DeepSurfels: Learning Online Appearance Fusion

Marko Mihajlovic\textsuperscript{1} Silvan Weder\textsuperscript{1} Marc Pollefeys\textsuperscript{1,2} Martin R. Oswald\textsuperscript{1}
\textsuperscript{1}Department of Computer Science, ETH Zurich \textsuperscript{2}Microsoft Mixed Reality and AI Zurich Lab

OnlineReconstruction.github.io/DeepSurfels

Abstract

We present DeepSurfels, a novel hybrid scene representation for geometry and appearance information. DeepSurfels combines explicit and neural building blocks to jointly encode geometry and appearance information. In contrast to established representations, DeepSurfels better represents high-frequency textures, is well-suited for online updates of appearance information, and can be easily combined with machine learning methods. We further present an end-to-end trainable online appearance fusion pipeline that fuses information from RGB images into the proposed scene representation and is trained using self-supervision imposed by the reprojection error with respect to the input images. Our method compares favorably to classical texture mapping approaches as well as recent learning-based techniques. Moreover, we demonstrate lower runtime, improved generalization capabilities, and better scalability to larger scenes compared to existing methods.

1. Introduction

Realistic 3D model reconstruction from images and depth sensors has been a central and long-studied problem in computer vision. Appearance mapping is often treated as a separate post-processing step that follows 3D surface reconstruction and is usually approached using batch-based optimization methods \cite{8, 9, 26} that are unsuitable for many applications that do not have access to the entire dataset at processing time, for instance, robot navigation \cite{8, 9, 26}, augmented reality \cite{51, 68}, and virtual reality \cite{14, 43, 44} applications, Simultaneous Localization and Mapping (SLAM) systems \cite{93}, online scene perception methods \cite{28, 69}, and many others.

Common online fusion methods like KinectFusion \cite{50} are well suited for online geometry fusion and can efficiently handle noise and topological changes. However, due to their high memory requirements at high voxel resolutions, they have strong limitations when it comes to encoding high-frequency appearance details on the surface. On the other hand, meshes with high-resolution texture maps \cite{19, 23, 87} are well-suited for encoding high-frequency appearance information in an efficient manner, but they have difficulties in handling topology changes in an online reconstruction setting. Moreover, recent learning-based approaches \cite{49, 54, 72, 73} have achieved high-quality results by learning geometry and texture mapping directly from RGB images. However, they are not well suited for local online updates, do not scale to large-scale scenes, and easily overfit to the training data.

In this paper, we approach the problem of online appearance reconstruction from RGB-D images by combining the advantages of 1) implicit grids, which easily handle topological changes and where low resolution is often sufficient to encode the scene topology, 2) scalable high-frequency appearance along the surface via texture maps or learned feature maps, and 3) a learned scene representation to build a framework for learning-based appearance fusion that allows for online processing and scalability to large scenes. To this end, we propose a novel scene representation DeepSurfels and an efficient learning-based online appearance fusion pipeline which is illustrated in Figure 1.

Our DeepSurfels representation is a hybrid between an implicit surface that encodes the topology and low-frequency geometric details and a surfel representation that encodes high-frequency geometry and appearance information in form of surface-aligned patches. These patches are arranged in a sparse grid and consist of surface-aligned texels that encode appearance information \textit{either} in the classical form of RGB color values \textit{or}, as proposed, via learned feature vectors. The sparse grid allows for efficient volumetric rendering and enables explicit scene updates that are crucial for online fusion, while the 2D patches enable quadratic memory storage complexity like meshes or sparse grid structures. Depending on the DeepSurfels parameters it can approximate between simple colored voxels (high grid resolution, \(1 \times 1\) patches) and textured meshes with high texture atlas resolutions (lower grid resolution, higher patch resolution). Our online appearance fusion pipeline iteratively fuses RGB-D frames into estimated DeepSurfels geometry and is optimized by using a differentiable renderer for self-supervision and the reprojection error as train-
Appearance Fusion

