Real-time Virtual Object Insertion for Moving 360° Videos

Joanna Tarko
University of Bath
j.k.tarko@bath.ac.uk

James Tompkin
Brown University
james_tompkin@brown.edu

Christian Richardt
University of Bath
christian@richardt.name

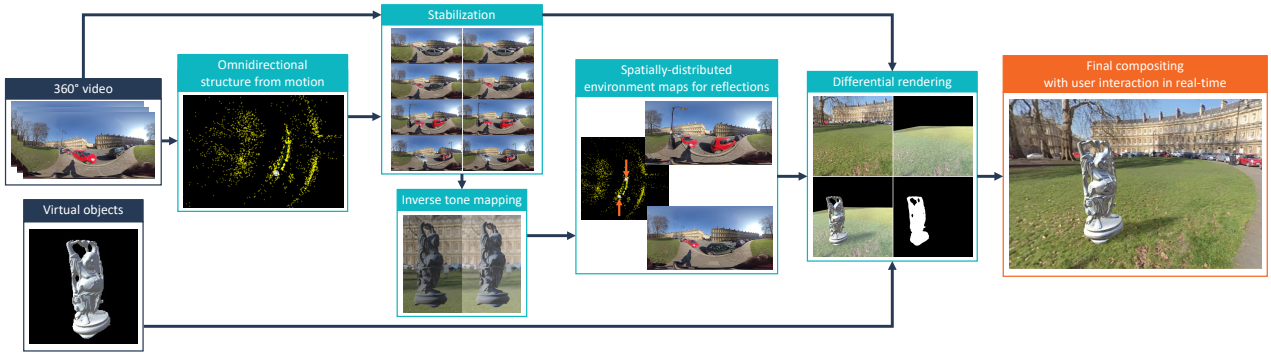

Figure 1: Our approach to insert virtual objects interactively into 360° videos, showing inputs (dark blue), processing steps (turquoise) and our result (orange). We use omnidirectional structure from motion for camera tracking and 360° stabilisation, image-based lighting with inverse tone mapping and spatially-distributed local environment maps, and differential rendering for image-based shadowing. Our Unity-based approach performs final compositing and user interaction in real time.

ABSTRACT

We propose an approach for real-time insertion of virtual objects into pre-recorded moving-camera 360° video. First, we reconstruct camera motion and sparse scene content via structure from motion on stitched equirectangular video. Then, to plausibly reproduce real-world lighting conditions for virtual objects, we use inverse tone mapping to recover high dynamic range environment maps which vary spatially along the camera path. We implement our approach into the Unity rendering engine for real-time virtual object insertion via differential rendering, with dynamic lighting, image-based shadowing, and user interaction. This expands the use and flexibility of 360° video for interactive computer graphics and visual effects applications.

KEYWORDS

Omnidirectional video, mixed reality, structure from motion, 3D reconstruction, lighting estimation, environment maps

ACM Reference Format:
Joanna Tarko, James Tompkin, and Christian Richardt. 2019. Real-time Virtual Object Insertion for Moving 360° Videos. In *The 17th International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI '19), November 14–16, 2019, Brisbane, QLD, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3359997.3365708

VRCAI '19, November 14–16, 2019, Brisbane, QLD, Australia

ACM ISBN 978-1-4503-7002-8/19/11…$15.00
https://doi.org/10.1145/3359997.3365708

1 INTRODUCTION

To photorealistically insert computer-generated (CG) objects into real video footage, we must solve three main technical challenges: (1) track the video so that the virtual and real cameras share the same view, (2) estimate the illumination to plausibly light virtual objects, and (3) render and composite CG objects with the video in real time.

Consumer 360° cameras capture images and video with a 360° horizontal and 180° vertical field of view, and typically store *spherical* or *omnidirectional* [Scaramuzza 2014] images or videos using *equirectangular* projection—a conversion of spherical 'latitude and longitude' coordinates into 2D image pixels (see Section 3.1). 360° cameras provide potential advantages for inserting virtual objects into interactive applications. Unlike for perspective video, users can freely look around to explore an environment as we capture all possible viewing directions. For tracking, the all-encompassing view can increase image-based reconstruction robustness such as for structure from motion. For illumination estimation, 360° video can provide easily-obtainable environment maps for image-based lighting and a source of reflections for reflective materials [Debevec 2002], not only for static environments but also for temporally-changing ones [Iorns and Rhee 2015; Rhee et al. 2017].

Challenges exist throughout: Tracking 360° camera motion requires robust structure from motion, with many current methods reconstructing individual views from the set of two, six, or even more input camera views. To create a single equirectangular image from the separate camera views, stitching algorithms tend to stretch the boundary pixels between the input images for visually seamless

alignment. This occurs not only in consumer cameras but also in professional ones. Distorted boundary pixels can lead to incorrect feature positions and thus tracking errors. Therefore, it can be justified to use a set of images before stitching for tracking to avoid stitching errors, even if it complicates the whole process. However, most consumer 360° video cameras produce an *already-stitched* 360° video in equirectangular format, which precludes existing many-input-view 360° structure-from-motion techniques. Reprojection of this equirectangular video back onto the separate views maintains any stitching errors. Nonetheless, with a large number of tracking features evenly distributed across the image, stitching errors do not contribute significantly to the final solution. Therefore, feature tracking directly in equirectangular format has its advantages.

For rendering CG objects, a single environment map is not sufficient illumination for spatially-correct reflections as the camera moves. However, each 360° video frame can provide an environment map from a different location. Further, we wish to recover environment maps with high dynamic range (HDR) to increase lighting realism. Finally, correct compositing relies on estimating (sparse) scene geometry from camera tracking, as well as real-time rendering of CG elements with location-dependent image-based lighting.

Previous approaches have only solved a subset of these technical issues. Rhee et al.'s MR360 system uses a 360° video for image-based lighting of inserted CG objects via a real-time game engine [Rhee et al. 2017]. However, their system only supports static cameras, and not moving cameras as is desirable. Michiels et al. use a 360° video for spatially-varying physically-based rendering [Michiels et al. 2014], but their structure-from-motion implementation does not work on equirectangular projections. Further, they do not recover environment maps with HDR, which reduces the visual quality of their renderings.

