Voxel-wise UV parameterization and view-dependent texture synthesis for immersive rendering of truncated signed distance field scene model

Soowoong Kim | Jungwon Kang

Abstract

In this paper, we introduced a novel voxel-wise UV parameterization and view-dependent texture synthesis for the immersive rendering of a truncated signed distance field (TSDF) scene model. The proposed UV parameterization delegates a precomputed UV map to each voxel using the UV map lookup table and consequently, enabling efficient and high-quality texture mapping without a complex process. By leveraging the convenient UV parameterization, our view-dependent texture synthesis method extracts a set of local texture maps for each voxel from the multiview color images and separates them into a single view-independent diffuse map and a set of weight coefficients for an orthogonal specular map basis. Furthermore, the view-dependent specular maps for an arbitrary view are estimated by combining the specular weights of each source view using the location of the arbitrary and source viewpoints to generate the view-dependent textures for arbitrary views. The experimental results demonstrate that the proposed method effectively synthesizes texture for an arbitrary view, thereby enabling the visualization of view-dependent effects, such as specularity and mirror reflection.

KEYWORDS

immersive rendering, multiview image processing, texture synthesis, view-dependent texture mapping, volumetric video representation

1 INTRODUCTION

Immersive video [1] has gained widespread attention as a key content of the virtual reality (VR) market owing to technological advancements of 3D capture and display systems. Immersive video technologies allow viewers to immerse themselves in the content by providing freedom of movement inside the content and experience of motion parallax from the viewpoint movement [2]. Recently, the creation of immersive videos using 360° videos [3], multiview + depth (MVD) [4], and plenoptic point clouds [5] has been actively studied by MPEG-I [6].

The truncated signed distance field (TSDF) fusion of multiple depth maps taken from different viewpoints [7–10] is one of the most popular volumetric scene reconstruction methods recently [11]. It is an implicit surface representation of the captured scene, which is represented by the 3D voxel grid and each voxel stores the truncated signed distance [12] between its center and the nearest surface measurement [13]. Furthermore, it can be...
converted into a high-quality mesh representation using the marching cube (MC) algorithm [14], which traverses all cubes formed by eight adjacent voxels and generates a set of triangles in predefined configurations. However, TSDF is not yet implemented as a technology for creating immersive videos owing to the lack of high-quality texture mapping and immersive rendering methods.

More so, one limitation in the mapping of high-quality textures to the TSDF scene model is that the meshes extracted from the TSDF scene models have many triangles, ranging from hundreds of thousands to millions, and conventional UV parameterization [15–17] for unwrapping these huge meshes into a 2D UV (texture) domain is computationally expensive.

To address the aforementioned problem, we proposed a voxel-wise UV parameterization method using a UV map lookup table, specially designed for the mesh extracted using the MC algorithm from the TSDF scene model. Since the UV map lookup table stores the precomputed UV map for each MC configuration, no effort is required for UV parameterization other than to query the lookup table. Additionally, recording the UV coordinates for associating mesh vertices with UV space is unnecessary since they can be found immediately on the UV map lookup table from the MC configuration of each TSDF voxel.

Thus, from our convenient UV parameterization, we proposed a view-dependent texture synthesis method for the TSDF scene model. First, our method generates multiview local texture maps that share an identical UV map for each voxel from multiview color images. Textures taken from different viewpoints are aligned in the UV domain. So, we analyzed the texture intrinsics from multiview local texture maps using texel-wise operations. By leveraging the texel-wise analysis, we extracted and assigned a single view-independent diffuse map and set of weight coefficients representing the variation of view-dependent specular maps for each voxel. During the texture synthesis phase for an arbitrary target view, the set of weights assigned to each voxel is combined with consideration to the location of the target and the source viewpoints, to estimate specular maps of the target view. Subsequently, the estimated specular maps and assigned diffuse maps are voxel-wise added to produce final local texture maps. Our contributions are the following contributions:

- We introduced a novel voxel-wise UV parameterization for the TSDF scene model using the UV map lookup table. Our UV parameterization enables fast, efficient, and high-quality texture mapping without complex computation to obtain UV coordinates.
- We presented a simple diffuse-specular map separation and view-dependent specular map estimation using our texture representation. These techniques allow viewers to observe the variation in the texture of specular surfaces under varying viewpoints.
- To the best of our knowledge, this is the first attempt at a view-dependent texture synthesis for the immersive rendering of the TSDF scene model.

