Inovis: Instant Novel-View Synthesis

Mathias Harrer*
mathias.mh.harrer@fau.de
Friedrich-Alexander-Universität
Erlangen-Nürnberg
Germany

Linus Franke*
linus.franke@fau.de
Friedrich-Alexander-Universität
Erlangen-Nürnberg
Germany

Laura Fink
laura.fink@fau.de
Friedrich-Alexander-Universität
Erlangen-Nürnberg
Germany
Fraunhofer IIS
Germany

Marc Stamminger
marc.stamminger@fau.de
Friedrich-Alexander-Universität
Erlangen-Nürnberg
Germany

Tim Weyrich
tim.weyrich@fau.de
Friedrich-Alexander-Universität
Erlangen-Nürnberg
Germany

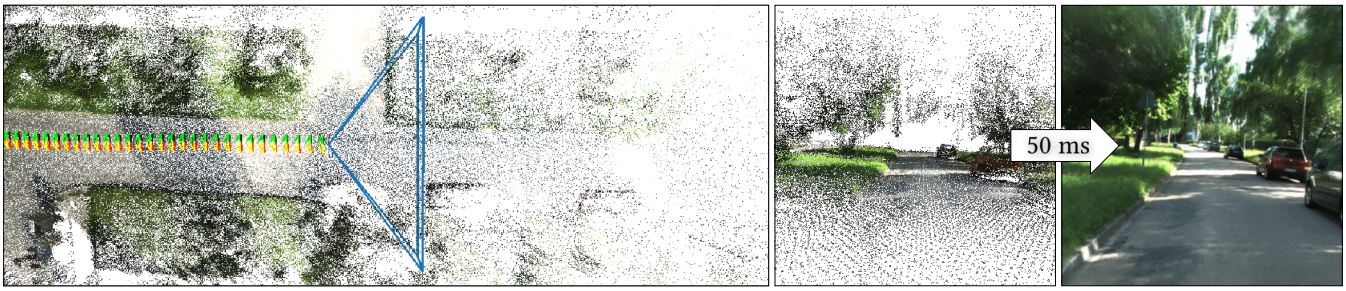

Figure 1: Our neural point-based rendering architecture is particularly suited for scenarious where novel, previously unseen content is dynamically added to a scene. *Left:* simulation of a gradually built up LiDAR-based reconstruction; top-down view of the novel views' frustum and frustums of past frames, which are used as input. Although the scene ahead of the camera is increasingly sparse with distance (*center*), our architecture manages to render a novel view with remarkable fidelty (*right*).

ABSTRACT

Novel-view synthesis is an ill-posed problem in that it requires inference of previously unseen information. Recently, reviving the traditional field of image-based rendering, neural methods proved particularly suitable for this interpolation/extrapolation task; however, they often require a-priori scene-completeness or costly preprocessing steps and generally suffer from long (scene-specific) training times. Our work draws from recent progress in neural spatio-temporal supersampling to enhance a state-of-the-art neural renderer's ability to infer novel-view information at inference time. We adapt a supersampling architecture [Xiao et al. 2020], which resamples previously rendered frames, to instead recombine nearby camera images in a multi-view dataset. These input frames are warped into a joint target frame, guided by the most recent (point-based) scene representation, followed by neural interpolation. The resulting architecture gains sufficient robustness to significantly improve transferability to previously unseen datasets. In particular, this enables novel applications for neural rendering where dynamically streamed content is directly incorporated in a (neural) image-based reconstruction of a scene. As we will show, our method reaches state-of-the-art performance when compared to previous works that rely on static and sufficiently densely sampled scenes; in addition, we demonstrate our system's particular suitability for dynamically streamed content, where our approach is able to produce high-fidelity novel-view synthesis even with significantly fewer available frames than competing neural methods.

*joint first authors

SA Conference Papers '23, December 12–15, 2023, Sydney, NSW, Australia

https://doi.org/10.1145/3610548.3618216

CCS CONCEPTS

• **Computing methodologies** → **Rendering**; **Image-based rendering**; Reconstruction.

KEYWORDS

Novel-View Synthesis, Point-based Graphics, Neural Rendering

ACM Reference Format:
Mathias Harrer, Linus Franke, Laura Fink, Marc Stamminger, and Tim Weyrich. 2023. Inovis: Instant Novel-View Synthesis. In *SIGGRAPH Asia 2023 Conference Papers (SA Conference Papers '23), December 12–15, 2023,*

Sydney, NSW, Australia. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3610548.3618216

1 INTRODUCTION

Novel-view synthesis creates previously unobserved images of a scene, from camera observations from different angles. As such, it is an inherently ill-posed problem. Recently, reviving the traditional field of image-based rendering, neural methods proved particularly suitable for this interpolation/extrapolation task; however, they often require dedicated and exhaustive training to learn the scene's appearance prior to rendering, usually a time-consuming process.

Our goal was to devise a system that is able to produce novel views of an initially incomplete scene, at interactive rates (i.e., instant novel-view synthesis) and without the requirement of preprocessing the scene beforehand, to enable immediate view generation as camera data becomes available, for instance in a live streaming scenario, such as virtual site inspection or chase cams.

Amongst prior art, two major directions have shown promise in the last years: implicit and proxy-based methods.

Proxy-based methods build upon a traditional approach of image-based rendering [Shum and Kang 2000] by using a geometry proxy (commonly a mesh or point cloud) and integrating captured views into a target view on this proxy. Recent examples of this are *Neural Point Based Graphics (NPBG)* [Aliev et al. 2020] and extensions [Rakhimov and Ardelean et al. 2022; Rückert et al. 2022], as well as *Stable View Synthesis (SVS)* [Riegler and Koltun 2021].

Implicit methods forgo this direct geometry for a neural implicit representation (commonly an MLP) and query this representation for novel views. Best known in this domain are NeRF [Mildenhall et al. 2021] and their variants [Barron et al. 2021; Turki et al. 2022; Yu et al. 2021; Zhang et al. 2020].

Recent years witnessed improvements in computational costs of methods in both classes. This includes the use of generalized feature encoders [Rakhimov and Ardelean et al. 2022; Riegler and Koltun 2021] or clever data structures [Müller et al. 2022a] to reduce preprocessing time from hours to seconds. Specialized renderers [Rückert et al. 2022; Schütz et al. 2022] reduce inference time, and learned scene priors can reduce the number of inputs required for some methods [Yu et al. 2021]. The above, however, depend on repeated observations of scene elements for high-quality reconstructions. We observe that, particularly for interactive applications that require instant novel-view synthesis in regions where the scene is just being observed, this poses a major limitation.

To address this, we devise an interactive (thus preprocessing-free) solution that supports instantaneous novel-view synthesis even in regions with poor coverage by input frames. It operates in the geometry proxy domain, directly on point clouds to avoid costly and potentially unstable meshing [Wolff et al. 2016] and capitalizing on point clouds being the native representation of 3D reconstruction methods such as multi-view stereo [Schönberger et al. 2016], RGBD streams [Keller et al. 2013], or LiDAR scanners [Liao et al. 2022].

Our method employs a neural point-based renderer that largely follows the design of NPBG [Aliev et al. 2020]. In their original design, appearance descriptors are optimized in an off-line process. In contrast, our approach seeks to work with dynamic input—and an evolving model—at render time, assuming a SLAM-like setting in which a (point-based) 3D representation is gradually built up.

