



Learning Generic Local Shape Properties for Adaptive Super-Sampling

Christian Reinbold  and Rüdiger Westermann 

Computer Graphics & Visualization Group, Technische Universität München, Garching, Germany

Abstract

We propose a novel encoder/decoder-based neural network architecture that learns view-dependent shape and appearance of geometry represented by voxel representations. Since the network is trained on local geometry patches, it generalizes to arbitrary models. A geometry model is first encoded into a sparse voxel octree of features learned by a network, and this model representation can then be decoded by another network in-turn for the intended task. We utilize the network for adaptive super-sampling in ray-tracing, to predict super-sampling patterns when seeing coarse-scale geometry. We discuss and evaluate the proposed network design, and demonstrate that the decoder network is compact and can be integrated seamlessly into on-chip ray-tracing kernels. We compare the results to previous screen-space super-sampling strategies as well as non-network-based world-space approaches.

1. Introduction

Scene representation networks (SRNs) have gained popularity for single- and multi-view object reconstruction [LGZL*20, MST*20, NMOG20, PFS*19, SZW19, YKM*20] and realtime rendering [GKJ*21, TLY*21]. SRNs represent models by features that are learnt by a network, so called latent codes. They can be accessed at arbitrary positions to obtain model-specific information such as shortest distance to the model or color. The networks overfit to the model they are trained for and do not generalize to other models. For shape reconstruction from 3D point clouds, Jiang et al. address this limitation by encoding implicit functions of local geometry patches in low dimensional latent codes, and optimizing for those during reconstruction [JSM*20]. Similarly, local geometric detail can be encoded in 3D style codes and transferred to coarse geometric representations for geometry upscaling [CKF*21], and to implement data-driven mesh subdivision [LKC*20].

We extend on current SRNs as follows: We present a novel encoder/decoder-based network architecture, called PatchNet, which learns generic local properties of geometric shapes on different levels of details (LoDs). The architecture generalizes to new models by learning a view-dependent encoding of local geometry patches, so that at a coarse scale the network can predict the appearance of geometric details. The learned feature representations are organized in a hierarchical LoD data structure to support coarse to fine look-ahead at various scales.

We utilize the aforementioned capabilities in ray-tracing, to predict super-sampling patterns when seeing coarse-scale geometry. Therefore, we ray-trace against a Sparse Voxel Octree (SVO) [LK11], where each node represents a voxel that intersects the ge-

ometry and stores averaged luminance L_V and normal N_V over the intersected surface. For each pixel, a ray is intersected with the SVO by traversing it in top-down manner until the projected voxel size is approximately equal to the pixel size. Then, the pixel appearance can be estimated by L_V at the intersected voxel. If the local geometry shows self-occlusions for the current view, or projects only to a sub-pixel area, L_V is likely to be a bad approximation. Super-sampling patterns attempt to identify this case and compute improved estimates by tracing several rays for each critical pixel.

We compare our approach to a non-network-based world-space approach as well as classical [LRU85, Mit87, RFS03a, RFS03b, XSSXZ07] and network-based [KKR18, WITW20] screen-space super-sampling strategies. Conceptually, screen-space approaches refine a pixel whenever its neighboring pixels show significant variation in appearance. They fail if neighboring pixels show similar, yet consistently wrong appearances. For instance, this effect occurs at fence-like structures where the gaps between laths are not preserved at coarser scales.

