Video Summagator: An Interface for Video Summarization and Navigation

Cuong Nguyen, Yuzhen Niu, and Feng Liu
Portland State University
Portland, OR 97207-0751
{cuong3,yuzhen,fliu}@cs.pdx.edu

ABSTRACT

This paper presents *Video Summagator* (*VS*), a volume-based interface for video summarization and navigation. *VS* models a video as a space-time cube and visualizes the video cube using real-time volume rendering techniques. *VS* empowers a user to interactively manipulate the video cube. We show that *VS* can quickly summarize both the static and dynamic video content by visualizing the space-time information in 3D. We demonstrate that *VS* enables a user to quickly look into the video cube, understand the content, and navigate to the content of interest.

Author Keywords

Video Visualization; Navigation; Summarization

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: User Interfaces—*Graphical User Interfaces*

INTRODUCTION

Video capturing and displaying devices are ubiquitous now. Capturing, sharing, and watching videos has become a common practice. Meanwhile, the human video interaction tools, mainly video players, remain almost unchanged in user interface design. A typical video player consists of a display window to show a video one frame at a time and a slider to navigate through the video. While such a player prevails, it is often inconvenient for video browsing. It cannot fully and quickly convey important aspects of the video content.

This paper describes *Video Summagator* (*VS*), a 3D volume-based interface for video summarization and navigation. *VS* considers a video as a cube, with time as the third dimension, as shown in Figure 1. It employs real-time volume rendering techniques to quickly visualize the cube in a 3D volume [10]. *VS* seamlessly combines video summarization and navigation. By visualizing and deforming the video cube, *VS* can quickly convey important visual content that otherwise would be time-consuming to find and easy to miss. *VS* allows a user to look into the video cube, summarize the video, quickly identify the content of interest and navigate to it.

CHI 12, May 5–10, 2012, Austin, Texas, USA.

Figure 1. ***Video Summagator*** **visualizes a video in 3D, allowing a user to look into the video cube, and enables rapid visualization and navigation.**

The main idea behind *VS* is to visualize a time sequence in 3D. *VS* is inspired by recent research that summarizes a video clip into a 2D image [1, 5, 12]. *VS* extends the 2D display space of these methods to 3D, which provides more visualization space and better conveys space-time video content. It also extends these methods with interactive video manipulation and visualization, and handles a wider variety of camera and object motion. An important aspect of *VS* is direct manipulation for video browsing inspired by [9, 13]. But *VS* is different from these methods because it visualizes the whole video, enables a user to manipulate the entire video cube, and then navigate to the interesting part by clicking the corresponding region in the cube.

RELATED WORK

A rich literature exists on video summarization that selects a key frame sequence or key segments from an input video as a concise video representation. A good review can be found in [22]. Our work is particularly relevant to video summarization methods that represent a video clip as a still picture like panorama and action synopsis [1, 5, 12, 21]. These methods can create high-quality pictures; however, they are typically time-consuming. Compared to these methods, our work models a video as a cube and uses real-time volume rendering techniques to show the video cube in 3D. Our method supports interactive video cube manipulation and navigation. While our method sometimes cannot create final

images equal in quality to these dedicated video summarization methods, our results are good for quick video browsing.

Our work builds on video cubism research that models a video as a space-time volume and manipulates a cut surface through the volume to display the video content [11]. This method has also been extended for non-photorealistic rendering [16]. Unlike these methods, our method visualizes the whole video volume instead of the cut surface to more concisely summarize the video. Our method also allows interactive volume manipulation to visualize complex motion. Bennett and McMillan extend video cubism for convenient space-time video editing [3]. Their method does not work for visualization. Daniel and Chen visualize a video in a transparent cube or other pre-defined volume [8]. This technique has been extended to visualize human activities [4, 20]. Our work extends it to support interactive video cube manipulation to facilitate video summarization and browsing.

Our work is relevant to research on video player design. Adaptive fast-forwarding is designed for quick video browsing with predefined semantic rules [6]. The video speed and playing speed are decoupled and content analysis is adopted to play interesting shots at an intelligible speed [18]. Besides the player bar, maps have also been used for browsing domain-oriented videos, such as lecture videos [7] and tour videos [17]. Our work is inspired by direct manipulation that allows users to browse videos by directly dragging the video content and frees them from the player bar [9, 13, 14, 15]. Unlike these methods, our method enables a user to manipulate the whole video cube and navigate to the interesting part by clicking the corresponding region in the cube. Our work is also relevant to video collection browsing in 3D that reconstructs the 3D scene and uses video-based rendering to create novel views [2]. Our method is different in that it does not require 3D scene reconstruction although it visualizes the video content in 3D. Our 3D visualization typically does not correspond to the 3D scene structure.

