# Effective Replays and Summarization of Virtual Experiences

Kevin Ponto, *Member, IEEE*, Joe Kohlmann, and Michael Gleicher, *Member, IEEE*

**Abstract**—Direct replays of the experience of a user in a virtual environment are difficult for others to watch due to unnatural camera motions. We present methods for replaying and summarizing these egocentric experiences that effectively communicate the users observations while reducing unwanted camera movements. Our approach summarizes the viewpoint path as a concise sequence of viewpoints that cover the same parts of the scene. The core of our approach is a novel content dependent metric that can be used to identify similarities between viewpoints. This enables viewpoints to be grouped by similar contextual view information and provides a means to generate novel viewpoints that can encapsulate a series of views. These resulting encapsulated viewpoints are used to synthesize new camera paths that convey the content of the original viewers experience. Projecting the initial movement of the user back on the scene can be used to convey the details of their observations, and the extracted viewpoints can serve as bookmarks for control or analysis. Finally we present performance analysis along with two forms of validation to test whether the extracted viewpoints are representative of the viewers original observations and to test for the overall effectiveness of the presented replay methods.

**Index Terms**—Virtual Reality, Viewpoint Similarity, Summarization, GPU, Bookmarking.

## 1 Introduction

Replays of users' experiences in virtual reality spaces (such as CAVEs or HMDs) and 3D desktop applications have the possibility to be extremely valuable for architects, designers and human factors researchers. Since users have full control of the viewpoint through head tracking or manual camera controls, the data needed to produce a replay of their activity with the correct virtual world perspective are readily available. Unfortunately, the naïve approach of simply replaying these movements is unlikely to be effective: head-tracked data is filled with movements that feel unnatural to the replay viewer, are often unpleasant to watch and difficult to interpret. For longer recorded experiences, direct replay is even less likely to be effective as high-speed replay makes the unnatural movements more difficult to watch. The inherent lack of structure provides no straight-forward means to summarize the experience, such as a synopsis of what the person was looking at.

For replays of these virtual experiences to be effective, they must achieve two goals simultaneously. First, they must be pleasing to watch, not only removing the high frequency jitter but also creating paths that are natural for a third-party viewer. Second, they must effectively convey the user's intent and experience, showing where the user was looking to convey the aspects of the environment the observer was most interested in, as well as conveying a sense of the space that was viewed. Balancing between these objectives is necessary because the goals are often conflicting: for example, a quick head turn is difficult for a third-party viewer to watch, but may convey a lot about the user's interest, whereas a wide-angle view may be effective at conveying a sense of the space, but not able to convey the user's focus.

We present methods for replaying and summarizing the experiences of a user in a virtual space that effectively communicate the user's observations while reducing unwanted camera movements. The core of our approach is a novel content-dependent metric that can be used to identify similarities between viewpoints. This enables viewpoints to be grouped by similar contextual view information, providing a means to determine what views were relevant to the observer. These resulting encapsulated viewpoints are used to synthesize new camera paths that convey the content of the original viewer's experience. Projecting the initial movement of the user back on the scene can be used to convey the details of the user's observations, and the extracted viewpoints can serve as bookmarks for control or analysis.

The major contribution of this paper is an approach to produce effective replays and summarization of virtual experiences. More specific contributions include:

- A content-dependent view similarity metric that affords efficient implementation on a GPU
- Methods to extract representative viewpoints from noisy head-tracked data and determine an optimal field of view
- Methods to synthesize camera paths from these extracted viewpoints

## 2 Overview

Effective replays of viewer experiences in virtual environments would be useful in a number of applications. An architect may view a replay of a client's walkthrough of a design to assess their interests or understand navigation difficulties. A Biochemist may replay their exploration of a molecule to revisit their discoveries for further analysis, and to remind themselves of the chain of exploration that led them there. A trainee may view a replay of a simulation with a coach to review his or her performance. A spectator may watch a replay of a "cyber-sports" event to appreciate the players' perspective. For these and most other applications of replay to be successful, the replay of the participant's experience must be *effective.* The replay must be comfortable enough for the viewer to watch, yet detailed enough to convey the experience of the participant.

Unfortunately, creating an effective replay from a user's experience is challenging. The viewpoint movements of first person experiences often do not make for good camera movements when viewed by a third party observer. One issue is that camera movements often contain quick motions and small jitters. Such movements are less of a problem for the user because they control their own head movements or direct a virtual camera system. A second issue is that the movements are not necessarily planned with an external viewer in mind and may seem confusing, unmotivated or unpleasant to watch. For these reasons, simple replay of a virtual experience is unlikely to be effective: they are likely to be difficult to watch and may not convey the experience effectively.

A related issue is that virtual experiences may be long and monotonous when viewed at their original speed. Replaying the view path at high speed exacerbates the issues in movement quality: the quick motions and jitter become even more objectionable, and important details may pass quickly. Therefore, effective replay mechanisms

- *Kevin Ponto is with the Department of Computer Sciences, University of Wisconsin, Madison, Email: kponto@cs.wisc.edu,*
- *Joe Kohlmann is with the Department of Computer Sciences, University of Wisconsin, Madison, Email: jkohlmann@wisc.edu*
- *Michael Gleicher is with the Department of Computer Sciences, University of Wisconsin, Madison, Email:gleicher@cs.wisc.edu.*

*Manuscript received 15 September 2011; accepted 3 January 2012; posted online 4 March 2012; mailed on 27 February 2012.*
*For information on obtaining reprints of this article, please send email to: tvcg@computer.org.*

Fig. 1. Viewpoints in camera paths. The green path, shown on the left, corresponds to the original motion of 1,400 viewpoints. The pink path in the middle demonstrates that the filtered path, while smooth, does not show all of the items seen in the original path. The extracted path, shown in blue on the right, comprises five viewpoints that demonstrate the views seen in the original path.

must not only smooth the view path before speeding it up, but must also identify important aspects of the path and ensure they are visible. One could use this information in other types of analysis, such as bookmarking interesting locations and creating visual summaries of the path as a sequence of images.

In this paper, we consider the problem of creating summaries and replays of virtual experiences. Effective replays must avoid jitter, minimize sudden movements and preferably consist of cinematic camera motions, all of which make them easier to follow. Replays must also effectively convey the user's interests and navigation strategies by showing the scene that was observed along the details of what was being viewed. To support the widest range of applications, replays should be generated automatically, based only on the available viewpoint path and scene geometry. Critical information about the user's interest and focus can be arduous or impossible to obtain, so the user's intent must be inferred from the data already available.

We tackle the problem of effective replay by first creating a summary of the user's view path as a concise sequence of viewpoints (Figure 1). Our approach selects a small subset of the original viewpoints that is sufficient enough to convey the user's experience. The method segments the viewpoint path into a set of time intervals, where the views within the interval can be summarized by a single viewpoint, showing scene content similar to the segmented views. These summarization viewpoints have a field of view that may be adjusted to widen or narrow the range of content seen in the segment. The core of this viewpoint selection process is a content-dependent similarity metric that measures the visibility coverage between different viewpoints.

In Section 3, we detail the viewpoint selection process by first describing the similarity metric (Section 4.1) and its efficient implementation on the GPU (Section 4.1.1). We then describe how the metric is used to segment the path to find the summary viewpoints (Section 4.3) and adjust their field of view (Section 4.4).

While the set of summary viewpoints provides a concise representation of a user's experience, it also serves as a basis for replays that more fully convey that experience. In Section 4, we describe a method for creating a camera path from the extracted viewpoint set that attempts to create camera movements that are visually pleasing while conveying the original observations. The method interpolates between the summary viewpoints and then adjusts this interpolated path to improve its smoothness and similarity to the original path.