2. Method

We propose a world-space approach that uses a view-dependent per-ray oracle function $f: (\mathcal{P}, \theta) \mapsto (\sigma_{\text{gt}}, L_{\text{gt}})$ to detect pixels that need refinement. Its input is a local geometry patch $\mathcal{P}(M, p)$ of M close to the intersection point p between the ray and the SVO, as well as additional rendering parameters θ . \mathcal{P} is projected into the current pixel according to θ , and the oracle returns the fraction $\sigma_{\text{gt}} \in [0, 1]$ of the pixel covered by the projection of \mathcal{P} as well as the average seen luminance $L_{\text{gt}} \in [0, 1]$. If $\sigma_{\text{gt}} \ll 1$ or $|L_{\text{gt}} - L_V| \gg 0$, the current pixel has to be refined to obtain an accurate pixel ap-

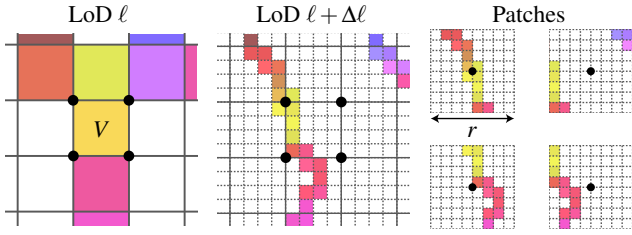


Figure 1: 2D Example showing patches for all corners of a voxel V with $\Delta\ell = 2$, $r = 8$. They are obtained by sampling blocks of fine voxels with side length r after descending $\Delta\ell$ levels of detail.

pearance. We suggest to realize f by an encoder/decoder network PatchNet, which can be evaluated efficiently on a per-pixel basis. The encoder $E: (M, p) \mapsto \mathcal{P}_{\text{enc}}(M, p)$ extracts the local geometry patch \mathcal{P} and transforms it to a low dimensional feature code \mathcal{P}_{enc} . The decoder $D: (\mathcal{P}_{\text{enc}}, \theta) \mapsto (\sigma_{\text{N}}, L_{\text{N}})$ acts as a lightweight neural renderer to compute approximate values $(\sigma_{\text{N}}, L_{\text{N}})$ for $(\sigma_{\text{gt}}, L_{\text{gt}})$.

2.1. Encoder

The encoder design is inspired by the work of Takikawa et al. [TLY*21], in which a collection \mathcal{Z} of trainable feature codes is structured in a SVO that corresponds to the geometric model. Each allocated voxel stores a d -dimensional feature code at each of its corners, with the features being shared at common corners of adjacent voxels. When SVO traversal stops by intersecting a voxel of LoD $\ell \in \mathbb{N}$ at $p \in \mathbb{R}^3$, the eight feature codes of the intersected voxel are trilinearly interpolated according to the relative position of p in the voxel. The interpolated d -dimensional vector forms the output \mathcal{P}_{enc} of the encoder. In a number of experiments we have found that $d = 10$ is sufficient to learn model independent features.

Patch Codes

In current scene representation networks, the feature codes in \mathcal{Z} are learned on a per model basis, i. e. a training process is required once per model. Instead of optimizing for the feature codes directly, we propose to train a Patch Encoder network E_{patch} that maps small patches of local geometry data to patch dependent feature codes, which we call *patch codes*. Given a corner $C \in \mathbb{R}^3$ of a voxel V at LoD ℓ in the SVO, its corresponding patch holds geometry data over a block of r^3 "fine" voxels at LoD $\ell + \Delta\ell$ that is centered at C (see Fig. 1 for a 2D example). Geometry information of each fine voxel is represented by a 5D vector with the first component encoding shape by setting it to 1 if the fine voxel is existing, and the remaining components containing L_V, N_V as stored in the fine voxel. The resulting 4D tensor of shape $(r, r, r, 5) \in \mathbb{R}^{r \times r \times r \times 5}$ is passed to $E_{\text{patch}}: \mathbb{R}^{r \times r \times r \times 5} \rightarrow \mathbb{R}^d$ to compute the patch code for C .

We found that E_{patch} can be efficiently realized by flattening the input to a $5r^3$ -dimensional vector and processing it in a multilayer network with three hidden layers of 256 channels each. Each layer is realized by taking the input vector x (or output of the previous layer) and transforming it to the output $y = \max(Mx + b, 0) \in \mathbb{R}^{256}$ with an appropriately sized matrix M and bias vector b containing learnable network weights. Experiments with 3D convolutions and

Voxception blocks [BLRW16] increased computation costs significantly with no significant effect on predictive power. We attribute this to the oracle function f lacking translation invariance.

