Immersive Panoramic Video

Thomas Pintaric, Ulrich Neumann & Albert Rizzo
Integrated Media Systems Center
University of Southern California
Los Angeles CA 90089
tpintari@usc.edu, uneumann@usc.edu, arizzo@usc.edu

A user navigates a panoramic video environment using a head-mounted display and tracker.

Introduction

Television and video images pervade our professional and home environments. For over fifty years video images have provided a "virtual eye" into distant times and locations. Over the same period, video technology has matured from gray-scale images to big-screen color and digitally processed imagery. One aspect of both the delivery technology and the content creation has remained largely unchanged however — *the view is controlled at the source and identical for all observers*. Panoramic video overcomes the passive and structured limitations of how video imagery is presented and perceived. The recent convergence of camera, processing, and display technologies make it possible to consider providing each viewer with individual control of their viewing direction. Viewers of panoramic video become virtual participants immersed in the observed scene, creating a new interactive dimension in the way people perceive video imagery within a "virtual environment".

Panoramic image acquisition is based on mosaic approaches developed in the context of still imagery [1, 2]. Mosaics are created from multiple overlapping sub-images pieced together to form a high-resolution, panoramic or wide field-of-view image. While still image mosaics and panoramas are common, we produce high-resolution panoramic video by employing an array of five video cameras viewing the scene over a combined 360-degrees of horizontal arc. Neighboring camera images share the exact same "virtual" viewpoint [3] and they overlap slightly to facilitate their merger [4, 5]. The five camera video streams feed into a digital recording and playback system developed for maintaining precise frame synchronization. The delivery of panoramic video is possible with both real time and post-processed video. The user experiences the video by wearing a head-tracked display (Fig. 3). The portion of the video in the direction of view is dynamically extracted and presented to the display in response to the user's head orientation. The challenges encountered in this process span issues in camera calibration,

image processing, compression, networking, computer graphics, and high-performance computing.

Camera Subsystem

Our approach is based on an array of multiple image sensors or cameras (Fig. 1). We make three observations about the advantages of multiple-sensor panoramic imaging systems over single sensor systems producing equivalent image quality. Firstly, at any point in time, the technology for a single high-resolution video image sensor is more expensive than multiple commodity sensors (e.g., the price of HDTV video cameras is currently more than an order of magnitude that of standard video cameras). Secondly, a single rectangular image sensor is inefficient for use in panoramic imaging since planar projections from curved mirrors or fish-eye lenses do not evenly or fully cover the pixel array (Fig. 2). Lastly, the use of multiple image sensors easily supports parallel approaches to managing the bandwidth and computation requirements in the system.

Fig 1 - Our five sensor panoramic camera system.

The disadvantages of multiple sensors include the need for calibration and optics that produce a single shared viewpoint. The shared viewpoint is often only approximated in systems used with far-field scenes [6]. Another disadvantage that often occurs with multiple sensor designs is their limited vertical field of view (FOV).

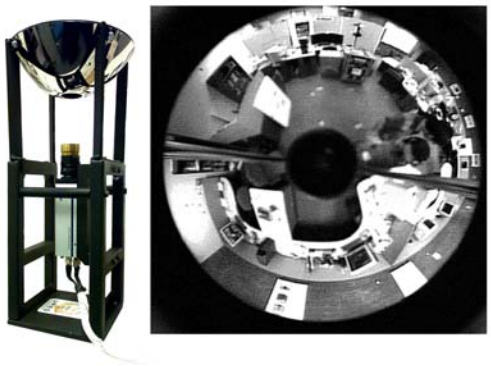

Fig 2 - A panoramic projection from a parabolic mirror leaves many sensor pixels unused.

While our methods are not restricted to any particular camera array, we describe our system in the context of a commercially available five-camera array [5]. Other camera arrays have been built using seven cameras [6] and as many as sixteen (to our knowledge). Our array produces five NTSC-format color images. Each camera is genlocked to ensure consistent video timing and it has a controller that sets parameters such as gain, hue, and shutter speed. In its final optical configuration, the camera array is statically calibrated, producing a mapping from a cylindrical panoramic projection to each camera image. This mapping takes into account the camera positions and orientations as well as lens distortions and mirror effects. These calibrations can be produced using methods employed for static image mosaics [1, 2, 7]. The end result is a camera system that logically produces a cylindrical projection of a scene with about 3520x480 pixels at 30Hz. The output format is a set of five S-Video signals and the reference video signal.