- **DeepSurfels.** A novel scalable and memory-efficient 3D scene representation closing the gap between traditional interpretable and modern learned representations.
- **Online Appearance Fusion Pipeline.** An end-to-end differentiable and efficient online appearance fusion pipeline compatible with classical and learned texture mapping. The method yields competitive texturing results without heavy optimization as every input frame is processed only once with a single network forward pass.
- **Generalized Novel View Synthesis.** Contrary to other learning-based methods [49, 72, 73] that overfit onto a single scene, our method generalizes to new scenes without retraining.

2. Related Work

Our method relates to, and builds upon previous work on scene representations and appearance estimation which are reviewed in the following subsections.

2.1. Scene Representations

Scene representations can be broadly divided into explicit geometric and learned representations.

**Explicit Geometric Representations.** The major advantage of explicit geometric representations is their direct interpretability. **Point clouds** [1, 20, 56] are a lightweight and flexible 3D representation being the raw output of many 3D scanners, RGB-D cameras, and LiDARs. However, they are less suitable for the extraction of watertight surfaces due to lacking topology and connectivity information. This also impedes realistic rendering with detailed textures and complex lighting. **Mesh** representations [29, 33, 41, 89] scale well and texture mapping is convenient. However, topological changes are difficult to handle in an online process. **Voxel-grids** [10, 24, 40, 62, 77, 97] – as a natural extension of pixels to 3D space – easily handle topological changes but are difficult to use with textures and com-
plex light models. Another problem arises from the cubic memory complexity of the dense representation, which makes it expensive to capture precise shape details of complex objects. Surface elements (Surfels) [59, 70, 88, 94] are non-connected point primitives that reduce geometry to the essentials needed for rendering, thus being more memory efficient than meshes while still providing good rendering properties. Our DeepSurfel representation provides several advantages over existing explicit representations: 1) it maintains better connectivity information than point clouds and surfels, 2) scales better than voxel grids while still compatible with octree [75, 101, 102] and voxel hashing [32, 53] approaches that improve memory efficiency, 3) provides better rendering quality than point clouds or voxel grids, 4) enables fast rendering, and 5) allows local updates.

**Learned Shape Representations.** Recent learned implicit representations have achieved impressive results in modeling geometry [13, 46, 47, 56]. They learn a neural network to predict the signed distance or occupancy for a given query point. These approaches struggle to scale to larger scenes and to capture high-frequency details as they tend to learn low-frequency functions which often result in over-smoothed geometry [60]. This problem has been approached by the idea of using local features [11, 31, 57, 98] or directly regressing local geometries [91, 92] for improved scalability and representation power of implicit representations. Our proposed DeepSurfels follows this direction using only local learned features for scalability. Unlike most learning-based methods, we do not encode the scene by network optimization. Instead, we train a network to estimate latent features following a data-driven approach, which better generalizes and facilitates online updates.

### 2.2. Appearance Estimation

**Classical Texture Mapping.** The classical way of coloring a surface from a set of input images with known camera pose is to un-project the image information onto the surface and perform a selection or blending operation to fuse the color information [3, 18, 95]. Due to errors in the camera alignment or in the surface geometry, blurry textures or patch seams affect results and additional texture alignment procedures have been proposed [5, 19, 22, 25, 37, 38, 79, 82, 87] to tackle these problems. Better texture mapping results have been achieved with an optical flow-like correction in texture space [19, 23, 87], patch-based optimization [6], or via 2D perspective warp techniques [36]. With significantly more computation effort, it is also possible to better leverage the redundancy of multiple surface observations from different views and to compute super-resolved texture maps via energy minimization [23, 27, 85, 86] or with deep learning techniques [39, 63]. A common strategy is to aggregate appearance information in patches or texture atlases with corresponding coordinates onto a mesh-based surface, while other works use voxel grids [35, 45, 50, 71, 78, 104], or mesh colors [4, 99]. An overview of texture mapping methods with different representations is given in [80, 100].