To assess the effectiveness of our approach, we have conducted a series of experiments described in Section 6. The first assesses the performance tradeoffs in our approach, showing that the method is practical. The second experiment assesses the ability for the system to select views that represent the user's interest. The third experiment assesses viewer response to playback, providing an initial assessment of the method's success at communicating the user's experience.

## 3 Related Work

The methods described in this paper take motivation from a variety of fields. We will describe some of the existing work on metrics to determine viewpoint similarity, techniques to create paths from constraints, methods to play back observations in virtual reality environments and video stabilization methods.

### 3.1 Viewpoint Entropy / Similarity

Other researchers have looked at metrics to compare viewpoints in means of finding "good" viewpoints in virtual environments. Viewpoint entropy is a measurement based on Shannon entropy [27] that uses visibility information for the projected area of visible faces on a bounding sphere. Andujar et al. used viewpoint entropy information to determine waypoints in a virtual environment to create automatic walkthroughs of virtual environments [2]. Mühler et al. used similar methodologies for determining good viewpoints in volumetric data [22].

Others have analyzed methods to determine visibility information in viewpoints. Fleishman et al. mapped visibility weights on polygons to determine how well they could be seen from a given viewpoint [10] in means to generate automatic camera placements in virtual environments. Cyr and Kimia projected 3D objects onto a plane at regularly sampled intervals in means to determine the similarity of views for a mean of object recognition [8].

While these methods provide ways of finding "good" viewpoints in virtual scenes, they do not provide pertinent information on the visibility coverage of two given viewpoints, nor do they consider whether the chosen views convey a particular path.

### 3.2 Cinematography and Camera Control

Filmmakers and videographers have long considered the challenges of creating pleasing, yet communicative camera motions. In turn, the art of cinematography has developed guidelines and conventions. There have been efforts to codify this art in a computational framework for virtual camera planning (see Christie et al. [7] for a survey). This work generally does not consider balancing the quality of the resulting movement with similarity to an initial path in order to convey its intent. An exception is the Re-Cinematography work of Gleicher and Liu [11] that attempted to process video in a way that conveyed the intent of the original while following cinematic conventions. Our work builds on their cinematographic model, but applies it to 3D paths and considers issues of summarization and coverage.

### 3.3 Virtual Reality Playback

Various techniques have been used to convey virtual observations to 3rd party viewers. Recognizing the unnatural motion for virtual observations, the ShowMotion technique [6] presented a different model for generating smooth camera motions by predefining points of interest and then automating the transition between them for design review. StyleCam [5] by Burtnyk et al. created camera surfaces to focus viewer's attention on a point of interest in a virtual space. While these methods provide smooth transitions between views, the transition points are set manually in means of creating an "interactive TV commercial". Wernert and Hansen created a method for a virtual guide to keep users oriented in virtual space and to point out areas of interest

in means of supporting collaborative exploration of virtual environments [31]. Their approach does not address viewpoint control, and is based on explicit expression of interests by the users.

Additional research has provided alternative ways to represent position information from a virtual environment traversal. Zhiyong and Chunsheng used the MPEG-4 standard to store information about a user's virtual reality session [14]. Murgia et al. created a tool to replay the gaze of multiple users in virtual environments [23]. Sun et al. created methods to replay interactions in collaborative virtual environments [28]. These methods store information in unique ways, but they do not solve the problems associated with with replay presentation.

Other methods have approached this problem by using juxtaposed views. For instance, in the Boom Chameleon project, viewers were presented with a replay of the user's egocentric viewpoint alongside a pre-positioned stable viewpoint [30]. While this approach gives users additional contextual information, viewers are still subjected to the original movements of the user in order to understand the virtual user's actions.

### 3.4 Video Stabilization and Summarization

Video stabilization addresses a related problem to ours: removing the unwanted artifacts of hand-held cameras from video as a post-process. The key challenges are in understanding the camera motion from the video and adjusting the video frames in a plausible way. Our work does not face these challenges: we know the camera path and scene geometry and can easily re-render the scene from novel viewpoints.

While most video stabilization methods work in 2D, some recent methods reconstruct and process the 3D motion and geometry, then use image-based rendering techniques to recreate the new video. Such 3D video stabilization was introduced by Buehler et al. [3], and a more practical implementation was given by Liu et al. [21]. While, in principle, these approaches allow reasoning about the camera paths in 3D, they can only make limited changes to the given viewpoints and have a limited model of the scene geometry. Therefore, most stabilization approaches have been limited to performing filtering of camera paths[1], and have not considered the summarization or content dependent path generation required for effective virtual experience replays.

Video abstraction (or summarization) has a similar goal to ours: summarizing a long sequence of images with either a set of key images or a shortened video. A rich field has evolved with techniques for summarizing videos as a series of keyframes or as a shortened video (see Truong et al. [29] for a recent survey). While we take much inspiration from this body of work, the problem of working with video makes the solutions for video abstraction quite different than what is needed for the analysis of 3D viewpoint paths. For example, for 3D viewpoints there is a need to consider the geometry of the camera path and the ability to synthesize novel viewpoints.

## 4 Summarization

Our summarization method takes as input the view path of the user. This input is a sequence of viewpoints, where for each viewpoint (V), we obtain:

- The viewing transformation (encoded in a matrix $V_m$)
- The field of view (a scalar)
- A timestamp for when this viewpoint occurs ($V_t$)

We also assume that we have access to the scene geometry, so that we can determine the visibility coverage between viewpoints. We encode the visible geometry as a depth image from the viewpoint. This depth image ($V_d$) can be recomputed when necessary, but it is practical and efficient to precompute and store these images. As we will show in Section 7.1, a small image is sufficient.

Our method seeks to segment the view path into shorter, temporally contiguous segments such that for each segment, there is a single "summary" viewpoint that is similar to all of the views in the segment

[1] An exception is the previously mentioned Re-Cinematography work [11].

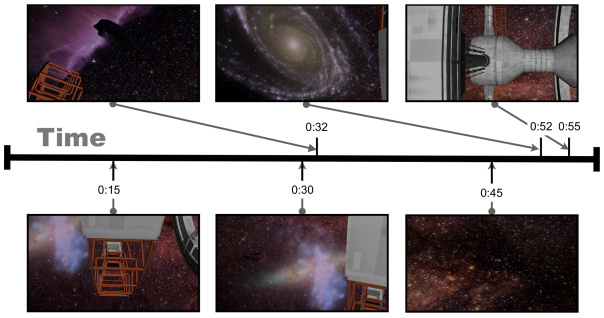



Fig. 2. Extracted views are shown at the top while the views resulting from regularly sampling (15 seconds) viewpoints are shown at the bottom for a one minute sequence of 3,583 viewpoints of an outer-space scene. The extracted viewpoints show the observer was interested in Horsehead Nebula, Messier 81 (M81 Galaxy) and then the space station model. The regularly sampled viewpoints not only fail to provide representative views, but make the view of the cage around the space station seem invalidly important.

(Figure 2). The kernel of this method is a similarity metric. Note that the metric is applied asymmetrically as it is used to compare a summary viewpoint to the members of the set it is summarizing.

### 4.1 Viewpoint Similarity

To determine how similar two viewpoints are to each other, we determine what amount of the view from one viewpoint is visible from the other viewpoint. Colloquially, this would be similar to setting one viewpoint as a flashlight and asking how much light can be seen from the second viewpoint. To determine this, we use a variant of shadow mapping, first described by Williams in 1978 [32].

