6-DOF VR Videos with a Single 360-Camera

Jingwei Huang*
Stanford University
Adobe Research
USA

Zhili Chen†
Adobe Research
USA

Duygu Ceylan‡
Adobe Research
USA

Hailin Jin§
Adobe Research
USA

Abstract

Recent breakthroughs in consumer level virtual reality (VR) headsets are creating a growing user-base in demand for immersive, full 3D VR experiences. While monoscopic 360-videos are perhaps the most prevalent type of content for VR headsets, they lack 3D information and thus cannot be viewed with full 6 degree-of-freedom (DOF). We present an approach that addresses this limitation via a novel warping algorithm that can synthesize new views both with rotational and translational motion of the viewpoint. This enables the ability to perform VR playback of input monoscopic 360-videos files in full stereo with full 6-DOF of head motion. Our method synthesizes novel views for each eye in accordance with the 6-DOF motion of the headset. Our solution tailors standard structure-from-motion and dense reconstruction algorithms to work accurately for 360-videos and is optimized for GPUs to achieve VR frame rates (>120 fps). We demonstrate the effectiveness our approach on a variety of videos with interesting content.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality; I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—Video analysis I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Motion

1 Introduction

We are witnessing an increasing interest in immersive, 360-degree virtual environments triggered by the recent breakthroughs in consumer level virtual reality (VR) hardware such as Oculus and Samsung Gear VR headsets. 360-videos are by far one of the most popular forms of content for such devices. Monoscopic 360-videos are easy to capture for novice users with commodity hardware. They can either be created by stitching footage from multiple regular cameras mounted in a custom-built rig or captured directly by a one-click 360-camera like Samsung Gear 360 and Ricoh Theta. The latter solution is gaining its popularity due to low cost and simplicity.

However, a few challenges arise when viewing 360-videos in VR headsets. First, with a monoscopic video, the same content is rendered for both eyes, resulting in a lack of 3D depth sensation. Second, monoscopic 360-videos can only respond to rotational motion while most VR headsets support full 6 degree-of-freedom (DOF) tracking which includes both the rotational and translational motion of the head. This limited DOF of the viewing experience is far less engaging and immersive compared to other full 3D VR content like games. Furthermore, the lack of DOF can also lead to more motion sickness because visual feedback does not exactly match the actual head movement. Unfortunately, capturing full 6-DOF 360-videos requires more complex hardware setups like camera arrays or light-field cameras that are not affordable and easy to use for novice users. Furthermore, there is value in improving the viewing experience of the large and growing collection of existing monoscopic 360-videos.

In this paper, we present an algorithm that enhances monoscopic 360-videos with a 6-DOF and stereoscopic VR viewing experience. Given an input monoscopic 360-video, in an offline stage we infer the camera path and the 3D scene geometry by adapting standard structure-from-motion techniques to work with 360 videos. We then playback the input video in a VR-headset where we track the 6-DOF motion of the headset and synthesize novel views that correspond to this motion. We synthesize a new view for each eye in parallel to achieve the stereoscopic viewing experience. Our main contribution is a novel warping algorithm that synthesizes novel views on the fly by warping the original content. Unlike any other previous method, this warping technique directly works on the unit sphere and therefore minimizes distortions when warping spherical panoramas. Moreover, we optimize our warping solution for GPUs and achieve VR frame rates (>120 fps). To summarize, our contributions include:

- a novel spherical panorama warping method that minimizes distortions,
- an optimized prototype that can synthesize novel views for each eye at VR frame rates (>120 fps) on a mainstream graphics card,
- tailoring state-of-the-art sparse and dense reconstruction algorithms to work with monoscopic 360-videos.

2 Related Work

The ability to re-render a recorded footage from novel viewpoints, likely for free-viewpoint or stabilization purposes, has been an active research area for a long time. We group the related work in this area based on the target application and the sensors being used.

Free-Viewpoint Video. Free-viewpoint video (FVV) refers to the ability to playback a captured (dynamic) scene from any 3D viewpoint. The research in this domain has been mainly pioneered by the work of Kanade et al. [14] who have introduced the concept of *virtualized reality*. This work used a dome of cameras to compute a 3D reconstruction of the captured scene which was then utilized for novel view synthesis. The follow up work [30, 19, 5] has mainly utilized the same principle of using a set of synchronized cameras and controlled capture setups. These approaches utilize expensive and pre-calibrated capture stages to reconstruct a detailed textured 3D reconstruction of the scene and directly render this geometry from novel viewpoints.Another thread of work utilizes image-based rendering techniques to synthesize new views of an object [29] or a scene [4] from a high quality reconstruction. Whether the data was captured in a controlled setup or in the wild, all these approaches focus on accurate and detailed 3D reconstruction to produce high-quality synthesized views. In comparison, we work with monoscopic 360-videos captured by amateur consumers using inexpensive hardware in uncontrolled

*e-mail: jingweih@stanford.edu
†e-mail: zlchen@adobe.com
‡e-mail: ceylan@adobe.com
§e-mail: hljin@adobe.com

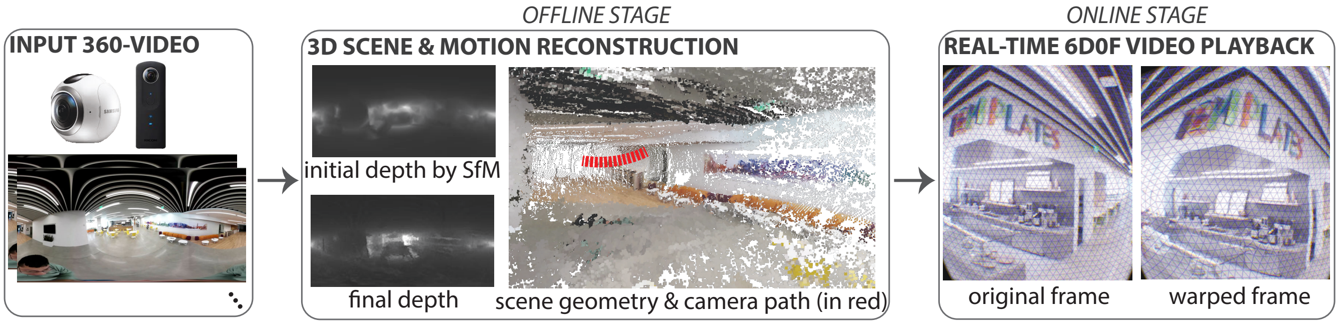

OFFLINE STAGE
ONLINE STAGE
INPUT 360-VIDEO
3D SCENE & MOTION RECONSTRUCTION
REAL-TIME 6D0F VIDEO PLAYBACK
initial depth by SfM
final depth
scene geometry & camera path (in red)
original frame
warped frame

