Project Geollery.com: Reconstructing A Live Mirrored World With Geotagged Social Media

Conference Paper · July 2019

DOI: 10.1145/3329714.3338126

3 authors, including:

Ruofei Du
Google Inc.
34 PUBLICATIONS **171** CITATIONS
SEE PROFILE

David Li
University of Maryland, College Park
7 PUBLICATIONS **22** CITATIONS
SEE PROFILE

Project Geollery.com: Reconstructing A Live Mirrored World With Geotagged Social Media

Ruofei Du*
me@duruofei.com
Department of Computer Science
University of Maryland, College Park

David Li†
dli7319@gmail.com
Department of Computer Science
University of Maryland, College Park

Amitabh Varshney
varshney@cs.umd.edu
Department of Computer Science
University of Maryland, College Park

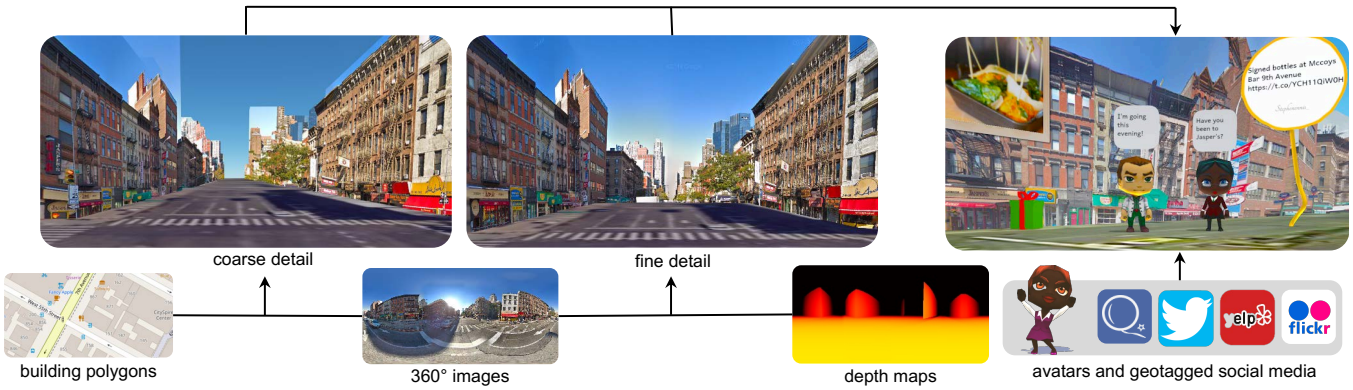


Figure 1: Results and overview of our interactive reconstruction pipeline. Our system is available at https://geollery.com.

ABSTRACT

Social media in virtual reality is in a high-growth market segment with influential products and services in virtual tourism, remote education, and business meetings. Nevertheless, previous systems have never achieved an online platform which renders a 6DoF mirrored world with geotagged social media in real time. In this paper, we introduce the technical detail behind Geollery.com which reconstructs a mirrored world at two levels of detail. Given a pair of latitude and longitude coordinates, our pipeline streams and caches depth maps, street view panoramas, and building polygons from Google Maps and OpenStreetMap APIs. At a fine level of detail for close-up views, we render textured meshes using adjacent local street views and depth maps. When viewed from afar, we apply projection mappings to 3D geometries extruded from building polygons for a coarse level of detail. In contrast to teleportation, our system allows users to virtually walk through the mirrored world at the street level. Our system integrates geotagged social media from both internal users and external sources such as Twitter, Yelp, and Flicker. We validate our real-time strategies of Geollery.com on various platforms including mobile phones, workstations, and head-mounted displays.

CCS CONCEPTS

• **Computing methodologies** → **Texturing**; **Image-based rendering**; *Mixed / augmented reality*; *Virtual reality*.

KEYWORDS

virtual reality, mixed reality, 360° image, GIS, 3D reconstruction, projection mapping, mirrored world, social media, WebGL

ACM Reference Format:
Ruofei Du, David Li, and Amitabh Varshney. 2019. Project Geollery.com: Reconstructing A Live Mirrored World With Geotagged Social Media. In *Web3D '19: The 24th International Conference on 3D Web Technology (Web3D '19), July 26–28, 2019, Los Angeles, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3329714.3338126

1 INTRODUCTION

Since the debut of Social Street View[1] [25] on Web3D 2016, several 3D social media platforms or prototypes have emerged. For example, High Fidelity[2] (early 2016), Facebook Spaces[3] (late 2016), and VR Chat[4] (early 2017) allow people to communicate in virtual avatars like massively multiplayer games (MMOs); Social Street View [25] presents a 3D social media platform with discrete panoramas, thus preventing users from virtually walking on the street;

*The work was fully conducted at University of Maryland, College Park while the first author is current working at Google, San Francisco.
†The first two authors contributed equally to this paper.

Web3D '19, July 26–28, 2019, Los Angeles, CA, USA

ACM ISBN 978-1-4503-6798-1/19/07…$15.00
https://doi.org/10.1145/3329714.3338126

[1]Social Street View: http://socialstreetview.com
[2]High Fidelity: https://www.highfidelity.com
[3]Facebook Spaces: https://facebook.com/spaces
[4]VR Chat: https://vrchat.net

VirtualOulu [2] showcased at CSCW 2017 presents a design prototype which allows virtual avatars to walk in an offline city model, yet is not available throughout the world in real time. To the best of our knowledge, none of the existing systems allows the users to virtually "walk" and explore social media on an arbitrary street of a mirrored world that establishes the correspondence between the physical world and the virtual environment. This led us to ask: *what if we could create an interactive mirrored world on the web, which allowed any one to virtually fly to a remote city, to share space-time stories with remote friends, and to collaborate with others on geotagged social media content?*

To answer these questions, we build Geollery.com, a mixed-reality social media platform which reconstructs a live mirrored world with geotagged social media. 3D models of the physical world are widely used in a diverse set of applications including virtual tourism [25, 56], geographical education [47], neighborhood auditing [10], and urban planning [3, 51]. However, interactive reconstruction of a mirrored world remains a significant challenge.

On the one hand, commercial products such as Google Earth[5], offer world-scale, textured meshes at the aerial level, but the texture quality downgrades significantly for close-up views. Moreover, it does not allow users to freely walk in the virtual environments due to occlusion from the satellite imagery and the 3D models are not publicly available. On the other hand, classic high-fidelity approaches to modeling the 3D world have concentrated on generating 3D meshes using raw input data. For example, structure-from-motion (SfM) pipelines [53, 59, 62] use hundreds or thousands of images in their 3D reconstruction systems to generate dense 3D meshes or point clouds. Despite the effectiveness of these offline systems, their data requirements and processing requirements make them unsuitable for mobile and web applications with processing and bandwidth constraints. Web applications are generally limited to a few megabytes which prevent them from downloading very dense 3D meshes or the data necessary to generate them. Visualizing dense point clouds on low-powered devices may require expensive server-side rendering techniques such as [13]. Furthermore, generating 3D meshes or dense point clouds at a large scale requires data unavailable to most developers.