**Learned Appearance Representations.** Recent learned appearance representations have achieved state-of-the-art results and outperformed most classical texturing methods. They encode visual information into learned features and store them in voxel grids [21, 44, 48, 58, 61, 74], point clouds [2], or meshes [64, 83, 103] which are rendered using neural networks. [54, 55] use a neural network conditioned on geometry to generate a learned texture representation. [52] combines geometry and appearance to generate a joint implicit representations. Worrall et al. [96] learn a disentangled representation to interpret and manipulate learned feature-based scene representations. SRNs [73] encode the scene into a neural network and render novel views using a neural ray-tracer. NeRF [49] inputs the viewing direction together with point coordinates which allows to also model illumination and complex non-Lambertian surfaces. Other works [65, 66] take advantage of local features for higher representation power, while [30, 84] uses appropriate loss terms to correct for geometric misalignment. Recent trends and applications of neural renderers are summarized in [81]. However, these methods are currently limited to fixed-size scenes, do not scale well to larger real-world scenes, or are unsuitable for online processing of appearance information. The global volumetric appearance reconstruction approach [7] additionally separates albedo, roughness, and lighting. Liu et al. [42] present a learned approach for shape and texture reconstruction that linearly fuses shape and color information in a voxel grid as in [17] and post-process the grid with a multi-resolution neural network. However, pure post-processing methods may not be able to revert errors of an incorrect earlier linear fusion.

**Global vs Local Appearance Representations.** Existing learned scene representations can be separated into global and local approaches. [49, 54] are global approaches encoding the scene into a single feature vector or the weights of a neural network, while [72] can be considered a local approach that uses a dense grid of feature vectors. For better scalability and higher representation power, we follow the local direction. Further, the local storage of appearance keeps the updates of the encoded information local. Moreover, it allows to exploit geometric relations better constraining the learning problem for improved generalization.

**Online Appearance Aggregation.** Most texture mapping methods process all input images in a batch-based way after the geometry estimation step and are implemented as a separate post-processing step, whereas only a minority addresses the problem of online appearance reconstruction. A popular work is KinectFusion [50] and related works [36, 45, 104], which estimate surface and appear-
ance information from a stream of RGB-D images. Other works fuse both geometry and appearance information directly into an oriented surfel cloud [70, 88, 94]. The major drawback of these methods is limited capacity to store high-frequency appearance along the surface and low-quality renderings. Therefore, we propose an efficient online appearance estimation pipeline mitigating these limitations.

3. DeepSurfels 3D Scene Representation

We propose DeepSurfels as a powerful, scalable, and easy-to-use alternative to mitigate previously mentioned problems of many scene representations.

Data Structure. DeepSurfels is a set of patches with $L \times L$ texels that can either store color information or learned feature vectors. The elementary building block is an oriented texel $\tau \in \mathbb{R}^6$ that is associated with its weight parameter $\omega$ and is stored on the objects’ surface, where $c$ denotes the number of feature channels. This number can be chosen arbitrarily for learned appearance fusion as suited for the problem setting, while we set $c = 3$ for deterministic RGB texturing. The texels $\tau$ are arranged in an $L \times L$ resolution patch $P_{xyz} : \{i, j \rightarrow \tau_{ij} : i, j \in [1, L]\}$ that is located in a sparse patch grid $P = \{P_{xyz} | x \leq Y, y \leq Z, z \leq Z\}$, where $X, Y, Z$ represent DeepSurfels’ grid resolution. Although the spatial patch size can be chosen arbitrarily, we empirically observed that texturing works best when the patch size is equal to the grid cell size such that there is no overlap between neighboring patches. For efficiency reasons, it is sufficient to store patches only for grid cells that intersect the objects’ surface. However, it is also possible to allocate more layers around the iso-surface to account for noisy geometry as it is common for geometric fusion approaches [50].

