

UTexGen: High-quality texture reconstruction for large-scale scenes using multi-view images

Hye-Sun Kim¹  | Yun-Ji Ban²  | Chang-Joon Park¹ 

¹Content Research Division, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea

²Honam Research Division, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea

Correspondence

Hye-sun Kim, Content Research Division, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea.

Email: hsukim@etri.re.kr

Funding information

This work was supported by the Culture, Sports and Tourism R&D Program through the Korea Creative Content Agency, Ministry of Culture, Sports and Tourism, Project Number: R2020040207, Development of large-scale space cloning and high quality real-time reconfiguration technology for realistic content.

Abstract

When reconstructing extensive terrain, it is essential to partition it into smaller tiles for individual processing. This paper introduces a texture reconstruction approach that ensures seamless and consistent final outputs, even when processed tile by tile. Among the stages of multi-view image-based reconstruction, texture reconstruction presents significant challenges during tile-based processing. Relying solely on local tile-level data complicates achieving precise texture mapping. The absence of occlusion details between tiles can lead to selecting incorrect images as the best visible ones or adjusting tile texture colors differently, resulting in noticeable grid-like texture seams in the final result. To mitigate these issues, we leverage global depth maps to accurately detect occlusions between neighboring tiles. Furthermore, by utilizing a shared texture candidate list, we establish uniform targets for texture color correction across tiles. Experimental findings demonstrate that leveraging global information for texture reconstruction on a tile-by-tile basis enables the creation of smooth and realistic texture maps, as validated through comparisons with existing methodologies.

KEYWORDS

large-scale scene reconstruction, multi-view image processing, seamless texture synthesis, texture reconstruction, tile-based processing

1 | INTRODUCTION

Three-dimensional reconstruction can be achieved through a variety of offline and online pipelines that utilize multi-view images or 3D sensor data. Key processes in this domain include camera pose estimation [1], 3D geometry generation using multi-view stereo (MVS) techniques [2,3], and mesh reconstruction via either implicit [4,5] or explicit methods [6,7]. Recent advances in deep learning have significantly enhanced the accuracy of multi-view camera pose estimation and 3D shape

inference from single or multiple images, as well as point clouds [8–12]. A critical final step in these pipelines is texture reconstruction, which plays a vital role in enhancing the realism of the final 3D model. To accurately represent an object, the original image must be precisely mapped onto the reconstructed model using high-quality texture data. While numerous texture reconstruction techniques exist, this paper presents an advanced method specifically designed for large-scale terrains.

The increasing popularity of digital twins for terrains and immersive metaverses has heightened the demand

for highly accurate terrain reconstruction models that closely resemble real-world landscapes. A notable characteristic of these models is their coverage of extensive areas, which poses challenges due to the substantial volume of input data that must be processed and the significant size of the geometry to be reconstructed, resulting in resource constraints during processing. To tackle this issue, we adopted a method that divides large terrains into smaller rectangular tiles for more manageable processing.

For each tile, we employed a well-established high-quality reconstruction pipeline comprising several key steps. First, the structure from motion (SfM) [1] method estimates the camera parameters from drone-captured images to efficiently capture the expansive terrain from above. This is followed by MVS [13], which generates a dense point cloud, and Poisson surface reconstruction [4], which performs meshing. This approach aligns with the prevalent pipeline introduced by Liu et al. [14] for reconstructing large-scale outdoor scenes and is also the method adopted by many commercial reconstruction software applications today.

However, the final stage of Liu et al.'s pipeline employs MVS-Texturing [15], which proves inadequate for large-scale terrain texture reconstruction. As elaborated in subsequent chapters, processing extensive scenes necessitates partitioning large areas into manageable tiles. This method introduces two main challenges: Individual tile processing can lead to visibility check errors due to occlusions between adjacent tiles, and the simultaneous rendering of reconstructed tiles may reveal seams from discrepancies in texture color. These issues arise from local color correction in tile-based texture reconstruction, which often overlooks global color information from neighboring tiles.

To address these challenges, we propose an advanced texture reconstruction method that effectively processes large-scale terrains in tiles. Our approach is versatile and capable of achieving high-quality texture reconstruction for both smaller objects and human figures as well.

2 | RELATED WORKS

Texture reconstruction approaches can be broadly categorized into two types based on the number of source images used to create a texture map for a single face. One approach involves blending two or more input multi-view images, while the other selects the optimal image from the input set for mapping. This section revisits various related approaches that generate realistic texture maps based on these classification criteria.

2.1 | Blending-based method

The blending-based method offers a rapid and straightforward approach to synthesizing texture maps by effectively blending multi-view images. A significant advantage of this method is its ability to create seamless texture maps, smoothing out color variations resulting from lighting effects or differences in camera intrinsic values when blending multiple images. While these methods primarily focus on blending multiple images using various weighted average strategies [16–18], they are susceptible to issues such as ghosting artifacts due to errors in camera estimation or geometry, as well as loss of detail during the blending process from multiple sources. Some approaches have attempted to mitigate these issues by segmenting input images into frequency bands and blending them with different weighting strategies [19,20], but these challenges have not been fully resolved. Despite these challenges, the simplicity and stability of the multi-band blending method have led to its adoption in commercial software such as Metashape [21] and RealityCapture [22].

In real-time reconstruction processing, this method remains widely used and actively researched. For example, in real-time reconstruction, color information from input images is accumulated into a 3D voxel grid according to predefined rules to enhance the realism of the reconstructed geometry [23–25]. A significant drawback of these approaches is the potential degradation in quality due to the limited texture color information that can be stored, depending on the resolution of the voxel grid. Recent efforts have aimed to mitigate this limitation by separating the structures storing geometry volume and texture information, thereby improving the quality of texture reconstruction [26,27].

2.2 | Projection-based method

The projection-based method selects the best view for each face of the 3D geometry, avoiding issues associated with blending but necessitating solutions for determining the optimal view selection and correcting color discrepancies between faces captured from different views. Due to variations in shooting parameters and environmental conditions, multi-view images may capture colors differently, leading to visible seams at boundaries where faces with different views meet. Typically, the optimal view is selected based on criteria such as frontal orientation, proximity, and sharp focus. However, achieving color consistency with neighboring faces requires probabilistic considerations, often implemented using Markov random field models [28,29]. These models focus on selecting the

highest quality view while minimizing seams by improving data and smoothness terms.

Despite efforts to minimize seams, challenges persist when adjacent faces selected from different views introduce color correction needs or require blending boundary pixels [28–30]. Building upon exemplary methods, Waechter et al. introduced enhanced color correction in MVS-Texturing [15], enabling the reconstruction of smooth textures without visible seams, akin to reality. This technique is recognized as a leading technology in high-quality texture reconstruction, widely adopted as a benchmark in numerous studies [13,31–34]. This paper benchmarks MVS-Texturing as a primary technology and focuses on enhancing modules, specifically addressing the texturing challenges of large-scale terrains.