Recording Subsystem / Hardware

Our camera output is fed through a custom videotape system comprising of five SONY DSR-70 DVCam recorders and one GrassValley VPE 351 edit controller. The tape system is inline with the video signals so that recording or playback is transparent to the downstream portion of the system. The difficulties in developing this system lie primarily in identifying an edit controller that has sufficient inputs and dealing with the pragmatics of frame-accurate synchronization. Reference video is fed to the edit controller. Tapes are pre-striped with SMPT timecode and the recorders are kept in strict synchronization by the edit controller. Although this is standard video-industry methodology, in practice we find the timecode technology prone to failures whose frequency increase with tape use and recorder wear. We also note that synchronization of audio recordings with the timecode signal is problematic due to dropouts. In practice we record audio on a separate set of synchronized multi-track recorders and establish video/audio synchronization during post processing.

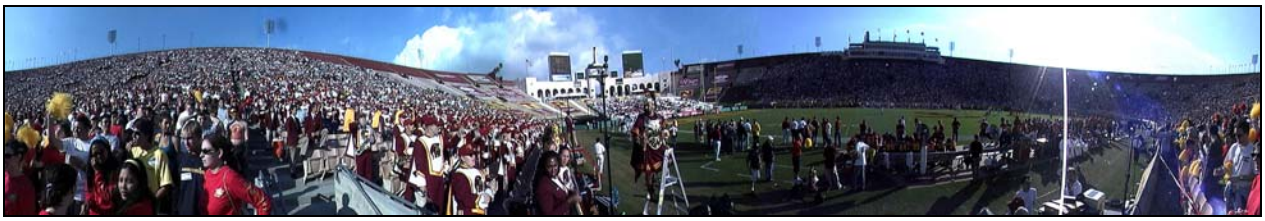

Fig. 3 - A panoramic image from our camera array taken at the USC Homecoming game at the LA Coliseum.

System Overview

Utilizing the expanding capabilities of new-generation 3D graphics hardware, we have created a framework for panoramic video production and playback. Among the main advantages of our approach over proprietary panoramic video playback systems (of which there exist only very few, e.g. the IPIX Video plug-in for Real Player) stands its ability to adapt to a variety of recording/playback configurations through an inherently modular architecture based on Microsoft's DirectShow SDK [8]. Additionally, our API enables programmers to easily implement panoramic video into any 3D environment. In our reference system, a user wears a head-mounted display with an orientation tracker (Fig 4). The video image presented in the display is extracted from the complete cylindrical projection as a function of the view direction. The footage shown here was recorded at the USC Homecoming game at the L.A. Coliseum in October 2000 (Fig 3).

Fig. 4 - A user wearing a head-mounted display with an orientation tracker.

Recording/Production Process
We allow for two main scenarios of panoramic video production:

- **Offline Production.** Fully edited on-demand panoramic movies/environments that can be distributed and played back using one of the standard media-formats (such as AVI files).
- **Live broadcasts.** Transmission of events over broadband networks as they happen.

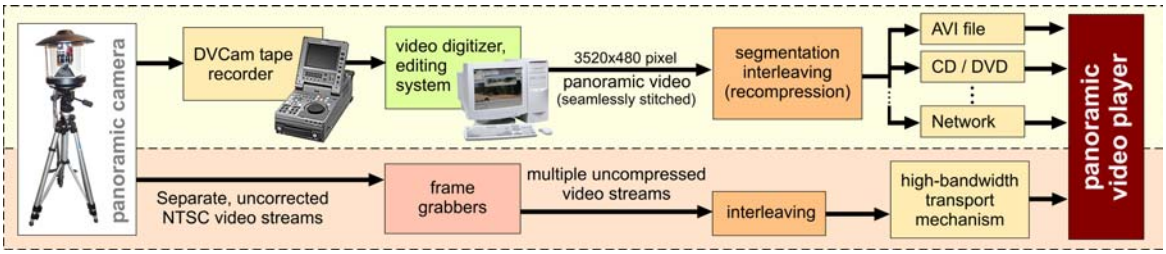

Fig 5 - The upper and lower paths describe the offline process and the live broadcast, respectively.