Surface Fitting. We propose a recursive algorithm to align each texel $\tau_{ij}$ of the patch with the implicit surface of the geometry. We compute the patch position and orientation from a signed distance function (SDF) representing the Euclidean distance to the closest surface.

Initially, every patch $P_{xyz}$ in the grid $P$ is positioned at the center of its grid cell. Then, the patches are shifted to the closest surface by using the pre-computed SDF, oriented according to the SDF gradient $\nabla$SDF in all $x, y, z$ directions, and rotated to maximize the surface coverage. These patches are subdivided into $\kappa^2$ non-overlapping patches of $\frac{L}{\kappa} \times \frac{L}{\kappa}$ resolution, where $\kappa \geq 2$ is the smallest integer to non-trivially divide $L$. Each sub-patch is aligned again using the SDF field, where we trilinearly interpolate the SDF value at non-integer grid positions. This patch subdivision and alignment is repeated recursively until the resolution reaches $1 \times 1$ when patches represent texels that lie on the isosurface. This process is visually illustrated in Figure 2.

4. Online Appearance Fusion Pipeline

We also propose a pipeline for learning appearance fusion (depicted in Figure 3) that incrementally fuses RGB measurements into DeepSurfels at every time step $t$ and yields DeepSurfel state $\hat{S}_t$. The input to our pipeline are intrinsic $K_t$ and extrinsic $R_t$ camera parameters, an RGB image $I_t \in \mathbb{R}^{H \times W \times C}$, and corresponding depth map $D_t \in \mathbb{R}^{H \times W}$, where $H, W$ and $C$ denote image height, width, and the number of channels respectively. The pipeline consists of four main components detailed in the following.

Differentiable Projection $\Pi$. The projection module renders a super-resolved feature map $\hat{F}_{t-1} \in \mathbb{R}^{H \times W \times C}$, where $k$ is an upsampling factor inspired by [16] to ensure dense coverage of the geometry. There are three steps to render this feature map from already stored scene content.

First, each pixel in the incoming frame $D_t$ is subdivided into $k^2$ distinct sub-pixels $p'_{ij}$ ($i \in [1, kH], j \in [1, kW]$), thus forming an upscaled image grid.

Second, by leveraging camera and depth information, the center of the sub-pixel $p'_{ij}$ is un-projected into the scene. From the un-projected scene point, the closest DeepSurfel texel and all texels within the surrounding $l_\infty$ ball are selected. The size of this ball is chosen proportional to the size of the un-projected sub-pixel in the world space.

Third, an efficient uniform average of the selected texels determines the value of the feature entry $\hat{F}_{t-1}^{i,j} \in \hat{F}_{t-1}$ (1):

$$\hat{F}_{t-1}^{i,j} = \frac{1}{|T'_i|} \sum_{\tau \in T'_i} \tau,$$

where $T'_i$ is the set of selected texels. This algorithm is simple, leverages the grid representation for fast rendering, and can flexibly render further optionally stored features or a surface normal map $\hat{N}_{t-1}$ that we jointly denote as meta features $\hat{M}_{t-1}$. Note that all operations are differentiable
and the selection can be implemented as a differentiable multiplication by an indicator function.

**Fusion Network.** The input image $I_t$ is deterministically upsampled $I_t^k \in \mathbb{R}^{kH \times kW \times C}$ by factor $k$ (nearest-neighbor interpolation) and stacked $\otimes$ with the super-resolved features $F_{t-1} \otimes M_{t-1} \otimes I_t^k$. This stacked representation is embedded into a higher-dimensional feature space by a trainable linear transformation (Feature Embedding module Figure 3). Then, the embeddings are refined by Blending Network that consists of five convolutional layers ($3 \times 3$ kernel size) interleaved with dropout and leaky ReLU activations. This network, based on a small receptive field, produces refined features aware of neighboring information that alleviates the problem of discretization artifacts, which can occur for low DeepSurfels resolutions. Lastly, these features are compressed by Feature Compression $W$ layer to a lower dimensional feature space that is defined by DeepSurfels’ number of channels. The final output is an updated feature map $\hat{F}_t$ that blends old information from $\hat{F}_{t-1}$ with the new appearance information from $I_t$.