We set $\Delta\ell = 2$, so that PatchNet is able to "look ahead" two additional levels of detail when deciding if refined sampling is required. The block resolution is set to $r = 2 \cdot 2^{\Delta\ell}$, yielding a block of twice the side length than that of V . Thus, the network is able to perceive all geometry that can influence the current pixel, even if the pixel's center ray just barely scratches V .

2.2. Decoder

The decoder D is realized by a small multilayer network that combines the feature code of the encoder and additional rendering parameters to form the output $(\sigma_{\text{N}}, L_{\text{N}})$. The following input is provided to the decoder: a) The d -dimensional encoder output \mathcal{P}_{enc} . b) The 6D *ray frame* for the current pixel, i.e., the ray direction and its up vector (the projection of the camera's up vector to the orthogonal complement plane of the ray). c) The 1D *pixel-voxel-ratio* encoding the relative size of the projected voxel to the current pixel. It is obtained by dividing the side length of the pixel by the side length of the voxel projected onto the screen. d) All evaluations of the 25D *spherical harmonics* up to degree 4 at the ray direction, to facilitate learning view-dependent features. 9 out of 10 dimensions of the patch code \mathcal{P}_{enc} are interpreted as spherical harmonic coefficients by multiplying them with the nine spherical harmonic base functions up to degree 2 evaluated at the ray direction.

The decoder consists of three layers with 48 channels each. Random dropout of network units to prevent overfitting was not performed, and normalization of activation values to the same scale didn't show any improvement. The output layer is followed by a shifted and rescaled non-linear sigmoid activation function to limit the network outputs to the interval $[-1, 2]$. Outputs are further clamped to $[0, 1]$ during inference, yet no additional clamping is performed during training to avoid vanishing gradients.

2.3. Training

E_{patch} and the decoder *once* are trained on randomly sampled geometry patches of a complex geometry model. A training sample is generated by first sampling a voxel V of the model's SVO, extracting a patch at LoD $\ell + \Delta\ell$ with resolution $(3 \cdot 2^{\Delta\ell})^3$ centered at V , and rendering it into an orthographic 64×64 viewport representing a single pixel. The pixel-voxel-ratio is uniformly sampled from $[\frac{1}{2}, 1]$ and determines the size of the viewport in world space. Camera parameters are obtained via rejection sampling such that the viewport's center ray intersects V . By choosing a patch resolution as stated above, it is ensured that the patch can fill out the whole viewport independently of where the center ray intersects V .

Each training sample is processed by invoking E_{patch} at each corner of V and retrieving \mathcal{P}_{enc} at the intersection point as described in Sec. 2.1. Next, the rendering parameters are derived from the training sample and the decoder is invoked to compute $(\sigma_{\text{N}}, L_{\text{N}})$. To train the network, its weights are repeatedly updated through a gradient descent optimizer that minimizes $|\sigma_{\text{N}} - \sigma_{\text{gt}}| + |L_{\text{N}} - L_{\text{gt}}|$. The gradients are computed by back-propagating them via the chain