Figure 1: Given a 360-video captured by a single spherical panorama camera, in an offline pre-processing stage, we recover the camera motion and the scene geometry first by performing structure-from-motion (SfM) followed by dense reconstruction. Then, in real-time we playback the video in a VR headset where we track the 6-DOF motion of the headset and synthesize new views by a novel warping algorithm.

setup which often lead to noisy and incomplete 3D reconstructions. Our contribution is a novel real-time image warping algorithm that synthesizes new views by utilizing this imperfect reconstruction only as a guidance.

Video Stabilization. Given a video sequence captured from a single camera, stabilization research focuses on synthesizing output frames along a new desired and possibly smoother camera path. Early works in this domain [22] have utilized 2D motion models due to their robustness and computational efficiency. However, these models are often too simple to handle complex motions leading to the development of 3D motion models [1]. Most 3D video stabilization algorithms work based on a 3-step approach. First, the 3D camera motion of the original video is tracked along with a sparse reconstruction of the scene. Next, the tracked path is smoothed or adjusted based on a desired camera path. Finally, new frames that would have been seen from viewpoints along the new camera path are synthesized via a real-time image warping algorithm.

Several 3D video stabilization algorithms have been introduced recently that focus on smart warping algorithms for novel view synthesis [16] or explore motion-specific constraints when recovering the original camera motion [17, 18]. Most of these approaches, however, work with narrow-field-of-view videos captured by standard perspective cameras. We extend these approaches to work with 360-videos. Moreover, our approach is applicable but not limited to video stabilization. Our goal is to synthesize novel viewpoints guided by the motion of the headset used to play back the video.

Closely related to our approach is the work of Kamali et al. [13] who use a sparse reconstruction obtained by structure-from-motion to stabilize omnidirectional videos. When synthesizing novel views along a smoother camera path, they propose to use a triangular grid sampled on the unit sphere of the camera instead of a conventional square grid to guide the image warping. This is beneficial as it provides a more uniform sampling in camera space. However, the distortion is still measured in the image space. In contrast, our warping field is directly generated on the unit sphere leading to more accurate results. Moreover, they utilize only a sparse 3D reconstruction which requires the new camera path to be sufficiently close to the original camera path. In contrast, we compute the dense scene geometry and show that this leads to more accurate results when synthesizing novel views with full 6-DOF head motion. We refer the reader to Section 5.2 for comparisons and a more detailed discussion.

Structure-from-Motion. Most novel view synthesis approaches are guided by a sparse or dense reconstruction of the captured scene obtained via structure-from-motion (SfM) algorithms [11]. There is an extensive body of work on SfM with narrow-field-of-view imagery [26, 9, 7]. More recently, algorithms that work with wide-field-of-view cameras such as fish-eye lenses and omnidirectional cameras have been proposed [3, 21, 23, 15, 24, 10, 2, 12]. Such methods are complementary to our approach, since we utilize the 3D reconstruction obtained with an SfM algorithm to guide the novel view synthesis.

3 Overview

We present a system that provides 6-DOF stereoscopic viewing of monoscopic 360-videos in a head-mounted display. Our approach consists of two main steps as shown in Figure 1. Given a monoscopic 360-video shot with a spherical panoramic camera, in an offline stage, we first recover the motion of the camera and a dense 3D reconstruction of the captured scene. We do this by first utilizing a structure-from-motion (SfM) algorithm to compute the camera motion and a sparse reconstruction of the scene. We then employ a dense reconstruction algorithm to recover a dense 3D geometry representation of the scene. We note that we accommodate both the SfM and dense reconstruction algorithms to work with 360-videos.

Once the 3D geometry of the scene and the camera path has been recovered offline, we allow online playback of the video in a head mounted display in real time. During this playback, we track the 6-DOF motion of the headset and synthesize novel viewpoints for each eye in real-time to provide a 6-DOF stereoscopic viewing experience. Novel view synthesis is done by a spherical image warping algorithm guided by the 3D scene geometry.

4 Approach

We now describe each stage of our solution in detail.

4.1 3D Scene and Motion Reconstruction

Sparse Scene Reconstruction. In order to recover the 3D scene geometry, we first employ a structure-from-motion (SfM) algorithm to compute the camera motion and a sparse 3D reconstruction. We refer the readers to the work of Hartley and Zisserman [11] for a detailed explanation of a standard SfM algorithm while we describe how we adopt this algorithm to work with 360-videos.

The input to a standard SfM algorithm is a set of feature points tracked in multiple images. Each tracked point is a subset of image points which are believed to be the projection of the same 3D point. The SfM algorithm recovers the position of each such 3D point as well as the camera parameters of each image observing these points. In order to track feature points in a panoramic video, we first map every frame of the video onto six planes of a cube map and then run a standard KLT tracker algorithm [20, 27] on each of these 6 image planes (Figure 2). However, an independent tracking of feature points on each of these image planes can potentially lead to artifacts due to feature points detected at the boundary of the faces of the cube. Such points are likely to be mapped to different faces of the cube at consecutive frames resulting in a tracking failure. To address this issue, when mapping each frame of the video to a cube map we utilize a field-of-view (FOV) greater than 45° (48° in our experiments) to generate overlapping regions between image planes corresponding to the neighboring faces of the cube. Once feature tracking is completed on the six images of the current frame, each feature tracked in the overlapping region is assigned back to the its original corresponding image plane. As a result, we can safely track feature points that are close to the edges of the cube as long as the camera motion between successive frames is not large.