While offline reconstruction has been well studied and can yield very high-quality results for reconstructing small scenes, it is infeasible for applications with limited resources requiring world-scale virtual environments. Applications requiring an accurate representation of the physical world have thus far been limited to 2D maps, panoramic images [4, 25], and handmade 3D models [2]. Since creating 3D models is very labor-intensive and requires constant updating, 2D maps and panoramas are the primary sources of data available to developers looking for a large virtual representation of the physical world.

Google Street View[6] [4] has led the effort in creating client-side reconstruction applications by generating small depth maps for each of their street view images with most noise, pedestrians, and small vehicles filtered out. These depth maps, compressed, are less than 10 kilobytes each, making them suitable for web applications. While depth information has been incorporated into Google Street View, they have only been used for positioning cursors and distorting street view images to make transitions. Client-side reconstruction by fusing multiple panoramic depth maps of street view images has yet to be explored.

In a previous paper [23], we present the design process, user study, and discussion of human factors in the first version of Geollery. In this paper, we detail the technical aspects of the second version, the online deployment, and the onsite demonstration of Geollery.com at CHI 2019 [22]. Specifically, we introduce an interactive pipeline of fusing 360° images for a mirrored world at two levels of detail (Figure 1) [24]. At a fine level of detail for close-up views, we incorporate multiple Google Street View panoramas and depth data to reconstruct textured meshes directly on the GPU. At a coarse level of detail when viewed from afar, we create extruded boxes with the building metadata from OpenStreetMap[7] and texture the meshes with street view panoramas. We contribute a web-based architecture to stream, cache, reconstruct, and render the mirrored world in real time. Our system is available at https://geollery.com.

2 BACKGROUND AND RELATED WORK

Our work builds upon the prior art in large-scale 3D reconstruction from depth cameras and 360° images.

2.1 3D Reconstruction from Depth Cameras

Traditionally 3D reconstruction has focused on structure-from-motion techniques [39, 57] to reconstruct small scenes from multiview photographs [29, 46, 55, 56, 63] or RGB video streams [6, 7, 11, 60, 64]. Recently, large-scale reconstruction spanning entire rooms or buildings has become possible due to the low-cost depth cameras such as Microsoft Kinect[8] [34] and Microsoft HoloLens[9] [14]. Furthermore, outdoor reconstruction using Project Tango[10] tablets have been shown to run at near interactive frame rates given continuous updates from a depth camera [36, 54]. Thanks to the recent advances of GPU technologies, real-time reconstruction from multiview stereo cameras has also become possible [15, 31, 37, 44]. While these real-time approaches to reconstruction from depth camera video have shown impressive results, they require a continuous source of depth information provided by depth cameras. Our approach uses existing, moderately sparse, depth maps and 360° images to reconstruct large scenes. Following previous techniques in view-dependent rendering, we blend images for projective texture mapping [12]. However, unlike previous systems, our pipeline focuses on client-side reconstruction using pre-processed depth images as opposed to raw depth video streams making our system feasible for real-time 3D web applications.

2.2 Reconstruction from Google Street Views

Since large-scale reconstruction using individual cameras has previously been unrealistic, most city-level reconstruction approaches have focused on using satellite and aerial imagery as their data source [42]. While researchers have also attempted large-scale

[5] Google Earth: https://www.google.com/earth
[6] Google Street View: https://www.google.com/streetview
[7] OpenStreetMap: https://openstreetmap.org
[8] Kinect: https://developer.microsoft.com/windows/kinect
[9] HoloLens: https://www.microsoft.com/en-us/hololens
[10] Project Tango: https://en.wikipedia.org/wiki/Tango

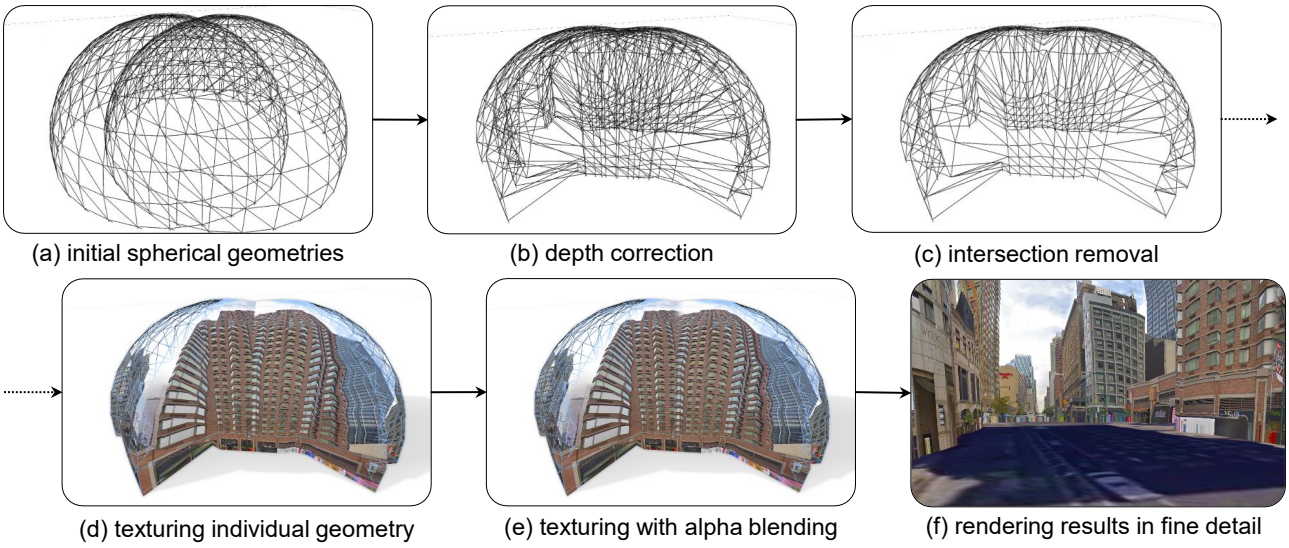


Figure 2: Our rendering pipeline for fusing 360° images in fine detail. (a) We start by generating spherical geometries to represent the 360° images. The segments of the sphere are greatly reduced for visualizing the geometries clearly. (b) In the vertex shader, we correct the depth value for each vertex by sampling the depth maps with its spherical coordinates. (c) In the fragment shader, we discard the pixels in the intersection of the adjacent geometries. (d) Texturing individual spheres with the corresponding 360° images may leave a visible seam. (e) Texturing the spheres with a weighted average according to the position to the camera yields a smoother result. (f) Finally, we interactively texture the ground plane with the corresponding satellite images, apply Gaussian filters in occluded regions, and allow the users to freely walk along the street while streaming the next street views on the go.

reconstruction from unstructured photo collections [1], these approaches require massive amounts of data and computational power, limiting their feasibility for web-based applications.

