INTEGRATING VIRTUAL CAMERA CONTROLS INTO DIGITAL VIDEO

Ramazan Savaş Aygün

Computer Science Department
University of Alabama in Huntsville
email:raygun@cs.uah.edu

Aidong Zhang

Dept. of Computer Science and Engineering
State University of New York at Buffalo
email:azhang@cse.buffalo.edu

ABSTRACT

Virtual camera controls (VCCs) for digital video enable the viewers to visualize interesting objects from their perspective. VCCs also allow to play the video from different angles. In this sense, VCCs do not only support visualization and browsing capabilities but also support the playback from different angles. VCC management requires accurate global motion estimation and accurate sprite generation. After motion parameters are detected and the precise sprite is generated, virtual camera controls are used to manipulate the sprite and the original frames to allow interactive spatial browsing & playback that enables panning, tilting, and zooming.

1. INTRODUCTION

In recent years, camera control has been studied by using different strategies at hardware and software levels. Although temporal browsing of video is supported by many applications, incorporation of spatial browsing has been delayed due to the late improvements in global motion estimation, sprite generation, and manufacturing of special cameras. In traditional applications, cameramen are responsible for capturing the most interesting events, activities, and scenes. Therefore, the stored video contains the scene recorded from the perspective of the cameramen. It has been realized that what has been important to the cameramen is not necessarily the most important object or scene for all viewers. There are a couple of ways to handle the camera control: transforming to a virtual environment, automating the camera to follow the pre-defined objects, and using advanced camera like panoramic camera and then extracting important regions.

Although spatial browsing is enabled in virtual environments, the scene is generated by using computer graphics tools as in interactive walk through applications [8]. However, these applications do not give the feeling of browsing through real images. In [1], a spatial navigation system is proposed by modeling the original video using VRML (Virtual Reality Modeling Language). The user is able to navigate using VRML and access spatial locations. Panoramic cameras are used to capture the environment in [5, 3]. The region of interests are detected from the panoramic scene and these parts are presented [5]. This approach is not applicable to previously developed videos. A spatially indexed camera is explained for navigation in [3]. VideoSpaceIcon [7] is a tool that allows to view video icons. One sample application is reproduction of object motion by clicking the mosaic. VideoSpaceIcon is a visualization tool. In [6], the salient stills are described and how salient stills could be generated where a zoom operation exists.

In this paper, our goal is to incorporate virtual camera controls (VCCs) into digital video. We do not assume that the scene is predefined. We consider scenes that are captured using a single camera. VCC management is a natural way of incorporation of spatial browsing & playback. The user can play the video using camera operations like pan left and right, tilt up and down, and zoom in and out. The background mosaic or sprite has to be extracted to provide these interactions. However, the ordinary sprite generation methods have some handicaps to be used directly in spatial browsing. For example, in MPEG-4 [4], the generation and playback of the new generated video depend on the motion compensation algorithms since the motion cannot be estimated accurately. The user can spatially browse the video and release browsing to see the actual video at any frame display. Our goal is not only to show a video object from a different angle. Our goal is also to play the video from different angles. Therefore, our system is not only a browsing system but also a playback system.

This paper is organized as follows. In the following section, the components of the system are explained briefly. The virtual camera controls are expressed in Section 3. The experiments are explained in Section 4. The last section concludes our paper.

2. SYSTEM COMPONENTS

Virtual camera control management has two components: preprocessing module and spatial browsing & playback module (Figure 1). In the preprocessing phase, the global motion estimation between frames and the generated sprite is