To overcome these limitations, we present an approach to create real-time applications with virtual objects inserted into pre-recorded 360° video captured with a *moving* camera. We propose solutions to all three main challenges in the context of 360° videos: tracking of a moving 360° video camera, reconstruction of HDR spatially-varying environment maps for image-based lighting, and real-time rendering and compositing of virtual elements and real footage using a state-of-the-art game engine. We contribute:

- An offline structure-from-motion pipeline for 360° videos, which tracks directly on stitched equirectangular video and reconstructs the sparse structure of the environment.
- An offline recovery of spatially-varying high dynamic range environment maps via inverse tone mapping, which plausibly reproduces real-world lighting along the camera path.
- Real-time rendering of mixed-reality objects into omnidirectional video with dynamic image-based lighting and shadowing, built into the interactive Unity game engine.

These contributions expand the use and flexibility of 360° video for interactive computer graphics and visual effects applications.

2 RELATED WORK

First, we review related work on tracking 360° cameras (Section 2.1), and particularly 360° image correspondences, multi-view geometry, and structure from motion. Then, we summarize recent work on realistic lighting estimation for mixed reality (Section 2.2) and 360° object insertion rendering (Section 2.3).

2.1 360 Camera Tracking

Camera tracking is well-understood for perspective cameras, e.g., using structure from motion (SfM) or visual simultaneous localization and mapping (SLAM) [Marchand et al. 2016], but less so for 360° cameras [Caruso et al. 2015].

Image Correspondences. Every SfM or SLAM algorithm begins by finding image correspondences. For basic proofs of concept, it might be sufficient to use synthetic [Chang and Hebert 2000; Fujiki et al. 2007] or manual correspondences [Chang and Hebert 2000; Ma et al. 2015]. For practical scenarios, a variety of feature descriptors have been proposed, including BRISK [Leutenegger et al. 2011] and Affine SIFT (ASIFT) [Pagani and Stricker 2011]. While these descriptors were originally designed for standard projective cameras, they have also proven to perform well for 360° images [Benseddik et al. 2015; Pathak et al. 2017]. They also outperform the popular SIFT descriptor when applied to equirectangular images [Benseddik et al. 2015; Silveira and Jung 2017]. Spherical SIFT (SSIFT) [Cruz-Mota et al. 2012] and Spherical ORB (SPHORB) [Zhao et al. 2015] introduce spherical versions of popular planar feature descriptors. For longer videos, computing and matching feature descriptors across all frames quickly becomes expensive. In addition, if a scene contains repetitive elements, the number of mismatched features increases. We track features over multiple frames to speed up execution and reduce mismatched repetitive features.

Feature Tracking. Im et al. [2016] used the KLT tracker [Lucas and Kanade 1981; Shi and Tomasi 1994; Tomasi and Kanade 1991] separately on the two fisheye images of a commodity 360° camera. Their implementation of bundle adjustment reflects this input data model, with a cost function divided into two components associated with front and back views. However, this approach does not work for longer videos or videos with substantial camera rotation, as features are not tracked when they move from the front to the back view and vice versa, which results in shorter feature trajectories. Our implementation treats the stitched image from the camera as a sphere and therefore does not fail to track arbitrary camera rotation.

However, the majority of approaches apply standard tracking techniques to the input perspective views of 360° images before stitching, or to 360° images projected onto a cube map with six perspective views. Michiels et al. [2014] performed tracking on undistorted images from separate cameras on a 360° multi-camera rig, and thus protected the solution from stitching errors, but their method requires feature matching between the cameras in every frame which loses the main benefits of the tracking approach. Huang et al. [2017] used projection on slightly overlapping cube map sides to reliably track features near side edges, but only for a small camera movement between the frames. The most popular commercial tracking tools [Foundry 2019; Mettle 2019; The Pixel Farm 2019] also adopt the perspective approach, building on their existing 3D tracking pipelines for standard perspective cameras.

Structure from Motion. One of the earliest works on 3D reconstruction from 360° images was by Kang and Szeliski [1997]. They proposed using cylindrical panoramas to extract 3D depth from the

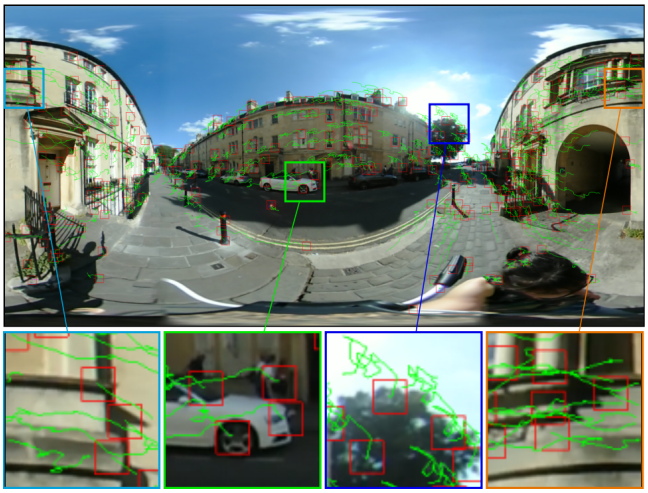

Figure 2: 360° feature tracking in progress (see Section 3.1). Note that the cyan crop (on the left edge of the equirectangular image) contains feature trajectories (green lines) that have been tracked to the orange crop (right, see red boxes).

scene to avoid merging errors which are associated with depth maps retrieved from a set of perspective images. Chang and Hebert [2000] were the first to formulate the epipolar geometry problem for 360° cameras. However, in their paper, they discuss only the catadioptric camera model and confine themselves to estimating the camera pose without reconstructing the scene. Torii et al. [2005] explained the foundations of this problem in more details and defined a general spherical camera model as consisting of a camera centre—the point in space where all viewing rays intersect—and the surface of a unit sphere surrounding the centre. They formulated two- and three-view geometry for spherical cameras by analogy to their pinhole equivalents. Fujiki et al. [2007] generalized the problem to a pair of cameras placed arbitrarily in space.

2.2 Illumination Estimation

Recovering environment illumination allows virtual objects to be rendered under matching illumination [Kronander et al. 2015]. Light field transfer provides the highest quality results [Cossairt et al. 2008], but is complex to capture. In practice, image-based lighting based on environment maps (or *light probes*) has much lower complexity, and has been shown to provide high-quality results under infinitely-far-away scene assumptions.