With the wide availability of 360° images, Google Street View has become the most promising data source for high-resolution large-scale outdoor reconstruction. Torii *et al.*[59] pioneered work in city-level reconstruction by using 4,799 images to reconstruct Pittsburgh using a Structure-from-Motion (SfM) pipeline. The drawback of using SfM pipelines to reconstruct large city spaces are their requirement of multiple images to detect motion and large amounts of processing power to generate the resulting point clouds.

An alternative approach to reconstructing cities purely from Google Street View is constructing building geometries from existing 2D map information and using Google Street View to texture them. For instance, Bulbul and Dahyot [5] use 2D building information from OpenStreetMap alongside low resolution Google Street View images (640×640 px) to generate their 3D cities. However, by fusing multiview street view images and re-calibrating numerous cameras, their approach is limited to offline applications and low resolution. The benefit of using 2D map information is their wide availability and ease of use: 2D building information can be obtained from government records and satellite images. Nevertheless, 2D map information is unable to capture complex geometries such as the Eiffel tower.

Prior work has targeted offline reconstruction using a moderate amount of low-quality noisy data, such as raw depth maps. Recent research has focused on live reconstructions using a large amount of noisy data such as LIDAR [61] and multiview depth cameras [15, 16]. In this paper, we present two real-time reconstruction approaches requiring only a small number of high-quality 360° images and metadata from Google Street View and OpenStreetMap. By using interactive reconstruction methods with low-bandwidth requirements, our system is able to access arbitrary locations of a mirrored world wherever the data is available in OpenStreetMap and Google Street View without the need for any server-side preprocessing or continuous updating of the data.

3 FUSING 360° IMAGES IN FINE DETAIL

In this section, we describe our real-time approach to fusing multiple 360° images, along with their associated depth maps at a fine level of detail. Our rendering pipeline is illustrated in Figure 2.

3.1 Preprocessing

Given a pair of longitude and latitude coordinates, our system sources street view images and depth information from Google Street View in JavaScript. The raw images are fetched as 512×512 px JPEG tiles, which could be decoded and stitched together into five different levels of detail as listed in Table 1. For mobile platforms with lower bandwidth and smaller screens, our system fetches level 2 street view images by default while for workstations, our system fetches level 3 or 4 to achieve a higher quality with minimum latency. We offer the users options to select custom levels of detail.

To demonstrate our interactive approach, we download, decode, and stitch together nearby street view images in a background

Table 1: Resolution, tile counts and file size of Google Street View (GSV) 360° images

level	pixels	resolution	number of tiles	file size
5	134.2M	16384×8192	32×16	$\sim 5M$
4	33.5M	8192×4096	16×8	$\sim 2M$
3	8.4M	4096×2048	8×4	$\sim 800K$
2	2.1M	2048×1024	4×2	$\sim 300K$
1	0.5M	1024×512	2×1	$\sim 90K$

thread in real time, leaving the main thread available for the rendering pipeline. Similarly, as depth information provided by Google is base64-encoded [35] and zlib-compressed[11], we also download and decode the neighboring depth maps in a separate background thread using JavaScript Web Workers.

3.2 Creating a Single Street View Geometry

To get an initial geometry, we first generate a spherical geometry with a fixed radius (Figure 2a) similar to the localized scene generation approach used in [50]. In our prototype, we set the radius R to 1000m, the farthest depth, to prevent the geometry from getting culled when navigating in the scene. The number of width and height segments of the sphere is chosen based on the resolution of our depth map with one additional height segment for the top and bottom vertices of the sphere. By matching the resolution of the depth map with the number of width and height segments of the sphere, each vertex of the sphere geometry, except the top and bottom, corresponds to exactly one pixel of the depth map. For a 512 by 256 depth map, we construct a sphere with 512 width segments and 257 height segments. This sphere would therefore have $512 \times 256 + 2 = 131,074$ vertices and $2 \times 512 \times 256 = 262,144$ faces.

To achieve interactive frame rates, we use a custom vertex shader running on the GPU to determine the exact positions of the sphere vertices to create a convincing geometry. In the vertex shader, we first compute the spherical coordinates of each vertex in the sphere geometry, $(\rho_0, \theta_0, \phi_0)$, by calculating the directional vector $\mathbf{d} = \mathbf{v} - \mathbf{s}$ from the location of the street view camera $\mathbf{s}$ to the vertex $\mathbf{v}$. Here ρ_0 is the initial radius of the sphere while (θ_0, ϕ_0) are the converted coordinates of $\mathbf{d}$ in the $\mathbb{SO}(2)$ space. We then look up the correct depth value ρ_1 by sampling the depth map at (θ_0, ϕ_0). Finally, we move each vertex to the correct spherical position $(\rho_1, \theta_0, \phi_0)$ by setting its Cartesian position in world coordinates to $\left(\frac{\rho_1}{\rho_0}\right)\mathbf{d} + \mathbf{s}$. With a vertex shader running on the GPU, all computation is executed in parallel and in real time.

3.3 Merging Multiple Street View Geometries

With depth maps, we can generate one geometry for each street view image, as shown in Figure 2(b). To encapsulate a wider area than what is possible with a single street view image, we aim to seamlessly fuse the corrected spherical geometries generated along each street.

[11] zlib: https://zlib.net

When positioning the sphere geometries based on the geographic coordinates of their associated street views, we get large intersections between adjacent geometries. Ideally, from any point within each sphere, we would like to see the farthest street view sphere. Hence, we decide to discard the intersection between adjacent geometries. To implement this technique, we compute whether each pixel of each geometry intersects with another geometry at runtime. With the metadata from Google Street View, we query and pass in the locations for adjacent street view images to the fragment shader. For each pixel located at $\mathbf{p}$, we compute the distance $d_0 = |\mathbf{p} - \mathbf{v}_0|$ from the current street view $\mathbf{v}_0$ to $\mathbf{p}$ and the distance $d_1 = |\mathbf{p} - \mathbf{v}_1|$ from adjacent street views $\mathbf{v}_1$ to $\mathbf{p}$. If the adjacent street views are closer in distance to the current pixel than the current street view, *i.e.,* $d_1 < d_0$, we discard the current pixel. As shown in Figure 2, this discards the interior of the intersections of the two geometries while preserving the remainder.

