Transitioning360: Content-aware NFoV Virtual Camera Paths for 360° Video Playback

Miao Wang*
State Key Laboratory of Virtual Reality Technology and Systems, Beihang University; Peng Cheng Laboratory

Yi-Jun Li†
State Key Laboratory of Virtual Reality Technology and Systems, Beihang University

Wen-Xuan Zhang‡
State Key Laboratory of Virtual Reality Technology and Systems, Beihang University

Christian Richardt§
University of Bath, UK

Shi-Min Hu¶
BNRist, Tsinghua University, Beijing

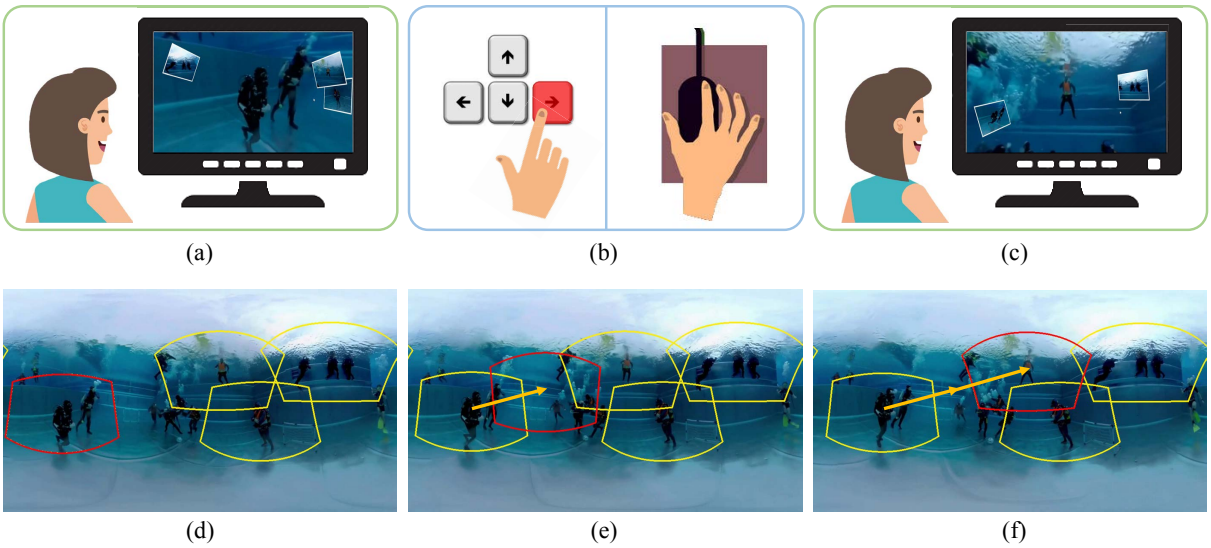


Figure 1: We present a new interaction method for transitioning between content-aware normal field of view (NFoV) camera paths that enables efficient 360° video playback on 2D displays. (a) The user watches an NFoV video corresponding to a camera path extracted from the input 360° video. The corresponding view boundary is marked in red and other candidate views are marked in yellow in (d). All NFoV camera paths are precomputed by the proposed content-aware and diverse virtual camera path optimization. (b) During playback, the user can either press the arrow key or click the thumbnail, to spatially transition from the current view to another view; the orange arrow in (e, f) indicates the trajectory of the transitioning view in red. When transitioning is finished, the user continues watching the content in the new view (c), preserving the awareness of directional and contextual information.

Abstract

Despite the increasing number of head-mounted displays, many 360° VR videos are still being viewed by users on existing 2D displays. To this end, a subset of the 360° video content is often shown inside a manually or semi-automatically selected normal-field-of-view (NFoV) window. However, during the playback, simply watching an NFoV video can easily miss concurrent off-screen content. We present *Transitioning360*, a tool for 360° video navigation and playback on 2D displays by transitioning between multiple NFoV views that track potentially interesting targets or events. Our method computes virtual NFoV camera paths considering content awareness and diversity in an offline preprocess. During playback, the user can watch any NFoV view corresponding to a precomputed camera path. Moreover, our interface shows other candidate views, providing a sense of concurrent events. At any time, the user can transition to other candidate views for fast navigation and exploration. Experimental results including a user study demonstrate that the viewing experience using our method is more enjoyable and convenient than previous methods.

Index Terms: Computing methodologies—Computer graphics—Graphics systems and interfaces—Virtual reality; Computing methodologies—Computer graphics—Image manipulation—Image processing

*e-mail: miaow@buaa.edu.cn (corresponding author)
†e-mail: yaoling@buaa.edu.cn
‡e-mail:zwx980624@gmail.com
§e-mail: christian@richardt.name
¶e-mail: shimin@tsinghua.edu.cn

1 Introduction

360° videos with omnidirectional field-of-view can nowadays be recorded easily, as commercial mobile 360° cameras, such as the Insta360 ONE X and RICOH Theta cameras, are becoming popular. 360° video content can be viewed via head-mounted displays (HMDs) such as HTC Vive and Oculus Rift [10], with which users can freely and easily rotate their heads to explore and watch interesting content under certain views [20]. However, HMDs are not always available, and a more general way to experience 360° videos and other virtual content is to navigate a normal-field-of-view (NFoV) viewport on 2D displays (typically varying from 60° to 110°). In

such cases, the user has to manually change the viewpoint with a mouse or via a touch screen, which degrades the level enjoyment and efficiency. Moreover, during playback, the user may miss important or interesting objects or events outside the selected view.

As virtual content playback and exploration are important in the VR and XR fields, several solutions have been proposed to address this problem [5, 8, 11, 15, 21–23]. An intuitive solution is to automatically analyze a scene to track a moving object, and display it in an NFoV viewport, especially for sports video [5, 21, 22]. To support smart interaction, Kang and Cho [8] proposed a system that computes an optimal virtual camera path considering saliency of the scene and temporal smoothness of the camera path. At any time, the user can adjust the view direction and the camera path will be gradually updated to focus on a new optimal viewing direction. To visualize contextual information of important content outside the current view, the Outside-In interface [15] overlays other candidate NFoV thumbnails on the main view, where the candidates are manually specified and fixed. However, these methods either yield single virtual camera paths, which miss off-screen content, or they visualize handcrafted fixed views without automatic camera path planning.

We present *Transitioning360* – a new technique for 360° video navigation and playback on 2D NFoV displays. Given a 360° video and the number of NFoV camera paths specified by the user, our method computes virtual NFoV camera paths considering content awareness and diversity of the paths. As a result, temporally stable camera paths are created to cover the most interesting and diverse video content possible. To make the camera path optimization problem tractable, we introduce a novel diversity term to jointly consider the interaction among paths and a coarse-to-fine optimization strategy that builds upon dynamic programming. While watching and exploring the main view similar to Kang and Cho [8], contents in other potential NFoV views are computed and visualized, allowing the user to easily transition to alternative views by simply clicking the arrow keys, or clicking the corresponding view thumbnails.

We evaluated the proposed method with a user study, which confirmed that Transitioning360 was generally preferred and provided better locating capability and ease of use. We believe the proposed method can inspire the community, and summarize our main contributions as follows:

- An algorithm for diverse content-aware NFoV virtual camera paths computation with a novel diversity constraint and coarse-to-fine dynamic programming optimization.
- A method for interactive 360° video navigation with spatial-aware transitioning between NFoV paths.
- A user study to evaluate the proposed method for 360° video navigation on 2D display.

2 Related Work

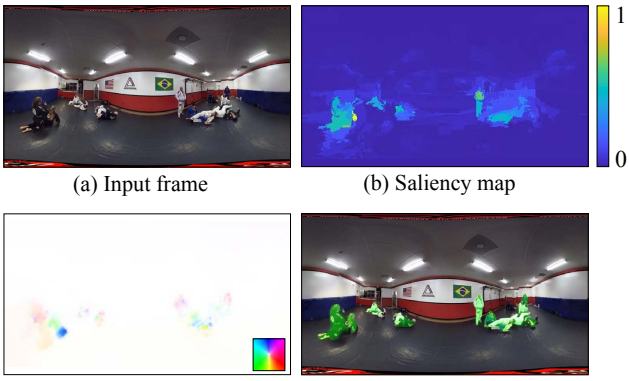
Our work is related to the general techniques of 360° image and video processing, and navigation and visualization of 360° imagery.