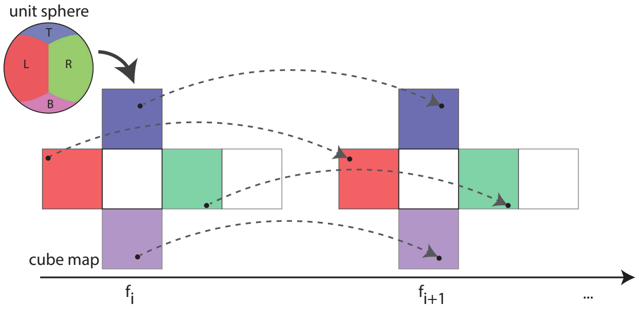


Figure 2: For each consecutive frame f_i and f_{i+1}, we map the unit spheres of the corresponding cameras to a cube map and perform feature tracking on each of the six image planes of the cube (T-top, B-bottom, L-left, R-right etc.) in parallel.

After we track a set of feature points across the duration of the video, our next step is to run the SfM algorithm to compute the sparse scene geometry and the camera parameters for each frame. Since the duration of a video can be arbitrarily long, we first run SfM on a set of keyframes (every 12^{th} frame in our experiments). We employ an incremental approach for recovering the camera parameters of each keyframe where the camera of the next keyframe is initialized with the camera parameters of the previous keyframe. (The camera projection matrix of the first keyframe is initialized as the identity matrix.) We then refine this initial camera guess by minimizing the reprojection error of the already reconstructed 3D points onto the new keyframe. Finally, at each step of this incremental approach, we run a bundle adjustment step to optimize both for the positions of the 3D points tracked so far and also the camera parameters of all of the current keyframes.

After the orientation and position of the camera for each keyframe is determined, we initialize the cameras of the in-between frames with a linear interpolation between the camera parameters of the neighboring keyframes (we interpolate the translation and the quaternion representation of the rotation of the camera independently). Similar to the case of keyframes, we first refine these initial camera guesses by minimizing the reprojection error of the reconstructed sparse 3D point cloud onto the in-between frames. We then perform one final bundle adjustment step to further refine the camera parameters of each frame (both keyframes and in-between frames) and position of the sparse 3D points corresponding to the tracked feature points. Finally, we note that given n 3D points and m frames, the bundle adjustment step aims to refine the 3D position of each point $\mathbf{p_i}$ and the camera parameters of each frame $\mathbf{c_j}$ such that the reprojection error is minimized:

$$\min_{\mathbf{p_i},\mathbf{c_j}} \sum_{1}^{n} \sum_{1}^{m} v_{ij}(\rho(\mathbf{p_i},\mathbf{c_j}) - \mathbf{x_{ij}})^2. \tag{1}$$

ρ denotes the projection of a 3D point onto the unit sphere of a camera. v_{ij} is a binary variable to indicate whether the 3D point $\mathbf{p_i}$ is tracked in frame j, and if so $\mathbf{x_{ij}}$ denotes the position of the corresponding feature point on the unit sphere of frame j. Measuring the reprojection error directly on the unit sphere instead of the panorama image leads to more accurate results as it avoids the additional distortions caused by the projection from the sphere to the image space.

Dense Scene Reconstruction. Once we obtain the sparse scene geometry and the camera motion, we adopt the method of Shen et al. [25] to compute the dense scene geometry. This method focuses on computing a depth map for each image by iteratively performing a sequence of random assignment and propagation operations. In our case, we initialize the depth map of each frame using the sparse reconstruction obtained in the previous step. Specifically, we map each 3D point reconstructed in the previous step onto the unit sphere of each frame as vertices and perform Delaunay Triangulation. We obtain the depth value for each point inside a triangle by linearly interpolating the depth values of its vertices. We obtain per-pixel depth values by rasterizing the triangulation on the sphere onto the panorama image.

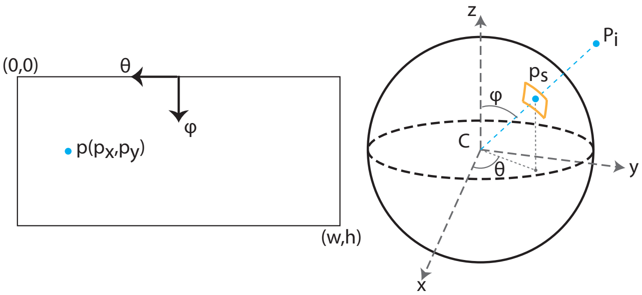


Figure 3: A 3D point P_i projects to the pixel $p(p_x, p_y)$ in the $(w \times h)$ panorama image and the point $p_s(cos\theta sin\phi, sin\theta sin\phi, cos\phi)$ on the unit sphere where $\theta = 2\pi p_x/w$ and $\phi = \pi p_y/h$. When computing the matching cost of this point with neighboring views, we define a local patch around p_s on the unit sphere (in yellow) with axes directions $(-sin\theta, cos\theta, 0)$ and $(-cos\theta cos\phi, -sin\theta cos\phi, sin\phi)$.

The propagation step aims to refine the depth value of each pixel via the depth of its neighboring pixels such that a matching cost is minimized. The matching cost evaluates the normalized cross correlation (NCC) score of a local window centered around a pixel in a reference image and its corresponding pixel in a neighboring image (i.e., neighboring frames of the video with sufficient overlap based on the relative camera transformations). For standard perspective images, this window is simply defined as a square aligned with the image axes. In our case, however, we define this local window directly on the unit sphere (see Figure 3) to uniformly capture the neighborhood of a point irrespective of its location on the sphere.

Once the per-frame depth maps are computed, we merge them into a single 3D point cloud representation of the scene. During this merge step, we filter out the points that incur a high matching cost and do not obey the visibility constraints to reduce the noise and outliers in the final 3D scene geometry. We note that we use a point cloud representation of the scene (see Figure 4) in stead of a triangulated mesh as each 3D point is directly utilized to guide the warping field as described next.