While fusing the geometries by discarding the intersections yields promising results, the imperfect correspondence between the depth maps of adjacent street views creates visible gaps at the seams where two spheres intersect as shown in Figure 3(a). To eliminate these gaps, we translate each vertex $\mathbf{v_0}$ near the gaps by a vector $\delta_0 = \hat{\mathbf{v}}_\mathbf{0} - \mathbf{v_0}$ from the initial vertex position $\mathbf{v_0}$ to a position $\hat{\mathbf{v}}_\mathbf{0}$ suggested by the other depth map, scaled by a factor of 0.7. An example is shown in Figure 3(b). Note that while this method eliminates many gaps caused by erroneous depth map in real time, it may not work for very large gaps and does not fuse the texture information which we discuss next.

Figure 3: Results before and after gap alignment for translating the vertices near the seams. (a) When rendering multiple street view geometries, a gap (highlighted in the red box) may appear at the seams where two geometries intersect. (b) By aligning the vertices around the seams, we mostly eliminate the gap.

3.4 Texturing the Geometries

We first texture the individual geometry by sampling from the corresponding 360° image with the spherical coordinates at each pixel. While this gives a perfect view from the center of the street view, we notice several issues including misalignment between adjacent geometries, distorted projection on the ground, and pixelation in occluded regions. Next, we present our efforts to mitigate these artifacts in real time.

3.4.1 Weighted Alpha Blending for Smoothing the Seams. Street view images usually do not line up at the edges where the geometries meet. This is due to both poor calibrations of the camera

positions and inaccurate depth estimation. An optimal solution may use optical flow over the mesh [9] and Poisson blending [48]. Nevertheless, these approaches are usually not available for real-time systems.

(a) without seam blending (b) with seam blending

Figure 4: Results before and after applying the weighted alpha blending near the seams. (a) Without blending, the seam (highlighted in the center of the red box) at the edges is noticeable due to the texturing misalignment. (b) Weighted alpha blending in the fragment shader makes the seam much less distinguishable.

In our rendering pipeline, we blend the textures of the adjacent two street view geometries at the seam with weighted alpha blending inspired by the Montage4D system [19]. For each pixel, we first compute its distance to each of the two adjacent street views, at distances d_0 and d_1. Note that a better but computationally expensive approach will involve computing the geodesics from each vertex to the seam, as explained in [20]. We next sample the textures from the two 360° images as $\mathbf{c_0}$ and $\mathbf{c_1}$. Finally, we color the pixels near the seams with the weighted average of $\left(0.5 + \frac{d_1 - d_0}{2\delta}\right)\mathbf{c_0} + \left(0.5 - \frac{d_1 - d_0}{2\delta}\right)\mathbf{c_1}$. As shown in Figure 4, by blending the pixels near the gap where the two geometries meet, the seams become much less distinguishable.

3.4.2 Texturing with Satellite Images. In dense urban areas, undesirable cars and pedestrians often appear in street view images. In most cases, these pedestrians get projected to the ground when incorporating Google's depth maps. While this projection is appropriate when viewed from the position the original street view was taken, it leads to distortions when viewed from another perspective. To eliminate the distorted cars and pedestrians, we choose to overlay Google Maps' satellite images instead. As shown in Figure 5, texturing the ground plane with satellite images results in a better visual appearance for our system. We acknowledge that this method is limited to the availability and lower resolution of the external satellite images and we envision further systems may take advantage of image in-painting [28, 43] or deep learning [65] to eliminate the artifacts on the ground.

3.4.3 Applying Gaussian Filters. Pixelation of street view images on the geometry occurs when portions of the real geometry are occluded from the street view image.

Stretching, distortion, and pixelation may occur when portions of the real geometry are occluded from the 360° textures. For example, as presented in Figure 6, when the camera is positioned in the center of two street views taken at a large distance apart, the sides of the building are not observed by either of the 360° images. Hence,

(a) texturing with street view images (b) texturing with satellite images

Figure 5: Results before and after texturing with satellite images for the ground plane. Note that the satellite textures eliminate the distorted vehicles and pedestrians projected onto the road.

the large occluded areas sample from very few pixels, resulting in undesirable artifacts.

(a) without Gaussian filtering (b) with Gaussian filtering

Figure 6: Results before and after applying a Gaussian filter in the occluded areas. The left image suffers from pixelation caused by occlusion and distortion, while the right figure shows a smoother sampling result.

To detect the occluded region, we first sample the depth map in the vertex shader four additional times to sample the depth of the adjacent vertices. The artifacts occur primarily when the depth drops off at the edge of the building. Therefore, if the depth of neighboring vertices differs drastically from the current vertex, we mark the current vertex as occluded. Finally, we apply a 5×5 Gaussian filter with a standard deviation of 7.0 in the fragment shader on the GPU[12] to smooth the occluded pixels.

4 FUSING 360° IMAGES AT A COARSE LEVEL OF DETAIL

Whereas the former rendering pipeline offers fine detail for close-up views, a bird's-eye view requires a different real-time approach. Once the camera dollies outside of the geometries used in close-up views, the outer face of the sphere-based geometries becomes visible as presented in Figure 2(e). Hence, fusing 360° images at a larger scale encompassing multiple streets requires larger-scale geometries. For reconstructing a bird's-eye view in real time, we create building geometries based on 2D map data rather than localized depth maps from each street view. We further project street view images onto the building geometries in a fragment shader running on the GPU.

[12]Code of Gaussian filter: https://shadertoy.com/view/ltBXRh

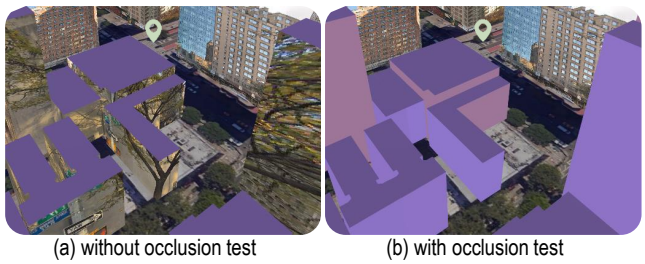
4.1 Building Geometries

Unlike the previous approach, we choose not to use the depth maps to generate building geometries since they are unable to capture all faces of a building. Instead, we source data from OpenStreetMap using the Overpass API[13] to obtain 2D polygons for buildings. While these polygons are not as widely available as street view images, we find that in urban areas such as New York City, 2D building polygons often come with useful metadata such as the height in meters or the number of floors for each building. To convert these 2D polygons into 3D, we extrude them to the correct height based on the information provided in the metadata. For instances where metadata is not available, we extrude them to a predefined height of 16 meters to represent a 4-story building.

