoint [55] for feature extraction and SuperGlue [56] for matching. Fig. 8.1 and Tab. 8.1 show that with our method we increase the number of matches and the inlier ratio supporting the homography. This leads to a more accurate homography estimation, characterized by the homography error [55] in the table which is optimally zero. Because of the increased accuracy of the homography we can therefore eliminate the manual sanity checks and replace them with an automated tool.

8.2 Enhancing synthetic trajectories

When creating large AR datasets, substantial human effort is needed to record AR devices trajectories. With recent advances in simulating natural human body

Table 8.1: Improved homography estimation. Using enhanced renderings improves the matching and homography estimation between the rendering and the raw image.

	# matches	inlier ratio	homography error
Rendering - Image	39.21	61.24%	4.86
Enhanced - Image	**58.89**	**78.16%**	**1.88**

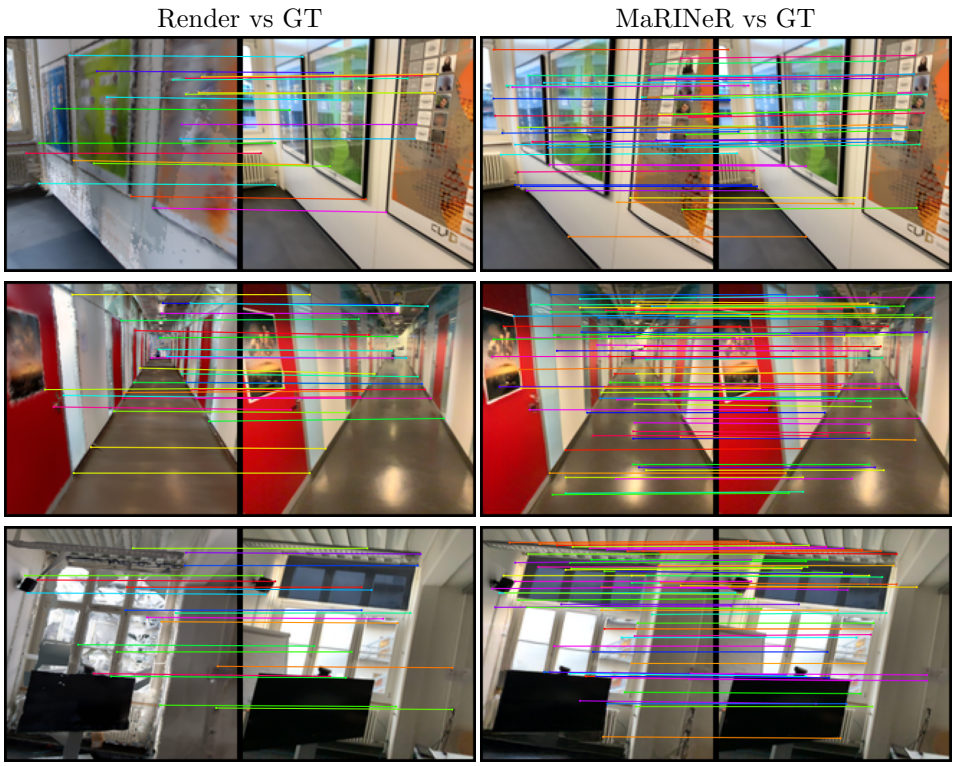


Figure 8.1: Better homography estimation. Using enhanced renderings of our method, estimating a homography between the GT is more accurate and can be used to automate manual sanity checks in the LaMAR [16] pipeline.

Table 8.2: NeRF posprocessing results. Our model successfully enhances renderings of NeRFs with respect to the perceived visual image quality.

	PSNR	SSIM	ERQA	LPIPS
Render	**21.14**	**0.622**	0.646	0.238
OURS	20.45	0.592	**0.701**	**0.167**

movements in 3D scenes such as EgoGen [17], synthetic trajectories can be generated effortlessly to extend the existing datasets. However, because of the quality of the underlying 3D representations, there is a gap between synthetic and real images. Fig. 8.2 shows that with our method we can take a synthetic trajectory and enhance it using a nearby existing reference image from previously recorded trajectories.