## 2 | RELATED WORKS

Many existing methods have used color volume-based texture mapping for the TSDF scene model [18–21]. The TSDF fusion phase integrates the depth and corresponding color information into the TSDF and the color volume that shares the same voxel grid structure, respectively. Then, the mesh and per-vertex colored mesh are generated using the MC algorithm from the TSDF and assigning interpolated color values from the color volume to all vertices of the mesh, respectively. Rendering of the per-vertex colored mesh only uses three vertex colors to fill the face of the triangle. Here, while the color volume-based texture mapping is easily performed, the quality of the rendered images depends on the voxel resolution. Since the size of the MC triangles depends on the voxel size used, a smaller voxel causing higher voxel resolution is required to represent the high-frequency texture details. However, this produces a higher quality of the rendered image, higher computational complexity, and a huge mesh. Figures 1A and 1B show the rendering results of two per-vertex colored meshes. Each mesh is generated from the TSDF model with a voxel size of 8 and 4 mm and has 44 134 and 162 402 triangles, respectively. Although the per-vertex colored mesh generated from the TSDF with a higher resolution showed better visual quality, triangles that required rendering significantly increased.

However, conventional UV mapping eliminates the dependence of voxel resolution and texture quality [24,25]. The UV map is a bijective mapping from a 3D mesh surface to a 2D UV plane [26], which improves the texture quality by storing more texture information in the UV plane than the per-vertex colored mesh. Figure 1C is the result of mapping the UV texture in Figure 1E to the mesh extracted from the TSDF model having a voxel resolution of 8 mm. Compared with results of color volume-based texture mapping, the UV texture-mapped result shows considerably improved visual quality, even when the same or lower voxel resolution is used. However, conventional UV mapping takes long to perform UV parameterization [16,26], which involves wrapping a target mesh into a 2D UV
plane and storing UV coordinates as attributes on all vertices of a mesh.

To avoid the complex UV parameterization process, the latest methods have been developed for mapping between the 3D surface and texture. Tang and others [27] divided the entire volume into non-overlapping blocks and separately parameterized the 3D surface in each block. Thus, block-based parameterization was performed such that triangles in each block are grouped by normal vectors, each group was projected onto their respective tangent planes, and the projection results were packed into a fixed single 2D plane. Through this simple parameterization, UV coordinates were unnecessary because they could be quickly inferred from the TSDF. Lee and others [28] assigned a 2D regular texture patch for every non-empty voxel. In the texture extraction phase, each texture patch was projected onto an input color image by selecting an axis direction that minimized sampling loss. Subsequently, the textures were extracted and stored within each texture patch for texture allocation or combined with the existing texture for texture optimization. This patch-based texture representation enabled high-quality texture mapping to the TSDF scene model without UV mapping.

Similar to Lee and others [28], we assigned a 2D regular texture patch for every nonempty voxel. However, the associating 3D surface to the texture patch of our proposed method corresponds to a UV map that is generated using the UV map lookup table. UV mapping allows an easy multiview texture analysis because it is possible to collect color values for a point on the 3D surface projected onto different-viewpoint color images [29]. By taking advantage of the UV mapping feature, we considered view-dependent texture mapping techniques that were not previously studied.

### 3 | TRUNCATED SIGNED DISTANCE FIELD FUSION AND MARCHING CUBE ALGORITHM

This section reviews the TSDF fusion and MC algorithm to provide the basic concepts underlying our voxel-wise UV parameterization method.

#### 3.1 Truncated signed distance field fusion

When all $N$ input depth maps $\{D_i\}_{i=1}^N$ taken from known viewpoints $T_i \in \mathbb{R}^3$ with camera intrinsics $\mathcal{K}_i$ are given in advance, TSDF fusion integrates multiview depth information into a TSDF volume $D \in \mathbb{R}^{X \times Y \times Z}$ in a weighted average manner as follows:

$$D(x) = \frac{\sum_{i=1}^{N} w_i(x) d_i(x)}{W(x)},$$  \hspace{1cm} (1)

$$W(x) = \sum_{i=1}^{N} w_i(x),$$  \hspace{1cm} (2)

where $x$ is the position of a voxel defined by its center, and $w_i(x)$ is the fusion weight related to the uncertainty of depth measurement [13]. Many approaches set $w_i(x) = 1$ for voxels that require an update, otherwise they are set to $w_i(x) = 0$ [7]. To obtain a truncated signed distance $d_i(x)$, a signed distance between the measured depth at the voxel at the 2D pixel location and that at the z-coordinate is initially computed. Then, the truncated signed distance is determined using a truncation value $d_{trunc}$:

$$s_i(x) = D_i(\pi(\mathcal{K}_iT_i,x)) - [T_i x]_z, \hspace{1cm} (3)$$

$$d_i(x) = \min(\|s_i(x)\|, d_{trunc}) \cdot \text{sign}(s_i(x)), \hspace{1cm} (4)$$