4.2 Projecting Street Views

While previous approaches [5] to generating 3D textured building geometries have achieved impressive results incorporating multiple street view images, their offline systems typically require significant preprocessing time and use hundreds of low-resolution street view images. Instead, we texture 3D geometries in real-time by using a single street view image to maintain real-time performance, preserve high-quality textures, and minimize image artifacts.

We project the street view onto building walls by sampling the nearest street view image from a fragment shader. In the fragment shader, we calculate the directional vector $\mathbf{d} = \mathbf{p} - \mathbf{s}$ from the position of the street view $\mathbf{s}$ to each pixel $\mathbf{p}$. Then we calculate the direction of the vector in spherical coordinates $(\rho_0, \theta_0, \phi_0)$ using the transformation $(\rho_0, \theta_0, \phi_0) = (|\mathbf{d}|, \arcsin(\frac{d.y}{|\mathbf{d}|}), \arctan 2(d.z, d.x))$. Finally, we sample the (θ_0, ϕ_0) point of the street view image to color each pixel of the building.



(a) without occlusion test (b) with occlusion test

Figure 7: Results before and after the occlusion test with a single 360° image (for illustration purpose) and multiple buildings. The light green icon marks approximately where the street view image is taken from. (a) When viewed from afar, projecting street view images onto every side of every building results in occluded areas being textured with inappropriate projections. (b) Therefore, we perform occlusion test in the fragment shader and only project 360° images onto the visible surfaces. The occluded surfaces, such as rooftops and occluded walled, are textured with solid colors.

To avoid texturing sides of buildings which are occluded from the street view, we use the fragment shader to detect whether individual faces of the geometry are occluded in the street view images, as illustrated in Figure 7. To accomplish this, we compare the normal vector $\mathbf{n}$ with the previous vector pointing to the building face $\mathbf{d}$. If $\langle \mathbf{n}, -\mathbf{d} \rangle < 0$, or equivalently $\langle \mathbf{n}, \mathbf{d} \rangle > 0$, we conclude that the angle between $\mathbf{n}$ and $-\mathbf{d}$ is greater than 90 degrees so the current face is occluded in the street view images. Instead of using the street view images, we texture occluded faces with a solid color.

5 EVALUATION AND DEPLOYMENT

Our rendering pipeline is implemented in WebGL and `Three.js`[14], a cross-browser and GPU-accelerated Javascript library. We evaluate the performance of our system in various platforms and present live demos in Geollery[15].

5.1 Performance on Different Platforms

To evaluate the performance of our real-time approach, we test our approach on mobile phones, workstations, and head-mounted displays. On a workstation equipped with an NVIDIA GTX 1080 GPU, we observed that it takes about 12ms to ship each 4096×2048 resolution street view image to the GPU after being decoded and stitched together in a background thread. For higher resolution 360° images, such as 8192×4096, it takes 40ms to ship the texture from system memory to the GPU. After the geometries are initialized and all textures are cached on the GPU, we achieve rendering at a fine level of detail at over 60 frames per second (fps). Furthermore, our rendering pipeline takes merely 2.0 ms, leaving 14.6 ms of GPU time to render additional objects.

In VR, our workstation is able to render at 90 fps to an Oculus Rift. On an Android phone with a Snapdragon 835, we observed that it takes about 100ms to ship each 4096×2048 resolution 360° image to the GPU. Afterwards, we achieve an average of 30 fps rendering close-up views. On an iPhone XS with an A12 bionic chip, the frame rate is around 40 fps. At a coarse level of detail when viewed from afar, the rendering performance becomes dependent on the number of buildings within view and the complexity of the buildings. In our testing, we are able maintain a smooth 60 fps on our workstations rendering to monitors and 90 fps rendering to an Oculus Rift with about 50 buildings visible in New York City .

5.2 Deployment and Implementation Details

To demonstrate an application of our method, we have incorporated our approach into a novel mixed-reality social platform, Geollery, as shown in Figure 8. Geollery is hosted on Amazon Web Services (AWS) and allows multiple users to see, chat, and collaborate with each other with virtual avatars. While this paper focuses on the technical aspect, we refer the readers to [23] for our design process and user study for the social platform.

In Geollery, we use a Least Recently Used (LRU) cache[16] to store five most recent 360° street views and depth images. This enables us to quickly fetch the previously accessed data from memory as users walk around, minimizing bandwidth utilization and improving responsiveness. When switching between the fine and coarse detail

[13] Overpass: https://wiki.openstreetmap.org/Overpass_API

[14] Three.js: http://www.threejs.org

[15] Geollery: https://geollery.com

[16] JS-LRU: https://github.com/rsms/js-lru

Figure 8: Real-time rendering results from Geollery, where the National Geollery of Art is reconstructed with geotagged social media when viewed from afar.

(use scrolling on workstation or pinch gestures on mobile platforms), our system apply an animated alpha blending to smoothly fade in the new geometries.

By fusing multiple street view images, we are able to cover a larger area resulting in fewer updates for loading new panoramas. Nevertheless, there is a trade-off between performance and creating many fine-detail geometries as each geometry has 131,074 vertices to be processed by the GPU. In the future, we intend to evaluate the possibility of fusing more than two street view geometries together. Fusing five or more street view images would create a larger environment making our approach more immersive for interactive applications. Additionally, only the farthest street view would need to be updated as the camera moves within the environment, making the loading of additional geometries less noticeable to users.

Since the debut of Geollery.com, we have attracted 733 individual users across 25 countries so far. During the onsite demonstration in Glasgow, United Kingdom, at CHI 2019 [22], Geollery was also presented to over 3,700 attendees. In addition to the study we conducted and reported with the first version [22, 23], we have received a few emails with feedback from the online community:

> *"Geollery/Social Street View has its own set of distinct offerings, as it is anchored within real-world settings, just mapped onto VR, whereas these are definitely more 'fantasy' type of arenas. In that way, as you have already done, I think there are multitude game challenges/tasks/feedback, like the balloons, to add in!"*

> *"I think it'd be cool if you could see posts by people in real time, along with the establishment they're in (like someone tweeting from inside McDonald's or a movie theater), if that makes sense. Sort of like checking in to a place on Facebook."*

5.3 Limitations

While we have demonstrated a real-time pipeline for fusing multiple 360° images, our system is not without limitations as discussed next.

5.3.1 Inaccurate Metadata. Both of our approaches rely on clean and high-quality metadata, including the raw depth map, building polygons, and their heights.

