Joint Stabilization and Direction of 360° Videos

CHENGZHOU TANG, Simon Fraser University
OLIVER WANG, Adobe Systems Inc
FENG LIU, Portland State University
PING TAN, Simon Fraser University

360° video provides an immersive experience for viewers, allowing them to freely explore the world by turning their head. However, creating high-quality 360° video content can be challenging, as viewers may miss important events by looking in the wrong direction, or they may see things that ruin the immersion, such as stitching artifacts and the film crew. We take advantage of the fact that not all directions are equally likely to be observed; most viewers are more likely to see content located at "true north", i.e. in front of them, due to ergonomic constraints. We therefore propose 360° video direction, where the video is jointly optimized to orient important events to the front of the viewer and visual clutter behind them, while producing smooth camera motion. Unlike traditional video, viewers can still explore the space as desired, but with the knowledge that the most important content is likely to be in front of them. Constraints can be user guided, either added directly on the equirectangular projection or by recording "guidance" viewing directions while watching the video in a VR headset, or automatically computed, such as via visual saliency or forward motion direction. To accomplish this, we propose a new motion estimation technique specifically designed for 360° video which outperforms the commonly used 5-point algorithm on wide angle video. We additionally formulate the direction problem as an optimization where a novel parametrization of spherical warping allows us to correct for some degree of parallax effects. We compare our approach to recent methods that address stabilization-only and converting 360° video to narrow field-of-view video. Our pipeline can also enable the viewing of wide angle non-360° footage in a spherical 360° space, giving an immersive "virtual cinema" experience for a wide range of existing content filmed with first-person cameras.

CCS Concepts: • **Computing methodologies → Image manipulation**; *Computational photography*;

Additional Key Words and Phrases: VR, 360 video, re-cinematography, video stabilization

ACM Reference Format:
Chengzhou Tang, Oliver Wang, Feng Liu, and Ping Tan. 2018. Joint Stabilization and Direction of 360° Videos. *ACM Trans. Graph.* 1, 1, Article 1 (January 2018), 13 pages.
https://doi.org/0000000.0000000_0

1 INTRODUCTION

VR headsets are rapidly gaining in popularity, and one of the most common use cases is viewing 360° videos, which provide added immersion due to the ability of the viewer to explore a wider field of view than traditional videos. However, this freedom introduces a number of challenges for content creators and viewers alike; viewers can miss important events by looking in the wrong direction, or they can see things that break immersion, such as stitching artifacts or the camera crew.

In this work, we address two important aspects of 360 video creation; direction of shots to draw the viewers' attention to desired regions, and smooth, intentional camera trajectories. Both of these parts are crucial in 360° video; viewer freedom makes direction challenging, and unstable motion (especially in the peripheral vision) can be disorienting, causing confusion and even nausea [Jerald 2016]. While traditional cinematography refers to the decisions made *during* filming, 360° video is particularly well suited to the task of modifying the camera direction in post (also known as re-cinematography [Gleicher and Liu 2008]), as all viewing directions are recorded at capture time, giving us greater control in post.

We therefore propose a method that uses direction constraints [Gandhi et al. 2014] (e.g., where to look, where not to look), that try to keep desirable content in the viewable space near true-north, and undesirable content largely behind the user. These are jointly optimized with smoothness constraints that reduce camera shake and rapid rotations, such as those caused by hand held cameras, motorized gimbals, or inconsistent direction constraints. We allow an editor to manually define desirable and undesirable regions in the video, as well as the ability to use automatically derived constraints such as saliency maps, forward motion, or stitching regions for known camera configurations. In the case of manual constraints, editors can either directly draw on the equirectangular projection, or alternately we propose a new type of interaction where the editor views the content in a VR headset as a "guide", and their viewing path is recorded and used as a constraint in the joint optimization.

In summary, we propose a solution for joint stabilization and direction of 360° videos, where undesirable camera motions (e.g., shake and rapid rotations) are removed while following a smooth and directed camera path. Our solution also works for wide-angle videos, enabling "virtual cinema" viewing in VR for a large library of existing footage. To achieve these goals, we present the following technical contributions:

- A motion estimation algorithm based on non-linear optimization which performs better than widely used five-point algorithm on 360° and wide-angle videos.
- A 3D spherical warping model derived from our motion estimation that and handles both rotation and translation which allows more control than the recently proposed method [Kopf 2016].
- A unified framework to define and add constraints from different sources on the resulting 360° video, including a new VR interface and automatic motion constraints.

To validate these contributions, we make both qualitative and quantitative comparisons and conduct user study to show that in our

ACM acknowledges that this contribution was authored or co-authored by an employee, contractor, or affiliate of the United States government. As such, the United States government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for government purposes only.
© 2018 Association for Computing Machinery.
0730-0301/2018/1-ART1 $15.00
https://doi.org/0000000.0000000_0

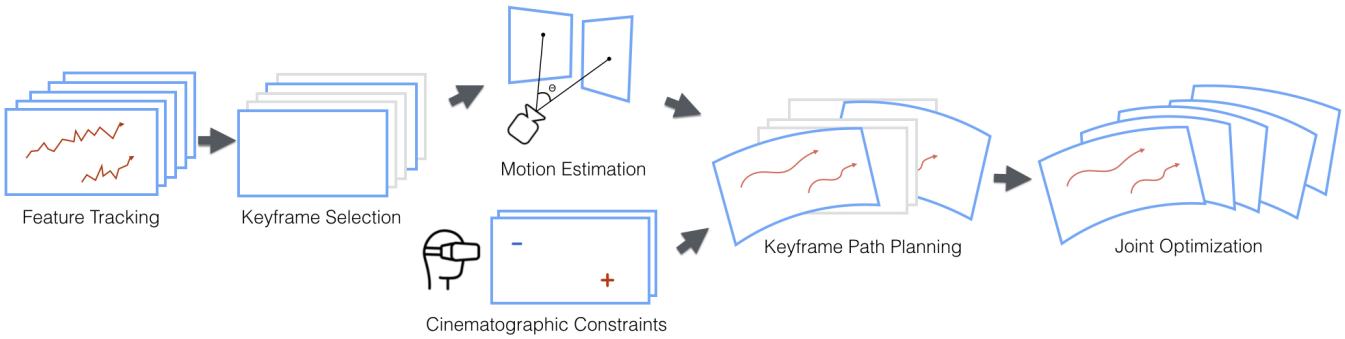


Fig. 1. Our method follows the above pipeline; features are tracked across all frames, after which keyframes are selected. A novel 3D camera motion estimation is combined with user-guided viewing constraints in the path planning step to produce a set of aligned keyframes, and the results are finally interpolated onto all frames of the video. 360° frames are visualized as rectangles.

directed results, viewers are much more likely to observe the desired parts of the sequence marked with positive constraints.

2 RELATED WORK

360° video can now be captured by both consumer handheld cameras, such as Ricoh Theta, Nikon KeyMission, and Kodak Pixpro SP360, and professional camera array systems [Anderson et al. 2016; Lee et al. 2016]. The captured 360° video often needs to be post-processed to deliver a pleasant viewing experience. We briefly review relevant prior research.

Video stabilization is a common approach to improve the camera motion of a video. Common ways to stabilize video involve tracking features over time and computing an image warp sequence that smooths feature trajectories and thus the apparent motion of the video. This can be achieved by applying a sequence of homographies or other 2D transformations that compensate the motion [Chen et al. 2008; Lee et al. 2009; Matsushita et al. 2006], by using a grid of homographies [Liu et al. 2013] for robustness, by using projective reconstruction [Goldstein and Fattal 2012], on a low dimensional feature subspace [Liu et al. 2011], as used in Adobe After Effects, or by fitting smooth cropping windows that minimize first and second order motion [Grundmann et al. 2011], as used on YouTube. Alternate approaches have proposed building a full 3D reconstruction of the scene, which can be used to synthesize a smooth virtual camera trajectory [Buehler et al. 2001; Kopf et al. 2014; Liu et al. 2009].

Recently, Kopf [2016] presented an extension of video stabilization to 360° videos. This approach computes a 3D geometric relationship between keyframes via the 5-point algorithm [Li and Hartley 2006], and smoothly interpolates keyframes using a deformable model. We use a similar approach with a few significant modifications. Whereas Kopf [2016] is largely used to compute a "total" stabilization (where fixed scene points will remain stationary throughout the video), we go beyond stabilization and combine artistic and *smoothness* constraints to produce an easy to watch 360° video with directed camera motion. One requirement to support this goal is that we need a full 3D motion rotation and translation estimation per frame. To achieve this, we introduce a new method for estimating rotation and translation on a sphere that is more robust than the 5-point algorithm.

Our work is inspired by previous work on re-cinematography [Gandhi et al. 2014; Gleicher and Liu 2008], where casually captured video is improved by integrating high-level content driven constraints and low-level camera motion stabilization. We extend this notion to 360° video, which benefits from all viewing angles being captured during filming, so cropping is not necessary, freeing our method from the trade off between guiding the viewer's attention and preserving video content. Similar path planning in traditional video has also been used for retargeting [Jain et al. 2015; Wang et al. 2009], which distorts or crops off less important content to fit the video into a different aspect ratio other than originally intended.

