View-Dependent Video Textures for 360° Video

Sean J. Liu **Maneesh Agrawala**
Stanford University
{lsean, maneesh}@cs.stanford.edu

Stephen DiVerdi **Aaron Hertzmann**
Adobe Research
{diverdi, hertzman}@adobe.com

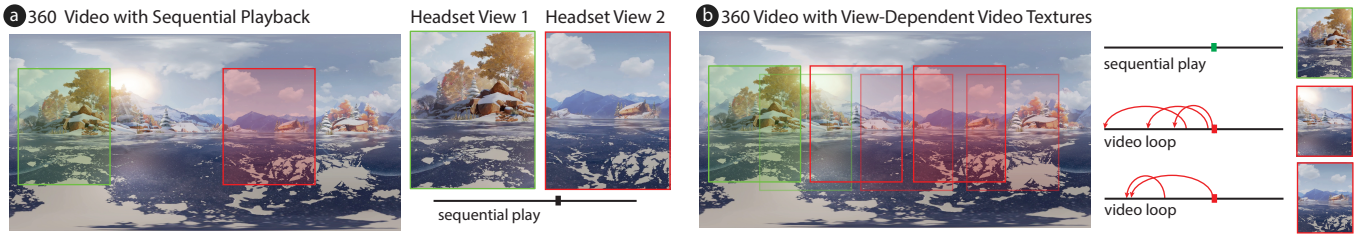

Figure 1. In 360° video, viewers can look anywhere at any time. In the opening scene of Invasion!, a rabbit emerges from a cave (a). In sequential playback, a viewer looking at the cave (green box) will see the rabbit emerge, whereas a viewer not looking at the cave (red box) will miss this event. We provide tools to guarantee that viewers see the region of interest (ROI) at the correct timecode to witness the event (b). We introduce the concept of gated clips, where playback only continues if the viewer satisfies a condition related to the ROI (green boxes). Otherwise, our player loops the video using view-dependent video textures (red boxes).

ABSTRACT

A major concern for filmmakers creating 360° video is ensuring that the viewer does not miss important narrative elements because they are looking in the wrong direction. This paper introduces **gated clips** which do not play the video past a gate time until a filmmaker-defined viewer gaze condition is met, such as looking at a specific region of interest (ROI). Until the condition is met, we seamlessly loop video playback using **view-dependent video textures**, a new variant of standard video textures that adapt the looping behavior to the portion of the scene that is within the viewer's field of view. We use our desktop GUI to edit live action and computer animated 360° videos. In a user study with casual viewers, participants prefer our looping videos over the standard versions and are able to successfully see all of the looping videos' ROIs without fear of missing important narrative content.

CCS Concepts

•Human-centered computing → Virtual reality; •Computing methodologies → Computer vision problems;

Author Keywords

view-dependent video texture, 360° video, virtual reality, cinematography, gaze guidance

UIST '19, October 20-23, 2019, New Orleans, LA, USA.

ACM ISBN 978-1-4503-6816-2/19/10 ...$15.00.
http://dx.doi.org/10.1145/3332165.3347887

INTRODUCTION

The medium of 360° video provides new artistic opportunities for filmmakers, allowing them to create videos with a greater sense of immersion and engagement than with conventional video. It also presents new challenges. In traditional cinematography, the director has full control over the camera orientation, field of view, zoom, and focus at all times. Traditional filmmakers use these controls to drive the narrative, ensuring that the viewer sees each important story element at the right time. With 360° videos, however, directors no longer have this control, and viewers can look in any direction at any time. As a result, viewers may miss important story content and become lost or confused as the story progresses.

For example, in the animated short "Invasion!" [7], the story begins with an establishing shot placing the viewer in the middle of an icy lake (Figure 1). Initially, the viewer is given time to look around and become familiar with their surroundings. A rabbit eventually emerges from a small cave. However, if the viewer is not looking at the cave entrance when the rabbit emerges, they will not see the rabbit, or what it does next. A second type of example is in the "Stranger Things: The VR Experience" [33] short film. As the tension rises, the viewer answers the phone, and is told to turn around. Then, a monster attacks from the direction opposite the phone. If the viewer does not turn around fast enough, they miss the monster attacking. A third type of example occurs in "Wild: The Experience" [14], where the viewer is placed between a hiker and an empty rock, on which a "ghost" appears only if the viewer is not looking at the rock. The hiker and the sound of her breathing is intended to get the viewer's attention away from the rock, so that the ghost can appear outside the viewer's field of view. However, if the viewer never looks away from the rock, the video player will reach the end of the video without the ghost ever appearing. Although the directors of these examples include passive gaze guidance techniques, such as

audio cues, to encourage the viewer to look in a particular direction, none of the techniques are foolproof.

This paper proposes a new filmmaking technique we call *gated clips*, designed to ensure that a viewer sees key elements of the narrative in a 360° video. Using our technique, the filmmaker can author a *gate* which ensures that playback may only proceed past a *gate time* only if a filmmaker-defined viewer gaze condition is met, such as looking at a specific region of interest (ROI). Viewing such gated clips requires a new kind of video player that seamlessly loops the 360° video playback until the gate condition is met.

For example, in the "Invasion!" short, we can place a gate just before the rabbit emerges from the cave and treat the gate ROI as the cave entrance. Our video player then seamlessly loops the video playback until the viewer is looking at the cave entrance when the rabbit is emerging and only proceeds past the gate at that time. Gated clips also enable new kinds of shots where action proceeds only when the viewer is looking away, e.g., in "Wild," our video player can ensure that the viewer has looked away from the rock before the ghost appears outside their view, and then that they have looked back at the rock to see the ghost.

In order to create gated clips, we introduce *view-dependent video textures* for 360° video. Our approach is based on video textures [40], a generalization of video looping, to allow the 360° video to seamlessly jump back to earlier frames, so that the viewer will not notice the looping. Conventional video texture algorithms only allow transitions between frames when the change is imperceptible anywhere in the frame. However, this criteria is too conservative for 360° video, where the entire view sphere is encoded in an equirectangular frame, since the change only needs to be imperceptible in the viewer's field of view (FOV). Our view-dependent video textures relax this constraint. We introduce a novel graph cut algorithm to convert a standard 360° video clip into a gated clip with such view-dependent video textures.

To prototype these ideas, we present a user interface for editing 360° videos with gated clips. Our interface is built on a conventional timeline interface, but with special shot types for gated clips. We demonstrate results on five different videos, four of them professionally produced and not intended for use with our technique. In our user study, 9 out of 11 users preferred videos with gating.

RELATED WORK

Our approach builds on three main areas of related work.

Interactive and looping video

Forms of interactive (or dynamic) video, where the video playback changes depending on viewer actions, have been explored for several decades. The first methods, including Movie-maps [31] and QuickTimeVR [11], allowed navigation in real and virtual environments, whereas some arcade video games, such as Space Ace [2] and Dragon's Lair [1], used interactive video for branching narratives.

Our method uses video textures to create video that loops seamlessly until certain conditions are met. Video textures were introduced by Schödl et al. [40], who demonstrated finding seamless non-repeating paths through short clips to create the experience of endlessly playing videos, as well as to drive video in different ways. This technique has been extended to loop video panoramas captured by a panning camera [3], motion capture videos of humans [15], videos orbiting around a moving object [26], and responsively looping video based on user interactions [23].

In order to increase the number of possibilities for seamless transition in a video texture, some methods have segmented the video into independently-moving regions [22,40], and then created separate video textures for each such region. In contrast, we introduce *view-dependent arcs*, specifically for applying video textures to 360° video. Our approach is complementary to segmentation; our method produces video textures by exploiting viewers' limited FOV in 360° video, whereas segmentation loops the entire scene by looping and compositing different moving parts of the scene.

Motion segmentation methods are also used in the construction of cinemagraphs [4, 5, 25, 27, 28], photograph-like images with some moving elements. These methods create pixelwise textures that are not appropriate for our structured scenes. Our view-dependent arcs are also akin to Chenney and Forsyth's [12] view-dependent approach to accelerating physical simulation, by eliding computation for scene elements not in view.

Gaze guidance

While 360° video provides a new dimension of viewer interactivity and agency, this freedom is accompanied by users struggling with not knowing where to look [35, 45]. Methods for guiding or forcing the viewer to look at a particular region of interest (ROI) are called *gaze guidance*. Our approach is complementary to gaze guidance, and we expect our method to be used together with gaze guidance techniques, such as audio cues to attract the viewer's attention. For a survey and taxonomy of gaze guidance techniques, see Nielsen et al. [34].