Many different approaches have been proposed for estimating the illumination in a scene—a problem known as *inverse lighting*. Detailed illumination can be captured directly by 360° cameras [Iorns and Rhee 2015; Michiels et al. 2014; Rhee et al. 2017; Walton and Steed 2018; Zhang and Lalonde 2017]; however, most video cameras—360° or not—do not capture imagery with high dynamic range (HDR). HDR image-based lighting improves visual results by overcoming muted low-contrast reflections [Debevec 1998, 2002] (Figure 5). Iorns and Rhee [2015] introduce basic inverse tone mapping, which they apply to low-dynamic-range 360° video and obtain satisfactory results, in some cases comparable to lighting the scene with HDR images. Their approach works in real time and produces results that can be viewed in a VR headset. Rhee et al. [Rhee et al. 2017] extend this work by adding an image-based shadow based on the brightest detected light source, i.e., usually the sun in outdoor environments. They demonstrate high-fidelity results from low-dynamic-range 360° videos in real time. Recent progress in deep learning has resulted in multiple concurrent techniques for inverse tone mapping [Eilertsen et al. 2017; Endo et al. 2017; Marnerides et al. 2018], which reconstruct HDR imagery from standard (low-dynamic-range) imagery. We employ inverse tone mapping [Endo et al. 2017] to increase the dynamic range of our environment maps.

2.3 Real-time 360 Virtual Object Insertion

One goal of object insertion is for augmented reality (AR) or mixed reality (MR): the fusion of physical and virtual realities when seen through an intermediary device such as a monitor, mobile device, or head-mounted display [Azuma 1997; Speicher et al. 2019; Tarko et al. 2017]. This is an extensive field of research, so we refer the reader to a recent book and survey for a general overview [Billinghurst et al. 2015; Schmalstieg and Höllerer 2016] and focus only on the most relevant topics.

Some works share this end goal for 360° imagery. Michiels et al. [2014] used 360° imagery for object insertion. Their approach renders reflections according to the position of the viewer and based on material properties of the inserted object. They use precomputed radiance transfer to achieve real-time physically-based global illumination. However, they appear to use 360° images directly as environment maps without first linearising the colour space or recovering HDR illumination information, which limits the visual fidelity of their results. In our approach, we use deep inverse tone mapping to recover the high dynamic range of real-world illumination.

Rhee et al. [2017] presented a complete pipeline for inserting virtual objects into 360° videos with real-time user interaction but for static cameras. This is not the case in our approach, as we adopt 360° structure from motion to reconstruct the path of a moving camera.

In summary, previous approaches only solved a subset of the technical challenges underlying real-time virtual object insertion for moving 360° videos, whereas our approach jointly solves these challenges to improve application quality and flexibility.

3 360 STRUCTURE FROM MOTION

We begin our CG element insertion by reconstructing the camera path and a sparse 3D point cloud: We implement a 360° structure-from-motion pipeline as there is no prior art for working directly in the equirectangular domain. We use a modified KLT tracker [Lucas and Kanade 1981; Tomasi and Kanade 1991] which finds point correspondences between equirectangular video frames. We apply spherical camera epipolar constraints to calculate relative camera poses and triangulate the 3D positions of the tracked points. Finally, we refine both the initial camera poses and reconstructed points with hierarchical omnidirectional bundle adjustment.

3.1 Feature Tracking

Our approach takes as input a sequence of equirectangular images and performs a modified version of KLT tracking to obtain point correspondences between consecutive frames (Figure 2). Sequential

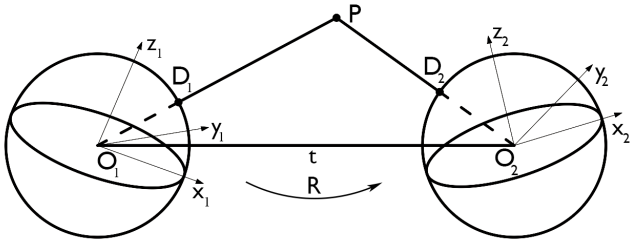

Figure 3: Omnidirectional camera epipolar geometry. World point P projects via camera centres O_1, O_2 onto sphere points D_1, D_2 with coordinate systems (x_1,y_1,z_1), (x_2,y_2,z_2) related by a translation vector t and a rotation matrix R.

tracking of video sequences produces fewer outliers and mismatched points, and is significantly faster than all-pairs feature detection, description, and matching when applied to video sequences of hundreds of frames or more, as in our case.

Our implementation is adapted from the Accord.NET KLT tracker [Jurić 2015]. We make four changes: (1) To accommodate the geometry of equirectangular projection and extend track lengths, we wrap feature point locations along the x-axis such that tracks do not stop at left and right image edges. (2) To improve robustness and reliability with long video sequences, we switch from frame-to-frame tracking to template-based tracking. This extracts and stores the template from the first frame in which a feature appears, which reduces sliding of the tracked point. (3) To create longer feature trajectories, particularly for features at the end of the video, we follow a forward pass of feature tracking with a backward pass which seeks to extend existing feature trajectories back in time. (4) We compute bidirectional tracking error between frames [Kalal et al. 2010]. If it exceeds two pixels then the feature trajectory is terminated.

3.2 Epipolar Geometry

For two perspective cameras, exploiting epipolar geometry for SfM requires estimating an intrinsic matrix **K** per camera which calibrates the field of view, and an essential matrix **E** which relates their positions and rotations. In our case with assumed stitched equirectangular images, the field of view is fixed at 360°, and so we only need to estimate the essential matrix to exploit epipolar geometry.

We estimate the essential matrix between every pair of consecutive frames using the eight-point algorithm adapted to 360° images [Chang and Hebert 2000]. Next, we decompose the essential matrix into a rotation matrix **R** and a translation vector **t** in the same way as for pinhole cameras [Hartley and Zisserman 2004]. From the four possible combinations of translation and rotation, we select the one which produces the smallest reprojection error after triangulating all points. Due to the small camera baselines between consecutive video frames, we use every 5^{th} frame instead.

Triangulating 3D Points. Given the relative pose (**R**,**t**) between two cameras, we reconstruct the point's 3D position **P** as observed at $\mathbf{D}_1$ and $\mathbf{D}_2$. For this, we use the midpoint triangulation method [Hartley and Sturm 1997], as discussed by Ma et al. [2015] in the context of 360° cameras, to find the 'intersection' of the rays from the centres of the cameras, $\mathbf{O}_1$ and $\mathbf{O}_2$, towards the observed directions to the point (Figure 3).