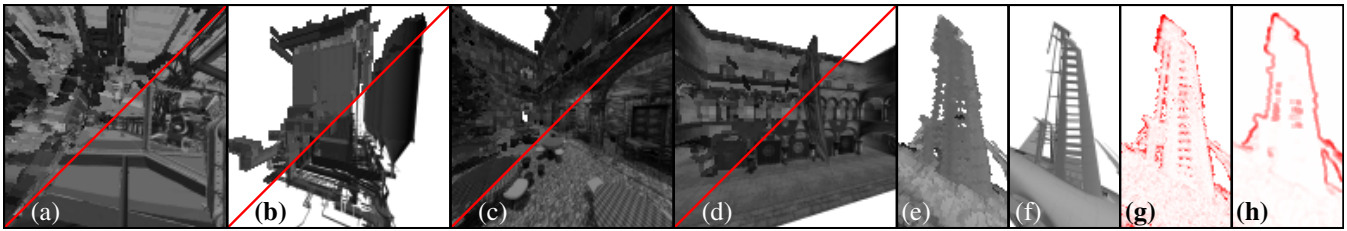


Figure 2: (a-d) Sample rendering splits (top left = 1^2 -spp, bottom right = 32^2 -spp) of all datasets used during training and evaluation. (a) Underbelly interior of Airplane, (b) Power Plant, (c) San Miguel, (d) Sponza. (e) 1^2 - and (f) 32^2 -spp renderings of the airplane's tail fin and refinement pattern as returned by (g) PN-Airplane and (h) SS-Variance.

rule through the network. The network is trained for 8 epochs on a dataset of 16 million samples, starting with a learning rate of 0.001 that is reduced by a factor of 0.1 after 4 and 7 epochs respectively.

2.4. Preprocessing & Rendering

In order to support realtime-performance during rendering, the per-voxel corner patch codes are precomputed and stored for all SVOs that are about to be rendered. This is performed by traversing each SVO and applying E_{patch} to the patches of all allocated voxel corners. Note that since E_{patch} only processes small patches of geometry, it can be readily applied to new models without costly retraining. The patch codes \mathcal{Z} are baked into the SVO, and at each intersection the codes at the corners of the intersected voxels are read.

3. Results

We evaluate PatchNet on four scenes (see Fig. 2a-d): the San Miguel scene, a bisected Sponza scene, and detailed CAD models of an airplane and power plant. Except for San Miguel, direct, diffuse lighting from two antipodal directional light sources is baked into voxel luminance. For San Miguel, we remove most plants and bake global illumination effects with the Cycles renderer of Blender. Additionally, camera tracks of 2000 frames per scene are rendered in a resolution of 130^2 pixels. All tracks start far away from geometry, close in over 1000 frames and then explore the scene's detailed geometry for another 1000 frames.

Different refinement strategies are compared by fixing a per frame budget of $X\%$ of active pixels—i. e. pixels that intersect geometry or have intersecting neighbors—that may be refined at least once. Refinement quality is measured by computing the MSE over all active pixels in any frame, with refined pixel errors being zero and unrefined pixel errors being the difference between rendering with 1^2 -spp and 32^2 -spp. Evaluations with other image metrics such as PSNR, SSIM and LPIPS yield similar results.

Fig. 3 shows the MSE values for each camera track when varying the per frame budget between 0% and 100%. Our PN-[Model] strategies are implemented by evaluating PatchNet, trained on patches of [Model], for each pixel intersecting the SVO. Pixels with high values of $|1 - \sigma_N| + \lambda \cdot |L_V - L_N|$, where $\lambda = 5$, are refined first. We compare against the—according to our findings—best performing classical screen-space based strategy SS-Variance [LRU85] that refines pixels showing a high variation of luminance in their surrounding 3×3 kernels first, as well as a world-space based strategy

WS-LUT that—for each pixel—looks up an error estimate based on the child mask of the intersected voxel. View-dependent error estimates are precomputed by rendering allocated child voxels from many views, and then stored as nine spherical harmonics coefficients for each of the 2^8 possible child configurations. We consider WS-LUT as a classical analog of our approach in which PatchNet is exchanged by a look up table. Both, SS-Variance and WS-LUT are implemented by us. Further, we evaluate against pretrained screen-space networks of Weiss et al. [WITW20] and Kuznetsov et al. [KKR18]. Both networks learn an intermediate sampling map from a low quality rendering and utilize new samples drawn according to the sampling map to enhance the results of a subsequent reconstruction/denoising pass. We feed in the 1^2 -spp rendering (plus G-buffers) and utilize the sampling map to decide which active pixels are refined first. Note that compared to the original approach by Weiss et al., we disable temporal coherence due to the lack of a flow field, and downsample the 4^2 -super-sampled sampling map by summing over 4×4 pixel blocks. Lastly, we also plot the optimal refinement strategy that assumes knowledge of the 32^2 -spp rendering and ranks pixels w.r.t. their deviation to the 1^2 -spp rendering.