4.2 Real-time 6DOF Video Playback

After we pre-process the input video to compute the dense 3D scene geometry and camera motion, we play it in a head-mounted display in a stereoscopic fashion in real time. During this playback, we track the viewpoint corresponding to left and right eye as the head moves (rotation and translation) and utilize it to generate a

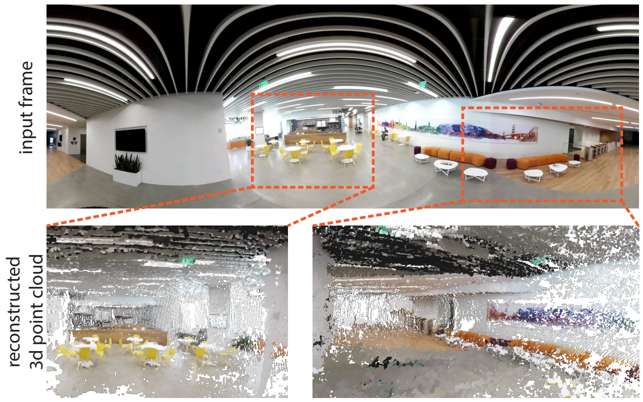


Figure 4: We show one frame of the input video and different parts of the 3D scene geometry obtained by the dense reconstruction step.

novel view for each eye by warping the current frame. As a result, the user can freely view each frame from a varying set of desired views, sensing a full 3D experience.We warp the original content by a novel warping algorithm that is specifically designed to work with spherical panorama images and aims to minimize distortions. We next describe this warping process for one eye and note that in practice we perform this twice, one for each eye, in parallel to generate a stereo pair. We use IPD provided by the VR headset to adjust the camera locations corresponding to two eyes so that scale of the scene looks true to life.

Our image warping is guided by a control triangular mesh G defined on a unit sphere by icosahedron tessellation. When warping a current frame to synthesize a novel view, we desire the motion of this control mesh to well represent the relative motion between the cameras, while being smooth to avoid distortions. Furthermore, our goal is to ensure each vertex of the control mesh moves only on the surface of the sphere. The vertex motion can simply be parametrized by a linear combination of two orthogonal vectors tangential to the spherical surface. While such orthogonal vectors can be randomly picked for each vertex, this strategy makes it hard to measure the motion consistency between neighboring vertices. Therefore, we instead define the parameterization using a conjugate direction field [6] which ensures a small angle difference between direction vectors defined on adjacent faces and thus results in a smooth parameterization of the sphere surface. More precisely, we define a conjugate direction field on G so that the motion of each vertex $V_j \in G$ is defined by the two tangential vectors $\mathbf{u_j^1}$ and $\mathbf{u_j^2}$: $f(V_j) = a_j\mathbf{u_j^1} + b_j\mathbf{u_j^2}$. Motion difference between neighboring vertices is then simply computed as the difference between the coordinates of the parameterization defined by the conjugate direction vectors. We note that the control mesh and the conjugate direction field is computed only once for a unit sphere and shared across all the frames.

Given a reference current frame with camera orientation and translation $(\mathbf{R}, \mathbf{t})$ and a desired novel viewpoint with orientation and translation $(\mathbf{R}', \mathbf{t}')$, our goal is to compute the parameters $\{a_j, b_j\}$ that warp the current frame to the desired frame. We setup an optimization that minimizes an energy function composed of data and regularization terms to compute this warping. While the data term ensures the warping accurately captures the relative motion between the views, the regularization term helps to avoid distortions.

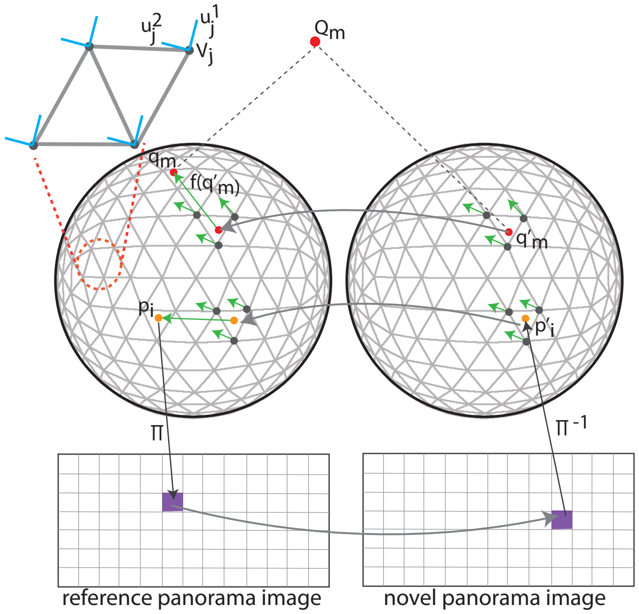


Figure 5: To synthesize a novel panorama image, we apply a warp to the current reference image guided by a control mesh defined on the unit sphere. This warping moves the projection of each 3D scene point Q_m in the new frame (q'_m) to its corresponding projection point in the reference frame (q_m). This movement, $f(q'_m)$, is defined by the movement of the vertices of the triangle the point belongs to. Furthermore, each triangle vertex V_j is moved by its conjugate vectors u_j^1 and u_j^2 (in blue) such that it is constrained to move only on the surface of the sphere. We utilize the computed warping field to map each pixel in the novel panorama image to its corresponding pixel in the reference frame to transfer the color information.