8.3 NeRF postprocessing

Training NeRFs can be computationally expensive and requires a large number of images to generate accurate 3D representations [14]. A sufficient number of images may not always be available and training a small model on a large scene with not enough data can lead to noisy representations. Fig. 8.3 shows how our off-the-shelf model removes the artifacts created by the nerfacto [78] model on the Floating tree and Egypt scenes. Both scenes are large and detail-rich outdoor scenes. We use the smallest nerfacto model with default parameters and enhance the evaluation images using the closest training image as reference. Tab. 8.2 shows that our model successfully enhances the nerfacto rendering with respect to ERQA and LPIPS.

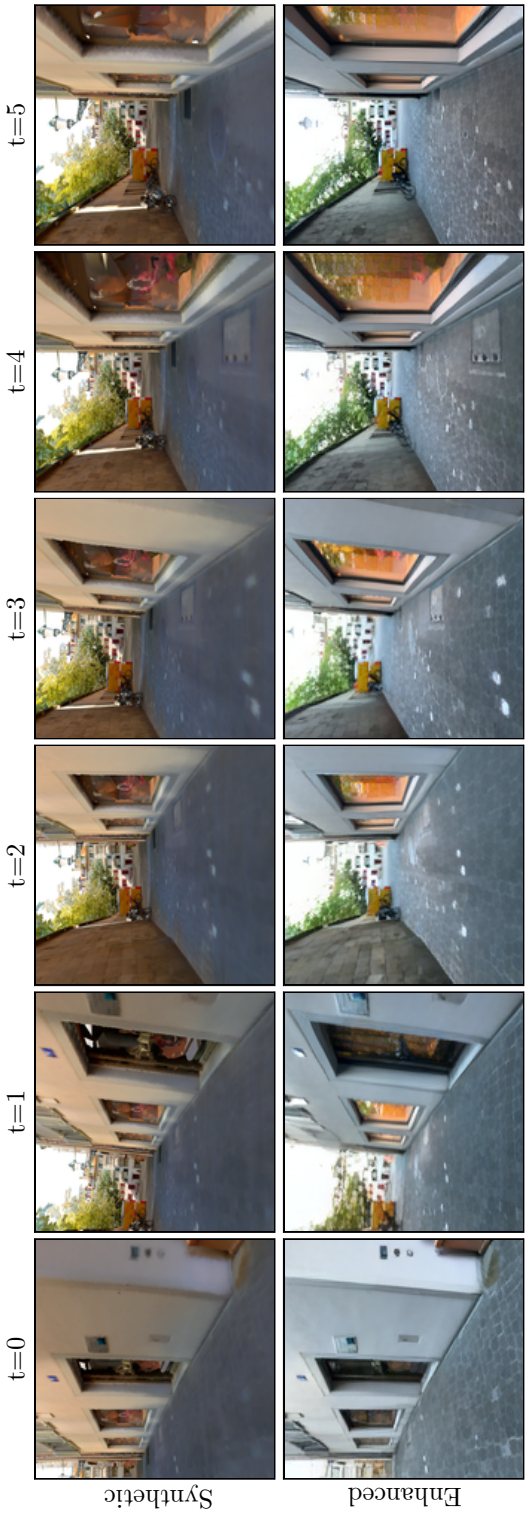


Figure 8.2: Enhancing synthetic trajectories with nearby localized images. The result exhibits increased realism and can extend the current dataset without introducing a gap between synthetic and human recorded trajectories.