As a key contribution of our method, our rendering system does not simply rely on the fused model representation but also capitalizes on the availability of original frames close to the target view to be rendered, by combining the model rendering with a more image-based approach that recombines these nearby frames (which we call *auxiliary frames*) to fill in higher-fidelity information where a consistent high-resolution model has not yet been built. Recombination and fusion with a direct (neural) rendering of the point-based representation is enabled by drawing from work on temporal supersampling [Xiao et al. 2020], re-applied in our context.

In effect, our method is designed to use fewer resources and capturing constraints: new scenes do not require new training (unlike Aliev et al. [2020] or Rückert et al. [2022]); the method works directly on sparse point clouds (unlike Stable View Synthesis [Riegler and Koltun 2021], which requires highest-quality geometry). As a result, our method works even on dynamic LiDAR or depth-map streams, allowing on-the-fly novel-view synthesis previously not possible in neural point renderings. In summary, our paper's contributions are:

- An instantaneous, high performance method for novel-view synthesis that compares favorably to the state of the art in similarly constrained scenarios.
- Novel-view synthesis in previously difficult or impossible scenarios, such as live depth map streams.
- An optional temporal feedback loop that reuses previously rendered novel views to augment the set of RGBD captured images for improved temporal coherence.
- An open-source implementation of the method (https://reality.tf.fau.de/publications/2023/harrerfranke2023inovis/).

2 RELATED WORK

Our work builds upon the state of the art in novel-view synthesis and point-based 3D reconstruction and rendering, while drawing from analogies to neural supersampling.

Novel-View Synthesis. Novel-view synthesis is a long-standing problem with a large body of work in computer vision and graphics. Early approaches employ *image-based rendering*, with large activity in the early 2000s [Kang et al. 2007; Shum and Kang 2000] and renewed interest with the advance of deep learning models that have the potential to replace or augment classic pipelines [Tewari et al. 2020, 2022]. Generally, methods vary in their utilization and representation of the underlying scene geometry, with light field approaches, for instance, forgoing explicit geometric representations (e.g., NeRF) while point or mesh-based methods build upon existing geometry.

Recent *implicit methods* follow NeRF [Mildenhall et al. 2021] and use a fully connected network combined with volume rendering to optimize a continuous representation. They achieve visually impressive results but come with significant challenges that several follow-up works address: reduction of the number of required views [Chibane et al. 2021; Yu et al. 2021; Zhang et al. 2020]; improvement of rendering speeds [Barron et al. 2021; Chen et al. 2021; Neff et al. 2021] and training times [Chen et al. 2021; Chibane et al. 2021; Müller et al. 2022a; Tancik et al. 2021; Turki et al. 2022].

In the domain of *explicit scene representations* (mainly point clouds and meshes), recent advances came through neural rendering techniques, that fall into two categories: approaches that use individual, scene-based overfitting, and those that generalize across scenes. For scene-specific methods, object texture optimization [Thies et al. 2019] and per-point descriptors [Aliev et al. 2020; Kopanas et al. 2021; Rückert et al. 2022] show great results integrating neural methods and explicit geometry. The generalizing methods partially follow in the footsteps of image-based rendering [Chaurasia et al. 2013; Debevec et al. 1998] and include pixel blending with learned factors [Hedman et al. 2018], integration of encoded features into blending [Riegler and Koltun 2020, 2021] and interpolation of nearby views with transformer-based architectures [Wang et al. 2021]. Lastly, feature-encoding of input images before aggregation into points [Rakhimov and Ardelean et al. 2022] proved successful.

Similar to implicit techniques, however, these explicit methods feature lengthy training times [Aliev et al. 2020; Rückert et al. 2022; Thies et al. 2019], challenging inference speeds [Riegler and Koltun 2021; Wang et al. 2021] and generally depend on a-priori scene completeness, prohibiting use on live camera streams.

Closest to our scope, recent work that bridges implicit and explicit approaches tackles the task of online scene reconstruction [Clark 2022; Müller et al. 2022b; Sucar et al. 2021]. These involve training a reconstruction and creating respective novel views during capture time; however, time to image (training plus rendering) is still in the order of seconds for high-fidelity images. Furthermore, faithful reconstructions require repeated observations of the same objects.

Our work tackles these shortcomings, supporting incomplete scene observations, generalized feature encoding without scene-specific pretraining, featuring fast inference without online training.

Capturing and Rendering Point Clouds. Point clouds can be captured using a variety of methods, the most common being RGBD cameras with depth projection or fusion techniques [Dai et al. 2017b; Keller et al. 2013; Whelan et al. 2016], LiDAR-based mapping (often present on cars [Liao et al. 2022]) or multi-view stereo techniques [Schönberger et al. 2016].

The advantages of point clouds, apart from the ease of capturing, include quite precise (if sparse) data points as well as many established fast rendering methods [Schütz et al. 2021, 2022, 2019] and well-established hole-filling neural networks proven in novel-view synthesis tasks [Aliev et al. 2020; Kopanas et al. 2021; Rakhimov and Ardelean et al. 2022; Rückert et al. 2022]. This combination forms a foundation of our proposed method.

Closest to our method is NPBG++ [Rakhimov and Ardelean et al. 2022], which integrates features from all input images (via a generalized feature encoder) within a point cloud that is then neurally interpolated to render novel views. It involves a few minutes of preprocessing and renders novel views in real time. In contrast to NPBG++, however, our method does not require preprocessing and is not bound to a-priori scene completeness; moreover, its lower per-point memory footprint improves scalability to larger scenes.

Neural Supersampling. A special case of novel-view synthesis is real-time (spatio-temporal) upsampling of lower-resolution inputs, with prominent use in video upsampling [Kappeler et al. 2016; Liu and Sun 2013; Tao et al. 2017] and, more recently, real-time upsampling for video games, such as Nvidia's DLSS [Edelsten et al. 2019]. Recent approaches increasingly use neural networks to upscale the input stream [Liu et al. 2022]. Another class of solutions achieves temporal supersampling mostly via temporal information projected to current frames [Guo et al. 2021; Thomas et al. 2022; Xiao et al. 2020].

A core insight of our paper is that upsampling from coarse data solves a similar problem as novel-view synthesis from a sparse point cloud, in that both aim to fill in missing information; and that recent solutions for both rely on feature encoding and interpolation stages.

In this paper, we draw from previous research in temporal supersampling and transfer insights from Xiao et al. [2020]'s neural supersampling feature encoding and feature reweighting concepts for a fast and lightweight novel-view synthesis pipeline that is able to self-correct most common warping artifacts.

3 METHOD

Our method employs a neural point-based renderer that largely follows the design of NPBG [Aliev et al. 2020]. Their scene representation are 3D points, each holding a low-dimensional feature vector that describes the view-depend appearance and local geometry of each point. (Their feature encoding is learned as part of an exhaustive preprocessing of the scene.) For a given viewpoint, these 3D points are sparsely rendered at multiple resolutions and transformed into a dense, high-resolution RGB rendering using a multi-scale U-Net.