360° image and video processing. Nowadays, 360° cameras and display devices are becoming increasingly popular, which contributes to the growth of 360° VR images and videos available online. In some cases, the recorded content may not be ready to watch due to the suboptimal configuration of cameras during capturing. For example, if the 360° camera is not set up parallel to the ground, the recorded content should be corrected using upright adjustment. Jung et al. [6] proposed a robust upright adjustment method by optimizing the horizontal and vertical lines in the scene. To improve the scalability without the requirement of presence of lines, a deep learning-based approach [7] was proposed, which was trained on panoramic images. Temporal stabilization is another common processing task for 360° video [9, 11, 23, 27]. Kopf et al. [9] and Tang et al. [23] proposed methods to robustly estimate feature trajectories in 360° video and remove high-frequency jitters. While stabilization improves 360° viewing comfort in headsets, other approaches are necessary for exploring 360° videos. Lai et al. [11] compute semantic segmentation, saliency and the focus of expansion from 360° video, and optimize a path for a first-person NFoV hyperlapse with a constant speed. Lee et al. [12] introduced a deep neural network to produce story-based temporal summarization of 360° videos. For more 360° image and video creation methods with deep learning, we refer to a recent survey [26]. Truong et al. [17] developed an interactive tool for NFoV shot extraction based on user-specified guidelines for event scenarios. Mixed reality rendering and composition is essential for immersive and interactive mixed reality experiences in 360° video. The MR360 system [19] seamlessly composites 3D virtual objects into a live 360° video using perceptually optimized image-based rendering techniques that synthesizes illumination from LDR 360° video. Virtual objects can also be inserted into moving 360° videos when the camera motion and scene are reconstructed using 360° structure from motion [24].

Navigation of 360° video. Previous navigation techniques can be divided into automatic or interactive. Automatic navigation methods focus on computing a virtual NFoV camera path that tracks important objects in the scene. Su et al. [21, 22] introduced methods to densely sample spatiotemporal glimpses for each frame, where each glimpse is rated with a capture-worthiness score. A virtual camera trajectory across frames is then constructed by maximizing the accumulated score subject to a camera motion smoothness constraint. The NFoV video is rendered by cropping the view along the path. Deep 360 pilot [5] uses an object detector to propose candidate bounding boxes, followed by an RNN-based selector to filter the optimal bounding box. Finally, an RNN-based regressor learns smooth transitions between selected bounding boxes across frames. Unlike previous methods that either impose loose constraints on camera motion or implicitly constrain camera smoothness, Kang and Cho [8] introduced an iterative virtual camera path refinement algorithm. It first calculates an initial path considering pixel-wise saliency and camera smoothness based on optical flow in the scene, then performs FoV-aware path planning to adjust the view using regional saliency, and finally performs temporal smoothing for the path. This method successfully tracks moving objects without jumping back and forth between multiple objects. The above methods compute a single camera path for navigation, which ignores all regions of interest except the most salient one.

Interactive navigation methods guide users with visualization techniques, and allow users to adjust their view while watching 360° videos. Pavel et al. [18] proposed a button-triggered shot reorientation technique to prevent users from getting lost during interactive navigation. However, the reset viewing direction is fixed. Lin et al. [14] investigated two navigation schemes: one provides active views similar to Hu et al. [5], the other provides the guidance about important off-screen content by showing an arrow indicating its direction. Wallgrün et al. [25] compared arrow, butterfly guide, and radar guidance mechanisms for image-based VR tours. However, the visualization of visual thumbnails and NFoV view adjustment are not addressed. The Interactive360 system [8] allows the user to interactively adjust the view. Once the user manually adjusts the view, a new camera path is updated in an online manner, with the new NFoV content rendered with low latency. Lin et al. [15] developed the Outside-In system to visualize candidate off-screen content by creating off-screen view thumbnails that are floating in the 3D space between the spherical video and the screen. However, the off-screen content is manually labeled, and fixed at constant directions, which does not extend to complex and dynamic scenes.

Our Transitioning360 aims to improve the navigation and playback experience of 360° videos, especially for complex scenes, by providing diverse NFoV views for fast transitioning between them.



(a) Input frame (b) Saliency map

(c) Optical flow (d) Human instances

Figure 2: Content-awareness features computed in the preprocessing step. (a) shows an input frame of the 360° video. (b) shows the corresponding saliency map visualized as a heat map. (c) optical flow vectors are visualized with the inset color palette. (d) visualizes the human instance segmentation results in green.

3 Overview

Our approach is based on the following design principles:

1) **Content-awareness.** Candidate NFoV views should focus on the most interesting content in the input video.
2) **Diversity of views.** The virtual cameras should show diverse views and not overlap too much with each other.
3) **Stability of views.** The views should move and transition smoothly over time.
4) **Efficient navigation.** The user can quickly and easily move to other views without getting lost when changing the view.
5) **Making users aware of candidate views.** During watching the NFoV video in the main view, the user should be aware of alternative interesting views.

Following the above principles, we compute temporally smooth NFoV virtual camera paths that cover potentially interesting targets, inspired by the single-path computation in Interactive360 (Section 4). Moreover, the diversity of targets is encouraged by automatically proposing multiple camera paths using a novel diversity constraint (Section 5). During the navigation from the main view corresponding to one camera path, the user can see the thumbnails of alternative views and efficiently change the view via a spatio-temporal transition that indicates the spatial relationship between the views (Section 6).

4 Single-Path Computation

Single-path navigation [8] is a basic technique our method builds on. Given a 360° video, a content-aware, temporally smooth virtual NFoV camera path is computed. In the following, we provide a self-contained introduction to the adjusted single-path computation algorithm and refer readers to the paper [8] for more details.

4.1 Preprocessing

The content-awareness of the NFoV virtual camera path in 360° video is indicated by visual features such as video saliency, motion, and optionally object instance segmentations [4] as features in a preprocessing step (see examples in Figure 2). We use the equirectangular projection to represent 360° video. Before computation, we uniformly resize the input video to a width of 360 pixels, which optimally balances computational efficiency and accuracy. For efficiency, we sample one key frame from every fourth frames in the original video. We also pad the left and right boundaries by 20 pixels in a circular fashion, and cut the top and bottom 10 pixels to avoid errors caused by severe distortions.

Saliency maps are widely used in computer vision as a measure of human visual attention [2]. We found that the recent 360° video saliency detection methods [1, 3] usually predict cluttered visual attention fixation heat maps, where object shapes are not modeled well. Instead, we use the method by Zhou et al. [28] to compute the saliency score $s_t(\mathbf{p}) \in [0,1]$ for key frame t at 2D position $\mathbf{p}$.

Motion is represented as optical flow, where $\mathbf{o}_t(\mathbf{p})$ is the 2D displacement vector of the pixel $\mathbf{p}$ from key frame t to the next key frame. We use SIFT flow [16] to compute the optical flow between adjacent video frames and accumulate the flows between consecutive key frames as the optical flow between the corresponding key frames.

Beyond the original method [8], we further use object instance segmentation maps to detect desired semantic labels such as "human". We compute human segmentation maps using Mask R-CNN [4] pretrained on the COCO dataset [13]. The segmentation score $m_t(\mathbf{p}) \in [0,1]$ at pixel $\mathbf{p}$ for key frame t reveals the confidence that the pixel belongs to the desired category. While we use "human" as labels in our experiments, other object categories included in the COCO dataset, such as "dog", "car" or "chair", can also be used, or the user can provide new images with ground-truth segmentation map labels for training other categories.

4.2 Path Computation

The virtual NFoV path is computed in two steps: an initial path is first optimized based on content-awareness and then temporally smoothed. Throughout our experiments, we set the vertical FoV to 60° with a fixed aspect ratio of 16:9.

Given a 360° video with T key frames $F = \{f_1, \ldots, f_T\}$, we compute the corresponding saliency maps $S = \{s_1, \ldots, s_T\}$, optical flows $O = \{\mathbf{o}_1, \ldots, \mathbf{o}_T\}$ and instance segmentation maps $M = \{m_1, \ldots, m_T\}$. We uniformly downsample the spatial dimension of S, O and M to size $W \times H$ with $W = 180$ pixels for computational efficiency. An initial path $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_t, \ldots, \mathbf{p}_T\}$ is computed with $\mathbf{p}_t$ representing a 2D pixel coordinate for the t-th key frame on the path. The initial path is obtained by minimizing the following objective:

$$E(P) = \sum_{t=1}^{T} (|1 - c_t(\mathbf{p}_t)|) + \omega_o \sum_{t=1}^{T-1} \|\mathbf{v}(\mathbf{p}_t, \mathbf{p}_{t+1}) - \mathbf{o}_t(\mathbf{p}_t)\|, \quad (1)$$

where the content-awareness is encouraged by the term $c_t(\mathbf{p}_t) = \lambda_s s_t(\mathbf{p}_t) + \lambda_m m_t(\mathbf{p}_t)$, a weighted combination of the saliency $s_t(\mathbf{p}_t)$ and segmentation score $m_t(\mathbf{p}_t)$ at pixel $\mathbf{p}_t$ in key frame t. $\mathbf{v}(p_t, p_{t+1})$ is the 2D vector from $\mathbf{p}_t$ to $\mathbf{p}_{t+1}$, defined in a horizontally circular form, encouraged to be close to the motion $\mathbf{o}_t(\mathbf{p}_t)$ in the scene. $\omega_o = 0.1$ balances the effects of content-awareness and motion. We use weights $\lambda_s = 5/6$ and $\lambda_m = 1/6$ if the specified object categories are detected in the scene, for static cameras. If strong camera motion exists, we alternatively set $\lambda_s = 1/6$ and $\lambda_m = 5/6$, because saliency values for objects could be suppressed. In all other cases, we set $\lambda_s = 1$ and $\lambda_m = 0$.