Subtle gaze guidance techniques provide visual and audio cues that attempt to guide the viewer's gaze without breaking immersion. This is particularly important for narrative content where a filmmaker wishes to keep the viewer fully absorbed by the story. Conventional filmmakers aggressively use subtle cues to guide the viewer's attention, and these effects are also being explored by 360° filmmakers and researchers [38, 42]. Several techniques apply subtle gaze guidance to normal field of view (FOV) video, i.e., on a desktop computer monitor. These methods include gradually blurring non-ROI parts of the image [21], applying shallow depth of field [44], modulating the scene's visual saliency [46], and presenting a small flickering distractor in the viewer's peripheral vision in the direction of the ROI [6]. For in-headset VR, subtle techniques in the literature include inducing optical flow in peripheral vision [9] and flickering elements in peripheral vision [18]. Grogorick et al. [19] perform an evaluation comparing different subtle guidance techniques. Each of these techniques finds some success in guiding viewer gaze, but none guarantee that viewers will see critical moments.

Figure 2. Our prototype desktop video editing interface with a *gated clip*. The upper left pane shows a preview of the headset FOV, which is output live to the Oculus Rift. The upper right pane shows the full equirectangular view and marks the current headset FOV (pink border). The video timeline on the bottom acts as a conventional video editor, with each dark blue rectangle representing a clip. The first clip is a gated clip (white border). Filmmakers specify a *gate timecode* (red vertical line on timeline), a *ROI* (green box) on the equirectangular frame, as well as other parameters shown in the settings box below the clip. View-dependent arcs are shown as backward arcs (red arrows) on top of the gated clip. In this example, the ROI is the aliens, and the *gate condition* is a *lookat gate*, i.e., playback may not advance past the gate timecode unless the viewer is looking at the ROI. To avoid static loops and reduce arcs with visual artifacts, the filmmaker can set thresholds on the length of arcs and on the perceptual difference of arc transitions. After setting all thresholds, the filmmaker can generate the view-dependent arcs by loading in (preprocessed) arc costs and applying the gate. To have viewers jump to the gate timecode as soon as they see the ROI, the filmmaker can also enable forward arcs that are under a perceptual threshold. Finally, the filmmaker can choose to cross-fade the audio during loop transitions or choose to mute the audio entirely.

Several methods actively control the viewer's gaze direction to focus on a ROI; depending on how this is implemented, it can substantially break immersion. This includes directly rotating the scene [29] or rotating the user in a motorized swivel chair [20]. Attempts to hide this rotation include applying very slow rotations [43], or applying gains to the user's own rotation [48]. Reorienting the scene during cuts [36] maintains immersion but can only be applied at cuts. A picture-in-picture visualization may also be shown of the ROI so that viewers always see important content [30].

View-dependent 360° video and animation

Our work is inspired by several short 360° video and animations that use different kinds of view-dependent playback. Several animated shorts use forms of gating to pause the action (without pausing the motion or audio), either waiting for the viewer to look at something, or to look away from something, including "Buggy Night," "Piggy," and "The Simpsons: Planet of the Couches," from Google Spotlight Stories[1] [17]. In "Buggy Night" and "Piggy," the viewer is essentially part of the story, either scaring the flies by looking at them in "Buggy Night," or catching Piggy stealing a cake. The ending of the "Batman: Arkham VR" [37] video game uses an offscreen action effect to simulate the player's hallucinations. Disney's "Cycles" [47] animation fades out the lighting and action whenever the viewer looks away. Each of these examples is animated, and, presumably, carefully hand-authored to run in real-time rendering engines; whereas our method can work with live action video with comparatively lightweight authoring effort.

[1]These shorts are viewable in the Google Spotlight Stories app for iOS and Android. On YouTube, they do not have view-dependent playback. "Piggy" also has a separate app in the Steam Store.

We are aware of only one example using live action video: "Wild: The Experience" [14] from Felix & Paul Studios, described in the introduction. In the original short, a character may appear and disappear after certain times, depending on the viewer's head movements. The video plays for a fixed duration regardless; there is no gating, and thus no guarantee that the viewer will see significant events.

GATED CLIPS

As described in the Introduction, it is easy for viewers to miss important moments in 360° experiences; making sure that the viewer sees the right things is a common concern among the 360° filmmakers we have spoken with. Filmmakers are accustomed to controlling the viewer's gaze with shot framing, zoom, depth of field, and camera movements; none of these techniques are available in 360° video, and, worse, the viewer can be looking in away when an important moment starts.

To address this problem, we introduce the *gated clip* (Figures 1 and 2). A gated clip is a portion of a video that loops until the viewer satisfies some viewing condition, such as looking in a specific direction. A gated clip is comprised of the following elements: The *gate timecode* is a specific frame in the clip. The video may only progress beyond the gate timecode when a *gate condition* is met. We use the term *timecode* to describe a frame index into the video timeline, and distinguish such timecodes from playback time, which may differ on each viewing due to looping.

Types of gates

We consider two types of gate conditions: (1) A *lookat gate* specifies that the viewer must see a specific region of interest (ROI) at the gate timecode for the video playback to proceed, i.e., that the ROI is within the viewer's field of view (Figures 1 and 2). (2) An *offscreen gate* is the inverse condition, i.e.

Figure 3. Discretized view directions. This figure visualizes the FOVs of $|\mathbb{V}| = 6$ discretized view directions evenly spaced around the equator on an equirectangular frame from an example video. In our implementation, we used $|\mathbb{V}| = 40$. The horizontal FOV of each discretized view matches the horizontal FOV of the Oculus Rift and covers the full vertical range of the video.

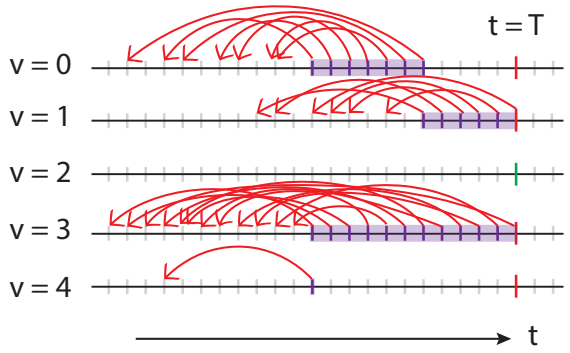

Figure 4. Visualization of view-dependent video textures for a gated clip. Each row corresponds to one of the discretized viewing directions (here we show the first five views only). The gate timecode is $t = T$, and $v = 2$ is the only view that satisfies the gate condition (green vertical line); other views do not (red vertical lines). Red arrows show the computed backward arcs. Tick marks correspond to frames, and purple ones are frames from which there are seamless backward arcs, i.e. arcs in which the transition frames have low perceptual difference.

it specifies that the viewer *must not* see the ROI at the gate timecode for video playback to proceed.

We give examples of three narrative use cases for these two types of gate conditions. A common trope in 360° filmmaking, such as in "Invasion!," (see Introduction section) is that the viewer is given a considerable amount of time to become familiar with a new environment before the first main action begins; this is akin to an establishing shot in conventional filmmaking. In such cases, authors could use a lookat gate when the viewer is initially placed in a new environment, with the gate timecode and ROI located at the time and spatial location where the main action begins. In the "Invasion!" example, the gate timecode would be placed at the time the rabbit first appears, and the ROI would be placed at the cave entrance. Another narrative use case is when the viewer is supposed to move their head in the middle of the story. For example, in "Stranger Things," the viewer is instructed to turn their head around in order to see the monster at the end of the hallway. In such cases, the author could again place a lookat gate right before the next action starts and set the ROI on the new target location. In this example, the ROI would be on the monster, and the gate timecode would be right before the monster attacks. Finally, for surprising entrances and disappearances, the author could use offscreen gates to ensure that the viewer does not see the actual appearance or disappearance. For example, in the "Wild" example, the author could place an offscreen gate right before the ghost appears and mark the rock as the ROI. That way, the viewer must look away for the ghost to appear offscreen, and then the viewer could turn to look at the rock and see the ghost.

VIEW-DEPENDENT VIDEO TEXTURES

Simply looping the video until the gate condition is met would often break immersion, because of *ghosting*—objects appearing, disappearing, or jumping—in the transition from the last frame of the loop to the first frame. Instead, we create seamlessly looping gated clips by generalizing the approach of *video textures* [40] to handle 360° video and gating.

A video texture is a video clip that can be played back endlessly by adding seamless transitions between non-sequential frames. Each transition occurs along an *arc* (t,t'), which transitions playback from frame t to frame t', over a user-specified cross-dissolve interval (our implementation uses a fixed 0.5 second cross-dissolve). By selecting arcs carefully, we can create seamless playback, in which ghosting is minimized. There are three types of arcs: *sequential arcs* $(t,t+1)$, used in normal playback; *backward arcs* $(t,t'),t' < t$, and *forward arcs* $(t,t'),t' > t+1$.

Constructing video textures normally involves finding *seamless arcs* (t,t') between non-sequential frames, such that cross-dissolving the video from frame t to frame t' is imperceptible to the viewer. The conventional approach is to measure some perceptual distance metric between the two frames. However, measuring the image distance for the entire 360° equirectangular image is too conservative, because viewers only see a small portion of the scene at any time; for example, typical VR headsets only have roughly an 80° horizontal field of view.