The Pano2Vid work by Su *et al.* [2016] performs a related, but different task of automatically producing a narrow field-of-view video from a 360° video. A recent follow-up work extended this to optimize for zoom as well [Su and Grauman 2017]. These approaches are complementary to ours; in our work, we combine viewing constraints with motion estimation to compute smooth directed camera paths for 360° viewing. Pano2Vid presents a learning based saliency computation, and computes a shortest path on these values. As it does not perform any motion estimation, it cannot be used to stabilize camera motion in shaky videos. We show that we can use the output from Pano2Vid as automatic saliency constraints for our method, which generates smooth camera paths where important content is placed in front of the viewers.

One of our main assumptions is that by rotating important objects in front of viewers, VR viewing becomes a more enjoyable experience. This is in some sense, reducing some control of the viewer to freely explore the space, as some rotations will be out of the control of the viewer. Whether this kind of motion is "allowable" is an open topic in VR film making, but we note that traditional wide angle video also started with static shots before filmmakers learned how to use camera motion as an artistic tool without disorienting the viewer/

In this domain, work by Sun et al. [2016] has shown that it is in fact possible to separate the viewer's real world head motion from the perceived motion in the VR space in order to map virtual

spaces into physical constraints such as room sizes without causing confusion. Sitzmann et al. [2018] also conducted user studies and concluded that the faster user attention gets directed towards the salient regions, the more concentrated their attention is.

3 METHOD

Figure 1 provides an overview of the approach. We first estimate the existing motion between keyframes using feature tracking and a novel pairwise motion estimation formulation for spherical and wide angle video. We then solve a joint optimization on the keyframes that enforces smoothness in the warped video sequence and a set of user-provided (or automatic) path planning constraints. Finally, we smoothly interpolate the motion between keyframes to produce the final output video. We now discuss each step in more detail.

3.1 Feature Tracking and Keyframe Selection

Similar to prior work [Kopf 2016], for 360° videos we remap the equirectangular image to a cube map and track feature points independently on each cube face using KLT feature tracking [Shi and Tomasi 1994]. For wide-angle videos, we perform the tracking directly on the video frames. This is the only stage of the process that is different between 360° and traditional wide-angle video, after tracking we project feature points onto a sphere and treat both types of videos identically. In the following, we will use unit vector $\mathbf{p}$ to denote the projected points on a sphere.

Similar to other approaches, we select feature points on keyframes and track them through the video [Kopf 2016; Kopf et al. 2014]. The first and last frames are selected as keyframes, and we create new keyframes every time the percentage of successful tracked features drops to 60% of the number of features initially detected. Finally, we only select feature points that are more than 2° away from any previously selected feature points.

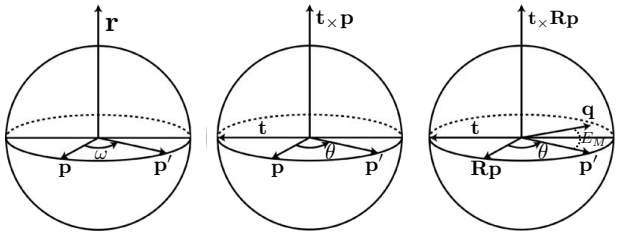
After feature tracking, we have a set of m feature trajectories $T = \{T_i | i = 1 \cdots m\}$ through the video, where each trajectory T_i is a list of points from several continuous frames:

$$T_i = \{\mathbf{p}_j^i | j = s_i \cdots e_i\}, \tag{1}$$

where s_i is the starting frame (which is always a keyframe), and e_i is the last keyframe where the point was successfully tracked.

3.2 Rotation and Translation Estimation

After collecting feature tracks, we estimate the relative 3D rotation and translation between neighboring pairs of keyframes. A common solution is to use the 5-point algorithm [Nister 2004], to first estimate the essential matrix, decompose it into a rotation matrix $\mathbf{R}$ and a translation direction vector $\mathbf{t}$, and then improve the estimated motion by iterative refinement [Triggs et al. 2000]. With this approach, the final quality relies on the accuracy of the essential matrix estimation as well as the motion decomposition. It is well known both of these steps are highly dependent on the quality of the camera calibration [Stewenius et al. 2005], feature trajectories, and global shutter camera [Dai et al. 2016]. As a result, prior 360° stabilization work only uses the estimated rotation between neighboring keyframe pairs, and discards the 3D translation, which can be unreliable. The final stabilization is then performed by smoothing feature trajectories directly on the spherical image [Kopf 2016].



(a) Rotation (b) Translation (c) Combined

Fig. 2. The motion of a point ($\mathbf{p}$ to $\mathbf{p}'$) on a sphere, when (a) induced by camera rotation around the axis $\mathbf{r}$, (b) induced by a camera translation in direction $\mathbf{t}$, and (c), both together. We estimate $\mathbf{t}$, $\mathbf{R}$, and per-feature θs by minimizing distances E_M for all tracked feature locations $\mathbf{q}$

We propose a new motion estimation method that is robust to poorly calibrated cameras, rolling shutter effects, and errors in the feature trajectories, and yields 3D rotation and translation direction estimates which are accurate enough to be directly used to stabilize the footage, as well as to provide direction such as a forward motion constraint (discussed later in Sec. 3.3). Both the 5-point algorithm and our method are two view motion estimation methods, where translation can only be computed up to an unknown scale. Therefore, we use the standard definition for translation $\mathbf{t}$ as a 3×1 unit vector [Hartley and Zisserman 2004], representing the translation *direction*.

Direct Spherical Motion Estimation We first consider the relationship between feature correspondences and the 3D transformation between two frames, a reference frame (with identity rotation and zero translation), and second frame, with rotation $\mathbf{R}$ and translation $\mathbf{t}$. Figure 2 illustrates the motion of a feature correspondence $\mathbf{p} \rightarrow \mathbf{p}'$ as a function of both 3D rotation $\mathbf{R}$ and translation $\mathbf{t}$. To gain a better understanding let's consider the pure rotational and pure translation cases independently. We use Rodrigues' formula [Gray 1980] to represent rotation:

$$\mathbb{R}(\mathbf{r}, \omega) = \mathbf{I} + \sin(\omega)\mathbf{r}_\times + (1 - \cos(\omega))\mathbf{r} \otimes \mathbf{r}, \tag{2}$$

where $\mathbf{r}$ is the rotation axis, ω is the rotation angle, $\mathbf{r}_\times$ is the cross product matrix of $\mathbf{r}$ and $\otimes$ is the tensor product.

When the camera undergoes a rotation, every feature $\mathbf{p}$ on the sphere rotates around the same axis $\mathbf{r}$ with the same angle ω independent of its depth, i.e. $\mathbf{p}' = \mathbb{R}(\mathbf{r}, \omega)\mathbf{p}$. Since every point rotates around the same axis by the same angle we simplify $\mathbb{R}(\mathbf{r}, \omega)$ as a global rotation $\mathbf{R}$.

When translating, each $\mathbf{p}$ rotates around a *feature dependent* axis $\mathbf{t}_\times\mathbf{p}$ by an angle θ (which is a function of point depth), towards the intersection of $\mathbf{t}$ and the sphere, i.e. $\mathbf{p}' = \mathbb{R}(\mathbf{t}_\times\mathbf{p}, \theta)\mathbf{p}$, where $\mathbf{p}$, $\mathbf{t}$ and $\mathbf{p}'$ are in the same plane, and $\mathbf{t}_\times\mathbf{p}$ is the normal vector of this plane, i.e. $(\mathbf{p}')^\top \mathbf{t}_\times \mathbf{p} = 0$.

Without loss of generality, we define the camera motion $M = [\mathbf{R}, \mathbf{t}]$ as a rotation $\mathbf{R}$ followed by a translation $\mathbf{t}$. To account for

noise in the feature tracks, we estimate the rotation $\mathbf{R}$, translation $\mathbf{t}$ and the feature dependent angle θ by minimizing the following error where $\mathbf{q}$ is the tracked feature location:

$$E_M(\mathbf{R}, \mathbf{t}, \theta) = \|\mathbf{q} - \mathbf{p'}\|_2 = \|\mathbf{q} - \mathbb{R}(\mathbf{t}_\times \mathbf{R}\mathbf{p}, \theta)\mathbf{R}\mathbf{p}\|_2 \quad (3)$$

over all feature correspondences between two keyframes. This energy measures the Euclidean distance between $\mathbf{q}$ and $\mathbf{p'}$ where $\mathbf{p'}$ is the point found by rotating $\mathbf{p}$ with $\mathbf{R}$ and then with $\mathbb{R}(\mathbf{t}_\times \mathbf{R}\mathbf{p}, \theta)$. We observe that the energy in Eq. 3 is hard to optimize directly because of the point-dependent angle θ. In practice, there are about 3000 feature correspondences between two frames and each correspondence has an θ, which gives a large amount of unknowns.

Instead, we directly estimate the rotation and translation between two frames by first ignoring θ, and minimizing the least squares error:

$$E_M(\mathbf{R}, \mathbf{t}) = \| \arcsin\left(\frac{\mathbf{q}^\top \mathbf{t}_\times \mathbf{R}\mathbf{p}}{\|\mathbf{t}_\times \mathbf{R}\mathbf{p}\|}\right)\|_2 \quad (4)$$

over all feature correspondences. Eq. 4 is a function of $\mathbf{R}$(three DoFs) and $\mathbf{t}$(three DoFs), which is much easier to solve then Eq. 3. As shown in Fig 2, Eq. 4 measures the angular distance between the tracked point $\mathbf{q}$ and the plane that contains $\mathbf{R}\mathbf{p}$ and $\mathbf{t}$, which is also the angular distance between $\mathbf{q}$ and $\mathbf{p'}$.