**Inverse Projection** $\Pi^{-1}$. While the fusion module and the explicit geometry representation preserve spatial coherency, this module is responsible for integrating the new appearance information in a temporally coherent way. Without temporal coherency, a new observation could overwrite old states minimizing the reprojection error for the current frame while erasing valuable prior information. The inverse projection module $\Pi^{-1}$ integrates the updated feature map $\hat{F}_t$ into the representation $S_{t-1}$ to produce the new state $S_t$. For efficiency reasons, only texel values $\forall \tau_{t-1} \in \bigcup_{i,j} T_{i,j}^{t-1}$ and their weights $\omega_{t-1}$ that were intersected by at least one of the sub-pixels are updated using the following moving average scheme:

$$\tau_t = \frac{1}{\omega_{t-1} + 1} \left( \tau_{t-1} \omega_{t-1} + \frac{\sum_{i,j}^{kH,kW} \hat{F}_t^k I_{\tau_{t-1} \in T_{i,j}^{t-1}}}{\sum_{i,j}^{kH,kW} I_{\tau_{t-1} \in T_{i,j}^{t-1}}} \right),$$

$$\omega_t = \omega_{t-1} + 1,$$

(2)

where $\mathbb{I}_E$ is an indicator function being one, if $E$ is true, and zero otherwise. The texel weights are initialized to $\omega_0 = 0$.

The new state $S_t$ is optimally computed in 2D space without interrupting the gradient flow. This way, the scene is seamlessly stored in RAM or disk and can only be partially loaded and updated, which is crucial for scalability.

**Appearance Rendering Module.** In a first step, this module extracts compressed scene content $S_t$ using $\Pi$ and embeds these features into a higher dimensional space via a transposed linear compressor (Feature Decompression $W^T$) which acts as a regularizer. Pre-computed meta features $M_{t-1}$ are optionally appended and all features are downsampled by a custom masked average pooling with a stride of $k$ and $k \times k$ kernel size, where the mask indicates which features to ignore (features that are empty or located outside the scene space). The current $H \times W$ resolution feature map is passed through the seven-layer convolutional Rendering Network refining features and filling potential holes that occur when the scene representation is sparsely populated. Lastly, the high-level features are decoded to RGB values by (Feature Decoder) three linear layers interleaved with Leaky ReLU activation functions. The final output is activated using HardTanh activation for generating valid normalized RGB values.

**Loss and Optimization.** The entire pipeline is trained end-to-end from scratch until convergence using the reprojection error between the rendered image $\hat{I}_t$ and input image $I_t$ as self-supervision. Thus, the network can learn to optimally fuse and encode appearance information from 2D training data without any ground-truth textures. Our pipeline is trained using a weighted combination of $L_1$ and $L_2$ loss between input image $I_t$ and rendered image $\hat{I}_t$ given by

$$L(I_t, \hat{I}_t) = \frac{1}{C \cdot H \cdot W} \sum_{p \in I_t, \hat{p} \in \hat{I}_t} ||p - \hat{p}||_1 + \frac{1}{2} ||p - \hat{p}||_2 \quad (3)$$

We empirically found that a $1 : \frac{1}{7}$ weight ratio worked best in our experiments. The entire pipeline has less than 0.6M parameters and was optimized using the Adam optimizer [34] with a learning rate of $10^{-4}$ and batch size 1, except for the generalization experiment, where we used 2. Please see the supplementary material for more details.