The core idea of view-dependent video textures is the use of *view-dependent arcs*. View-dependent arcs allow specific runtime transitions as long as the viewer is looking in a particular range of directions. These transitions are selected to minimize perceptual difference *within* the field of view. Specifically, we discretize the view-sphere with a fixed set of directions $v \in \mathbb{V}$. A view-dependent arc is then represented as a triplet (v,t,t') and is computed based on the pixels visible within v.

We chose a discretization of $|\mathbb{V}| = 40$ views. The views are equally-spaced around the equator (Figures 3 and 4). In our test videos, there is more motion around the equator and virtually no motion near the poles, so we chose the view FOV to have higher granularity of coverage in the horizontal direction than in the vertical direction. Thus, we set the horizontal FOV of each view to match the horizontal FOV of the Oculus Rift headset and the vertical FOV to cover the video's full vertical height ($w = 80.65°$, $h = 180°$). The FOVs of adjacent views overlap by 71.65°, with their centers 9° apart. Additional views could easily be added to the discretization, or coverage

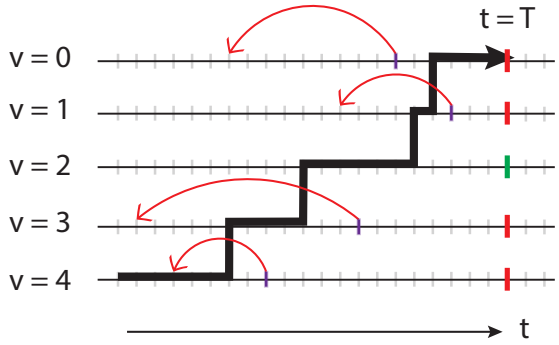

Figure 5. One naive solution is to find one loop (i.e., one backward transition arc) for each view that does not satisfy the gate condition. In this example, the gate timecode is $t = T$, and the view $v = 2$ satisfies the gate condition. All other views include one backward arc forming a loop. The thick black line shows an example viewer's head trajectory through the views over time (slowly turning their head from $v = 4$ to $v = 0$). Unfortunately the viewer can still get past the gate time through another view $v \neq 2$ via certain head motions, so the naive solution does not provide a guarantee that the gate condition is met before playback progresses.

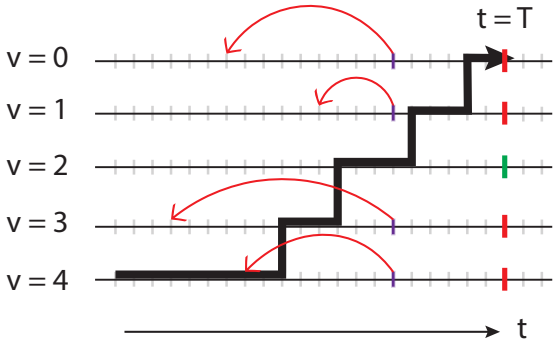

Figure 6. Another naive solution is to find one loop (i.e. one backward transition arc) for each view that does not satisfy the gate condition, such that all the arcs originate at the same timecode. In this example, the gate timecode is $t = T$, and the view $v = 2$ satisfies the gate condition. Unfortunately, if the originating timecode for all arcs does not equal the gate timecode T, then the viewer can still get past the gate time through a view $v \neq 2$, as shown by the thick black line, which gives an example of viewer's head trajectory (slowly turning their head from $v = 4$ to $v = 0$) that gets past the gate without satisfying the gate condition.

areas expanded as needed. See Discussion Section for more details on the trade-off between the number of discrete views and the FOV size of each view.

GENERATING VIEW-DEPENDENT TEXTURES

Given a user-specified gate and an existing video clip, we wish to generate a view-dependent texture that satisfies the following properties: (1) The video proceeds past the gate timecode only if the viewer satisfies the gate condition. (2) The gate timecode is reachable if the viewer is looking in a direction that satisfies the gate condition (i.e., for lookat gates, in a direction where ROI is visible; for offscreen gates, in a direction where ROI is *not* visible). (3) The transitions taken along arcs minimize or eliminate ghosting. (4) All arcs satisfy user-set thresholds on the length of the arcs and on the perceptual difference of arc transitions. In order to discourage repetitious or static loops, the arc length threshold requires all backward arcs to be longer than a minimum threshold duration. To minimize ghosting and visual artifacts, our tool allows authors to specify a perceptual threshold, which is the highest perceptual difference of frames that arcs can have in order to be considered seamless arcs.

Given these constraints, our goal is to specify the playback behavior for each combination of frame t and discretized view direction v. There are only two possibilities from each (v,t): either play forward to $t+1$, or transition backward to some previous timecode $t' < t$. Our algorithm outputs the action (play forward to next frame or transition backward to an earlier frame) to take from each (v,t). Then, during playback, our player looks up the run-time frame t and the nearest discretized v, and follows the selected behavior at (v,t).

A naive solution is to create a separate video loop (i.e., one backward transition arc) for each view direction, except for those that satisfy the gate condition. That is, for each view direction $v \in \mathbb{V}$, we could independently search for a backward arc (v,t,t') that minimizes ghosting. Unfortunately, this approach does not guarantee that the viewer satisfies the gate condition. As illustrated in Figure 5, it is possible for viewers to move their heads in a way that allows them to pass the gate timecode through a view that does not satisfy the gate condition. We also considered a version of this approach which finds a single timecode t for all backward arcs (with independent destination timecodes t'), but this approach similarly does not guarantee that viewers satisfy the gate condition. If t is earlier than the gate timecode, viewers can still get past the gate timecode T through a view that does not satisfy the gate condition (Figure 6). Requiring t to occur at the gate timecode is too restrictive to work for general videos. For example, if a view (not satisfying the gate condition) is static for all frames $t < T$, but an object in the view moved at $t = T$, then the view would not have any seamless backward arcs originating at $t = T$, even though there are many pairs of frames before T that can form seamless arcs.

Graph Cut Formulation

We formulate the problem in terms of graph theory, specifically an *s-t* graph cut [8, 16, 41]. The graph construction represents playback as a state machine, but with some modifications, so that a minimal graph cut produces a solution to our problem of generating view-dependent video textures for gated clips.

The graph includes one node (v,t) for each pair of view direction and frame in the clip, from the start frame 0 to frame $T+1$, which is the frame immediately after the gate timecode. Let $v \in \mathbb{H}$ be the set of views that satisfy the gate condition. For lookat gates, $\mathbb{H}$ is the set of views in which the ROI is visible; for offscreen gates, $\mathbb{H}$ is the set of views in which the ROI is *not* visible. We call (v,T), where $v \in \mathbb{H}$, *gate nodes*.

Graph Partition. Our goal is to partition the graph into two parts, a "safe zone," which the viewer must stay within before they satisfy the gate condition, and an "unsafe zone" that the

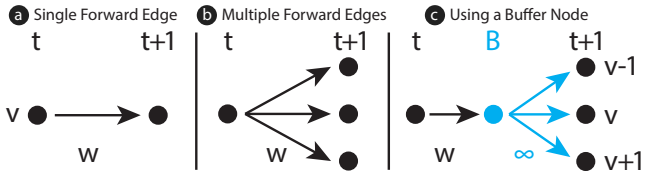

Figure 7. (a) A single forward edge between consecutive nodes $(v,t) \rightarrow (v,t+1)$ represents normal video playback of sequential frames but does not model head rotation. (b) Adding forward edges to adjacent viewing directions models head rotation below some velocity—i.e. $(v,t) \rightarrow (v',t+1)$, for all $v' \in \mathbb{N}(v)$. But cutting these edges corresponds to disallowing some head motions, which we cannot control. (c) A buffer node $(v,t)_B$ (cyan) creates one edge (black) that can be cut (to remove the video frame advance for this view), while infinite-weight edges (cyan) cannot be cut (to properly model free user head rotation). The edge weight w is designed to strongly prefer arcs with perceptual error below the user-set threshold. More details of how w is determined is in the Appendix.

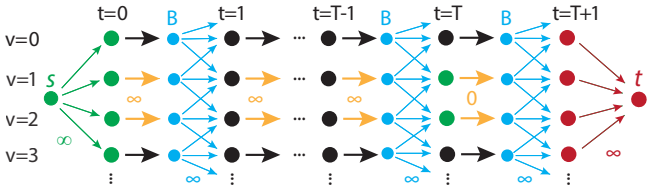

Figure 8. Our graph representation of a gated clip. The gate frame is $t = T$ and the views that satisfy the gate condition are $\mathbb{H} = \{1,2\}$. We add buffer nodes (cyan) between consecutive frame nodes. There are infinite-weighted edges from each buffer node to possible views that viewers might see at the next frame. In this figure, $n = 1$ for the number of adjacent views that viewers can visit in one frame, but in our result videos we used $n = 2$. Buffer edges connect each frame node to its buffer node. Buffer edges of $\{(v,t)|v \in \mathbb{H}, t < T\}$ have infinite weight, while buffer edges of gate notes $\{(v,T)|v \in \mathbb{H}\}$ have a weight of 0. Source node s is connected to the starting frame nodes $t = 0$ in each view, while terminal node t is connected to all nodes at $t = T+1$. After performing graph cut, the initial nodes and gate nodes (green) are partitioned from end nodes (red).

viewer must *not* visit before the gate condition is met. The safe zone must include all nodes at the starting frame $(v,0)$ as well as the gate nodes, because the viewer can start in any view direction, and the viewer must be able to visit the gate nodes in order to satisfy the gate condition. The unsafe zone must include nodes $(v,T+1)$ for all v, because viewers should not visit the frame immediately after the gate timecode if the gate condition has not been satisfied. With this construction, the viewer can exit the safe zone of the graph only by passing through a gate node. Otherwise, for boundary nodes in the safe zone that border the unsafe zone, our graph cut algorithm finds seamless backward arcs from which to transition back in time, so that the viewer does not enter the unsafe zone. If the viewer is at a node in the unsafe zone before the gate condition is met, they might see ghosting and/or pass through the gate timecode in a view $v \notin \mathbb{H}$.