In their original design, appearance descriptors are precomputed in an off-line process, and the multi-resolution renderings to be fed into the U-Net are rendered in feature space. In contrast, our approach seeks to work with dynamic input—and an evolving model—at render time. Accordingly, and with the convenient side-effect of memory savings that allow for larger scenes, point and image attributes are kept in the original RGB(D) format for longer, and feature vectors are extracted only on the fly, see our pipeline overview in Fig. 2. (The encoder is pre-trained using a class of similar input scenes.)

In general, we assume a SLAM-like setting, in which a (in our case point-based) 3D representation is gradually built up while additional RGBD data is progressively received and fused into the model.

As a key contribution of our method, our rendering system does not simply rely on the fused model representation but also capitalizes on the availability of original frames close to the target view to be rendered, by combining the model rendering with a more image-based approach that recombines these nearby frames to fill in higher-fidelity information where a consistent high-resolution model is not yet available. Notably, these additional frames (we call them *auxiliary views*) may contain only sparse depth data (as, e.g., in the case of LiDAR sensors) but are assumed to be dense in RGB.

While similar in spirit to traditional image-based rendering and texturing [Buehler et al. 2001; Schönberger et al. 2016], we blend feature descriptors rather than RGB values and borrow architectural aspects from the design of Xiao et al. [2020].

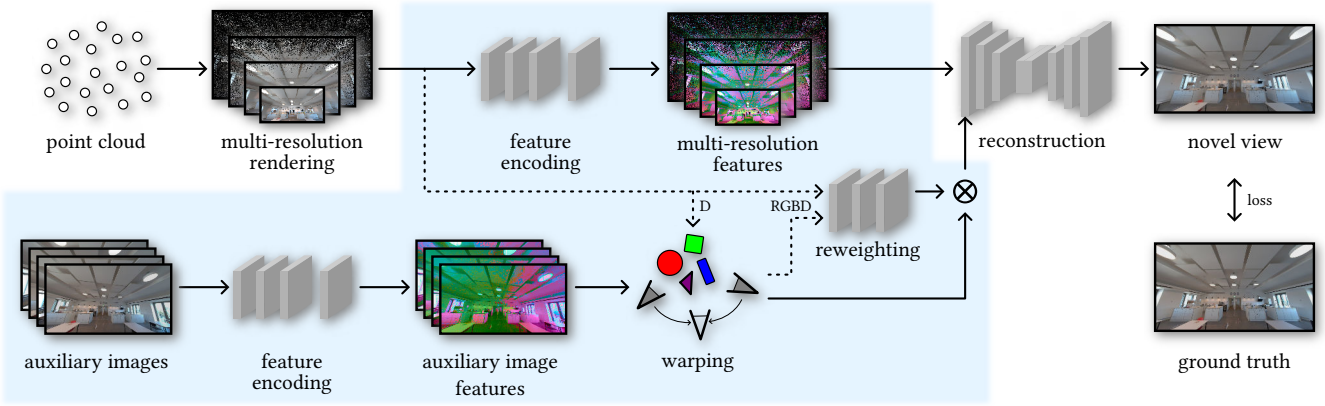

Figure 2: Our rendering pipeline extends the approach of NPBG; novel components are highlighted in blue. From a set of auxiliary (RGBD) images from nearby input (or previously rendered) views, relevant views are selected and encoded. Guided by a rendering of the point cloud, they are warped and reweighted to mitigate occlusion and ghosting effects. Subsequently, features from the multi-resolution rendering of the point cloud are extracted, and these buffers are passed to the reconstruction network for the final rendering. The pipeline is trained end-to-end against ground truth images.

In concrete terms, for every output (target) frame, our system determines nearby original camera images (Sec. 3.2), converts them into feature space (Sec. 3.3) and warps them into that target frame (Sec. 3.4). The warp is assisted by depths of the so far accumulated 3D model. The different source pixels are blended using a reweighting network [Xiao et al. 2020], which leads to superior blending results over explicit weighting functions and hole-filling strategies (Sec. 3.5). Subsequently, features from the multi-resolution rendering of the point cloud are extracted (Sec. 3.1) and combined with the blended (feature) images before being passed to the reconstruction network for rendering (Sec. 3.6).

Any (supervised) pre-training takes place using this complete pipeline (Sec. 4). In Sec. 5 we test the system both with previously observed data (in order to allow comparison to previous works) as well as with dynamic input outside the training dataset, a scenario not normally considered by previous work.

3.1 Target View Feature Extraction

The backbone of our novel-view rendering architecture is an NPBG-inspired neural point-based renderer that roughly divides into two parts: sparse multi-resolution encoding of a target frame (which we discuss here); and U-Net-based neural rendering that transforms this representation into an RGB rendering (Sec. 3.6).

First, we render the raw (RGB) point cloud with a depth-testing, one-pixel per point OpenGL hardware renderer to obtain a sparse RGBD image. In a second step, we then use a feature encoder with the same structure as the auxiliary view's feature extractor (Sec. 3.3), except a filter number of 32 and output of 8 features derived from the rendered sparse point cloud. On output, the resulting features are concatenated again with the network input.

These two steps are repeated for every resolution level fed into the multi-scale rendering U-Net described (Sec. 3.6). In our experiments we use three lower-resolution point renderings, at 1/2, 1/4, and 1/8.

The multi-res structure serves two purposes: more general occlusion features can be learned from these coarser representations; during training, the feature extractor sees a more varying range of point distributions, making the feature extractor more robust.

3.2 Auxiliary View Selection

We collect n_{views} *auxiliary images*, taken from nearby captured RGB images. Depending on the sensor types and acquisition system, captured depth images are used (RGBD scenes), or (sparse) corresponding depth samples are generated from the LiDAR data using the same point renderer we use in Sec. 3.1. Either depth representation is used without any further preprocessing.

We generally determine the n_{views} nearest views by selecting those that minimize the following metric based on a positional factor f_p and a view directional factor f_d:

$$f_p = 0.5 + \max\left(\|\vec{p} - \vec{p}_{\text{target}}\|^2, 0.5\right), \tag{1}$$

$$f_d = 1 - \left(\frac{\vec{v}}{\|\vec{v}\|}\right)^{\top} \cdot \frac{\vec{v}_{\text{target}}}{\|\vec{v}_{\text{target}}\|}. \tag{2}$$

Here, p_{target} and v_{target} are the position and view direction of the target view, and p and v are the auxiliary buffers' position (viewpoint) and view direction. In general, positions close to the target position will always contribute more, thus scoring lower—as do similar view directions. Combining these two factors then results in a similarity score s, where lower values indicate better view choices:

$$s = f_p\,(1 + \alpha f_d). \tag{3}$$

Hereby, α is used to balance these two factors, with larger α prioritizing tighter view directions. We use $\alpha = 100$ for a trade-off that penalises distances of around 10 meters similarly to 90-degree view offsets. Views are always ordered from best to worst, thus guiding the network to put more emphasis on the closest view. The resulting view weighting and ranking in spirit resembles traditional view-dependent rendering approaches [Buehler et al. 2001]; more

point rendering | no environment encoding | environment encoding | closest ground truth

Figure 3: Comparison of environment encoding vs no environment encoding vs point rendering vs closest ground truth. Environment encoding significantly improves image quality in areas unoccupied by points. Even without environment encoding, the surrounding areas of points are filled with information, showing the capability of encoding a patch around a pixel into a feature tensor which is then warped, using a sparse point rendering.

complex view selection schemes, such as ORB-SLAM [Campos et al. 2021], could be used but were deemed to offer diminishing returns within the scenarios at hand.

