Instant Panoramic Texture Mapping with Semantic Object Matching for Large-Scale Urban Scene Reproduction

Jinwoo Park*, Ik-beom Jeon, *Student Members, IEEE*, Sung-eui Yoon, and Woontack Woo†, *Members, IEEE*

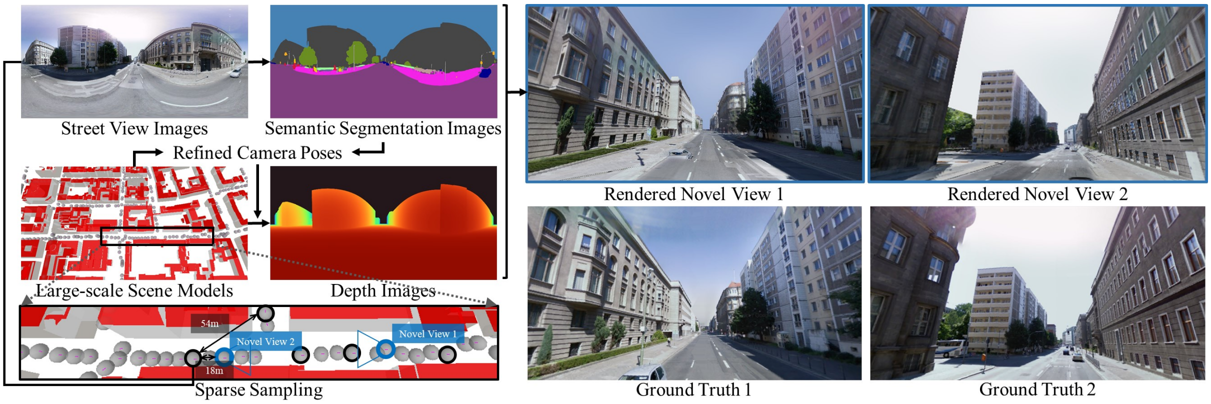


Fig. 1: Our proposed method extracts meaningful information, such as semantic segmentation, from readily obtainable street-view images, refined extrinsic camera parameters, and rendered depth, for use in real-time processes. After a semantic object-matching test, sparsely sampled street-view images are mapped onto open 3D scene models with proper blending weights, considering the user's position, followed by semantic 3D inpainting. Finally, novel views can be realistically synthesized to provide users with free walk-through experiences in large-scale urban streets.

Abstract—This paper proposes a novel panoramic texture mapping-based rendering system for real-time, photorealistic reproduction of large-scale urban scenes at a street level. Various image-based rendering (IBR) methods have recently been employed to synthesize high-quality novel views, although they require an excessive number of adjacent input images or detailed geometry just to render local views. While the development of global data, such as Google Street View, has accelerated interactive IBR techniques for urban scenes, such methods have hardly been aimed at high-quality street-level rendering. To provide users with free walk-through experiences in global urban streets, our system effectively covers large-scale scenes by using sparsely sampled panoramic street-view images and simplified scene models, which are easily obtainable from open databases. Our key concept is to extract semantic information from the given street-view images and to deploy it in proper intermediate steps of the suggested pipeline, which results in enhanced rendering accuracy and performance time. Furthermore, our method supports real-time semantic 3D inpainting to handle occluded and untextured areas, which appear often when the user's viewpoint dynamically changes. Experimental results validate the effectiveness of this method in comparison with the state-of-the-art approaches. We also present real-time demos in various urban streets.

Index Terms—Panoramic texture mapping, large-scale urban-scene rendering, novel-view synthesis, semantic object matching, real-time inpainting, image-based rendering, virtual reality

1 Introduction

Photorealistic reproduction of real-world urban scenes has played a significant role in empowering various virtual reality (VR) applications that reflect the physical world, including virtual touring [20], geotagged social media [16], or information visualization [5]. As a core technique for maximizing the realism of such experiences, image-based rendering (IBR) is one of the most active and important research topics in both computer graphics and vision. In particular, according to the geometry-image continuum of IBR [46], physically-based approaches [10, 14], which generally map captured real images as textures onto a corresponding virtual proxy geometry, are often employed to synthesize a novel view with a wide range of free viewpoints for the user.

However, such traditional IBR methods typically require several decades or hundreds of neighboring images from a user's viewpoint in order to cover a restricted local area, thus causing difficulties in their application to large-scale urban scenes. In addition, prior methods for adjacent novel-view synthesis have depended on local geometric information [23, 41, 50], which also limits the extensibility, although these methods realistically synthesized novel views through the reconstruction of high-fidelity and dense meshes of local proxy geometry from depth images [23, 41] or point clouds [50]. Hence, such approaches cannot support a full six degrees of freedom (DoFs), including translation and rotation, for free walk-through experiences in large scenes.

With the development of global data obtained by leading companies or research groups, advanced techniques for handling large-scale scenes have recently been proposed. In particular, given the vast number of street-level panoramic images and corresponding camera parameters provided by Google Street View [2], a wide range of urban scenes can be reconstructed [32, 48] from vision-based algorithms, such as structure from motion (SfM), which can be used for IBR. Nevertheless, such methods still require large databases of resource images and proportionally high computation times to represent a scene at scale. In addition, the reconstructed point clouds are too large to be used in a real-time IBR system.

As a more interactive way to support users with highly realistic

- *Jinwoo Park, Ik-beom Jeon, and Woontack Woo are with KAIST UVR Lab.. E-mail: {jinwooaa | ikbeomjeon | wwoo}@kaist.ac.kr*
- *Sung-eui Yoon is with KAIST. E-mail: sungeui@kaist.ac.kr*

Manuscript received 9 Sept. 2020; revised 15 Dec. 2020; accepted 8 Jan. 2021. Date of publication 24 Mar. 2021; date of current version 7 Apr. 2021. Digital Object Identifier no. 10.1109/TVCG.2021.3067768

fly-over experiences in world-scale urban scenes, Google Earth [20] reconstructed in advance almost all global scene meshes with textures using aerial photographs and rendered them in real time. Nevertheless, this gigantic project mainly focused on rendering global scenes at a bird's-eye view; thus, the quality of the geometry and textures at a street level significantly decreased. A more applicable method for constructing walk-through experiences in urban streets was employed by Geollery [16], which adopted an efficient transformation of a dense spherical mesh to construct a local proxy geometry based on the depth maps from Google Street View. Since this method depended on two nearby street-view images and local depth information, frequent resource updates were required according to the user's position, thus causing temporally unstable results.

To cover a wide range of urban scenes, the proposed method also depends on the global street-view images from Google Street View and a mass of simplified 3D models that can be easily obtained from open databases [1] as well as maps such images onto the scene geometry as textures based on the seminal idea of view-dependent texture mapping (VDTM) [13]. Specifically, since our method uses sparsely sampled street-view images containing high-resolution and omnidirectional scene information at their camera positions, the use of a large number of image resources or frequent sampling of street-view images to synthesize an adjacent novel view is not necessary. This property results in temporally stable rendering quality regardless of the freely changing view of the user, with a low computational cost.

However, handling gaps between simplified scene models and the real geometry captured in street-view images remains as an important issue. This gap causes erroneous texture mapping and perceptually undesirable rendering results. To solve this issue, our key idea is to extract semantic information, *S*, from the given street-view images and to deploy this powerful information in proper intermediate steps by the proposed system to enhance both the mapping accuracy and performance time. First, *S* is used in the outline-matching-based image-model registration to refine the raw extrinsic camera parameters provided by Google Street View. Then, to correctly assign texture colors to geometric surfaces, *S* is used for testing semantic matching between a real object in a street-view image and a target object to be textured. Finally, to handle visual holes due to complex occlusions, *S* also helps with the proposed semantic 3D inpainting, which uses both semantic information and 3D geometry, thus providing perceptually convincing hole-filling results with real-time performance. Ultimately, a user can have free walk-through experiences in large-scale urban scenes with high-quality novel views in real time.

Overall, main contributions of our work are summarized as follows:

- A novel, real-time, physically-based IBR system utilizing easily accessible open resources for high-quality free walk-through experiences in various large-scale urban scenes at a street level.
- Efficient and accurate panoramic texture mapping, which makes full use of the inferred semantic information of a scene in proper intermediate steps of the proposed system pipeline.
- Effective real-time inpainting using both 3D geometry and semantic information to support a dynamically changing novel-view synthesis without visual holes due to complex occlusions.

2 Related Work

Novel-View Synthesis Using Proxy Geometry. Generally, earlier IBR approaches used various types of proxy geometries, such as 3D meshes, depth images, or point clouds. Chaurasia et al. [7] suggested a warping method using semi-automatically selected silhouettes and densely reconstructed point clouds. In their following work [6], they used shape-preserving warping based on superpixels and synthesized depths for novel-view synthesis. Moreover, Penner et al. [41] suggested a soft 3D representation of scene geometry based on local depth maps and vote volumes. Those approaches aimed at plausible rendering of challenging scenes with complex objects and uncertain geometries.