Within the texture reconstruction framework, techniques for warping selected view images to align with underlying geometry are fundamental components. When image projection mapping distorts due to camera drift and inaccuracies in computed geometries, mere color adjustments or partial pixel editing prove insufficient for correction. Advanced procedures, such as warping source images to conform to geometric features or adjusting camera parameters [35–37], are essential, particularly in real-time reconstruction pipelines prone to computational errors. Conversely, in offline pipelines leveraging accurate camera estimation and geometry processing, emphasis on texture warping diminishes.

3 | PROBLEM OF LARGE-SCALE TEXTURE RECONSTRUCTION

Reconstructing a large-scale area all at once poses significant challenges due to the substantial storage capacity required for processing high-resolution images and sensor data. For example, simply loading 4000 4 K color images would necessitate approximately 100 GB of memory resources, with additional capacity needed for processing. While advancements in hardware processors and resource capacities might suggest a resolution to this issue, the increasing demand for 3D immersive metaverse spaces and the proliferation of digital clones in virtual worlds indicate that the need for 3D reconstruction will also grow proportionally. Therefore, despite improvements in hardware capabilities, this challenge is expected to persist.

3.1 | Tiling

To address this issue, we opted to partition the large-scale area into manageable tiles for independent processing.

This approach mitigates concerns about inadequate computing resources, as the required capacity for processing remains modest and consistent, regardless of the area's size. Figure 1 illustrates the complete process of segmenting the large area into tiles. Following logical partitioning, each tile undergoes individual dense point, mesh, and texture reconstruction. Each module has been meticulously designed to ensure seamless integration within a tile-based processing framework.

The tiling process involves spatially dividing tiles and selecting relevant input data. First, tiles are created as empty cuboid boundaries based on the desired accuracy of the reconstruction result. These boundaries will later serve as effective buckets for reconstructing points, meshes, and textures. Next, it is essential to select and assign relevant input images for each tile. Only inputs providing relevant information for the specific tile should be selected, while excluding those that are not visible or too distant.

Figure 2 illustrates the process of dividing a large-scale area into cuboid tiles of desired dimensions and assigning selected input data to each tile. The

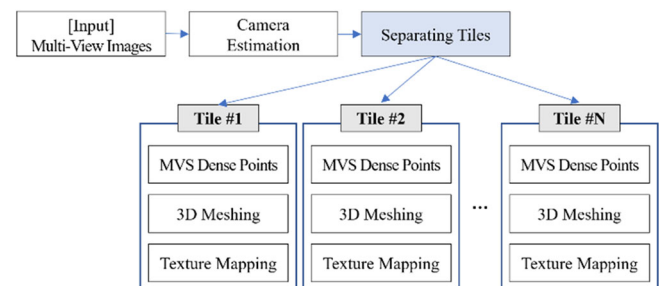


FIGURE 1 Tiling system overview.

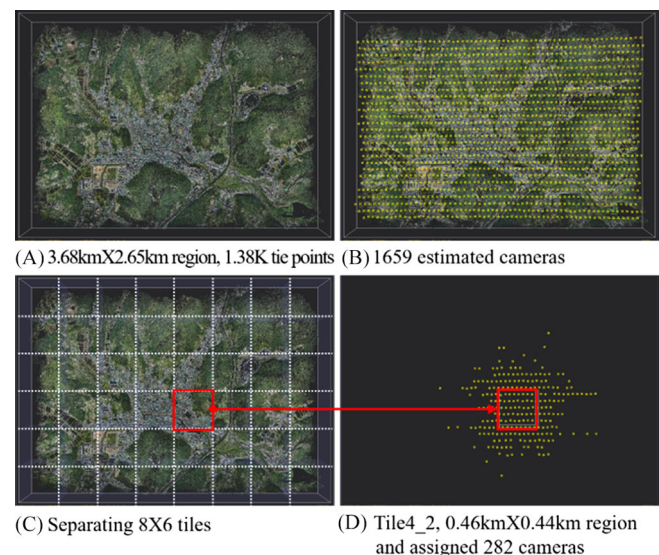


FIGURE 2 Tiling and assigning inputs.

experimental dataset pertains to an area measuring 3.682 km by 2.651 km, captured by a drone using 1659 4 K resolution images. This area was subdivided into an 8×6 grid of tiles. For instance, tile4_2, covering an area of 0.464 km by 0.446 km, required processing of only 282 input images.

Tiling offers several advantages beyond resource management. By enabling independent computations on each tile, this approach facilitates parallel processing across multiple systems. Furthermore, it simplifies post-reconstruction model data management and enhances the application of various techniques in large-scale scene visualization. For instance, maintaining data in a partitioned 3D model facilitates the implementation of methods such as level of detail (LOD) rendering and view frustum culling, thereby enhancing overall efficiency and performance.

3.2 | 3D geometry reconstruction

The process of generating dense points through MVS and subsequently constructing meshes relies on globally available and continuous input image data. Thus, even with tiling, seamless reconstruction of large areas is feasible. Despite the selective use of input data based on tiles, neighboring tiles share pertinent input, facilitating smooth connectivity in the resultant dense point set and mesh. Both dense points and meshes intentionally extend slightly beyond the 3D box of the tiles. Specifically, meshes overlap by approximately 1% to obscure the visibility of tile partition lines during visualization. Similarly, dense points overlap by about 2% during meshing to align curvature based on point normal, ensuring smoother transitions.

3.3 | Two issues in texture reconstruction

During the tile-based processing pipeline, challenges primarily arise during the texture reconstruction phase. At this stage, only the mesh of a single tile is accessible as input, rather than the entire mesh, leading to a notable absence of information regarding neighboring tiles' mesh data. This discontinuity between adjacent tiles gives rise to two main issues:

The first issue is visibility check errors. Initially, meshes are projected onto all images to compile candidate lists for texture mapping at the triangle face level. The visibility check process, which verifies whether candidate images effectively capture the face without occlusion, is critical. Figure 3 illustrates the causes of errors

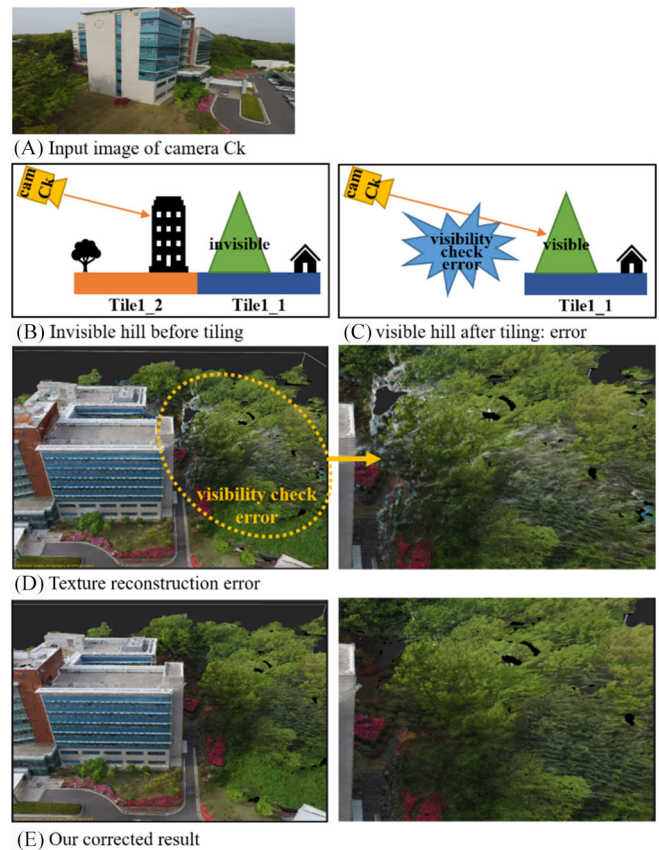


FIGURE 3 Visibility check error in tile-based processing: The building image is projected onto the green texture of the hill, causing it to appear white.