The objective in Equation 1 is recursively optimized by minimizing $E_t(\mathbf{p}_t)$, the energy of a path from the first key frame to the t-th key frame, ending at $\mathbf{p}_t$:

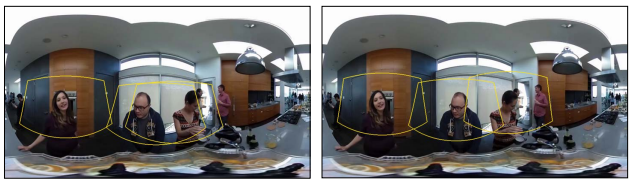
$$E_t(\mathbf{p}_t) = E_{t-1}(\dot{\mathbf{p}}_{t-1}) + |1 - c_t(\mathbf{p}_t)| + \omega_o \|\mathbf{v}(\dot{\mathbf{p}}_{t-1}, \mathbf{p}_t) - \mathbf{o}_{t-1}(\dot{\mathbf{p}}_{t-1})\|, \quad (2)$$

where

$$\dot{\mathbf{p}}_{t-1} = \arg\min_{\dot{\mathbf{p}} \in \mathcal{N}(\mathbf{p}_t)} \{E_{t-1}(\dot{\mathbf{p}}) + \omega_o \|\mathbf{v}(\dot{\mathbf{p}}, \mathbf{p}_t) - \mathbf{o}_{t-1}(\dot{\mathbf{p}})\|\}. \quad (3)$$

Here, $\mathcal{N}(\mathbf{p}_t)$ is a 31×31 spatial neighborhood of $\mathbf{p}_t$, which is large enough to track fast-moving objects. We compute the optimal $E_t(\mathbf{p}_t)$ by increasing t from 1 to T, and backtracking from the globally optimal solution $E_T(\mathbf{p}_T)$ to obtain the optimal path P.

After planning the initial path P, we compute a temporally smooth path $\hat{P} = \{\hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_T\}$ by minimizing an objective for suppressing



(a) With FoV-aware adjustment (b) Without FoV-aware adjustment

Figure 3: Comparison of paths with and without FoV-aware adjustment. (a) shows a frame where three NFoV views with yellow boundaries are computed with FoV-aware path adjustment. Two of the three NFoV views overlap each other severely. (b) shows the camera paths without FoV-aware adjustment; each camera is focusing on a different person.

the velocity and acceleration of the virtual camera trajectory while approximating the initial path:

$$\hat{E}(\hat{P}) = \sum_{t=1}^{T} \|\hat{\mathbf{p}}_t - \mathbf{p}_t\|^2 + \omega_v \sum_{t=1}^{T-1} \|\hat{\mathbf{p}}_{t+1} - \hat{\mathbf{p}}_t\|^2 + \omega_a \sum_{t=2}^{T-1} \|\hat{\mathbf{p}}_{t+1} - 2\hat{\mathbf{p}}_t + \hat{\mathbf{p}}_{t-1}\|^2, \tag{4}$$

where $\omega_v = 20$ and $\omega_a = 20$ are weights balancing the data, velocity and acceleration terms.

In Interactive360 [8], an intermediate, FoV-aware path P' is computed before the path smoothing, to reflect the surrounding context based on the initial path P for each key frame:

$$E'(\mathbf{p}'_t) = \left|1 - c'_t(\mathbf{p}'_t)\right| + \omega_p \left\|\mathbf{p}'_t - \mathbf{p}_t\right\|, \tag{5}$$

where $\mathbf{p}'_t \in P'$ is the FoV-aware path at the t-th frame, $c'_t(\mathbf{p}'_t)$ is the corresponding regional content-awareness defined as the average of content-awareness $c_t(p)$ within the FoV centered at $\mathbf{p}'_t$. However, when computing multiple paths, this adjustment results in nearby NFoV views to overlap each other and degrades the diversity of paths, see Figure 3. We thus omit the FoV-aware adjustment step.

During online navigation, the user is allowed to manually adjust the view. Once the user assigns a new view, we use a path finder to optimize a new path starting at the current view, and use a video player to render and show the partially updated path [8].

5 Diverse Virtual Camera Paths

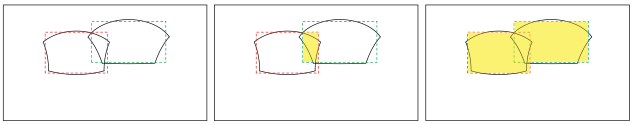
We build on the techniques in Section 4 to compute multiple diverse NFoV camera paths. The number N of views can be manually specified by the user, or automatically determined as follows: we set N as the number of visually salient objects where the average area ratio of minimum bounding box to maximum bounding box among candidate detected objects for all key frames is larger than 0.5. Such automatic view number setting is suitable when the number of interesting objects in the video is stable over time.

As a pilot trial, we have used a greedy algorithm by iteratively extracting paths. We show the drawbacks of the greedy algorithm-based solution, and then propose a dynamic programming-based optimization algorithm with a coarse-to-fine strategy.

5.1 Greedy Algorithm: A Pilot Trial

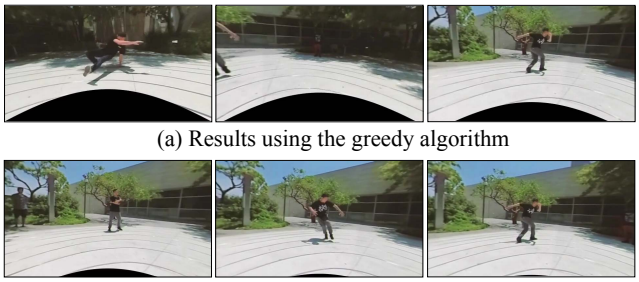
A straightforward idea to compute multiple paths is to iteratively compute single paths as in Section 4.2, and to suppress the content-awareness features along the previous optimal paths. The modification of the content-awareness term prevents the paths from being substantially the same in subsequent iterations. Let $\hat{P}^k$ denote the optimal path computed in the k-th iteration. The saliency maps S^k at the k-th iteration can be suppressed around the path as follows:

$$s_t^k(\mathbf{q}_t) = \begin{cases} 0, & \text{if } k>1 \text{ and } \mathrm{IOU}(R(\mathbf{q}_t), R(\hat{\mathbf{p}}_t^k)) \geq \delta \\ s_t^{k-1}(\mathbf{q}_t), & \text{otherwise (with } s_t^0 := s_t), \end{cases} \tag{6}$$



(a) Two NFoV Views (b) Intersection (c) Union

Figure 4: Approximate intersection-over-union (IOU) of irregular regions using rectangles. See the supplementary material for details.



(a) Results using the greedy algorithm

(b) Results using the dynamic programming-based joint optimization

Figure 5: Multi-path computation: greedy algorithm vs. dynamic programming. (a) shows the NFoV frames from the first path computed by the greedy algorithm (Section 5.1); (b) shows the NFoV frames from one path computed using dynamic programming, which jointly considers multiple paths and can track contents more robustly.

where $\mathbf{q}_t$ is a 2D coordinate in key frame t, $\hat{\mathbf{p}}_t^k$ is the 2D coordinate in key frame t along path k, $\mathrm{IOU}(R(\mathbf{q}_t), R(\hat{\mathbf{p}}_t^k))$ is the approximate intersection-over-union of the two NFoVs $R(\mathbf{q}_t)$ and $R(\hat{\mathbf{p}}_t^k)$ centered at $\mathbf{q}_t$ and $\hat{\mathbf{p}}_t^k$, respectively (see Figure 4), and δ is a threshold set to 0.2. The instance segmentation maps M^k are processed similarly.

While the above greedy algorithm can successfully extract N paths with diverse content, there are inevitably drawbacks. First, the importance levels of the paths are not uniformly distributed: the first path is always more important than others, while the last path is with least interest. Second, when targets move at different times, the first path tends to capture all moving targets by changing the target, thus being more dynamic than subsequent paths. An example is shown in Figure 5(a), where two dancers perform one after the other. The first path computed by the greedy algorithm tries to capture significant motions, and it switches the target from the first dancer to the second dancer once the former stops the performance.