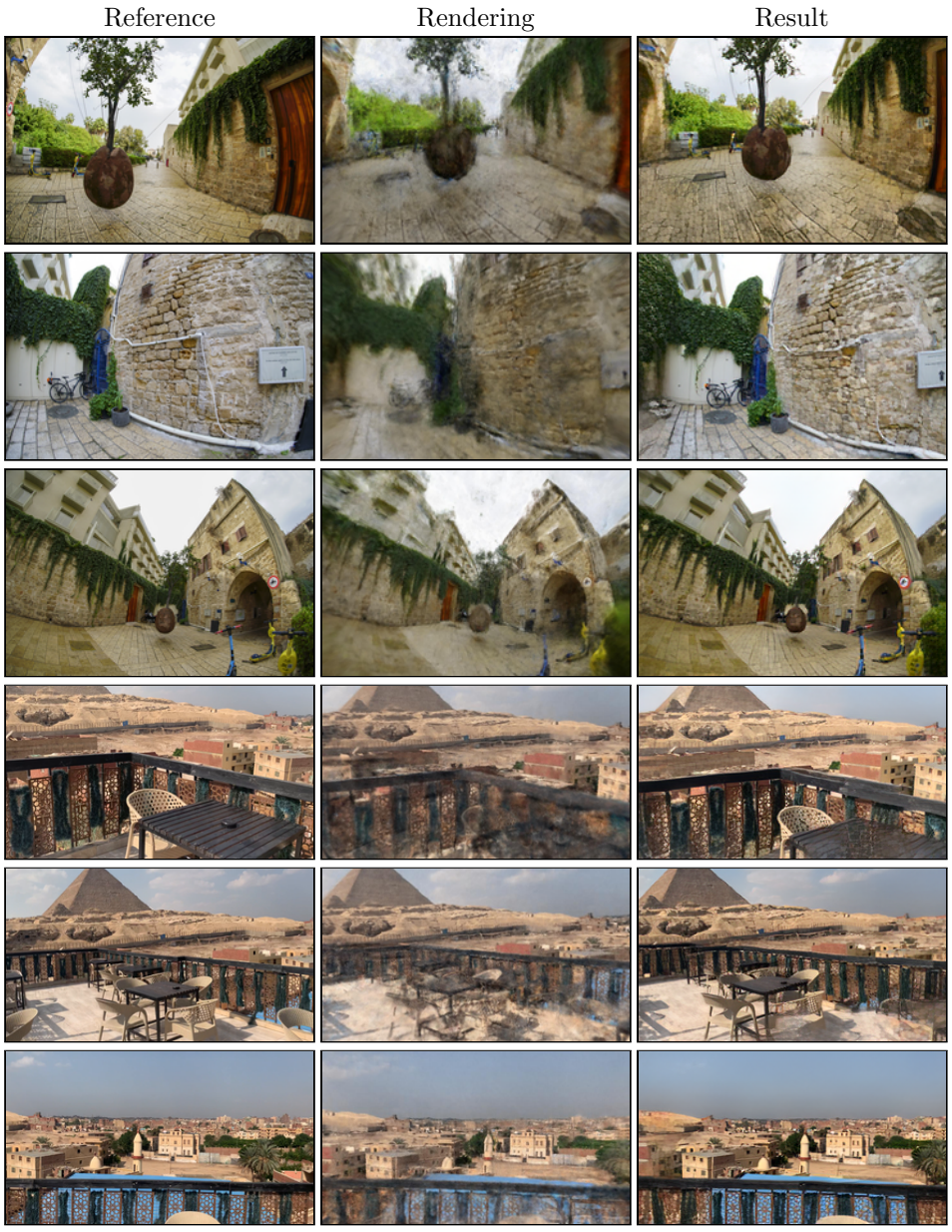


Figure 8.3: NeRF posprocessing results. Training a nerfacto [78] model on the Floating tree and Egypt data. We use the smallest nerfacto model and the result contains artifacts which our model can successfully remove.

Chapter 9

Limitations and Future Work

Fig. 9.1 shows an overview over the current limitations of **MaRINeR**. While the model detects and removes rendering artifacts, it is also possible that some content is wrongly detected as an artifact and removed. This can lead to blurry or smeared out image parts. It can happen that the model finds no matches in the reference for some image parts. In such cases the model comes up with artificial details, which can look non realistic to the human eye. The model preserves the content of the rendering, but may transfer additional content from the reference. Currently the model works best on images with resolution in the order of 160 with any aspect ratio. Larger resolutions are only indirectly supported, by first down-scaling the rendering, running our model and then up-scaling the image again using a super resolution method, such as Real-ERSGAN [64]. This could be addressed in the future by transitioning to a more advanced matching pipeline such as LoFTR [60] or CroCro [79] which would come at the cost of more inference time. The model is targeted to enhance low quality renderings, thus high quality renderings are only improved with very close references. The method matches objects on a texture level and not on a semantic level. This means that the objects should have similar texture, where the rendering is a low quality version. The current model may introduce flickering between neighboring frames of a sequence. For video prediction, the pipeline may be further extended to ensure temporal consistency between the generated frames.