encountered during tile-based visibility checks. In Image (a), an actual image captured by Camera Ck shows significant occlusion of the hill area by buildings. However, during the tiling process, when the region between the hill and the buildings is divided into Tile1_1 and Tile1_2, the texture reconstruction process for Tile1_1 loses awareness of the buildings in Tile1_2. Consequently, when projecting the building image from Camera Ck onto Tile1_1, erroneous mapping of buildings onto the hill occurs.

The second issue arises during the final stage, specifically the color adjustment process. To generate high-quality texture maps, a projection-based method maps one image to each face of the mesh. However, this method often results in seams between neighboring mesh faces due to color discrepancies. These discrepancies stem from each image capturing the subject with different color values influenced by environmental conditions (weather, illumination, time variations) and internal camera settings. Particularly when capturing large-scale areas over extended periods, these input images exhibit varying color values, exacerbating color differences.

The MVS-Texturing method applies vertex color adjustments to minimize color discrepancies between neighboring faces. However, this approach relies on having knowledge of adjacent mesh faces and texture mapping details. When meshes are tiled for individual processing, obtaining information about neighboring meshes on other tiles and their texture specifics becomes impractical. Consequently, while adjusting texture color within each tile's mesh is feasible, rectifying texture color seams between adjacent tiles remains unachievable. Figure 4 illustrates the outcomes of tiled texture reconstruction using the MVS-Texturing method, clearly demonstrating the issue of tiled texture color seams resulting from information discontinuity between tiles.

Popular commercial software such as PhotoMesh and Context Capture have adopted a different approach known as color equalization [38,39], which also operates at the tile level. As depicted in Figure 5, this approach results in texture seams between tiles. Texture seams at the tile level can lead to significant side effects during

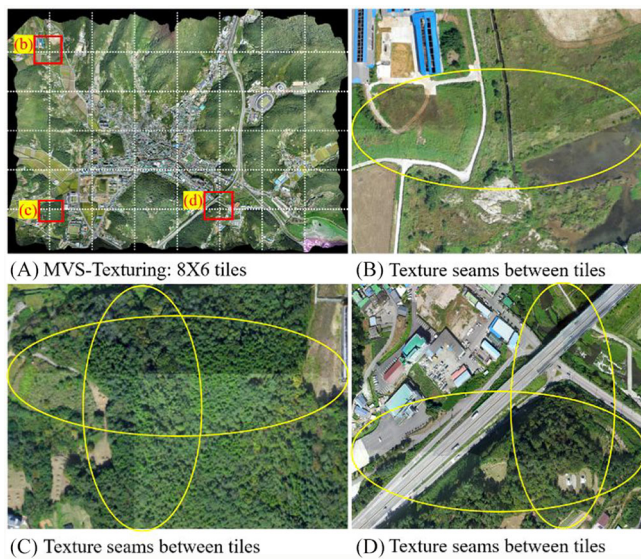


FIGURE 4 MVS-Texturing results: texture seams due to tile-based individual processing. MVS, multi-view stereo.

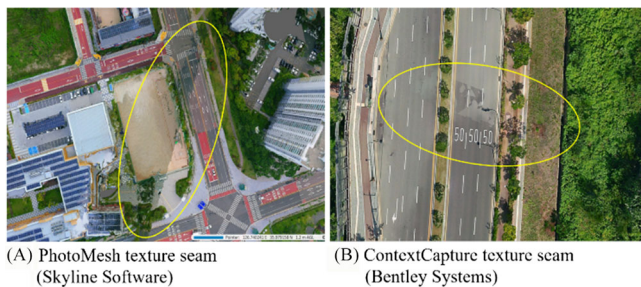


FIGURE 5 PhotoMesh and ContextCapture results: texture seams due to tile-based individual processing.

final 3D visualization, including flickering in overlapped areas between tiles.

It is evident that utilizing existing exemplary methods unchanged for reconstructing textures of tiled meshes is not feasible due to the aforementioned issues. Chapter 4 will introduce innovative approaches capable of addressing these concerns, and Chapter 5 will present outstanding texture reconstruction results achieved using these novel methods.

4 | TEXTURE RECONSTRUCTION

4.1 | Technical procedure

The conventional projection-based texture reconstruction method comprises three primary steps, as depicted in Figure 6. Firstly, all potential candidate views that could serve as sources for the texture are enumerated for each face. A crucial part of this step is performing a visibility check to ensure that each face is sufficiently captured from the camera view without being occluded by other objects. In the subsequent chapter, we introduce an enhanced method involving a two-pass visibility check to improve this process.

In the second step, the best view is selected from candidate views, following principles akin to MVS-Texturing. Initially, a photo consistency check is performed on the faces to exclude images capturing different content due to moving objects. Subsequently, a probabilistic evaluation determines the best image that is well-exposed, clear, captured perpendicularly, and smoothly connected with adjacent faces. The Markov random field energy formulation, as proposed by Lempitsky [28], is employed for selecting the best view.

Finally, to mitigate texture seams that the probabilistic formulation cannot resolve, direct color correction between different texture selections is performed. Because these seam issues arise from the locality of tile-based processing, we address this by leveraging color information from global reference images. This approach represents a core innovation of the texture reconstruction method proposed in this paper, with further elucidation provided in the subsequent chapter.

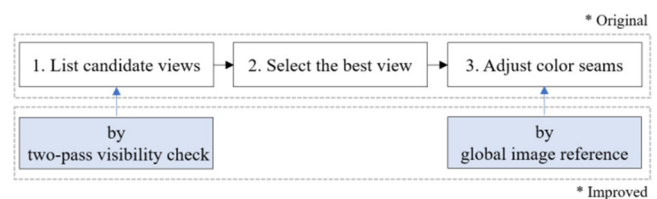


FIGURE 6 Texture reconstruction procedure original and improved (blue).

4.2 | Two-pass visibility check

To verify face visibility from camera views, two methods are viable: a simple depth map check and a sophisticated ray-triangle intersection analysis, which MVS-Texturing favors. While the first approach of directly comparing depth values is notably fast, its accuracy is constrained by limitations in depth sampling. Despite its computational intensity, the latter technique traces rays from the camera's perspective to faces, meticulously to determine visibility. As illustrated in Figure 3B,C, visibility is confirmed when a ray intersects with the target face first, without intersecting any other faces.