At a fine level of detail for close-up views, we use Google Street View images and their depth maps. Google Street View's depth maps are filtered to remove pedestrians and vehicles but do not manage to filter out everything. Sometimes, obstacles such as construction zones and parked trucks cause distortions in the geometries we generate from the depth map as seen in Figure 9. These vehicles cause the resulting geometry to have undesirable indents into the road, representing the obstacle rather than the buildings behind it.

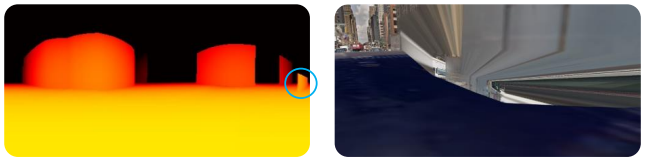

(a) inaccurate raw depth map (b) resulting occlusion

Figure 9: (a) shows an inaccurate depth map from Google Street View. The rightmost polygon within the blue circle is a typical example of noise caused by obstacles such as large vehicles. (b) shows how the large vehicle blocks the depth sensors used by Google Street View. Vertices are pulled into the middle of the road to represent the truck instead of the buildings behind it. The depth map is visualized in a yellow-red-black color scheme, with code provided at https://shadertoy.com/view/llVBDD.

When viewed from afar, we use OpenStreetMap to construct building geometries. However, OpenStreetMap occasionally reports inaccurate or missing values for building heights as their data is crowd-sourced.

5.3.2 Level-of-Detail of Geometries. For our rendering pipeline in fine detail, each geometry results in 131,074 vertices regardless of the complexity of the ground truth. Reducing the number of vertices in the geometry would greatly improve the performance of our approach. As of May 2019, compute shaders in WebGL run merely on Google Chrome or Microsoft Edge Insider launched with debugging flags. Besides, altering the number of vertices on the CPU may result in a severe performance penalty. We envision future systems may use a view-dependent strategy and determine the level of tessellation with the geometry shader on the GPU. Foveated rendering [30, 41] may also be adopted to accelerate the frame rate on low-end mobile phones or laptops.

6 CONCLUSION AND FUTURE VISION

In this paper, we have presented the technical pipeline behind Geollery.com, including two real-time methods of reconstructing a live mirrored world with geotagged social media. We discuss challenges with each method and our approaches to addressing them. Finally, we evaluate the performance and quality of each approach on various platforms.

At a fine level of detail for close-up views, we reconstruct an approximate geometry based on the depth maps associated with

each street view and propose ways of seamlessly aligning the adjacent street view geometries. This approach takes advantage of the high resolution of the street view images while incorporating low-resolution depth maps to generate an approximate geometry. At a coarse level of detail when viewed from afar, we propose the texturing of 3D geometries created from extruded 2D polygons. Neither method requires any server-side preprocessing and all client-side processing can be done in background threads for interactive applications.

Figure 10: An experimental feature in Geollery.com allows special users to place customized buildings to replace the automatic reconstruction from the live streamed data. With crowdsourcing from domain experts, a digital campus with geotagged social media may recur in the mirrored world.

In future, one may evaluate additional methods for better addressing the limitations of our approach. Fast upsampling of the depth map paired with noise removal using techniques from [38] and [45] may result in smoother geometries that are better aligned with buildings. Removing pedestrians and vehicles from street view images using techniques similar to Flores and Belongie [28] may eliminate the need to overlay lower resolution Google Maps satellite images. Augmenting our outdoor reconstruction with indoor buildings [27] will further increase the potential applications of our approach. As the quality and quantity of publicly available data improves, we hope our methods will enable large-scale applications to take advantage of just-in-time high-resolution 3D reconstructions.

With the rapid advances in virtual and augmented reality, we envision a system that fuses a variety of multimedia data to create a vivid mirrored world [17], in which it fuses not only 360° images and geotagged social media [26], but also 3D reconstruction of the user [58], mid-air sketches [49], data visualization of the streaming events, and elaborately reconstructed museums and university campuses [52](Figure 10). From the perspective of time, it fuses the past events (geotagged images, text, videos, and holographic memories) with the present (live surveillance videos [18], crowd-sourced 360° cameras, the user's live telepresence [40, 44], and haptic feedback [8, 21, 32, 33]), and look into the future by learning the trajectory of pedestrians and cars, clustering the topics and emotions of social media, and measuring the spatiotemporal saliency of the real-time information in the world. Such a system may eventually change the way we communicate with each other and consume the information throughout the world.

ACKNOWLEDGEMENT

We would like to thank *Sai Yuan, Akanksha Shrivastava, Xiaoxu Meng, Shuo Li, Eric Krokos, Xuetong Sun*, all user study and CHI 2019 participants, and the anonymous reviewers for the insightful feedback on the Geollery system and this manuscript.

This work has been supported in part by the NSF Grants 1823321, 1564212, 1429404, and the State of Maryland's MPower initiative. Any opinions, findings, conclusions, or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the research sponsors.