We can then compute θ by projecting $\mathbf{q}$ onto the same plane with $\mathbf{R}\mathbf{p}$ and $\mathbf{t}$, giving the projection as $\mathbf{p'}$, and then measure the angular distance between $\mathbf{p}$ and $\mathbf{p'}$ as θ. We found it to be unnecessary to constrain $\mathbf{t}$ to be a unit vector during the optimization, as Eq. 4 is already normalized, and the scale of $\mathbf{t}$ does not affect the value of E_M. We therefore only normalize $\mathbf{t}$ after the optimization is finished.

Before minimizing Eq. 4, we identify inlier feature correspondences using RANSAC with the fundamental matrix as a model, estimated by the 7-point algorithm [Hartley and Zisserman 2004]. The reason for not choosing the more often used 8-point algorithm is that the normalization in 8-point algorithm is inapplicable to spherical points, and the 7-point algorithm is more efficient in the presence of noise. While the fundamental matrix is unable to give us the rotation and translation directly, it is helpful to identify feature tracks inconsistent with possible camera motion, especially for scenes with moving objects.

We now can compute the relative motion $M_{i,i+1} = [\mathbf{R}_{i,i+1}, \mathbf{t}_{i,i+1}]$ between all neighboring pairs of keyframes $k_i \rightarrow k_{i+1}$ by minimizing Eq. 4 using a non-linear least squares solver. We then chain rotations to align each keyframe in a global coordinate frame (e.g., that of the first keyframe):

$$\mathbf{R}_{k_i} = \prod_{j=1}^{i-1} \mathbf{R}_{k_j, k_{j+1}}. \quad (5)$$

We use the relative translation direction $\mathbf{t}_{k_i, k_{i+1}}$ to compute the final warped frames and for the forward motion constraint.

We then compute the relative camera motion for all remaining frames between a given pair of keyframes k_i and k_{i+1}. To do this, we set the neighboring keyframes to be the reference frame, and separately solve for the in-between frame by minimizing Eq. (4), averaging the result from both nearest keyframes (previous and next). We evaluate the results both qualitatively and quantitatively in Sec 4.

3.3 Joint Stabilization and Direction

Now that we have estimated the input camera path, we are able to define our joint stabilization and direction optimization, which consists of two terms:

$$E(W) = E_d(W) + E_s(W), \quad (6)$$

where E_d is an energy that captures directional constraints and E_s encourages the smoothness in the resulting spherical video. W is a set of camera transformations that correspond to the optimal (virtual) camera trajectory. In this optimization, we use a combined rotation and translation model W to transform the input video, which we render using image warping.

We first solve Eq. 6 considering W restricted to *rotation only* and then later solve the two terms $E_d(W)$ and $E_s(W)$ considering models that handle both rotation and translation. In the following two sections, and implementation, we present rotation in quaternions.

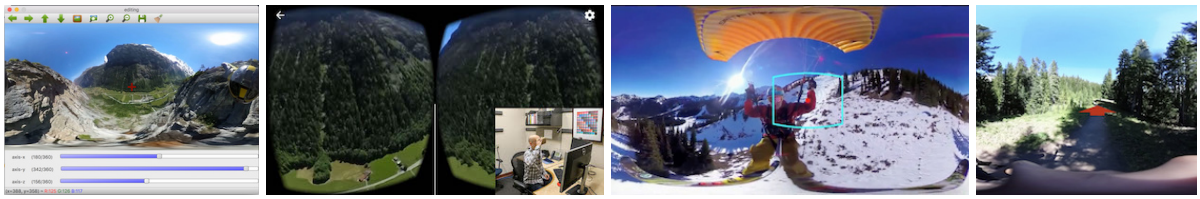
3.4 Directional Constraints

Directional constraints can be either positive, which specify salient events that the editor would like viewers to see, or negative, indicating things that *should not* be seen, for example content that is uninteresting, that contains elements that do not belong in the scene, such as the camera crew, or stitching seams. Fig. 3 shows the various types of constraints we use.

Source of Constraints We provide two types of user provided constraints. In one case, the editor simply clicks directly on the ER projection, while in the other, the editor provided a "guided" viewing session, where they watch the video in a VR headset, and their viewing direction is recorded over time, and used as positive constraints. To support this, we developed an app that records the users head rotation during the playback of a video in a Google Cardboard headset. This trajectory is then sampled at every second and used in our optimization to guide the look-at direction over the course of the video.

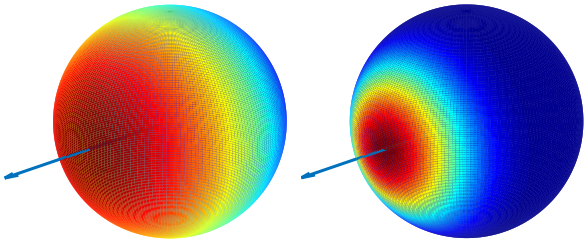
Our method can also be easily integrated with automatically generated constraints. Automatic saliency methods for 360° video (e.g., [Su et al. 2016]) determine the most visually salient regions, which can be directly interpreted as positive constraints. Alternately, as we are estimating 3D translation direction in Section 3.2, we can use this to keep the camera pointed roughly in the forward motion direction, which provides a comfortable "first-person" viewing experience. Automatic negative constraints can also be added, for example for seam locations if the camera geometry is known a priori. We show examples of all of these types of constraints in the result section and supplemental material.

Constraint Formulation A positive constraint is represented as a look-at point $\mathbf{p}_i$ located on a spherical video frame f_i. The goal is to transform the frame f_i by a rotation quaternion $\mathbf{q}_{f_i}^W$ such that $\mathbf{p}_i$ is as close to the true north direction as possible, thereby making it more likely to be seen. Similarly, a negative constraint $\mathbf{n}_j$ located on frame f_j is one that we want to avoid appearing in the front, i.e., we search for a transformation $\mathbf{q}_{f_j}^W$ to make this point appear outside of the user's likely viewing direction.



(a) (b) (c) (d)

Fig. 3. Our method can be used with a variety of input sources. Here we show some ways to provide directional constraints. A user can manually select regions on an ER projection (left), or create a guided viewing experience in a VR headset (b), or we can use automatic constraints such as saliency e.g., [Su et al. 2016] (c), or from forward motion (Sec. 3.2) (d).



L2 Loss Our Loss for Negative Constraints

Fig. 4. L2 loss vs. robust loss for negative constraints. We make the penalty close to zero as long as the negative constraint is out of visible region, and the penalty increases sharply when the negative constraint moves towards visible region. The forward direction is marked as blue arrows.

Therefore, for a set P of positive constraints, and N of negative constraints, the directional term can be expressed as:

$$E_d(W) = \sum_{i \in P} \|\mathbf{q}_{f_i}^W \cdot \mathbf{p}_i - \mathbf{f}\|_2^2 + \sum_{j \in N} \rho(\|\mathbf{q}_{f_j}^W \cdot \mathbf{n}_j - \mathbf{b}\|_2^2), \quad (7)$$

where $\mathbf{q}_{f_i}^W \cdot \mathbf{p}_i$ denotes rotating $\mathbf{p}_i$ by quaternion $\mathbf{q}_{f_i}^W$, and $\mathbf{q}_{f_i}^W$ minimizes the distance between $\mathbf{p}_i$ and the front vector $\mathbf{f} = [0, 0, 1]^\top$, and $\mathbf{q}_{f_j}^W$ maximizes the distance of $\mathbf{n}_j$ to the front vector $\mathbf{f}$, which is equivalent to minimizing the distance to the back vector $\mathbf{b} = [0, 0, -1]^\top$.

Because we may only want the negative constraint to be outside the of view (rather than exactly behind the user), we use a robust loss function $\rho(x) = \alpha e^{-\frac{\beta}{x}}$ on the negative constraint. We set $\alpha = 3200$ and $\beta = 26.73$, which causes $\rho(x)$ to yield a low cost until it enters a visible region for the average human field of view (which is roughly 114°horizontally [Strasburger et al. 2011]), after which the penalty increases sharply, as shown in Fig. 4.

3.5 Smoothness Constraints

From Sec. 3.2, we have computed the global rotation $\mathbf{R}_i$ for all frames, with respect to the first frame. We use a smoothness model similar to [Kopf 2016], but with small modifications. Importantly, we define the first-order and second-order smoothness terms directly on the estimated camera rotations instead of the feature trajectories, i.e.,

$$E_s(W) = \alpha_1 E_{s1}(W) + \alpha_2 E_{s2}(W), \quad (8)$$

where

$$E_{s1}(W) = \sum_{i=1}^{n-1} \left| \mathbf{q}_{i+1}^W \mathbf{q}_{i+1} - \mathbf{q}_i^W \mathbf{q}_i \right|_p \quad (9)$$

is the first order term and,

$$E_{s2}(W) = \sum_{i=1}^{n-2} \left| (\mathbf{q}_{i+2}^W \mathbf{q}_{i+2})^{-1} \mathbf{q}_{i+1}^W \mathbf{q}_{i+1} - (\mathbf{q}_{i+1}^W \mathbf{q}_{i+1})^{-1} \mathbf{q}_i^W \mathbf{q}_i \right|_p, \quad (10)$$

is the second order term.