This study employs a two-step approach, as depicted in Figure 7. The first step involves inspecting global depth maps to roughly assess visibility across tiles. The second step utilizes ray-triangle intersection checking to precisely determine visibility locally.

In the first step, after completing the reconstruction of the meshes for all tiles, it is essential to pre-generate a global depth map based on the camera parameters of each image. The depth maps for the entire tile mesh are rendered on the GPU and stored according to the camera parameters of the input images. A face is considered visible if its depth value is closer than that of the corresponding depth value in the global depth map. Figure 8A illustrates the mesh of tile 4_2 along with the cameras mentioned in Figure 2. The 3D geometry highlighted in red corresponds to the camera associated with the 1085th

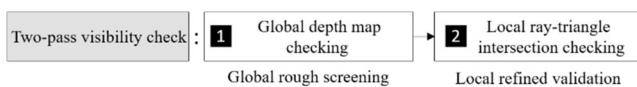


FIGURE 7 Two-pass visibility check.

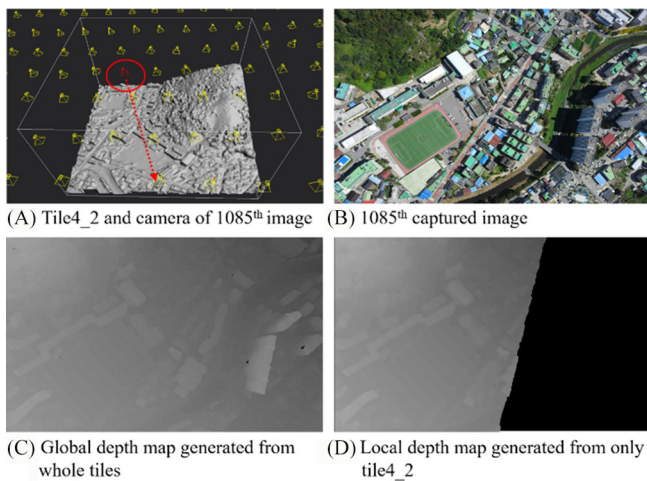


FIGURE 8 Global depth map and local depth map of tile 4_2 using 1085th camera parameters.

image, while Figure 8B shows the captured input image. Figure 8C,D compare the global depth map generated by re-rendering all tiles with the local depth map generated solely from tile 4_2. The pre-generated global depth map integrates depth information from all tiles, enabling comprehensive global visibility checks even in the absence of mesh data from adjacent tiles.

In the second step, we re-evaluate faces with minimal depth differences, which are difficult to assess using the depth map alone, through the ray-triangle intersection method. Figure 8A demonstrates the comparison between the depth line from the depth map (red) and the surface line of the 3D mesh (blue). The depth map, which captures only the closest depth value to the camera at the pixel level, is inherently prone to sampling errors. During the pixel shading process in GPU rendering, only one depth value at a specific position within the sub-pixel area is computed and stored, making it impossible to represent varying depth values within that region.

Consequently, the blue points in Figure 9A may be invisible in depth map checks but are deemed visible when evaluated through ray-triangle intersection. While the depth map records only the central depth value (red), suggesting occlusion due to it appearing numerically farther away, the points can be visually confirmed as visible. To address this issue, some texture techniques opt to rely solely on ray-triangle intersection rather than depth maps. However, we find that global depth map checking, which incorporates information from adjacent tiles, is essential. As a result, we implement a two-pass approach that conducts detailed ray-triangle intersection checks on faces within the zone of uncertainty (green), where sampling errors are likely to arise. The size of this zone is defined as 0.1% of the total depth range.

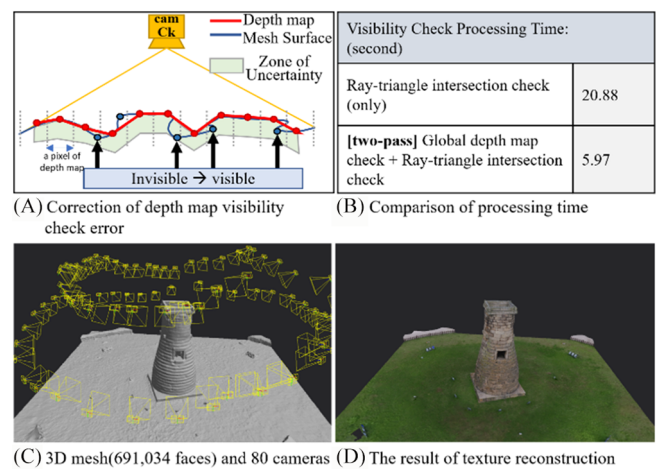


FIGURE 9 Global depth map checking error and two-pass visibility checking processing time.

By applying this method, views with significant depth differences, caused by occlusion from neighboring tiles or other faces within the same tile, are excluded during the first pass, greatly reducing the number of slow ray-triangle intersection checks. Figure 9C shows a scene containing 691 034 faces and 80 camera views. If only ray intersection checks are performed, as illustrated in (b), it takes 20.8 s. However, by first conducting a fast global depth map check and limiting ray intersection checks to faces within the zone of uncertainty, the time is reduced to 5.97 s. Although this adds additional processing steps, the overall time is significantly reduced.

4.3 | Color adjustment by global image reference

To address the challenge of texture seams between tiles, we implemented a method that utilizes global image referencing during color adjustment operations, as briefly introduced in the paper [40]. Instead of performing computations at the pixel level, we adjusted vertex color values within OpenGL shaders to achieve a smooth shading effect with the selected texture source. Subsequent sections will detail the methodology for determining appropriate vertex color values.

The procedure unfolds in two distinct phases, illustrated in Figure 10. Initially, we adjust the colors of boundary vertices of texture patches, where color seams occur, by referencing global information. A texture patch refers to a collection of faces that share the same best view image. Figure 11B visually represents these texture patches, each charted by a unique color, while (c) displays the result after image mapping. A detailed examination in image (d) reveals significant color discontinuities at the patch boundaries.

Many previous methods have focused on adjusting vertex or pixel colors within tiles to minimize texture color differences between patches. However, these approaches often encountered difficulties in correcting boundary seams between adjacent tiles. This is because calculating texture color differences between adjacent tiles becomes impossible when processing independently at the tile level. To address this issue, we reused the candidate view list compiled during the best view selection stage to maintain overall texture color consistency.

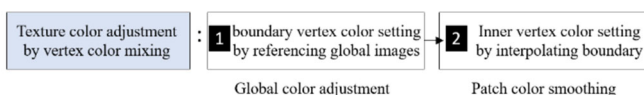


FIGURE 10 Texture color adjustment by vertex color mixing.