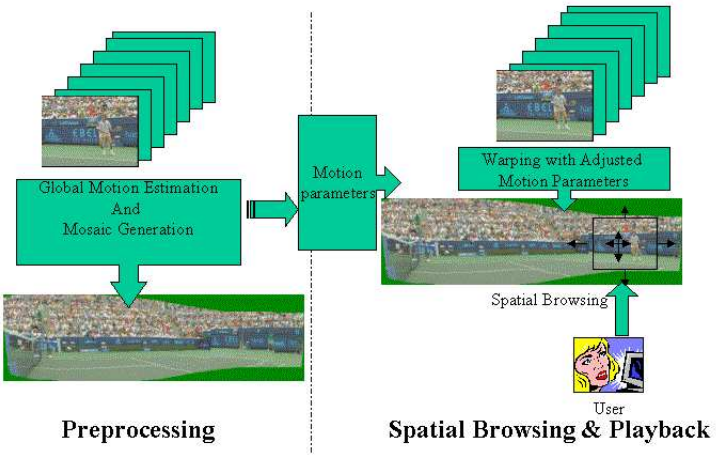

Figure 1: Components of Virtual Camera Control Management.

performed; and the sprite is generated. Since the sprite is used in spatial browsing, our goal is to generate a high resolution sprite while reducing blurring. The motion parameters for each frame is stored and made available for spatial browsing & playback. The user can browse the video using panning, tilting, and zooming operations. The frames and the sprite are merged by adjusting motion parameters.

3. VIRTUAL CAMERA CONTROL

Spatial browsing requires integration of sprite and the frames. In [5], since the original scene is a panoramic scene, the region of interest from the scene is detected and presented to the user. In their case, there is no need to integrate frames and the background sprite. In our case, the sprite is generated using the frames and it is important how these frames are related spatially.

VCC management is different from the digital zoom operation, which takes a high resolution image and then presents a cropped low-resolution image. VCC is applied on video and the history of the background in terms of sprite is exploited to provide VCCs. The use of VCCs is one of the most user-friendly ways of accessing and playing the spatial content of a video. The original frame size is kept the same. It actually gives the feeling of using a camcorder. In previous approaches [7], [6], the access to the spatial content is not natural as camera controls.

The video can be seen in two ways at any time of the video: original display and with camera controls. The user can gain the camera control at any frame display at any time by clicking the camera control gain button shown in Figure 2. The user can also release the virtual camera control and watch the original video by again clicking the camera control gain button. When the user gains the camera control, the

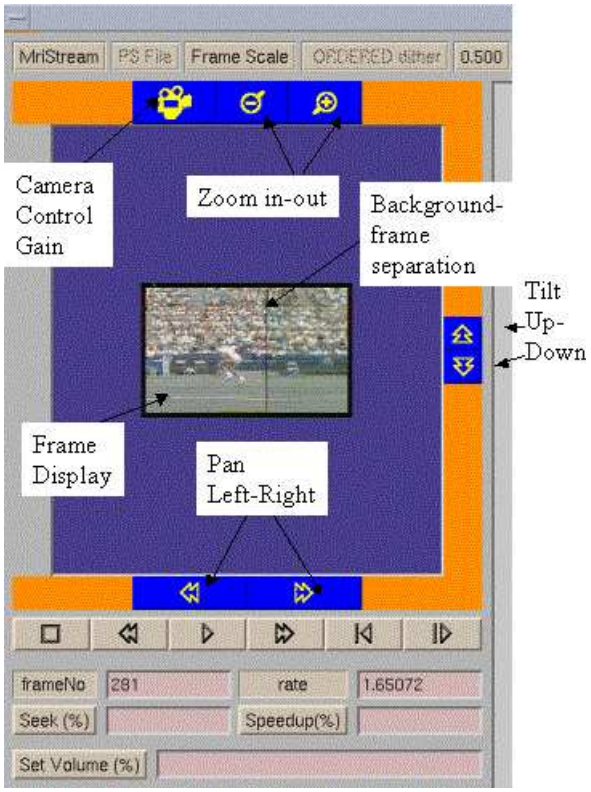

Figure 2: User interface for virtual camera controls.

camera can be moved left, right, up, and down and zoomed in and out (Figure 2). This yields camera stabilization and provides spatial browsing & playback.

Spatial browsing & playback can be performed using the segmentation masks of the objects if they are available or using the complete frames. When objects are available, they are warped into the background. If the objects are not available, original frames are warped into the background.