Given the 3D dense geometry of the captured scene, we project each 3D point $\mathbf{Q_m}$ to the unit spheres corresponding to the reference and the desired frames to obtain the projections $\mathbf{q_m}$ and $\mathbf{q'_m}$ respectively. Each such projected point falls inside a triangle of the control mesh (see Figure 5). We denote the triangle that contains $\mathbf{q'_m}$ by $t(\mathbf{q'_m})$ and the indices of the vertices of this triangle are denoted as $t(\mathbf{q'_m}, 0)$, $t(\mathbf{q'_m}, 1)$, and $t(\mathbf{q'_m}, 2)$ respectively. During warping, the movement of any projected point $\mathbf{q'_m}$ is defined as a linear combination of the movement of the vertices of its covering triangle: $f(\mathbf{q'_m}) = \sum_{k\in\{0,1,2\}} w_{k,m} f(V_{t(\mathbf{q'_m},k)})$, where $w_{k,m}$ denotes the barycentric coordinates of the point. The data term E_d forces each point $\mathbf{q'_m}$ to move in the desired frame to its corresponding position $\mathbf{q_m}$ in the reference frame:

$$\begin{aligned} E_d &= \sum_{\mathbf{q'_m}} \| q'_m + f(\mathbf{q'_m}) - q_m \|^2, \\ &= \sum_{\mathbf{q'_m}} \| q'_m + (\sum_{k\in\{0,1,2\}} w_{k,m} f(V_{t(\mathbf{q'_m},k)})) - q_m \|^2. \end{aligned} \tag{2}$$

The regularization term E_r, on the other hand, enforces that each pair of vertices connected by an edge $e(V_i, V_j) \in G$ move similarly to avoid distortions:

$$\begin{aligned} E_r &= \sum_{e(V_i,V_j)\in G} \| f(V_i) - f(V_j) \|^2, \\ &= \sum_{e(V_i,V_j)\in G} \| (a_i\mathbf{u_i^1} + b_i\mathbf{u_i^2}) - (a_j\mathbf{u_j^1} + b_j\mathbf{u_j^2}) \|^2. \end{aligned} \tag{3}$$

We combine the data and regularization terms and optimize for the warping parameters $\{a_j, b_j\}$ that minimize the resulting energy:

$$\min_{\{a_j,b_j\}} E_d + \lambda E_r, \tag{4}$$

where λ denotes the relative weighting of the two terms (set to 50 in our experiments).

Once the warping parameters are computed, we utilize them to synthesize the new desired frame. Specifically, for each pixel in the desired panorama image, we first map its location onto the unit sphere of the desired frame via inverse spherical projection (π^{-1}) to obtain the point p'_i (see Figure 5). Our goal is to find the corresponding point on the unit sphere of the reference frame. We first find the triangle $t(p'_i)$ in the desired frame that p'_i belongs to and utilize the movement of the vertices of this triangle $\sum_{k\in\{0,1,2\}} w_{k,i} f(t(p'_i),k)$ to obtain the corresponding point $p_i = p'_i + \sum_{k\in\{0,1,2\}} w_{k,i} f(t(p'_i),k)$ in the reference frame. Finally, we map p_i to the panorama image of the reference frame via spherical projection (π) and retrieve the color that should be transferred to the desired panorama image. We note that since the mapping between a unit sphere and a panorama image is fixed, for each pixel in a panorama image we pre-compute the triangle that contains its inverse projection onto the unit sphere and the corresponding barycentric coordinates.

While it is possible to warp the entire panorama image for each frame, in practice only a certain region of this image is visible to the user at any given time. Thus, at each frame we first identify the region of interest in the panorama image visible to the user and perform warping only for this region. Our experiments prove that this optimization leads to nearly 35% computational efficiency without any loss in the warping quality, as shown in Table 1.

Implementation Details. We implement the real-time warping algorithm on the GPU using OpenGL. Specifically, we first stream the necessary data (i.e., the video frames and the control mesh) to GPU. Equation 4 is then solved using the Jacobi method implemented inside a GLSL compute shader. The results are loaded into the fragment shader as an OpenGL texture where the color of each visible pixel in the novel view is computed directly as illustrated in Figure 5. This GPU implementation achieves frame rates greater than 120 fps and results in a smooth VR experience.

5 Evaluation

We have evaluated our approach on a variety of input videos. We first provide performance statistics and then discuss our main findings in detail.

5.1 Performance Statistics

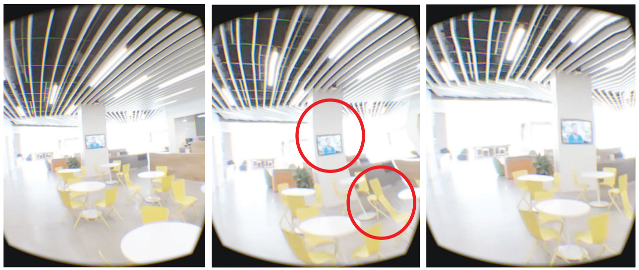
We have run our experiments on a computer with a Intel Xeon E5-2640 CPU and an NVIDIA Titan X graphics card. All the videos are captured with a single Samsung Gear 360 camera with 4K resolution at 30 fps.

The SfM and dense reconstruction steps of our approach are implemented on a CPU and run in an offline pre-processing stage. For a video sequence of 240 frames, the SfM and dense reconstruction steps take 28 and 114 seconds respectively. While the sparse scene reconstruction contains 576 points, the dense reconstruction results in a point cloud with 307k points.

During playback of the video on a VR headset, the warping algorithm runs in realtime on the GPU. The viewpoint corresponding to each eye as the head moves is tracked and provided by the VR headset API. We then warp the current video frame for each eye in parallel to achieve the stereoscopic experience. The computational efficiency of the warping step mainly depends on the resolution of the control mesh utilized to guide the warping. While a finer control mesh results in more accurate results, it incurs more computational cost. In Table 1, we provide timing statistics obtained with control meshes of different resolution for an input video with a 3D point cloud of 307k points. We compare the visual quality of the warping results corresponding to different control mesh resolutions in Figure 6. For the remaining of our experiments, we use a fixed resolution control mesh with 10240 vertices and 20480 triangles.

Table 1: We show the timings of the image warping algorithm with control meshes of different resolutions. When warping only the visible region as opposed to the entire panorama image, we gain nearly 35% computational efficiency. Even when using a high resolution control mesh, we achieve frame rates higher than 120 fps.