Given two Viewpoints, A and B, we can create a vector for each point in space visible to Viewpoint A. Using the method of shadow mapping, we can then transform this vector from Viewpoint A's space into Viewpoint B's camera view, giving us a resulting vector R. By using the x and y coordinates of R as a lookup into the depth information for Viewpoint B, we can determine if the point in space visible to Viewpoint A is also visible to Viewpoint B (Equation 1).

$$L(A_c, B_c) = \begin{cases} 1, & |\, R.z - B_{R.xy}.z \,| <= \varepsilon \\ 0, & |\, R.z - B_{R.xy}.z \,| > \varepsilon \end{cases} \quad (1)$$

The total visibility of Viewpoint B in context of Viewpoint A, $V(A,B)$, is determined by the ratio of the visible pixels against to the total number of pixels in the view (N).

$$V(A,B) = \sum_{p=0}^{N} \frac{L(A_p, B_p)}{N} \quad (2)$$

As $V(A,B)$ is not equivalent to $V(B,A)$, as demonstrated in Figure 3 , the similarity value between Viewpoint A and B is determined as a weighted sum of the two visibilities:

$$S(A,B) = w_A V(A,B) + w_B V(B,A) \quad (3)$$

Sections 4.2 and 4.4 discuss considerations for setting the values of $w_A$ and $w_B$.

#### 4.1.1 GPU Implementation

Shadow mapping is highly parallelizable, as first shown by [26], so it is an incredibly effective method to implement on the GPU. Equation 1 uses the shadow mapping approach, so implementing it on graphics hardware is very straight forward. Unfortunately, producing screen space images of shadowed regions is not directly useful, as in order to evaluate Equation 2 we must actually count the number of pixels that are similarly visible. This can be done by rendering the Viewpoints to a buffer, reading it back into memory, and iteratively adding the resulting pixels values on the CPU. While this method is orders of

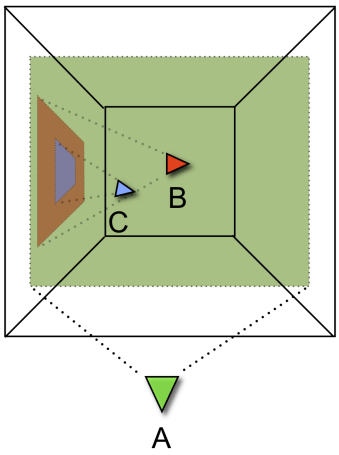



Fig. 3. Three viewpoints viewing in a hallway. Viewpoint A looks down the hallway while Viewpoints B and C look at the wall. The entirety of Viewpoint C is visible to Viewpoint A, but the majority of Viewpoint A is not visible to Viewpoint C.

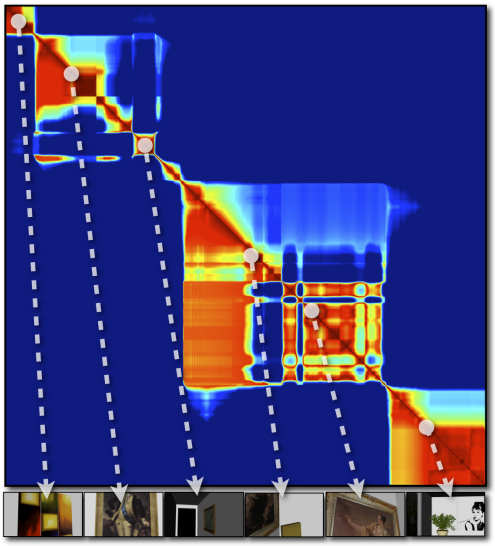


Fig. 4. Viewpoint similarity matrix for one minute of ego-centric data is shown at the top. The resulting extracted viewpoints are shown below.

magnitude faster than the same pixel accurate visibility checks on the CPU, it is unfortunately inefficient in the data transfer mechanisms. As opposed to sending an array of values from the CPU to the GPU, we would much rather send a single value representing $V(A,B)$.

Modern graphics cards have methods to query the number of pixels which pass through the pipeline [16]. These methods, named Occlusion Queries, have enabled a wide variety of general purpose computing applications to be performed on the GPU [25]. The summation in Equation 2 can be acquired by setting up a render query, rendering viewpoints A and B in a fragment shader that terminates any pixel in which $L(A_c,B_c)$ results in 0 and querying the number of samples that fully pass through the pipeline. Dividing this result by the number of pixels in the viewpoint buffer produces the value for $V(A,B)$. Beneficially, many occlusion queries can run in tandem, allowing tests of many viewpoints to be pipelined. By using occlusion queries, the bottleneck of data transfer is removed, providing an additional order of magnitude performance gain as shown in the results section below.

### 4.2 Viewpoint Selection Biasing

As opposed to only using the similarity metric for comparison purposes, we can also use it for *selection* purposes. This can be accomplished by changing the way the weights ($w_A$ and $w_B$) that are assigned in $S(A,B)$ (Equation 3).

Our goal is to select viewpoints that are tight and focused as opposed to being pulled back and broad. For instance, Figure 3 shows Viewpoints A, B and C, with Viewpoint A looking down a hallway and Viewpoints B and C looking at an object on a wall. While Viewpoints B and C are completely visible to Viewpoint A, Viewpoint A would not be a good representative view as Viewpoints B and C fill only part of Viewpoint A sees. In this case, Viewpoint C would more likely be a representative viewpoint as it is more likely to pinpoint regions of interest from the original observations. Therefore, we would like to bias our selection towards viewpoints whose views fill the frame. For this reason, we default $w_A$ to 0.8 and $w_B$ to 0.2 so that we prioritize close-up shots but preserve the user's center of focus. We can interactively change these values if necessary, but in practice these defaults were effective. We compensate for this near object biasing through our field of view optimization below.

### 4.3 Viewpoint Extraction

In order to find relevant viewpoints, we segment the original observations in search of viewpoints with long periods of high similarity. For each viewpoint, the preceding and succeeding viewpoints are evaluated (Equation 3) in order to find the duration that the summary viewpoint encompasses. This can be described as:

$$D(A,B) = \begin{cases} 1, & S(A,B) <= \tau \;\&\; D(A,B_{-1}) \\ 0, & otherwise \end{cases} \quad (4)$$

where $\tau$ is the minimum amount of similarity needed between two viewpoints to consider them to be similar. We defaulted to $\tau = 0.5$ to achieve a balanced number of representative viewpoints. As this may be an expensive process, we generate a similarity matrix (Figure 4) for each viewpoint in the original path, allowing $\tau$ to be modified without needing to recompute $S(A,B)$.

As finding the best subset of representative viewpoints is a difficult optimization problem, similar to the famous knapsack problem, we find an approximate subset through a greedy segmentation algorithm, similar to [11]. The viewpoint that encompasses the longest period of time is selected as the extracted viewpoint. The viewpoints that are temporally encompassed by this viewpoint are removed from further iterations of this greedy segmentation algorithm. The greedy selection process is then repeated until all of the viewpoints are accounted for, or until the duration of the remaining viewpoints is lower than the user defined minimum threshold. We chose one second as the default value to remove non-representative viewpoints symptomatic of inconsequential head movements. After extraction, we remove camera roll to produce views that are level in the horizontal axis, adhering to standard cinematographic practices.