Hedman et al. [25] jointly used a coarse global 3D geometry and detailed per-view meshes of an indoor scene using RGB-D input images. For general purposes, they constructed panoramic 3D photographs [22] with a texture, a normal map, and multilayered meshes of a scene from casually captured RGB images, although dense reconstruction took a few hours using multi-view stereo (MVS). In a follow-up work [23], a faster and more novice-friendly method was suggested for constructing 3D panoramas using an input sequence of color-and-depth image pairs captured from a dual-lens cell phone camera. However, such methods still require dozens of input images to render a local scene or provide limited free viewpoints; moreover, they are difficult to apply to global-tour scenarios. Although convolutional neural networks (CNNs) [33] have been used by recent IBR methods [18,19,24,31,37,53] and yielded plausible results, they still support a narrow range of possible novel viewpoints with distortion or blurring errors.

Without using detailed and thus time-consuming reconstruction of a local scene, for our ultimate goal of novel-view synthesis in large-scale urban scenes, we use simplified and fixed 3D models of various real cities obtained from open resources. Moreover, to provide a coherently rendered scene even when the user's viewpoint dynamically changes, our system deploys sparsely sampled panoramic street-view images as textures, substituting dozens of nearby images with limited FoV.

3D Urban Reconstruction and Texturing. According to the types of outputs, various urban-scene-reconstruction and rendering methods were grouped by Musialski et al. [38]. To effectively handle large-scale urban scenes, previous approaches often employed polygon-based geometric representations that were simplified and refined from initial reconstructions using SfM or MVS. One representative work is by Xiao et al. [50], who constructed simplified 3D building models from RGB images captured at a street level. They separated the point clouds reconstructed by SfM into the respective building blocks using a semantic segmentation method, and then modeled and textured the simplified facades of the building blocks.

Additionally, urban reconstruction methods have used more suitable input resources, such as airborne images, point clouds from light detection and ranging (LiDAR) scans, and panoramic images with GPS and IMU data, rather than relying solely on typical RGB or depth images with limited FoVs. Siu et al. [44] took advantage of the wide FoVs of panoramic images to obtain a scalable proxy geometry. To reconstruct a plausible facade with its texture, Li et al. [35] complementarily used RGB images and LiDAR data. Recently, with the development of open resources, such as Google Street View [2] or OpenStreetMap [21], a vast urban scene can be reconstructed more conveniently [32,40,43,48]. Although those methods can realistically render urban scenes, they commonly depend on heavy precomputation to reconstruct detailed geometry or to refine textures, which may be improper to large scenes.

Our method provides effective instant texture mapping of an entire scene without significant dependence on a vast amount of precomputation. To compensate for the use of simplified geometry and sparsely sampled street-view textures, our key idea is to deploy inferred semantic information from street-view images in proper intermediate steps of the whole process for an accurate and efficient texture mapping.

Inpainting for Untextured Holes. To deal with untextured areas, which are invisible to all available cameras or occluded by other geometry when dynamically changing the user's viewpoint, an effective hole-filling algorithm is necessary. In previous IBR methods, simple interpolation techniques were typically employed. For example, in the pioneering work of VDTM [13], untextured regions were filled with colors of the surrounding polygons. Du et al. [16] employed a Gaussian filter to smoothly interpolate adjacent pixel colors in erroneous regions. Other methods, which mostly focused on facade-texture mapping, used a constant color to fill untextured sides of a reconstructed building [28,50]. Although such approaches were suitable for a real-time system, the inpainted results were blurred or unnatural, thus lacking semantic awareness of a scene.

In an image space, plausible inpainting results have been yielded by various methods that used Poisson's equations [42], or a randomized nearest neighbor algorithm to find proper patch matches [4]. PixMix [27] found the best pixels to fill holes by mainly minimizing two cost functions: pixel distances in the 2D image space and color differences between adjacent pixels. However, they still could not ob-

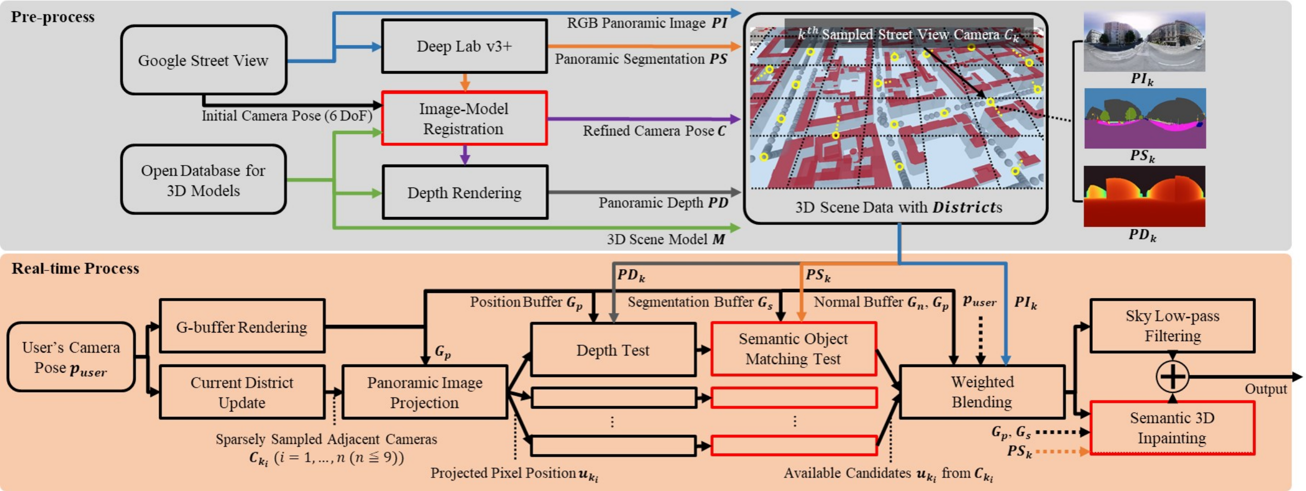


Fig. 2: Overview. In a pre-process, our system constructs *3D Scene Data*, which contains five different input resources: street-view images, 3D models, estimated panoramic-segmentation images, synthetic panoramic-depth images, and refined extrinsic camera parameters. For sparse sampling of street-view images according to a user's current position, *3D Scene Data* is divided into smaller *Districts*. In a real-time process, pixels in a position buffer at a current view are projected onto sampled street-view images to get texture colors. After passing a depth test and a semantic object matching test, filtered candidate colors are properly blended as a final color. As a final step of complete scene rendering without visual holes, low-pass filtering and semantic 3D inpainting are applied to sky and non-sky areas respectively. Note that our main contributions lie on utilizing semantic information in the proper intermediate steps (marked in red boxes) to enhance both rendering quality and performance time.

tain semantic information on what to fill. Recent deep learning-based methods trained semantically aware networks using a large dataset. Conversely, Iizuka et al. [30] used expensive post-processing, and Yang et al. [51] restricted the shapes of holes to be rectangular. As solutions, context-aware inpaintings using partial convolutions (PCs) [36] and gated convolutions [52] were suggested to handle irregular holes, but they had difficulty dealing with high-resolution images in real time and performing geometrically correct inpainting due to the lack of 3D information (e.g., edges of a building in a hole were rarely preserved).

For complete urban-scene rendering with a free user's viewpoint, our method proposes a novel semantic 3D technique for natural, context-aware inpainting that can handle large holes in real time.

3 Method

With the ultimate aim of providing a highly realistic experience of real-world urban streets in VR, this paper proposes a novel instant texture-mapping system using sparsely sampled panoramic street-view images and open 3D models that cover large-scale global urban scenes. As presented in Fig. 2, the proposed system comprises two main parts: a pre-process for obtaining input resources and constructing a 3D scene to be rendered (Sect. 3.1), and a real-time process that involves both panoramic texture mapping (Sect. 3.2) and semantic 3D inpainting (Sect. 3.3). To effectively enhance the rendering quality, our main idea is to deploy semantic information extracted from the street-view images in proper intermediate steps, such as image-3D model registration for the refinement of raw extrinsic camera parameters from Google Street View, semantic object-matching testing, and semantic 3D inpainting (Sect. 3.3), as indicated in red boxes in Fig. 2.