Figure 9.1: Limitations of MaRINeR. Some objects can be wrongly detected as artifacts and partially vanish. This limits the method to enhance high quality renderings, here shown as a blurry version of a real image. Also content can be transferred from the reference that was not present before in the rendering. In some situations, the content of the rendering is also changed, where for example doors can be closed.

Chapter 10

Conclusion

In this work, we propose a novel method to enhance renderings of 3D reconstructions. Specifically, we use localized images in the 3D scene to enhance renderings from the 3D reconstruction. Our experiments verify that our model **MaRINeR** enhances the rendering better than existing models in the domains of Reference-based Super-Resolution or Style Transfer. It is scene and device agnostic, robust to mesh resolution changes, generalizes to greyscale and reliably removes 3D reconstruction artifacts. Possible applications include the automatizing of manual sanity checks in ground-truthing pipelines, enhancing synthetic trajectories data and improving the rendering results of neural renderings trained with limited data or resources.

Bibliography

[1] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," 2019.

[2] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," 2020.

[3] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," 2020.

[4] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," 2023.

[5] H. Zhang, C. Wang, S. Tian, B. Lu, L. Zhang, X. Ning, and X. Bai, "Deep learning-based 3d point cloud classification: A systematic survey and outlook," *Displays*, vol. 79, 2023.

[6] K. Litomisky and B. Bhanu, "Removing moving objects from point cloud scenes," in *International Workshop on Advances in Depth Image Analysis and Applications*, 2012.

[7] M. Bassier, M. Vergauwen, and F. Poux, "Point cloud vs. mesh features for building interior classification," *Remote Sensing*, 2020.

[8] R. Huang, S. Peng, A. Takmaz, F. Tombari, M. Pollefeys, S. Song, G. Huang, and F. Engelmann, "Segment3d: Learning fine-grained class-agnostic 3d segmentation without manual labels," 2023.

[9] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *International Conference on 3D Vision (3DV)*, 2017.

[10] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[11] T. Schöps, T. Sattler, and M. Pollefeys, "BAD SLAM: Bundle adjusted direct RGB-D SLAM," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[12] T. Schöps, J. L. Schönberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger, "A multi-view stereo benchmark with high-resolution images and multi-camera videos," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. Turner, E. Undersander, W. Galuba, A. Westbury, A. X. Chang, M. Savva, Y. Zhao, and D. Batra, "Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai," 2021.

[14] F. Remondino, A. Karami, Z. Yan, G. Mazzacca, S. Rigon, and R. Qin, "A critical analysis of nerf-based 3d reconstruction," *Remote Sensing*, vol. 15, no. 14, 2023.

[15] Z. Zhang, Z. Wang, Z. Lin, and H. Qi, "Image super-resolution by neural texture transfer," 2019.

[16] P.-E. Sarlin, M. Dusmanu, J. L. Schönberger, P. Speciale, L. Gruber, V. Larsson, O. Miksik, and M. Pollefeys, "Lamar: Benchmarking localization and mapping for augmented reality," 2022.

[17] G. Li, K. Zhao, S. Zhang, X. Lyu, M. Dusmanu, Y. Zhang, M. Pollefeys, and S. Tang, "Egogen: An egocentric synthetic data generator," 2024.

[18] J. Valentin, A. Dai, M. Nießner, P. Kohli, P. Torr, S. Izadi, and C. Keskin, "Learning to navigate the energy landscape," 2016.