Reprojection error. 3D space is projected into equirectangular images in a highly non-linear way, and so pixels in different parts of the equirectangular image cover potentially different solid angles in 3D space. Therefore, the distance in pixels in the equirectangular domain is not a meaningful measure of the reprojection error, and so we must compute error in the spherical domain. Pagani and Stricker [2011] described and compared three possible types of reprojection error in the spherical domain, including the Euclidean and tangential distances:

$$\varepsilon_e = \left\| \frac{\mathbf{P}}{\|\mathbf{P}\|} - \mathbf{D}_i \right\| \quad \text{and} \quad \varepsilon_t = 2\sqrt{\frac{1-\frac{\mathbf{D}_i^\top \mathbf{P}}{\|\mathbf{P}\|}}{1+\frac{\mathbf{D}_i^\top \mathbf{P}}{\|\mathbf{P}\|}}}. \tag{1}$$

Here, **P** is the 3D point in the coordinate system of the unit sphere i (centred at the origin), and $\mathbf{D}_i$ is the vector on the sphere representing the observation of **P** in frame i as described in Section 3.1. We use the Euclidean reprojection error ε_e as it is the fastest to compute of the two reprojection errors discussed above. For each of the four solutions obtained from decomposing the essential matrix , we triangulate all point correspondences, compute the mean Euclidean reprojection error ε_e across both cameras, and select the minimum as the best solution. Finally, some outlier points may have been triangulated on the wrong side of the sphere. To cull these, we set a reprojection error threshold between $\varepsilon_e = 0$ (no error) and $\varepsilon_e = 2$ (the opposite side of the unit sphere). We experimentally discard $\varepsilon_e > 0.5$ for this task.

3.3 Multi-view Geometry

After calculating the relative camera pose (**R**,**t**) between each pair of consecutive frames, we convert all poses into consistent global coordinates. Given the pose of the first video frame, we unify the coordinate systems sequentially by updating the poses of cameras:

$$\mathbf{R}_{\text{curr}} = \mathbf{R}_{\text{rel}}\mathbf{R}_{\text{prev}}, \tag{2}$$

$$\mathbf{t}_{\text{curr}} = \mathbf{t}_{\text{prev}} + \mathbf{R}_{\text{prev}}\mathbf{t}_{\text{rel}}, \tag{3}$$

where $\mathbf{R}_{\text{curr}}$, $\mathbf{t}_{\text{curr}}$, $\mathbf{R}_{\text{prev}}$, $\mathbf{t}_{\text{prev}}$ are the current and previous-frame rotation and translation of the camera in global coordinate system, and $\mathbf{R}_{\text{rel}}$,$\mathbf{t}_{\text{rel}}$ are relative rotation and translation between the current and previous keyframes. At this point, the translation between two cameras is calculated up to scale [Hartley and Zisserman 2004], so the relative translation between each pair of consecutive frames is a unit vector. This is corrected in the next step of our approach: bundle adjustment.

3.4 360 Bundle Adjustment

We jointly optimize the camera motion and scene structure globally using bundle adjustment [Triggs et al. 2000] for 360° cameras, implemented with the Ceres solver library [Agarwal et al. 2018]. Our camera model comprises seven extrinsic parameters: four for a rotation quaternion and three for a translation vector. The cost function to be minimized is the average tangential reprojection error (Equation 1), which has shown the best convergence behaviour [Pagani and Stricker 2011]. We wrap the reprojection error within a robust Huber loss function $\rho(\cdot)$ to reduce the influence of outliers in our objective:

$$\underset{\{\mathbf{P}_p\},\{\mathbf{C}_c\}}{\operatorname{argmin}} \frac{1}{2}\sum_p \sum_c V_c^p \cdot \rho\left(\varepsilon_t^2(\mathbf{P}_p,\mathbf{C}_c,\mathbf{D}_c^p)\right), \tag{4}$$

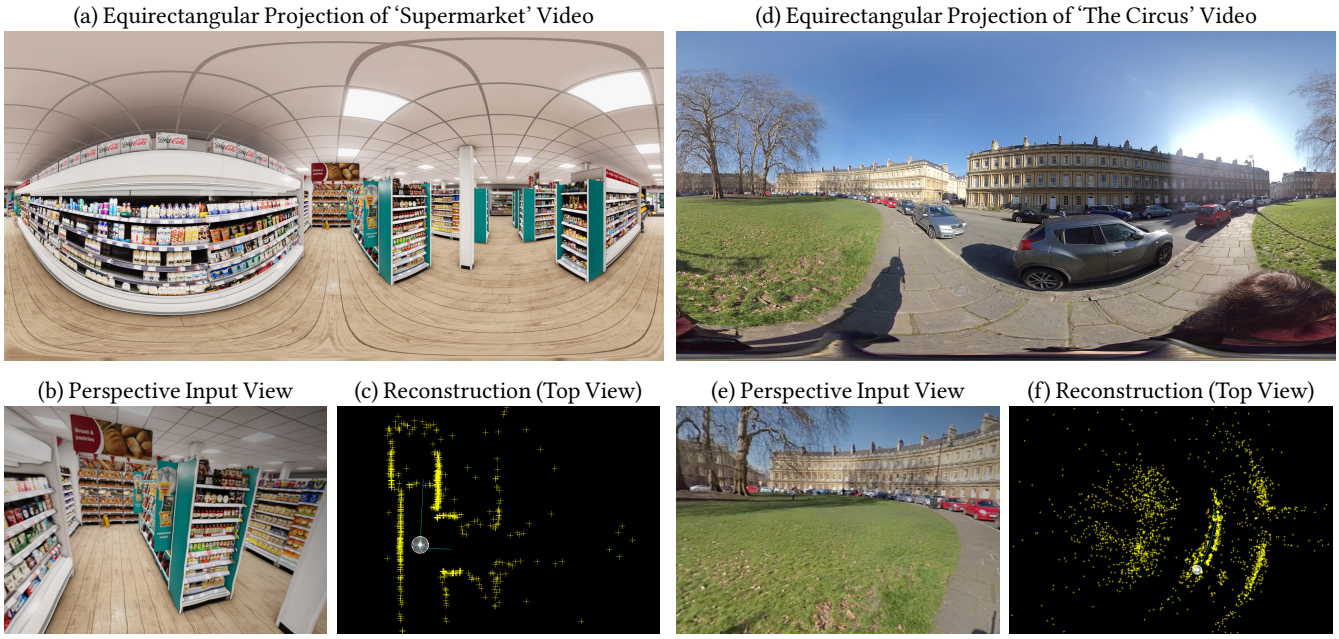

Figure 4: Results of our omnidirectional structure-from-motion pipeline. (a, d) Input frames from synthetic (a) and real (d) omnidirectional videos. (b, e) A perspective view of the input omnidirectional video. (c, f) Top views of the reconstructed camera path (cyan), with the view in (b, e) shown as white wire-frame sphere, and sparse scene geometry (yellow points).

where p iterates over all 3D points and c over all cameras, $V_c^p \in \{0,1\}$ represents visibility of point p in camera c, ε_t is the tangential reprojection error (Equation 1), $\mathbf{P}_p$ is the 3D position of point p, $\mathbf{D}_c^p$ is its projection in camera c, $\mathbf{C}_c = [q_w, q_x, q_y, q_z, t_x, t_y, t_z]$ parametrizes the pose of camera c, and ρ is the Huber loss of the squared residual [Agarwal et al. 2018].