### 4.4 Field of View

In order to preserve the original observations, each extracted viewpoint needs to determine the optimal field of view to ensure proper coverage of the viewpoints that it encapsulates. To accomplish this, we use the similarity metric $S(A,B)$ to iterate between the extracted viewpoint (Viewpoint A) and the original encompassed observation viewpoints (Viewpoints i through j) while varying the field of view. The field of view with the maximum summation of similarity is selected:

$$F(A,B_{i,j}) = \arg\max_{\theta}\{\sum_{B=i}^{j} S(A,B)\} \quad (5)$$

The key in this step is to use a new weighting scheme for $S(A,B)$. Figure 5 shows an example of a stabilized viewpoint and four encompassed viewpoints looking at a planar wall. In this case, setting $w_A$ = 1.0 and $w_B$ = 0.0 will result in the optimal narrow field of view ($\theta_A$). If the field of view was widened beyond this field of view ($\theta_A$), the view

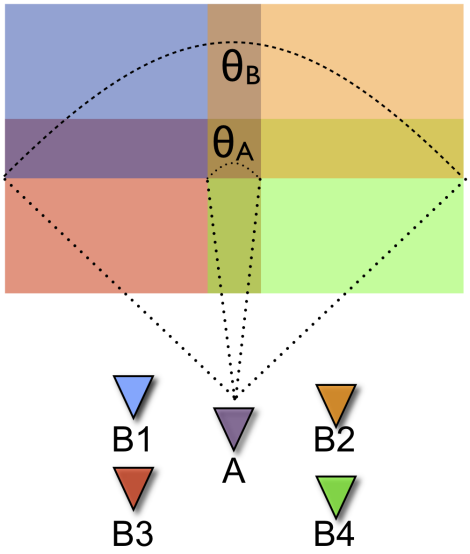



Fig. 5. Extracted Viewpoint A is compared against a series of Viewpoints (B1-4) which all view a planar surface. The optimal minimum field of view is represented by $\theta_A$, occurring where Viewpoint A's overlap is maximized. The optimal maximum field of view, $\theta_B$, occurs when the greatest region of viewpoints (B1-4) is encompassed by Viewpoint A.

for Viewpoint A would not be fully encompassed, thus decreasing the result of $F(A, B_{i,j})$.

By reversing the weights, $w_A = 0.0$ and $w_B = 1.0$, the function in Equation 5 will be maximized when the field of view expands to make the most of $B_{i,j}$ visible. This results in the optimal wide field of view as any larger field of view will not increase visibility of A in viewpoints $B_{i,j}$.

By choosing weights that create a wide field of view, we can compensate for choosing viewpoints that are close to objects. By default we choose weights that are the inverse of our selection weights (i.e. $w_A = 0.2$ and $w_B = 0.8$) in order to widen the field of view while preserving a representative coverage of the original encompassed viewpoints.

It is worth noting the opposite weighting scheme could have been chosen in the selection and field of view optimization algorithms. This would have resulted in "far object" biasing with a narrowed field of view. The "near object" biasing was selected as it provided better selections of relevant viewpoints and lessened the feeling that the camera was being towed behind the original observation path.

### 4.5 Summarization

As the goal of the previous steps has been to find relevant views, the extracted viewpoints provide a means of summarizing the user's experience. These user and content-dependent methods for extracting relevant viewpoints were generally able to reduce a five minute experience to a series of tens of viewpoints (See Figure 12 below). This not only provides a rather terse summarization of the user's experience, but also a means of path compression. Unlike camera filtering techniques, which simply lessen the magnitude of camera movements, our method is able to remove unnecessary camera views. This enables a pictorial summary of the user's experiences, which can be exported as a series of images (Figure 2). As shown in Section 7.2, the majority of these viewpoints are representative of the user's interest. As the replay can arbitrarily move between these viewpoints, this method also provides a way of generating automatic bookmarks.

Unfortunately, as the field of view has been widened to accommodate the encapsulated viewpoints, these viewpoints may lack precision as to what precisely the observer was focused on. We mitigate this problem through the use of projection as described below.

### 4.6 Projection

While these extracted viewpoints represent a relevant region of the virtual scene, it may be necessary to give viewers a finer level of detail as to what the user was observing. While researching pointing gestures, Krahmer and van der Sluis found that the metaphor of a headlamp drastically improved the understanding of what a user was trying to indicate, as opposed to looking at the pointing finger as a single ray [19]. Lablack et al. used a similar projection method on top of 2D images in order to determine more information about the user's gaze [20].

As the goal of our replay method is to provide information about the user's experience, we have developed a headlamp analogy. From the synthesized camera viewpoint, the original head tracked viewpoint is projected as a light onto the scene. This method allows observers to easily understand what the viewer was originally looking at, while not being subjugated to the viewer's movements. Furthermore, the high frequency movements of a flashlight are familiar occurrences, while violent egocentric camera motion is not. As shown in the results section below, this headlamp analogy was able to greatly aid in viewers' understanding of virtual observations.

## 5 Path Synthesis

Our approach uses the extracted summary viewpoints to create a replay path. We seek to create camera paths that effectively convey the experience and are easy to watch. Our approach creates smooth camera paths by interpolating key viewpoints. The extracted summary viewpoints serve as an initial set to interpolate; however, we add additional points to improve both the smoothness of the video and its effectiveness at conveying the user's experience.

Our approach interpolates the view transforms directly rather than factoring the transformations and interpolating the various components independently. Methods for interpolating transforms directly, known as exponential maps, were introduced to the graphics community by Alexa [1] and rely on performing the linear operations (e.g. interpolation) in the logarithm space of the matrices. The concept is to apply the logarithm of the transformation before applying linear operations. Equation 7 demonstrates the interpolation matrix $I_m$ between Viewpoints A and B with $t$ being a normalized time between the timestamps $A_t$ and $B_t$ (Equation 6). Although this method is only an approximation [12], this technique provides a proven means to move between camera poses.

$$t = \frac{Time - A_t}{B_t - A_t} \tag{6}$$

$$I_m(A, B, t) = e^{(1-t)logA_m + tlogB_m} \tag{7}$$

We use exponential maps as opposed to parameter interpolation as, in common cases, they do a better job of keeping points of interest centered in view, as shown by Hawkins and Grimm [13]. This, in turn, creates arcing paths that produce less artificial movements. While these arcing paths have the possibility of drifting from the original observations, we describe a method for detecting this and correcting for it in Section 5.2 below.

While exponential maps do not produce an interpolation with a constant velocity, the non-constancy is small and bounded [15]. As abrupt accelerations and decelerations are jarring for viewers, we chose to implement an "ease-in/ease-out" function [11]. This method creates an initial acceleration vector and then interpolates between matrices at a constant speed before decelerating to a stop.

### 5.1 Corner Smoothing

The method described above interpolates between viewpoints temporally, but it may also be advantageous to smooth the simplified path spatially. This may occur when the extracted viewpoints create "square" motion paths that feel very robotic and unnatural.

For example, Figure 6 shows three Viewpoints (A, B, and C) that create a path with sharp right angle. As some users may find the movement displeasing, we create an optional smoothing method. The

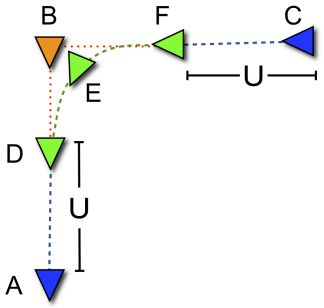



Fig. 6. Viewpoints D, E and F are used to smooth the path between Viewpoints A, B and C.

value U is used to determine how far along the path between Viewpoints A and B and Viewpoints B and C the smoothing should occur. Viewpoints D and F are created such that $D_m$ is equivalent to $I_m(A,B,1.0-U)$ and $F_m$ is equivalent to $I_m(B,C,U)$. Viewpoint B is then replaced with Viewpoint E using a low-pass filter between Viewpoints D, B and F as shown in Equation 8.

$$E_m = e^{(\frac{logD_m}{4} + \frac{logB_m}{2} + \frac{logF_m}{4})} \quad (8)$$

We use this smoothing approach as to opposed to representing the path as splines (e.g. a series of Bézier curves) as it maintains the viewpoint semantic used in the summarization method described above.