3.1. Camera Stabilization

When the user gains the camera control, the camera is stabilized. In this case, since the camera is static, the object might enter and leave the scene if the original video is tracking the object.

Let f_c frame be the frame when the user gained the camera control. All the other frames have to be mapped onto the background generated from the sprite. Since all the motion parameters are kept according to the initial frame in the video, the relative motion with respect to the f_c has to be calculated. Let $M^m(t)$ be the motion parameter for frame f_t with respect to frame f_m. For affine motion in Cartesian coordinate system, the parameters for frame f_t are calculated using

$$M^c(t) = (M^0(c))^{-1} \times M^0(t) \quad (1)$$

If each frame f_t is warped according to the new relative motion parameters, the camera is stabilized.

3.2. Camera Operations

The user can perform pan left and right, tilt up and down, and zoom in and out. These operations are performed in the transformed coordinates. We have 3 parameters to handle panning, tilting and zooming: $panLeftRight$ (pLR), $tiltUpDown$ (tUD), and $zoomInOut$ (zIO). We also have 3 thresholds to determine the strength of the interaction: τ_p, τ_t, and τ_z. If the user clicks pan right button, $pLR \leftarrow pLR + \tau_p$. If the user clicks pan tilt up button, $tUD \leftarrow tUD - \tau_t$. If the user clicks zoom in button, $zIO \leftarrow zIO/\tau_z$. If the user clicks zoom out button, $zIO \leftarrow zIO * \tau_z$.

The environment should be set according to the new user settings. The background and the foreground should be updated. Panning and tilting require modification on the translational parameters. The transformation matrix becomes

$$\begin{bmatrix} a_2^0(c) * zIO & a_3^0(c) * zIO & a_0^0(c) - pLR \\ a_4^0(c) * zIO & a_5^0(c) * zIO & a_1^0(c) - tUD \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

3.3. Data for Spatial Browsing & Playback

The motion parameters must be available for the spatial browsing & playback. In MPEG-4 sprite coding, the coordinates of trajectories are maintained for generation of motion parameters. The upcoming frames are warped according to the first frame in the sequence. For each frame, the motion parameters are estimated and stored in the database. After warping each frame, the center of the frame in the sprite frame is also maintained.

Since the frames are warped onto the original frame, the relative motion parameters are maintained instead of the motion parameters between the sequential frames. The motion transformation can be written briefly as:

$$v' = Mv \quad (3)$$

where M contains the motion parameters for the first matrix; v is the coordinate in the previous frame; and v' is the transformed coordinate. The relative motion is computed as

$$v'' = M'v' = M'(Mv) = (M'M)v \quad (4)$$

where v'' is the vector for the new transformed coordinates; M' and t' hold the current motion parameters; and M and t hold the motion parameters up to the current frame.

(a) ‘ (b)

Figure 3: (a) Frame 186 of ’coastguard’ sequence (b) frame 186 after panning left.

For each frame f_i, we have the feature set $F_i = \{a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, center_x, center_y\}$ where a_i is the motion parameters and $center_x$ and $center_y$ are the center coordinates of the frame in the sprite.

4. EXPERIMENTS

We have tested our tool on MPEG test sequences ’stefan’, ’foreman’, ’coastguard’ and our lecture videos. For stefan, we also have the segmentation mask and used it in the sprite generation of ’stefan’ sequence. The affine motion model is used in the motion estimation of ’foreman’, ’stefan’ and lectures. Translational motion model is used in the estimation of ’coastguard’. We have used these motion models to be comparable with other work in the literature. We have applied the sprite generation method in [2] to reduce the blurring in the sprite. Figure 4 shows a sprite for the stefan sequence. We used the segmentation mask for generation of the mask. Without segmentation mask, we also get a similar sprite since the foreground object displaces its location frequently. Figure 2 depicts a snapshot from ’stefan’ sequence using the sprite in Figure 4. Figure 3 shows frame 186 of the ’coastguard’ sequence before and after panning.