5.2 Dynamic Programming-based Optimization

We present a dynamic programming-based optimization algorithm for multiple path computation. It jointly optimizes paths using content-awareness features while considering the interaction between them. Similar to the two-step single-path computation, we first compute initial paths, and then temporally smooth them. Let $\mathbb{P} = \{P^1, \ldots, P^N\}$ denote the N initial paths, where $P^n = \{\mathbf{p}_1^n, \ldots, \mathbf{p}_T^n\}$ is the n-th path with the 2D coordinate $\mathbf{p}_t^n$ representing the viewing direction in key frame t. Similar to Equation 1, the overall objective to minimize is defined as:

$$E(\mathbb{P}) = \sum_{n=1}^{N} \left[\sum_{t=1}^{T} |1 - \hat{c}_t(\mathbf{p}_t^n)| + \omega_o \sum_{t=1}^{T-1} \left\| \mathbf{v}(\mathbf{p}_t^n, \mathbf{p}_{t+1}^n) - \mathbf{o}_t(\mathbf{p}_t^n) \right\| \right], \tag{7}$$

where $\hat{c}_t(\mathbf{p}_t^n)$ includes the saliency and instance segmentation maps of the n-th path, and further considers the interaction of other paths imposed on it. We define $\hat{c}_t(\mathbf{p}_t^n)$ as follows:

$$\hat{c}_t(\mathbf{p}_t^n) = c_t(\mathbf{p}_t^n) \prod_{\substack{1 \leq k \leq N \\ k \neq n}} \max(0, 1 - \omega_d \cdot \mathrm{IOU}(R(\mathbf{p}_t^n), R(\mathbf{p}_t^k))), \tag{8}$$

where $c_t(\mathbf{p}_t^n)$ combines saliency and instance segmentation as in Section 4.2, $\mathbf{p}_t^k$ is the 2D coordinate corresponding to path k in key

frame t, IOU$(R(\mathbf{p}_t^n), R(\mathbf{p}_t^k))$ is the IOU of two NFoV views $R(\mathbf{p}_t^n)$ and $R(\mathbf{p}_t^k)$ centered at $\mathbf{p}_t^n$ and $\mathbf{p}_t^k$, respectively, and $\omega_d = 1.5$ is a constant parameter that controls the diversity of paths. Intuitively, if no other view is overlapping with $R(\mathbf{p}_t^n)$, the corresponding $\hat{c}_t(\mathbf{p}_t^n)$ equals to $c_t(\mathbf{p}_t^n)$. Otherwise, $\hat{c}_t(\mathbf{p}_t^n)$ is suppressed, determined by the IOU introduced by any other overlapping view $\mathbf{p}_t^k$.

The objective in Equation 7 can be solved using dynamic programming. Let $E_t(\mathbb{P}_t)$ denote the total energy for the N optimal paths from the first key frame up to key frame t, with paths ending at $\mathbb{P}_t = \{\mathbf{p}_t^1, \ldots, \mathbf{p}_t^N\}$. $E_t(\mathbb{P}_t)$ is recursively computed as follows:

$$E_t(\mathbb{P}_t) = E_{t-1}(\dot{\mathbb{P}}_{t-1}) + \sum_{n=1}^{N} |1 - \hat{c}_t(\mathbf{p}_t^n)| + \omega_o \left\| \mathbf{v}(\dot{\mathbf{p}}_{t-1}^n, \mathbf{p}_t^n) - \mathbf{o}_{t-1}(\dot{\mathbf{p}}_{t-1}^n) \right\|, \quad (9)$$

where the optimal paths $\dot{\mathbb{P}}_{t-1} = \{\dot{\mathbf{p}}_{t-1}^1, \ldots, \dot{\mathbf{p}}_{t-1}^N\}$ up to key frame $t-1$ are computed as:

$$\dot{\mathbf{p}}_{t-1}^n = \underset{\dot{\mathbf{p}}^n \in \mathcal{N}(\mathbf{p}_t^n)}{\arg\min} \{E_{t-1}(\dot{\mathbf{p}}^n) + \omega_o \|\mathbf{v}(\dot{\mathbf{p}}^n, \mathbf{p}_t^n) - \mathbf{o}_{t-1}(\dot{\mathbf{p}}^n)\|\}. \quad (10)$$

As before, $\mathcal{N}(\mathbf{p}_t)$ is the 31×31 spatial neighborhood of $\mathbf{p}_t$. The optimal paths $E_t(\mathbb{P}_t)$ can be computed by increasing t from 1 to T, and backtracking from the globally optimal solution $E_T(\mathbb{P}_T)$ to obtain the optimal paths $\mathbb{P}$. We then perform temporal smoothing of paths following Equation 4.

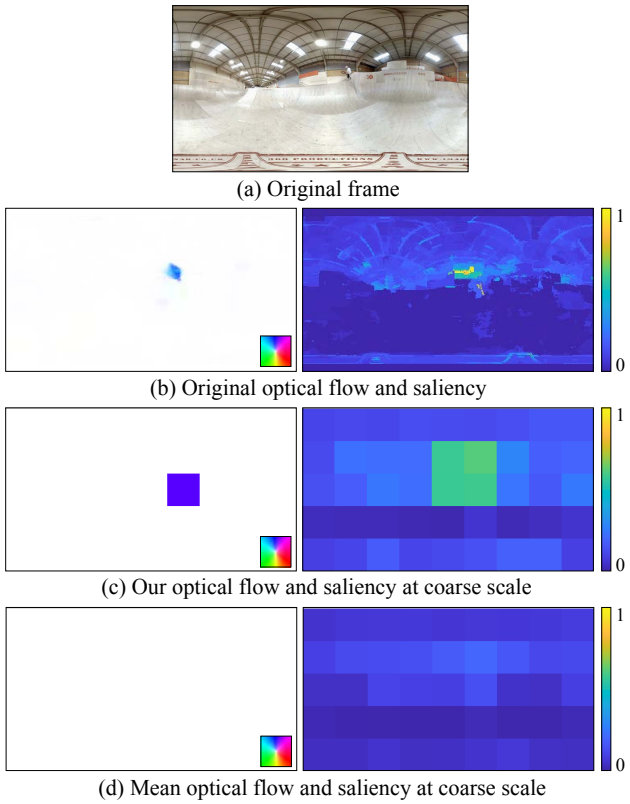
The dynamic programming-based solution, as mentioned, jointly considers the content-awareness and diversity of the N paths, which can thus avoid unnecessarily changing targets. Figure 5(b) shows a path computed by our algorithm that consistently tracks one of the targets while leaving other targets to be tracked by other paths.

5.3 Coarse-to-fine Strategy

The complexity of the proposed dynamic programming algorithm in Equation 9 increases exponentially with the number N of views. Meanwhile, directly optimizing the objective at resolution $W \times H$ is memory consuming due to the huge state space of dynamic programming. To make the problem tractable, we present a coarse-to-fine strategy. In a nutshell, the joint dynamic-programming-based optimization is performed at a coarse scale, followed by a fine-scale refinement based on single-path optimization.

We downsample the saliency S, optical flow O, and instance segmentation M to obtain coarse-scale versions $\tilde{S}$, $\tilde{O}$, $\tilde{M}$ of dimension $\tilde{W} \times \tilde{H}$, where $\tilde{W} = W/s_w$ and $\tilde{H} = H/s_h$, using scaling factors s_w and s_h. In our experiments, we set $\tilde{W} = 9$ and $\tilde{H} = 5$. Without loss of generality, we use $\mathbf{q} = (q_x, q_y)$ to denote a 2D coordinate at the coarse scale, and $F(\mathbf{q})$ to denote the spatial mapping function to the fine scale that corresponds to $\mathbf{q}$, where $F(\mathbf{q}) = \{(x,y) \mid s_w \cdot q_x \le x < s_w \cdot (q_x+1), s_y \cdot q_y \le y < s_y \cdot (q_y+1)\}$.

We note that the downsampling method for $\tilde{S}$, $\tilde{O}$ and $\tilde{M}$ is crucial for correct path optimization at such a coarse scale. For example, if the downsampled optical flow $\tilde{\mathbf{o}}_t$ at key frame t is computed as the mean of the corresponding flows in $F(\mathbf{q})$ for a pixel $\mathbf{q}$ at the coarse resolution, i.e., $\tilde{\mathbf{o}}_t(\mathbf{q}) := \frac{1}{|F(\mathbf{q})|} \sum_{\mathbf{g} \in F(\mathbf{q})} \mathbf{o}_t(\mathbf{g})$, then the downsampled flow $\tilde{\mathbf{o}}_t(\mathbf{q})$ tends to be close to zero, which will prevent cameras from moving. To compute more representative saliency $\tilde{s}_t(\mathbf{q})$, instance segmentation $\tilde{m}_t(\mathbf{q})$ and optical flow $\tilde{\mathbf{o}}_t(\mathbf{q})$ at coarse scale, we present a method that leverages motion coherence and a voting strategy. We argue that any strong coherent motion at the fine scale should also be revealed in the corresponding coarse-resolution optical flow. To this end, we first sort the fine-scale flow vectors within the footprint $F(\mathbf{q})$ of each coarse-scale pixel $\mathbf{q}$ according to their magnitude, and keep the Top-K (K=10 in our experiments) candidates $\{\mathbf{o}_t(\mathbf{g}'_1), \ldots, \mathbf{o}_t(\mathbf{g}'_K)\}$ originally from $\{\mathbf{g}'_1, \ldots, \mathbf{g}'_K\}$ at the fine scale. Then, the Top-K optical flows are used to vote for a 2D coordinate $\mathbf{q}'$ at the coarse scale, where the majority of the optical flow end points $\{\mathbf{g}'_k + \mathbf{o}_t(\mathbf{g}'_k) \mid 1 \le k \le K\}$ are located inside $F(\mathbf{q}')$.