### 5.2 Auto-Correction for Viewpoint Discrepancies

While the interpolated camera paths are smooth, they may deviate substantially from the original path. To balance smoothness and faithfulness to the original path, we insert additional viewpoints into the path to correct large discrepancies. As the presented method aims to simplify the original camera path, it is possible for the synthesized path to vary substantially. This can occur when the user spins 180 degrees—the user may have originally rotated to the left, but the synthesized path might show rotation to the right. Synthesized paths may also fly through objects and walls if the movement between extracted viewpoints is not considered to be relevant by our segmentation algorithm.

Fortunately, this situation can be both detected and corrected. During playback, the visibility ($V(A,B)$) is computed between the synthesized path (A), and the original path (B). Periods of time when the synthesized path cannot see the original observations are detected when $V(A,B) < 0$. For each of these periods of time, a new viewpoint is added to the synthesized path that had the highest relevance value from the method described in Section 4.3. This added viewpoint may not completely fix all of the visibility discrepancies, however. That is, it is only guaranteed to resolve visibility discrepancies for this precise time. Therefore, this process of inserting keyframes may need to be repeated iteratively.

Figure 7 provides an example of this situation. The user's movement between two rooms was not deemed relevant for viewpoint extraction, so the synthesized path only contains viewpoints within the two rooms, with no transition viewpoints in between. This results in the initial synthesized path moving in an smooth arc between the viewpoints and subsequently moving through a wall. On the first iteration of our correction algorithm, the added viewpoint results in a camera path that moves through the adjacent wall. In the second iteration of the correction algorithm, the method produces a camera path that does not pass through any walls. Furthermore, the resulting camera path is very similar to the original camera path, but only requires one-twentieth of the number of viewpoints.

Note that if an interpolated path passes through an object, the view is likely to change radically and be very dissimilar to the original path. This situation would invoke a correction (unless the original path has a similar intersection). In this way, the path correction process often avoids collisions between the virtual camera and objects without explicit collision detection.

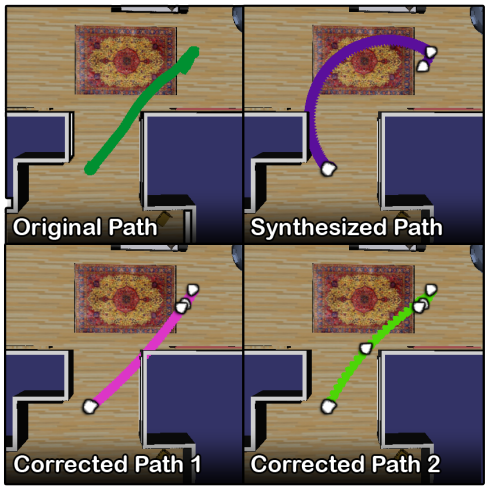



Fig. 7. Four different paths comparing the original path, the initial synthesized path and two versions of the corrected path with extracted and added viewpoints shown in white. In the synthesized path (upper right), the camera path goes through a wall. In the first correction (lower left), the additional viewpoints cause the path to clipped by a different wall. In the second correction (lower right), the resulting path now closely mirrors the original path (upper left).

## 6 Implementation

We have implemented our approach as a stand-alone replay system built for Windows PCs. Our application utilizes OpenSceneGraph [4] to read scene files from a variety of sources, notably those used in our Virtual Reality environment. We have added a small script to our standard VR software environment to capture streams of head motions as encoded, time-stamped view transformations. We have also built a desktop application to play back these head motions within the scene.

The replay application reads view paths and their associated scenes, enabling the viewer to create customized replays using either our proposed method or a traditional path smoothing operation, such as a low-pass filter. All the parameters and weights can be easily configured through a graphical user interface. Viewers can inspect the environment and any camera path via game controllers or standard mouse and keyboard interaction.

After viewers have configured all of the properties to their liking, they can either generate a draft-quality image sequence of the path, based on realtime OpenGL renderings, or export the path for use in with higher-quality 3D renderer. When working in such a rendering environment, such as Blender or POVRay, the previously discussed headlamp effect 4.6 can be achieved by also importing the original

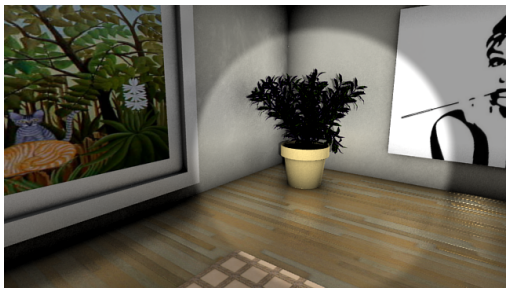


Fig. 8. The results of a high quality rendering. The illuminated section represents the original view, in this case, focusing on the plant.

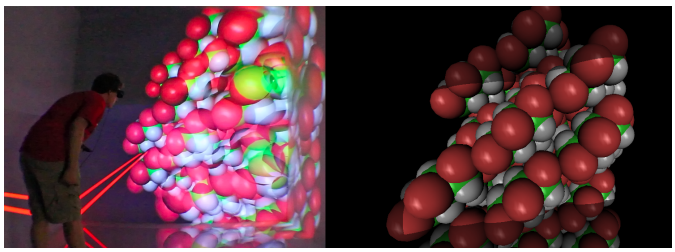


Fig. 9. User analyzing a model of crystalline cellulose (left). A screenshot taken from the replay of this interaction (right) shows our method's ability to better communicate the overall experience.

observation path as a light object. This enables the viewer to generate very high quality videos and renderings, as shown in Figure 8.

## 7 Results

We have used our replay system with several familiar scenes, from confined spaces, such bathrooms and bedrooms, to larger environments, such as parks and futuristic models. For these larger scenes, users tended to stand in place while virtually traversing the scene though the joystick on a controller. While our method produced effective replays of these interactions, it significantly enhanced replays of interactions in which the user physically walked around the scene. For example, when we had the user walk around a model of crystalline cellulose (Figure 9), we observed that the original replay of the user's experience lacked essential contextual cues to help viewers understand motion and positioning. The arced paths produced by our method resulted an initial replay that smoothly rotated the molecule from one side to another, just as the user had walked around the molecule. When combined with a "headlamp" projection of the user's original observations, the final replay offered an effective communication of the overall experience.

Running on our test system, a 2.8 GHz Intel Core i7 workstation with 3 GB of RAM and an Nvidia GeForce GT 240 graphics card, our application is able to process a five minute replay in under ten minutes. While content dependent, the method generally produced three to six keyframes for each minute of observation data using the default parameters.

In order to evaluate our approach, we conducted a series of three experiments. For these experiments, we crafted a virtual museum environment with a number of well-defined objects of interest that required users to non-linearly navigate between the various objects and rooms.

The three experiments each tested a different aspect of our system. The first explored the performance tradeoffs in our Viewpoint Similarity metric. The second experiment assessed the ability of the summarization process to effectively select viewpoints that represent the user's interest. The third experiment assessed the resulting replay videos, considering both viewers' subjective responses and their ability to interpret the user's experience.

### 7.1 Performance Testing

Even though our method is an offline process, fast processing enables viewers to tweak parameters to their liking. The bottleneck for our method is the similarity matrix generation in the viewpoint extraction step (Section 4.3). As constructing the similarity matrix requires $O(n^2)$ viewpoint comparisons, it was important for this process to be as fast as possible. An advanced implementation may not need to build the entire similarity matrix. For example, if a bound of $k$ frames is placed on the segment length, only a banded matrix of bandwidth $k$ needs to be constructed. Other acceleration strategies, such as sparse sampling or lazy construction of the similarity are possible, but challenging as they may actually hurt performance by destroying the coherence of the process.