Accounting for Head Motion. At any instant, the viewer may rotate their head. Hence, from any node (v,t), the view direction at the next time instant may be from a neighbor set $\mathbb{N}(v)$, determined as a function of the field of view, the view discretization, and how fast viewers typically rotate their heads. Based on the work of Bussone [10], we assume a typical maximum head velocity movement of 9.03 rad/s. For $|\mathbb{V}| = 40$ and 30fps video, this means that viewers can move across $n = 2$ adjacent views in either direction over the course of one frame interval, and therefore the neighbor set $\mathbb{N}(v)$, of view v contains 5 views including v.

One might imagine representing sequential playback with allowance for head motion by including an edge from each node $(v,t) \rightarrow (v',t+1)$, for all $v' \in \mathbb{N}(v)$. However, this approach would allow the graph cut algorithm to cut some of these edges and not others, which would correspond to allowing some head movements and not others. Since we cannot control the viewer's head movements, we cannot use such a representation. Hence, we introduce *buffer nodes* $(v,t)_B$ between consecutive nodes (Figure 7). From each node (v,t), we insert an edge to its buffer node $(v,t)_B$, and, from the buffer node, we add edges to the subsequent nodes $(v',t+1)$ for $v' \in \mathbb{N}(v)$. The edges $(v,t) \rightarrow (v,t)_B$ are called *buffer edges*.

Cutting a buffer edge indicates that, in the output video texture, the corresponding sequential arc $(v,t) \rightarrow (v,t+1)$ is omitted from the video texture, and that a backward arc must be taken whenever (v,t) is reached.

Edge Weights. The weight of a buffer edge $(v,t) \rightarrow (v,t)_B$ depends on the best backward arc available from the node (v,t), since some backward arcs may introduce more ghosting than others. The buffer edge weight is designed to strongly prefer arcs with perceptual error below the user-set threshold. Details of how we determine buffer edge weights are given in the Appendix. Edges connecting buffer nodes to subsequent nodes $(v,t)_B \rightarrow (v',t+1)$ for $v' \in \mathbb{N}(v)$ represent the set of views a viewer might transition into due to head movement. Since we cannot control viewer's head motion, these edges should not be cut, so we assign them a weight of infinity.

If the viewer is looking in a direction $v \in \mathbb{H}$ that satisfies the gate condition, they should be able to reach the gate node (v,T). In other words, all nodes (v,t), where $v \in \mathbb{H}$, $t < T$, should play forward (i.e., not traverse backward arcs). Thus, the buffer edges of these nodes should not be cut, so we set their weights to infinity.

The *s-t* graph cut formulation involves a source node **s**, which we connect with infinite-weight edges to the nodes that must be in the safe zone, i.e., initial nodes $(v,0)$ for all view directions $v \in \mathbb{V}$. We do not explicitly connect **s** to gate nodes, because they are guaranteed to be partitioned into the safe zone due to the infinite-weight buffer edges $(v,t) \rightarrow (v,t)_B, v \in \mathbb{H}, t < T$. The sink node **t** is connected with infinite-weight edges to all nodes $(v,T+1)$, which must be in the unsafe zone. The complete gated clip graph is shown in Figure 8.

In order to perform the s-t graph cut, we need at least one buffer edge in each $v \in \mathbb{H}$ to have non-infinite weight, so we set buffer edge weights of gate nodes $(v,T) \rightarrow (v,T)_B$ to 0. Consequently, edges $(v,T) \rightarrow (v,T)_B$ are always cut for $v \in \mathbb{H}$. Normally, cutting a buffer edge $(v,t) \rightarrow (v,t)_B$ indicates that a backward arc must be taken whenever (v,t) is reached. However, when the viewer reaches one of the gate nodes (v,T),

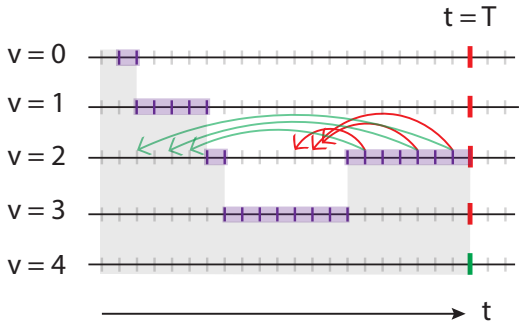

Figure 9. The graph cut algorithm may inadvertently create backward arcs (red arrows) into the "unsafe zone" when there are discontiguous cuts along a viewing direction. Our post-processing step replaces those backward arcs with new backward arcs (green arrows) that terminate in the "safe zone." The safe zone includes the both gray and purple shaded regions. However, since the purple frames border the unsafe zone, it is possible for viewers to turn their head into the unsafe zone as the player transitions into a purple frame, so our heuristic looks for replacement arcs that end in the gray shaded area.

where $(v,T) \rightarrow (v,T)_B$ is cut, we simply keep playing forward to pass the gate, instead of taking a backward arc.

Properties of Cut. The graph-cut algorithm solves for the set of buffer edges to remove with minimum total cost, such that the sink node **t** is not reachable from the source node **s**. This partition corresponds to segmenting the graph into a safe zone, including the start nodes and the gate nodes, and an unsafe zone, which include paths that violate the gate condition. Our implementation uses the min-cut solver of Boykov and Kolmogorov [8] and takes an average of 0.14 seconds to compute the cut for the clips we have tested (Table 1).

Postprocessing

Once we run graph-cut on the graph, our video player extracts from the resulting partition a binary decision for each node: whether to (1) play forward sequentially from that frame or to (2) take a backward arc from that frame. If a buffer edge $(v,t) \rightarrow (v,t)_B$ is *not* cut, the video player plays forward sequentially from (v,t). As discussed in the Edge Weights section, the weight of a buffer edge $(v,t) \rightarrow (v,t)_B$ depends on the best backward arc available from the node (v,t). So if a buffer edge $(v,t) \rightarrow (v,t)_B$ is cut, then the video player traverses the best backward arc from (v,t).

It is possible for our graph cut algorithm to cut edges and produce backward arcs that end in the unsafe zone, i.e., a node that the viewer is not meant to reach before satisfying the gate condition. As shown in Figure 9, when there are discontiguous cuts along a viewing direction, it is possible for some backward arcs in that view to end on a node which is partitioned into the unsafe zone. We use the following heuristic post-process to correct these cases. We first identify the origin timecode t_C of the earliest backward arc along view direction v ($t_C = \min_{\{(v,t,t')\}} t$). We are guaranteed that all nodes before this time are in the "safe zone." Thus, we replace each backward arc (v,t,t') that ends after t_C ($t' > t_C$), with the best backward arc from (v,t) that ends before t_C. This approach can produce

Figure 10. Stills from our example videos. The first four were professionally-created videos, not intended for use with gated clips. The fifth we shot for this project.

video textures with some poor backward arcs; we highlight such poor backward arcs in the user interface, and the user may perform further adjustments to generate better results, i.e., re-run the algorithm with different parameters (e.g., gate timecode, ROI, arc length) .

Forward Arcs to the Gate

Our graph-cut algorithm creates backward arcs in views that do not satisfy the gate condition, i.e., $v \notin \mathbb{H}$. If the viewer looks at a view $v \in \mathbb{H}$, by default our player just plays normally until the viewer reaches the gate timecode. However, this may take some time, and the filmmaker may want the viewer to get to the gate timecode as soon as they look in the right direction. Thus, we provide filmmakers with the option to add forward arcs to views v that satisfy the gate condition. The forward arcs allow playback to jump directly to the gate when the viewer is in a view $v \in \mathbb{H}$ without forcing them to wait for the remaining duration of the gated clip. If the author selects this option, our system automatically adds forward arcs to all nodes $(v,t), v \in \mathbb{H}, t < T - 0.5$ sec for which the transition cost from (v,t) to $(v, T - 0.5 \text{ sec})$ is below a user-set perceptual threshold. We set forward arcs to jump to 0.5 seconds before the gate timecode to allow the transition cross-dissolve to finish by the time the viewer gets to the gate.