For some datasets, varying quality of nearby views could lead to moderate flickering in the output video. In these cases, adding the previously *rendered* view to the set of candidates for auxiliary views helps mitigate flicker: that view often scores highly on the nearby-view metric and, if included amongst the n_{views} frames, improves temporal coherence across output frames. For training, however, previous frames are never considered; our pipeline is robust enough to easily handle a previously rendered frame as input, considering that it is fully compatible with the original RGBD-captured frames.

3.3 Feature Encoding for Auxiliary Views

After view selection, a lightweight feature encoding network transforms RGBD pixels into 12-dimensional feature vectors. Note that the encoder captures spatial context as well, so that even under sparse (point-wise) reprojections, as applied in the following section, relevant spatial information is preserved.

We use a three-stage gated convolution setup with ReLU activations and a filter number of 16. (Xiao et al. [2020] use 32; our experiments showed no change in training loss when going from 16 to 32, so we went with fewer weights.)

Also in contrast to Xiao et al., we use gated convolutions [Yu et al. 2019], which have learnable masks for scaling inputs. As our auxiliary images' depth maps are sparsely filled, passing this map to gated convolutions allows the feature encoding network to identify areas which can not be correctly warped (due to missing depth) and to amplify this information to adjacent, warpable pixels. Thus, gated convolutions in our case allow for a form of "inverse" masking, where information warping is able to account for sparse depth maps similarly to small-scale attention mechanisms in transformers [Vaswani et al. 2017]. Lastly, this allows us to use a smaller number of filters, which generally improves performance and reduces the risk of overfitting.

3.4 Warping

For each auxiliary view, both features and RGBD are warped to the target view: for each point of the point cloud visible from the target perspective, its pixel value (RGBD plus feature vector), at its back-projection into the auxiliary view, is copied to the new location. Each auxiliary image is warped separately and the resulting images are mixed together in later pipeline steps.

Pixels that do not feature rasterized points miss warped information. These areas are optionally augmented by warping the auxiliary image onto a distant background plane behind the point proxy geometry. This helps to greatly improve image quality in areas that do not feature points, especially for far away areas (e.g., sky or far-away buildings). This environment handling results in those areas being filled with plausible content from the background, as the disparity is barely noticeable at these large distances, see Fig. 3.

Then, the RGBD images from the auxiliary views and the original rendering are passed to the reweighting network, which computes a pixel-wise weighting factor for each view to scale the features.

3.5 Reweighting

This step helps to lessen the effect of disocclusion and warping artifacts. Reweighting mostly follows the work of Xiao et al. [2020]. The reweighting network takes RGBD information from both warped auxiliary views and the rendered target view to determine per-pixel weights for all n_{views} auxiliary images as output. The warped image data is weighted accordingly by multiplying the weights with the auxiliary features. The reweighted data is concatenated and passed to the front layer of the neural render network.

The network consists of three blocks of convolution and ReLU with filter numbers of 32, outputting n_{views} multiplication factors, scaled to $[0, 10]$ as stated in the original paper [Xiao et al. 2020].

3.6 Neural Render Network

The neural rendering network outputs the final stable novel view of the requested extrinsics and intrinsics. It is a U-Net with five levels, similar to established point-cloud rendering inpainting networks [Aliev et al. 2020; Rückert et al. 2022]. Each downsampling block uses a gated convolution and max pooling layer, while each upsampling block uses bilinear upsampling and a gated convolution with skip connections between the blocks. Our filters are 32, 32, 32, 32, 32, which provided the best trade-off between quality and performance and converge faster than deeper networks.

We input the full-resolution encoded features of the target and auxiliary views to the network, with the lower-resolution target feature maps progressively being added to their respective downsampling blocks.

4 TRAINING METHODOLOGY

Scenes. Tab. 3 summarises the scenes used for training (and evaluation), which cover a wide range of scenarios and setups, including depth capturing through LiDAR, multi-view stereo, and RGBD cameras; various indoor and outdoor environments, as well as inside-out and outside-in capturing setups.

We use the *Playground* and *M60* scenes from the commonly used Tanks and Temples dataset [Knapitsch et al. 2017], which has depths estimated with multi-view stereo. Several sequences from *Kitti-360* [Liao et al. 2022] offer sequential recordings of a driving car captured with LiDAR, which we modified to simulate the order in which depth samples and RGB images would come in within a live LiDAR system (based on field of view and distance of the original Kitti 360 point cloud). Additionally, we used our own captured scene, *Office*, using a portable indoor LiDAR, with capture positions roughly 2m apart. Extensive preprocessing yielded a cleaned-up point cloud with a resolution of 5mm.

For live-RGBD scenes, we use the ScanNet [Dai et al. 2017a] and Redwood [Choi et al. 2016] scenes, which are captured RGBD streams for which we estimated positions on the fly with ORB-SLAM [Campos et al. 2021] where no poses where present. Apart from that, we captured *custom outdoor scenes* using a stereo camera, featuring sequential traversal of environments without repeated observation of the same objects. We obtained the poses and keyframes for these scenes with Snake-SLAM [Rückert and Stamminger 2021]. For all live-RGBD scenes, we create timestamped point clouds from the last 10 images in the sequence.

Training. Training is generally initialized with a network pretrained on the Office scene, owing to its large spatial extent combined with large baselines that prime training against ghosting. Subsequently, networks are refined for individual classes of scenes, without temporal feedback frames (using captured data only). Testing, naturally, always takes place on previously unseen data.

Loss. As training loss, we use a mix of VGG16 [Johnson et al. 2016] and SSIM [Wang et al. 2004]. The VGG loss is a common feature-based function, which is especially well suited in our case (as seen in Sec. 5.4). However, regular patterns can appear on surfaces, which can be removed by adding a small amount of SSIM loss. Our final loss function is the following ($w_{\text{ssim}} = 0.25$ and $w_{\text{vgg}} = 1$):

$$\text{loss}(\tilde{x}, x) = w_{\text{ssim}} \cdot (1 - \text{SSIM}(\tilde{x}, x)) + w_{\text{vgg}} \cdot \text{VGG16}(\tilde{x}, x) \quad (4)$$

Inference. Our method is particularly effective when applied to live datasets that supply a steady stream of new images, such as Kitti-360, Redwood, ScanNet, and our custom outdoor scenes. The images are encoded during inference and selected based on their capture position and rotation, enabling us to choose images without having to store them in memory. This makes our approach well-suited for integration with a traditional streaming approach, which only holds a subset of images in memory, thereby minimizing memory usage.

Additionally, we only store RGB data in our point cloud, compared to competing methods such as NPBG++ [Rakhimov and Ardelean et al. 2022], who store large multi-channel descriptors per point. All in all, our rendering is lightweight and is easily interactive (~50ms/frame rendering time on Kitti and Redwood datasets).

5 RESULTS

In the following, we present our evaluation of Inovis comparing to related work, as well as ablation studies to give insights on what makes our method effective. For qualitative results, see Fig. 7.