In order to determine how much the optimizations listed in Section 4.1 affected the performance of the system, we tested our method on a one minute walk-through of the aforementioned museum environment, consisting of approximately 1,000 viewpoints with a screen resolution of $960 \times 540$. For comparison, we tested the same operations using standard CPU operations as well as GPU processing with texture readback for the generation of the entire $1,000 \times 1,000$ similarity matrix on our test system. As shown below, by moving these comparisons from the CPU to the GPU and subsequently using occlusion queries, we were able improve the time to completion by many orders of magnitude (Figure 10).

By storing depth information in subsampled buffers, this process can be further accelerated. While this may change the results of the segmentation algorithm, in practice, the buffer size can generally be reduced without an overly noticeable effect (Figure 4). In this case, using a depth texture that was $120 \times 68$ resulted in the exact same extracted viewpoints as the baseline path, while only taking one-thirtieth of the time. However, reducing the depth textures below $64 \times 64$ did not reduce processing times and created artifacts in the similarity matrix. For the example shown in Figure 11, the $60 \times 38$ sized buffer resulted in two viewpoints having slight discrepancies in times from the baseline results (the time segments that the viewpoints encompassed were off by $\pm$ 0.1 seconds).

Computing the segmentation from this matrix and the optimal field of view varies by number of extracted viewpoints, the duration which these extracted viewpoints encompass and the number of variations of the field of view parameter that are sampled. For the example above, determining the extracted viewpoints and the optimal field of view took four seconds, with ten samplings of the field of view for each extracted viewpoint. In practice, the extraction and refinement steps generally take five to ten seconds to compute. Note that these refinement steps do not need the similarity matrix to be recomputed, meaning that while the initial synthesized path may take 30 seconds to be generated, further synthesized paths can generally be created in under ten seconds. In comparison, the video stabilization techniques in Final Cut Pro X take approximately five minutes to run on the same observation data.

### 7.2 Summarization Testing

In order to assess the effectiveness of our summarization, we created an experiment to see if objects that users found interesting would appear in the views for our extracted viewpoints. To accomplish this, we asked four users to traverse the virtual museum environment inside of a C6 CAVE system while verbally annotating the items that they were visually inspecting. These verbal annotations were then reviewed and processed to create an ordered list of the items viewed.

Using our method, we extracted the representative viewpoints from these virtual traversals. Each user identified approximately 20 objects, spent from four to eight minutes inside the environment, and generated from 3,000 to 6,000 viewpoints[2] (Figure 12).

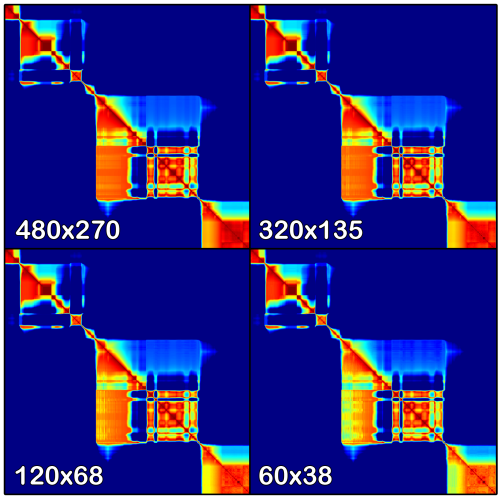
As shown, the number of extracted viewpoints was roughly proportional to the number of objects identified. As our method has no notion of what represents an "object", and because head tracked data does not have the precision of eye tracked data, a viewpoint may encompass several objects. Additionally, some viewpoints may be transition viewpoints, representing the travel path in between objects. Finally, an object may encompass multiple viewpoints if the observer inspected the object from different angles.

The majority of the extracted viewpoints corresponded to a one-to-one mapping of view to object. This is ideal as it provides a clear pictorial summarization of the user's experience. In cases where more

[2]The fact that the viewpoints are not exactly proportional to time spent in the environment is due to slight inconsistencies in the rendering frame rate.

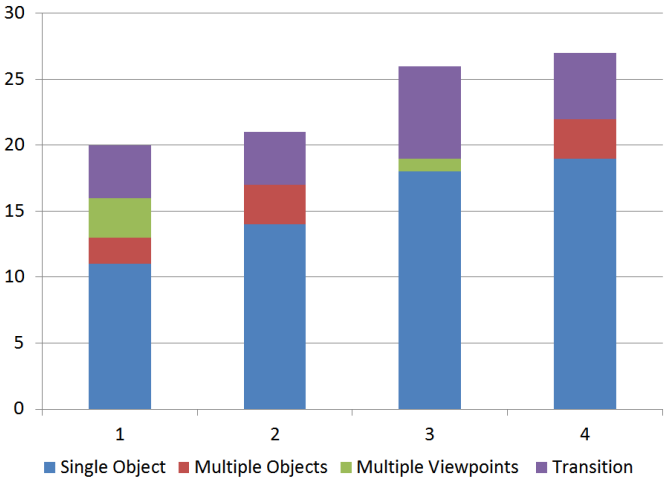
| Approach | CPU | GPU with readback | GPU with occlusion queries |
|---|---|---|---|
| Time | 1.7 days | 1.5 hours | 8 minutes |

Fig. 10. Utilizing the GPU substantially reduced the processing time needed to create a $1,000 \times 1,000$ similarity matrix.




| Depth Buffer Size | 480 × 270 | 320 × 135 | 120 × 68 | 60 × 38 |
|---|---|---|---|---|
| Time | 152 (s) | 36 (s) | 18 (s) | 18 (s) |
| Correctness | 100% | 100% | 100% | 75% |

Fig. 11. The similarity matrices for different sizes of depth buffer are shown above. The table below shows the time taken to compute these matrices and the percentage of the extracted viewpoints that were an exact match of the baseline path.




| Participant | Time | Viewpoints | Objects Identified | Extracted Viewpoints |
|---|---|---|---|---|
| 1 | 3:50 | 2,877 | 18 | 20 |
| 2 | 5:45 | 3,927 | 20 | 21 |
| 3 | 7:03 | 6,110 | 18 | 26 |
| 4 | 7:43 | 6,102 | 19 | 27 |

Fig. 12. The results of four different viewers traversing a virtual museum. The table shows the duration of the event, the number of viewpoints observed, the number of objects identified and the number of viewpoints extracted. The graph shows the breakdown of what these extracted viewpoints represent.

than one object filled a viewpoint, the projection method from Section 4.6 allowed viewers to distinguish what objects were being focused on during playback. The instances when multiple viewpoints were extracted, presenting a single object, often provided different perspectives on the object, and thus were still useful for the purposes of replay and summarization. While viewpoints did not represent an object were not useful for bookmarking, they reduced the need for path autocorrection 5.2. In each walk-through, all of the objects mentioned by the participant appeared in one or more viewpoints.

### 7.3 Replay Effectiveness Testing

In order to assess the quality of the replays generated with our approach, we conducted an experiment that measured viewers' responses to replay videos. The experiment was designed to compare five different approaches to replay generation as its conditions:

- **Original:** Direct use of the user's original observation path
- **Final Cut Pro:** Application of a commercial video stabilization tool to the results of the direct video
- **Filtered:** Low-Pass filtering of the viewpoint path
- **No Projection:** Application of our replay generation approach to create the path
- **Projection:** Application of our technique, including the headlamp projection (Section 4.6)

For stimulus, we generated five different paths through the museum environment. Each path contained one minute of observer data, viewing five to seven objects. The videos presented to the participants were played back at 4× their original speed.

Our experiment was a full within-subjects design. Each study participant experienced all conditions. Each condition was shown using a different path for each participant. The combination of conditions and paths were stratified; that is, every combination of path and condition was seen by an equal number of participants. The presentation order was randomized.