To assign various types of panoramic-scene information captured from a k^{th} camera, including an RGB color in PI_k (Panoramic Image), a semantic object label in PS_k (Panoramic Segmentation), and a depth value in PD_k (Panoramic Depth), to a 3D surface point, it is necessary to know mapping relationship between the pixel position on the panoramic image and the corresponding surface point [13]. For this, our system basically uses projection function E from 3D point p^k_{view} in the camera coordinates to pixel u_k on a panoramic image with camera index k, based on the well-known spherical projection E_{sphere} and equirectangular projection E_{equi} in sequence:

$$u_k = E(p^k_{view}) = E_{equi}(E_{sphere}(p^k_{view})). \qquad (1)$$

Please refer to the detailed explanation of this fundamental transformation in our supplementary material.

3.1 Resource Pre-acquisition

To render a wide variety of urban scenes using the proposed panoramic texture-mapping system in real time, it is important to obtain five different fundamental resources in advance, including a set of RGB street-view images PI, 3D geometric data M, sets of semantically segmented images PS, depth images PD, and refined extrinsic camera parameters C. The last three sets are paired with the corresponding PI, as described in *Pre-process* in Fig. 2. They are also managed using *3D Scene Data* to create a virtual scene that is actually rendered through the following real-time process.

Open Data for Street-View Images and 3D Models. Thanks to the development of techniques for global data acquisition using airborne or panoramic images with an omnidirectional camera or inertial sensors, it has become easier to obtain various global resources. Furthermore, the existing map-service platforms or open databases usually provide these data; thus, our system assumes that the basic resources PI and M can be supported in most cases. In this study, Google Street View images with 6,656×3,328 resolutions, which can be downloaded using APIs provided by the Google Maps platform, are used as PI and City GML 3D models from the open database [1] are used as M.

The three remaining types of resources need separate acquisition processes, as explained in the following subsections. Since the semantic information (PS) plays a significant role in our system in enhancing the quality of the proposed panoramic texture mapping, we estimate such information from PI using a deep learning-based method. Then, the inferred PS is used for the image-model registration between PI and M to refine the extrinsic camera parameters C. Finally, we render panoramic-depth (PD) images using M and C for a depth test in real-time texture mapping.

Deep Learning-Based Semantic Segmentation. To extract semantically segmented pixel information from a street-view image, we deploy DeepLabv3+ [9], which is one of the leading deep learning-based approaches, as ranked by the Cityscapes Benchmark Suite[1]. To encode multi-scale contextual information, it used Atrous Spatial Pyramid

[1] https://www.cityscapes-dataset.com/benchmarks/

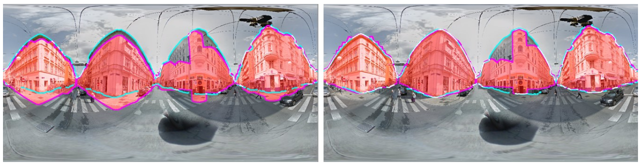


Fig. 3: Camera-pose refinement result. Compared with a registration result using a raw camera pose from Google Street View (left), building models (red) on the right-hand image are better aligned with a street-view image when using a refined camera pose computed by image-model registration deploying inferred semantic information.

Pooling (ASPP) applied to its previous version [8], and also suggested an effective decoder for refining the boundaries of segmented objects.

Although this network was not trained for panoramic-input images, which may result in distortion errors, we found that segmentation results were acceptable for our tested data. One reason for this may be that the network has been well trained on the Cityscapes dataset [12], which provides 5,000 large-scale urban scene images with high-quality dense pixel annotations of 30 classes, which are fit to our target scenes. For example, a segmentation image in the first row in Fig. 1 demonstrates properly segmented boundaries of the sky, buildings, and the ground, which are fundamental classes used by our system.

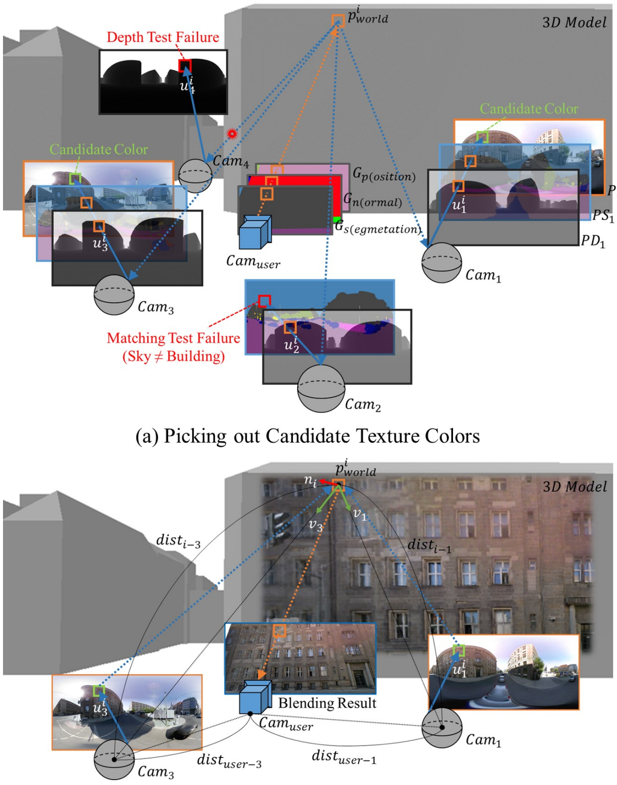
Camera-Pose Refinement Using Image Registration. For accurate panoramic image projection E in Equation 1, it is important to properly obtain the camera parameters used to capture a resource texture. Even though Google Street View provides extrinsic camera parameters, including latitude, longitude, and elevation values, estimated by GPS and IMU [2] for a corresponding panoramic image, such parameters still have errors of many meters, especially in the case of urban environments, for which it is challenging to measure GPS data [32].

In this study, an effective image-3D model registration [47] based on outline alignment is employed to refine a six-DoF extrinsic camera pose C_k. The advantage of this method is that we can simply perform optimization for individual cameras, unlike camera-pose estimation using SfM [32] or bundle adjustment, which jointly consider multiple cameras and require much time to operate, especially when a target scene becomes larger. This image registration method optimizes an energy function, which reaches its zero value when u_k projected from a world-space point, p_{world}, of the 3D models exactly overlaps on PI_k, thus causing the outlines from the 3D models to be aligned with those from PI_k. To realize this, we use a raw camera pose from Google Street View as an initial value and then minimize a loss function:

$$\underset{c_k}{\arg\min} \left\| \Lambda(C_k) - L_k \right\|_1, \tag{2}$$

where the outlines of the 3D models can be efficiently rendered through function Λ using C_k with camera index k in a Unity shader (e.g., pink lines in Fig. 3), and the boundaries L_k from PI_k can be extracted using a previously obtained semantic segmentation map PS_k (e.g., cyan lines in Fig. 3). Each pixel in those outline images has a binary value (0: non-outline pixel, 1: outline pixel), and Equation 2 is minimized when the optimal C_k is found, realizing the best registration.

To effectively find the best C_k, we deploy particle-swarm optimization [11] using an evolutionary sampling method, where a set of random camera poses, with user-defined thresholds for translation and rotation, are iteratively sampled and for each iteration, a better sample of C_k is searched by considering both a locally found optimum and a globally found one shared by all the other samples. Note that our method depends on deep learning-based semantic segmentation, unlike the classification method used in the original work [47]. That method employed conditional random fields [34] trained on 70 labeled images, which required additional iterative refinement during the image registration process. In Fig. 3, the registration result using a refined camera pose demonstrates qualitatively better alignment than that using a raw camera pose from Google Street View.



(a) Picking out Candidate Texture Colors

(b) Weighted Blending

Fig. 4: Visualized panoramic texture-mapping process. After mapping world-space point p^i_{world} to pixel position u^i_k in the panoramic image space of k^{th} camera Cam_k, p^i_{world} is required to pass a depth test and a semantic object-matching test. In the case of Cam_4, p^i_{world} cannot pass a depth test as it is occluded by geometry and invisible to Cam_4. Although p^i_{world} passes a depth test for Cam_2, its semantic information does not match an object class of u^i_2 in a street view PI_2 due to misalignment between real and virtual geometries. Colors from PI_1 and PI_3 are good candidates for blending described in (b). When deciding a blending weight, distances between p^i_{world} and Cam_k ($dist_{i-k}$), and between user's camera Cam_{user} and Cam_k ($dist_{user-k}$), as well as the cosine between normal vector n_i of p^i_{world} and camera vector v_k are fully considered, providing more user location optimal blending results.

Panoramic-Depth Image. Our method renders a panoramic-depth image PD_k using 3D scene models M and a previously refined camera pose C_k in advance and deploys PD_k for a depth test. More details can be found in our supplementary material.

Sampling Street-View Images from Districts. For the effective resource management, we employ a sparse sampling technique based on the regularly divided world space similar to the partitioned global geometry [25]. During preprocessing, the X-Z plane is separated into small tiles, which we call *District*s, based on the user-selected values (50 m, in our system) of width and height. Of the street-view cameras located in a district, we sample the one nearest to the district's center position, as indicated by yellow circles in *3D Scene Data with Districts* (Fig. 2). For the user's current district, a maximum of eight adjacent districts are deployed in a real-time texture-mapping process. Please refer to our supplementary material for more details.