where $\pi$ normalizes homogeneous 2D pixel coordinates to inhomogeneous coordinates. Thus, $D_i(\pi(\mathcal{K}_iT_i,x))$ is the measured depth when the voxel $x$ is projected onto the depth map $D_i$. Accordingly, $[T_i x]_z$ is the z-coordinate of...
the voxel \( x \) in the \( i \)th camera coordinate system. The signed distance \( s_i(x) \) indicates the distance of the voxel \( x \) to the nearest surface measurement. A positive or zero signed distance indicates that the voxel \( x \) is inside the surface, whereas a negative signed distance indicates that the voxel \( x \) is outside the surface. Since large signed distances are irrelevant in surface reconstruction [12], \( s_i(x) \) can be truncated using \( d_{\text{trunc}} \) so that \( d_i(x) \) ranges from \(-d_{\text{trunc}}\) to \(+d_{\text{trunc}}\).

3.2 Marching cube algorithm

The MC algorithm [14,30] is the most popular technique for creating a triangular mesh from the TSDF representation. It iterates (“marching”) the algorithm over the cubes formed by eight adjacent voxels. In each iteration, the eight voxels of the current cube and twelve edges between the voxels were indexed in the prescribed order (Figure 2), and an 8-bit MC configuration index was obtained by expressing in binary whether the TSDF value of the \( i \)th voxel is greater than or less than zero. Thus, the MC algorithm ensures that one of \( 2^8 = 256 \) possible configurations (Figure 3A) is assigned for each zero-indexed voxel of every cube. Configuration 0 or 255 indicates empty cubes because all eight voxels lie inside or outside the surface.

Each entry in MC lookup table [31] contains an edge sequence representing the mesh topology of the corresponding configuration. For example, given a MC configuration index of 3, the MC lookup table returns the following sequence: \{1, 8, 3, 9, 8, 1, -1, -1, -1, -1, -1, -1, -1\}. This sequence indicates the vertices placed on edges 1, 8, and 3, and edges 9, 8, and 1 are connected to produce two triangles. The 3D coordinate of each vertex is interpolated using the 3D coordinates and TSDF values of voxels at the edge where the vertex is placed. When a vertex \( p \) is placed on the edge where voxels \( x_i \) and \( x_j \) are connected, the 3D coordinates of \( p \) are linearly interpolated as follows:

\[
p = x_i + \frac{D(x_j)(x_i - x_j)}{D(x_j) - D(x_i)}.
\]

The MC algorithm sequentially extracts a local mesh for each cube using the mesh topologies predefined in the MC lookup table and the 3D coordinates of the vertices. Finally, the local meshes of the cubes are merged to generate the complete mesh for the entire scene.

4 Voxel-wise texture map generation

The TSDF scene model visualization is achieved using the widely employed per-vertex colored mesh generation from TSDF and color volume [18–21]. However, the rendered image of a per-vertex colored mesh is blurry and has a low visual quality because the number of pixels that each triangle of the mesh renders is greater than the number of colors mapped to the triangle. Therefore, instead of the color volume, we employed full texture mapping based on UV mapping to enhance the visual quality of the TSDF scene model. Contrary to conventional UV mapping, our proposed method performs UV parameterization to generate a local UV map for the local surface of each cube using a UV map lookup table.

4.1 UV map lookup table

The MC algorithm extracts one to five triangles for each nonempty cube from a TSDF scene model, resulting in the final mesh having many triangles. For example, 400K triangles were extracted for our smallest scene model (Room-Near with 8 mm voxel res.) and 15M triangles for the biggest model (Rest.-Far with 4 mm voxel res.). To compute the UV map, we employed several steps to unwrap these huge meshes onto a 2D plane using traditional UV parameterization. These steps include [32] mesh simplification, mesh chartification, chart parameterization, and atlas packing, which is computationally expensive. To mitigate this, we designed a UV map lookup table consisting of 256 UV maps that respond to each topology configuration since the MC algorithm uses the predefined 256 topology configurations (Figure 3).

The proposed UV map lookup table consists of a precomputed 256 UV map. Wherein, each entry in the lookup table consists of a set of UV triangles, and a UV triangle is represented by three UV coordinates in the range \([0, 1]\). For example, with MC configuration index 3, our UV map lookup table returns the following five sets of UV coordinates:
\{(1.0,1.0),(0.0,0.0),(1.0,0.0)\},
\{(0.0,1.0),(0.0,0.0),(1.0,1.0)\},
\{(-1.0,-1.0),(-1.0,-1.0),(-1.0,-1.0)\},
\{(-1.0,1.0),(-1.0,-1.0),(-1.0,-1.0)\},
\{(-1.0,-1.0),(-1.0,0.0),(-1.0,0.0)\},
\{(-1.0,-1.0),(-1.0,1.0),(-1.0,1.0)\}\}. 