Our experiment showed each participant five videos. After each video, the participant was asked a series of questions. First, they wrote which object they felt was most important to the user as a means to ensure the participants were actively engaged. Second, participants rated their opinion of how important objects were to the user on a four point scale, from not important to very important. Next, participants rated their subjective response to the video playback in terms of how easy the video was to watch, how disorienting the video was and how confident the participant was that they understood what the user was looking at, with answers on a seven point scale. The participant then rated the camera movement in the video on a seven point scale, from chaotic to calm and from jerky to smooth. Next participants answered questions on the spatial relationships of objects, relating an object to a specified letter on a floor layout. They were also asked to determine the relative positions of objects, for example, selecting which object was furthest to the left of a given viewpoint. Finally, participants were asked which of two objects were shown for a longer duration in the video. After viewing all five videos, participants were asked which video they enjoyed the most and why.

The experiment was run through the Amazon Mechanical Turk service. We followed the practices described by Kittur et al. [17] and Downs et al. [9] to avoid issues in crowdsourcing participation in our study. Specifically, we chose a standard rate of pay and created questions to verify that the user was human, had the technical ability to participate and was actively engaged. Participants were compensated based on an estimate that the experiment would require approximately 20 minutes of their time.

The study attracted 20 participants and took an average of 16.5 minutes to complete. The results of the study were run through multivariate analysis of variance (MANOVA) using the Pillai test [24] to ensure that the data were statistically significant.

#### 7.3.1 Results of Qualitative Questions

The results for each of the qualitative questions on camera movement and ease of understanding produced highly significant results ($p \leq .01$). When asked how easy the video was to watch, the No Projection version of our technique was considered best ($F(4,95) = 4.92$, $p = 0.001$) as shown in Figure 13. Both the Filtered version and the Projection version were considered easy to watch. The Original version was considered more difficult to watch, while the Final Cut Pro version was considered very difficult to watch. These results confirm that

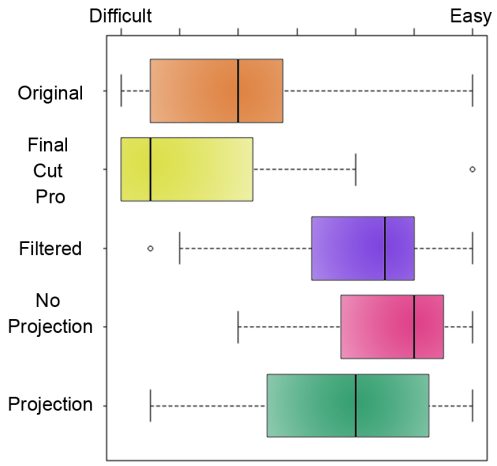



Fig. 13. Participants' ratings for videos in terms of how easy they were to watch. The Filtered, Projection and No Projection videos were generally considered easy to watch.

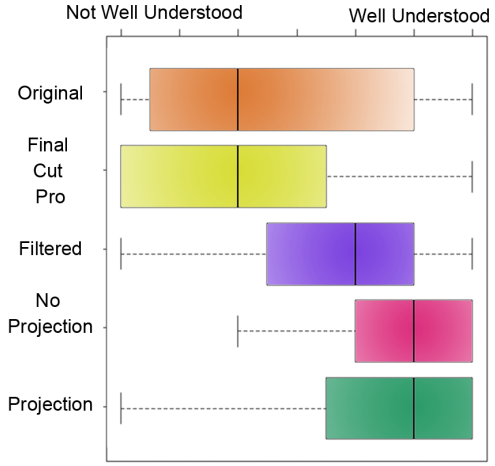



Fig. 14. Participants' ratings for videos in terms of their ability to provide an understanding of the user's experience. The Projection and No Projection versions of our method generally outperformed the traditional techniques.

our approach to minimize camera movement does lead to videos that are easier to watch.

When participants rated the videos in terms of their ability to provide an understanding of the user's experience, our methods also did extremely well (Figure 14). Participants felt that they had a firm understanding of the user's experience after viewing the Projection and No Projection versions of our techniques ($F(4,95) = 3.89$, $p = 0.005$). They also felt they had a moderate understanding of the experience when viewing the Filtered version, but felt like they had little understanding of the experience when viewing the Final Cut Pro version or the Original version.

The other three qualitative questions produced similar results, with the Projection version, the No Projection version and the Filtered version receiving positive results (calm, smooth, and not disorienting). Meanwhile, the Final Cut Pro version and Original version generally received negative results (chaotic, jerky and disorienting).

### 7.3.2 Results of Quantitative Questions

With difficult paths, we were able to show that the Projection version of our method was able to convey the importance of objects to participants better than other methods ($F(4,15) = 4.62$, $p = 0.03$). On average, participants provided answers that correctly matched what the user was viewing with 83% accuracy when viewing the Projection version of a path. The next best method for answering these questions came from the Original version, with 76% accuracy. The other three methods, the No Projection versions, the Filtered versions and Final Cut Pro versions all fared pretty evenly, with participants answering questions with 68% accuracy. When the paths were not difficult, accuracy across the different versions was less distinguished. In terms of spatial comprehension, the study did not produce statistically significant results ($p > .05$).

For the final reflection question, seven of the twenty participants wrote that they enjoyed the No Projection version the most. These responses stated that the video "[had the] smoothest motion and was the easiest to follow", "was calm" and "was the easiest to watch of them all". Four of the participants selected the Projection method, stating, "I liked how it was more calm and smooth, and it was easier to pinpoint what the person was looking at", "it really made me focus on the cats in the picture" and "[the video was] easier to watch and understand". Three of the participants selected the Filtered version, stating "It was the slowest and easiest to follow" and was "the most pleasant". Two of the participants selected the Original version, stating that the camera movements seemed "purposeful". Interestingly, two participants selected the Final Cut Pro version, stating that while the camera movements were "chaotic", the video made the questions challenging in an enjoyable way. The remaining participants stated that they had no preferences between the videos they had seen.

## 8 Discussion

These results of our effectiveness testing provide verification that our method was able to effectively communicate a user's experience. In terms of understanding what was important to the user, the Projection versions gave the participants the greatest amount of information. While the Filtered versions produced results that were easy to watch, they lacked the pinpoint view information needed to answer these questions.

Although the spatial comprehension study did not produce statistically significant results ($p > .05$), the Projection version was able to produce the highest scores in four of five trials. As there is great variability in individuals' ability to spatially understand a novel environment [18], we would like to further investigate how camera movements and parameters affect participants' spatial understanding of virtual environments.

As the major focus of our method is replaying virtual experiences from walk-throughs, our methods are designed for static virtual environments. The Viewpoint Similarity metric, described in Section 4.1, uses transformations and depth information to determine if pixels visible in Viewpoint A are visible in Viewpoint B. For these methods to work correctly, all world transformations that change the results of the depth image must be encapsulated in $V_m$. Whether the user moves the world, or the world moves around the user, the similarity metric will work equally well.

Dynamic objects inside of the virtual environment present additional ways to modify the transformations embedded in the viewpoint. This can result in the visibility test producing incorrect results. This effect can be resolved by storing additional viewpoint information for each of the dynamic objects in the scene. In this way, the scene can be thought of as a layering of motion objects.

Also, while the Viewpoint Similarity metric is not designed to work for dynamic objects, the metric will often produce acceptable results for Viewpoint Extraction.[3] If an object has moved, it is likely that the Viewpoint Similarity metric will overestimate areas of dissimilarity ($L(A_c, B_c)$ and $L(B_c, A_c)$ both equal zero). After a moving object comes to a stop, the subsequent Viewpoint Similarity calculations will produce correct results.

In the future, it would be incredibly beneficial for viewers to see what a user is experiencing in realtime. The fact that the Viewpoint

[3] It is important to note that these kinds of dynamic behaviors need to also be recorded so that they can replayed correctly.