5.1 Qualitative Evaluation

We compare our approach to established neural novel view synthesis techniques. These include *Stable View Synthesis* (SVS) [Riegler and Koltun 2021], NPBG++ [Rakhimov and Ardelean et al. 2022], IBR-net [Wang et al. 2021] and ADOP [Rückert et al. 2022]. For our method, a subset of scenes is used to train a network tuned to one scene type. Baseline methods that have generalizing capabilities (all except ADOP) were trained on the whole Redwood and ScanNet datasets, except the unseen evaluation scenes. ADOP was individually trained on each evaluation scene to allow for a comparison. We accumulate metrics over five unseen scenes from ScanNet (0, 20, 30, 40, 50, 80), five Redwood motorcycles (05489, 05751, 05984, 06186, 06190) and three Redwood sofas (00577, 05477, 07294). The results of this evaluation can be seen in Tab. 1, Tab. 2 and Fig. 6. In terms of quality, we consistently outperform NPBG++, while they show slightly better render times. SVS shows similar results with significantly slower rendering. ADOP and IBR-Net usually provide better metrics, but neither method in its current form is suitable for online novel view synthesis: ADOP requires scene-specific training (~6h) and IBR-Net shows long rendering times for single images (~2 min). In general, our approach provides state-of-the-art results with fewer limitations than competing approaches.

Table 1: Approximate preprocessing and render times (ScanNet 1296×968).

	preprocess	rendering
ADOP	~ 6 h	~ 13 ms
IBR-Net	< 1 min	~ 2 min
NPBG++	~ 1 min	~ 100 ms
SVS	~ 4 h	~ 3 s
Inovis (ours)	-	~ 131 ms

Table 2: Comparison with SVS, IBRnet, ADOP and NPBG++. Except for ADOP, no scene-specific finetuning is performed.

		PSNR↑	LPIPS↓	SSIM↑
Motorcycle	ADOP	20.34	0.236	0.585
	IBR-Net	22.93	0.193	0.730
	NPBG++	14.70	0.543	0.497
	SVS	19.21	0.273	0.658
	Inovis (ours)	20.63	0.195	0.698
Sofa	ADOP	24.26	0.172	0.598
	IBR-Net	27.45	0.159	0.809
	NPBG++	17.02	0.392	0.637
	SVS	22.35	0.253	0.728
	Inovis (ours)	24.19	0.187	0.769
ScanNet	ADOP	24.78	0.208	0.694
	IBR-Net	24.96	0.248	0.802
	NPBG++	20.90	0.396	0.758
	SVS	23.76	0.316	0.796
	Inovis (ours)	22.78	0.309	0.792

To further demonstrate generalization, we used a subset of scenes from the Kitti dataset to train a network that was also used to create all evaluation images of our own custom dataset (Fig. 8 and supplementary video). While these are also outdoor scenes, their characteristics are different, further corroborating generalizing power.

Table 3: Evaluation datasets. We use a wide range of possible scenarios to evaluate our approach. Scene tagged with *live* indicate an expanding live dataset.

Dataset	# Points per scene	Point Cloud capturing	# Image per scene	Image Resolution	Capturing Methodology	Environment Type
Tanks & Temples	~10M	MVS	~300	1920 − 2144 × 1088	outside-in	various
Kitti-360	~8M	**Live**-LiDAR	~630	1408 × 352	sequential	outdoor driving
ScanNet	~30M	**Live**-DepthCam	~120	1280 × 960	inside-out	indoor rooms
Redwood	~35M	**Live**-DepthCam	~150	640 × 480	outside-in	object scans
Office	~70M	LiDAR	~700	960 × 544	inside-out	indoor rooms
Custom	0.5M-40M	**Live**-StereoCam	40-250	1280 − 1920 × 720 − 1080	sequential	outdoor scans

Table 4: Performance numbers in ms: inference (Inf), point rendering (PR) and total rendering time (T) on single scenes.

Dataset	Inf	PR	T
Kitti (3.6M pts, 1408×376)	53.8	1.2	56.2
ScanNet (70M pts, 1296×968)	120.6	9.7	131.7
Redwood (36M pts, 640×480)	36.7	9.3	47.4
Office (40M pts, 960×540)	58.9	12.9	73.1
Custom (0.6M pts, 1280×720)	88.3	0.6	90.2

5.2 Performance Evaluation

Training. our network end-to-end takes around ~15–24h on an NVIDIA V100, depending on dataset size and resolution.

Rendering. Training is done once per scene type, hence runtime performance depends on rendering (i.e., novel-view synthesis) only, which highly depends on the number of auxiliary views (cf. Sec. 5.4) and resolution. See Tab. 4 for an overview of rendering times for different datasets, obtained with an NVIDIA A5000. We measured interactive frame times of up to 20 fps for 960×540 images and ~11 fps for 1280×720 images. Even for large point clouds (tens of millions of points), inference still dominates rendering time, but differences in visible point count cause variations in total rendering time, as illustrated in Fig. 4.

To assess potential for streaming scenarios, we compare our approach against two competing methods: ADOP [Rückert et al. 2022], as a high-quality point-based approach with very fast neural rendering; InstantNGP [Müller et al. 2022a], which features remarkable real-time training of a NeRF, especially when applied on outside-in scenarios. Considering that neither ADOP's nor InstantNGP's publicly available implementations were designed to incrementally ingest streamed content, we devised an evaluation regime in their favor that, for a selection of time stamps within a stream of keyframes, grants each algorithm combined training and rendering time roughly equivalent to the time passed until that timestamp; see Fig. 8 for the results. Our approach outperforms both methods, as no training is needed and produces high fidelity images from the start, while both compared methods only gradually improve visual quality with increasing time budget. We outperform both methods in all comparisons w.r.t. time-to-image (training plus rendering) and achieve our highest quality even within the minimum time budget.

5.3 Use-Case Evaluation

Our method maps well on different categories of problems rarely tackled in novel-view synthesis approaches.

Live LiDAR Car Data. For live sequences from driving scenarios, take a look at Fig. 7 and Fig. 8 as well as the supplemental video. Using the car dataset Kitti-360, where previously unseen content continuously emerges in front of the car-mounted camera, we show how our approach excels in synthesising views in regions where the available data is sparse. As long as the captured data is within a similar class to the scene our model has been trained on, we are able to instantly create novel views from a small set of captured images and the current point cloud.

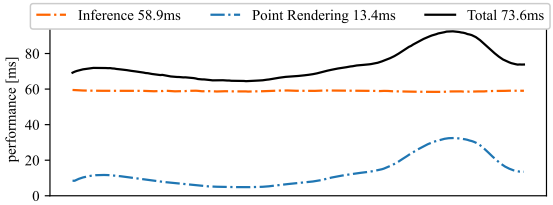

Figure 4: Performance over time during traversal of the office scene (960×540, 4 auxiliary views), featuring a large point cloud (40M points). Due to the large point count, point rendering time is significant. Inference time remains constant while rendering time varies with the number of visible points.

Large Point Clouds. Large point clouds can be challenging for neural methods. In contrast to that, our method easily handles large point clouds, as seen with the Office dataset having 75M points. Results for this scene can be seen in Fig. 7 and the supplementary video.

Sparse Point Clouds. Since our approach encodes the surroundings of pixels into feature vectors and warps these vectors instead of colors, we can augment comparatively sparse point data with image information, as shown with the Kitti-360 dataset, which features relatively sparse LiDAR-captured point clouds. See Figs 3, 7, and 8, and the supplementary video.