This set represents the first triangle of the 3rd MC configuration which connects the vertices on edges 1, 8, and 3 is mapped to a triangle on the 2D UV domain with coordinates of (1.0, 1.0), (0.0, 0.0), and (1.0, 0.0). Likewise, the second triangle of the 3rd MC configuration connecting the vertices of edges 9, 8, and 1 is mapped to a UV triangle connected by (0.0, 1.0), (0.0, 0.0), and (1.0, 0.0). Like-wise, the second triangle of the 3rd MC configuration connecting the vertices of edges 9, 8, and 1 is mapped to a triangle on the 2D UV domain with coordinates of (1.0, 1.0), (0.0, 0.0), and (1.0, 0.0). The remaining −1 filled set is dummy entries, which indicates that there are no more triangles.

To obtain a precomputed UV map, 256 template meshes of the MC algorithm were constructed and parameterized. For each MC configuration, the parameterization was performed using the following steps: ① First, we connected vertices located at the center of each edge of the unit cube according to the MC configuration to form the template mesh. ② When the template mesh has a complex geometry, it is subdivided into two or three mesh pieces based on angular changes of 66°; ③ Each mesh piece (or non-divided template mesh) is projected onto the tangent plane orthogonal to the average normal. ④ Finally, the 2D projection results are arranged inside a common 2D unit plane to create the UV map. After parameterization, the UV lookup table comprises all 2D vertex coordinates of each projected template mesh in the corresponding entry. Figure 3 shows 15 basic cases of 256 MC configurations and the corresponding UV maps in the UV map lookup table.

4.2 Local texture map generation

Each voxel is assigned its corresponding UV map in the UV map lookup table using the assigned MC index. The real mesh of each voxel extracted by the MC algorithm and the template mesh constructed to create a corresponding UV map are different but share the same topology. Here, all 3D triangles in the local surface of each voxel and 2D triangles of the corresponding UV map are uniquely mapped. Therefore, UV texture maps for the surface of every nonempty voxel were generated by referring to their UV maps.

Suppose \(\mathbf{t}_{x,i}\) denotes \(S \times S \times 3\) local texture map for observing the voxel \(X\)'s surface from the viewpoint \(i\). Each texel color of \(\mathbf{t}_{x,i}\) can be obtained by projecting the 3D point associated with the texel point onto the color image \(C_i\):

\[
\mathbf{t}_{x,i}(\mathbf{q}) = C_i(\pi(X; T; \mathbf{p}))
\]

where \(\mathbf{q}\) is a 2D texel point and \(\mathbf{p}\) is the associated 3D point on the voxel \(X\)'s surface. The 3D world coordinates associated with 2D texel coordinates can be computed using the barycentric mapping, and vice versa [25].

To generate a common local texture map \(\bar{\mathbf{t}}_x\) that applies to any viewpoint, we used the visibility-weighted average of local texture maps from all \(N\) input viewpoints (Figure 4) as follows:

\[
\bar{\mathbf{t}}_x = \left(\sum_{i=1}^{N} \mathbf{t}_{x,i} \odot \mathbf{m}_{x,i}\right) \odot \left(\sum_{i=1}^{N} \mathbf{m}_{x,i}\right)
\]

where \(\odot\) and \(\odot\) denote texel-wise multiplication and division, respectively. \(\mathbf{m}_{x,i}\) is a binary matrix representing the visibility of each texel of \(\mathbf{t}_{x,i}\) and obtained using the Z-buffer algorithm.

Each averaged local texture map was assigned to its corresponding voxel. However, UV coordinates need not be stored separately because they are embedded in the UV lookup table. Therefore, to render the TSDF scene model, the MC configuration and mesh for each voxel of the TSDF were recalculated. First, the UV map was retrieved from the UV lookup table and the assigned

![Figure 3](image-url) (A) 15 basic cases of the MC algorithm and (B) 15 corresponding UV maps in the UV map lookup table. (C) Proposed UV parameterization is performed by assigning a predefined UV map for each nonempty voxel.
local texture was mapped to the mesh by referring to the UV map.

### 4.3 Texture packing

Since our method extracts textures locally and separately for each voxel located on the surface, creating a single global texture map requires a method of packing the textures. Here, we arranged local texture maps of (7) in square space according to the 3D Morton code [22] of the voxel position (Figures 1F and 6C). By employing 3D Morton code, the packing method easily calculated the location of a local texture map in a global texture map, while most conventional packing methods require storing the locations of the local texture map.