Similarity metric is highly optimized should allow for realtime approximations of the segmentation algorithm. Additional information could also be used to aid in the viewpoint selection process. For instance, we currently do not use color information in the viewpoint analysis. By using saliency models, we may be able to further refine the selection of representative viewpoints. Also, while we tested our methods with and without projection, it may be useful to test other approaches to convey the original observations. Future work will explore alternative visualization techniques.

## 9 Conclusion

We define effective replays as having two simultaneous goals, to be watchable and to be able to convey the user's experience. While existing methods of camera path filtering and video stabilization fall short of these goals, our presented method is able to provide viewers an experience that is easy to watch while simultaneously being informative. The key to our approach is a novel content dependent metric that can be used to identify similarities between viewpoints, enabling viewpoints to be grouped by similar contextual view information. Furthermore, in practice, the presented method is able to create an effective summary of a user's experience, reducing minutes and thousands of initial observer viewpoints to tens of representative viewpoints.

## 10 Acknowledgments

We would like to acknowledge the support of the the Living Environments Laboratory, the Wisconsin Institute of Discovery, and the CIBM training program. This project was supported in part by NLM award 5T15LM007359 and NSF awards IIS-0946598 and CMMI-0941013. We would specifically like to thank Aaron Bryden, Patricia Brennan, Hyun Joon Shin, Kendra Jacobsen, Andrew Cherry, Michelle Craft, and Robert Radwin for their support and assistance. We would finally like to thank Claire Pointer, Ericka Kreutz, Lisa Adams, Nolan Haims, James Compton, Bob Loehr, Julie Loehr, Jane Elias, Dana Snyder, Aasif Mandvi, Jim Wisniewski, Eileen Stevens and Tomas B. Christiansen for their assistance in path creation.

## References

[1] M. Alexa. Linear combination of transformations. *ACM Trans. Graph.*, 21:380–387, July 2002.

[2] C. Andújar, P. Vázquez, and M. Fairén. Way-finder: guided tours through complex walkthrough models. In *Computer Graphics Forum*, volume 23, pages 499–508. Wiley Online Library, 2004.

[3] C. Buehler, M. Bosse, and L. McMillan. Non-metric image-based rendering for video stabilization. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–609 – II–614 vol.2, 2001.

[4] D. Burns and R. Osfield. Open scene graph a: Introduction, b: Examples and applications. In *Proceedings of the IEEE Virtual Reality 2004*, page 265. IEEE Computer Society, 2004.

[5] N. Burtnyk, A. Khan, G. Fitzmaurice, R. Balakrishnan, and G. Kurtenbach. Stylecam: interactive stylized 3d navigation using integrated spatial & temporal controls. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, UIST '02, pages 101–110, New York, NY, USA, 2002. ACM.

[6] N. Burtnyk, A. Khan, G. Fitzmaurice, and G. Kurtenbach. Showmotion: camera motion based 3d design review. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, I3D '06, pages 167–174, New York, NY, USA, 2006. ACM.

[7] M. Christie, P. Olivier, and J.-M. Normand. Camera control in computer graphics. *Computer Graphics Forum*, 27(8):2197–2218, 2008.

[8] C. Cyr and B. Kimia. A similarity-based aspect-graph approach to 3d object recognition. *International Journal of Computer Vision*, 57(1):5–22, 2004.

[9] J. S. Downs, M. B. Holbrook, S. Sheng, and L. F. Cranor. Are your participants gaming the system?: screening mechanical turk workers. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 2399–2402, New York, NY, USA, 2010. ACM.

[10] S. Fleishman, D. Cohen-Or, and D. Lischinski. Automatic camera placement for image-based modeling. *Computer Graphics Forum*, 19(2):101–110, 2000.

[11] M. L. Gleicher and F. Liu. Re-cinematography: Improving the camerawork of casual video. *ACM Trans. Multimedia Comput. Commun. Appl.*, 5:2:1–2:28, October 2008.

[12] V. GOVINDU. Lie-algebraic averaging for globally consistent motion estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004.

[13] A. Hawkins and C. M. Grimm. Camera keyframing using linear interpolation of matrices. *journal of graphics, gpu, and game tools*, 12(3):55–69, 2007.

[14] Z. He and C. Zhao. A recording method of virtual reality interaction. In *Communication Software and Networks, 2009. ICCSN '09. International Conference on*, pages 666 –669, feb. 2009.

[15] L. Kavan, S. Collins, C. OSullivan, and J. Zara. Dual quaternions for rigid transformation blending. *Technical report TCDCS200646 Trinity College Dublin*, 2009.

[16] E. Kilgariff and R. Fernando. The geforce 6 series gpu architecture. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[17] A. Kittur, E. H. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 453–456, New York, NY, USA, 2008. ACM.

[18] L. Kozlowski and K. Bryant. Sense of direction, spatial orientation, and cognitive maps. *Journal of Experimental Psychology: Human Perception and Performance*, 3(4):590, 1977.

[19] E. Krahmer and I. van der Sluis. A new model for generating multimodal referring expressions. In *Proceedings of 9th European Workshop on Natural Language Generation (ENLG-2003)*, pages 47–54. Citeseer, 2003.

[20] A. Lablack, F. Maquet, N. Ihaddadene, and C. Djeraba. Visual gaze projection in front of a target scene. In *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, pages 1839 –1840, 28 2009-july 3 2009.

[21] F. Liu, M. Gleicher, H. Jin, and A. Agarwala. Content-preserving warps for 3d video stabilization. *ACM Transcations on Graphics (Proceedings of ACM SIGGRAPH 2009)*, 28(3), 2009.

[22] K. Mühler, M. Neugebauer, C. Tietjen, and B. Preim. Viewpoint selection for intervention planning. In *IEEE/eurographics symposium on visualization (EuroVis)*, pages 267–274. Citeseer, 2007.

[23] A. Murgia, R. Wolff, W. Steptoe, P. Sharkey, D. Roberts, E. Guimaraes, A. Steed, and J. Rae. A tool for replay and analysis of gaze-enhanced multiparty sessions captured in immersive collaborative environments. In *Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, DS-RT '08, pages 252–258, Washington, DC, USA, 2008. IEEE Computer Society.

[24] R. O'Brien and M. Kaiser. Manova method for analyzing repeated measures designs: an extensive primer. *Psychological Bulletin*, 97(2):316, 1985.

[25] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krger, A. E. Lefohn, and T. J. Purcell. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26(1):80–113, 2007.

[26] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P. Haeberli. Fast shadows and lighting effects using texture mapping. *SIGGRAPH Comput. Graph.*, 26:249–252, July 1992.

[27] C. Shannon and W. Weaver. *The mathematical theory of communication*. Citeseer, 1959.

[28] T. Sun, J. Karlsson, W. Peng, P. Wall, and Z. Langford. Managing interactions in the collaborative 3d docuspace for enterprise applications. In *Proceedings of the International Conference on Human-centric Computing 2011 and Embedded and Multimedia Computing 2011: HumanCom and EMC 2011*, volume 102, page 135. Springer, 2011.

[29] B. T. Truong and S. Venkatesh. Video abstraction: A systematic review and classification. *ACM Trans. Multimedia Comput. Commun. Appl.*, 3, February 2007.

[30] M. Tsang, G. W. Fitzmaurice, G. Kurtenbach, A. Khan, and B. Buxton. Boom chameleon: simultaneous capture of 3d viewpoint, voice and gesture annotations on a spatially-aware display. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, UIST '02, pages 111–120, New York, NY, USA, 2002. ACM.

[31] E. A. Wernert and A. J. Hanson. A framework for assisted exploration with collaboration. *Visualization Conference, IEEE*, 0:39, 1999.

[32] L. Williams. Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph.*, 12:270–274, August 1978.