(a) Original frame

(b) Original optical flow and saliency

(c) Our optical flow and saliency at coarse scale

(d) Mean optical flow and saliency at coarse scale

Figure 6: Downsampled saliency map and optical flow. (a) shows an input 360° video frame, whose original optical flow and saliency visualizations are given in (b). (c) shows the optical flow and saliency results using our motion-coherent voting strategy. (d) shows the naive downsampling of the fine-scale saliency and optical flow, which can miss small, highly salient targets.

Finally, the coarse optical flow vector at key frame t is computed as $\tilde{\mathbf{o}}_t(\mathbf{q}) = \mathbf{q}' - \mathbf{q}$. Let's denote the subset of $\overline{K}$ optical flow vectors (for $\overline{K} \leqslant K$) that contribute to the voting to $F(\mathbf{q}')$ as $\{\mathbf{o}_t(\mathbf{g}_1^*), \ldots, \mathbf{o}_t(\mathbf{g}_{\overline{K}}^*)\}$, starting at pixels $\mathbf{G}^* = \{\mathbf{g}_1^*, \ldots, \mathbf{g}_{\overline{K}}^*\}$. The coarse-scale saliency $\tilde{s}_t(\mathbf{q})$ and instance segmentation $\tilde{m}_t(\mathbf{q})$ are computed as the average of saliency and instance segmentation values of the fine-scale pixels $\mathbf{G}^*$. Figure 6 demonstrates the effect of our motion-coherent voting strategy for content-aware feature downsampling in coarse-to-fine dynamic programming. Our method captures the motion and saliency at the fine scale better than the average downsampling method.

When optimizing the initial paths $\tilde{\mathbb{P}} = \{\tilde{\mathbf{p}}^1, \ldots, \tilde{\mathbf{p}}^n, \ldots, \tilde{\mathbf{p}}^N\}$ at coarse scale following Equation 9, we set the spatial neighborhood $\tilde{\mathcal{N}}(\tilde{\mathbf{p}}_t^n)$ to 3×3 pixels and use the coarse-scale motion parameter $\tilde{\omega}_o = 0.1$. We further refine the coarse-scale paths $\tilde{\mathbb{P}}$ at the fine scale by enumerating possible coordinates within the spatial neighborhood $F(\tilde{\mathbf{p}}_t^n)$ and optimizing the objective in Equation 1. With this strategy, our method can efficiently compute up to $N = 4$ camera paths in minutes, which we have found to be sufficient to capture the most important and interesting video contents. Figure 7 shows an illustration of the coarse-to-fine refinement process. Finally, each path at the fine scale is temporally smoothed and rendered as an NFoV video for the final results.

Table 1: We list seven representative 360° videos we explored using Transitioning360, with their computation times for the coarse-scale joint dynamic programming (DP), fine-scale refinement and temporal smoothing stages. Further examples are provided in the supplementary material.

	Parkour	Party I	ScubaSoccer	Wrestling	Skateboard	Party II	Monkey
Thumbnail							
Resolution	1280 × 720	1280 × 720	1280 × 720	1280 × 720	1920 × 1080	1920 × 1080	1920 × 1080
Length	29 s	4 s	21 s	11 s	10 s	15 s	8 s
NFoV Views	4	3	4	3	2	4	3
Coarse-scale DP Time	235.50 s	8.11 s	173.55 s	16.99 s	23.10 s	121.94 s	12.72 s
Refinement Time	15.31 s	2.11 s	11.58 s	4.85 s	4.22 s	7.93 s	3.45 s
Smoothing Time	0.37 s	0.56 s	0.19 s	0.17 s	0.01 s	0.08 s	0.01 s

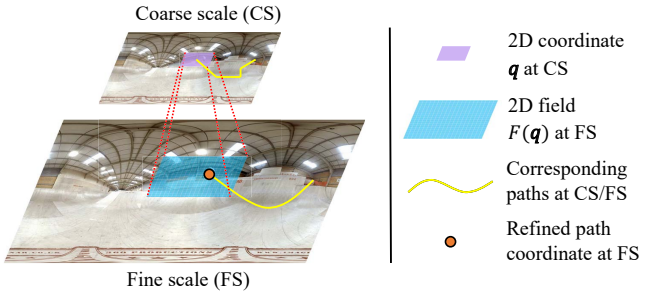
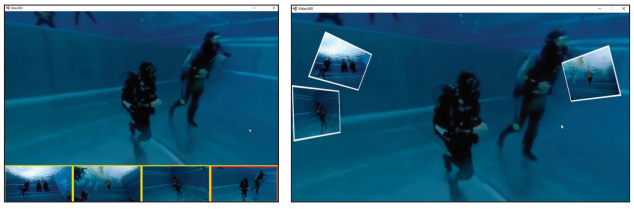


Figure 7: Coarse-to-fine strategy. A coarse-scale path determined by the joint dynamic programming optimization is refined at the fine scale. Note that the refined path coordinate corresponding to $\mathbf{q}$ at coarse scale is searched within the matching field $F(\mathbf{q})$ to be more accurate.



(a) Horizontal thumbnails (b) Outside-In visualization

Figure 8: Path visualization styles in Transitioning360. (a) The NFoV video thumbnails are horizontally placed below the main view. The thumbnail with the red boundary indicates the currently selected video. (b) Outside-In visualization. Candidate videos are displayed within the main navigation view, with their positions, sizes and shapes indicating their distances and relative locations.

6 Transitioning-based Interaction

Once the paths are precomputed, their views can be experienced via the interaction module of Transitioning360. Following the design principles in Section 3, the motivation of the interaction is to allow users to easily reach NFoV views corresponding to paths with spatio-temporal transitions, and to provide previews of other views using video thumbnails. In this section, we introduce these interactions in terms of transitions between paths and visualization of paths.

6.1 Transitioning Between NFoV Paths

Although each of the NFoV paths computed in Section 5 reveals local contents of a 360° video, the spatial relationship between paths are known, which informs the spatial-awareness of our transitioning operation. During the playback of one NFoV video, the user can simply transition to another NFoV video by clicking an arrow button (i.e., up, down, left, right) or slightly moving the mouse, which indicates a transitioning direction $\mathbf{d}^{\text{trans}}$. Once the user requests a transition at frame t^{req}, our method immediately starts to transition the current view to the neighboring video with the transitioning direction closest to $\mathbf{d}^{\text{trans}}$ within a period of time t^{trans}.

The spatial transitioning from one path to another can be performed either while the video is playing or while it is paused, with the video continuing to play afterwards. We conducted a pilot study on the two transitioning styles with three participants, and observed that they prefer transitioning without pausing the video, as pausing interrupts the flow of time. We thus perform spatio-temporal transitioning starting at frame t^{req} from the current path $\hat{\mathbf{p}}_{\text{cur}}$ to the nearest neighboring path $\hat{\mathbf{p}}_{\text{nn}}$ along the transition direction $\mathbf{d}^{\text{trans}}$ by spatially moving the current NFoV view centered at the 2D coordinate $\hat{\mathbf{p}}_{\text{cur}}^{t^{\text{req}}}$ towards the target coordinate $\hat{\mathbf{p}}_{\text{nn}}^{t^{\text{req}}+t^{\text{trans}}}$ in the 360° video with a constant transitioning velocity of $v_{\text{trans}} = 60°/\text{s}$. We note that there can be severe spatial FoV overlap between $\hat{\mathbf{p}}_{\text{cur}}$ and $\hat{\mathbf{p}}_{\text{nn}}$ around time t^{req}. The transitioning between such paths can result in annoying jump cuts. In this case, we temporarily consider $\hat{\mathbf{p}}_{\text{nn}}$ to be an invalid transition until it is more distant from $\hat{\mathbf{p}}_{\text{cur}}$ again. If the user requests to transition to an invalid path, we simply skip it and transition directly to the next valid path. An example of spatio-temporal transitioning between two NFoV views is illustrated in Figure 1(d–f).