### 5 VIEW-DEPENDENT TEXTURE SYNTHESIS

For texture mapping of the TSDF scene model, color volume-based and recent methods [27,28] have generated a single texture model from multiview images that can be mapped onto the surface of the TSDF scene model independent of the rendering viewpoint. Although, a viewpoint-independent texture model can be optimized to render all viewpoints in high quality, in some cases, such as rendering glossy and mirror-like surfaces, it is impossible to show changes in texture as viewpoint changes, thereby reducing the viewer’s sense of reality.

To deal with this problem, we proposed a method to synthesize view-dependent texture using view-dependent specular map estimation. To estimate the view-dependent specular map, a simple diffuse-specular map separation was performed using our texture representation, followed by an eigenspace analysis of the separated specular maps.

#### 5.1 Diffuse-specular map separation

The dichromatic reflection model [33] describes light reflected from an object surface as the sum of the diffuse reflection independent of the viewpoint and specular reflection dependent on the viewpoint. We separated the local texture maps generated for $N$ input views at the location of a voxel $\mathbf{x}$ into a single diffuse texture map and $N$ specular maps by applying the dichromatic reflection model. (Some voxels are assigned less than $N$ local texture maps because they are invisible from some viewpoints, but we assumed that all nonempty voxels were assigned $N$ local texture maps to simplify the notation.) With the averaged texture $\mathbf{t}_x$ calculated using (7) considered a diffuse texture map, the specular texture map $\mathbf{t}_{x,i}$ for an input viewpoint $i$ and voxel $\mathbf{x}$ is calculated as follows:

$$\mathbf{t}_{x,i} = \frac{\mathbf{t}_{x,i} - \mathbf{t}_x}{C_0} \cdot \mathbf{m}_{x,i}. \quad (8)$$

Since texel values at the same location of local texture maps $\mathbf{t}_{x,1}, ..., \mathbf{t}_{x,N}$ are color values observed on $N$ different viewpoints for the same 3D point, we can calculate specular maps defined in the dichromatic reflection model using the texel-wise subtraction of the local texture maps and the diffuse texture map (Figure 4).
5.2 View-dependent specular map estimation

To render an image of arbitrary viewpoint \( \theta \) in our texture mapping, the view-dependent specular map \( \mathbf{t}_{x,\theta} \) was added to the view-independent diffuse map \( \mathbf{t}_x \) to obtain a local texture map \( \mathbf{t}_{x,\theta} \) for the surface of the voxel \( x \) visible at the viewpoint \( \theta \):

\[
\mathbf{t}_{x,\theta} = \mathbf{t}_x + \mathbf{t}_{x,\theta}. \tag{9}
\]

If an arbitrary viewpoint does not belong to the set of input viewpoints, the specular map \( \mathbf{t}_{x,\theta} \) is estimated from the available specular maps.

To estimate the specular map, the full voxel grid was divided into non-overlapping voxel blocks and the distribution of the specular maps for each voxel block was modeled. Let \( \mathcal{X} = \{ \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M \} \) be a set of \( M \) nonempty voxels in a \( K \times K \times K \) voxel block. All specular maps associated with the voxel set \( \mathcal{X} \) are spread out into column vectors and stacked horizontally to create data matrix \( \mathbf{T}_\mathcal{X} \):

\[
\mathbf{T}_\mathcal{X} = [\mathbf{t}_{x_1,1}, \mathbf{t}_{x_1,2}, ..., \mathbf{t}_{x_M,K}], \tag{10}
\]

where \( \mathbf{t}_{x_i,j} \) is a column-vectorized representation of a specular map and has a length of \( L = S \times S \times 3 \). Therefore, the data matrix \( \mathbf{T}_\mathcal{X} \) has a size of \( L \times M \times N \). Now, we performed singular value decomposition (SVD) \([34]\) on the covariance matrix of specular vectors \( \mathbf{T}_\mathcal{X}^\top \mathbf{T}_\mathcal{X} \) to determine the eigenvectors and their corresponding eigenvalues:

\[
\mathbf{T}_\mathcal{X}^\top \mathbf{T}_\mathcal{X} = \mathbf{U}_\mathcal{X} \mathbf{V}_\mathcal{X} \mathbf{U}_\mathcal{X}^\top, \tag{11}
\]

where \( \mathbf{V}_\mathcal{X} \) is a diagonal matrix containing the eigenvalues of \( \mathbf{T}_\mathcal{X}^\top \mathbf{T}_\mathcal{X} \), assuming that the eigenvalues are sorted from the largest to the smallest. \( \mathbf{U}_\mathcal{X} \) is an orthonormal matrix in which columns are eigenvectors of \( \mathbf{T}_\mathcal{X}^\top \mathbf{T}_\mathcal{X} \). Thus, each individual specular vector \( \mathbf{t}_{x_i,j} \) can be represented by a linear combination of the eigenvectors:

\[
\mathbf{s}_{x_i,j} = \mathbf{U}_\mathcal{X}^\top \mathbf{t}_{x_i,j}, \tag{12}
\]

\[
\mathbf{t}_{x_i,j} = \mathbf{U}_\mathcal{X} \mathbf{s}_{x_i,j}. \tag{13}
\]

The projection of the specular vector onto the eigenspace yields the specular weights \( \mathbf{s}_{x_i,j} \), which linearly approximates the specular vector \( \mathbf{t}_{x_i,j} \).