EXAMPLE VIDEOS

We used our editing tool to add gated clips to five 360° videos (Figure 10). We selected a range of video genres and scenarios and varied our gate types to demonstrate a range of narrative use cases, detailed in Table 1. We created four videos (Invasion!, Stranger Things, Wild, and Lions) based on existing, professional videos, and shot the fifth video (Murder Mystery) ourselves. Note that the professional videos were not shot with gating in mind; we added gating in order to demonstrate

our method. We cut each video down to one or two minutes in length. For each video we created three gated clips and made sure to place the gate at important story events.

We authored audio manually for the gated clips. By default, we cross-dissolved the audio during transitions, just as we cross-dissolved the video; we used this approach for "Stranger Things," "Wild," and "Murder Mystery." For "Lions," the narrator sometimes speaks during a gated clip. To avoid looping the narration, we played the audio normally (without transitions), separate from the visual content which may be looping. If the gated clip audio ended before the viewer passed the gate, we paused the audio until the viewer did, after which we resumed audio with the next clip. For "Invasion!", the original soundtrack includes music; we found that audio dissolves were jarring, so we muted the audio entirely.

Invasion! [7]. We added lookat gates to focus the user's attention at three key moments: the rabbit's entrance in the opening scene, the aliens' comedic entrance from the spaceship, and the aliens' attempt to attack the rabbit. The lookat gates help pace the story as the viewer looks back and forth between the rabbit and the aliens.

Stranger Things [33]. In this video, the viewer starts out in the living room. The camera then automatically moves first towards the dining room, and then in an opposite direction down a hallway. We used a lookat gate to ensure that viewers look at the dining room and down the hallway before the camera starts moving, so that they are looking in the direction they move towards. Otherwise, the unanticipated camera motion could be confusing and disorienting. We used a lookat gate to ensure that the viewer turns around before the monster attacks the viewer.

Wild [14]. In this video, a hiker rests on a rock and sees the "ghost" of her mother, who appears and disappears opposite the hiker. The viewer must look back and forth to see one and then the other. We added two lookat gates; one for the hiker when the ghost appears, so that viewers see the main character and do not witness the ghost's appearance, and subsequently one for the ghost. Finally, we added an offscreen gate with forward arcs, so the ghost immediately disappears when the viewer looks away.

National Geographic Lions [32]. In this documentary, the narrator occasionally refers to specific lions within a group, who each briefly become the main character. We added lookat gates to wait for the viewer to look at the correct lion before allowing the narration for that lion to begin. In addition, we added a lookat gate right before a lion attacked another lion, to ensure viewers see this important action.

Murder Mystery. This video is similar to Wild, in that a ghost appears opposite from the main character (with the viewer in-between the characters) and disappears when the character looks away; however, there is more background motion which makes looping more difficult. We add a lookat and an offscreen gate for the ghost's appearance and disappearance, as well as another lookat gate for the position where the ghost was standing, so the viewer sees that the ghost has vanished.

Video	Genre	Length (sec), Type					
		1^{st}		2^{nd}		3^{rd}	
Invasion!	Comedy	7	L	7	L	4	L
Lions	Docu.	7	L	4	L	5	L
Stranger Things	Horror	4	L	5	L	4	L
Wild	Drama	10	L	3	L	7	O*
Murder Mystery	Mystery	6	L*	4	O*	7	L*

Table 1. Summary of example videos. For each video, we added three gated clips. We show the length (in seconds) and type of each gated clip. L: lookat gate, O: offscreen gate, *: enabled forward arcs.

Our source code and the gated clip metadata used to produce these examples are available at the project website: https://lseancs.github.io/viewdepvrtextures/

USER STUDY

In order to understand the effects of gated clips, we asked viewers to watch the five videos described in the previous section, and conducted a study to obtain qualitative feedback on their viewing experience. While the videos we used include some passive gaze guidance cues, we did not explicitly compare our method to passive (e.g., Nielsen et al. [34]) or active ([20,29,36]) gaze guidance techniques, because our method is complementary to them. Our method guarantees viewers see the ROI, whereas passive techniques do not. Active guidance techniques guarantee viewers see a ROI, but they also limit viewer interaction and can reduce immersion, as Nielsen et al. observed, whereas our method does not.

For the study, we used gated clips produced by an earlier version of our system, in which the vertical FOV of each discretized view was equal to the Oculus Rift FOV ($h = 96.02°$), instead of the the full vertical range of $h = 180°$. Using a vertical FOV smaller than the full height might introduce visual artifacts if the viewer looks up or down beyond the view FOV. However, since there was virtually no motion near the poles of these videos, the smaller vertical FOV was not a problem. See Discussion Section for more details on choosing view discretization and FOV.

Each participant watched each of the five videos in one of two conditions: either a Gated version or a Standard (non-looping) version; participants only saw one version of each video. The ordering and condition were random. Each participant saw at least one video in each condition. They watched videos on an Oculus Rift VR headset, while we recorded their head orientation data.

In pilot experiments, in an attempt to single-blind the study, we did not explain the two conditions (Gated and Standard) to the participants. However, we found that because they did not understand the conditions and how they were different, they could not specify which version they preferred. Thus, in order to capture viewers' preferences between Standard and Gated clips, in our main study we informed participants as to which version of each video they were watching.

Before beginning the study, we explained to participants the two versions of videos they might watch; "Standard" version for normal video playback, and "View-Dependent" (Gated) version in which playback would wait for them if they were

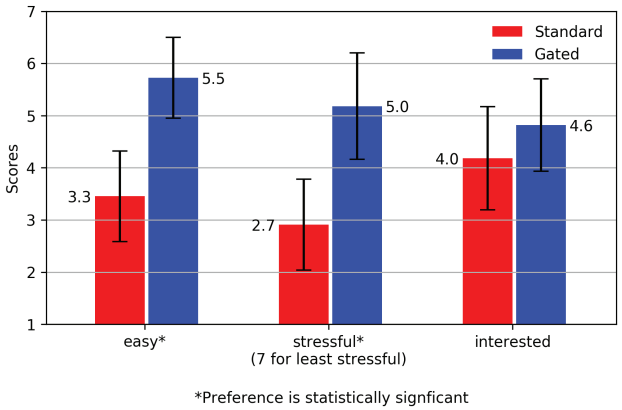

Figure 11. User study scores comparing Standard and Gated clips in three categories: (1) how easy it was to follow the story, (2) how stressful it was to follow the story (7 for least stressful), and (3) how interested they were in the stories. Confidence intervals are computed as 2×Standard Error. We found significant differences between the scores for "how easy" and "how stressfull", as indicated by the *'s, but not for "how interested".

looking in the wrong direction when an important story element occurs. Before showing each video, we only told participants whether the video was "Standard" or "View-Dependent."

After each video, participants removed the headset and took a break while filling out a questionnaire. We asked the participants to describe the story in their own words and to share feedback on how natural they thought the video playback was.

After all 5 videos were shown, we asked participants to complete a survey comparing the Standard and Gated versions on three 7-point Likert items: how easy it was to follow the stories, how stressful it was to follow the stories, and how interested they were in the stories. At the end, the survey included a binary-choice question asking which version they preferred overall and also included free-response questions asking what they liked and disliked about each version.

There were a total of 11 (5 female and 6 male) participants, with ages ranging from 24 to 36. All participants had some level of prior VR experience, such as watching VR videos or playing VR games. All except one participant watched all 5 videos; one participant preferred not to watch a VR horror video (Stranger Things) but watched the other 4 videos. We instructed participants to stop if they felt sick, but no participants reported sickness during the study.

Study Results

Overall Preference: Standard vs Gated Clips. Most participants (9 out of 11) preferred the Gated videos over the Standard videos. Only one participant preferred the Standard version, because they thought the Gated videos shown were slow (however, their complaints were largely about the pacing in the Gated version of the Lions video). The other participant was ambivalent: they preferred the Standard version if there were strong guidance cues on where to look, but if there were no strong cues, and if the event triggered immediately after they looked at the right thing, they preferred the Gated version.

How Easy, Stressful, Interesting? Participants scored the Standard and Gated versions in three 7-point Likert items: how easy it was to follow the stories, how stressful it was to follow the stories, and how interested they were in the stories (Figure 11). For each category, we performed the Wilcoxon signed-rank test on each pair of answers (Standard and Gated) from the same person. We applied continuity correction by adjusting the Wilcoxon rank statistic by 0.5 towards the mean value when computing the z-statistic [24]. For "how easy," there was significant preference for the Gated version ($p < 0.009$, $W = 1.5$, $r = 0.02$). For "how stressful," there was a significant preference for Gated version being *less* stressful ($p < 0.03$, $W = 5$, $r = 0.076$). We found no statistical significance in the "how interested" answers.