The results show that our approach yields superior sampling patterns compared to Weiss et al. [WITW20] and Kuznetsov et al. [KKR18]. However, this is not entirely surprising, since their sampling masks were trained for different rendering algorithms. When refining at least 13% of active pixels, PatchNet performs consistently better than all reference methods, going as far as yielding half the MSE at 30% pixels for Airplane when comparing against SS-Variance. Further, except for San Miguel, PatchNet performs almost equally well independently of the dataset it was trained on. That is, PatchNet generalizes to other models. For San Miguel, training on other datasets yields notable worse results, although still being better than SS-Variance. We argue that San Miguel is the most complex dataset of all, including rich geometry, texture and advanced lighting. Hence, one can generalize *from* San Miguel, but not necessarily *towards* San Miguel.

Fig. 2e-h shows that PatchNet can refine porous geometry such as the airplane's tail fin. However, SS-Variance fails to detect the fine grid structure as it is lost in the coarse voxel representation rendered to screen. Similarly, as shown in Fig. 4, it cannot detect erroneous renderings of table surfaces in the San Miguel scene, where the unlit tabletop bleeds dark color into the bright tablecloth it is covered by. PatchNet detects the errors up to a certain distance, depending on the look-ahead parameter $\Delta\ell$ of PatchNet.

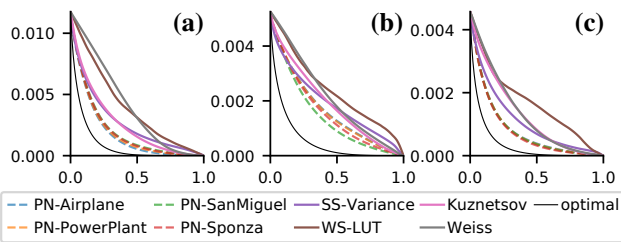


Figure 3: Evaluations on (a) Airplane, (b) San Miguel and (c) Sponza showing refinement quality in MSE for various refinement strategies plotted over per frame budgets from 0% to 100%. Results on Power Plant are similar to Airplane. Lower means better.



Figure 4: San Miguel courtyard rendered at various distances from close (block 1) to far (block 3). First image in block indicates the actual error between 1²-spp and 32²-spp renderings. Other images in block depict refinement patterns (redder areas are refined first) of strategies that fail (Image 2) or succeed (Image 3) in detecting errors at tablecloths. SS-Var stands for SS-Variance. PN- Δx represents PatchNet trained with a look-ahead of $\Delta\ell = x$.

4. Conclusion

We have presented an encoder/decoder network PatchNet that learns model independent patch codes to predict super-sampling patterns from coarse-scale geometry. Our experiments have shown that patterns predicted by our architecture are significantly more effective than those of reference methods, in particular for medium to high refinement counts, and that PatchNet generalizes to models not seen during training.

In the future, we intend to reduce inference timings during rendering by storing and traversing the SVO on the GPU, as well as evaluating the decoder in shared GPU memory. Nonetheless, we expect that inference timings will be inferior to brute-force 4x-super-sampling when casting primary rays only. As a next step, we intend to utilize PatchNet in recursive ray tracing applications, where samples are far more costly to acquire. This makes brute-force super-sampling less attractive than invoking PatchNet to identify dispensable samples. Lastly, we plan to apply model independent patch codes for model compression, learning spatially varying BSSRDF models, and predicting cone split events at rough and/or pointy surfaces in differential cone-tracing.