---

Figure 3. **Overview of our learned appearance fusion pipeline.** The pipeline consists of an Appearance Fusion module that integrates a new RGB measurement $I_t$ into DeepSurfels $S_{t-1}$ and a differentiable Appearance Rendering module that interprets and renders the content of representation for a given viewpoint. White blocks denote deterministic operations, rectangular blocks denote data, rounded rectangular blocks are trainable modules, and $\otimes$ is a feature stacking operation.
5. Evaluation

We evaluate our method by comparing the representation power of both, our learned and our deterministic approach (direct rendering from RGB surfels) with state-of-the-art methods on novel view synthesis tasks. We further demonstrate how our method generalizes to different scenes for a small number of distinct training samples and provide an ablation study to validate the design choices for our model. The supplementary material provides further details.

Datasets. We conduct experiments on datasets generated from Shapenet [12], publicly available human and
Ours deterministic & GT & Ours deterministic & GT & Ours deterministic & GT & Ours deterministic & GT

![Image](image1.png)

Figure 6. **Qualitative results of our model on unseen scenes from ShapeNet** [12]. Compared to deterministic rendering from RGB values, DeepSurfels with learned (3+3)-channel $6 \times 6$ patches on a sparse $32^3$ grid yields significantly more high-frequency details. As it is shown in the close ups, storing learned features in the texels is particularly useful to correct discretization artifacts.

![Image](image2.png)

Figure 7. **Comparison of SRNs** [73] and DeepVoxels [72] to our learned DeepSurfels fusion with a $64^3$ grid of 8-channel $1 \times 1$ resolution feature patches on the synthetic cube dataset from [72]. Our method produces fewer blur artifacts and multi-view inconsistencies and overall yields significantly better images reconstruction results than both baselines. Note that both baselines perform global appearance fusion with unknown geometry.

cat\(^1\) models, the indoor Replica dataset [76], and the cube scene from [72]. Replica dataset images were rendered with Habitat-Sim [67] and all other models with Blender [15].

**Metrics.** We quantify model performances with the following two metrics [90]. **PSNR:** The Peak Signal-to-Noise Ratio is the ratio between the maximum pixel value in the ground-truth image and the pixel-wise mean-squared error between ground-truth and rendered image. **SSIM:** The Structural Similarity Index measures similarity between patches of rendered and ground-truth images. We omit other perceptron based metrics because we are interested in recovering the true pixel value as our fusion approach can be used for more general types of data.

**Novel View Synthesis.** The model is optimized on 500 randomly rendered $512 \times 512$ training images for the cat and human model and the results for a single unseen frontal viewpoint\(^2\) are compared with state-of-the-art batch (Fu \textit{et al.} [23], Texture Fields [54], Waechter \textit{et al.} [87]), and online methods (SurfelMeshing [70], TSDF Coloring [17]) on a $128^3$ grid. The results in Figure 4 demonstrate that our approach compares favorably even to slower batch-based methods in representing high-frequency textures. Figure 7 further shows that our approach does not suffer from blurry artifacts as the recently proposed SRNs [73], or from multi-view consistency issues like DeepVoxels [72]. Note that these approaches jointly estimate geometry and appearance while we only estimate appearance. Table 2 shows the effect of varying the number of channels on the cube dataset for DeepSurfels of $4 \times 4$ patches on a $64^3$ sparse grid.

**Generalization.** Our pipeline scales and generalizes well on realistic room-size scenes. We trained our pipeline on 288 $480 \times 640$ images of one Replica [76] room represented with DeepSurfels of $11$cm voxel size with (3+3)-channel $6 \times 6$ resolution patches. We disentangled 3 color channels to improve generalization. The pipeline evaluation is performed on every 25th unseen frame in a sequence of frames generated by a moving agent in the Habitat Sim [67]. Figure 5 shows results of our trained pipeline on an optimized (left) and a non-optimized (right) scene. Our learned approach outperforms baselines in representing fine details.

