Video Enhanced Gigapixel Panoramas

Sören Pirk
University of Konstanz

Michael F. Cohen
Microsoft Research

Oliver Deussen
University of Konstanz

Matt Uyttendaele
Microsoft Research

Johannes Kopf
Microsoft Research

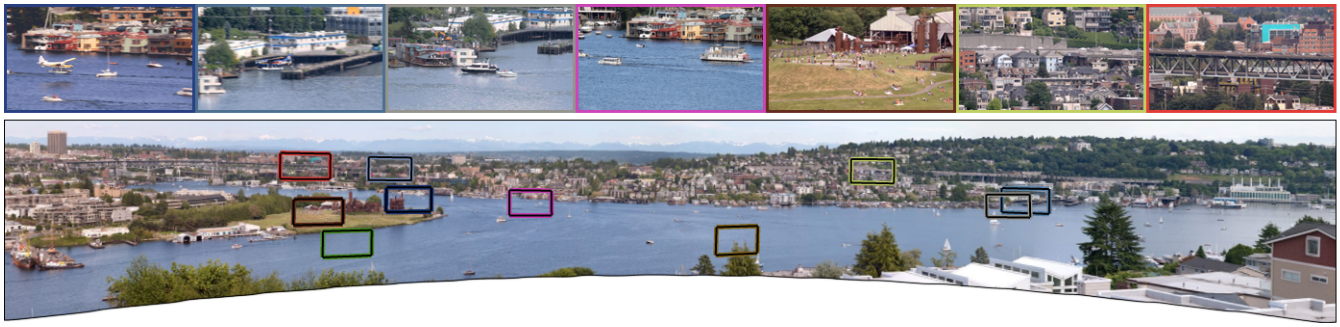

Figure 1: *A gigapixel image with integrated sparse video clips. Combining high-resolution images with embedded video (example frames in the top row) allows users to explore complex scenes in a new way.*

Abstract

We present a method for embedding video clips within gigapixel scale imagery. The combination of high-resolution imagery and video enables users to pan and zoom across the gigapixel panorama to explore complex scenes with motion. The sparsity of the video content within the gigapixel context introduces several challenges which we overcome by optimizing the traversal of the scene coupled with appropriate playback of the embedded video. We also discuss aligning the video clips both geometrically and photometrically to reduce visible seams between the dynamic and static content. Embedding video in large scale panoramas fills a gap between static gigapixel images and video footage and thus presents a new interactive medium.

CR Categories: I.3.8 [Computer Graphics]: Applications;

Keywords: Gigapixel Panorama, Embedded Video, Interactive Exploration

1 Introduction

Our eyes are efficient sensors for exploring very complex scenes. They combine extreme sharpness in the center of our visual attention and a wide opening angle. By scanning a scene, we assemble a high resolution mental image of our surroundings. In much the same way, recent advances in hardware and software make it possible to capture ever increasing resolution (gigapixel) imagery. Automated panning hardware coupled with panoramic stitching software provide simple tools for creating gigapixel sized images captured over a period of minutes. The result itself is static. However, fully exploring the detail in such an image requires dynamically panning and zooming over the result.

An exciting next step would be to create a gigapixel scale video. Such a capture would allow panning/zooming as well as playing through time providing a much closer experience of "being there". Capturing a gigapixel image typically requires multiple shots captured over minutes which are stitched into a single image. This precludes creating a full frame-rate (30fps) gigapixel video. High end cameras let you trade off spatial resolution for temporal resolution, but none are currently capable of capturing gigapixel video. That said, most SLRs, point-and-shoot cameras, and mobile phone cameras can also capture video, albeit at $1/1000^{th}$ the resolution of the full panorama.

Given this constraint, we enhance previously static gigapixel imagery by embedding short video clips of small sub-regions of the full gigapixel panorama. These are captured with the same long lens used to construct the original panorama. Our aim is to integrate these videos into the gigapixel image and to create an experience of a lively animated high-resolution image that captures not only the visual complexity and beauty of a scene but also its dynamism. Having only a sparse subset of the full gigapixel video presents a number of challenges which we address in our work. In particular, we provide user interface affordances for exploring the dynamic elements within the scene while avoiding, as best we can, artifacts introduced by the spatial and temporal edges of the video clips. Spatial edges arise because motion may overlap the frame of the region captured (e.g., cars driving into and out of the frame), and/or due to photometric changes between the time when that area of the panorama was captured and when the video was shot. Temporal edges arise from the fact that only a short timespan is captured for each clip thus the video has a distinct start and stop time.

Our approach to reducing the visual artifacts are two-fold. First we position the video frames, adjust the photometric qualities as well as mask foreground objects to blend the video frames into the gigapixel background. Second, we optimize the pan and zoom path from the current position to view a newly requested video clip. At the same time we control when the video plays. This provides the means to seamlessly view the video clips in context. By combining manual and automated controls over panning and zooming we provide an immersive and dynamic experience.