6.2 Visualization of Paths

During the navigation of an NFoV video, a clear visualization of other NFoV views can help the user to know what is happening in the context around the current view, so that they can transition to another view efficiently. Along with the main view that plays the current NFoV video, we propose to display interactive video thumbnails of other paths that play synchronously. The user can click on any of the thumbnails to transition to the corresponding video. We clarify that the visualization is not a new technique from Transitioning360, but can enhance and supplement the transitioning interaction. We have explored two visualization styles: (1) placing the video thumbnails horizontally below the main video view, and (2) utilizing the outside-in technique [15] that spatially displays thumbnails inside the main view, with the thumbnails warped following their approximate geometry projections. Figure 8 shows an example of the visualization interfaces. As will be shown in Section 8, users generally preferred to use the outside-in visualization.

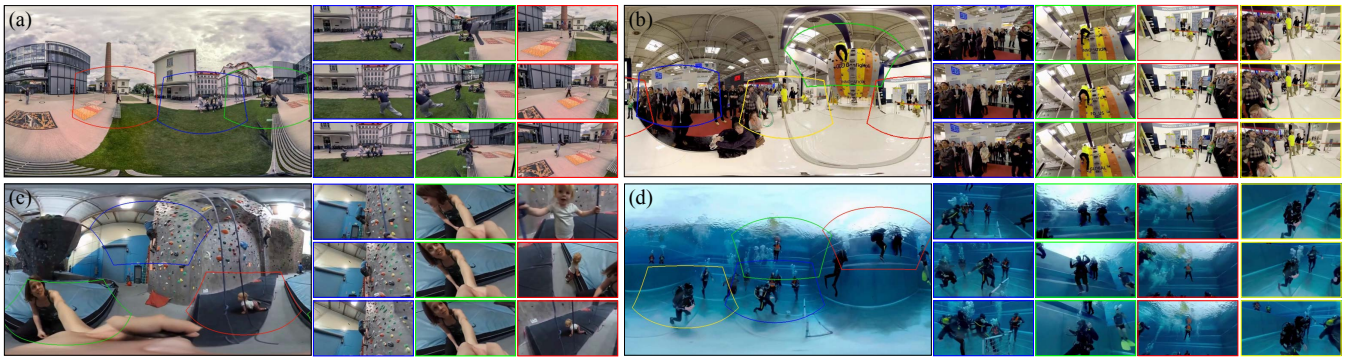
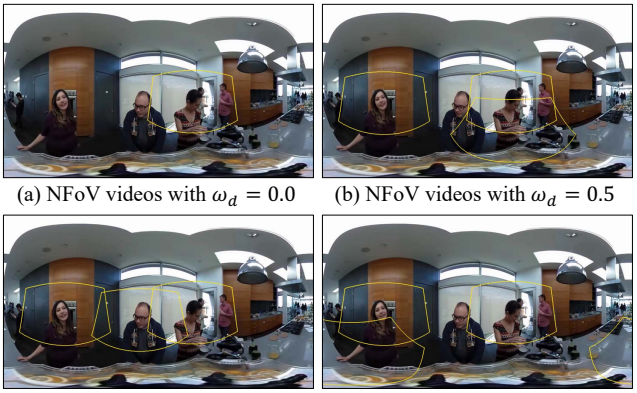


Figure 9: A gallery of our NFoV video results. (a–d) give four representative 360° frames with the computed NFoV videos marked in different colors. In each example, the corresponding NFoV video frame sequences with colored boundaries are listed vertically on the right-hand side.

(a) Transitioning by clicking the arrow key

(b) Transitioning by clicking the thumbail

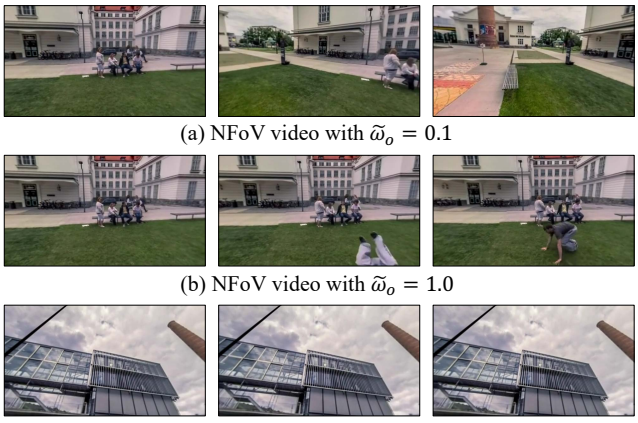
Figure 10: Interaction and transitioning process via Transitioning360 with Outside-In visualization; 3 frames of each transitioning process are shown. (a) The user requests a transition by clicking the arrow key indicating "transition to the view to the right"; the corresponding direction is shown in the center of the frame. (b) The user clicks the video thumbnail to request a transitioning to the corresponding video.



(a) NFoV videos with $\omega_d = 0.0$ (b) NFoV videos with $\omega_d = 0.5$

(c) NFoV videos with $\omega_d = 1.5$ (d) NFoV videos with $\omega_d = 2.0$

Figure 11: Effect of different diversity penalties ω_d on the diversity of views. Boundaries of three NFoV views are highlighted in yellow in (a–d). Larger ω_d results in less interesting content in paths, while smaller ω_d makes paths overlap more (fully overlapped in (a)).



(a) NFoV video with $\tilde{\omega}_o = 0.1$

(b) NFoV video with $\tilde{\omega}_o = 1.0$

(c) NFoV video with $\tilde{\omega}_o = 10.0$

Figure 12: Effect of different $\tilde{\omega}_o$ values at the coarse scale. (a) A small $\tilde{\omega}_o$ value does not track targets reliably, while a large $\tilde{\omega}_o$ (c) may sacrifice content awareness by capturing static parts of the scene. (b) Our setting of $\tilde{\omega}_o = 1$ robustly captures people in the scene.

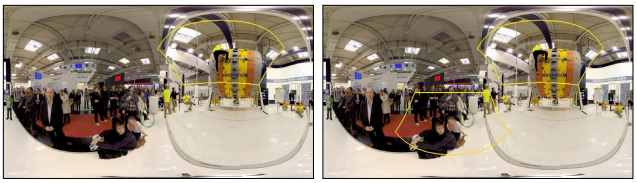
7 Experimental Results

In this section, we show experimental results generated by Transitioning360. We implemented the tool using the Unity3D game engine (2018.2.13f1), with off-the-shelf saliency detection [28], optical flow [16] and instance segmentation [4] modules based on the authors' implementations. Source code is available at `https://github.com/yaoling1997/Transitioning360`. We use a PC with an Intel Core-i7 3.6 GHz CPU and 32 GB RAM. We tested the tool using 360° videos from Su et al. [22], Kang and Cho [8] and YouTube. The length of videos varies from 4 to 34 seconds. The offline preprocessing of saliency, optical flow and instance segmentation takes about 270 seconds for a 10-second video. The online interactive navigation process using Transitioning360 is real-time. Table 1 lists representative videos captured by both static and dynamic cameras, with input video attributes and processing times for the path optimization steps. We manually set the number of camera paths for *Parkour*, *Party I*, *ScubaSoccer* and *Wrestling*, and automatically set the number of views for the remaining examples.

Figure 9 shows a gallery of automatically created NFoV video computation results. Different from previous methods [8, 22], our method computes more than one path that collectively present potentially important or interesting content. The paths are diverse and temporally stable for various scenes. Figure 10 shows typical interaction and transitioning processes via Transitioning360 using the Outside-In visualization. While watching an NFoV video, users can request transitions by clicking the arrow keys or clicking a video thumbnail. We encourage the reader to watch the supplementary video for the full experience of our results.

We have shown examples in Figure 3 to illustrate the effect of skipping the FoV-aware adjustment. Figure 5 shows the advantage of performing joint path optimization over greedy algorithm-based optimization in path computation. Figure 6 demonstrates the effect of our downsampling method with the motion coherent voting strategy.

In Figure 11, we evaluate the effect of the view diversity parameter ω_d in Equation 8, which controls the balance between content-



(a) $N = 1$ (b) $N = 2$

(c) $N = 3$ (d) $N = 4$

Figure 13: Examples of NFoV paths for $N = 1, 2, 3, 4$ views.

awareness and view diversity. As can be seen, smaller ω_d leads to severe overlaps between camera paths, while larger ω_d makes paths more diverse, with potentially less interesting content per path. We set $\omega_d = 1.5$ throughout our experiments to optimally balance content-awareness and view diversity.