Due to the small camera baseline between consecutive video frames, direct global optimization of all poses would be unstable as the initialization is unreliable. Instead, we implement *hierarchical two-pass bundle adjustment*. For the first pass, we use the keyframe poses computed in Section 3.3 as initial poses. As initial structure, we use the points triangulated from the first pair of keyframes in which each trajectory is observed. For the second pass, we interpolate the camera poses for in-between frames from the keyframe poses and use the previously reconstructed structure for initialization. We use linear interpolation for translation vectors and spherical linear interpolation (slerp) for quaternion interpolation [Dam et al. 1998]. This produces the final camera motion path and scene structure reconstruction (Figure 4).

4 VIRTUAL OBJECT INSERTION

High-quality virtual object rendering requires both illumination estimation to light the object plausibly, and real-time rendering to composite the objects with the video footage. We perform lighting estimation in a preprocess, and then implement real-time rendering in the Unity engine for interactive applications.

For image-based lighting, first we stabilize the 360° video to align all environment maps with the Unity global coordinate system. Then, we apply inverse tone mapping to the stabilized 360° video to recover HDR environment maps. We import these into Unity to act as a local environment map for reflections, along with the reconstructed camera path and 3D points from Section 3.4. Finally, we implement differential rendering to cast virtual shadows on top of the video footage.

360 Video Stabilization. The Unity rendering engine expects environment maps to be aligned to the global coordinate system. However, in general, this is not the case for an arbitrary 360° input video. Therefore, we need to stabilize the video to correctly orient the lighting and reflections for image-based lighting (Section 4). For each input video frame, we rotate the recovered camera in its opposite orientation, i.e., by $\mathbf{R}^\top$, so that all frames are aligned with the camera coordinate system of the first frame (which we consider to be the global coordinate system). This camera rotation corresponds to a resampling of the equirectangular image according to the rotation applied, for which we use bilinear interpolation. This eliminates shaky rotations and produces a view with a consistent horizon and up direction, and makes our video suitable for image-based lighting in Unity.

Inverse Tone Mapping. Image-based lighting with high-dynamic-range (HDR) environment maps visibly improves rendering results by overcoming muted low-contrast reflections [Debevec 2002]. Thus, we integrate inverse tone mapping into our approach, which aims to recover HDR images from low-dynamic-range (LDR) input images [Eilertsen et al. 2017; Endo et al. 2017; Marnerides et al. 2018]. We use the approach of Endo et al. [2017], which learns to estimate an exposure-bracketed set of images from a single input image via a deep convolutional neural network. Then, it merges this set of exposures into an HDR radiance map using Debevec and Malik's approach [Debevec and Malik 1997]. Figure 5 shows a comparison of image-based lighting for low- and high-dynamic-range environment maps.

Figure 5: Comparison of image-based lighting with LDR (left) and HDR (right) environment maps from the same input. The HDR version is better exposed in shadows and highlights.

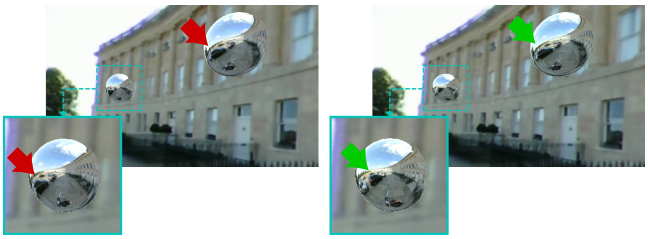

Figure 6: We insert virtual mirror spheres at two different locations in the scene. Left: A single global environment map results in identical reflections in both mirror spheres. This is incorrect, e.g., the black car indicated by arrows does not move between the near and far spheres. Right: Using multiple frames along the camera path as environment maps produces different reflections in each sphere.

Image-based Lighting in Unity. After all preprocessing steps, we import the estimated camera path, scene points, and HDR video frames as environment maps into the Unity engine for real-time dynamic image-based lighting [Debevec 2002]. Unity 2018 offers two mechanisms to use environment maps: (1) as a *skybox* which represents distant illumination, and (2) as *reflection probes* which represent nearby illumination. Every rendered frame usually uses a single skybox, based on the current video frame during playback, but multiple reflection probes can be distributed throughout the environment at the same time to model spatially-varying illumination (Figure 6).

For rendering, we do not use all video frames as reflection probes as this could easily comprise hundreds of HDR images corresponding to similar locations in space (and so similar lighting environments). Instead, we divide the camera path into a number of segments, as specified by the user, and place a reflection probe in each segment. At runtime, Unity uses the skybox for distant image-based lighting and the nearest reflection probe to compute reflections.