3.2 Panoramic Texture Mapping

In this section, we explain the proposed real-time panoramic texture mapping that deploys previously acquired resources for large-scale urban-scene reproduction. In each frame, our method renders G-buffers, including position buffer G_p, segmentation buffer G_s, and normal buffer

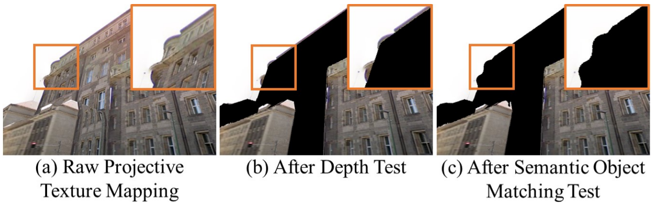

(a) Raw Projective Texture Mapping
(b) After Depth Test
(c) After Semantic Object Matching Test

Fig. 5: Effects of the texture-filtering tests. When using raw projective texture mapping without any tests, textures pass through geometries and are mapped onto all intersecting surfaces (a). Although depth tests roughly filter out incorrectly projected textures (b), errors still remain due to imperfect depth information from simplified 3D models. Our system enhances projection accuracy using semantic information (c).

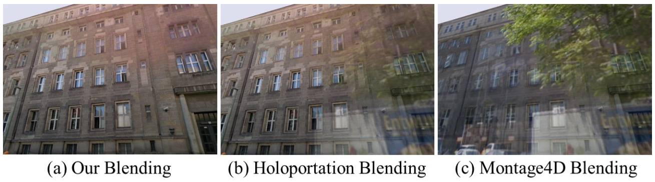


Fig. 6: Blending result comparison. Compared with our blending weight that considers the user's current position, Holoportation [39] sets a weight using cosine between a surface normal vector and a resource-camera vector, whereas Montage4D [15] uses cosine between a user's camera vector and a resource-camera vector. Since our method can recognize that trees behind a user are invisible in the current view, proper building-texture colors are mapped without the trees.

G_n, according to the user's current camera view. Then, using the view matrix and projection E in Equation 1, every world-space point p^i_{world} in G_p is mapped onto the corresponding pixel position u^i_k on each sampled street-view image PI_k with camera index k. Finally, through this mapping, texture colors can be assigned from the sampled PI_ks to possible pixels in the current view, as presented in Fig. 4-(a). However, not all such colors can be used to produce a final color for a specific surface pixel due to possible projection errors in Fig. 5-(a).

Thus, our system selects available candidate texture colors from sampled PI_ks and calculates proper weights to blend them. To effectively enhance texture mapping accuracy, we propose to use not only PD_ks to filter out wrongly projected texture colors on occluded surface points in a depth test, but also PS_ks to choose contextually correct candidates in a semantic object-matching test, followed by novel blending weights to render perceptually natural results, as shown in Fig. 4.

Depth Test. The purpose of the depth test is to handle a well-known penetrating problem of naive projective texture mapping [13], whereby the pixel colors of texture PI_k visible to camera k may be incorrectly mapped onto the occluded geometry behind them. Since our system already rendered a synthetic panoramic-depth image, PD_k, of the 3D scene models according to refined extrinsic camera parameters C_k, it can simply decide whether or not an arbitrary 3D point p^i_{world} in G_p is visible to camera k by comparing the distance between the camera position and p^i_{world} with the depth value of pixel u^i_k in PD_k.

Although this depth test can eliminate most projection errors, it is not an optimal solution due to the following reasons: (1) an imperfect depth map PD_k can arise with simplified 3D models, and (2) calibration errors of the refined extrinsic camera parameters C_k may result in misregistration between the real geometry in a street-view image PI_k and the synthetic scene geometry. Consequently, texture colors can still be projected onto a semantically inappropriate target object, as shown in Fig. 5-(b). Therefore, our method performs an additional semantic object-matching test to precisely remove such remaining errors.

Semantic Object-Matching Test. Unlike previous works focused on mapping a texture onto a specific object of interest [15,50], the proposed system aims at simultaneous texture mapping of various objects in a whole scene, requiring more elaborate matching between a texture and a target object. Therefore, to solve the remaining projection errors in a depth test, the deployment of semantic information on the respective objects in the current scene can be an effective solution to precisely assign texture colors. We are not the first to utilize semantic information to address this problem. Xiao et al. [50] also used such information to separate a reconstructed point cloud into respective buildings and to compose their textures. However, its offline usage was limited to dealing with buildings without considering other important elements, such as the sky or the ground, which should be properly rendered for realistic urban scenes.

Our method uses panoramic semantic segmentation map PS_k in the real-time texture-mapping process to assign a color from PI_k to a correct target object by conducting an object-matching test. Since PS_k estimated previously from the DeepLabv3+ network [9] can classify each PI_k pixel based on pre-trained object classes and various urban scenes in the Cityscapes dataset [12], we can obtain accurate and clear boundaries for the different object areas. Specifically, assuming that a 3D model has a known label defined in DeepLabv3+, the semantic object-matching test succeeds when the label of a surface point p^i_{world} on the model and an estimated class of u^i_k in PS_k are the same. Hence, our system can filter out an incorrect texture color projected onto a mismatched object (e.g., a removed texture color of a building projected onto the sky, as in Fig. 5).

Multi-texturing Using Weighted Blending. After completing the previous tests, the selected candidate texture colors from the sampled street-view images PI_k must be interpolated to synthesize the final color. Basically, given N candidate colors, output color o_i is interpolated using normalized weights $\hat{w}^i_k$ as follows:

$$o_i = \sum_{k=1}^{N} \hat{w}^i_k \cdot I(k, u^i_k), \tag{3}$$

where u^i_k denotes a projected pixel position of a surface point p^i_{world} on PI_k with camera index k through the projection function E in Equation 1, and texture function I gives a pixel color at u^i_k. For high-quality texture mapping, a proper blending method needs to be established considering three key factors that are described in [15]: smoothness for lowering discontinuities between different textured areas, sharpness for preserving details of the original texture, and temporal consistency of the blending results when a user's viewpoint changes.

Starting from the pioneering work of Debevec et al. [13], which set blending weights based on the angle between a surface normal and a texture-camera vector, various methods have been introduced for choosing proper weights for image-space blending [17, 42] or real-time VDTM [15, 39]. In particular, the Holoportation [39] and Montage4D [15] algorithms effectively decreased ghosting effects by deploying dilated depth maps and majority-color voting with well-calibrated camera parameters in an indoor environment. However, these are insufficient for application to our system, which uses fewer and more sparsely sampled textures with imperfect camera parameters C_k to cover a large-scale urban scene.

Therefore, we use a novel blending weight motivated by a fundamental energy-transfer equation [3] for calculating intensity of incident light upon a surface based on the inverse-square law between energy and distance. Furthermore, the proposed weight considers geometric relationships not only between texture camera C_k and p^i_{world}, but also between C_k and the user's current position as follows:

$$w^i_k = \frac{max(0,\, n_i \cdot v_k)^{\alpha}}{(dist_{i-k})^2 (dist_{user-k})^2}, \tag{4}$$

where n_i denotes a normal vector on surface point p^i_{world}, and v_k denotes a camera vector from p^i_{world} to k^{th} camera position t_k. α denotes a smoothness term ($\alpha = 4$, in our system). Also, $dist_{i-k}$ and $dist_{user-k}$ denote distances between p^i_{world} and t_k and between the user's position and t_k, respectively. As demonstrated by the comparative examples in

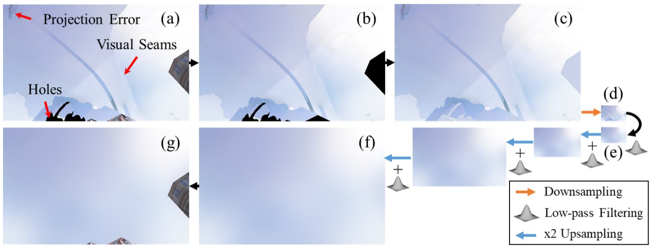


Fig. 7: Seamless sky low-pass filtering. After panoramic texture mapping, the raw-input sky still has visual errors (a). So, our system first separates sky areas (b), and then fills in holes with an average sky color (c). After downsampling this image (d), a 5×5 Gaussian filter is applied for convolution (e), and iterative ×2 upsampling and low-pass filtering are executed until the image has the original resolution (f). Finally, the refined sky is composed with previously separated non-sky areas (g).