This energy is summed over all n frames, and $|\mathbf{q}_a - \mathbf{q}_b|$ is the difference between two quaternion rotations, α_1 and α_2 are weights, and p is the norm of the smoothness that we solve for. In our implementation, we use $\alpha_1 = 10$ and $\alpha_2 = 100$ for all results shown, but these values could be changed to control the trade-off between the direction constraints and video smoothness.

3.6 Optimization

We can now directly minimize Eq. 6 using the Levenberg-Marquardt algorithm in Ceres and achieve a final rotation-only directed video. However, for efficiency, we propose the following optimization scheme which gradually propagates a good initialization, speeding up convergence.

We also observe that some axes of rotation can be confusing for the viewer. In particular, camera movement in the roll axis is uncommon in many videos recorded on a ground plane, and can be confusing. We therefore default to fixing the roll axis in ψ and allow for rotation only in the pitch and yaw of the camera, unless otherwise specified by the user (for example in the wing-suit video (Fig. 17), we enable rotation on the roll axis).

We can choose between smoothness norms in our optimization, in particular prior works have used L1 and L2 norms. Using an L1 norm tends to give clear distinctions between fixed and moving shots, while L2 paths are smoother overall. We provide examples of L1 and L2 smoothed paths in the supplemental material, and allow the user to choose which norm ($p = 1$ or $p = 2$) they want to minimize.

3.7 Translation-Aware Transformation

Until now, we have considered rotation-only transformation for W, which have the advantage of being quick to compute, as there are

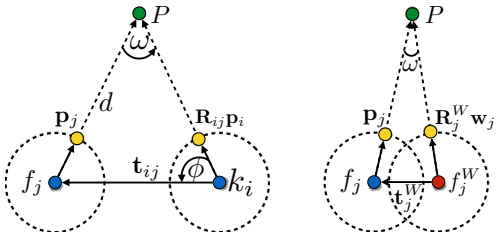


Fig. 5. Spherical Point Projection. Left: The point projection in the input video between a frame f_j and a keyframe k_i. Right: The point warping between an input frame f_j and a target frame f_j^W.

only three DoFs for each frame, and are guaranteed to not introduce any local distortion into the video. However, as observed in prior work [Kopf 2016], camera shake often contains a significant *translation* component to it, which requires local warping to compensate for parallax effects. One common solution is to smooth individual feature trajectories, however, this can break the spatial distance between feature trajectories on sphere and introduce geometric distortions in the output. Previous works have used Structure-from-Motion (SfM) or subspace constraints to generate smoothed feature trajectories while preserving scene structure, but these strategies are either computationally expensive [Liu et al. 2009] or cannot be directly applied to 360° videos [Liu et al. 2011]. For 360° videos, [Kopf 2016] represents feature points by an interpolation of six evenly distributed vertices on a sphere, and then constrains the rotation of the six vertices to be similar to avoid large distortions.

Different from all these methods, we propose to solve for a virtual camera, represented by the transformation W that includes rotation and translation, such that the feature trajectories as seen from this camera are smooth. This is possible due to the estimated per-frame 3D rotation and translation in Sec 3.2. The goal of our transformation is therefore to generate smooth feature trajectories that maintain the spatial structure and directional constraints for each frame. This approach has fewer degrees of freedom than directly optimizing for image-space warps, as it is restricted to geometrically plausible reconstructions, and we show that it can handle strong parallax better than [Kopf 2016]. At the same time, we do not require computational expensive SfM [Liu et al. 2009]. We first describe a **spherical point projection** model that represents transformed points as the function of our translation-aware transformation. We then present a modified version of the smoothness term in Eq. 6, that we minimize to find the transformed feature trajectories. Finally, we can warp the image using these transformed feature trajectories in Sec. 3.8.

Spherical Point Projection Given an inner frame f_j and a neighboring keyframe k_i, we know the relative motion $M_{ij} = [\mathbf{R}_{ij}, \mathbf{t}_{ij}]$ from Sec 3.2. As shown in Fig 5, we consider local coordinates centered at f_j, where the position of f_j is the origin, the position k_i is $-\mathbf{t}_{ij}$, and $\mathbf{P}$ is the 3D point corresponding to the spherical feature point $\mathbf{p}_j$ at a distance d from f_j. According to the sines rule of triangles, d can be computed as:

$$d = \frac{\sin(\phi)}{\sin(\omega)}\|\mathbf{t}_{ij}\| = \frac{\sin(\phi)}{\sin(\omega)}, \tag{11}$$

where ω is the angle between $\mathbf{p}_j$ and $\mathbf{R}_{ij}\mathbf{p}_i$ as described in Sec 3.2, ϕ is the angle between $\mathbf{p}_j$ and the unit translation direction $-\mathbf{t}_{ij}$. Therefore, we have

$$\mathbf{P} = d\mathbf{p}_j = \frac{\sin(\phi)}{\sin(\omega)}\mathbf{p}_j. \tag{12}$$

We now want to transform the point $\mathbf{P}$ onto a target spherical frame f_j^W by a camera transformation $W_j = [\mathbf{R}_j^W, \mathbf{t}_j^W]$. Since $\mathbf{P}$ is already known, the transformed point $\mathbf{w}_j$ can be represented as a function of the transformation $W_j = [\mathbf{R}_j^W, \mathbf{t}_j^W]$ as:

$$\mathbf{w}_j = (\mathbf{R}_j^W)^\top\left(\frac{\mathbf{P} - \mathbf{t}_j^W}{\|\mathbf{P} - \mathbf{t}_j^W\|}\right). \tag{13}$$

Dividing $\sin(\omega)$ in Eq. 11 and Eq. 12 makes $\mathbf{P}$ numerically unstable when $\sin(\omega)$ is close or equal to zero. This issue happens when the translation is near zero, or $\mathbf{P}$ is a far scene point. To avoid this problem, we scale both $\mathbf{P}$ and $\mathbf{t}_j^W$ in Eq. 13, by $\sin(\omega)$, which leads to the same result but one that is numerically more stable.

Translation-Aware Optimization We next introduce how to optimize for the final smoothed and directed camera transformations. By representing a transformed point $\mathbf{w}_j$ as the function of a rotation $\mathbf{R}_j^W$ and a translation $\mathbf{t}_j^W$ in Eq. 14, we can extend the rotation only transformation in Eq. 6 to a full rotation and translation transformation:

$$\mathbf{w}_j = (\mathbf{R}_j^W)^\top\left(\frac{\sin(\omega)(\mathbf{P}-\mathbf{t}_j^W)}{\|\sin(\omega)(\mathbf{P}-\mathbf{t}_j^W)\|}\right) = (\mathbf{R}_j^W)^\top\left(\frac{\sin(\phi)\mathbf{p}_j - \sin(\omega)\mathbf{t}_j^W}{\|\sin(\phi)\mathbf{p}_j - \sin(\omega)\mathbf{t}_j^W\|}\right). \tag{14}$$

The optimization for the joint rotation and translation W:

$$E_s(W) = \sum_{i=1}^{|T|}\left(\alpha_1\sum_{j=s_i}^{e_i-1}\|\mathbf{w}_j^i - \mathbf{w}_{j+1}^i\| + \alpha_2\sum_{j=s_i}^{e_i-2}\|\mathbf{w}_{j+2}^i - 2\mathbf{w}_{j+1}^i + \mathbf{w}_j^i\|\right) \tag{15}$$

which is an objective function over the unknown camera transformation, s_i and e_i are the starting and ending frame of the i-th feature trajectory as in Eq. 1, α_1 and α_2 are the same weight between first-order and second-order smoothness as in Eq. 8.

The final objective function combines the smoothness term Eq. 15 and the directional term, (Eq. 6). And minimizing this yields the transformation W that represents the collection of rotation and translation $W = \{[\mathbf{R}_i^W, \mathbf{t}_i^W]\}_{i=1\cdots n}$ that transform all the n frames to a smoother and better directed camera path. After W is estimated, we transform the points of feature trajectories by Eq. 14. In the example illustrated in Fig. 6, we can see that the transformed feature trajectories are much smoother than the original input, while satisfying directional constraints.

3.8 3D Spherical Mesh Warping

We now render these transformations by spherical mesh warping. For each frame f_j we have $\mathbf{p}$ as the original feature point in the input and $\mathbf{w}$ as the corresponding points in the transformed frame f_j^W. We ignore the superscript W and subscript j since we consider warping now only for single frames. Based on the point correspondences $\mathbf{p} \rightarrow \mathbf{w}$, we can re-render an input image as if the image was

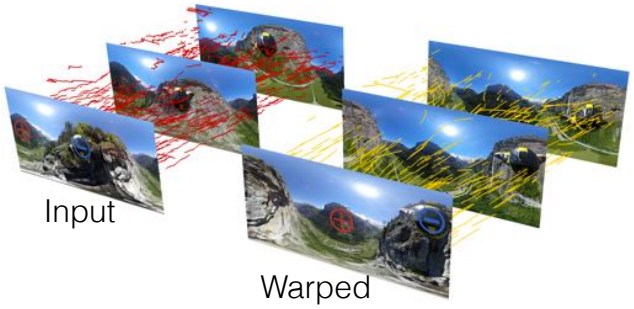


Fig. 6. Translation-Aware Optimization. The input trajectories (red) are shaky while the optimized trajectories (yellow) are smoother. At the same time, the positive constraint (red '+' circle) is transformed to the front, while the negative constraint (blue '−' circle) is transformed to the back of the sphere.