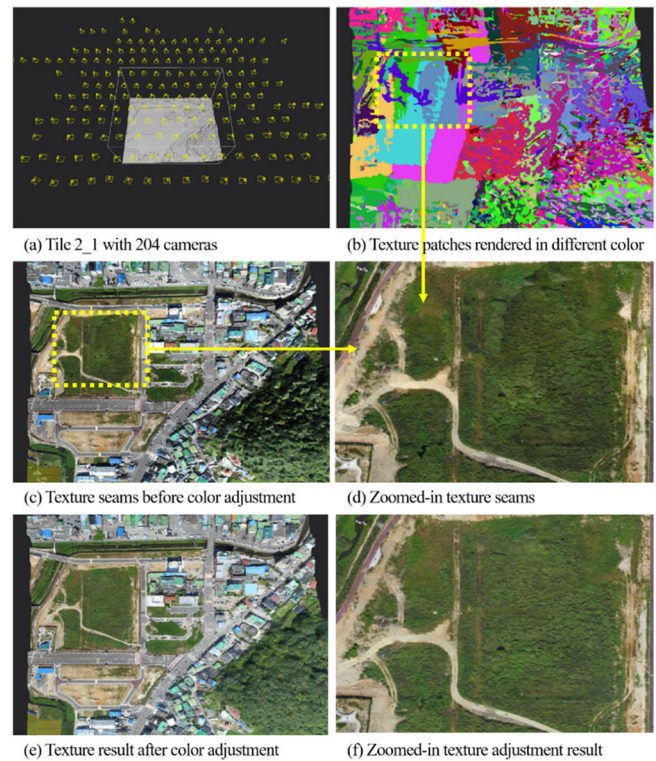


FIGURE 11 Texture seams occurring along the shape of the texture patch.

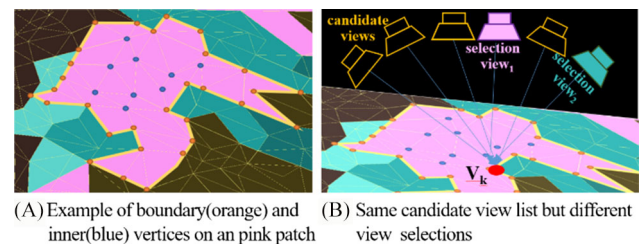


FIGURE 12 Boundary and inner vertices on a patch and view selection.

Figure 12 demonstrates that vertices sharing the same position will always have a globally identical set of candidate view images, as determined by visibility checks, even if they are processed separately due to tile division. Figure 12 illustrates that boundary vertex V_k selects view1 when it belongs to the pink patch and view2 when it belongs to the green patch, but both patches ultimately share the same list of six candidate views. Although the best view may differ, the candidate view list remains globally consistent. Based on this, we set the target texture color as the average of the pixel colors projected from vertex V_k across all candidate views. By adjusting the selected view's color to match the target color, regardless of which patch the boundary vertex V_k belongs to, we can prevent seams from forming at the boundaries.

The following Equation (1) expresses the adjustment color VC_k for the k th vertex. $IC_{x,y}^s$ represents the pixel color at position (x, y) , projected from vertex V_k onto the best view image s . Similarly, $IC_{x,y}^i$ refers to the pixel color projected from the i th candidate view. The variable n indicates the sampling window size in the image. To prevent colors from standing out in high-frequency images, a color sampled from a specific area is utilized. After calculating the average color of the candidate views, serving as the target texture color, we obtain the desired texture adjustment color VC_k by subtracting the pixel color from the currently selected view. Although this adjustment color is later added back to the selected view texture in the OpenGL Shader, restoring it to the target color, it is stored per vertex for use in the subsequent adjustment color interpolation process.

$$VC_k = \frac{\sum_{i \in \text{candidates}} \sum_{x=px_i, y=py_i}^{n,n} IC_{x,y}^i}{|\text{candidates}|} - \sum_{x=px_s, y=py_s}^{n,n} IC_{x,y}^s \quad (1)$$

In the first step, we made initial adjustments to reduce sharp seams along the boundaries of the texture patches. In the second step, it is necessary to smoothly interpolate the colors of the inner vertices within the patch. Because the texture patch is a closed planar graph without self-intersections, we employ the mean value coordinates (MVC) technique to interpolate the adjustment colors of the inner vertices [41]. The MVC method is effective for achieving smooth color interpolation for the inner vertices by considering their geometric relationships with the boundary vertices.

Figure 13 illustrates the process of correcting the texture seam shown in (a). It includes the determination of boundary vertex colors in (b) and the rendering of interpolated colors for the inner vertices in (c). Because the adjustment colors can be negative, a gray value was added to all colors for display purposes. However, the experimental results indicate that the actual adjustment color values are not as pronounced as represented in Figure 13.

Figure 14 outlines the comprehensive process of correcting texture seams depicted in (a) using our proposed

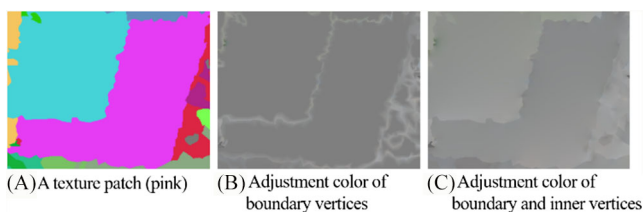


FIGURE 13 Determination process of vertex adjustment color values for a texture patch.

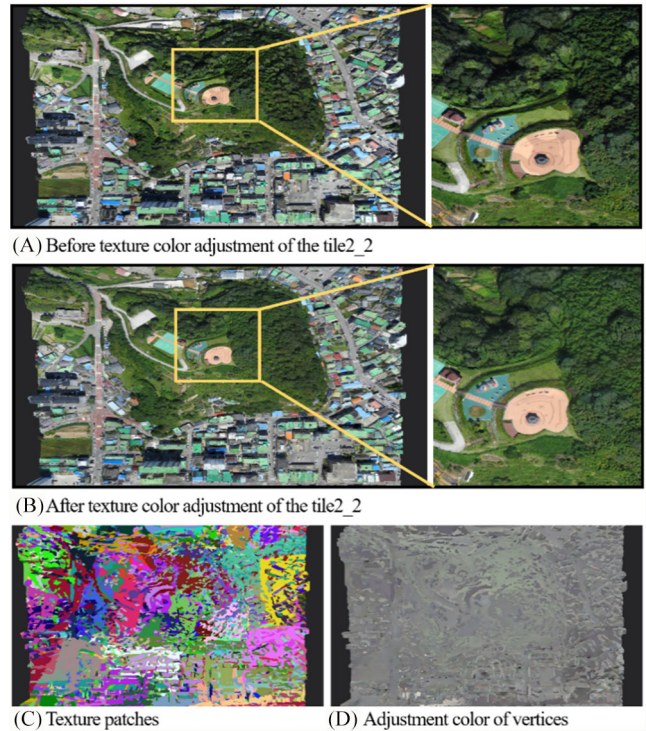


FIGURE 14 The comprehensive process of rectifying the texture seams and rendering the resulting vertex adjustment color.

methodology, resulting in the generation of a smooth and high-quality texture map shown in (b). Image (d) depicts the adjustment color values of all vertices. The final outcome (b) is achieved by adding the selected view image from (a) with the adjustment color values from (d) through OpenGL shading. During the final texture saving stage, the adjustment colors of vertices and the selected view image fragments are baked and stored.