Standard vs Gated Clips: Likes and Dislikes. Most participants liked the Standard video because it did not hold up the story and had better flow, but they did not like the fact that they had to worry about missing important narrative elements. They liked the Gated video for being able to explore scenes at their own pace without worrying they might miss something. One stated reason for disliking the Gated version was having to look around and figure out what to look at to trigger the next event. Some participants disliked the fact that sometimes looking at the right thing did not immediately trigger the next story event. This happened occasionally when the clip they watched did not have seamless forward jump arcs to take them to right before the gate time; in such cases, they had to wait until the video played normally to the gate timecode.

How Natural was the Playback? Most participants interpreted the question "how natural did the playback seem to you" broadly, answering in terms of how natural the story content was, how natural it felt to have the interactive component in the story, or how natural the size of characters in the stories appeared to them (e.g., the rabbit from Invasion! was larger than real-life rabbits). Some participants were not accustomed to live-action videos waiting for them (e.g., the character breathing and waiting), and so thought the interactive aspect was unnatural. Only one participant noticed a ghosting artifact of two distant, moving pedestrians in the Murder Mystery video. In the original video, the two pedestrians walk steadily away from the camera the entire clip (which does not provide a view-dependent video texture an opportunity to loop), and in the postprocess stage, our tool could not find good backward arcs that end before the earliest cut t_C.

ROI Hit Rate for Standard Clips. In addition to the qualitative feedback, we also analyzed the head orientation data of all participants. In Gated versions of videos, participants had to see the ROI at the corresponding gate timecodes in order to proceed. We checked how often participants who watched the Standard version missed the ROI at the same gate timecodes. Overall, only an average of 61.9% of participants saw the ROI at the corresponding times ($\sigma = 31.4\%$).

Time Elapsed for Gated Clips. We also looked at how long it took participants to pass a gate, relative to the length of the gated clip without any looping. For Gated clips in which forward arcs were not enabled, we found that participants took on average 2.25 times the original clip length to pass the gate

($\sigma = 1.77$, average delay of 7.9s). For Gated clips in which forward arcs were enabled, participants took 1.79 times the clip length to pass the gate ($\sigma = 1.15$, average delay of 4.7s). However, this varied considerably; for example, in the second gate of Murder Mystery, viewers passed the gate faster with forward arcs than with the Standard version.

DISCUSSION

User Study Conclusions. In our user study, most viewers reported they preferred Gated videos over Standard ones. Overall, they find it easier and less stressful to follow stories in Gated than in Standard videos. However, because participants were aware of which videos were produced by our system, their feedback may be biased. The study indicated that a disadvantage for Gated videos was the need to figure out where to look in order to pass a gate. Thus, we suggest filmmakers use gaze guidance techniques in conjunction with our gating method, such as motion or lighting cues [18], in order to direct viewers' attention.

Design Choices for Gated Video. In order to produce video textures that create a good experience, we recommend directors take the looping structure into account when creating gated clips. In particular, directors should pay attention to structured (i.e., non-periodic, non-stochastic) motion within the scene. Views that do *not* satisfy the gate condition need to be looped, so the director should design the shot, e.g., shoot for a longer period, so that those views have some period of time with no structured motion. For example, video of a car moving across one view cannot be seamlessly looped, because no two frames have the car in the same position. However, if the director films for a longer period of time and captures additional footage of the car moving out of the view, or of the car coming to a stop within the view, then our method could find seamless loops using just the frames after the car leaves, or of the car at rest.

Our method may loop structured motion spanning multiple views. For example, consider a car moving from left to right across most of the scene. In a middle view between the starting and ending views of the car our system might generate a backward arc transitioning from a frame after the car leaves the view to a frame before the car enters the view. A viewer looking at this middle view might then see the car pass through the scene repeatedly. If the car is in the background, it may be fine for viewers to see the car loop in this manner. However, if the car is an important object that the director wants viewers to see, then seeing the car repeat its motion could be confusing. Thus, the director should choose the gate time and ROI carefully in such cases. For example, in the Patio video from our supplemental material, a character stands on the right side of the scene, walks to the left side, and stops. If the director places the ROI on the character after she comes to a stop, the graph-cut algorithm produces arcs that loop the walk, which may not be desirable. If the director instead places the gate ROI on the character *before* she starts walking, as shown in our supplemental material, the director can prevent viewers from seeing the walk loop repeatedly.

Number of View Discretizations & View FOV. The director should consider the trade-offs when choosing parameters for the number of view discretizations $|\mathbb{V}|$ and the FOV per view $v_{i=1\ldots|\mathbb{V}|}$. Recall that, during playback, our video player looks up the nearest discretized v_i and follows the arcs in v_i. Thus, increasing $|\mathbb{V}|$ increases the chance that, at run-time, the actual viewer's FOV will completely overlap with a view v_i, and thereby reduces the chance of seeing artifacts when following arcs in v_i. However, a larger number of views also increases computational cost.

Arc computations for each view only consider the pixels within the view FOV. Thus, a view FOV that is smaller than the FOV of the head-mounted display (HMD) may introduce artifacts during playback, since the HMD would show pixels that fall outside of the corresponding view FOV when playing loops. A view FOV that is larger than the HMD FOV makes arc computations more conservative and reduces the flexibility in finding seamless arcs, since it includes costs of pixels that fall outside the HMD FOV which viewers actually see. In our examples, we used $|\mathbb{V}| = 40$ and a view FOV of ($w = 80.65^\circ$, $h = 180^\circ$), which we found to be a good trade-off.

LIMITATIONS AND FUTURE WORK

Our algorithm does not account for audio when generating the view-dependent video textures. By default, our tool simply cross-dissolves the audio during transitions; we have found that this approach usually hides the seams in the loops well for ambient or environmental audio. For clips that have structured audio, directors may need to handle the audio tracks separately when generating gated clips. For example, music tracks could be looped independent of the video [39], whereas audio cues must be carefully synced. Future work could explore ways of looping audio in conjunction with the video.

While our algorithm can loop structured motion spanning multiple views, it may not be able to find seamless loops for *intra-view* structured motion, i.e., structured motion contained within one view. Future work could improve the applicability of our approach, by combining view-dependent arcs with motion segmentation and/or using frame synthesis for better looping video generation. For instance, if a view contains two people performing different repetitive actions, our method may not be able to find a seamless loop, but segmentation approaches [22, 40] could segment the two people and loop them separately. Frame synthesis could generate new frames to increase looping flexibility within a view.

Gated clips open up a considerable design space for the filmmaker to work within when creating their desired experience. For example, should the viewer be required to dwell on the ROI for the gate to be passed? Should it be sufficient that the viewer has seen the ROI at some time in the past? In theory, a gate could be used for every single important moment in the story, but such an arrangement might introduce awkward pauses and disrupt the pacing of the story, so there is also a space for determining where and how to place gates in a narrative. Future work could examine how different types of gate conditions and combinations thereof help achieve a variety of narrative goals.

ACKNOWLEDGMENTS

We thank Geoffrey Oxholm for his help; Baobab Studios, Netflix, Felix & Paul Studios, Fox, and IVAR studios for giving us permission to use their videos; Mitchell L. Gordon and Jane E for acting in our Murder Mystery video; Jingyi Li and Jane E for their help with assembling figures and videos; Xinwei Yao for helpful discussions; and all of our user study participants for their valuable feedback. This work was partially supported by the David and Helen Gurley Brown Institute for Media Innovation.

REFERENCES

[1] Advanced Microcomputer Systems. 1983. Dragon's Lair. Cinematronics [Arcade version]. (1983).

[2] Advanced Microcomputer Systems. 1984. Space Ace. Cinematronics [Arcade version]. (1984).

[3] Aseem Agarwala, Ke Colin Zheng, Chris Pal, Maneesh Agrawala, Michael Cohen, Brian Curless, David Salesin, and Richard Szeliski. 2005. Panoramic Video Textures. In *ACM SIGGRAPH 2005 Papers (SIGGRAPH '05)*. ACM, New York, NY, USA, 821–827. DOI: http://dx.doi.org/10.1145/1186822.1073268

[4] Jiamin Bai, Aseem Agarwala, Maneesh Agrawala, and Ravi Ramamoorthi. 2012. Selectively De-animating Video. *ACM Trans. Graph.* 31, 4, Article 66 (July 2012), 10 pages. DOI: http://dx.doi.org/10.1145/2185520.2185562

[5] Jiamin Bai, Aseem Agarwala, Maneesh Agrawala, and Ravi Ramamoorthi. 2013. Automatic Cinemagraph Portraits. (2013), 17–25. DOI: http://dx.doi.org/10.1111/cgf.12147

[6] Reynold Bailey, Ann McNamara, Nisha Sudarsanam, and Cindy Grimm. 2009. Subtle Gaze Direction. *ACM Trans. Graph.* 28, 4, Article 100 (Sept. 2009), 14 pages. DOI:http://dx.doi.org/10.1145/1559755.1559757

[7] Baobab Studios. 2016. Invasion! (2016). https://www.baobabstudios.com/invasion.