Fig. 6, the main advantage of the proposed blending weight is its ability to calculate more accurate colors at the user's location while observing the three key factors explained for blending.

3.3 Real-Time Inpainting Using Semantic Segmentation

Even though sparsely sampled street-view images can be mapped onto most of the 3D scenes currently in the user's view, occluded and untreated surfaces invisible to any texture camera still remain. Since our system aims to provide six DoFs for the user's viewpoint, such untextured areas should be properly handled for the complete rendering.

In the first trial, we applied previous 2D inpainting methods [27, 42] and recent deep learning-based methods [36, 52], which yielded advanced context-aware inpainting results but still required much time, making them unsuitable for our real-time system. Therefore, we suggest a novel inpainting method that separately manages the sky and non-sky areas to effectively reduce the computation time. Specifically, cascaded low-pass filters are applied to the sky areas and deal with untextured holes and remaining visual errors simultaneously. For non-sky areas, the proposed inpainting method fills geometrically and semantically matching texture colors in real time.

Seamless Sky with Cascaded Low-Pass Filtering. When mapping multiple textures onto the sky, we found three notable artifacts: 1) visual seams due to differing conditions of the sky, such as brightness and shapes of clouds, as captured in respective street-view images; 2) remaining projection errors caused by imperfect classification of the employed DeepLabv3+; and 3) untextured sky areas, which cannot be covered by sampled street-view images due to occlusion by skyscrapers, as shown in Fig. 7-(a).

To effectively handle these problems, we focus on the fact that the sky contains relatively lower-frequency information than buildings, which have sophisticated textures, such as patterns on exterior walls or signboards. Thus, instead of using expensive algorithms to remove visual seams and fill-in holes, we apply multi-level low-pass filtering to sky areas to solve these issues simultaneously. As shown in Fig. 7, a previously texture-mapped scene is first divided into sky and non-sky parts; then, the untextured sky areas are filled with an average sky color for the current frame. Subsequently, we use the k-nearest neighbor downsampling to resize this image into 32×16 resolutions. Here, a 5×5 Gaussian filter with a standard deviation of 5.0 is applied to the image to convolute and naturally blur pixel colors. Finally, the width of the image doubles through every upsampling with the low-pass filter until the image size returns to its original value. Thus, our method yields a convincing result without visual seams, projection errors, or untextured pixels of the sky while still containing meaningful color variations, such as the top-right bright areas of white clouds in Fig. 7-(f).

Semantic 3D PixMix. Unlike the sky, untextured non-sky areas need to be seamlessly filled with high-frequency and semantically proper texture colors to provide a realistic urban scene. Therefore, the proposed method uses an advanced inpainting technique, which considers the 3D geometric and semantic information of a current scene based on the high-quality 2D inpainting method called PixMix [27].

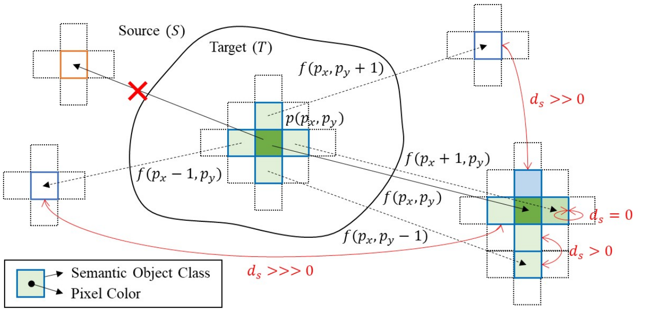


Fig. 8: Semantic 3D PixMix inpainting. A pixel $p(p_x, p_y)$ in T needs optimal transformation function $f(p_x, p_y)$ to find a proper source pixel in S. Our system uses a total cost function containing a spatial cost, an appearance cost, and a matching cost. With regard to the spatial cost, $f(p_x+1, p_y)$ has zero cost while maintaining the same spatial structure both in T and S. Contrarily, $f(p_x-1, p_y)$ is the worst case, as it aggravates the spatial cost. In the case of the appearance cost, adjacent pixel colors of p need to be similar to those of transformed p in S. Finally, to minimize the matching cost, a replaceable source pixel should have the same label as that of p. Since p is labeled in blue, it cannot be replaced by a source pixel with an orange label.

PixMix yielded convincing results by iteratively finding per-pixel transformation function $f : T \rightarrow S$ from target pixel p in untextured area T to a source pixel in already-textured area S, as described in Fig. 8. When the proper f is found for every p, the color of each p is replaced by a determined source color in S for a final image. To obtain the best mapping function f, PixMix solved the global minimization problem of a total function consisting of two types of costs, i.e., spatial cost $Cost_s$ and appearance cost $Cost_a$ [27]:

$$Cost_s(p) = \sum_{\overrightarrow{v} \in N_s} d_s[f(p+\overrightarrow{v}), f(p)+\overrightarrow{v}] \cdot w_s(\overrightarrow{v}), \quad (5)$$

$$Cost_a(p) = \sum_{\overrightarrow{v} \in N_a} d_a[h(p+\overrightarrow{v}), h(f(p)+\overrightarrow{v})] \cdot w_a(p+\overrightarrow{v}). \quad (6)$$

Here, $p = (p_x, p_y)^\top$ denotes a pixel position in T, and $N_{(\cdot)}$ denotes relative positions of neighboring pixels around p, which contains element vectors $\overrightarrow{v}$ from p to neighboring pixels, except $\overrightarrow{v} = 0^\top$. As in the original work, patch sizes of 3×3 and 5×5 were used for N_s and N_a respectively, and importance weight $w_{(\cdot)}$ basically has uniform weight $w_{(\cdot)}(\overrightarrow{v}) = 1/|N_{(\cdot)}|$. However, in the case of w_a, an adjacent pixel on the border between T and S selectively has a large weight for a high impact on $Cost_a$, which can reduce the border effects. For distance functions d_s and d_a, a squared Euclidean distance is used. Intuitively, minimizing $Cost_s$ means that neighbors of target pixel p are nearly mapped onto those of $f(p)$ while maintaining similar surrounding structures. When $h(p)$ denotes a pixel color of p, minimizing $Cost_a$ helps p have neighboring pixels colors similar to those of $f(p)$.

Although PixMix also yielded convincing real-time video-inpainting results, it had difficulty filling in holes with a non-planar background and lacked semantic matching between the source and target pixels. Our method extends PixMix for deployment to 3D geometry and semantic information of a current scene using the following total cost function:

$$Cost_{total}(p) = \alpha \cdot Cost_s(p) + (1-\alpha) \cdot Cost_a(p) + Cost_m(p). \quad (7)$$

Here, the matching cost $Cost_m(p)$ is 0 when a semantic label of p and that of $f(p)$ are the same, whereas it is ∞ in the other cases. Note that we can obtain the label of target pixel p from a known class of a 3D model to which p belongs; however, the label of source pixel $f(p)$ is obtained from PS_k of the most influential candidate texture PI_k selected in the previous weighted blending step. This facilitates in more accurately finding a semantically proper source pixel than by using just

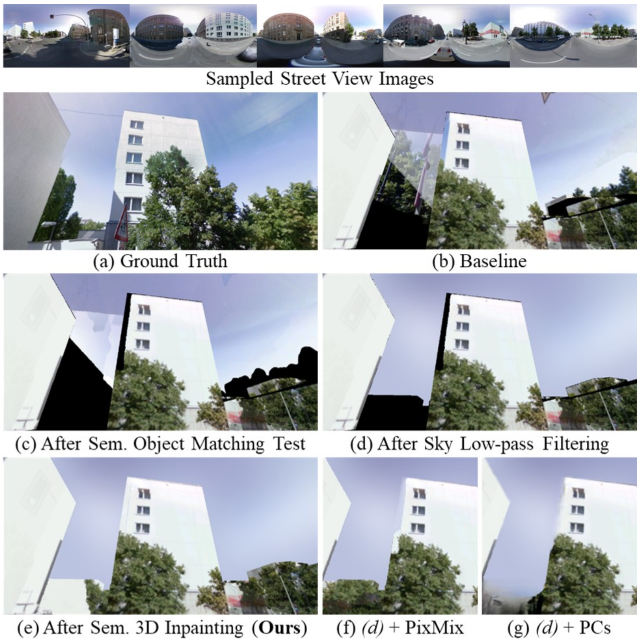

Sampled Street View Images
(a) Ground Truth
(b) Baseline
(c) After Sem. Object Matching Test
(d) After Sky Low-pass Filtering
(e) After Sem. 3D Inpainting (**Ours**)
(f) *(d)* + PixMix
(g) *(d)* + PCs

Fig. 9: Intermediate results with qualitative comparisons against recent inapinting methods. *Baseline* uses panoramic texture mapping with only depth testing (b). After semantic object-matching test, most projection errors in sky areas can be removed (c). With sky low-pass filtering, visual errors and holes are simultaneously handled (d). Finally, our semantic 3D inpainting naturally fills remaining holes, which requires more detailed textures (e). Note that, contrary to ours, PixMix [26] cannot fill in contextually correct colors on a building side (f), whereas deep learning-based method using partial convolutions (PCs) [36] loses geometric information, causing blurred building edges in the holes (g).

semantic labels of the 3D models for both p and $f(p)$. To consider 3D geometric structures already known from G_p, we calculate the spatial cost using 3D positions of $f(p+\overrightarrow{v})$ and $f(p)+\overrightarrow{v}$, and a 3D distance function d_s. To set the control parameter $\alpha \in [0,1]$, we use 0.5.