# triangles	5120	20480	81920
entire image	6.892 ms	7.156 ms	10.554 ms
visible region	4.573 ms	4.813 ms	7.226 ms



original frame

low-res control mesh (80 triangles)

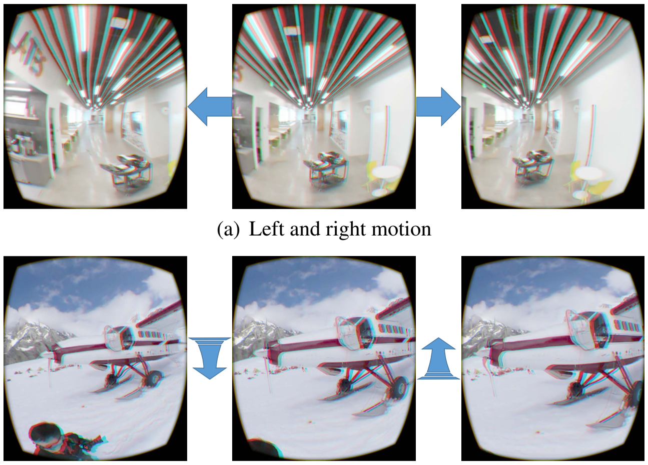
high-res control mesh (5120 triangles)

Figure 6: We show warping results generated by utilizing control meshes of different resolution. Notice that how a low-resolution control mesh creates significant distortions on the column.

5.2 Results

We illustrate some representative warping results in Figure 7 and refer the reader to the supplementary video for a complete set of results.

An important parameter affecting the quality of the novel view synthesis results is the parameter λ which denotes the relative



(a) Left and right motion

(b) Forward and backward motion

Figure 7: Anaglyph images of the same frame warped with different camera motion. Note that 3D geometry guided warping generates parallax effect.

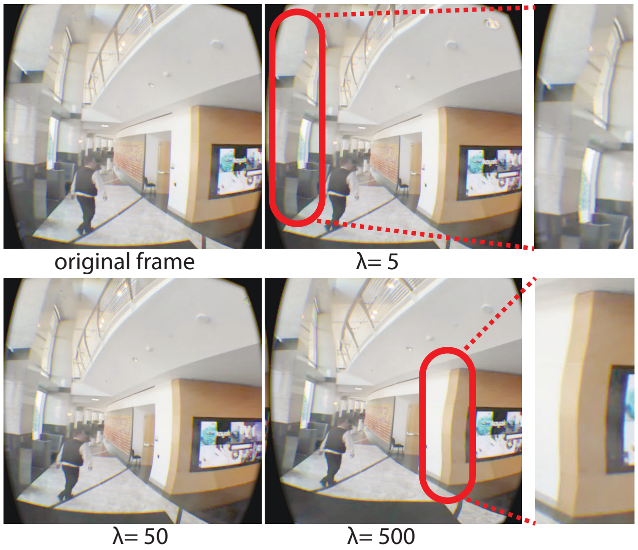


Figure 8: We show warping results with a fixed dense scene reconstruction but varying regularization weight (λ in Equation 4). While a small λ leads to local distortions (wavy patterns on the wall), a large λ results in global distortions (curvy brown edge). In our experiments we use $\lambda = 50$ which provides a good trade-off.

weighting between the data and regularization terms of the warping field optimization. In Figure 8, we demonstrate novel views synthesized by different choices of λ where an increase in λ results in a smoother warping field. While it is possible to adjust this parameter based on the quality of the 3D reconstruction of the scene (e.g., a higher λ can be preferred for noisy reconstructions), we find that setting $\lambda = 50$ provides a good tradeoff between accuracy and visual smoothness.

Comparisons. One of our main contributions in this work is to define the warping field directly on the unit sphere of a 360-camera. In comparison, all other previous works define the warping energy in the image space often with a simple rectangular grid. While Kamali et al. [13] propose to use a triangular grid sampled on the unit sphere, they still measure distortions in the image space. Due to the non-linearity of the mapping from the unit sphere to the image space, this leads to piecewise linear artifacts as seen in Figure 9. On the contrary, our approach avoids such distortions by computing the warp utilizing a conjugate vector field defined directly on the unit sphere.

In contrast to Kamali et al. [13] who propose to use a sparse scene representation for stabilizing omnidirectional videos, we utilize a dense reconstruction to guide the novel view synthesis. With a sparse reconstruction, many triangles of the control mesh do not receive any 3D point projection and thus solely rely on the regularization term which results in an over-smooth warping field. While this may be suitable for video stabilization purposes where relative camera motion is small, we observe that it leads to distortion artifacts for 6-DOF viewing experience as shown in Figure 10. In contrast, our warping algorithm can accurately capture the 3D details of the scene while still running at VR frame rates even when coupled with a high resolution control mesh (see Table 1).

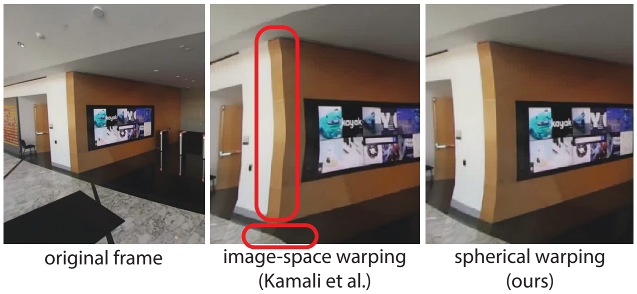


Figure 9: Even though Kamali et al. [13] utilize a triangular grid sampled on the unit sphere for warping, they still measure distortions on the image plane leading to artifacts. In contrast, by computing the warp field directly on the sphere, we avoid such distortions.