[8] Yuri Boykov and Vladimir Kolmogorov. 2004. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. *IEEE Trans. Pattern Anal. Mach. Intell.* 26, 9 (Sept. 2004), 1124–1137. DOI: http://dx.doi.org/10.1109/TPAMI.2004.60

[9] Gerd Bruder, Frank Steinicke, Phil Wieland, and Markus Lappe. 2012. Tuning Self-Motion Perception in Virtual Reality with Visual Illusions. *IEEE Transactions on Visualization and Computer Graphics* 18, 7 (July 2012), 1068–1078. DOI: http://dx.doi.org/10.1109/TVCG.2011.274

[10] William R. Bussone. 2005. *Linear and Angular Head Accelerations in Daily Life*. Master's thesis. Virginia Polytechnic Institute and State University, Blacksburg, Virginia. http://hdl.handle.net/10919/34615

[11] Shenchang Eric Chen. 1995. QuickTime VR: An Image-based Approach to Virtual Environment Navigation. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. ACM, New York, NY, USA, 29–38. DOI: http://dx.doi.org/10.1145/218380.218395

[12] Stephen Chenney and David Forsyth. 1997. View-dependent Culling of Dynamic Systems in Virtual Environments. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics (I3D '97)*. ACM, New York, NY, USA, 55–58. DOI: http://dx.doi.org/10.1145/253284.253307

[13] Franklin C. Crow. 1984. Summed-area Tables for Texture Mapping. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '84)*. ACM, New York, NY, USA, 207–212. DOI: http://dx.doi.org/10.1145/800031.808600

[14] Felix & Paul Studios. 2015. Wild: The Experience. (2015). https://www.felixandpaul.com/?projects/wild.

[15] Matthew Flagg, Atsushi Nakazawa, Qiushuang Zhang, Sing Bing Kang, Young Kee Ryu, Irfan Essa, and James M. Rehg. 2009. Human Video Textures. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games (I3D '09)*. ACM, New York, NY, USA, 199–206. DOI: http://dx.doi.org/10.1145/1507149.1507182

[16] Lester Randolph Ford Jr. and Delbert Ray Fulkerson. 1962. *Flows in networks*. RAND Corporation, Santa Monica, CA. https://books.google.com/books?id=fw7WCgAAQBAJ Report number R-375-PR.

[17] Google ATAP. 2019. Google Spotlight Stores. (2019). https://atap.google.com/spotlight-stories/.

[18] Steve Grogorick, Georgia Albuquerque, and Marcus A. Magnor. 2018. Comparing Unobtrusive Gaze Guiding Stimuli in Head-Mounted Displays. In *2018 IEEE International Conference on Image Processing, ICIP 2018, Athens, Greece, October 7-10, 2018*. IEEE, Athens, Greece, 2805–2809. DOI: http://dx.doi.org/10.1109/ICIP.2018.8451784

[19] Steve Grogorick, Michael Stengel, Elmar Eisemann, and Marcus Magnor. 2017. Subtle Gaze Guidance for Immersive Environments. In *Proceedings of the ACM Symposium on Applied Perception (SAP '17)*. ACM, New York, NY, USA, Article 4, 7 pages. DOI: http://dx.doi.org/10.1145/3119881.3119890

[20] Jan Gugenheimer, Dennis Wolf, Gabriel Haas, Sebastian Krebs, and Enrico Rukzio. 2016. A Demonstration of SwiVRChair: A Motorized Swivel Chair to Nudge Users' Orientation for 360 Degree Storytelling in Virtual Reality. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct (UbiComp '16)*. ACM, New York,

NY, USA, 281–284. DOI: http://dx.doi.org/10.1145/2968219.2971363

[21] Hajime Hata, Hideki Koike, and Yoichi Sato. 2016. Visual Guidance with Unnoticed Blur Effect. In *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '16)*. ACM, New York, NY, USA, 28–35. DOI: http://dx.doi.org/10.1145/2909132.2909254

[22] Mingming He, Jing Liao, Pedro V. Sander, and Hugues Hoppe. 2017. Gigapixel Panorama Video Loops. *ACM Trans. Graph.* 37, 1, Article 3 (Nov. 2017), 15 pages. DOI:http://dx.doi.org/10.1145/3144455

[23] Corneliu Ilisescu, Halil Aytac Kanaci, Matteo Romagnoli, Neill D. F. Campbell, and Gabriel J. Brostow. 2017. Responsive Action-based Video Synthesis. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 6569–6580. DOI: http://dx.doi.org/10.1145/3025453.3025880

[24] Eric Jones, Travis Oliphant, Pearu Peterson, and others. 2001–. SciPy: Open source scientific tools for Python. (2001–). http://www.scipy.org/

[25] Neel Joshi, Sisil Mehta, Steven Drucker, Eric Stollnitz, Hugues Hoppe, Matt Uyttendaele, and Michael Cohen. 2012. Cliplets: Juxtaposing Still and Dynamic Imagery. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. ACM, New York, NY, USA, 251–260. DOI: http://dx.doi.org/10.1145/2380116.2380149

[26] Philippe Levieux, James Tompkin, and Jan Kautz. 2012. Interactive Viewpoint Video Textures. In *Proceedings of the 9th European Conference on Visual Media Production (CVMP '12)*. ACM, New York, NY, USA, 11–17. DOI:http://dx.doi.org/10.1145/2414688.2414690

[27] Jing Liao, Mark Finch, and Hugues Hoppe. 2015. Fast Computation of Seamless Video Loops. *ACM Trans. Graph.* 34, 6, Article 197 (Oct. 2015), 10 pages. DOI: http://dx.doi.org/10.1145/2816795.2818061

[28] Zicheng Liao, Neel Joshi, and Hugues Hoppe. 2013. Automated Video Looping with Progressive Dynamism. *ACM Trans. Graph.* 32, 4, Article 77 (July 2013), 10 pages. DOI:http://dx.doi.org/10.1145/2461912.2461950

[29] Yen-Chen Lin, Yung-Ju Chang, Hou-Ning Hu, Hsien-Tzu Cheng, Chi-Wen Huang, and Min Sun. 2017a. Tell Me Where to Look: Investigating Ways for Assisting Focus in 360° Video. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 2535–2545. DOI: http://dx.doi.org/10.1145/3025453.3025757

[30] Yung-Ta Lin, Yi-Chi Liao, Shan-Yuan Teng, Yi-Ju Chung, Liwei Chan, and Bing-Yu Chen. 2017b. Outside-In: Visualizing Out-of-Sight Regions-of-Interest in a 360° Video Using Spatial Picture-in-Picture Previews. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17)*. ACM, New York, NY, USA, 255–265. DOI: http://dx.doi.org/10.1145/3126594.3126656

[31] Andrew Lippman. 1980. Movie-maps: An Application of the Optical Videodisc to Computer Graphics. *SIGGRAPH Comput. Graph.* 14, 3 (July 1980), 32–42. DOI:http://dx.doi.org/10.1145/965105.807465

[32] National Geographic. 2017. Lions 360°. (2017). https://www.youtube.com/watch?v=sPyAQQklc1s.

[33] Netflix. 2016. Stranger Things: Virtual Reality / 360 Experience. (2016). https://www.youtube.com/watch?v=yg29RvYNSDQ.

[34] Lasse T. Nielsen, Matias B. Møller, Sune D. Hartmeyer, Troels C. M. Ljung, Niels C. Nilsson, Rolf Nordahl, and Stefania Serafin. 2016. Missing the Point: An Exploration of How to Guide Users' Attention During Cinematic Virtual Reality. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology (VRST '16)*. ACM, New York, NY, USA, 229–232. DOI: http://dx.doi.org/10.1145/2993369.2993405

[35] Peter J. Passmore, Maxine Glancy, Adam Philpot, Amelia Roscoe, Andrew Wood, and Bob Fields. 2016. Effects of Viewing Condition on User Experience of Panoramic Video. In *Proceedings of the 26th International Conference on Artificial Reality and Telexistence and the 21st Eurographics Symposium on Virtual Environments (ICAT-EGVE '16)*. Eurographics Association, Goslar Germany, Germany, 9–16. DOI: http://dx.doi.org/10.2312/egve.20161428

[36] Amy Pavel, Björn Hartmann, and Maneesh Agrawala. 2017. Shot Orientation Controls for Interactive Cinematography with 360 Video. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17)*. ACM, New York, NY, USA, 289–297. DOI: http://dx.doi.org/10.1145/3126594.3126636

[37] Rocksteady Studios. 2016. Batman: Arkham VR. (2016). http://rocksteadyltd.com/#arkham-vr.