5 | EXPERIMENTAL RESULTS

The experiment was conducted on two datasets captured using different methods. The first dataset contains complex and relatively small-scale scene data captured from close range with angled shots, while the second dataset consists of large-scale scene data captured vertically from 500 m above ground. These datasets were used for texture reconstruction with RealityCapture (a blending-based method), MVS-Texturing (a projection-based method), and the proposed technique. The results were then compared and analyzed.

Figure 15A–E show the experimental results for a small rocky island (60 m × 70 m) located in Danyang, South Korea, captured using 190 images at 4 K resolution. Two image sets, taken on different days, were used, and additional images were captured to more accurately

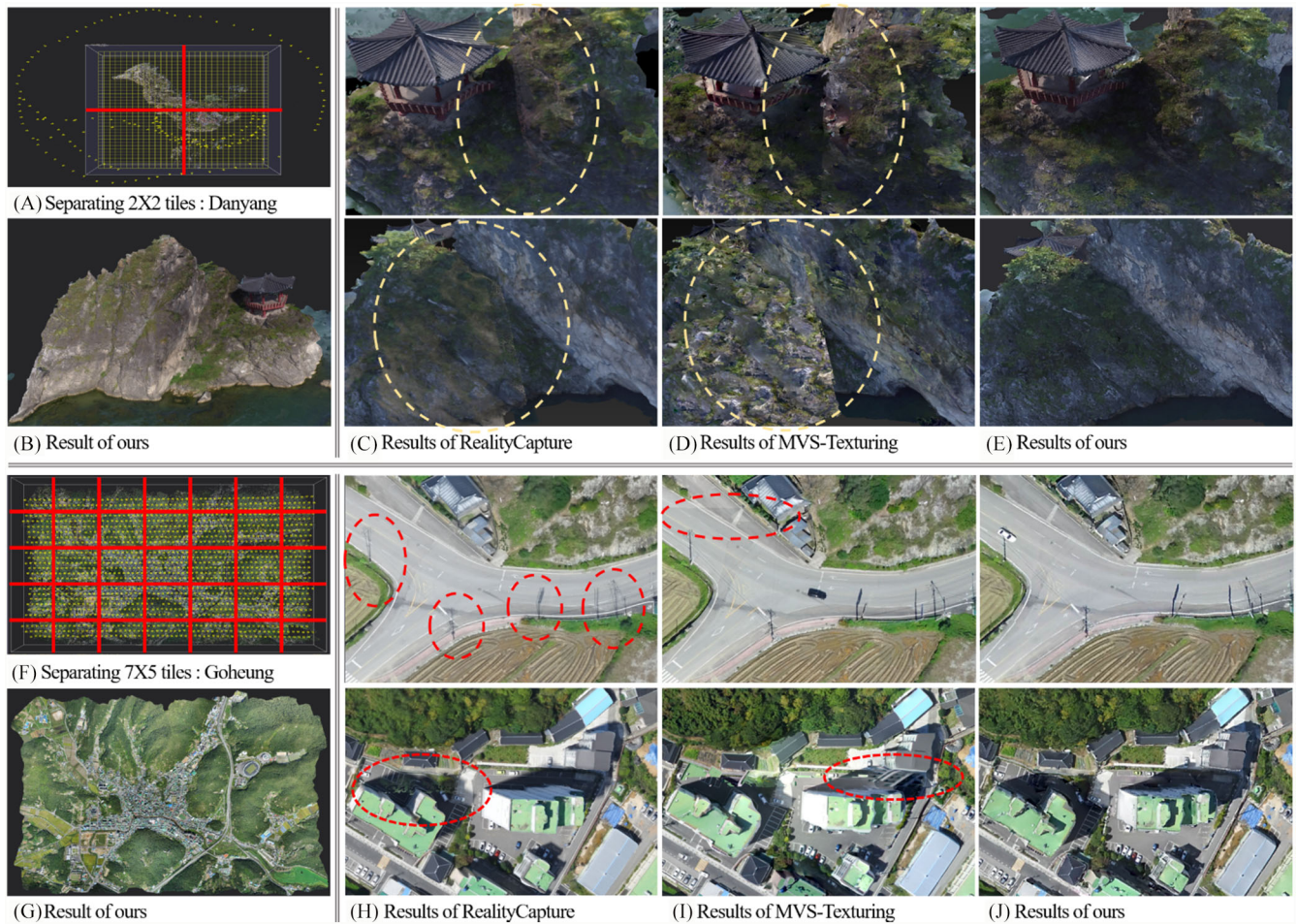


FIGURE 15 Seamless texture reconstruction results for a complex scene: comparison with other techniques.

reconstruct the pavilion on the island. However, many of these images were taken in low-light conditions, resulting in significant color differences between the images. Because the island was captured from an oblique angle, as shown in Figure 15A, occlusions occurred between the pavilion and the surrounding peaks. For the experiment, we divided the scene into a 2×2 grid (as indicated by the red line in Figure 15A), with a tile boundary falling near the heavily occluded pavilion area. Figure 15C,D show texture reconstruction results using RealityCapture and MVS-Texturing, respectively, for comparison.

RealityCapture, as mentioned in Section 2, applies a multiband blending technique, which theoretically prevents color discrepancies between tiles, despite its disadvantages such as low quality and ghosting artifacts. However, as shown in Figure 15C, texture seams still appeared between tiles due to visibility check errors, which incorrectly incorporated unrelated view images into the texture blending process. The MVS-Texturing result in Figure 15D showed even more pronounced texture seams, as expected. Visibility check errors and color correction issues led to the selection of incorrect view

images, further emphasizing the seams. On the other hand, the proposed method (Figure 15E) produced high-quality results without texture errors or visible seams.

Figures 15F–J present results from a larger-scale experiment involving the Goheung region, covering an area of $3.68 \text{ km} \times 2.65 \text{ km}$, as discussed in Chapter 3. The region was captured vertically using a fixed-wing drone, reducing the visibility check errors seen in the Danyang dataset. However, temporal changes in the scene data, due to the extended capture period, were still evident.

Figure 15H shows the result from the RealityCapture software. The upper image highlights a common issue with blending-based methods: ghosting, caused by changes in shadow positions during the long capture period. While not directly related to tiling, this issue is frequently cited as a limitation of blending methods. Visibility check errors are visible throughout the image, such as in the lower image where the green roof of a building is mistakenly blended into the ground. Figure 15I shows the results from MVS-Texturing. While ghosting is absent in the upper image, color inconsistencies are present

between the tiles. Similar visibility check errors occurred at tile boundaries in the lower image, where incorrect images were mapped.