[19] J. Wald, T. Sattler, S. Golodetz, T. Cavallari, and F. Tombari, "Beyond controlled environments: 3d camera re-localization in changing indoor scenes," in *European Conference on Computer Vision (ECCV)*, 2020.

[20] H. Yue, X. Sun, J. Yang, and F. Wu, "Landmark image super-resolution by retrieving web images," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 4865–4878, 2013.

[21] Z. Li, Z.-S. Kuang, Z.-L. Zhu, H.-P. Wang, and X.-L. Shao, "Wavelet-based texture reformation network for image super-resolution," *IEEE Transactions on Image Processing*, vol. 31, pp. 2647–2660, 2022.

[22] H. Zheng, M. Ji, L. Han, Z. Xu, H. Wang, Y. Liu, and L. Fang, "Learning cross-scale correspondence and patch-based synthesis for reference-based super-resolution," in *British Machine Vision Conference*, 2017.

[23] H. Zheng, M. Ji, H. Wang, Y. Liu, and L. Fang, "Crossnet: An end-to-end reference-based super resolution network using cross-scale warping," 2018.

[24] Y. Xie, J. Xiao, M. Sun, C. Yao, and K. Huang, "Feature representation matters: End-to-end learning for reference-based image super-resolution," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 230–245.

[25] G. Shim, J. Park, and I. S. Kweon, "Robust reference-based super-resolution with similarity-aware deformable convolution," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8422–8431.

[26] F. Yang, H. Yang, J. Fu, H. Lu, and B. Guo, "Learning texture transformer network for image super-resolution," 2020.

[27] Y. Jiang, K. C. K. Chan, X. Wang, C. C. Loy, and Z. Liu, "Robust reference-based super-resolution via c2-matching," 2021.

[28] L. Lu, W. Li, X. Tao, J. Lu, and J. Jia, "Masa-sr: Matching acceleration and spatial adaptation for reference-based image super-resolution," 2021.

[29] R. Dong, L. Zhang, and H. Fu, "Rrsgan: Reference-based super-resolution for remote sensing image," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–17, 2022.

[30] J. Cao, J. Liang, K. Zhang, Y. Li, Y. Zhang, W. Wang, and L. V. Gool, "Reference-based image super-resolution with deformable attention transformer," 2022.

[31] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," 2021.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[33] S. Wang, Z. Sun, and Q. Li, "High-to-low-level feature matching and complementary information fusion for reference-based image super-resolution," *The Visual Computer*, vol. 40, pp. 1–10, 02 2023.

[34] X. Mei, Y. Yang, M. Li, C. Huang, K. Zhang, and P. Lió, "A feature reuse framework with texture-adaptive aggregation for reference-based super-resolution," 2023.

[35] L. Zhang, X. Li, D. He, F. Li, Y. Wang, and Z. Zhang, "Rrsr:reciprocal reference-based image super-resolution with progressive feature alignment and selection," 2022.

[36] J. Zheng, Y. Liu, Y. Feng, H. Xu, and M. Zhang, "Contrastive attention-guided multi-level feature registration for reference-based super-resolution," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 20, no. 2, oct 2023.

[37] X. Yan, W. Zhao, K. Yuan, R. Zhang, Z. Li, and S. Cui, "Towards content-independent multi-reference super-resolution: Adaptive pattern matching and feature aggregation," in *European Conference on Computer Vision*, 2020.

[38] M. Pesavento, M. Volino, and A. Hilton, "Attention-based multi-reference learning for image super-resolution," 2021.

[39] K. Zhao, H. Tan, and T. F. Yau, "Multi-reference image super-resolution: A posterior fusion approach," 2022.

[40] L. Zhang, X. Li, D. He, E. Ding, and Z. Zhang, "Lmr: A large-scale multi-reference dataset for reference-based super-resolution," 2023.

[41] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Universal style transfer via feature transforms," 2017.

[42] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," 2017.

[43] J. An, S. Huang, Y. Song, D. Dou, W. Liu, and J. Luo, "Artflow: Unbiased image style transfer via reversible neural flows," 2021.

[44] Y. Deng, F. Tang, W. Dong, C. Ma, X. Pan, L. Wang, and C. Xu, "Stytr^2: Image style transfer with transformers," 2022.