Given an arbitrary viewpoint \( \theta \), we selected the three closest input viewpoints \( i, j, \) and \( k \) to an arbitrary viewpoint in Euclidean distance, and computed the barycentric weights \( \omega_i, \omega_j, \) and \( \omega_k \) \([29]\). From the viewpoint \( \theta \), the approximate estimates of the specular vector of voxel \( x \) were calculated using the barycentric weights as follows:

\[
\mathbf{s}_{x,\theta} = \omega_i \mathbf{s}_{x_i,1} + \omega_j \mathbf{s}_{x_j,1} + \omega_k \mathbf{s}_{x_k,1}, \tag{14}
\]

\[
\mathbf{t}_{x,\theta} = \mathbf{U}_\mathcal{X} \mathbf{s}_{x,\theta}. \tag{15}
\]

Finally, the resulting specular vector \( \mathbf{t}_{x,\theta} \) was reshaped to convert the estimated specular map \( \mathbf{t}_{x,\theta} \).

By assigning the diffuse map and set of specular weights to each nonempty voxel and storing the eigenvectors of each voxel block, we estimated local specular maps and generated local texture maps for an arbitrary viewpoint at rendering time.

6 EXPERIMENTAL EVALUATION

Datasets. We created four multiview RGB-D datasets of static indoor scenes by capturing two synthetic models twice, as shown in Figure 5. A virtual 9 × 9 RGB-D camera array with a fixed baseline (6 cm) captured each scene near and far from the wall. Each dataset consists of 81 8bit-RGB and 16bit-depth maps. All RGB images and depth maps have a resolution of 3840 × 2160.

Truncated signed distance field and texture generation. Two TSDF models with voxel resolutions of 4 mm and 8 mm were created for each dataset using the TSDF fusion described in Section 3.1. So, eight TSDF scene models were used. Also, corresponding color volumes were integrated for comparison with our texture.
representation. For our local texture map generation and specular map modeling, we set the size of the local texture map and voxel block size to $8 \times 8$ and $8 \times 8 \times 8$, respectively.

**Source view reconstruction.** To quantitatively evaluate the texture quality, we measured the PSNR of the reconstructed source views using color volume, our view-independent texture of (7), and view-dependent texture of (9). In this paper, we did not consider conventional UV texture mapping for comparison due to the following reason: When the LSCM parameterization [15], which a conventional UV texture mapping, was performed on the test scene models using the blender [35], it produced severe rendering artifacts that significantly reduced PSNR by generating overlapped UVs (Figure 6).

PSNR was measured from the reconstructed views of the $5 \times 5$ viewpoints in the center of the $9 \times 9$ source viewpoints of each dataset. Table 1 shows the average PSNR for each texture representation. The view-dependent texture showed produced the best results, followed by the view-independent texture which showed better results than color volume, even if the resolution of the voxel was low. Even with a higher voxel resolution, the color volume and view-independent texture did not significantly improve the source view reconstruction performance. However, a great improvement was recorded with the use of our view-dependent texture.

In Figure 7, the results of rendering highly specular surfaces using color volume and view-independent texture were heavily blurred and did not render specular effects. However, our view-dependent textures rendered both specular effects and texture details well.

**Arbitrary view synthesis.** We rendered images of test scene models at arbitrary viewpoints using our view-dependent texture synthesis method. Figure 8 shows the results of rendering $5 \times 5$ arbitrary views while moving $1$ cm in the $XY$ direction from a location, not in the source viewpoints. The changes in the texture of the specular surfaces, such as a metal object and glass, according to the observation viewpoint, were smoothly represented. The synthesized arbitrary view videos are available in YouTube: https://youtu.be/wNN5e3ip3hQ and https://youtu.be/fK51tZ-gO0U.

**Compression usability.** Similar to the experiment in Volino and others [29], we measured the performance change of source view reconstruction by the number of eigenvectors used to estimate the specular maps (Figure 9). For each voxel block, our test setup produced $8 \times 8 \times 3 = 192$ eigenvectors from the matrix $U_N$ in (11). When the number of eigenvectors used to estimate the specular map was reduced to 32, approximately 1-dB PSNR loss occurred in the source view reconstruction. By properly reducing the dimension of the eigenvectors and specular weights, our method can be used for lossy texture compression or texture data reduction.