Offline Production
Our camera captures a 360ºx76º FOV scene, using five frame-synchronized NTSC cameras. We record the five streams and their timecode on Sony DSR-70 DVCam recorders. The post-production step involves digitizing the video using a Pinnacle DC2000 MPEG-2 encoder [10] at 704x480 pixels @ 25Mbit/sec.; aligning and un-distorting the streams in a compositing application (Adobe After Effects [11]), using our custom distortion correction plug-in with bilinear interpolation resulting in a seamlessly stitched 3520x480 video file. This large video file is then segmented into an arbitrary number of smaller region (e.g. 55 regions of 64x480 pixels each) streams that are separately compressed, interleaved and written back to a file (or streamed over a network, etc.), which can be opened by our player application.

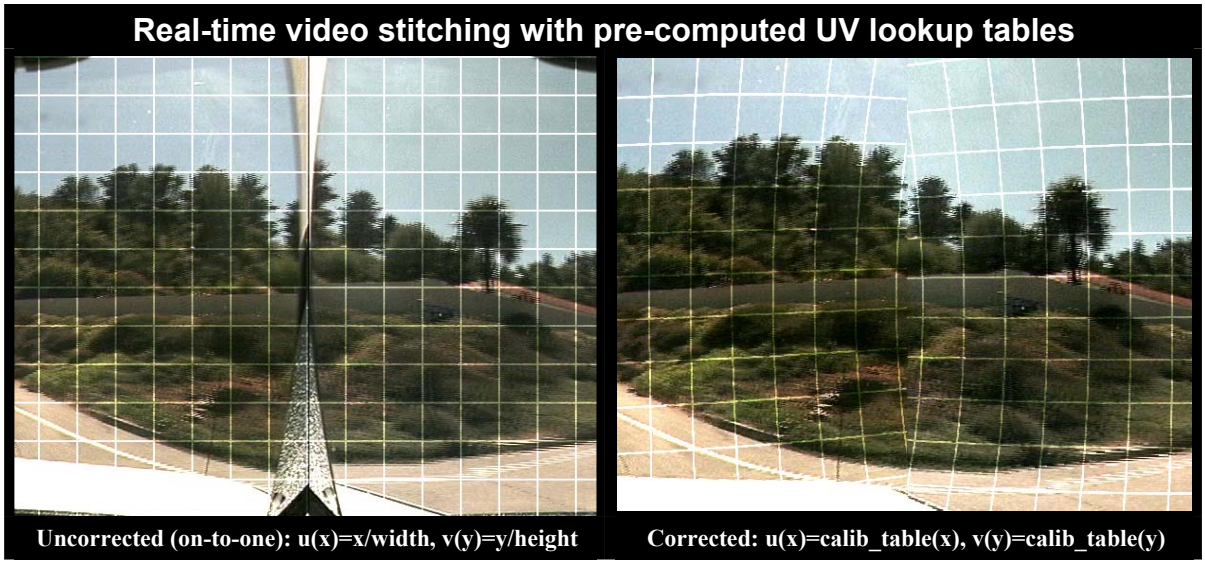

Fig. 6 - Image stitching makes use of texture coordinate tables

Live Broadcasts
This procedure varies from the previously discussed method insofar as there exists no intermediate compression/recompression and un-distortion steps. The video is acquired in an uncompressed "raw" format and sent directly to the player application. As shown, this requires a very high-bandwidth medium and so far we have only implemented this approach with the player running on the same machine reading the video through memory-to-memory transfers. Compression/decompression would have to be added to allow remote player applications over broadband networks. In this live scenario the player application is responsible for correcting the camera's lens distortion and stitching by adjusting the video texture's UV coordinate mapping to reflect the camera calibration mapping described earlier (Fig.6).

The various tasks and how they are handled in the offline and live scenarios are summarized in figure 7.