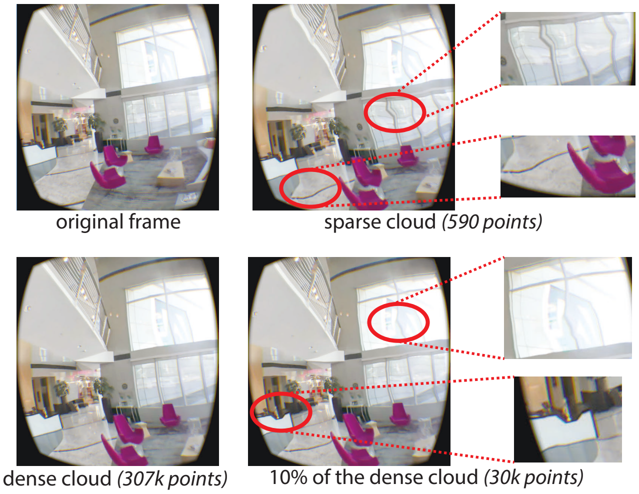
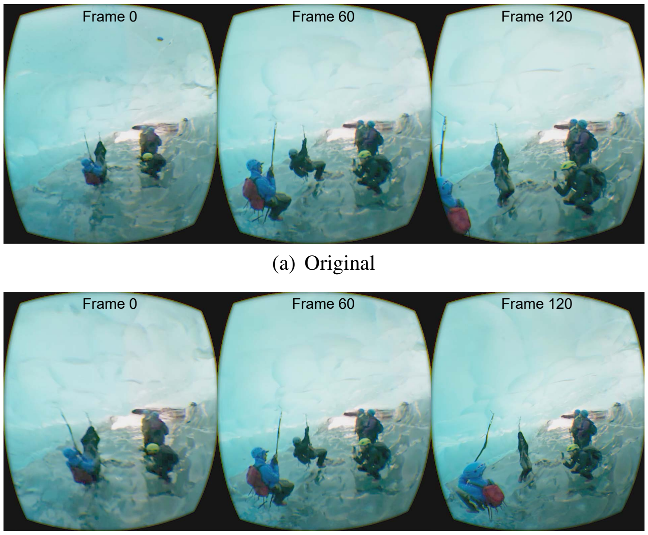

original frame
sparse cloud (590 points)
dense cloud (307k points)
10% of the dense cloud (30k points)

Figure 10: We show warping results generated by using the sparse point cloud obtained by the SfM step, the uniformly down-sampled (10%) dense point cloud, and the entire dense point cloud. Please note that the distortion artifacts (in red) occurring when using a sparse scene reconstruction are eliminated when the dense scene geometry is utilized. Note that, to ensure equal relative weighting between the data and the regularization terms in Equation 4, we adjust λ based on the number of points in the utilized scene reconstruction (λ = 50 * (# points in sparse reconstruction) / (# points in dense reconstruction)).

Applications. While we focus on 6-DOF playback of monoscopic 360-videos, our approach can easily be applicable to video stabilization. Once the original camera path is recovered, it can be smoothed to remove the shaking motion and novel views along this new path can be synthesized. We refer the reader to the supplementary video for such examples. Moreover, since we support 6-DOF warping, it is also possible to stabilize different components of the camera motion independently. For example, vertical oscillations can be removed while keeping off-vertical-axis motions if desired or translation of the camera can be re-synthesized with constant speed to remove accelerations that can lead to motion sickness.

Finally, many 360-videos are captured by a 360-camera carried by a moving person or vehicle. The playback of such videos may lead to strong motion sickness even after stabilization. An alternative to continuously updating the camera location during playback is to *teleport* between a set of fixed camera locations sampled along the camera path. Our approach can be utilized to synthesize novel views from these fixed camera locations as shown in Figure 11 and the supplementary video. For these examples, given the input video, we manually select a set of fixed viewpoints along the original camera path. We then segment the input video so that each



(a) Original

(b) Fixed view

Figure 11: Playback with a fixed viewpoint. In the original video (a), the camera travels about 4 meters forward within the ice cave. In the warped video (b), the camera is fixed at mid-point of its original path. We can achieve the ”teleporting” effect by fixing cameras at selected locations along camera path.

frame in a particular segment can be re-rendered from the fixed viewpoint closest in time with our warping method. We ensure the segments overlap (we use an overlap period of 0.2 seconds in our experiments) to enable smooth transitions between the fixed viewpoints (see Figure 12). Specifically, for each frame in the overlap region associated with two fixed viewpoints, we perform linear blending between the two warped views. We note that while we utilize a simple heuristic to define the fixed viewpoints and segment the input video, more advanced schemes taking into consideration the distance between the original and new viewpoints and the resulting warping error can be adapted.

User feedback. We invited a group of users to try our system where we asked them to view the same 360 video both in its stabilized monoscopic form and our 6-DOF stereoscopic version. The group consisted of 5 users who only had limited experience with mobile VR. The study was conducted with an Oculus Rift head mounted display while users sitting on a chair. The users were told that their motion did not need to be limited to the rotation of the head. We observed that with our 6-DOF system, users constantly moved around to explore the scene whereas with a monoscopic video they limited their movement to head rotation only after realizing the video did not respond to translational motion. The most typical translational movement for 6-DOF display was leaning forward to look closer at some objects. The users reported that doing the same with monoscopic view caused discomfort. All the users reported less motion sickness with our system and stated that they prefer the 6-DOF viewing experience because of the extra freedom of movement. However, users also reported that moving far away from their initial position while standing up resulted in discomfort due to large distortions in view warping. Overall, our limited and informal user study shows that the 6-DOF stereoscopic experience is more engaging and causes less motion sickness if user motion is limited within a space about the size of a desk.

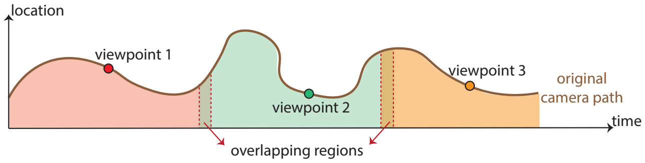


Figure 12: Teleporting: Given an input video, we manually choose a set of fixed viewpoints along the original camera path and form overlapping segments of the video. Each frame in a particular segment is re-rendered from the viewpoint closest in time while for frames in the overlapping regions we perform linear blending between two warped views to ensure smooth transition between viewpoints.