During our experiments, we have reduced the frame rate to be able to visualize the camera operations. VCC has two stages: grabbing the area from the sprite and warping the current frame with camera motion parameters from VCC. Grabbing of the area takes around 16ms and warping the current frame takes around 46ms. We have obtained these results without any optimization under Java environment. We have run these performance experiments on Pentium 4 processor machine. These results indicate decrease in frame rate. However, in some cases either of the stages may be eliminated due to camera motion parameters from VCC. Sometimes, the new view area is the out of the scope of the current frame (i.e., only first stage is needed) or the new view area is within the current frame (i.e., only second stage is needed). This work is left as a further research.

The moving objects in the video have to be captured completely. If a video object does not appear completely

Figure 4: Sprite for 'stefan'.

Figure 5: Object partially seen in Frame 110 of 'coastguard' sequence

in a video frame, the invisible parts are still invisible during spatial browsing & playback. For example, it is not possible to view the partially seen object completely in frame 110 of coastguard sequence by using the sprite (Figure 5). There are a couple of ways to avoid this problem. One of them is to limit spatial browsing & playback when the video object is not captured completely.

We assume that the camera does not make significant rotational motion around its axis. In such a case, the generated sprite should be mapped to a cylindrical sprite. The approach can also be applied for Video $360'$ applications. In those cases, the sprite has to be static.

5. CONCLUSION AND FUTURE WORK

The virtual camera controls require high quality sprite generation. Once the motion vectors and the sprite are generated, virtual camera controls provide interactive spatial browsing. Virtual camera controls provide panning, tilting and zooming interactions. Virtual camera controls allow the user to gain and release the camera control at any frame display. When the camera control is gained, all frames are mapped according to the frame which camera control is gained. In addition to browsing, it enables camera stabilization. One of the applications of VCCs is distance education where students can view the classroom, the lecturer, and the board using VCCs. This gives the feeling that student is actually following the class.

The display rate may decrease during camera stabilization since each frame has to be mapped on the stabilized frame. Efficient strategies need to be developed for playing the video in the nominal rate when the camera is stabilized. For example, there are cases when the stabilized frame and the current frame may not overlap at all.

6. REFERENCES

[1] S. Arbeeny and D. Silver. Spatial navigation of video streams. In *ACM Multimedia 2001 Conference Proceedings*, pages 467–470, Ottawa, Canada, October 2001.

[2] R. Aygun and A. Zhang. Reducing Blurring-Effect in High Resolution Mosaic Generation. In *International Conference on Multimedia Expo*, Lausanne, Switzerland, August 2002.

[3] D. Kimber, J. Foote, and S. Lertsithicai. Flyabout: Spatially indexed panoramic video. In *ACM Multimedia 2001 Conference Proceedings*, pages 339–347, Ottawa, Canada, October 2001.

[4] T. Sikora. The mpeg-4 video standard verification model. *IEEE Trans. Circuits Syst. Video Technology*, 7:19–31, February 1997.

[5] X. Sun, J. Foote, D. Kimber, and B. Manjunath. Panoramic video capturing and compressed domain virtual camera control. In *ACM Multimedia 2001 Conference Proceedings*, pages 329–338, Ottawa, Canada, October 2001.

[6] L. Teodosio and W. Bender. Salient video stills: content and context preserved. In *Proceedings of the first ACM international conference on Multimedia*, pages 39–46, Anaheim, California, September 1993.

[7] Y. Tonomura, A. Akutsu, K. Otsuji, and T. Sadakata. VideoMAP and VideoSpaceIcon: Tools for Anaomizing Video Content. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 131–136, Amsterdam, Netherlands, May 1993.

[8] A. Wilson, M. C. Lin, M. Dinesh, B.-L. Yeo, and M. Yeung. A video-based rendering acceleration algorithm for interactive walkthroughs. In *ACM Multimedia 2000 Conference Proceedings*, pages 75–84, Los Angeles, USA, November 2000.