![Figure 6](image_url) (A) Conventional UV texture map for the mesh generated using the Room-Near scene model. Overlapped UVs are highlighted in orange color. (B) Rendering artifacts caused by overlapped UVs in the conventional texture-mapped result. (C) Proposed local texture maps. (D) Proposed texture-mapped result. The proposed UV parameterization does not produce overlapped UVs.

![Table 1](image_url) Average PSNR of the source view reconstruction

| Method voxel size | Room-Near | Room-Far | Rest.-Near | Rest.-Far | Total Avg. |
|------------------|-----------|----------|------------|-----------|------------|
| Color vol. 8 mm  | 28.18     | 26.46    | 32.89      | 33.79     | 30.33      |
| Color vol. 4 mm  | 29.99     | 28.06    | 34.63      | 35.22     | 31.98      |
| Our view-ind. 8 mm | 33.12     | 31.81    | 40.49      | 38.19     | 35.90      |
| Our view-ind. 4 mm | 33.74     | 32.27    | 40.95      | 39.14     | 36.52      |
| Our view-dep. 8 mm | 36.32     | 37.68    | 44.75      | 48.03     | 41.69      |
| Our view-dep. 4 mm | 40.32     | 42.75    | 49.55      | 53.26     | 46.47      |
Waste-rate of texture space. The texture space wasted in conventional UV texture maps and our texture maps was measured, respectively. Conventional texture maps were produced using UV maps calculated by blender (Figure 6A). Our texture maps were created, as mentioned in Section 4.3 (Figure 6C). On average, 12.81% and 1.94% of the texture space were not used for the conventional UV texture maps and our proposed texture maps. Therefore, compared with a conventional UV map, our texture representation used texture space more efficiently.

Limitation. The synthesized views showed some distracting ghosting artifacts in the boundary regions of two
triangles that were far apart in the 3D space, but adjacent in the image plane. These artifacts are due to the inaccurate geometry of the mesh extracted from TSDF [36]. This problem is solved by increasing the voxel resolution, which in turn increases the geometric accuracy of the mesh. However, the amount of texture data will significantly increase. This tradeoff can easily be resolved if a hierarchical TSDF structure [37] is used instead of a regular voxel grid structure.

7 CONCLUSION

In this paper, we presented a view-dependent texture synthesis for a TSDF scene model using voxel-wise UV parameterization. The proposed voxel-wise UV parameterization assigns the precomputed UV map for each voxel using its MC configuration by referring to the UV map lookup table. Thus, no processing is required to calculate the UV map nor is storage required to record UV coordinates. Furthermore, we estimated textures for an arbitrary view simply and efficiently using the proposed voxel-wise UV parameterization, which showed visually satisfactory results. In the future, we will develop the proposed method to be an efficient representation for compression and immersive rendering of the real-world RGB-D dataset.

CONFLICTS OF INTEREST

The authors declare that there are no conflicts of interest.

FUNDING INFORMATION

Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2017-0-00072, Development of Audio/Video Coding and Light Field Media Fundamental Technologies for Ultra Realistic Tera-media).

ORCID

Soowoong Kim https://orcid.org/0000-0001-8584-834X
Jungwon Kang https://orcid.org/0000-0003-4003-4638