[45] N. Kolkin, M. Kucera, S. Paris, D. Sykora, E. Shechtman, and G. Shakhnarovich, "Neural neighbor style transfer," 2022.

[46] Y. Zhang, C. Fang, Y. Wang, Z. Wang, Z. Lin, Y. Fu, and J. Yang, "Multimodal style transfer via graph cuts," 2020.

[47] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, "Neural style transfer: A review," 2018.

[48] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[49] J. Yoo, Y. Uh, S. Chun, B. Kang, and J.-W. Ha, "Photorealistic style transfer via wavelet transforms," 2019.

[50] Y. Zhou, G. Wu, Y. Fu, K. Li, and Y. Liu, "Cross-mpi: Cross-scale stereo for image super-resolution using multiplane images," 2021.

[51] Y. Huang, X. Zhang, Y. Fu, S. Chen, Y. Zhang, Y.-F. Wang, and D. He, "Task decoupled framework for reference-based super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 5931–5940.

[52] X. Huang, W. Li, J. Hu, H. Chen, and Y. Wang, "Refsr-nerf: Towards high fidelity and super resolution view synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 8244–8253.

[53] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," 2017.

[54] D. Cohen-Steiner and F. Da, "A greedy delaunay-based surface reconstruction algorithm," *The Visual Computer*, vol. 20, pp. 4–16, 2004.

[55] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," 2018.

[56] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," 2020.

[57] W. Falcon and The PyTorch Lightning team, "PyTorch Lightning," Mar. 2019.

[58] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration," *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*, vol. 6, no. 3, p. 67, 2017.

[59] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[60] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, "Loftr: Detector-free local feature matching with transformers," 2021.

[61] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[62] P. Lindenberger, P.-E. Sarlin, and M. Pollefeys, "Lightglue: Local feature matching at light speed," 2023.

[63] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P.-Y. Huang, H. Xu, V. Sharma, S.-W. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "Dinov2: Learning robust visual features without supervision," 2023.

[64] X. Wang, L. Xie, C. Dong, and Y. Shan, "Real-esrgan: Training real-world blind super-resolution with pure synthetic data," in *International Conference on Computer Vision Workshops (ICCVW)*, 2021.

[65] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[66] F. Pittaluga, S. J. Koppal, S. B. Kang, and S. N. Sinha, "Revealing scenes by inverting structure from motion reconstructions," in *CVPR*, 2019.

[67] P. Krawczyk, M. Gaertner, A. Jansche, T. Bernthaler, and G. Schneider, "Artifact generation when using perceptual loss for image deblurring," 07 2023.

[68] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[69] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard gan," 2018.

[70] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018.

[71] A. Kirillova., E. Lyapustin., A. Antsiferova., and D. Vatolin., "Erqa: Edge-restoration quality assessment for video super-resolution," in *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP,*, INSTICC. SciTePress, 2022, pp. 315–322.

[72] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *Proceedings of the 2010 20th International Conference on Pattern Recognition*, ser. ICPR '10. USA: IEEE Computer Society, 2010, p. 2366–2369.

[73] Z. Wang, E. Simoncelli, and A. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 2, 2003, pp. 1398–1402 Vol.2.

[74] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[75] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, p. 84–90, may 2017.

[76] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla, "Benchmarking 6dof outdoor visual localization in changing conditions," 2018.

[77] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration," 2017.

[78] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. Mcallister, J. Kerr, and A. Kanazawa, "Nerfstudio: A modular framework for neural radiance field development," in *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*, ser. SIGGRAPH '23. ACM, Jul. 2023.

[79] Weinzaepfel, Philippe and Leroy, Vincent and Lucas, Thomas and Brégier, Romain and Cabon, Yohann and Arora, Vaibhav and Antsfeld, Leonid and Chidlovskii, Boris and Csurka, Gabriela and Revaud Jérôme, "CroCo: Self-Supervised Pre-training for 3D Vision Tasks by Cross-View Completion," in *NeurIPS*, 2022.