	Step	Off-line Production	Live Broadcast
RECORDING and PROCESSING	Recording	Frame-sync'd Sony DSR-70 DVCam recorders (one per camera)	n/a (directly digitising from source)
	Digitising	Pinnacle DC2000 MPEG-2 encoder at 25Mbit/sec (communicating with Sony recorders through serial link)	Imagenation PXC200 frame grabbers, capturing uncompressed 640x480 frames at 30Hz (one per camera)
	Editing	Adobe After Effects (seamless stitching using **pre-computed calibration tables**)	Limited ability to manipulate frame-buffer contents
	Processing	Segmentation into multiple tiles (120x480 pixels), interleaving with audio	No re-segmentation (video tiles remain 640x480 pixels), interleaving at frame-level
	Transfer	Bitstream is written to a file or streamed over a network	In-memory or broadband-network transfer of bitstream to playback machine
PLAYER	Receiver	Reads bitstream from arbitrary source with matching input filter and dynamically extracts visible and partially-visible video tiles	
	Display	Texture-maps video on 3D cylinder using **one-to-one UV mapping**	Texture-maps video on 3D cylinder using **pre-computed calibration tables** (Fig 6)

Fig. 7 - Differences in recording/playback procedure for off-line production and live broadcasts.

Playback Applications
Fig 8 shows the player structure. We use the real time video texture capabilities found in current PC and gaming graphics systems. The cylindrical video projection produced by the camera is textured onto a matching 3D cylindrical model and the viewing is controlled by the tracker data [11] and a real-time adjustable scale parameter. Rather than updating the texture for the complete cylinder every frame, we decode and update only the regions of the texture that can be seen during any given frame. Each region of the complete texture is encoded as an independent video stream. Multiple regions are interleaved (with audio) to create the complete panoramic data stream.

A playback application typically instantiates a DirectShow filter graph, which itself starts its single modules, called "filters", as separate threads, and a thread for rendering and window management. Based on the transport mechanism, we choose an appropriate source filter such as an asynchronous file reader for AVI files or a network interface for streaming video.

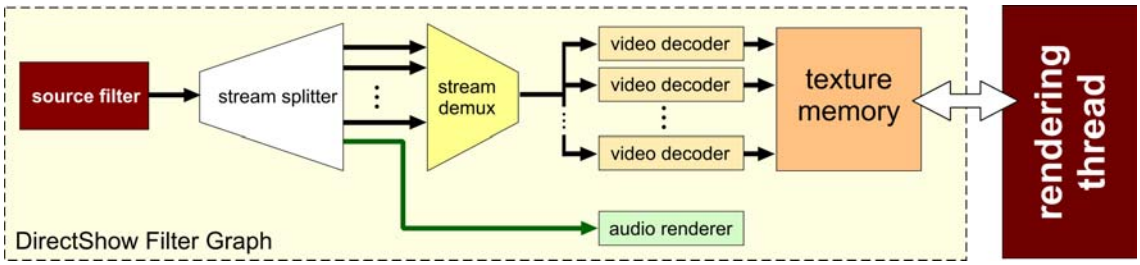

Fig. 8 - The player is built upon real time video texturing available in PC and game graphics systems.

A splitter filter separates the interleaved (binary) data read from the source filter into its single audio/video streams and passes them on in (compressed) single-frame packets. According to the currently selected viewing angle, not all of the streams are actually visible, so a demultiplexer forwards only those streams that are at least partially visible while discarding all others.

Decoding the video segments is done in parallel with the resulting uncompressed RGB frame being copied into the graphics card's texture memory. (Unfortunately this is an additional step incurred since commonly used commercial video codecs do not support an interface for directly decoding into video memory). The rendering thread synchronizes its access to texture memory with the filter graph and only updates the displayed textures after all regions belonging to the current display-frame have been updated.

Key Advantages of Our API

- Video stream(s) can be compressed and played back using any available video VfW (Video for Windows) codec.
- Various transportation mechanisms (such as network streaming) can be implemented by adding custom source filters in front of the playback graph.
- Since the rendering is done in hardware, an arbitrary projection format (planar, cylindrical, spherical) can be used at no additional (performance-)cost.
- The rendering thread (which may be implemented with any possible API that supports texture mapping, e.g. Direct3D, OpenGL, etc.) can be used to enhance the scene by drawing "true" 3D objects on top of the video environment.
- All "low-level" synchronization and timing tasks are handled by DirectShow.
- Our architecture performs its most computationally intense tasks (video decoding) in parallel threads, thus profiting from multiprocessor systems.