VIDEO SUMMAGATOR

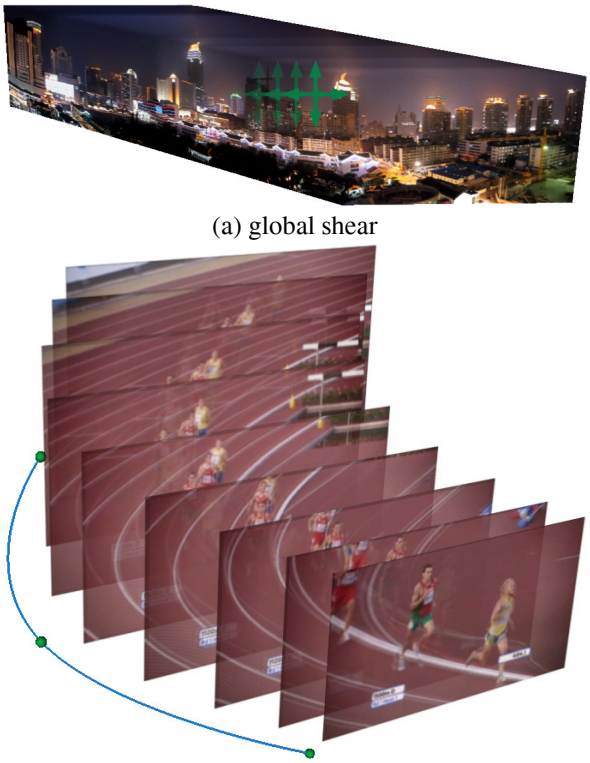
Two common types of user interaction with videos are quickly skimming through a video for overview and slowly navigating in interesting segments for detail [18]. We develop *Video Summagator* (*VS*), a 3D volume-based interface, to support both video summarization and navigation. Below we first describe how we model and render a video as a cube (volume), with time as the third dimension. We then describe how *VS* supports a user to interactively manipulate the video cube to visualize video content. We finally describe exemplary scenarios of using *VS* for summarization and navigation.

Video Cube Modeling and Rendering

We consider a video as a cube V that contains a set of voxels:

$$V = \{v_{x,y,t} | 1 \leq x \leq w, 1 \leq y \leq h, 1 \leq t \leq n\} \quad (1)$$

where voxel $v_{x,y,t}$ corresponds to pixel (x, y) at frame t. w and h are the video frame width and height, and n is the number of video frames. Each voxel is associated with a set of features, including the voxel's 3D coordinates, color, and opacity. A voxel takes its color from the corresponding pixel. Its coordinates and opacity are controlled by a user for video summarization and navigation, as described in the next subsection. With the volume features, *VS* uses the state-of-the-art volume rendering techniques from the Visualization Toolkit [1] to quickly render the video cube.



(a) global shear

(b) skeleton-based deformation

Figure 2. Video cube deformation. *VS* supports a user to globally shear the video cube to create a panoramic summarization (a). The user can also use a spline interface to deform the cube to better arrange the video content in the 3D space. For the example in (b), the user define three control points (in green) to define a new spline to deform the cube.

User Interaction

A user interacts with *VS* by controlling three features of a video cube: opacity, shape, and video frame sampling.

The voxel opacity value ranges from 0 to 1, with 0 being fully transparent and 1 fully opaque. *VS* uses off-the-shelf computer vision algorithms to detect dynamic voxels and assigns higher opacity values to them than the stationary ones to clearly convey the video dynamics. *VS* allows a user to control the opacity values of the dynamic, static, and boundary voxels using a slider interface.

VS supports a user to translate, rotate, scale, and deform a video cube to clearly visualize its content. These user interactions are mapped to the coordinates of the voxels. We use the standard 3D interface for translation, rotation and scaling. We design two interactive deformation modes in *VS* . One is to shear the cube globally, as shown in Figure 2 (a). This is implemented by applying a shearing transformation to the

[1] http://www.vtk.org

(a) front view

(b) side view

Figure 3. Video summarization. Rendering a video cube from the front view often suffers from self-occlusion, as shown in (a). This problem can often be solved by simply rotating the video cube, as shown in (b).

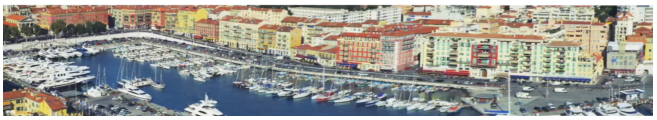
cube. The other is skeleton-based local shearing transformation. We define a video skeleton as a spline that is perpendicular to each video frame. For the original cube, its skeleton is a line parallel to the t axis. We provide a spline-based interface for adjusting the skeleton to deform the video cube and create the output volume. As shown in Figure 2 (b), a user can create control points and move them to define a new skeleton.