Live RGBD Data. Our approach is a natural fit for live-streamed RGBD data. Provided a SLAM computes camera poses in real time and a network for a similar scene has already been trained, Inovis creates new images in the matter of dozens of milliseconds without any scene-specific training.

5.4 Ablation Studies

In this section, we provide studies to show the root of the effectiveness of our method. Variants were evaluated by using our loss function (Sec. 4) on the hold-out set of captured frames.

Feature Extraction. Our pipeline encodes small patches of auxiliary view information into feature vectors, allowing this information to be transferred to novel views, as seen in Fig. 3. As described in Sec. 3, we use gated convolutions, which usually provide state-of-the art holefilling capabilities. In contrast to that, we use their

masking abilities inversely for auxiliary image encoding, with the sparse depth map guiding information amplification. For results of this, see Tab. 5. In sparse point cloud scenarios (such as LiDAR in cars) this addition provides great improvement in the relevant (usually small) areas, thus increasing training convergence and providing improvements of up to 5% overall.

Table 5: Loss by convolution type of feature encoding of auxiliary images, by dataset.

Dataset	basic conv.	gated conv.
Office	0.1867	0.1798
Kitti-360	0.5807	0.5561

Number of Auxiliary Images. We use a comparatively small number of auxiliary views compared to competing methods, which for our setup suffices. As seen in Fig. 5, using more auxiliary images shows diminishing results especially considering the increased computation time per frame, which increases linear with number of images used. Four seems the best tradeoff between quality and inference speed. In our experience, using four images also helps to improve quality when deviating from ground truth views, although two images already show a similar loss compared to four images.

Loss function. We compare different loss functions commonly used in novel view synthesis methods in our method. For results see Tab. 6. Numerically, our metric performs favorably, only slightly surpassed by the loss function of Neural Supersampling [Xiao et al. 2020]; subjectively, however, we find that our metric provides higher color accuracy for our datasets, suggesting that real-world datasets require differing loss compositions than synthetic data.

Table 6: Loss functions on the Office dataset. *NSS* indicates the loss function used by Xiao et al. [2020].

Loss Function	PSNR↑	LPIPS↓	SSIM↑
Train w. MSE	24.14	0.141	0.891
Train w. SSIM	24.16	0.110	0.917
Train w. VGG	23.85	**0.090**	0.913
Train w. NSS	**24.27**	0.102	**0.919**
Train w. Ours	24.22	0.093	0.917

6 DISCUSSION

Input Quality. Our method relies on good quality captures to produce high-quality results. This includes the capture positions of the images not be too far apart in order for the method to work effectively. In general, regions without overlap from auxiliary images show poor quality due to missing information to be interpolated. We fall back on averaged pixel colors in our point cloud rendering, but these can exhibit artifacts.

Temporal Stability. While our method is able to handle continuous streams of new images, it may still struggle with scenes that have significant changes over time. E.g., siginificant changes of auxiliary images result in visible flickering, as can be observed in the supplementary video.

Camera Intrinsics. Our pipeline is trained to produce output images with similar camera intrinsics and resolutions as the input images. Thus, if encoded patches are interpreted with different camera characteristics, the output image can be blurred or squished.

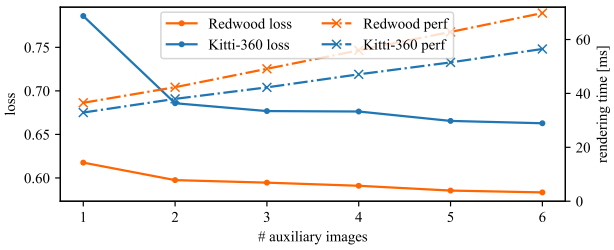

Figure 5: Loss and rendering performance vs auxiliary images used.

Streaming Scenario. In principle, to minimize memory footprint during view synthesis, only a small candidate set of nearby auxiliary images as well as a point cloud of the current vicinity would have to be kept in GPU memory. Further (nearby) images and points could be dynamically streamed in and swapped out during run time; however, in setups with limited transfer bandwidth, larger sets should be kept in memory to avoid temporarily decreased quality until the newer nearby data arrives. While our demonstrator does not include such a streaming mechanism, we argue that our results are indicative of favorable performance under this regime.

7 CONCLUSION

We presented a system for neural novel-view synthesis. We adapted a supersampling architecture, which resamples previously rendered frames, to instead recombine nearby camera images in a multi-view dataset. The resulting architecture gains sufficient robustness to significantly improve transferability to previously unseen datasets. In particular, our system enables novel applications for neural rendering where dynamically streamed content is directly incorporated in a (neural) image-based reconstruction of a scene. We show that our method reaches state-of-the-art performance when compared to previous works that rely on static scenes; in addition, we demonstrated our system's performance for dynamically streamed content, a scenario not accessible to previous works in neural rendering.

Ultimately, we believe that our approach is opening up a number of valuable applications, including wide-baseline video stabilisation, VR conferencing with free viewpoint control within the remote scene, virtual car mirrors, etc.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) under the NHR project b162dc. NHR funding is provided by federal and Bavarian state authorities. NHR@FAU hardware is partially funded by the German Research Foundation (DFG) – 440719683. Linus Franke was supported by the Bayerische Forschungsstiftung (Bavarian Research Foundation) AZ-1422-20. We thank NavVis GmbH for providing the office dataset. We thank Darius Rückert for helping to capture the Custom dataset.

REFERENCES

Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. 2020. Neural point-based graphics. In *European Conference on Computer Vision*. Springer, 696–712.

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. 2021. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 5855–5864.

Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. 2001. Unstructured Lumigraph Rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 425–432. https://doi.org/10.1145/383259.383309

Carlos Campos, Richard Elvira, Juan J. Gómez, José M. M. Montiel, and Juan D. Tardós. 2021. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM. *IEEE Transactions on Robotics* 37, 6 (2021), 1874–1890.

Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. 2013. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG)* 32, 3 (2013), 1–12.

Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. 2021. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14124–14133.

Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. 2021. Stereo radiance fields (srf): Learning view synthesis for sparse views of novel scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7911–7920.

Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. 2016. A Large Dataset of Object Scans. https://doi.org/10.48550/ARXIV.1602.02481

Ronald Clark. 2022. Volumetric bundle adjustment for online photorealistic scene capture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6124–6132.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. 2017a. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.

Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. 2017b. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)* 36, 4 (2017), 1.

Paul Debevec, Yizhou Yu, and George Borshukov. 1998. Efficient view-dependent image-based rendering with projective texture-mapping. In *Eurographics Workshop on Rendering Techniques*. Springer, 105–116.

Andrew Edelsten, Paula Jukarainen, and Anjul Patney. 2019. Truly next-gen: Adding deep learning to games and graphics. In *In NVIDIA Sponsored Sessions (Game Developers Conference)*.

Jie Guo, Xihao Fu, Liqiang Lin, Hengjun Ma, Yanwen Guo, Shiqiu Liu, and Ling-Qi Yan. 2021. ExtraNet: real-time extrapolated rendering for low-latency temporal supersampling. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–16.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–15.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*. Springer, 694–711.