The main advantage of using semantic information is that the search range for $f(p)$ can be effectively decreased by sampling contextually matching source pixels. Furthermore, our method can better fill large holes with a non-planar background than the method in the previous work, as it uses 3D geometric structures, as shown in Fig. 9. Please refer to our supplementary material for more detailed parameters.

4 Results and Analysis

4.1 Evaluation of System Performance

To validate necessity of the suggested techniques, including the semantic object-matching test, low-pass filtering for seamless sky areas, and semantic 3D inpainting, we first demonstrate how those respective steps can increase the quality of novel-view synthesis by comparing each intermediate-rendering result with a ground-truth novel view sampled from a Google Street View image.

Experimental Design. To sample test-image sets, we construct experimental areas of $1km^2$ in three different global urban scenes, including Berlin in Germany, Vienna in Austria, and Daejeon in South Korea; their 3D models can be obtained from open databases [1,45]. For each test area, Google Maps provides about 850 street-view images, of which 350 are typically used for the proposed texture-mapping method when we use a district size of $50m^2$. Then, we randomly select 20% of the remaining 500 street-view images, such that 100 testable street-view images with corresponding camera parameters can be selected.

Then, using each selected street-view image, a ground-truth novel view is rendered with a random camera direction, a FoV of 60°, and image resolutions of 1024×512, as shown in Fig. 9-(a). With the same camera and image configurations of the corresponding ground truth images, we also render novel views using respective progressive steps of the proposed system. Specifically, for the *Baseline* rendering shown in Fig. 9-(b), colors from sampled street-view images are mapped onto surfaces of 3D models after a depth test and after weighted blending is performed, similar to typical VDTM works [13, 15]. Then, using the respective proposed steps, the intermediate results are rendered.

Table 1: Quantitative results for intermediate steps. Using 300 arbitrary test views from 3 different urban settings, respective steps, as shown in Fig. 9, are measured against ground truth by RMSE and SSIM.

Intermediate Steps	RMSE	SSIM
Baseline	0.309	0.510
+ Semantic Object Matching Test	0.306	0.539
+ Sky Low-Pass Filtering (Ours w/ Holes)	0.305	0.553
+ Semantic 3D Inpainting (**Ours**)	**0.244**	**0.584**
Ours w/ Holes + PixMix [26]	0.259	0.561
Ours w/ Holes + Partial Convolutions [36]	0.252	0.559

Errors between the ground-truth and intermediate-rendering results are measured using two different metrics: the root-mean-square error (RMSE) and the structural similarity index measure (SSIM) [49]. Note that the measurement of only absolute per-pixel differences using RMSE may be insufficient as even subtle errors in the obtainable camera parameters and geometries can have excessive effects. In this regard, SSIM can be an appropriate supplementary metric for measuring perception-based errors while considering structural similarities between two images.

Results. Quantitative results presented in Table 1 indicate that each proposed step enhances the rendering quality in terms of both RMSE and SSIM. However, overall variance of SSIM is bigger than that of RMSE, showing that each step has a greater effects on the structural and perceptual accuracy than on pixel-by-pixel correctness. For example, in Fig. 9-(c) and (d), when using a semantic object-matching test to select candidate texture colors and low-pass filtering for sky areas, it is noticeable that erroneously mapped object colors and visual seams can be appropriately removed from the sky areas in respective steps, whereas those steps hardly reduce RMSE. The biggest improvement is observed in both RMSE and SSIM after the suggested semantic 3D inpainting is performed, thus filling in the occluded and untextured areas. This final step plays a significant role in our system that uses sparsely sampled textural images to provide users with a completely free viewpoint at a street level. Finally, compared with *Baseline*, our method demonstrates quantitative improvements in the RMSE and SSIM of 21% and 14.5%, respectively.

In addition, for more comprehensive experiments, we also compare our inpainting method with two recent works, namely, PixMix [26] and a deep learning-based method using PCs [36]. Although those two methods enhance rendering quality well (as shown in Table 1), PixMix usually lacks semantic information, and the learned network generates geometrically inadequate textures in holes, as shown in Fig. 9-(f) and (g). Note that our inpainting method also cannot fill in accurate texture colors when compared with the ground truth, but tends to show a more advanced structural similarity, as demonstrated by the SSIM results.

System Performance. When tested on a computer equipped with an Intel Core i7-6700k CPU and NVIDIA GeForce GTX 1080 GPU, the proposed system can guarantee an average of 50 fps with image resolutions of 1024×512. However, since the rendering performance of our method depends on the complexity of the geometry, it can be controlled when we properly manage the number of vertices rendered in a single frame. Specifically, when the geometry is highly complex and has 120 k vertices, the whole rendering process is completed in 18 ms (55 fps), whereas a simpler geometry with 77 k vertices can be rendered in 9.8 ms (102 fps). In the proposed pipeline, semantic 3D inpainting is the most time-consuming step, which accounts for about 30% of the whole computation time.

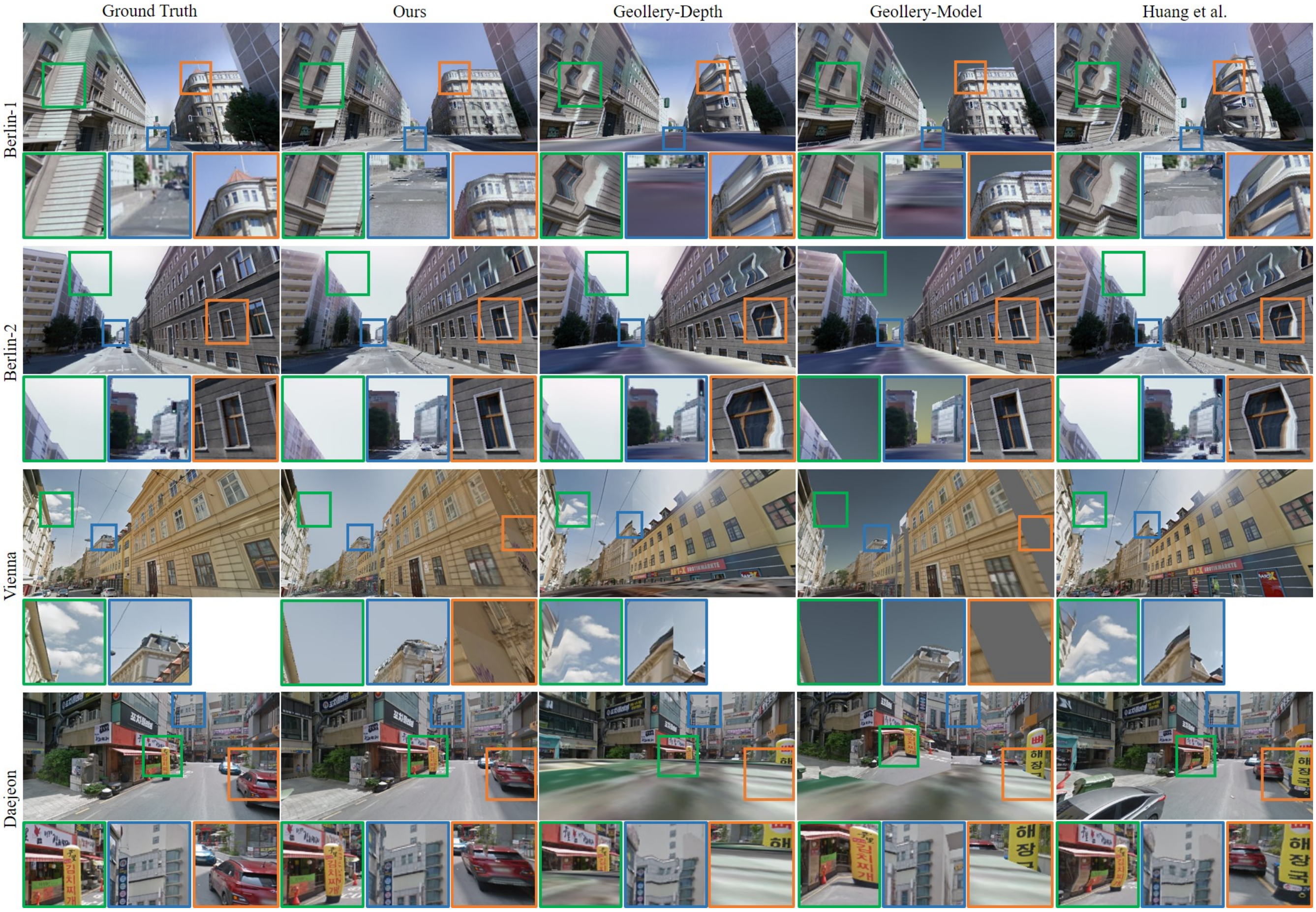


Fig. 10: Qualitative comparisons in various test scenes. Unlike our method, those of Geollery-Depth and Huang et al. often show distortion errors due to imperfect sphere-based geometry, and Geollery-Model cannot reproduce sky areas; refer to details in the corresponding part in the paper.