REFERENCES

[1] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Sietz, and Rick Szeliski. 2009. Building Rome in a Day. In *2009 IEEE 12th International Conference on Computer Vision (CVPR)*. IEEE, 72–79.
[2] Toni Alatalo, Timo Koskela, Matti Pouke, Paula Alavesa, and Timo Ojala. 2016. VirtualOulu: Collaborative, Immersive and Extensible 3D City Model on the Web. In *Proceedings of the 21st International Conference on Web3D Technology (Web3D)*. ACM, 95–103.
[3] Toni Alatalo, Matti Pouke, Timo Koskela, Tomi Hurskainen, Ciprian Florea, and Timo Ojala. 2017. Two real-world case studies on 3D web applications for participatory urban planning. In *Proceedings of the 22nd International Conference on 3D Web Technology*. ACM, 11.
[4] Dragomir Anguelov, Carole Dulong, Daniel Filip, Christian Frueh, Stéphane Lafon, Richard Lyon, Abhijit Ogale, Luc Vincent, and Josh Weaver. 2010. Google Street View: Capturing the World at Street Level. *Computer* 43, 6 (2010), 32–38.
[5] Abdullah Bulbul and Rozenn Dahyot. 2017. Social Media Based 3D Visual Popularity. *Computers & Graphics* 63 (2017), 28–36.
[6] Cedric Cagniart, Edmond Boyer, and Slobodan Ilic. 2010. Probabilistic Deformable Surface Tracking From Multiple Videos. In *ECCV'10 Proceedings of the 11th European Conference on Computer Vision: Part IV*. Springer, 326–339.
[7] Dan Casas, Margara Tejera, Jean-Yves Guillemaut, and Adrian Hilton. 2013. Interactive Animation of 4D Performance Capture. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 19, 5 (2013), 762–773.
[8] Inrak Choi, Eyal Ofek, Hrvoje Benko, Mike Sinclair, and Christian Holz. 2018. Claw: A multifunctional handheld haptic controller for grasping, touching, and triggering in virtual reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 654.
[9] Ming Chuang, Linjie Luo, Benedict J Brown, Szymon Rusinkiewicz, and Michael Kazhdan. 2009. Estimating the Laplace-Beltrami Operator by Restricting 3D Functions. *Computer Graphics Forum* 28, 5 (2009), 1475–1484.
[10] Philippa Clarke, Jennifer Ailshire, Robert Melendez, Michael Bader, and Jeffrey Morenoff. 2010. Using Google Earth to Conduct a Neighborhood Audit: Reliability of a Virtual Audit Instrument. *Health & Place* 16, 6 (2010), 1224–1229.
[11] Edilson De Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. 2008. Performance Capture From Sparse Multi-View Video. *ACM Transactions on Graphics (TOG)* 27, 3 (2008), 98.
[12] Paul Debevec, Yizhou Yu, and George Borshukov. 1998. Efficient View-dependent Image-based Rendering with Projective Texture-mapping. In *Rendering Techniques 98*. Springer, 105–116.
[13] Sören Discher, Rico Richter, and Jürgen Döllner. 2018. A Scalable WebGL-based Approach for Visualizing Massive 3D Point Clouds Using Semantics-dependent Rendering Techniques. In *Proceedings of the 23rd International ACM Conference on 3D Web Technology (Web3D '18)*. ACM, New York, NY, USA, Article 19, 9 pages.
[14] Samuel Dong and Tobias Höllerer. 2018. Real-Time Re-Textured Geometry Modeling Using Microsoft HoloLens. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 231–237.
[15] Mingsong Dou, Philip Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, and Shahram Izadi. 2017. Motion2fusion: Real-Time Volumetric Performance Capture. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 246.
[16] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich, and Shahram Izadi. 2016. Fusion4D: Real-Time Performance Capture of Challenging Scenes. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 114.
[17] Ruofei Du. 2018. *Fusing Multimedia Data Into Dynamic Virtual Environments*. Ph.D. Dissertation. University of Maryland, College Park.
[18] Ruofei Du, Sujal Bista, and Amitabh Varshney. 2016. Video Fields: Fusing Multiple Surveillance Videos Into a Dynamic Virtual Environment. In *Proceedings of the 21st International Conference on Web3D Technology (Web3D '16)*. ACM, 165–172.
[19] Ruofei Du, Ming Chuang, Wayne Chang, Hugues Hoppe, and Amitabh Varshney. 2018. Montage4D: Interactive Seamless Fusion of Multiview Video Textures. In

Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '18). ACM, 124–133.

[20] Ruofei Du, Ming Chuang, Wayne Chang, Hugues Hoppe, and Amitabh Varshney. 2019. Montage4D: Real-Time Seamless Fusion and Stylization of Multiview Video Textures. *Journal of Computer Graphics Techniques* 1, 15 (17 Jan. 2019), 1–34.

[21] Ruofei Du and Liang He. 2016. VRSurus: Enhancing Interactivity and Tangibility of Puppets in Virtual Reality. In *CHI EA '16 Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '16)*. ACM, 2454–2461.

[22] Ruofei Du, David Li, and Amitabh Varshney. 2019. Experiencing a Mirrored World With Geotagged Social Media in Geollery. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems (CHI EA)*. INT040:1–4.

[23] Ruofei Du, David Li, and Amitabh Varshney. 2019. Geollery: a Mixed Reality Social Media Platform. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI)*. ACM, 13.

[24] Ruofei Du, David Li, and Amitabh Varshney. 2019. Interactive Fusion of 360° Images for a Mirrored World. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR '19)*. 2.

[25] Ruofei Du and Amitabh Varshney. 2016. Social Street View: Blending Immersive Street Views With Geo-Tagged Social Media. In *Proceedings of the 21st International Conference on Web3D Technology (Web3D '16)*. ACM, 77–85.

[26] Ruofei Du and Amitabh Varshney. 2016. Systems, Devices, and Methods for Generating a Social Street View. US Patent App. 15/559,955.

[27] Jakub Dziekoński and Krzysztof Walczak. 2018. A Configurable Virtual Reality Store. In *Proceedings of the 23rd International ACM Conference on 3D Web Technology (Web3D '18)*. ACM, New York, NY, USA, Article 29, 2 pages.

[28] Arturo Flores and Serge Belongie. 2010. Removing Pedestrians From Google Street View Images. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*. IEEE, 53–58.

[29] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. 2009. Reconstructing Building Interiors from Images. In *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 80–87.

[30] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. 2012. Foveated 3D Graphics. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 164.

[31] Marc Habermann, Weipeng Xu, Michael Zollhöfer, Gerard Pons-Moll, and Christian Theobalt. 2019. LiveCap: Real-Time Human Performance Capture From Monocular Video. *ACM Transactions on Graphics (TOG)* 38, 2 (2019), 14.

[32] Zhenyi He, Fengyuan Zhu, and Ken Perlin. 2017. Physhare: Sharing Physical Interaction In Virtual Reality. In *Adjunct Publication of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM, 17–19.

[33] Zhenyi He, Fengyuan Zhu, Ken Perlin, and Xiaojuan Ma. 2018. Manifest the Invisible: Design for Situational Awareness of Physical Environments in Virtual Reality. *arXiv preprint arXiv:1809.05837* (2018).

[34] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. 2011. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. ACM, 559–568.

[35] Simon Josefsson. 2006. *The Base16, Base32, and Base64 Data Encodings*. Technical Report. The Internet Society.

[36] Matthew Klingensmith, Ivan Dryanovski, Siddhartha Srinivasa, and Jizhong Xiao. 2015. Chisel: Real Time Large Scale 3D Reconstruction Onboard a Mobile Device Using Spatially Hashed Signed Distance Fields. In *Robotics: Science and Systems*, Vol. 4.

[37] Adarsh Kowdle, Christoph Rhemann, Sean Fanello, Andrea Tagliasacchi, Jonathan Taylor, Philip Davidson, Mingsong Dou, Kaiwen Guo, Cem Keskin, Sameh Khamis, et al. 2018. The Need 4 Speed In Real-time Dense Visual Tracking. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 220.

[38] J. Lei, L. Li, H. Yue, F. Wu, N. Ling, and C. Hou. 2017. Depth Map Super-Resolution Considering View Synthesis Quality. *IEEE Transactions on Image Processing* 26, 4 (April 2017), 1732–1745.