Practical considerations

We have made several additional observations concerning the practical use of our video-playback architecture.

Since the player's demux-filter can only switch between streams at synch-points (without corrupting the image), it is advised to choose key-frame only compression methods (e.g. Motion-JPEG). Otherwise, the system will be unable to decode frames belonging to a currently inactive video stream on a rapid change of the user's viewing angle until the next synch-point (e.g. an I-frame for an MPEG-encoded stream).

On re-segmenting the combined wide-angle video into smaller tiles we seek to minimize the CPU time "wasted" on decoding invisible parts (if frames are too large) while avoiding an excessive context switching overhead occurring between parallel decoding threads (if frames are too small). We have found that frame sizes between 64x480 and 352x480 pixels are an ideal choice.

Applications

Our initial exploratory field-testing examined performance characteristics of panoramic video recordings under a variety of conditions. The following targeted test environments were chosen that allowed for assessment across a range of lighting, external activity and camera movement conditions. These environments included:

1. An outdoor mall with the camera in a static position in daytime lighting with background structures and moderate human foot traffic, both close-up and at a distance. (*3rd St Promenade in Santa Monica, CA.)*
2. An outdoor ocean pier with the camera in a static position in extremely intense lighting conditions (direct intense late afternoon sunlight with high reflectance off of the ocean) with both long shots of activity on a beach and close-up activity of human foot traffic and amusement park structures on the pier.
3. Inside of an outside facing glass elevator with the camera in a static position and the elevator smoothly rising 15 floors from a low light (street level shielded) position to more intense lighting as the elevator ascended.
4. The camera mounted at the front of a pickup truck bed near the back of the cab (Fig. 9), traveling on a canyon road for 30 minutes at speeds ranging from 0-40 mph under all daylight ranges of lighting (low shaded light to intense direct sun).
5. Same as #4, except at night on a busy well lit street (*Sunset Blvd. In L.A. CA.),* and on a freeway traveling at speeds from 0-60 mph.
6. A University of Southern California Football game within the Los Angeles Coliseum from both static and moving positions in daytime lighting, with extreme close-ups of moving people and massive crowd scenes (40-60 thousand people).
7. An indoor rock concert (*Duran Duran at the Anaheim, CA. House of Blues venue*) from a static position under a variety of extreme lighting conditions in the midst of an active crowd, slightly above average head level with 3D immersive audio recording.
8. A Virtual "Mock-Party" with the camera in a static position in the center of a room with approximately 16 participants. This was a "scripted" scenario designed to address social phobia that was shot while systematically directing and controlling the gradual introduction of participants into the scene and orchestrating their proximity and "pseudo-interaction" with the camera [12].

Fig. 9 – A mobile panoramic video recording setup.

References

[1] S. E. Chen, "QuickTime VR: An Image-Based Approach to Virtual Environment Navigation," Computer Graphics (SIGGRAPH 95), pp 29-38, August 1995.

[2] R. Szeliski, H.-Y. Shum, "Creating Full View Panoramic Mosaics and Environment Maps," Computer Graphics (SIGGRAPH 97), pp 251-258, August 1997.

[3] "Panoramic Viewing System with offset virtual optical centers", US Patent 6111702

[4] Panoram Technologies Inc. www.panoramtech.com

[5] FullView.com Inc. www.fullview.com

[6] D. Kotake, T. Endo, F. Pighin, A. Katayama, H. Tamura, and M. Hirose. "Cybercity Walker 2001: Walking Through and Looking Around a Realistic Cyberspace Reconstructed from the Physical World," Proceedings of ISMR'01, Yokahama, Japan. March, 2001.

[7] S. K. Nayar, "Catadioptric Omnidirectional Camera," Proc. of IEEE Computer Vision and Pattern Recognition (CVPR), June 1997.

[8] Microsoft DirectX SDK msdn.microsoft.com/directx

[9] Pinnacle Systems www.pinnaclesys.com

[10] Adobe Systems Inc. www.adobe.com

[11] Intersense Inc. www.isense.com

[12] Rothbaum, B.O., & Hodges, L.F. "The use of Virtual Reality Exposure in the Treatment of Anxiety Disorders." *Behavior Modification*, 23(4), 507-525 (1999).