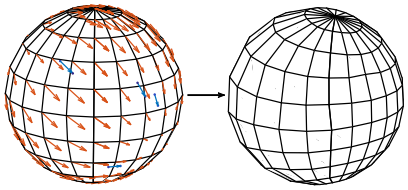


Fig. 7. Spherical Mesh Warping. A set of feature correspondences (blue arrows) are used to drive a warp computed over the vertices of a sphere. The computed offset vectors at each vertex (red arrows) are used to warp the vertices to the sphere on the right.

captured by a camera physically located on a well directed and smooth 3D camera path without requiring full 3D reconstruction, which can be slow and unreliable [Liu et al. 2009]. As shown in Fig 7, the general idea of our spherical mesh warping is to propagate the point-wise correspondence to each mesh vertex in the image and then warp the spherical image based on these vertices. Due to our 3D structure preservation, we can allow for a higher degree of deformation than prior work (we use a 20×10 mesh), while avoiding geometric distortion. Given the high mesh resolution, we found it to be sufficient to use a regular grid defined on an equirectangular map, however different spherical tessellations could be used if desired.

As described in Sec. 3.2, the translation induced motion can be represented by feature dependent axis and angle. Since we've already estimated the transformation rotation $\mathbf{R}$ and translation $\mathbf{t}$, we know the axis as $\mathbf{t}_\times\mathbf{R}\mathbf{p}$ for each feature point, and we get the angle ω in the same way as in Sec. 3.2.

We now have the rotation and translation between the input and warped image and have estimated the angle ω (which is a parametrization of the point depth) for each feature in the frame. To propagate the warping motion from a set of feature points F to a set of *mesh vertices* V in the corresponding spherical mesh warp, we apply the rotation $\mathbf{R}$ to all vertices and then solve for E_ω, the field of translation angles for all vertices by a 1D minimization

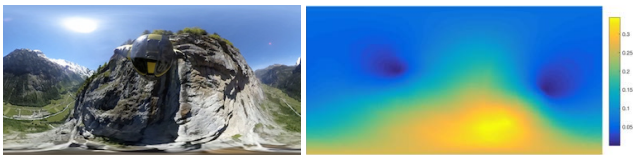


Fig. 8. A warped image and the interpolated angle ϕ shown for all pixels. This demonstrates how the warping angles have structure roughly similar to the inverse depth map but smooth, e.g., pixels closer to the camera on the mountain have larger value than farther pixels under the same camera motion.

parameterized *only* on the angle ω.

$$E_\omega = \sum_{p\in F} \|\omega_p - \mathbf{b}_p^\top \mathbf{v}_p\|_2^2 + \sum_{i,j\in V} \|v_i - v_j\|_2^2, \tag{16}$$

where p belongs to the set of feature points, i and $j \in V$ are neighboring vertices on sphere, $\mathbf{v}_p$ are the angles for the four vertices that cover the p-th feature point [Liu et al. 2009], and $\mathbf{b}_p^\top$ is the corresponding bilinear interpolation coefficients for the four vertices. The spatial smoothness term $\sum \|v_i - v_j\|_2^2$ guarantees that angles are spatially smooth and avoids local distortion.

We solve Eq. 16 in a least squares sense, giving us the final output position for each vertex and use these vertex positions to render a warped sphere (Fig. 7). The 1D optimization defined in Eq. 16 has the advantage of being efficient to compute, while restricting our solution to warps that are geometrically regularized. We bi-linearly interpolate the angle to all pixels and visualize it in Fig 8. Because the warping must only correct for residual motion, we observed, similar to prior work [Kopf 2016], that the warping function is largely smooth, and as such we have found this approach to be robust to input videos and parameter settings.

3.9 Implementation

Our method was implemented in C++. We solve Eq. 4 and Eq. 6 using the Levenberg-Marquardt algorithm in Ceres. We fixed parameters for all the results shown here, although if desired we can easily control the amount of smoothing by changing α_1 and α_2. We report average running times for the different parts of our method in Tab. 1 computed on a 2015 Macbook Pro laptop with 2.5GHz i7 CPU and 16GB memory, on HD (1920×960) equirectangular video frames.

Section	Average running time in ms/frame
Ingest and cube map conversion	5
Feature tracking and keyframe selection (Sec. 3.1)	23
Relative motion estimation (Eq. 4)	10
Re-cinematography computation (Eq. 6)	1
Full deformable warping (Eq. 16)	10
Rendering in OpenGL	6
Total	55

Table 1. Running times for different parts of our method, computed on 1920×960 video frames.

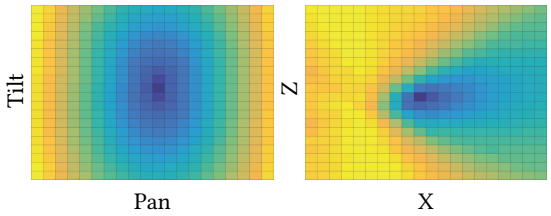


Fig. 9. Representative slices of the energy surface formed by Eq. 4, with blue showing areas of lower energy. Here we show 2D slices of the original 6D error space. This visualization indicates that the surface is largely smooth, which yields a robust and efficient optimization, despite the nonlinear energy terms.

4 RESULTS AND EVALUATION

4.1 Motion Estimation Evaluation

We first compare our motion estimation to the commonly used 5-point algorithm [Nister 2004]. As opposed to the 5-point algorithm which estimates an essential matrix, we directly solve for **R** and **t** using non-linear least squares. We solve this minimization using Levenberg-Marquardt optimization [Agarwal et al. 2017] with manually derived analytic derivatives which we show in the Appendix A. We empirically found this to converge by 10 iterations, taking roughly 20ms to estimate the motion for a pair of cameras with about 3000 feature correspondences. Surprisingly, the optimization converges to a reasonable solution even when initialized with an identity rotation matrix and a random unit translation vector. To understand why, we visualize the error space by uniform sampling (two slices of which are shown in Figure 9), which shows that the energy function is close to convex, making it efficient to solve robustly. We also found that our motion estimation works well even in cases with pure rotation, which is most likely due to the fact that our method uses all feature points available in a 360° image, which helps to disambiguate rotation and translation motion. Compared to the 5-point algorithm [Nister 2004], where epipolar constraint are enforced exactly (Eq. 2 equals to zero) within each 5-point group and the solution is selected as the one with maximum inliers, our method seeks for a solution with a minimum energy over all points, which benefits from a greater number of inliers, improving the accuracy of the motion estimation and mesh-based image warping.

We next evaluate the motion estimation quality in the presence of noise in the feature correspondences. A qualitative visualization of feature trajectories after stabilization using our method in place of the 5-point algorithm is shown in Fig. 11, with additional examples in Sec. 4. Quantitative evaluation on real world data is challenging due to the difficulty of collecting ground truth data for 3D pose estimation. We therefore employ the same validation technique used by prior motion estimation works [Kneip and Lynen 2013; Kneip et al. 2012; Nister 2004]. The approach is to use synthetic data, where the first camera is fixed at the origin with identity rotation and a second camera chosen at a random position at most τ units from the first with a relative rotation generated from random Euler angles bounded to κ° . We then create a uniformly distributed 3D point cloud with fixed maximum distance to the origin γ. Feature correspondences can then be computed by projecting the 3D points into the two spherical cameras with added noise. Finally, the relative motion of the cameras are computed from this data and compared to the known relative camera motion.

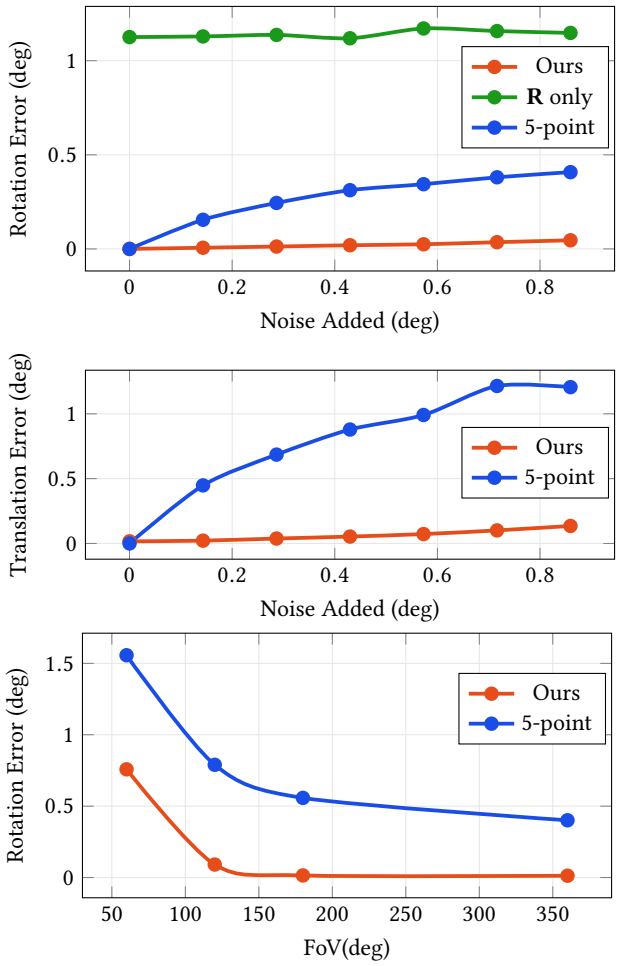
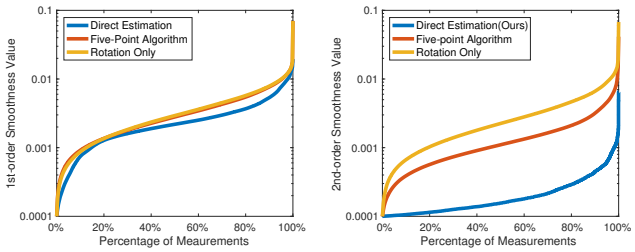


Fig. 10. Validation of our proposed method on synthetic data. Top: A rotation-only model **R** (green) performs significantly worse than our full approach that models both **R** and 3D translation direction **t** (red), which outperforms the commonly-used 5-point algorithm (blue) in the presence of increasing noise. Middle: We observe a similar difference when comparing our recovered 3D translation direction to the 5-point algorithm. Bottom: When comparing the result quality at different FoV settings, we can see that our direct estimation has consistently less error.