Figure 12 shows the effect of the parameter $\tilde{\omega}_o$ in Section 5.3, which controls the motion sensitivity at the coarse scale. The fine-scale motion parameter ω_o is fixed to 0.1 as in the original approach [8]. Sampled NFoV video frame sequences are shown with different $\tilde{\omega}_o$ values. In Figure 12(a), with a small $\tilde{\omega}_o$, the NFoV view floats without tracking important contents, because of the reduced motion consistency penalty. Figure 12(b) shows that with $\tilde{\omega}_o = 1$, the camera robustly captures the people within the FoV. In Figure 12(c), as the penalty of motion is enlarged, the camera is conservative and captures the static scene rather than humans, which sacrifices content-awareness. We set $\tilde{\omega}_o = 1.0$ in all our experiments.

Figure 13 shows a comparison of NFoV paths computed for one to four cameras on the same input video. Using only a single NFoV camera, in Figure 13(a), focuses on the region with most motion and saliency: the climbing athlete. Additional camera views, as in Figure 13(b–d), focus on other meaningful video content, like the audience, until all dynamic content is captured. Further camera views would overlap with existing cameras or the static background.

8 User Study

To further evaluate whether Transitioning360 improves the experience of 360° video navigation and playback on 2D displays, we conducted a user study. Briefly, we invited participants to experience 360° videos using different interaction interfaces via a 2D display. After playback of each video, the participants were asked to rate each method in terms of locating capability, convenience for navigation and overall preference.

8.1 Participants

We recruited 15 participants from our university with diverse study backgrounds. From the 15 participants (7 female, 8 male, mean age 21.3 years, SD=2.3), 9 indicated previous VR and HMD experience.

8.2 Data

We collected six 360° videos for the user study: three were labeled "static" (S1, S2, S3) where most of the targets' positions in the scene are relatively fixed, while the other three videos were "dynamic" (D1, D2, D3) with some targets moving in the scene. The lengths and virtual NFoV camera path numbers of the videos were: S1 (30 s, 2 paths), S2 (22 s, 3 paths), S3 (20 s, 4 paths), D1 (20 s, 3 paths), D2 (21 s, 3 paths), D3 (22 s, 4 paths). We also used a training video (21 s, 2 paths) to train participants before the main study.

8.3 Methods and Measures

We evaluated the following interaction methods:

- **Baseline interaction with horizontal thumbnails ("BH").** The traditional interactive method widely used on YouTube. The user can click and drag the mouse within the main view to manually adjust the direction of the NFoV camera. The thumbnails below the main view are not interactive and only used for references of other interesting objects outside the main view.
- **Interactive360 with horizontal thumbnails ("IH").** The Interactive360 interaction [8] with candidate video thumbnails shown horizontally below the main view as in Figure 8(a). The user can click and drag the mouse within the main view to manually adjust the direction of the NFoV camera. The thumbnails below the main view are not interactive and only used for references of other interesting objects outside the main view.
- **Transitioning360 with horizontal thumbnails ("TH").** The Transitioning360 interaction with candidate video thumbnails shown horizontally below the main view as in Figure 8(a). The user can click and drag the mouse within the main view to manually adjust the direction of the NFoV camera. Moreover, the user can press an arrow key or click a video thumbnail for transitioning to the corresponding view.
- **Transitioning360 with Outside-In ("TO").** The Transitioning360 interaction with candidate video thumbnails displayed within the main view using Outside-In [15] as shown in Figure 8(b). The user can click and drag the mouse within the main view to manually adjust the direction of the NFoV camera. Moreover, the user can press an arrow key or click a video thumbnail for transitioning to the corresponding view.

Similar to previous work [8, 15], each experience of a video using any method was measured by the participants using 7-point Likert scales for questions on:

- **Locating capability:** "Please rate this method on the capability level of locating important targets: 1–lowest to 7–highest."
- **Ease of use:** "Please rate this method on the convenience level to navigate the video: 1–lowest to 7–highest."
- **Intention to use:** "Please rate this method on the intention to use level: 1–lowest to 7–highest."

8.4 Procedure

We used a 23″ 1920×1080 HD monitor for the video experience, with a viewing distance of 40–60 cm. The main navigation view was located in the screen center with a fixed size of 1280×720. Each participant experienced one randomly selected static video and one dynamic video with all four interaction methods. The order of methods was randomized, except for the baseline interaction, which was always tested first. There are two reasons to start with the baseline method: (1) to avoid participants becoming bored when experiencing the baseline method after experiencing other methods, and (2) to set a standard for rating the other interaction methods.

After welcoming participants, we asked them to fill out demographic information. Then, each participant sat in front of the monitor and was instructed about the user study. Before formal tests of each method, each participant was trained how to operate the interface for navigation with the training video. After learning the interaction usage, the participant started a trial until they understood the interaction method. This training procedure took about 2–4 minutes per method. The participant then experienced each method using both the static and dynamic videos. After playing back each video as often as wanted, participants rated the experience (a combination of {method, video}) using the aforementioned measures. The participant was also encouraged to share comments on the interaction and interface after each experience. On average, the whole procedure took around 20 minutes.

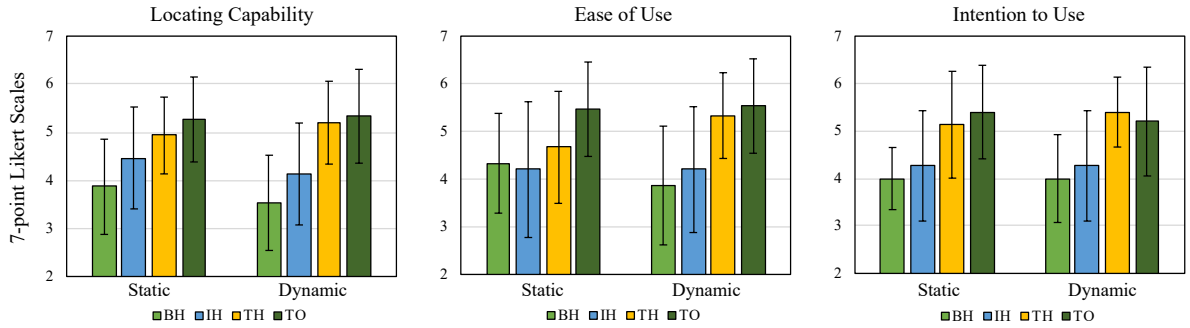


Figure 14: User study ratings on 7-point Likert scales with the error bars showing standard deviations. We use the following abbreviations: BH: *Baseline method with horizontal thumbnails*; IH: *Interactive360 [8] with horizontal thumbnails*; TH: *Transitioning360 with horizontal thumbnails (ours)*; TO: *Transitioning360 with Outside-In (ours)*. Note that our interaction methods (TH, TO) outperform the baselines (BH, IH) in most cases.

Table 2: Statistical significance analyses between interaction methods, summarized using mean±standard deviation. Asterisks indicate significant differences. Green entries indicate that the method in the row is better than the method in the column; red entries indicate worse results.

	Locating capability					**Ease of use**					**Intention to use**				
Method	M±SD	BH	IH	TH	TO	M±SD	BH	IH	TH	TO	M±SD	BH	IH	TH	TO
BH	3.70±0.98	—	**	***	***	4.10±1.33	—	—	***	***	4.00±0.62	—	—	***	***
IH [8]	4.30±1.11	**	—	**	***	4.20±1.82	—	—	**	***	4.27±1.31	—	—	***	***
TH (*ours*)	5.07±0.68	***	**	—	—	5.00±1.17	***	**	—	*	5.27±0.89	***	***	—	—
TO (*ours*)	5.30±0.84	***	***	—	—	5.50±0.95	***	***	*	—	5.30±1.11	***	***	—	—

* statistically significant ($p < 0.05$) ** statistically highly significant ($p < 0.01$) *** statistically strongly significant ($p < 0.001$)

8.5 Results

In total, we have collected 15 (participants) × 4 (methods) × 2 (videos) × 3 (questions) = 360 ratings. Figure 14 shows user ratings on the 7-point Likert scales. We further analyze the results with repeated measures ANOVA and paired t-tests, which revealed that there were no significant interaction effects between the video types and interactive method types. There was no significant main effect between static and dynamic videos, but significant main effects among interactive methods were found, in terms of locating capability, ease-of-use level and intention-to-use level.

For locating capability, Transitioning360 was rated significantly better than alternative methods, and the visualization styles do not affect the locating of objects in the scene. For ease of use, Transitioning360 with Outside-In visualization was rated the best. The ratings for "intention of use" indicated that participants were generally more willing to use Transitioning360 as their preferred tool for 360° video navigation, with either horizontal thumbnail or Outside-In visualization. Table 2 summarizes the significance analyses; full data are provided in the supplementary material.

8.6 Open Comments

We further collected comments from participants on the interaction methods. One participant reported that Interactive360 [8] required frequent use of the mouse to click and drag content searching, which was less convenient than simply clicking the arrow keys in TH or TO (our methods). Another participant commented that it is convenient to change view using arrow keys in TH and TO, while also being able to use the mouse to slightly adjust the view for more active exploration. One participant suggested adding a feature to change the mode among methods so that users can have more choices. Another participant proposed adding camera acceleration and deceleration to our transitioning methods TH and TO, to improve the visual effect.