VS uses two methods to select key frames, as rendering all the video frames will lead to a cluttered visualization. The first is frame-based uniform sampling (default rate 1/20). A user can control the sampling rate through a slider-based interface. Uniform sampling is used to create all the examples except Figure 5. The second is importance sampling, which is more applicable for a long video. We consider that the probability of a frame being a key frame is proportional to the amount of foreground motion. Then, we select a sequence of key frames according to this distribution [19].

VS allows a user to get a quick overview of a video typically with two steps: adjust the opacity values of the stationary and dynamic pixels to see the activities and rotate the video cube if there exists self-occlusion. Since *VS* renders the cube at an interactive speed, these two steps together take less than 5 seconds, which is typically shorter than the video length. With a preview of the video, the summarization can be further improved by adjusting the cube parameters.

Video Summarization

We now show how *VS* can be used to summarize a range of videos with different camera and scene motion.



(a) scene panorama

(b) action synopsis

Figure 4. Video summarization. A user creates scene panorama or action synopsis by globally shearing the video cube.

Static camera with moving objects. Figure 3 shows a street show video captured by an almost static camera. Dynamic voxels are assigned bigger opacity values than the background ones to reveal the magician's activity. For this video, since the camera is almost static and the magician moves in a small region, looking into the video cube suffers from self-occlusion, as shown in Figure 3 (a). This problem can be simply solved by rotating the cube, as shown in Figure 3 (b). For (b), the opacity values for the left, back, and bottom cube faces are set to a big value to better visualize the background scene.

Panning camera motion. Consider a video with the camera panning horizontally. *VS* enables a user to easily create a panorama by shearing the video cube, as shown in Figure 4 (a). For this application, the opacity values for all the voxels are uniformly set to 1. If the video contains a moving object, *VS* then creates an action synopsis that visually depicts the object activity, as shown in Figure 4 (b). For this example, the moving objects are automatically separated horizontally as the cube is being sheared. The opacity value for the back and right cube face is set to 1 to visualize the background.

Moving camera following actions. Consider a video with the camera following the athletes. *VS* supports a user to first create a mosaic image by shearing the cube globally. Then, the user refines the result with local adjustments using the skeleton-based interface, as shown in Figure 2 (b). Since *VS* provides online visual feedback, this can be done quickly. For this example, the opacity values for the voxels on the key frames are set to 0.9 and the others are set to 0.

Video Navigation

VS also provides an intuitive interface for video navigation. *VS* associates the (possibly curved) time axis with a 3D volume-based summarization. With the overview of the video, a user can quickly navigate to the interesting video segment by clicking and selecting the corresponding area in the summarization. For example, given a surveillance video, a user rotates the whole video cube so that the evolution of video content shows clearly in the display window. Looking into the video cube, the user spots the region in the cube with human activity and selects this region by drawing a line indicating the region of interest, as shown in Figure 5. *VS* maps the selected region to the time axis by projection and automatically navigates to the corresponding video segment.

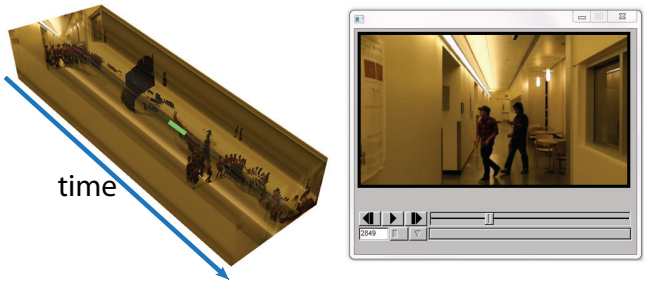


Figure 5. Video navigation. Video Summagator enables a user to look into the video cube and navigate to the interesting part by selecting the corresponding area (marked in green) in the 3D summarization.

To handle a long video, we down-sample the video frame size and adaptively sample the video frames using importance sampling. On a desktop machine with 6G memory and AMD Phenom II X6 2.8 GHz CPU, *VS* currently can allow a user to interactively manipulate a video cube with as many as 3000 key frames with frame size 480×270. Figure 5 shows a video cube that samples 1500 frames from an input video with totally 10000 frames.