[39] H. C. Longuet-Higgins. 1987. Readings in Computer Vision: Issues, Problems, Principles, and Paradigms. *Nature* 293, 5828 (1987), 61–62.

[40] Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, Adarsh Kowdle, Christoph Rhemann, Dan B Goldman, Cem Keskin, Steve Seitz, Shahram Izadi, and Sean Fanello. 2018. LookinGood: Enhancing Performance Capture with Real-time Neural Re-rendering. *ACM Trans. Graph.* 37, 6, Article 255 (Dec. 2018), 14 pages.

[41] Xiaoxu Meng, Ruofei Du, Matthias Zwicker, and Amitabh Varshney. 2018. Kernel Foveated Rendering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 5 (15-18 May. 2018), 1–20.

[42] Przemyslaw Musialski, Peter Wonka, Daniel G Aliaga, Michael Wimmer, Luc Van Gool, and Werner Purgathofer. 2013. A Survey of Urban Reconstruction. *Computer Graphics Forum* 32, 6 (2013), 146–177.

[43] Angelo Nodari, Marco Vanetti, and Ignazio Gallo. 2012. Digital Privacy: Replacing Pedestrians From Google Street View Images. In *2012 21st International Conference on Pattern Recognition (ICPR)*. IEEE, 2889–2893.

[44] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, Vladimir Tankovich, Charles Loop, Philip A.Chou, Sarah Mennicken, Julien Valentin, Vivek Pradeep, Shenlong Wang, Sing Bing Kang, Pushmeet Kohli, Yuliya Lutchyn, Cem Keskin, and Shahram Izadi. 2016. Holoportation: Virtual 3D Teleportation in Real-Time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST)*. 741–754.

[45] J. Park, H. Kim, Yu-Wing Tai, M. S. Brown, and I. Kweon. 2011. High Quality Depth Map Upsampling for 3D-TOF Cameras. In *2011 International Conference on Computer Vision*. 1623–1630.

[46] Robert Patro, Cheuk Yiu Ip, Sujal Bista, and Amitabh Varshney. 2011. Social Snapshot: a System for Temporally Coupled Social Photography. *IEEE Computer Graphics and Applications* 31, 1 (2011), 74–84.

[47] Todd C Patterson. 2007. Google Earth As a (not Just) Geography Education Tool. *Journal of Geography* 106, 4 (2007), 145–152.

[48] Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson Image Editing. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 313–318.

[49] Ken Perlin, Zhenyi He, and Fengyuan Zhu. 2018. Chalktalk VR/AR. *International SERIES on Information Systems and Management in Creative eMedia (CreMedia)* 2017/2 (2018), 30–31.

[50] Lohit Petikam, Andrew Chalmers, and Taeyhun Rhee. 2018. Visual Perception of Real World Depth Map Resolution for Mixed Reality Rendering. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 401–408.

[51] Nicholas Polys, Cecile Newcomb, Todd Schenk, Thomas Skuzinski, and Donna Dunay. 2018. The Value of 3D Models and Immersive Technology in Planning Urban Density. In *Proceedings of the 23rd International ACM Conference on 3D Web Technology*. ACM, 13.

[52] Matti Pouke, Johanna Ylipulli, Ilya Minyaev, Minna Pakanen, Paula Alavesa, Toni Alatalo, and Timo Ojala. 2018. Virtual Library: Blending Mirror and Fantasy Layers into a VR Interface for a Public Library. In *Proceedings of the 17th International Conference on Mobile and Ubiquitous Multimedia (MUM 2018)*. ACM, New York, NY, USA, 227–231.

[53] Bc Russell and R Martin-Brualla. 2013. 3D Wikipedia: Using Online Text to Automatically Label and Navigate Reconstructed Geometry. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–10.

[54] Thomas Schöps, Torsten Sattler, Christian Häne, and Marc Pollefeys. 2017. Large-Scale Outdoor 3D Reconstruction on a Mobile Device. *Computer Vision and Image Understanding* 157 (2017), 151 – 166. Large-Scale 3D Modeling of Urban Indoor or Outdoor Scenes from Images and Range Scans.

[55] Hyewon Seo, Young In Yeo, and Kwangyun Wohn. 2006. 3D Body Reconstruction From Photos Based on Range Scan. In *Technologies for E-Learning and Digital Entertainment*. Springer Berlin Heidelberg, Berlin, Heidelberg, 849–860.

[56] Noah Snavely, Steven M Seitz, and Richard Szeliski. 2006. Photo Tourism: Exploring Photo Collections in 3D. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 835–846.

[57] Minas E. Spetsakis and John (Yiannis) Aloimonos. 1990. Structure From Motion Using Line Correspondences. *International Journal of Computer Vision* 4, 3 (01 Jun 1990), 171–183.

[58] Danhang Tang, Mingsong Dou, Peter Lincoln, Philip Davidson, Kaiwen Guo, Jonathan Taylor, Sean Fanello, Cem Keskin, Adarsh Kowdle, Sofien Bouaziz, Shahram Izadi, and Andrea Tagliasacchi. 2018. Real-time Compression and Streaming of 4D Performances. *ACM Trans. Graph.* 37, 6, Article 256 (Dec. 2018), 11 pages.

[59] Akihiko Torii, Michal Havlena, and Tomáš Pajdla. 2009. From Google Street View to 3D City Models. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. IEEE, 2188–2195.

[60] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. 2008. Articulated Mesh Animation From Multi-View Silhouettes. *ACM Transactions on Graphics (TOG)* 27, 3 (2008), 97.

[61] Hao Wang, Jun Wang, and Wang Liang. 2016. Online Reconstruction of Indoor Scenes From RGB-D Streams. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3271–3279.

[62] Jianxiong Xiao, Tian Fang, Peng Zhao, Maxime Lhuillier, and Long Quan. 2009. Image-Based Street-Side City Modeling. *ACM Transactions on Graphics (TOG)* 28, 5, Article 114 (2009), 114:1–114:12 pages.

[63] Jianxiong Xiao and Yasutaka Furukawa. 2014. Reconstructing the WorldâĂŹs Museums. *International journal of computer vision* 110, 3 (2014), 243–258.

[64] Feng Xu, Yebin Liu, Carsten Stoll, James Tompkin, Gaurav Bharaj, Qionghai Dai, Hans-Peter Seidel, Jan Kautz, and Christian Theobalt. 2011. Video-Based Characters: Creating New Human Performances From a Multi-View Video Database. *ACM Transactions on Graphics (TOG)* 30, 4 (2011), 32.

[65] Raymond A Yeh, Chen Chen, Teck-Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. 2017. Semantic Image Inpainting With Deep Generative Models. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 4.