The experiment demonstrated that both traditional blending-based and projection-based methods encounter issues in tile-based processing, highlighting the need for

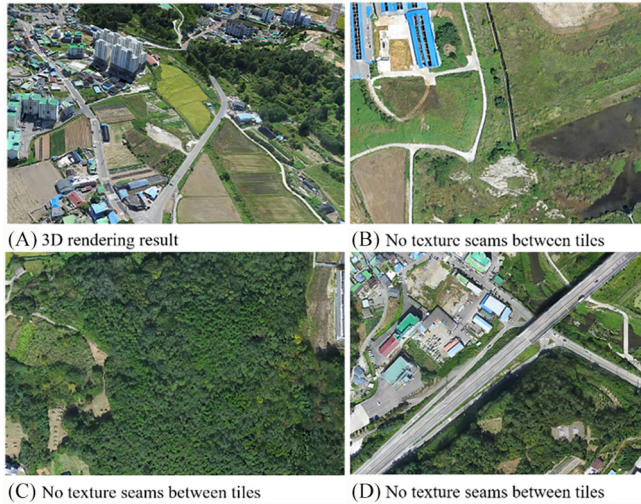


FIGURE 16 Seamless texture reconstruction results for a large-scale scene: comparison with Figure 4.

a specialized approach to avoid errors. Figures 15E,J show that the texture maps reconstructed using the proposed method were of high quality, free from errors, and color seams.

The results in Figure 15 were zoomed in to highlight visibility check errors, making it difficult to assess seamless color adjustments between tiles. Therefore, Figure 16 presents a zoomed-out view of the Goheung dataset processed using the proposed technique. Figures 16B–D show the same area for comparison, where the texture seams caused by local color adjustment in MVS-Texturing were eliminated, resulting in smooth and realistic textures.

Figure 17 presents the reconstructed result of Songak Mountain Fortress in Jeju Island, captured using 2034 drone images at 4 K resolution. The entire area, measuring $1151 \text{ m} \times 1075 \text{ m}$, was processed by dividing it into a 5×5 tile grid. Despite the tiling, the reconstruction was clearly generated at high quality without any visible texture seams.

The proposed technique primarily utilizes face and vertex-level computations along with OpenGL shading functions, avoiding time-consuming operations. The most time-consuming phase occurs during the texture packing stage after processing is complete [42]. Table 1 below



FIGURE 17 High-quality texture reconstruction for large-scale scene: Songak Mountain Fortress in Jeju Island 1.1 km^2 , processed into 5×5 tiles.

TABLE 1 Processing time per pipeline step and experimental setup.

1. List Candidate view	2. Select The best view	3. Adjust Color seams	4. Save Texture maps (baking + packing)
5.879 s	5.995 s	4.476 s	39.524 s
Total: 44.334 second			
[Experimental environment]		[Experimental data spec.]	
CPU: Intel® Core TM i9-12900L		Mesh face num: 492 905	
3.20GHz		Mesh Vertex num: 247 537	
RAM: 128GB		Input image num: 271	
GPU: Nvidia RTX A6000		Input image resolution: 5472×3648	
OS: Windows 11 64bit			

illustrates the time taken for processing tile_{2_2} of the Goheung data from Figures 11 and 16 by each pipeline step. Processing one tile of the Goheung data required an average of approximately 50 to 60 s. The entire area, comprising 48 tiles and 18 506 421 mesh faces, took approximately 43 min to process.

6 | CONCLUSION

In this paper, we have presented an advanced texture reconstruction technique tailored for large-scale scenes, utilizing tile-based processing to effectively address visibility check errors and texture seams. By leveraging a global depth map, we ensured comprehensive visibility information across the entire scene despite processing each tile locally. Additionally, we utilized a universally shared texture candidate list to prevent texture seams caused by discontinuities between tiles. Our method, which employs a one-source projection-based approach without blending, has been empirically validated to generate high-quality texture maps.

As discussed, tile-based processing is crucial for managing large-scale scenes, requiring the development of specialized techniques to ensure seamless connectivity in the final output. Moving forward, our future research will continue to explore and expand upon technologies suitable for tile-based processing throughout the entire 3D reconstruction pipeline. Specifically, we are focused on employing deep learning techniques to achieve smooth surface reconstructions that seamlessly bridge across tiles.

ACKNOWLEDGMENTS

This research was supported by the Culture, Sports and Tourism R&D Program through the Korea Creative Content Agency grant funded by the Ministry of Culture, Sports and Tourism in 2022 (Project Name: Development of large-scale space cloning and high quality real-time reconfiguration technology for realistic content, Project Number: R2020040207).


CONFLICT OF INTEREST STATEMENT

The authors declare that there are no conflicts of interest.

ORCID

Hye-Sun Kim  <https://orcid.org/0000-0002-4879-5178>

Yun-Ji Ban  <https://orcid.org/0000-0003-4118-8612>

Chang-Joon Park  <https://orcid.org/0009-0008-5266-6134>

REFERENCES

- J. L. Schönberger and J. -M. Frahm, *Structure-from-motion revisited*, In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, 4104–4113.
- S. Fuhrmann, F. Langguth, N. Moehrl, M. Waechter, and M. Goesele, *MVE—an image-based reconstruction environment*, *Comput. Graph.* **53** (2015), no. Part A, 44–53. PMID: ISSN 0097-8493.
- Y. Furukawa and C. Hernández, *Multi-view stereo: a tutorial*, *Foundations Trends. Comput. Graph. Vision* **9** (2015), no. 1–2, 1–148.
- M. Kazhdan and H. Hoppe, *Screened Poisson surface reconstruction*, *ACM Trans. Graph.* **32** (2013), no. 3, 1–13.
- S. Fuhrmann and M. Goesele, *Floating scale surface reconstruction*, *ACM Trans. Graph.* **33** (2014), no. 4. PMID: article no. 46.
- M. Jancosek and T. Pajdla, *Multi-view reconstruction preserving weakly supported surfaces*, *IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)* (2011), 3121–3128.
- P. Wang, Z. Wang, S. Xin, X. Gao, W. Wang, and C. Tu, *Restricted Delaunay triangulation for explicit surface reconstruction*, *ACM Trans. Graph.* **41** (2022), no. 5, 1–20.
- B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, *NeRF: representing scenes as neural radiance fields for view synthesis*, *Commun. ACM* **65** (2021), no. 1, 99–106.
- B. Kerbl, G. Kopanas, T. Leimkuehler, and G. Drettakis, *3D Gaussian splatting for real-time radiance field rendering*, *ACM Trans. Graph.* **42** (2023), no. 4, 1–14.
- Z. Liao and S. L. Waslander, *Multi-view 3D object reconstruction and uncertainty modelling with neural shape prior*, In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, 3098–3107.
- Y. Hong, K. Zhang, J. Gu, S. Bi, Y. Zhou, D. Liu, L. Feng, K. Sunkavalli, T. Bui and H. Tan, (2023). LRM: large reconstruction model for single image to 3d. arXiv preprint arXiv: 2311.04400.
- S. Peng, Z. Xu, J. Dong, Q. Wang, S. Zhang, Q. Shuai, H. Bao, and X. Zhou, *Animatable implicit neural representations for creating realistic avatars from videos*, *IEEE Trans. Pattern Anal. Mach. Intell* (2024).
- J. L. Schönberger, E. Zheng, J. M. Frahm, and M. Pollefeys, *Pixelwise view selection for unstructured multi-view stereo*, *Eur. Conf. Comput. Vision (ECCV)* (2016).
- Z. Liu, Z. Xu, C. Diao, W. Xing, and D. Lu, *Benchmarking large-scale multi-view 3D reconstruction using realistic synthetic images*, In *Eleventh International Conference on Graphics and Image Processing (ICGIP 2019)*, SPIE, 2020, 741–747.
- M. Waechter, N. Moehrl, and M. Goesele, *Let there be color! Large-scale texturing of 3D reconstructions*, *ECCV 2014* (2014), 836–850.
- F. Bernardini, I. M. Martin, and H. Rushmeier, *High-quality texture reconstruction from multiple scans*, *IEEE Trans. Vis. Comput. Graph.* **7** (2001), no. 4, 318–332.
- L. Grammatikopoulos, I. Kalisperakis, G. Karras, and E. Petsa, *Automatic multi-view texture mapping of 3d surface projections*. *3D Virtual Reconstruction & Visualization of Complex Architectures*, pages 12–13
- M. Callieri, P. Cignoni, M. Corsini, and R. Scopigno, *Masked photo blending: mapping dense photographic data set on high-resolution sampled 3D models*, *Comput. Graph.* **32** (2008), no. 4, 464–473.