[38] Christoph Alexander Rosenberg. 2017. *Over There! Visual Guidance in 360-Degree Videos and Other Virtual Environments*. Master's thesis. Universität des Saarlandes. https://umtl.cs.uni-saarland.de/files/thesis_ma_christoph-rosenberg_compressed.pdf

[39] Steve Rubin and Maneesh Agrawala. 2014. Generating Emotionally Relevant Musical Scores for Audio Stories. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 439–448. DOI: http://dx.doi.org/10.1145/2642918.2647406

[40] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa. 2000. Video Textures. In *Proceedings of the*

27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 489–498. DOI: http://dx.doi.org/10.1145/344779.345012

[41] Robert Sedgewick. 2001. *Algorithms in C++ Part 5: Graph Algorithms (3rd Edition)*. Addison-Wesley Professional, Reading, Massachusetts. https://books.google.com/books?id=0rLN_tcvD-IC

[42] Alia Sheikh, Andy Brown, Zillah Watson, and Michael Evans. 2016. Directing attention in 360-degree video. In *IBC 2016 Conference*. IBC, Amsterdam, Netherlands, 29–37. DOI:http://dx.doi.org/10.1049/ibc.2016.0029

[43] Travis Stebbins and Eric D. Ragan. 2019. Redirecting View Rotation in Immersive Movies with Washout Filters. In *Proceedings of the IEEE Virtual Reality*. IEEE, Osaka, Japan, Article 1193, 11 pages. https://www.cise.ufl.edu/~eragan/papers/Stebbins-VR2019-redirected-movies.pdf

[44] Zhaolin Su and Shigeo Takahashi. 2010. Real-time Enhancement of Image and Video Saliency using Semantic Depth of Field. In *International Conference on Computer Vision Theory and Applications (VISAPP) (2)*. INSTICC, Angers, France, 370–375. http://web-ext.u-aizu.ac.jp/~shigeo/research/enhancement/index-e.html

[45] Marc Van den Broeck, Fahim Kawsar, and Johannes Schöning. 2017. It's All Around You: Exploring 360° Video Viewing Experiences on Mobile Devices. In *Proceedings of the 25th ACM International Conference on Multimedia (MM '17)*. ACM, New York, NY, USA, 762–768. DOI: http://dx.doi.org/10.1145/3123266.3123347

[46] Eduardo E. Veas, Erick Mendez, Steven K. Feiner, and Dieter Schmalstieg. 2011. Directing Attention and Influencing Memory with Visual Saliency Modulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 1471–1480. DOI: http://dx.doi.org/10.1145/1978942.1979158

[47] Walt Disney Animation Studios. 2018. Cycles. SIGGRAPH Immersive Pavilion. (2018). Dir. Jeff Gipson.

[48] Jingxin Zhang, Eike Langbehn, Dennis Krupke, Nicholas Katzakis, and Frank Steinicke. 2018. Detection Thresholds for Rotation and Translation Gains in 360° Video-Based Telepresence Systems. *IEEE Transactions on Visualization and Computer Graphics* 24, 4 (April 2018), 1671–1680. DOI: http://dx.doi.org/10.1109/TVCG.2018.2793679

APPENDIX

VIEW-DEPENDENT ARC COST COMPUTATION

We now provide more detail into how arc costs are computed. Recall that an arc is a transition between two frames (t,t'), and a view-dependent arc is a transition between two frames in a particular view v, represented as a triplet (v,t,t').

Our goal is to find arcs that transition seamlessly, so that viewers don't notice the transitions when they occur. In a pre-processing step, we assign costs to all possible view-dependent arcs (v,t,t') within a gated clip. The cost measures how seamless the arc transition is. As mentioned before, we only consider the field of view that the viewer sees when computing view-dependent arcs. We first discretize the view-sphere of all possible viewing directions into V views, and for each view, compute the costs of all possible arcs within that view. Therefore, the total number of arcs is $V * f^2$, where f is the number of frames in the gated clip.

View-Dependent Arc Cost Matrix

There are f^2 total arcs in each viewing direction, where f is the number of frames in the gated clip. We construct a cost matrix of size $f \times f$ that represents the arc cost between each pair of frames.

We define the cost of an arc between two frames for a given view as follows. Our goal is to penalize visually-noticeable changes when cross-dissolving between the two frames, such as a person appearing or disappearing, while ignoring minor changes due to pixel noise. We then apply a user-defined threshold to determine if an arc is noticeable or not.

The cost for an arc from time i to time j in view direction k is a summation over every pixel visible to the view, comparing the frames before and after the arc, summed over the duration N of the cross-dissolve:

$$C(v_k,t_i,t_j) = \sum_{x=0}^{N} \sum_{\ell \in \text{pixels}(k)} d(I_{\ell,i+x}, I_{\ell,j+x}) \max(e_{\ell,i+x}, e_{\ell,j+x}) \quad (1)$$

where $I_{\ell,i}$ is the RGB value of pixel ℓ at time i, and $e_{\ell,i} \in [0,1]$ is a binary edge map at pixel ℓ at time i. The edge maps are computed by Canny edge detection with a 3×3 Sobel filter, and min/max thresholds of 80 and 100 for the intensity gradient. The difference function ignores pixel differences below a threshold τ:

$$d(a,b) = \begin{cases} ||a-b||^2, & ||a-b||^2 \geq \tau \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where a and b are 3-dimensional vectors that represent RGB values in the range $[0..1]$. We use $\tau = 0.015$ to 0.2, depending on how much high-frequency, stochastic motion there is in the clip. The τ threshold prevents pixel changes due to stochastic motion, such as moving tree leaves, from overly penalizing the arc. Empirically, we found that a clip with no trees in the foreground works well with $\tau = 0.015$, whereas a clip with large foreground trees moving in the wind requires a larger $\tau = 0.2$.

For computational efficiency, we scaled our 360° videos in equirectangular format down to 640×320 before computing the cost matrices. We used a Summed Area Table [13] to accelerate computation, since the summation otherwise would

include considerable overlapping computations for overlapping views. The Summed Area Table computation for a 7s 30fps clip takes about 3.3 hours to complete on a 3.1 GHz Intel Core i7 processor, in single-threaded unoptimized Python.

BUFFER EDGE COSTS

We describe in more detail how buffer edge costs are assigned in our graph cut formulation. For views in the direction of the gate ($v \in \mathbb{H}$), we set the buffer edge weights to infinity for all timecodes $t \in 1 : T-1$.

For views $v \notin \mathbb{H}$, by default, we set the weight of the buffer edge from (v,t) to $(v,t)_B$ to the cost of the best backward arc (lowest-cost) from this node, from among all backward arcs that satisfy the user-specified minimum backward arc length. The user specifies a threshold γ for how large a cost is perceptually acceptable. If a node (v,t) has a backward arc with $C(v,t,t') < \gamma$, we call the node a *safe* node; otherwise, we call it an *unsafe* node. With our view discretization and FOV, it is possible that the viewer's FOV may partially extend outside the FOV used to compute an arc cost, so even though $C(v,t,t')$ may be less than γ, $C(v+\varepsilon,t,t')$ may not be and may have ghosting in the periphery. Hence, we prefer taking arcs from nodes that are neighbored by safe nodes, in order to decrease the likelihood of peripheral ghosting.

Thus, to compute the buffer edge weight $E_B(v,t)$, we find contiguous blocks of safe nodes in each viewing direction, i.e., each row of the graph (Figure 8), and add a small penalty to arcs of safe nodes that are within K frames near the ends of the blocks, or do not fully overlap with any blocks in adjacent views:

$$E_B(v,t) = \begin{cases} \infty, & \text{if } v \in \mathbb{H} \\ \omega_1(v,t) + \omega_2(v,t), & \text{if } v \notin \mathbb{H} \text{ and } (v,t) \text{ is safe} \\ \min\limits_{t' \le t-M} C(v,t,t'), & \text{if } v \notin \mathbb{H} \text{ and } (v,t) \text{ is unsafe} \end{cases} \tag{3}$$

where M is the minimum loop length in number of frames. ω_1 assigns a small penalty (between 0 and 1) based on how close the node is to the ends of the contiguous block, and ω_2 assigns a small penalty based on whether the block it is in has complete overlap with any block in neighboring views. Let $D(v,t)$ represent the contiguous block of safe nodes that node (v,t) is in.

$$\alpha = \min(t - D_{\text{start}}(v,t),\ D_{\text{end}}(v,t) - t) \tag{4}$$

$$\beta = \min\left(K, \frac{D_{length}(v,t)}{2}\right) \tag{5}$$

$$\omega_1(v,t) = \begin{cases} 1 - \dfrac{\alpha}{\beta}, & \alpha \le \beta \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

$$\omega_2(v,t) = \alpha \sum_{v' \in \mathbb{N}(v)} \max_{t'} \delta(D(v,t) \subseteq D(v',t')) \tag{7}$$

where δ is an indicator function that shows whether the frames of the first block is a subset of the frames in the second block. α is the amount of penalty for each neighbor in which the block $D(v,t)$ is not a subset of. We used $\alpha = 0.1$ and $K = 15$.

This heuristic reduces the chance of getting bad arcs in the post-process step, because it favors cutting on sequential arcs in the contiguous blocks, as opposed to arcs of non-contiguous, isolated safe nodes. For large contiguous blocks of safe nodes, it is likely that the frames have similar levels of motion (e.g., static), so safe nodes in large contiguous blocks are more likely to have a good backward arc that ends at or before the first safe node of that block.