Image-based Shadows. Inserted CG elements must cast shadows to appear realistic as objects without shadows do not exist in the real world. Differential rendering for light transport in mixed-reality scenes allows shadow to be cast onto a virtual scene plane, such as the ground [Debevec 1998]. With this technique, the background image, CG elements, and local scene reconstructions are rendered separately, and the final composite C is calculated per pixel and per

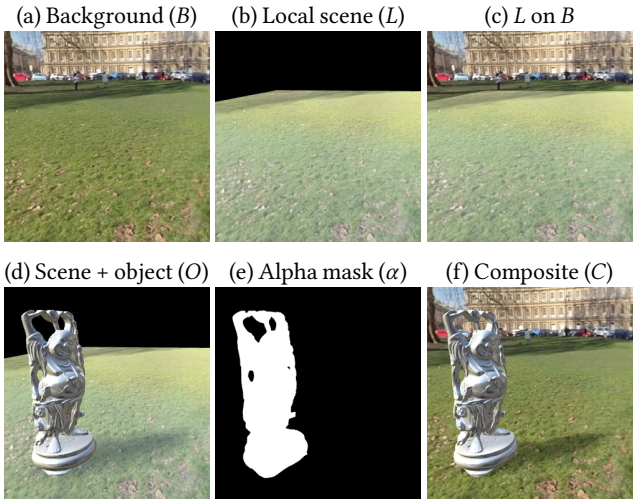

Figure 7: Components of differential rendering: (a) the background image B, (b) local scene L using a plane, (c) L rendered on top of the background B, (d) the local scene with objects O, (e) the object's alpha mask α, and (f) the composite C.

colour channel:

$$C = \alpha \cdot O + (1-\alpha) \cdot (B + O - L), \tag{5}$$

where α is an alpha matte for the inserted virtual objects, O is the image showing the rendered objects and modelled local scene, B is the background image into which the objects are to be inserted, and L is the image with only the local scene rendered. We fit the scene plane automatically to a user-selected subset of the point cloud recovered from sparse scene reconstruction. This approach uses the pixels of the rendered CG objects O within the alpha mask α, and otherwise updates the background image B with the difference $O-L$ between the local scene renderings with and without objects, which captures both shadows and inter-reflections (Figure 7).

Often, the local scene reflectance model is estimated iteratively. Instead, we apply a screen-space shader to the plane using the video as a dynamic texture to provide the diffuse colour. This ensures the correct colour of the produced shadows (Figure 8). For outdoor scenes, we manually place a light source for the sun to generate the shadow; existing techniques can also estimate this location automatically [Rhee et al. 2017].

5 RESULTS

We test our approach with video from 'Ricoh R' and 'Insta360 ONE X' 360° cameras, and content rendered from several synthetic 3D scenes. Our captured input video sequences were handheld, and our approach can track and recover sparse geometry from these sequences.

First, we test our structure-from-motion pipeline on two videos in depth (see Figure 4). The reconstructions match the input environments well, as can be seen when comparing them to a perspective sub-view of the omnidirectional input image. Next, we place virtual mirror spheres in the scene at different distances from the camera, and compare the reflections produced by two types of environment maps (Figure 6). With a single global environment map centred in the

Figure 8: Our approach enables insertion of CG objects into 360° video with real-time object manipulation. Left: The camera moves forward as the objects reflect the environment. Right: The objects translated and rotated within the scene. Top: A silver Buddha is inserted into 'The Circus' video. Note the reflection of the red car on the belly. Middle: A silver dragon inserted into the 'Crescent' video. Bottom: A golden angel, a small poster stand and a conference banner are inserted into the 'Parade' video.

global coordinate system, reflections do not show the correct perspective. Our set of spatially-varying reflection maps make virtual object reflections more convincing. Using this approach, results are most accurate when objects are placed *on* the original camera path; becoming less accurate as their distance from the camera path increases.

Figure 8 shows four examples of geometrically-complex computer-generated objects inserted into 360° videos in real time. As the virtual camera moves forward, the reflections on the objects change accordingly and the inserted virtual objects can be manipulated interactively. Objects can be moved, rotated and potentially animated, with realistic image-based lighting and shadowing in real time.

Tracking Accuracy. We test the pipeline on seven synthetic video sequences using a 360° renderer for the Facebook Replica dataset [Straub et al. 2019]. The first four sequences have camera paths which are synthetic: a straight line and a circle, with and without structured random jitter to simulate a handheld camera. The last three sequences have camera paths from the TUM RGBD SLAM dataset [Sturm et al. 2012], which we assume to be representative of real-world camera motion. Table 1 shows the average Euclidean error between our reconstruction and ground-truth camera paths. Before calculating the error, the reconstructed trajectory undergoes the Procrustes superimposition to align it with the ground truth and scale it accordingly. The structure-from-motion pipeline performed well on video sequences with no or slight vertical camera rotation, but the last two sequences with the camera rotating through the poles revealed its limitations and produced high average error. With such types of rotation, features are heavily distorted and are frequently lost by the tracker. This, in turn, leads to only a small number of good features for reconstruction and so inaccurate estimation of camera extrinsics between pairs of keyframes.

Table 1: Quantitative evaluation on synthetic 360° videos rendered from the Facebook Replica dataset [Straub et al. 2019], using the average Euclidean error (in millimeters) between reconstruction and ground-truth camera path. The first four video sequences use synthetic camera paths, with the other three camera paths coming from the TUM RGBD SLAM dataset [Sturm et al. 2012]. The high reconstruction error in the last two is caused by camera rotation over the poles, which leads to short feature trajectories as the tracker drops heavily distorted points.

Video	#frames	Error (mm)
Straight line	446	5.4 ± 2.0
+ jitter	446	4.0 ± 1.7
Circular pan	716	8.6 ± 3.5
+ jitter	716	18.9 ± 11.0
TUM path 1	696	2.9 ± 1.0
TUM path 2	696	81.9 ± 54.9
TUM path 3	996	426.2 ± 260.6

Computation Time. All timings are for an Intel Core i7 2.7 GHz processor with 16 GB RAM. Our feature tracking (Section 3.1) has an average run time of less than a second per equirectangular video frame of resolution 1920×960 pixels. Our structure-from-motion computation takes on average 13 minutes to converge for a sequence with 750 frames (averaging one second per video frame). Endo et al.'s inverse tone mapping [Endo et al. 2017] takes approximately 6 minutes per video frame of resolution 2048×1024 pixels.

Discussion. The initial estimation of camera extrinsics, and in consequence the accuracy and stability of the final optimisation, depends upon a feature template size, a minimum distance between the features, and the number of keyframes. These in turn are related to the video resolution, quality, content, and type of camera movement. As in perspective camera tracking, the values of these parameters can be optimized by a camera tracking artist.

In stabilizing the playback of the 360° video, existing artefacts such as stitching boundaries between cameras can become more noticeable as they now appear to move independently of the view. As camera stitching quality improves, this effect will be reduced.

Limitations. Our approach brings higher quality real-time rendering to 360° virtual object insertion applications; however, challenges remain. In pose estimation, our template-based tracker becomes lost when the feature distortion becomes too large towards the poles in the equirectangular image. This results in shorter trajectories and incorrectly reconstructed points caused by the tracker picking the same or spatially-very-similar features to those lost in the previous frame. Usually, there are sufficient points in the rest of the image for this not to be a problem, but it depends on the distribution of good features to track in the image and the speed of camera rotation. In principle, a better handling of these points would lead to a more robust solution.