1. J. L. Schönberger and J. -M. Frahm, *Structure-from-motion revisited*, In *2016 IEEE Conference on Computer Vision and*

19. C. Allene, J. -P. Pons, and R. Keriven, *Seamless image-based texture atlases using multi-band blending*, In *2008 19th International Conference on Pattern Recognition*, IEEE, Tampa, FL, USA, 2008, 1–4.
20. Z. Chen, J. Zhou, Y. Chen, and G. Wang, *3D texture mapping in multi-view reconstruction*, In *Advances in visual computing. ISVC 2012. Lecture notes in computer science*, G. Bebis et al. (eds.) Vol. **7431**, Springer, Berlin, Heidelberg.
21. Agisoft Metashape: user manual. <https://www.agisoft.com/downloads/user-manuals/>
22. RealityCapture Help. Adjusting coloring and texturing settings. https://rhelp.capturingreality.com/en-US/tools/texturing_part2.htm
23. 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.
24. T. Whelan, M. Kaess, H. Johannsson, M. F. Fallon, J. J. Leonard, and J. B. McDonald, *Real-time large-scale dense RGB-D SLAM with volumetric fusion*, *Int. J. Robot. Res.* **34**, 598–626.
25. 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 j.* **44**, 51–61.
26. J. H. Lee, H. Ha, Y. Dong, X. Tong, and M. H. Kim, *TextureFusion: high-quality texture acquisition for real-time RGB-D scanning*, In *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, Seattle, WA, USA, 2020, 1269–1277.
27. J. Kim, H. Kim, H. Nam, J. Park, and S. Lee, *TextureMe: high-quality textured scene reconstruction in real time*, *ACM Trans. Graph.* **41** (2022), no. 3, 1–18.
28. V. Lempitsky and D. Ivanov, *Seamless mosaicing of image-based texture maps*, In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Minneapolis, MN, USA, 2007, 1–6.
29. R. Gal, Y. Wexler, E. Ofek, H. Hoppe, and D. Cohen-Or, *Seamless montage for texturing models*, *Comput. Graph. Forum.* **29** (2010), 479–486.
30. L. Velho and J. Sossai Jr., *Projective texture atlas construction for 3D photography*, *Visual Comput* **23** (2007), 621–629.
31. H. Liu, L. Xie, X. Li, and W. X. Wang, *Automatic texture mapping method for 3D models to circumvent occlusion through 3D spatial through-view relationships between images, models, and point clouds*, *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, XLIII-B2–2022, 417–424.
32. Y. Li, Y. Li, J. Yao, Y. Gong, and L. Li, *Global color consistency correction for large-scale images in 3-D reconstruction*, *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.* **15** (2022), 3074–3088.
33. J. Choi, D. Jung, T. Lee, S. Kim, Y. Jung, D. Manocha, and D. Lee. TMO: textured mesh acquisition of objects with a mobile device by using differentiable rendering. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),
34. J. Yang, L. Liu, J. Xu, Y. Wang, and F. Deng, *Efficient global color correction for large-scale multiple-view images in three-dimensional reconstruction*, *ISPRS J. Photogramm. Remote Sens.* **173**, 209–220.
35. E. Aganj, P. Monasse, and R. Keriven, *Multi-view texturing of imprecise mesh*. *Computer Vision – ACCV 2009*, In *Lecture notes in computer science*, Vol. **5995**, Springer, Berlin, Heidelberg.
36. Q. Zhou and V. Koltun, *Color map optimization for 3D reconstruction with consumer depth cameras*, *ACM Trans. Graph.* **33** (2014), no. 4 2014. PMID: 10 pages.
37. Y. Fu, Q. Yan, L. Yang, J. Liao, and C. Xiao, *Texture mapping for 3D reconstruction with RGB-D sensor*, In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, 4645–4653.
38. Skyline Software, PhotoMesh general information, https://support.sky_linesoft.com/hc/en-us/sections/360003999980-General-Information
39. Bentley Product Documentation, ContextCapture user guide, <https://docs.bentley.com/LiveContent/index.html>
40. H. Kim, Y. Ban, and C. Park, (2018). A seamless texture color adjustment method for large-scale terrain reconstruction. *ACM SIGGRAPH 2018 Posters*.
41. K. Hormann and M. S. Floater, *Mean value coordinates for arbitrary planar polygons*, *J. ACM Trans. Graph. (TOG)* **25** (2006), no. 4, 1424–1441.
42. thekla_atlas, Atlas Generation Tool, Github open source project, https://github.com/Thekla/thekla_atlas

AUTHOR BIOGRAPHIES



Hye-Sun Kim earned her BS and MS degrees in Computer Science from Pusan National University, Pusan, South Korea, in 1999 and 2001, respectively. Since joining the Electronics and Telecommunications Research Institute (ETRI) in Dae-

jeon, South Korea, in 2001, she has been involved in developing technologies for multi-platform game engines, high-quality computer graphics, and large-scale 3D reconstruction. Currently serving as a principal researcher, her research interest includes 3D reconstruction pipelines leveraging neural rendering techniques.



Yun-Ji Ban received her BS in electrical and electronics engineering from Kyungpook National University, Daegu, Republic of Korea, in 2003, and MS degree in electrical engineering from Korea Advanced Institute of Science and Technology,

Daejeon, Republic of Korea, in 2005. She has been working at the Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea, since 2005. Her research interests include computer vision, computer graphics, and 3D reconstruction for digital contents. Her current research interests include digital healthcare, integrating AI technologies into digital contents.



Chang-Joon Park received his MS and PhD degrees in electronic engineering in 1996 and 2000, respectively, from Kyungpook National University, Daegu, Republic of Korea. He has been working at the Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea, since 1998. He is currently a principal researcher. His current research interests include image-based wide-area terrain 3D reconstruction and neural rendering.

How to cite this article: H.-S. Kim, Y.-J. Ban, and C.-J. Park, *UTexGen: High-quality texture reconstruction for large-scale scenes using multi-view images*, *ETRI Journal* **47** (2025), 983–995, DOI [10.4218/etrij.2024-0320](https://doi.org/10.4218/etrij.2024-0320)