For our experiments we use $\tau = 2$, $\kappa = 30^\circ$, $\gamma = 8$. We fix the outlier percentage to 10% and solve for the relative poses 1000 times at each noise level, recording the final average accuracy. As shown in Fig 10 our direct motion estimation is more accurate than the 5-point algorithm for 360° video, and is more robust to increasing noise levels due to the fact that it can consider a larger number of background tracks. Estimating both rotation and translation also enables us to derive the 3D spherical warping proposed in Sec. 3.7. We also found the approach to be particularly robust to the field of view (FoV) parameter for traditional video, which allowed us to use

rotation only 5-point Ours

Fig. 11. Visualization of feature tracks after running our full method using the 5-point algorithm for estimating 3D rotation and translation (left) vs using our proposed approach (right), showing that our result is smoother. Please see the supplemental video for a comparison.



First-order Smoothness Second-order Smoothness

Fig. 12. Quantitative comparison of first-order and second-order smoothness by different motion estimation methods. The curves show the cumulative distribution function of (a) first-order and (b) second-order smoothness costs for the feature trajectories stabilized by different motion estimation algorithms. The y-axis represents the value of the first and second-order smoothness, and x-axis represents the percentage of the values larger than a corresponding y value.

a rough estimate of 120° horizontal FoV for all wide-angle videos, with the vertical FoV dependent on the aspect ratio.

In addition to the quantitative evaluation of our motion estimation in Sec. 3.2, we also show the smoothness of final results achieved by different motion estimation strategies. Similar to [Kopf 2016], we collect all feature trajectories and evaluate the value of first-order and second-order smoothness terms in Eq. 15. Fig. 12 shows the cumulative distribution functions (CDFs) for these quantities by different motion estimation methods. We can see that feature trajectories optimized by our translation-aware transformation are smoother than 5-point algorithm, and that, as also noted in [Kopf 2016] that rotation only model can not stabilize the video sufficiently.

4.2 Qualitative Evaluation

In this section, we perform ablation studies and compare alternatives for each component of our method. For evaluation, we collected 12 360° and 8 wide angle videos from YouTube, as well as 3 360° videos from the Pano2Vid dataset [Su et al. 2016]. The resolution ranges from 1920×960 to 3840×2160, for 360° videos, and 1280×720 to 3840×2160 for wide angle videos, with unknown FoVs. We then trim long videos such that the final duration ranges from 30-60 seconds, which is long enough for viewers to fully explore the video and much longer than prior datasets used for narrow FOV video stabilization [Liu et al. 2013]. In Fig.14, we compare smoothness norms by visualizing the path of a scene point across frames. Please see the supplemental video for reference.

We present all video results in the supplemental material in equirectangular format, which we encourage to be viewed on a VR headset if possible. If this is not possible, we provide a link to a desktop viewer where the videos can be explored by mouse as well as a normal FoV video that presents select results.

Direction w/o Stabilization. Some recent works [Hu et al. 2017; Su and Grauman 2017; Su et al. 2016] focus on converting spherical videos to traditional narrow FoV videos. These works re-direct the video, but do not consider the camera motion in the original videos, assuming that the input is captured by static or smooth cameras. This assumption does not hold for many consumer videos captured by hand-held cameras with shaky movement. As shown in Fig. 15, feature trajectories in the output of Pano2Vid are as shaky as the input, while our result not only focus on the important content but achieves this with a smooth camera path.

Stabilization w/o Direction. Using our method we can also generate stabilized-only results, and we provide some examples of this in the supplemental material. In general, videos that are entirely stabilized are hard to watch, as camera motion and objects of interest can drift away from true north, causing viewers to get lost. This can be seen in Fig. 18 (c,d), where when viewed with stabilization but without direction (b), the opposing kendo player moves around the camera quickly, and many viewers loose track of the players, requiring some time for their gaze to catch up to the action. However, in the directed version (c), the opposing player is kept in the center, which makes viewing much easier. It is also possible to reintroduce a smoothed version of the original viewing direction back into the video [Kopf 2016], however this is not sufficient in many cases, as the original camera direction may often times not be ideal as shown in Fig. 16, and also can be seen in Fig. 18 (a,b).

Finally, as observed in our user study, and collaborated by other perceptual studies [Sitzmann et al. 2018], many viewers tend to watch 360° content passively. We can see that the average viewing direction is highly centered in the true north direction, independent of the video content. Therefore, it is crucial to direct interesting events to this region.

Two-stage vs Joint Optimization. One question is whether it is necessary to optimize the direction and smoothness jointly, or if satisfactory results could be obtained by stabilizing the inputs first and then directing the output of stabilization. Some recent works [Hu et al. 2017] uses the later strategy to generate 2D hyperlapse from 3D spherical videos. We compare the two-stage optimization with our joint approach by first stabilizing the video ignoring the direction constraint in Eq. 6, then we adjust the direction of the stabilized result by smooth interpolation of the positive direction constraints. As shown in Fig. 17, our approach creates smoother camera paths overall, as direction constraints may be inconsistent across time, and being able to jointly solve for both gives us more flexibility to choose between multiple valid stable paths. This is especially true with automatically generated constraints such as forward motion or

Fig. 13. 360° Video Sequences. Thumbnails from the video sequences used for results. These include 12 360 video sequences from Youtube, and the sequences HIKING2, HIKING3 and SOCCER from the Pano2Vid dataset. Additional wide-angle videos are delineated with an *.

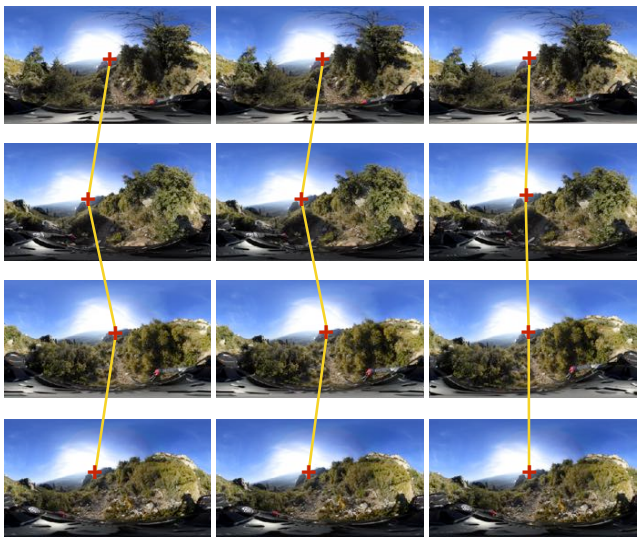


Fig. 14. We show the horizontal movement of a fixed scene point with different smoothness constraints. We can see that in this case the L2 smoothness term generates a more stable result than L1 smoothness.

Fig. 15. The feature trajectories of the input and Pano2vid [Su et al. 2016] are shaky while our result is more stable and maintains the direction constraint from pano2vid.

saliency. Please refer to the supplementary videos for more detailed comparison.

4.3 User Study

We additionally validate our approach by conducting a user study. In this study, we attempt to answer two questions. The first is whether our solution actually increases the chance that users will see the desired targets in the 360° video. Second, we attempt to qualitatively evaluate our results by determining user preferences of our results as compared to the inputs and fully stabilized versions.

To do this, we recruited 20 users to participate in our study, with an age range from 25 to 50 years old. Users were seated to simulate common viewing conditions for watching 360° video at home, and the videos were viewed using a Google Cardboard VR headset with an iPhone 6s. We then conducted a two-alternative forced choice

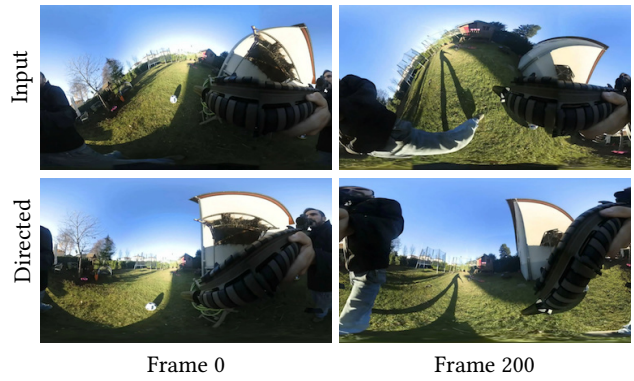


Fig. 16. In the input video (top) the camera is mounted on a toy gun, which is pointed down at the ground frequently . Using manual editing constraints we can keep the video pointed ahead even when the gun is lowered, making it much easier to watch.