Sing Bing Kang, Yin Li, Xin Tong, Heung-Yeung Shum, et al. 2007. Image-based rendering. *Foundations and Trends® in Computer Graphics and Vision* 2, 3 (2007), 173–258.

Armin Kappeler, Seunghwan Yoo, Qiqin Dai, and Aggelos K Katsaggelos. 2016. Video super-resolution with convolutional neural networks. *IEEE transactions on computational imaging* 2, 2 (2016), 109–122.

Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. 2013. Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion. In *Proc. of Joint 3DIM/3DPVT Conference (3DV)*. 1–8. Selected for oral presentation..

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. *ACM Transactions on Graphics* 36, 4 (2017).

Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. 2021. Point-Based Neural Rendering with Per-View Optimization. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 29–43.

Yiyi Liao, Jun Xie, and Andreas Geiger. 2022. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 3 (2022), 3292–3310.

Ce Liu and Deqing Sun. 2013. On Bayesian adaptive video super resolution. *IEEE transactions on pattern analysis and machine intelligence* 36, 2 (2013), 346–360.

Hongying Liu, Zhubo Ruan, Peng Zhao, Chao Dong, Fanhua Shang, Yuanyuan Liu, Linlin Yang, and Radu Timofte. 2022. Video super-resolution based on deep learning: a comprehensive survey. *Artificial Intelligence Review* (2022), 1–55.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Thomas Müller, Alex Evans, Christoph Schied, Marco Foco, András Bódis-Szomorú, Isaac Deutsch, Michael Shelley, and Alexander Keller. 2022b. Instant neural radiance fields. In *ACM SIGGRAPH 2022 Real-Time Live!* 1–2.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022a. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)* 41, 4 (2022), 1–15.

Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H Mueller, Chakravarty R Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. 2021. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 45–59.

Ruslan Rakhimov and Ardelean, Andrei-Timotei Rakhimov and Ardelean, Victor Lempitsky, and Evgeny Burnaev. 2022. NPBG++: Accelerating Neural Point-Based Graphics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15969–15979.

Gernot Riegler and Vladlen Koltun. 2020. Free view synthesis. In *European Conference on Computer Vision*. Springer, 623–640.

Gernot Riegler and Vladlen Koltun. 2021. Stable view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12216–12225.

Darius Rückert, Linus Franke, and Marc Stamminger. 2022. Adop: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–14.

Darius Rückert and Marc Stamminger. 2021. Snake-SLAM: Efficient global visual inertial SLAM using decoupled nonlinear optimization. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 219–228.

Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*.

Markus Schütz, Bernhard Kerbl, and Michael Wimmer. 2021. Rendering point clouds with compute shaders and vertex order optimization. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 115–126.

Markus Schütz, Bernhard Kerbl, and Michael Wimmer. 2022. Software rasterization of 2 billion points in real time. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 5, 3 (2022), 1–17.

Markus Schütz, Katharina Krösl, and Michael Wimmer. 2019. Real-time continuous level of detail rendering of point clouds. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 103–110.

Harry Shum and Sing Bing Kang. 2000. Review of image-based rendering techniques. In *Visual Communications and Image Processing 2000*, Vol. 4067. SPIE, 2–13.

Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. 2021. iMAP: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6229–6238.

Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P Srinivasan, Jonathan T Barron, and Ren Ng. 2021. Learned initializations for optimizing coordinate-based neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2846–2855.

Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. 2017. Detail-revealing deep video super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision*. 4472–4480.

Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. 2020. State of the art on neural rendering. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 701–727.

Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, W Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. 2022. Advances in neural rendering. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 703–735.

Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.

Manu Mathew Thomas, Gabor Liktor, Christoph Peters, Sungye Kim, Karthik Vaidyanathan, and Angus G Forbes. 2022. Temporally Stable Real-Time Joint Neural Denoising and Supersampling. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 5, 3 (2022), 1–22.

Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. 2022. Mega-NeRF: Scalable Construction of Large-Scale NeRFs for Virtual Fly-Throughs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12922–12931.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. 2021. IBRNet: Learning Multi-View Image-Based Rendering. In *CVPR*.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. 2016. ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research* 35, 14 (2016), 1697–1716.

Katja Wolff, Changil Kim, Henning Zimmer, Christopher Schroers, Mario Botsch, Olga Sorkine-Hornung, and Alexander Sorkine-Hornung. 2016. Point cloud noise and outlier removal for image-based 3D reconstruction. In *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, 118–127.

Lei Xiao, Salah Nouri, Matt Chapman, Alexander Fix, Douglas Lanman, and Anton Kaplanyan. 2020. Neural supersampling for real-time rendering. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 142–1.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. 2021. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4578–4587.

Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. 2019. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE/CVF international conference on computer vision*. 4471–4480.

Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. 2020. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492* (2020).

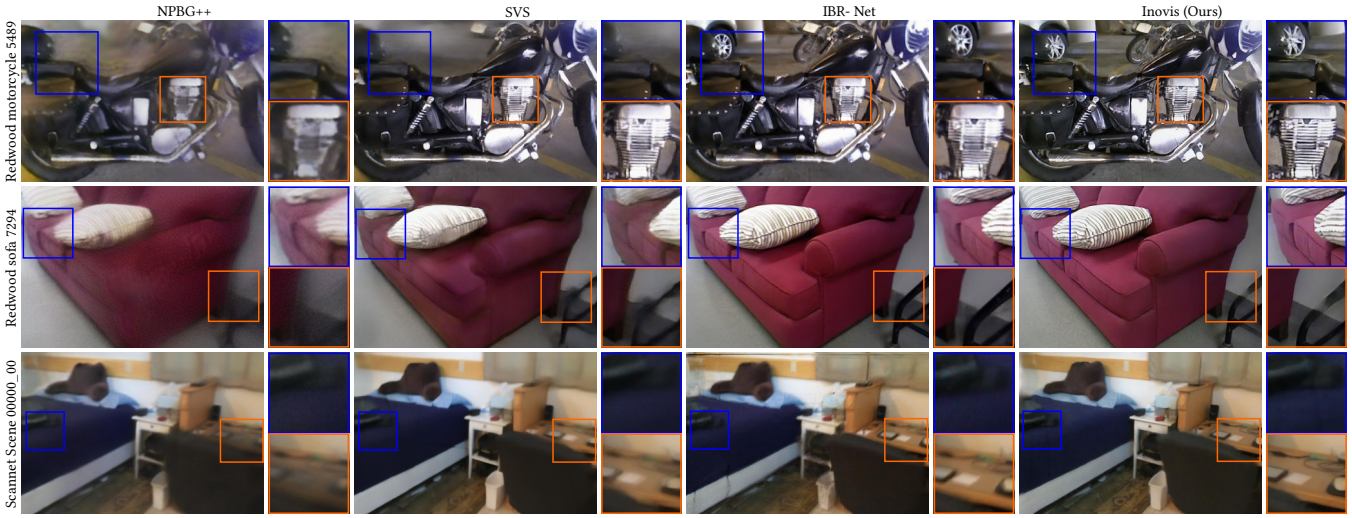

Figure 6: Visual comparison of Inovis with NPBG++, SVS and IBR-Net. Our method produces visually similar results to IBR-Net, while outperforming NPBG++ and SVS.