CONCLUSION

This paper described Video Summagator, a 3D volume-based interface for interactive video summarization and navigation. We model a video as a space-time cube and visualize the video cube in a 3D volume using volume rendering techniques. We show that Video Summagator supports a user to manipulate the video cube to quickly summarize a video, identify the content of interest, and navigate to it.

Currently, we use off-the-shelf computer vision algorithms for moving object detection, which is sometimes not very reliable and causes visual artifacts, as shown in Figure 1. The advance in computer vision research can benefit our results. In future, we plan to extend *VS* to handle a streaming video such as webcam videos. We also plan to integrate more video analysis to better handle complex videos. Our visualization is not necessarily physically correct, so we plan to design user study to more thoroughly evaluate how people perceive the video content from our summarization.

REFERENCES

1. Assa, J., Caspi, Y., and Cohen-Or, D. Action synopsis: pose selection and illustration. *ACM Trans. Graph. 24* (July 2005), 667–676.
2. Ballan, L., Brostow, G. J., Puwein, J., and Pollefeys, M. Unstructured video-based rendering: interactive exploration of casually captured videos. *ACM Trans. Graph. 29* (2010), 87:1–87:11.
3. Bennett, E. P., and McMillan, L. Proscenium: a framework for spatio-temporal video editing. In *ACM Multimedia* (2003), 177–184.
4. Botchen, R. P., Bachthaler, S., Schick, F., Chen, M., Mori, G., Weiskopf, D., and Ertl, T. Action-based multifield video visualization. *IEEE Trans. Vis. Comput. Graphics 14* (2008), 885–899.
5. Caspi, Y., Axelrod, A., Matsushita, Y., and Gamliel, A. Dynamic stills and clip trailers. *Vis. Comput. 22* (September 2006), 642–652.
6. Cheng, K.-Y., Luo, S.-J., Chen, B.-Y., and Chu, H.-H. Smartplayer: user-centric video fast-forwarding. In *ACM CHI* (2009), 789–798.
7. Corsten, C. Dragonfly: spatial navigation for lecture videos. In *ACM CHI EA* (2010), 4387–4392.
8. Daniel, G., and Chen, M. Video visualization. In *IEEE Visualization* (2003), 409–416.
9. Dragicevic, P., Ramos, G., Bibliowitcz, J., Nowrouzezahrai, D., Balakrishnan, R., and Singh, K. Video browsing by direct manipulation. In *ACM CHI* (2008), 237–246.
10. Engel, K., Hadwiger, M., Kniss, J., Rezk-Salama, C., and Weiskopf, D. *Real-Time Volume Graphics*. 2006.
11. Fels, S., Lee, E., and Mase, K. Techniques for interactive video cubism. In *ACM Multimedia* (2000), 368–370.
12. Goldman, D. B., Curless, B., Salesin, D., and Seitz, S. M. Schematic storyboarding for video visualization and editing. *ACM Trans. Graph. 25* (2006), 862–871.
13. Goldman, D. B., Gonterman, C., Curless, B., Salesin, D., and Seitz, S. M. Video object annotation, navigation, and composition. In *ACM UIST* (2008), 3–12.
14. Karrer, T., Weiss, M., Lee, E., and Borchers, J. Dragon: a direct manipulation interface for frame-accurate in-scene video navigation. In *ACM CHI* (2008), 247–250.
15. Karrer, T., Wittenhagen, M., and Borchers, J. Pocketdragon: a direct manipulation video navigation interface for mobile devices. In *MobileHCI* (2009), 47:1–47:3.
16. Klein, A., Sloan, P.-P. J., Colburn, R. A., Finkelstein, A., and Cohen, M. F. Video cubism. In *Microsoft Research Technical Report* (2001), MSR–TR–2001–45.
17. Pongnumkul, S., Wang, J., and Cohen, M. Creating map-based storyboards for browsing tour videos. In *ACM UIST* (2008), 13–22.
18. Pongnumkul, S., Wang, J., Ramos, G., and Cohen, M. Content-aware dynamic timeline for video browsing. In *ACM UIST* (2010), 139–142.
19. Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. *Numerical Recipes: The Art of ScientiŒc Computing*. 2007.
20. Romero, M., Vialard, A., Peponis, J., Stasko, J., and Abowd, G. Evaluating video visualizations of human behavior. In *ACM CHI* (2011), 1441–1450.
21. Teodosio, L., and Bender, W. Salient stills. *ACM Trans. Multimedia Comput. Commun. Appl. 1*, 1 (2005), 16–36.
22. Truong, B. T., and Venkatesh, S. Video abstraction: A systematic review and classification. *ACM Trans. Multimedia Comput. Commun. Appl. 3*, 1 (2007), 3:1–3:37.