One participant pointed out that the order of horizontal thumbnails below the main view in BH, IH and TH cannot reveal the spatial relationship of the views, which sometimes led to confusion about the spatial context. Several participants commented that in TO, the video thumbnails can sometimes occlude the contents in the main view and distract viewer's attention. The comments from participants were generally positive towards Transitioning360. Participants found the transitioning between views useful as it improved the convenience level.

9 Conclusion

In this paper, we presented Transtioning360, a new interactive 360° video navigation approach for 2D displays, which builds upon a coarse-to-fine joint optimization algorithm for multiple virtual NFoV camera path computation. The paths are content-aware and can locate and track potentially interesting and important 360° video content. The paths are also diverse to cover different important targets, and temporally stable. We introduced the joint objective of camera paths, and solved it using dynamic programming. To make the problem tractable, we proposed a coarse-to-fine strategy with paths initially solved at the coarse scale and further refined at the fine scale. NFoV videos corresponding to the paths can be easily experienced on 2D displays. Moreover, users can spatio-temporally transition between precomputed content-aware videos with just a single-clicking action rather than tedious manual view adjustments with many mouse clicking and dragging actions. We conducted a user study to evaluate the locating capability, ease of use and intention to use of the proposed interaction method with visualization interfaces. The results demonstrated that Transitioning360 is generally more preferred as a navigation tool with better locating capability and easier usage, and using the Outside-In visualization can enhance the Transitioning360 interaction.

9.1 Limitations and Future Work

Our method also has limitations. First, throughout our experiments, at most four cameras paths were computed for any 360° video. We observed that four cameras were generally sufficient as the horizontal FoV of each camera is about 90°, and so additional cameras would overlap more, which would degrade the view diversity. Nevertheless, an algorithm to compute more pleasant camera paths is

desirable. Second, the FoV of each camera was fixed throughout our experiments. If the person is close to the camera, only part of their body will be visible. Therefore, incorporating zooming would be an interesting extension. Finally, the content awareness in our method is estimated using saliency, motion and instance segmentation. Additional semantic cues, such as human-object-interactions (HOI), could be explored to improve content awareness.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (Project Number: 61902012 and 61932003), RCUK grant CAMERA (EP/M023281/1), and an EPSRC-UKRI Innovation Fellowship (EP/S001050/1).

REFERENCES

[1] M. Assens Reina, X. Giró-i-Nieto, K. McGuinness, and N. E. O'Connor. SaltiNet: Scan-path prediction on 360 degree images using saliency volumes. In *ICCV Workshops*, 2017. doi: 10.1109/ICCVW.2017.275

[2] A. Borji, M.-M. Cheng, Q. Hou, H. Jiang, and J. Li. Salient object detection: A survey. *Computational Visual Media*, 5(2):117–150, 2019. doi: 10.1007/s41095-019-0149-9

[3] H.-T. Cheng, C.-H. Chao, J.-D. Dong, H.-K. Wen, T.-L. Liu, and M. Sun. Cube padding for weakly-supervised saliency prediction in 360° videos. In *CVPR*, 2018. doi: 10.1109/CVPR.2018.00154

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(2):386–397, Feb. 2020. doi: 10.1109/TPAMI.2018.2844175

[5] H.-N. Hu, Y.-C. Lin, M.-Y. Liu, H.-T. Cheng, Y.-J. Chang, and M. Sun. Deep 360 pilot: Learning a deep agent for piloting through 360° sports video. In *CVPR*, 2017. doi: 10.1109/CVPR.2017.153

[6] J. Jung, B. Kim, J.-Y. Lee, B. Kim, and S. Lee. Robust upright adjustment of 360 spherical panoramas. *The Visual Computer*, 33(6):737–747, June 2017. doi: 10.1007/s00371-017-1368-7

[7] R. Jung, A. S. J. Lee, A. Ashtari, and J.-C. Bazin. Deep360Up: A deep learning-based approach for automatic VR image upright adjustment. In *IEEE VR*, 2019. doi: 10.1109/VR.2019.8798326

[8] K. Kang and S. Cho. Interactive and automatic navigation for 360° video playback. *ACM Trans. Graph.*, 38(4):108:1–11, July 2019. doi: 10.1145/3306346.3323046

[9] J. Kopf. 360° video stabilization. *ACM Trans. Graph.*, 35(6):195:1–9, Nov. 2016. doi: 10.1145/2980179.2982405

[10] G. A. Koulieris, K. Akşit, M. Stengel, R. K. Mantiuk, K. Mania, and C. Richardt. Near-eye display and tracking technologies for virtual and augmented reality. *Comput. Graph. Forum*, 38(2):493–519, May 2019. doi: 10.1111/cgf.13654

[11] W.-S. Lai, Y. Huang, N. Joshi, C. Buehler, M.-H. Yang, and S. B. Kang. Semantic-driven generation of hyperlapse from 360° video. *IEEE Trans. Vis. Comput. Graph.*, 24(9):2610–2621, Sept. 2018. doi: 10.1109/TVCG.2017.2750671

[12] S. Lee, J. Sung, Y. Yu, and G. Kim. A memory network approach for story-based temporal summarization of 360° videos. In *CVPR*, 2018. doi: 10.1109/CVPR.2018.00153

[13] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. doi: 10.1007/978-3-319-10602-1_48

[14] Y.-C. Lin, Y.-J. Chang, H.-N. Hu, H.-T. Cheng, C.-W. Huang, and M. Sun. Tell me where to look: Investigating ways for assisting focus in 360° video. In *CHI*, 2017. doi: 10.1145/3025453.3025757

[15] Y.-T. Lin, Y.-C. Liao, S.-Y. Teng, Y.-J. Chung, L. Chan, and B.-Y. Chen. outside-in.

[16] C. Liu, J. Yuen, and A. Torralba. SIFT flow: Dense correspondence across scenes and its applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):978–994, May 2011. doi: 10.1109/TPAMI.2010.147

[17] A. Mahzari, A. Taghavi Nasrabadi, A. Samiei, and R. Prakash. FoV-aware edge caching for adaptive 360° video streaming. In *MULTIMEDIA*, 2018. doi: 10.1145/3240508.3240680

[18] A. Pavel, B. Hartmann, and M. Agrawala. Shot orientation controls for interactive cinematography with 360 video. In *UIST*, 2017. doi: 10.1145/3126594.3126636

[19] T. Rhee, L. Petikam, B. Allen, and A. Chalmers. Mr360: Mixed reality rendering for 360° panoramic videos. *IEEE Transactions on Visualization and Computer Graphics*, 23(4):1379–1388, 2017.

[20] C. Richardt, J. Tompkin, and G. Wetzstein. Capture, reconstruction, and representation of the visual real world for virtual reality. In M. Magnor and A. Sorkine-Hornung, eds., *Real VR – Immersive Digital Reality*, pp. 3–32. Springer, 2020. doi: 10.1007/978-3-030-41816-8_1

[21] Y.-C. Su and K. Grauman. Making 360° video watchable in 2D: Learning videography for click free viewing. In *CVPR*, 2017. doi: 10.1109/CVPR.2017.150

[22] Y.-C. Su, D. Jayaraman, and K. Grauman. Pano2Vid: Automatic cinematography for watching 360° videos. In *ACCV*, 2016.

[23] C. Tang, O. Wang, F. Liu, and P. Tan. Joint stabilization and direction of 360° videos. *ACM Trans. Graph.*, 38(2):18:1–13, Mar. 2019. doi: 10.1145/3211889

[24] J. Tarko, J. Tompkin, and C. Richardt. Real-time virtual object insertion for moving 360° videos. In *VRCAI*, pp. 14:1–9, 2019. doi: 10.1145/3359997.3365708

[25] J. O. Wallgrün, M. M. Bagher, P. Sajjadi, and A. Klippel. A comparison of visual attention guiding approaches for 360° image-based VR tours. In *IEEE VR*, 2020. doi: 10.1109/VR46266.2020.00026

[26] M. Wang, X.-Q. Lyu, Y.-J. Li, and F.-L. Zhang. VR content creation and exploration with deep learning: A survey. *Computational Visual Media*, 6(1):3–28, 2020. doi: 10.1007/s41095-020-0162-z

[27] M. Wang, G.-Y. Yang, J.-K. Lin, S.-H. Zhang, A. Shamir, S.-P. Lu, and S.-M. Hu. Deep online video stabilization with multi-grid warping transformation learning. *IEEE Trans. Image Process.*, 28(5):2283–2292, 2019. doi: 10.1109/TIP.2018.2884280

[28] F. Zhou, S. B. Kang, and M. F. Cohen. Time-mapping using space-time saliency. In *CVPR*, 2014. doi: 10.1109/CVPR.2014.429