Limitations. Even though we have demonstrated our method on various examples, evidently there are certain limitations. The quality of the synthesized novel views heavily depends on the quality of the 3D reconstruction and how far the novel view is from the original input view. During warping, multiple 3D points contribute to the deformation of each vertex in the control mesh. Averaging such contributions together with the regularization term (Equation 4) helps to avoid artifacts due to the noise in the reconstructed point cloud as demonstrated in our examples of indoor and outdoor scenes with many planar and linear structures. However, in case of severe noise and holes (e.g., due to very large textureless regions, occlusions, illumination changes), artifacts will be apparent when warping from an original viewpoint to a novel viewpoint far away. In our experiments we observe that we obtain visually plausible results as long as the difference between the original and novel viewpoints are approximately below 1 meter. Finally, our current reconstruction algorithms assume the scene is static. Thus, dynamic subjects such as moving people are perceived to be moving on the static geometry.

6 Conclusion

We have presented an approach that enables the playback of monoscopic 360-videos in VR headsets with 6-DOF and in a stereoscopic manner. Specifically, we have enhanced standard structure-from-motion (SfM) and dense reconstruction algorithms to work with 360-videos. We have presented an image warping algorithm that computes a warping field directly on the unit sphere to minimize distortion artifacts. Finally, we have optimized our implementation for GPUs to achieve VR frame rates (>120 fps) on mainstream graphics cards.

While our system provides the first full 6-DOF stereoscopic playback of 360-videos, there is still room for future improvements. To begin with, we assume the input videos can perfectly be mapped to a unit sphere. In reality, however, distortions due to lenses and manufacturing inaccuracies cannot be avoided. Our future work includes enhancing the SfM algorithm to account for such distortion parameters. Moreover, we assume the input video contains translational motion of the camera to provide sufficient baseline for accurate 3D reconstruction. Extending recent approaches that handle rotation-only camera motion [28] to work with 360-videos is a promising direction. Finally, while dynamic scene reconstruction is still an active research area, billboard-type representations commonly used for sports broadcasting [8] are worth exploring.

References

[1] C. Buehler, M. Bosse, and L. McMillan. Non-metric image-based rendering for video stabilization. In *IEEE CVPR*, volume 2, 2001.

[2] D. Caruso, J. Engel, and D. Cremers. Large-scale direct slam for omnidirectional cameras. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 141–148. IEEE, 2015.

[3] P. Chang and M. Hebert . Omni-directional structure from motion. In *Proc. of IEEE Workshop on Omnidirectional Vision*, pages 127 – 133, June 2000.

[4] G. Chaurasia, S. Duchêne, O. Sorkine-Hornung, and G. Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM TOG*, 32, 2013.

[5] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan. High-quality streamable free-viewpoint video. In *ACM SIGGRAPH*, pages 69:1–69:13, 2015.

[6] O. Diamanti, A. Vaxman, D. Panozzo, and O. Sorkine-Hornung. Designing n-polyvector fields with complex polynomials. In *Computer Graphics Forum*, volume 33, pages 1–11. Wiley Online Library, 2014.

[7] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE PAMI*, 32(8):1362–1376, Aug. 2010.

[8] M. Germann, A. Hornung, R. Keiser, R. Ziegler, S. Würmlin, and M. Gross. Articulated billboards for video-based rendering. In *CGF Eurographics*, 2010.

[9] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. In *IEEE ICCV*, 2007.

[10] C. Häne, L. Heng, G. H. Lee, A. Sizov, and M. Pollefeys. Real-time direct dense matching on fisheye images using plane-sweeping stereo. In *Int. Conf. 3D Vision*, pages 57–64, 2014.

[11] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[12] S. Im, H. Ha, F. Rameau, and H. Jeon. All-around depth from small motion with a spherical panoramic camera. In *ECCV*, 2016.

[13] M. Kamali, A. Banno, J.-C. Bazin, I. S. Kweon, and K. Ikeuchi. Stabilizing omnidirectional videos using 3d structure and spherical image warping. In *Conf. Machine Vis. App.*, 2011.

[14] T. Kanade, P. Rander, and P. J. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE MultiMedia*, 4(1):34–47, Jan. 1997.

[15] S. Li. Binocular spherical stereo. *IEEE Trans. on Intelligent Transportation Systems*, 9(4):589–600, 2008.

[16] F. Liu, M. Gleicher, H. Jin, and A. Agarwala. Content-preserving warps for 3d video stabilization. In *ACM SIGGRAPH*, pages 44:1–44:9, 2009.

[17] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala. Subspace video stabilization. *ACM TOG*, 30(1):4:1–4:10, Feb. 2011.

[18] F. Liu, Y. Niu, and H. Jin. Joint subspace stabilization for stereoscopic video. In *IEEE ICCV*, 2013.

[19] Y. Liu, Q. Dai, and W. Xu. A point-cloud based multiview stereo algorithm for free-viewpoint video. *IEEE TVCG*, 2010.

[20] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Int. Joint Conf. on Artificial Intelligence - Volume 2*, pages 674–679, 1981.

[21] B. Micusik and T. Pajdla. Structure from motion with wide circular field of view cameras. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(7):1135–1149, July 2006.

[22] C. Morimoto and R. Chellappa. Evaluation of image stabilization algorithms. In *ICASSP*, volume 2, 1998.

[23] D. Scaramuzza, A. Martinelli, and R. Siegwart. A flexible technique for accurate omnidirectional camera calibration and structure from motion. In *IEEE ICVS*, 2006.

[24] M. Schönbein and A. Geiger. Omnidirectional 3d reconstruction in augmented manhattan worlds. In *IEEE Int. Conf. Intelligent Robots and Systems*, pages 716–723, 2014.

[25] S. Shen. Accurate multiple view 3d reconstruction using patch-based stereo for large-scale scenes. *IEEE transactions on image processing*, 22(5):1901–1914, 2013.

[26] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *ACM SIGGRAPH*, pages 835–846, 2006.

[27] C. Tomasi and T. Kanade. Detection and tracking of point features. *IJCV*, 1991.

[28] J. Ventura. Structure from motion on a sphere. In *ECCV*, 2016.

[29] S. Yaguchi and H. Saito. Improving quality of free-viewpoint image by mesh based 3d shape deformation. In *WSCG*, pages 57–64, 2006.

[30] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. In *ACM SIGGRAPH*, pages 600–608, 2004.