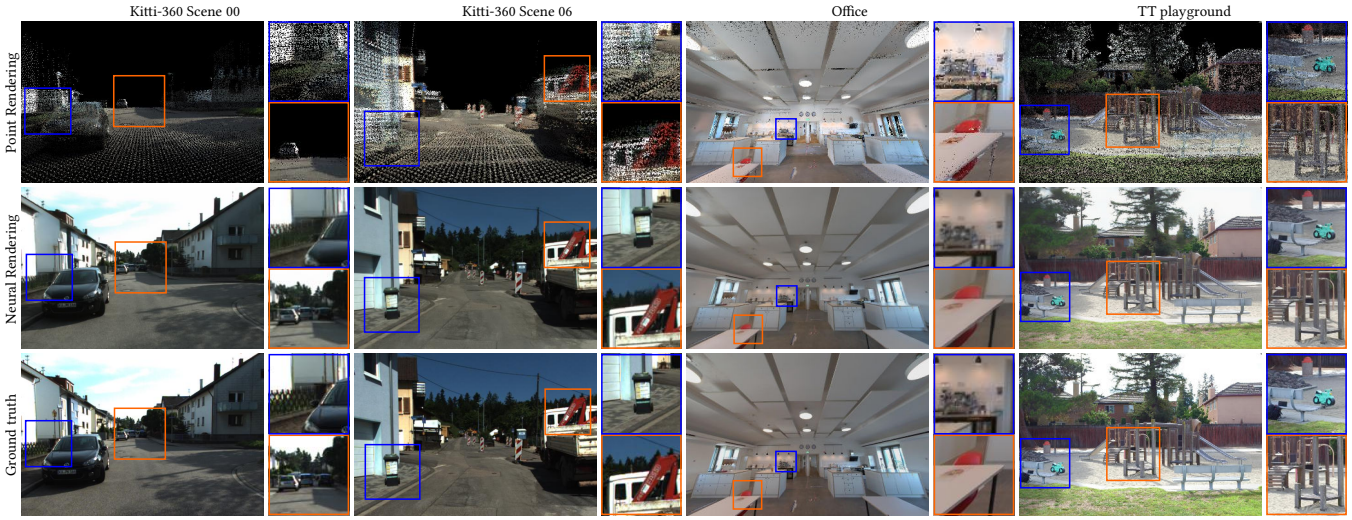

Figure 7: Neural Renderings of our approach for three datasets: the Office dataset and the Tanks and Temples dataset, both containing high quality point clouds and the Live LiDAR dataset Kitti-360. Our method produces high quality neural renderings for all datasets.

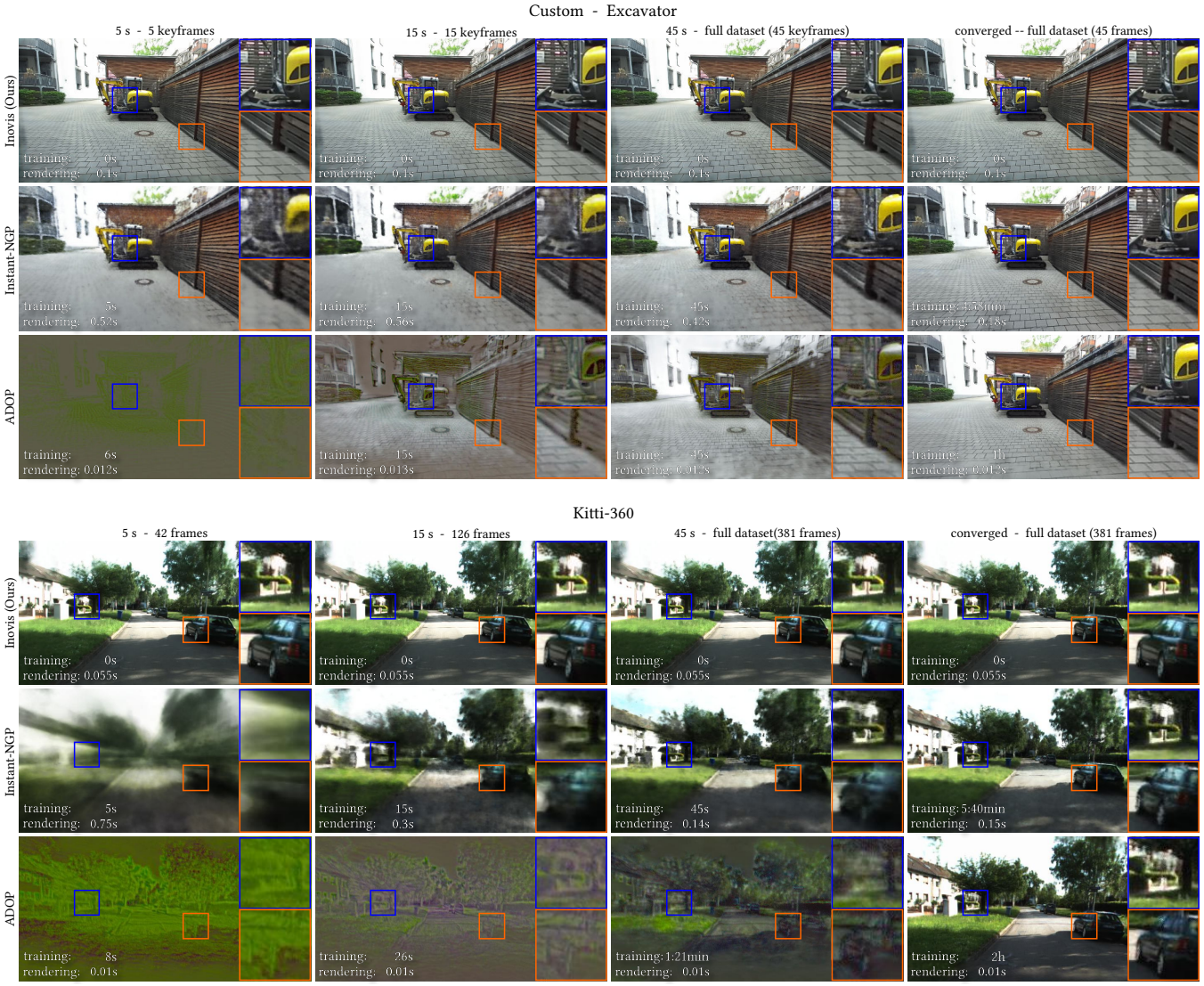

Figure 8: Comparison of streaming capabilities between ADOP, Instant-NGP and Inovis (Ours). Each method is provided with different time budgets that match the data capture time frame of the respective dataset, e.g., a time budget of 15s for creating a novel view from 15 keyframes. (*column 1–3*). The last *column* contains the results for unlimited time budgets. Both used datasets resemble a sequential trajectory through a scene, rather than continuous observation of a single object: walking past an excavator (Custom scene; *top*); a car driving through a neighborhood (Kitti-360; *bottom*). Training and rendering times, which result in total time to image by adding up, are displayed in each image. For all examples shown, our system was pre-trained on a subset of Kitti-360 scenes, excluding the one on display. Without further training, our network generalizes well to the shown scenes, thus limiting the total time to image to mere rendering time (~50–100ms) and resulting in the exact same image, no matter the time budget. ADOP and Instant-NGP gradually improve with increasing time budget but do not reach our quality as long as a time limit is in place.