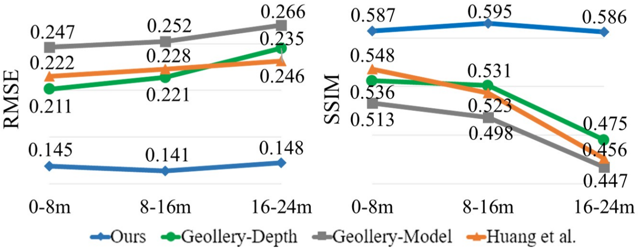


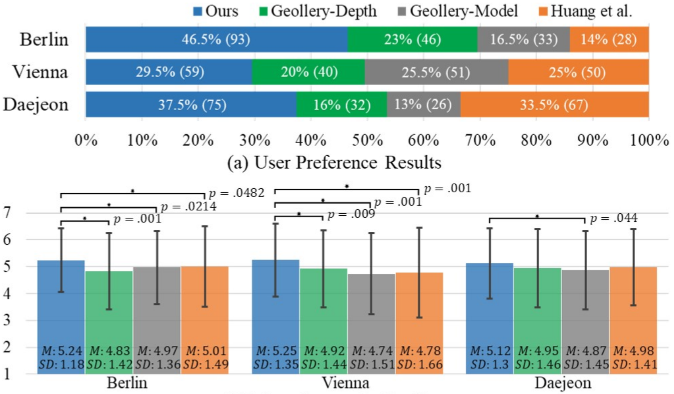
Fig. 11: Quantitative comparison results. Using 270 views in three urban test settings, the quality of novel-view synthesis is measured in terms of the RMSE and SSIM according to the distance between a novel view and the nearest-street view image used for texture mapping. Note that our method yields better and relatively more stable results than the other three methods, regardless of the distance, whereas the errors of the other methods increase with the distance.

4.2 Comparisons

Experimental Design. To compare the proposed method with the state-of-the-arts, we employ three different approaches, including the respective depth- and model-based panoramic texture mappings suggested in Geollery [16] by Du et al. for real-time urban scene experiences and the sphere-based image warping by Huang et al. [29], which targets six-DoF novel views using image streams. Specifically, the depth-based approach of Geollery (*Geollery-Depth*) samples the street-view image nearest to the user's current position and its corresponding panoramic-depth map provided by Google Street View [2]. Moreover, it deforms a spherical mesh according to the depth map to efficiently obtain the local proxy geometry at which colors of the street-view image are mapped. Conversely, a model-based approach (*Geollery-Model*) uses 3D models from open databases, such as OpenStreetMap [21], and also maps the nearest street-view image onto a geometry with a simple back-face test to prevent unseen surfaces from being textured. The work of Huang et al. warps a single panoramic RGB image from a reference view to a novel view based on the optimization between two spherical meshes. Note that, in this experiment, we reconstruct dense point clouds from depth images from Google Street View. Although a video input is used in the original work, this cannot be obtained for the global test scenes.

For evaluation, the same test scenes and metrics are used as in Sect. 4.1, whereas testable street views are differently sampled for an intensive comparison of not only the rendering quality at a random novel viewpoint but also the rendering accuracy according to the distance range between a novel viewing position and the nearest reference street view sampled for texture mapping. For each urban scene, we randomly sample 30 street-view images for each of three different distance ranges. Since our system uses a district length of 50 m and samples street view PI_k used for texture mapping near the center of the district, the average distance between PI_k and one of the furthest novel views in the district is about 25 m. Thus, for the experiment, we selected 8 m as a proper distance for three different ranges, namely, 0-8, 8-16, and 16-24 m, where a user's novel view can be rendered. Thus, a total of 90 pairs of novel views rendered by respective methods and a ground truth are constructed for each of the Berlin, Vienna, and Daejeon scenes.

Results. For a qualitative comparison, Fig. 10 presents four exemplary novel views rendered by four different methods. Our method qualitatively yields better rendering results in the overall test scenes



(a) User Preference Results

(b) Mean Scores for Realism

Fig. 12: User test results. For each scene, preferred methods are selected by 200 online participants in (a), and each video is scored with a 7-Likert scale in (b). P-values are calculated using the paired t-test, showing significant differences between two means.

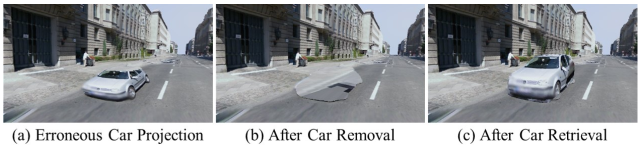
with fewer distortion errors than Geollery-Depth and the work of Huang et al.. Moreover, it offers better texture-model registration and sky reproduction than Geollery-Model. For example, in the *Berlin-1* scene, the green and orange boxes indicate that ours provides significantly better rendering quality compared with Geollery-Depth and the work of Huang et al., which employ sphere-based limited geometry, thus resulting in distortion errors of geometry or texture mapping. With regard to the blue boxes, Geollery-Depth and Geollery-Model render a road using low-resolution satellite images, whereas the other methods map colors from reference street-view textures. However, distorted textures can still easily be found in the work of Huang et al..

In the *Berlin-2* scene, the green box in our result indicates well-represented sky areas when compared with the ground truth based on the proposed cascaded low-pass filtering. Note that Geollery-Model cannot cover sky rendering, and Geollery-Depth and the work of Huang et al. cannot correctly map sky textures onto the sky areas without any object-texture-matching test, although their results seem convincing. One interesting difference between our result and that of Geollery-Model can be seen in the orange boxes in the *Vienna* scene. Ours can naturally fill in untextured areas using the proposed semantic 3D inapinting method, whereas Geollery-Model simply fills in holes with a single color. Note that our method has limitations when recovering detailed patterns of the sky, as shown in the green box. We will further discuss the issue of erroneous texture projection, as indicated by the orange box in *Daejeon* in Sect. 5.

In terms of the quantitative comparison presented in Fig. 11, our method shows the lowest per-pixel errors and the highest structural similarity with ground-truth images, regardless of the distance ranges between a reference street view and a novel view. One interesting point is that errors of the other methods tend to increase in proportion to the distance ranges, whereas our model has a relatively constant error; this indicates that the quality of a synthesized novel view can be consistently preserved when a user's position dynamically changes.

We attribute the main reason for this result to be that ours can render a novel view even with sparsely sampled street-view images by deploying the proposed weighted blending based on a current user's position and the proper inpainting technique. Contrarily, the other methods mainly depend on the nearest resource texture, which only covers a limited range of user movement, thus resulting in a decrease in the rendering quality when a user gets far from the sampled texture. Note that Geollery's original work required frequent sampling steps before a previous resource texture gets further away from a current user's position, which results in a fluctuating rendering quality [16].

User Test. In order to assess how realistic urban scene rendering results of different methods is for typical users, we conducted a perceptual online user test, instead of a face-to-face manner due to the COVID-19 pandemic. All procedures were approved by an independent Institutional Review Board. Using Amazon Mechanical Turk, we obtained 200 paid participants, including 128 males and 72 females, ranging from 20 to 70 years old (35.5 average), regardless of familiarity with VR or related rendering systems.



(a) Erroneous Car Projection (b) After Car Removal (c) After Car Retrieval

Fig. 13: To handle projection errors of detailed objects (a), we may remove objects using semantic information (b) or map textures on retrieved object models from accessible databases (c).

When a test started, each participant was required to watch 15-second short videos with 1024×512 resolutions, for virtual walk-through experiences in three test urban scenes. In each scene, the same path was rendered by four different methods as used in the previous experiment, and such four videos were shown in a random order. As the first question for user preference, we asked participants to select one video which seemed the most realistic for each scene. In addition, participants were asked to assess realism of each video, based on the question of Zivrek et al. [54], using a Likert scale with 7 positions, starting from low to high. For the analysis of realism, we used the paired sample t-test ($\alpha = .05$) to determine significant differences between means of two methods.