We further demonstrate that our pipeline generalizes well when trained on a larger set of distinct scenes. We render $100$ $312 \times 312$ training images from 150 Shapenet [12] car scenes and test the pipeline on 50 unseen scenes by fusing 100 views and evaluating results on additional 60 unseen viewpoints. The whole pipeline is trained to be frame order independent by randomly shuffling scenes and frames after each optimization step. Results on test scenes (Figure 6, Table 1) indicate that our learned approach improves for discretization artifacts and overall yields sharper results which are supported by higher PSNR and SSIM scores.

**Ablation Study.** For the unobserved test car scenes, we quantify in Table 1 the impact of: (i) depth as a meta feature that helps our method to reason about the confidence of updates since pixels with larger depth values are less important; (ii) multi-view consistency regularization that corrects for geometric misalignments and improves interpolation among neighboring viewpoints by adding an additional error signal (3) for a viewpoint closest to the fused frame; (iii) pixel ray directions with surface orientation map to improve reasoning about light information and non-Lambertian sur-

---

\(^1\)3D models from \texttt{freeld.com} and \texttt{turboxqlid.com}.

\(^2\)For a fair comparison with the results of Texture Fields [54].
we present results on unseen correspond-
entangled color channels (denoted as 3+3 in Ta-
ble). For almost all experiments use 3 feature and 3 dis-

(iv) DeepSurfel parameters (grid and patch resolution). For almost all experiments use 3 feature and 3 dis-
entangled color channels (denoted as 3+3 in Table 1) which outperforms the baseline methods. We observe that every attribute (i-iii) improves generalization and that higher DeepSurfel resolution (iv) consistently benefits reconstruction quality.

(v) Lastly, we demonstrate the benefit of explicitly modeling the reprojection of pixel colors via texels which can be seen as subfeatures of a large feature vector stored in every voxel. We compare the proposed 6 × 6 patches, 3+3 channels to 1 × 1 patches, 213+3 channels which amounts to the same number of features per voxel. We argue that the benefit of using more features per voxel quickly saturates unless subfeatures are anchored along the surface and trained with separate pixel data as supported by our results.

Real-world data. In Fig. 8 we present results on unseen real-world data from [45] for which our method yields the most detailed appearance reconstructions.

Runtime. Our method takes 57ms and 21ms for fusing and rendering a single 312 × 312 frame on 323 DeepSurfels with 6-channel patches with resolution 6 × 6 (Table 1). This is significantly faster compared to other deep learning methods that overfit on a single scene. For example, the state-
of-the-art method NeRF [49] requires ~ 2 days for training on a single scene being unable to generalize to other scenes, while our method can easily be used on unseen scenes without any optimization as demonstrated in Figure 5, which is a speed up of over a thousand times on unobserved scenes for comparable or even favorable results.

Limitations. Similar to traditional methods like TSDF Coloring, our method is sensitive to camera-geometry mis-

Alignment which can lead to blurry results. Moreover, we currently do not model view-dependent effects which may result in washed-out colors due to the local feature averaging. When trained on small datasets our method may distort colors that were not seen during training.

6. Conclusion

We introduced DeepSurfels, a novel scene representation for geometry and appearance encoding, that combines explicit and implicit scene representation to improve for scalability and interpretability. It is defined on a sparse voxel grid to maintain topology relations and implements 2D geometry oriented patches to store high-frequency appearance information. We further presented a learned approach for online appearance fusion that compares favorably to existing offline and online texture mapping methods since it learns to correct for typical noise and discretization artifacts.

As future work we consider the joint online fusion of shape and appearance and address some weaknesses of our appearance fusion pipeline such as the limitation in filling large missing parts and rendering translucent surfaces.

Acknowledgments. This research was partly supported by Toshiba and Innosuisse funding (Grant No. 34475.1 IP-ICT).
References


[28] Christian Häne, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, Paul Furgale, Torsten Sattler, and Marc Polle-


ence on 3D Vision (3DV), 2020.