In rendering, common problems of sampling in image-based rendering occur. Unity picks the nearest probe for lighting, as interpolating between lighting environments would result in ghosting artefacts. As probes are placed more densely, this error decreases while memory and storage load increase. Distant probes are generally fine as light sources for low-frequency lighting, but are more problematic for high-frequency reflection content of shiny materials. Our approach also cannot support inter-reflections between the video and inserted virtual elements.

Accurate shadowing of virtual objects requires a dense reconstruction of the world with a surface representation such as a mesh. Our current approach only reconstructs sparse world points, and uses a single estimated point light source for shadowing. Recovering dense world geometry is a significant open problem, but would allow realistic world occlusion and shadowing of virtual objects.

6 CONCLUSION

We proposed a system for real-time virtual object insertion for moving 360° videos which enables user interaction with the inserted objects. Our approach comprises a structure-from-motion pipeline which works directly on stitched equirectangular video and reconstructs both camera motion and the sparse structure of the environment. Further, we employ inverse tone mapping to recover high dynamic range environment maps for better lighting reproduction, and composite image-based shadows, to plausibly reproduce real-world lighting conditions for inserted computer-generated elements. We implemented our approach in the Unity engine for real-time object rendering with dynamic lighting and user interaction. This expands the use and flexibility of 360° video for graphics applications.

Future Work. We wish to use our current sparse scene reconstruction as guidance for primitive-based scene reconstruction, such as a ground plane and some walls. This improves interactions between CG elements and the captured real scene, for example by casting more complex virtual shadows onto real objects.

ACKNOWLEDGMENTS

We thank Selena Ling for creating the synthetic ground-truth videos. This research was supported by the EPSRC Centre for Doctoral Training in Digital Entertainment (EP/L016540/1), by Checkmate VR and Dc-activ as part of their R&D programme, RCUK grant CAMERA (EP/M023281/1), and an EPSRC-UKRI Innovation Fellowship (EP/S001050/1).

REFERENCES

Sameer Agarwal, Keir Mierle, and Others. 2018. Ceres Solver. http://ceres-solver.org. http://ceres-solver.org

Ronald T. Azuma. 1997. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments* 6, 4 (August 1997), 355–385. https://doi.org/10.1162/pres.1997.6.4.355

H E Benseddik, H Hadj-Abdelkader, B Cherki, and O Djekoune. 2015. Binary Feature Descriptor for Omnidirectional Images Processing. In *IPAC*. https://doi.org/10.1145/2816839.2816848

Mark Billinghurst, Adrian Clark, and Gun Lee. 2015. A Survey of Augmented Reality. *Foundations and Trends in Human–Computer Interaction* 8, 2–3 (2015), 73–272. https://doi.org/10.1561/1100000049

David Caruso, Jakob Engel, and Daniel Cremers. 2015. Large-scale direct SLAM for omnidirectional cameras. In *IROS*. 141–148. https://doi.org/10.1109/IROS.2015.7353366

Peng Chang and Martial Hebert. 2000. Omni-directional structure from motion. In *CVPR Workshops*. 127–133. https://doi.org/10.1109/OMNVIS.2000.853819

Oliver Cossairt, Shree K. Nayar, and Ravi Ramamoorthi. 2008. Light field transfer: global illumination between real and synthetic objects. *ACM Trans. Graph.* 27, 3 (August 2008), 57:1–6. https://doi.org/10.1145/1360612.1360656

Javier Cruz-Mota, Iva Bogdanova, Benoît Paquier, Michel Bierlaire, and Jean-Philippe Thiran. 2012. Scale Invariant Feature Transform on the Sphere: Theory and Applications. *Int. J. Comput. Vision* 98, 2 (June 2012), 217–241. https://doi.org/10.1007/s11263-011-0505-4

Erik B. Dam, Martin Koch, and Martin Lillholm. 1998. *Quaternions, Interpolation and Animation.* Technical Report DIKU-TR-98/5. Datalogisk Institut, Københavns Universitet.

Paul Debevec. 1998. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH*. 189–198. https://doi.org/10.1145/280814.280864

P. Debevec. 2002. Image-based lighting. *IEEE Comput. Graph. Appl.* 22, 2 (March/April 2002), 26–34. https://doi.org/10.1109/38.988744

Paul E. Debevec and Jitendra Malik. 1997. Recovering High Dynamic Range Radiance Maps from Photographs. In *SIGGRAPH*. 369–378. https://doi.org/10.1145/258734.258884

Gabriel Eilertsen, Joel Kronander, Gyorgy Denes, Rafał K. Mantiuk, and Jonas Unger. 2017. HDR image reconstruction from a single exposure using deep CNNs. *ACM Trans. Graph.* 36, 6 (November 2017), 178:1–15. https://doi.org/10.1145/3130800.3130816

Yuki Endo, Yoshihiro Kanamori, and Jun Mitani. 2017. Deep Reverse Tone Mapping. *ACM Trans. Graph.* 36, 6 (November 2017), 177:1–10. https://doi.org/10.1145/3130800.3130834

Foundry. 2019. Cara VR. https://www.foundry.com/products/cara-vr

Jun Fujiki, Akihiko Torii, and Shotaro Akaho. 2007. Epipolar Geometry Via Rectification of Spherical Images. In *Proc. Mirage*. 461–471. https://doi.org/10.1007/978-3-540-71457-6_42

Richard Hartley and Andrew Zisserman. 2004. *Multiple View Geometry in Computer Vision.* Cambridge University Press. https://doi.org/10.1017/CBO9780511811685

Richard I. Hartley and Peter Sturm. 1997. Triangulation. *Comput. Vision Image Understanding* 68, 2 (November 1997), 146–157. https://doi.org/10.1006/cviu.1997.0547

Jingwei Huang, Zhili Chen, Duygu Ceylan, and Hailin Jin. 2017. 6-DOF VR videos with a single 360-camera. In *IEEE VR*. 37–44. https://doi.org/10.1109/VR.2017.7892229

Sunghoon Im, Hyowon Ha, François Rameau, Hae-Gon Jeon, Gyeongmin Choe, and In So Kweon. 2016. All-around Depth from Small Motion with A Spherical Panoramic Camera. In *ECCV*. https://doi.org/10.1007/978-3-319-46487-9_10

Thomas Iorns and Taehyun Rhee. 2015. Real-Time Image Based Lighting for 360-Degree Panoramic Video. In *PSIVT Workshops*. 139–151. https://doi.org/10.1007/978-3-319-30285-0_12

Darko Jurić. 2015. Accord.NET Extensions. https://github.com/dajuric/accord-net-extensions.

Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. 2010. Forward-Backward Error: Automatic Detection of Tracking Failures. In *ICPR*. 2756–2759. https://doi.org/10.1109/ICPR.2010.675

Sing Bing Kang and Richard Szeliski. 1997. 3-D Scene Data Recovery Using Omnidirectional Multibaseline Stereo. *Int. J. Comput. Vision* 25, 2 (November 1997), 167–183. https://doi.org/10.1023/A:1007971901577

Joel Kronander, Francesco Banterle, Andrew Gardner, Ehsan Miandji, and Jonas Unger. 2015. Photorealistic rendering of mixed reality scenes. *Comput. Graph. Forum* 34, 2 (May 2015), 643–665. https://doi.org/10.1111/cgf.12591

Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. 2011. BRISK: Binary Robust invariant scalable keypoints. In *ICCV*. 2548–2555. https://doi.org/10.1109/ICCV.2011.6126542

Bruce D. Lucas and Takeo Kanade. 1981. An iterative image registration technique with an application to stereo vision. In *IJCAI*.

Chuiwen Ma, Liang Shi, Hanlu Huang, and Mengyuan Yan. 2015. 3D Reconstruction from Full-view Fisheye Camera. (2015). https://arxiv.org/abs/1506.06273 arXiv:1506.06273.

E. Marchand, H. Uchiyama, and F. Spindler. 2016. Pose Estimation for Augmented Reality: A Hands-On Survey. *IEEE Trans. Vis. Comput. Graph.* 22, 12 (December 2016), 2633–2651. https://doi.org/10.1109/TVCG.2015.2513408

Demetris Marnerides, Thomas Bashford-Rogers, Jonathan Hatchett, and Kurt Debattista. 2018. ExpandNet: A Deep Convolutional Neural Network for High Dynamic Range Expansion from Low Dynamic Range Content. *Comput. Graph. Forum* 37, 2 (May 2018), 37–49. https://doi.org/10.1111/cgf.13340

Mettle. 2019. SkyBox Studio V2. https://www.mettle.com/product/skybox-studio-v2/

Nick Michiels, Lode Jorissen, Jeroen Put, and Philippe Bekaert. 2014. Interactive Augmented Omnidirectional Video with Realistic Lighting. In *Augmented and Virtual Reality*. 247–263. https://doi.org/10.1007/978-3-319-13969-2_19

Alain Pagani and Didier Stricker. 2011. Structure from Motion using full spherical panoramic cameras. In *ICCV Workshops*. 375–382. https://doi.org/10.1109/ICCVW.2011.6130266

Sarthak Pathak, Alessandro Moro, Hiromitsu Fujii, Atsushi Yamashita, and Hajime Asama. 2017. 3D reconstruction of structures using spherical cameras with small motion. In *ICCAS*. 117–122. https://doi.org/10.1109/ICCAS.2016.7832307

T. Rhee, L. Petikam, B. Allen, and A. Chalmers. 2017. MR360: Mixed Reality Rendering for 360° Panoramic Videos. *IEEE Trans. Vis. Comput. Graph.* 23, 4 (April 2017), 1379–1388. https://doi.org/10.1109/TVCG.2017.2657178

Davide Scaramuzza. 2014. Omnidirectional Camera. In *Computer Vision: A Reference Guide*, Katsushi Ikeuchi (Ed.). Springer, 552–560.

Dieter Schmalstieg and Tobias Höllerer. 2016. *Augmented Reality: Principles and Practice*. Addison-Wesley.

Jianbo Shi and Carlo Tomasi. 1994. Good features to track. In *CVPR*. 593–600. https://doi.org/10.1109/CVPR.1994.323794

Thiago Lopes Trugillo Silveira and Claudio Rosito Jung. 2017. Evaluation of Keypoint Extraction and Matching for Pose Estimation Using Pairs of Spherical Images. In *SIBGRAPI*. 374–381. https://doi.org/10.1109/SIBGRAPI.2017.56

Maximilian Speicher, Brian D. Hall, and Michael Nebeling. 2019. What is Mixed Reality?. In *CHI*. 537:1–15. https://doi.org/10.1145/3290605.3300767

Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. 2019. The Replica Dataset: A Digital Replica of Indoor Spaces. (2019). https://github.com/facebookresearch/Replica-Dataset arXiv:1906.05797.

Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. 2012. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *IROS*. https://vision.in.tum.de/data/datasets/rgbd-dataset

Joanna Tarko, Christian Richardt, and Peter Hall. 2017. Dynamic Mixed-Reality Compositing with Unity. In *CVMP Short Papers*.

The Pixel Farm. 2019. Spherical Tracking Toolset for PFTrack. https://www.thepixelfarm.co.uk/pftrack/

Carlo Tomasi and Takeo Kanade. 1991. *Detection and Tracking of Point Features*. Technical Report CMU-CS-91-132. Carnegie Mellon University.

Akihiko Torii, Atsushi Imiya, and Naoya Ohnishi. 2005. Two- and Three-View Geometry for Spherical Cameras. In *ICCV Workshops*. http://www.fieldrobotics.org/~cgeyer/OMNIVIS05/final/Torii.pdf

Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew Fitzgibbon. 2000. Bundle Adjustment – A Modern Synthesis. In *ICCV*. 298–372. https://doi.org/10.1007/3-540-44480-7_21

David R. Walton and Anthony Steed. 2018. Dynamic HDR Environment Capture for Mixed Reality. In *VRST*. 18:1–11. https://doi.org/10.1145/3281505.3281531

Jinsong Zhang and Jean-François Lalonde. 2017. Learning High Dynamic Range from Outdoor Panoramas. In *ICCV*. 4529–4538. https://doi.org/10.1109/ICCV.2017.484

Qiang Zhao, Wei Feng, Liang Wan, and Jiawan Zhang. 2015. SPHORB: A Fast and Robust Binary Feature on the Sphere. *Int. J. Comput. Vision* 113, 2 (June 2015), 143–159. https://doi.org/10.1007/s11263-014-0787-4