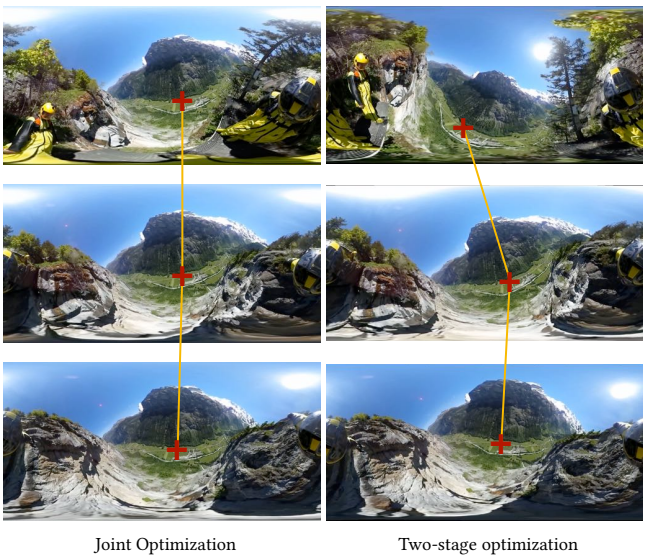


Fig. 17. A two-stage optimization can not guarantee the direction in the output is consistent since the direction constraint is generated for each edited frame independently. So it is necessary to optimize with direction constraints and smoothness constraints jointly to regularize the direction constraint.

(2AFC) preference experiment, where viewers were presented with two versions of a video and after viewing both sequentially were asked which they preferred. In addition, the gaze directions of each viewer was recorded, to measure the overlap with the positive constraint regions. The study included 6 videos, with an average length of 30 seconds. For each video, we compare the following versions: a) the original video, b) the stabilized only video, c) the directed and stabilized video via two-stage optimization and d) our directed and stabilized video via joint optimization. Comparisons are made as pairwise choices, yielding 6 different pairwise comparisons per

	Original Video	Ours	Stabilized
Mean distance (deg)	57.89	**25.87**	76.69
Percent seen (%)	34.79	**56.18**	13.23

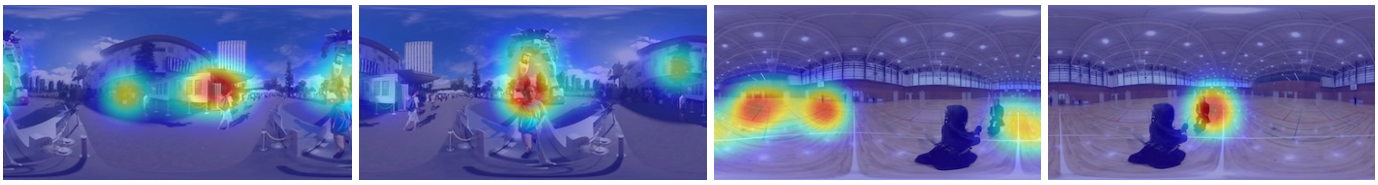
Table 2. Measured distance of participants' gaze directions to the positive directional constraints (lower is better). We also show the percent of positive constraints likely seen, which corresponds to the fraction of constraints that were within 30° of a viewer's gaze (higher is better).

video. Within each pair, video positions are randomized, and the order of the 6×6=36 pairs is drawn randomly. Viewers were not given any specific viewing instructions besides simply exploring the videos as they like.

To validate whether our method can be used to improve the viewing experience by directing the camera towards important events, we compute two quantitative measures using the recorded viewing tracks. First, we measure the mean distance of all users to all positive constraints in the videos. Second, we compute what percentage of viewers gaze was within 30° of the positive constraints. We can see the results in Table 2, indicating that adding direction to the camera significantly increased the chance that viewers will witness the events that the editor chooses to highlight. Figure 18 shows an example frame where the viewers have missed an important event in the undirected version. This is an important step to validate, given that the main goal of our approach is to increase the chance of seeing positively marked events. This is mostly due to the inherent preference for forward-facing viewing, and the problem with getting lost when camera motion is not smooth. Please see the supplemental material for a visualization of viewing direction of participants.

In the second experiment, we perform a qualitative test, where users were asked for each pair, "which video did you prefer to watch?". The results of this study are shown in Fig. 19. We use one-way analysis of variance (ANOVA) [**?**] to validate the statistical significance of our results. To analyze the multiple preference test in a single ANOVA pass, we count the times that a specific version was selected as the preferred video, and use this as a score. After collecting scores for all four different versions on all videos, we run one-way ANOVA to evaluate the statistics significance of the differences between different models. We find that the p-value of the ANOVA is 7.269×10^{-14}, indicating that the conclusions of our subjective evaluation are reasonable. As shown in Fig. 19, when compared with the input, a vast majority of users preferred our directed and stabilized version to both the input and stabilized footage. When compared within the directed and stabilized version, the joint optimization approaches is slightly preferred over the two-stage optimization, indicating that the joint optimization has found a good balance between stability and direction, although this result is less statistically significant. We also conduct a comparison between the stabilized version and the input, and found a slight preference of the stabilized version.

We note that we conducted a relatively long term user study, lasting 36 minutes for each viewer. To avoid viewers exceeding the



(a) Input (a) Directed (c) Stabilized (d) Directed

Fig. 18. Gaze directions from our user study visualized as a heatmap over the video frames. In (a), a large portion of viewers will miss seeing the robot from the front, as they remain looking around the direction of motion. In (b), the camera rotates quickly, moving the important action out of view. It takes some time for users to find the kendo player again after this, whereas in the directed version, they can watch the whole scene without getting lost.

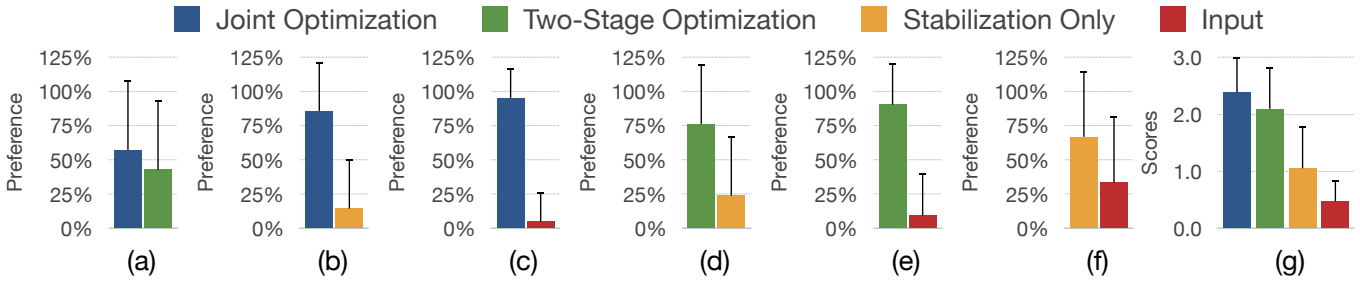


Fig. 19. Percentage of obtained votes from the user study, showing a slight preference for (a) joint direction and stabilization v.s. two-stage direction and stabilization, and strong preferences for (b) joint direction and stabilization v.s. stabilization only, (c) stabilization only v.s. input, (d) two-stage direction and stabilization v.s. stabilization only, and (e) two-stage direction and stabilization v.s. input. We also show a slight preference of (f) stabilization only v.s. input, and (g) the mean scores for each version.

maximum time that they were comfortable viewing 360°video, we split the 36 comparison pairs into 3 groups and let the viewers watch one group each time with rests in between. Longer experiments would be needed to measure differences in viewing comfort over extended viewing sessions.

4.4 Limitations

Our method directs and stabilizes 360° or wide-angle videos based on estimated 3D camera motion and directional constraints. While this approach works well in many cases, it has some limitations. For one, we found that when videos were captured using multi-camera setups with imprecise temporal synchronization, the 3D scene assumptions in Sec. 3.2 are no longer valid, and our method is unable to compensate the shakiness in the input video.

Second, the joint direction and stabilization optimization in Sec. 3.2 assumes that the input direction satisfies the cinematography rules. When the input direction constraint violates cinematographic rules, our joint optimization can not correct the direction constraint by smoothness. We can see examples of this when viewing constraints are derived for example from automatic saliency computations in the supplemental material. Additionally, when foreground objects, especially those with coherent motion tracks, occupy the majority of the viewing sphere, our optimization is unable to identify only the correct background tracks. Please see the supplemental video for examples of these cases.

5 CONCLUSION AND FUTURE WORK

In conclusion, we have presented an approach for joint stabilization and direction of 360° video. 360° video is particularly well suited for this task, as all views are present at capture time, allowing full control over viewing direction in post. In our work, we address modifying only the look-at direction, however this is just one small part of computational cinematography for 360° video, and there are many interesting areas for follow up work. For example, we do not experiment with changing focal length (zoom), as current 360° cameras do not have suitable resolution for close-up shots. Recent work [Serrano et al. 2017] has studied how people react to cuts in 360° viewing, especially when important content regions are inconsistent across the cut. We validate our approach with a similar experiment, in that we look at the percentage of fixation points of viewers inside regions of interest over short videos. However, our study is complementary, and shows how direction affects the viewing experience. These findings suggest our method could be used in conjunction with observations from Serrano et al. [2017] to automatically direct content to be consistent over cuts. Finally, we believe that virtual cinema experiences with wide angle footage is a good way to bridge the gap between the wide availability of 360° viewing devices and the limited library of content. To this end, our approach can be used with common wide angle first person cameras, as well as the recently introduced family of 180° VR cameras.