REFERENCES

1. S. Moezzi et al., Immersive video, in Proc. IEEE 1996 Virtual Reality Annu. Int. Symp. (Santa Clara, CA, USA), 1996, pp. 17–24. https://doi.org/10.1109/VRAIS.1996.490506
2. M. Wien et al., Standardization Status of Immersive Video Coding, IEEE J. Emerg. Selected Topics Circuits Syst. 9 (2019), no. 1, 5–17. http://doi.org/10.1109/jetcas.2019.2889848
3. M. Domaski et al., Immersive visual media — MPEG-I: 360 video, in Proc. Virtual Navigation Beyond, 2017 Int. Conf. Syst., Signals Image Process. (Poznan, Poland), May 2017, pp. 1–9. https://doi.org/10.1109/IWSSIP.2017.7965623
4. B. Salahieh, S. Bhatia, and J. Boyce, Multi-pass renderer in MPEG test model for immersive video, in Proc. Picture Coding Symp. (Ningbo, China), Nov. 2019, pp. 1–5. https://doi.org/10.1109/PCS48520.2019.8954515
5. G. Sandri, R. L. de Queiroz, and P. A. Chou, Compression of plenoptic point clouds, IEEE Trans. Image Process. 28 (2019), no. 3, 1419–1427.
6. G. Lafruit, D. Bonatto, C. Tulvan, M. Preda, and L. Yu, Understanding MPEG-I coding standardization in immersive VR/AR applications, SMPTE Motion Imaging J. 128 (2019), no. 10, 33–39.
7. R. A. Newcombe et al., KinectFusion: Real-time dense surface mapping and tracking, in Proc. IEEE Int. Symp. Mixed Augmented reality (Basel, Switzerland), Oct. 2011, pp. 127–136. https://doi.org/10.1109/ISMAR.2011.6092378
8. M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, Real-Time 3D reconstruction at scale using voxel hashing, ACM Trans. Graph. 32 (2013), no. 6, 1–11. https://doi.org/10.1145/2508363.2508374
9. M. Slavcheva, M. Baust, and S. Ilic, Variational Level Set Evolution for Non-rigid 3D Reconstruction from a Single Depth Camera, IEEE Trans. Pattern Anal. Machine Intell. 43 (2021), no. 8, 2838–2850. https://doi.org/10.1109/tpami.2020.2976065
10. L. Yang, Q. Yan, Y. Fu, and C. Xiao, Surface reconstruction via fusing sparse-sequence of depth images, IEEE Trans. Vis. Comput. Graph. 24 (2018), no. 2, 1190–1203.
11. M. Zollhöfer, P. Stotko, A. Görlich, C. Theobalt, M. Nießner, R. Klein, and A. Kolb, State of the art on 3D reconstruction with RGB-D cameras, Comput. Graph. Forum 37 (2018), no. 2, 625–652.
12. D. Werner, A. Al-Hamadi, and P. Werner, Truncated signed distance function: experiments on voxel size, in Proc. Int. Conf. Image Anal. Recogn. (Vilamoura, Portugal), Oct. 2014, pp. 357–364. https://doi.org/10.1007/978-3-319-11755-3_40
13. B. Curless and M. Levoy, A volumetric method for building complex models from range images, in Proc. Annu. Conf. Comput. Graphics Interactive Techniques, Aug. 1996, pp. 303–312. https://doi.org/10.1145/237170.237269
14. W. E. Lorensen and H. E. Cline, Marching cubes: a high resolution 3D surface construction algorithm, SIGGRAPH Comput. Graph. 21 (1987), no. 4, 163–169. https://doi.org/10.1145/37402.37422
15. B. Lévy, S. Petitjean, N. Ray, and J. Maillot, Least squares conformal maps for automatic texture atlas generation, ACM Trans. Graph. (TOG) 21 (2002), no. 3, 362–371.
16. A. Sheffer, E. Praun, and K. Rose, Mesh parameterization methods and their applications, Found. Trends. Comput. Graph. Vis. 2 (2006), no. 2, 105–171. https://doi.org/10.1561/0600000011
17. K. Zhou et al., ISO-charts: stretch-driven mesh parameterization using spectral analysis, in Proc. Eurographics/ACM SIGGRAPH Symp. Geometry Processing, SGP ’04, Association Comput. Machinery (New York, NY, USA), July 2004, pp. 45–54. https://doi.org/10.1145/1057432.1057439
18. E. Bylow et al., Real-time camera tracking and 3D reconstruction using signed distance functions, Robotics: Sci. Syst., 2013, http://doi.org/10.15607/RSS.2013.IX.035
KIM AND KANG

AUTHOR BIOGRAPHIES

Soowoong Kim has been a senior researcher in the Telecommunications & Media Research Laboratory at the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea, since 2019. Before joining ETRI, he worked as an associate researcher engineer in the Identification & Biometrics Section at Suprema Inc. from 2013 to 2015 and as a senior research engineer in the Video Analysis Project at KT (Korea Telecom) Corp, from 2017 to 2019. He received his PhD degree in electrical engineering from the Ulsan National Institute of Science and Technology (UNIST), Ulsan, Rep. of Korea, in 2016. His current research interests are deep learning-based computer vision, immersive video processing, and multimedia applications.

Jungwon Kang received her BS and MS degrees in electrical engineering in 1993 and 1995, respectively, from Korea Aerospace University, Seoul, Rep. of Korea. She received her PhD degree in electrical and computer engineering in 2003 from the Georgia Institute of Technology, Atlanta, GA, US. Since 2003, she has been a principal member of the research staff in the Telecommunications & Media Research Laboratory, ETRI, Republic of Korea. She had made contributions to the HEVC and VVC standards. Her current research interests include video/image compression, immersive video processing, and multimedia applications.

How to cite this article: S. Kim and J. Kang, *Voxel-wise UV parameterization and view-dependent texture synthesis for immersive rendering of truncated signed distance field scene model*, ETRI Journal, 44 (2022), 51–61. https://doi.org/10.4218/etri.2021-0300