As shown in Fig. 12-(a), users preferred ours in all test scenes among the four different methods, although there was small gap between ours and the work of Huang et al. in the Daejeon scene. Furthermore, as described in Fig. 12-(b), with specific mean and standard deviation values, ours also showed the highest mean scores for realism in three different scenes. Note that specified p-values demonstrate significant differences between the mean value of ours and that of each compared method in most cases, except a few cases in Daejeon. We think the reason is that the Daejeon scene has complex geometries with various small objects that our method cannot fully cover, thus resulting in more projection errors and decrease of realism.

5 Discussion and Conclusion

This paper proposed a novel panoramic texture-mapping system using open global scene data to provide realistic walk-through experiences in large-scale urban streets, which can be available for various VR applications reflecting the real world. Our system mainly used semantic information obtained from street-view images to enhance both the rendering quality and performance time. Moreover, with sparsely sampled street-view images, our inpainting technique supported perceptually natural scene rendering, even when the user's viewpoint dynamically changed, as is also shown in the supplementary material and video.

Our method has some remaining issues that need to be solved. Although we further calibrated camera poses of street-view images using image-model registration, accurate values could not be found; this resulted in erroneous texture mapping on 3D scene models. Also, since the current system's performance depends on the number of vertices of 3D scene models, it is hard to render all geometries located in the far distance. Thus, a proper way to effectively manage renderable vertices is required. On the extension, the gap between the real geometry and the simplified virtual geometry should be decreased by using 3D models with a high level of detail and proper resource management which can affect real-time performance. Furthermore, since the current system does not support detailed objects geometries, projection errors of such objects are also found on buildings or roads, as shown in Fig. 13. We suggest some possible solutions for the removal or retrieval of those objects, which should be properly dealt with in the future work. Finally, our inpainting method fills in holes with contextually proper texture colors, but it still needs a temporally stable algorithm (e.g., instead of searching for locally optimal source pixels in each frame, the global solution can be obtained by considering previous frames, potentially resulting in more temporally coherent inpainting).

ACKNOWLEDGMENTS

This work was supported by Institute for Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2019-0-01648, Development of 360 Degree VR Content Authoring Platform based on Global Street View and Spatial Information, and No.2019-0-01270, WISE AR UI/UX Platform Development for Smartglasses).

REFERENCES

[1] 3d geoinformation research group. https://3d.bk.tudelft.nl/opendata/.

[2] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver. Google street view: Capturing the world at street level. *Computer*, 43(6):32–38, 2010.

[3] A. Appel. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pp. 37–45, 1968.

[4] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. In *ACM Transactions on Graphics (ToG)*, vol. 28, p. 24. ACM, 2009.

[5] A. Bulbul and R. Dahyot. Social media based 3d visual popularity. *Computers & Graphics*, 63:28–36, 2017.

[6] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG)*, 32(3):1–12, 2013.

[7] G. Chaurasia, O. Sorkine, and G. Drettakis. Silhouette-aware warping for image-based rendering. In *Computer Graphics Forum*, vol. 30, pp. 1223–1232. Wiley Online Library, 2011.

[8] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[9] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[10] S. E. Chen and L. Williams. View interpolation for image synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pp. 279–288, 1993.

[11] M. Clerc and J. Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[12] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

[13] P. Debevec, Y. Yu, and G. Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In *Rendering Techniques' 98*, pp. 105–116. Springer, 1998.

[14] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 11–20, 1996.

[15] R. Du, M. Chuang, W. Chang, H. Hoppe, and A. Varshney. Montage4d: Real-time seamless fusion and stylization of multiview video textures. *Journal of Computer Graphics Techniques*, 1(15):1–34, 2019.

[16] R. Du, D. Li, and A. Varshney. Project geollery. com: Reconstructing a live mirrored world with geotagged social media. In *The 24th International Conference on 3D Web Technology*, pp. 1–9, 2019.

[17] M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. De Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. Floating textures. In *Computer graphics forum*, vol. 27, pp. 409–418. Wiley Online Library, 2008.

[18] J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2367–2376, 2019.

[19] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5515–5524, 2016.

[20] N. Gorelick, M. Hancher, M. Dixon, S. Ilyushchenko, D. Thau, and R. Moore. Google earth engine: Planetary-scale geospatial analysis for everyone. *Remote sensing of Environment*, 202:18–27, 2017.

[21] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[22] P. Hedman, S. Alsisan, R. Szeliski, and J. Kopf. Casual 3d photography. *ACM Transactions on Graphics (TOG)*, 36(6):1–15, 2017.

[23] P. Hedman and J. Kopf. Instant 3d photography. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018.

[24] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018.

[25] P. Hedman, T. Ritschel, G. Drettakis, and G. Brostow. Scalable inside-out image-based rendering. *ACM Transactions on Graphics (TOG)*, 35(6):1–11, 2016.

[26] J. Herling and W. Broll. Pixmix: A real-time approach to high-quality diminished reality. In *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 141–150. IEEE, 2012.

[27] J. Herling and W. Broll. High-quality real-time video inpaintingwith pixmix. *IEEE Transactions on Visualization and Computer Graphics*, 20(6):866–879, 2014.

[28] F. Huang, Y.-J. Wu, J.-S. Hsu, and A. Tsai. 3d modeling of street buildings from panoramic video sequences and google map image. In *GRAPP/IVAPP*, pp. 109–114, 2012.

[29] J. Huang, Z. Chen, D. Ceylan, and H. Jin. 6-dof vr videos with a single 360-camera. In *2017 IEEE Virtual Reality (VR)*, pp. 37–44. IEEE, 2017.

[30] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):1–14, 2017.

[31] N. K. Kalantari, T.-C. Wang, and R. Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, 35(6):1–10, 2016.

[32] B. Klingner, D. Martin, and J. Roseborough. Street view motion-from-structure-from-motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 953–960, 2013.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[34] L. Ladickỳ, C. Russell, P. Kohli, and P. H. Torr. Associative hierarchical crfs for object class image segmentation. In *2009 IEEE 12th International Conference on Computer Vision*, pp. 739–746. IEEE, 2009.

[35] Y. Li, Q. Zheng, A. Sharf, D. Cohen-Or, B. Chen, and N. J. Mitra. 2d-3d fusion for layer decomposition of urban facades. In *2011 International Conference on Computer Vision*, pp. 882–889. IEEE, 2011.

[36] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 85–100, 2018.

[37] M. Liu, X. He, and M. Salzmann. Geometry-aware deep network for single-image novel view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4616–4624, 2018.

[38] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. Van Gool, and W. Purgathofer. A survey of urban reconstruction. In *Computer graphics forum*, vol. 32, pp. 146–177. Wiley Online Library, 2013.

[39] S. Orts-Escolano, C. Rhemann, S. Fanello, W. Chang, A. Kowdle, Y. Degtyarev, D. Kim, P. L. Davidson, S. Khamis, M. Dou, et al. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pp. 741–754, 2016.

[40] M. Over, A. Schilling, S. Neubauer, and A. Zipf. Generating web-based 3d city models from openstreetmap: The current situation in germany. *Computers, Environment and Urban Systems*, 34(6):496–507, 2010.

[41] E. Penner and L. Zhang. Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017.

[42] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. In *ACM SIGGRAPH 2003 Papers*, pp. 313–318. 2003.

[43] F. Prandi, F. Devigili, M. Soave, U. Di Staso, and R. De Amicis. 3d web visualization of huge citygml models. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 40, 2015.

[44] A. M. Siu, A. S. Wan, and R. W. Lau. Modeling and rendering of walk-through environments with panoramic images. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 114–121, 2004.

[45] SPACEN. Vworld. http://vworld.kr/.

[46] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[47] A. Taneja, L. Ballan, and M. Pollefeys. Geometric change detection in urban environments using images. *IEEE transactions on pattern analysis*

and machine intelligence, 37(11):2193–2206, 2015.

[48] A. Torii, M. Havlena, and T. Pajdla. From google street view to 3d city models. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pp. 2188–2195. IEEE, 2009.

[49] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[50] J. Xiao, T. Fang, P. Zhao, M. Lhuillier, and L. Quan. Image-based street-side city modeling. In *ACM SIGGRAPH Asia 2009 papers*, pp. 1–12. 2009.

[51] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6721–6729, 2017.

[52] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4471–4480, 2019.

[53] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *European conference on computer vision*, pp. 286–301. Springer, 2016.

[54] K. Zibrek, S. Martin, and R. McDonnell. Is photorealism important for perception of expressive virtual humans in virtual reality? *ACM Transactions on Applied Perception (TAP)*, 16(3):1–19, 2019.