References

[BLRW16] BROCK A., LIM T., RITCHIE J. M., WESTON N.: Generative and Discriminative Voxel Modeling with Convolutional Neural Networks. [arXiv:1608.04236v2](https://arxiv.org/abs/1608.04236v2). 2

[CKF*21] CHEN Z., KIM V. G., FISHER M., AIGERMAN N., ZHANG H., CHAUDHURI S.: DECOR-GAN: 3D Shape Detailization by Conditional Refinement. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 15735–15744. 1

[GKJ*21] GARBIN S. J., KOWALSKI M., JOHNSON M., SHOTTON J., VALENTIN J.: FastNeRF: High-Fidelity Neural Rendering at 200FPS. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 14346–14355. 1

[JSM*20] JIANG C., SUD A., MAKADIA A., HUANG J., NIESSNER M., FUNKHOUSER T.: Local Implicit Grid Representations for 3D Scenes. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 6000–6009. 1

[KKR18] KUZNETSOV A., KALANTARI N. K., RAMAMOORTHY R.: Deep Adaptive Sampling for Low Sample Count Rendering. *Computer Graphics Forum* 37, 4 (2018), 35–44. 1, 3

[LGZL*20] LIU L., GU J., ZAW LIN K., CHUA T.-S., THEOBALT C.: Neural Sparse Voxel Fields. In *Advances in Neural Information Processing Systems* (2020), vol. 33, pp. 15651–15663. 1

[LK11] LAINE S., KARRAS T.: Efficient Sparse Voxel Octrees. *IEEE Transactions on Visualization and Computer Graphics* 17, 8 (2011), 1048–1059. 1

[LKC*20] LIU H.-T. D., KIM V. G., CHAUDHURI S., AIGERMAN N., JACOBSON A.: Neural Subdivision. *ACM Trans. Graph.* 39, 4 (2020). 1

[LRU85] LEE M. E., REDNER R. A., USELTON S. P.: Statistically Optimized Sampling for Distributed Ray Tracing. *SIGGRAPH Comput. Graph.* 19, 3 (1985), 61–68. 1, 3

[Mit87] MITCHELL D. P.: Generating Antialiased Images at Low Sampling Densities. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (1987), SIGGRAPH '87, pp. 65–72. 1

[MST*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHY R., NG R.: NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Computer Vision – ECCV 2020* (2020), pp. 405–421. 1

[NMOG20] NIEMEYER M., MESCHEDER L., OECHSLE M., GEIGER A.: Differentiable Volumetric Rendering: Learning Implicit 3D Representations Without 3D Supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020). 1

[PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019). 1

[RFS03a] RIGAU J., FEIXAS M., SBERT M.: Entropy-based Adaptive Sampling. In *Graphics Interface* (2003), vol. 2, pp. 79–87. 1

[RFS03b] RIGAU J., FEIXAS M., SBERT M.: Refinement Criteria Based on f-Divergences. In *Proceedings of the 14th Eurographics Workshop on Rendering* (2003), EGRW '03, pp. 260–269. 1

[SZW19] SITZMANN V., ZOLLHOEFER M., WETZSTEIN G.: Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations. In *Advances in Neural Information Processing Systems* (2019), vol. 32. 1

[TLY*21] TAKIKAWA T., LITALIEN J., YIN K., KREIS K., LOOP C., NOWROUZSAHRAI D., JACOBSON A., MCGUIRE M., FIDLER S.: Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes. In *CVPR* (2021), pp. 11353–11362. 1, 2

[WITW20] WEISS S., IŞIK M., THIES J., WESTERMANN R.: Learning Adaptive Sampling and Reconstruction for Volume Visualization. *IEEE Transactions on Visualization and Computer Graphics* (2020). 1, 3

[XSXZ07] XU Q., SBERT M., XING L., ZHANG J.: A Novel Adaptive Sampling by Tsallis Entropy. In *Computer Graphics, Imaging and Visualisation* (2007), pp. 5–10. 1

[YKM*20] YARIV L., KASTEN Y., MORAN D., GALUN M., ATZMON M., RONEN B., LIPMAN Y.: Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance. In *Advances in Neural Information Processing Systems* (2020), vol. 33, pp. 2492–2502. 1