REFERENCES

Sameer Agarwal, Keir Mierle, and Others. 2017. Ceres Solver. https://code.google.com/p/ceres-solver/. (2017).

Robert Anderson, David Gallup, Jonathan T. Barron, Janne Kontkanen, Noah Snavely, Carlos Hernández, Sameer Agarwal, and Steven M. Seitz. 2016. Jump: Virtual Reality Video. *ACM Trans. Graph.* 35, 6 (2016), 198:1–198:13.

Chris Buehler, Michael Bosse, and Leonard McMillan. 2001. Non-Metric Image-Based Rendering for Video Stabilization. In *IEEE Conference on Computer Vision and Pattern Recognition*. 609–614.

Bing-Yu Chen, Ken-Yi Lee, Wei-Ting Huang, and Jong-Shan Lin. 2008. Capturing Intention-based Full-Frame Video Stabilization. *Computer Graphics Forum* 27, 7 (2008), 1805–1814.

Y. Dai, H. Li, and L. Kneip. 2016. Rolling Shutter Camera Relative Pose: Generalized Epipolar Geometry. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4132–4140. https://doi.org/10.1109/CVPR.2016.448

Vineet Gandhi, Remi Ronfard, and Michael Gleicher. 2014. Multi-clip video editing from a single viewpoint. In *Proceedings of the 11th European Conference on Visual Media Production*. ACM, 9.

Michael L Gleicher and Feng Liu. 2008. Re-cinematography: Improving the camerawork of casual video. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 5, 1 (2008), 2.

Amit Goldstein and Raanan Fattal. 2012. Video Stabilization Using Epipolar Geometry. *ACM Trans. Graph.* 31, 5 (2012), 126:1–126:10.

Jeremy J. Gray. 1980. Olinde Rodrigues' paper of 1840 on transformation groups. *Archive for History of Exact Sciences* 21, 4 (1980), 375–385. https://doi.org/10.1007/BF00595376

Matthias Grundmann, Vivek Kwatra, and Irfan Essa. 2011. Auto-directed video stabilization with robust L1 optimal camera paths. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 225–232.

R. I. Hartley and A. Zisserman. 2004. *Multiple View Geometry in Computer Vision* (second ed.). Cambridge University Press, ISBN: 0521540518.

Hou-Ning Hu, Yen-Chen Lin, Ming-Yu Liu, Hsien-Tzu Cheng, Yung-Ju Chang, and Min Sun. 2017. Deep 360 Pilot: Learning a Deep Agent for Piloting through 360Âř Sports Video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Eakta Jain, Yaser Sheikh, Ariel Shamir, and Jessica Hodgins. 2015. Gaze-Driven Video Re-Editing. *ACM Trans. Graph.* 34, 2 (2015), 21:1–21:12.

Jason Jerald. 2016. *The VR Book: Human-Centered Design for Virtual Reality*. Association for Computing Machinery and Morgan & Claypool.

L. Kneip and S. Lynen. 2013. Direct Optimization of Frame-to-Frame Rotation. In *2013 IEEE International Conference on Computer Vision*. 2352–2359. https://doi.org/10.1109/ICCV.2013.292

Laurent Kneip, Roland Siegwart, and Marc Pollefeys. 2012. Finding the exact rotation between two images independently of the translation. In *Proc. of ECCV*.

Johannes Kopf. 2016. 360Âř video stabilization. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 195.

Johannes Kopf, Michael F Cohen, and Richard Szeliski. 2014. First-person hyper-lapse videos. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 78.

Jungjin Lee, Bumki Kim, Kyehyun Kim, Younghui Kim, and Junyong Noh. 2016. Rich360: Optimized Spherical Representation from Structured Panoramic Camera Arrays. *ACM Trans. Graph.* 35, 4 (2016), 63:1–63:11.

Ken-Yi Lee, Yung-Yu Chuang, Bing-Yu Chen, and Ming Ouhyoung. 2009. Video stabilization using robust feature trajectories. In *IEEE International Conference on Computer Vision*. IEEE, 1397–1404.

Hongdong Li and Richard Hartley. 2006. Five-point motion estimation made easy. In *18th International Conference on Pattern Recognition (ICPR'06)*, Vol. 1. IEEE, 630–633.

Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. 2009. Content-preserving warps for 3D video stabilization. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 44.

Feng Liu, Michael Gleicher, Jue Wang, Hailin Jin, and Aseem Agarwala. 2011. Subspace video stabilization. *ACM Transactions on Graphics (TOG)* 30, 1 (2011), 4.

Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. 2013. Bundled camera paths for video stabilization. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 78.

Yasuyuki Matsushita, Eyal Ofek, Weina Ge, Xiaoou Tang, and Heung-Yeung Shum. 2006. Full-Frame Video Stabilization with Motion Inpainting. *IEEE Trans. Pattern Anal. Mach. Intell.* 28, 7 (2006), 1150–1163.

D. Nister. 2004. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, 6 (June 2004), 756–770. https://doi.org/10.1109/TPAMI.2004.17

Ana Serrano, Vincent Sitzmann, Jaime Ruiz-Borau, Gordon Wetzstein, Diego Gutierrez, and Belen Masia. 2017. Movie editing and cognitive event segmentation in virtual reality video. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 47.

Jianbo Shi and Carlo Tomasi. 1994. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 593–600.

V. Sitzmann, A. Serrano, A. Pavel, M. Agrawala, D. Gutierrez, B. Masia, and G. Wetzstein. 2018. Saliency in VR: How Do People Explore Virtual Environments? *IEEE Transactions on Visualization and Computer Graphics* 24, 4 (April 2018), 1633–1642. https://doi.org/10.1109/TVCG.2018.2793599

H. Stewenius, D. Nister, F. Kahl, and F. Schaffalitzky. 2005. A minimal solution for relative pose with unknown focal length. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 2. 789–794 vol. 2. https://doi.org/10.1109/CVPR.2005.36

Hans Strasburger, Ingo Rentschler, and Martin Jüttner. 2011. Peripheral vision and pattern recognition: A review. *Journal of vision* 11, 5 (2011), 13–13.

Y. Su and K. Grauman. 2017. Making 360Âř Video Watchable in 2D: Learning Videography for Click Free Viewing. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1368–1376. https://doi.org/10.1109/CVPR.2017.150

Yu-Chuan Su, Dinesh Jayaraman, and Kristen Grauman. 2016. Pano2Vid: Automatic Cinematography for Watching 360° Videos. In *Asian Conference on Computer Vision*.

Qi Sun, Li-Yi Wei, and Arie Kaufman. 2016. Mapping virtual and physical reality. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 64.

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

Yu-Shuen Wang, Hongbo Fu, Olga Sorkine, Tong-Yee Lee, and Hans-Peter Seidel. 2009. Motion-aware temporal coherence for video resizing. *ACM Transactions on Graphics (TOG)* 28, 5 (2009), 127.

A DERIVATIVES OF MOTION ESTIMATION

In this appendix, we give the analytic derivatives of our motion estimation. For the convenience of representation, we denote $\mathbf{t}_\times\mathbf{R}\mathbf{p}$ in Eq. 4 as $\mathbf{n}$ and multiply the angle ω to the rotation axis $\mathbf{r}$ as $\mathbf{w}$. We first give $\frac{\partial E}{\partial \mathbf{n}}$, $\frac{\partial \mathbf{n}}{\partial \mathbf{w}}$ and $\frac{\partial \mathbf{n}}{\partial \mathbf{t}}$, and then $\frac{\partial E}{\partial \mathbf{w}}$ and $\frac{\partial E}{\partial \mathbf{t}}$ can be calculated using chain rule.

$\frac{\partial E}{\partial \mathbf{n}}$ is derived as:

$$\frac{\partial E}{\partial \mathbf{n}} = \frac{\mathbf{q}^\top}{\sqrt{1-\left(\frac{\mathbf{q}^\top\mathbf{n}}{\|\mathbf{n}\|}\right)^2}}\left(\frac{\mathbf{I}}{\|\mathbf{n}\|} - \frac{\mathbf{n}\mathbf{n}^\top}{\|\mathbf{n}\|^3}\right), \tag{17}$$

where $\mathbf{I}$ is a 3×3 identity matrix.

$\frac{\partial \mathbf{n}}{\partial \mathbf{w}}$ and $\frac{\partial \mathbf{n}}{\partial \mathbf{t}}$ are:

$$\frac{\partial \mathbf{n}}{\partial \mathbf{w}} = \mathbf{t}_\times(\mathbf{R}\mathbf{p})_\times, \tag{18}$$

$$\frac{\partial \mathbf{n}}{\partial \mathbf{t}} = (\mathbf{R}\mathbf{p})_\times. \tag{19}$$

By chain rule we can finally get $\frac{\partial E}{\partial \mathbf{w}}$ and $\frac{\partial E}{\partial \mathbf{t}}$ as $\frac{\partial E}{\partial \mathbf{w}} = \frac{\partial E}{\partial \mathbf{n}}\frac{\partial \mathbf{n}}{\partial \mathbf{w}}$ and $\frac{\partial E}{\partial \mathbf{t}} = \frac{\partial E}{\partial \mathbf{n}}\frac{\partial \mathbf{n}}{\partial \mathbf{t}}$.