2 Related Work

Beautiful gigapixel images were first captured by the artist and engineer Graham Flint using a specialized camera with very large custom film back[1]. The uniqueness of the camera precludes normal users from creating such images. The alternatives in the digital

[1] http://gigapxl.org

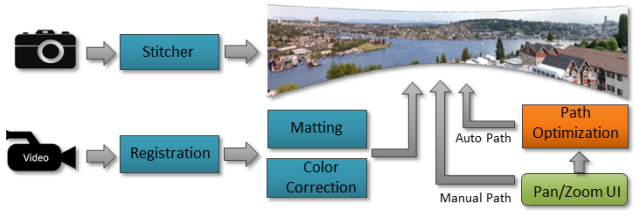

Figure 2: *System Diagram*

world are to use a large array of lower resolution cameras, or a single camera that pans over the scene capturing an array of individual lower resolution images. In both cases, a set of individual images must be *stitched* into a coherent whole. Panoramic stitchers such as Adobe Photoshop[2] and Microsoft Research ICE[3] handle geometric and photometric alignment across the scene. We use the latter in our work.

Kopf *et al.* [2007] describe a pipeline to create gigapixel images using a standard SLR camera in combination with a motorized camera mount. They also discuss dynamically varying the projections of the imagery when viewing very wide angle panoramas and narrow zoomed in views of the same data. Recently, Gigapan[4] has developed inexpensive, easy to use motorized mounts for capturing panoramic imagery which we use in our work.

Quicktime VR [Chen 1995] is a pioneering work in interactive panoramas. In addition to introducing the notion of interacting with panoramas, they demonstrated embedded video loops as well. However, their work did not deal with the issues surrounding very high resolution imagery.

A number of works integrate stochastic video textures into panoramas. Agarwala *et al.* [2005] show how to amend panoramic imagery with video textures such as water waves and blowing leaves to enliven content. Rav-Acha *et al.* [2005] use a sweeping video to create a panoramic video of, for example, a very wide waterfall, by allowing time to vary across the panorama. In contrast, we wish to depict less stochastic action such as people, cars, boats, or airplanes traversing a scene embedded within very high resolution panoramas.

The idea of embedding information in the form of text and audio annotations in a gigapixel image was explored in the work of Luan *et al.* [2008]. In this work, as one pans and zooms about the gigapixel imagery, audio and text annotations are revealed much as we embed video clips within the gigapixel image.

The work of Sargent *et al.* [2010] is the only work we are aware of that truly depicts gigapixel scale video. In their work, due to the constraint of requiring minutes to capture each gigapixel panorama, they focus on gigapixel time lapse video. They demonstrate an impressive system which deals with the challenges of accessing appropriate parts of the gigapixel video as one pans and zooms. In contrast, we wish to display motion that takes place at a normal time scale, thus requiring 30fps capture. Our contributions address the constraints of being able to only populate sparse regions of the gigapixel imagery with video.

3 Overview

Creating the dynamic gigapixel imagery starts with gathering data. Our setup includes one or two Canon SLR cameras with long (100–400mm) lenses. One is mounted on a Gigapan Epic Pro which is able to pan over the scene to collect a sequential series of still images that can be stitched into a gigapixel panorama. Either the second camera (if available) or the same camera (after the gigapan capture) records 1920x1080 pixel videos. For video clip collection, the panning and zooming is under manual control and a user finds interesting dynamic events in the scene and records short video clips to be embedded in the panorama.

Figure 2 show the basic structure of the system. Hundreds of stills are aligned and stitched using the Microsoft Research ICE stitcher, which generates a pyramid of multi-resolution tiles. A viewer similar to Kopf *et al.* [2007] provides the means to interactively pan and zoom over the scene.

The videos are further processed to align them to the panorama, correct for color changes due to exposure and light differences, and optionally create a coarse matte for the foreground object. Having performed these steps the gigapixel image is integrated with video and ready for interactive exploration. We discuss the alignment processes in the next section. We then present two ways of exploring the aligned and embedded videos: In the interactive mode (Section 5.1) the user has full control of the pan/zoom and videos are triggered automatically. We also provide a mode where the user can click on a thumbnail image in a side bar which triggers an automatic transition (Section 5.2). This forms the heart of our contribution.

4 Alignment

In a preprocess we align each image to the gigapixel background. For *geometric* alignment we use a standard feature based approach [Szeliski 2006]. Next, we align the embedded video frame by frame *photometrically* by matching its color histogram to the underlying gigapixel area.

To seamlessly blend the video frames with the background we develop a matte for each frame. For videos with moving objects across the field, we simply feather the outer portions of the rectangle. For videos with a single moving object, we use off-the-shelf software (Adobe After Effects) to track the object of interest in a given sequence and to generate the mattes.

5 Exploring the gigapixel+videos

Given a set of videos that have been aligned with an underlying gigapixel panorama, we are challenged with the task of deciding when to play each video clip. In addition to allowing freeform interactive panning and zooming, we also allow users to easily find video clips by clicking on a representative frame of the video. In this second mode, we optimize a path through the pan and zoom parameters to create a more seamless experience.

5.1 Interactive Exploration

In the first exploratory mode, interactive exploration allows the user full control of the view pan and zoom. The question is how to trigger the embedded videos. We want to avoid artifacts from videos "popping" in or out randomly as much as possible, however this is not always possible. Also, we want to make sure that the user can find the embedded videos, as they might be sparsely distributed across the gigapixel image.

During interactive exploration, videos are embedded in a frozen state with their first frame appearing. When the user navigates sufficiently close to one of the videos it starts to play. To indicate where the videos are we render a colored bounding box when the user is zoomed out. As the view comes closer to the video we fade the bounding box out.

[2] http://www.adobe.com/products/photoshop.html
[3] http://research.microsoft.com/ivm/ice
[4] http://gigapan.com

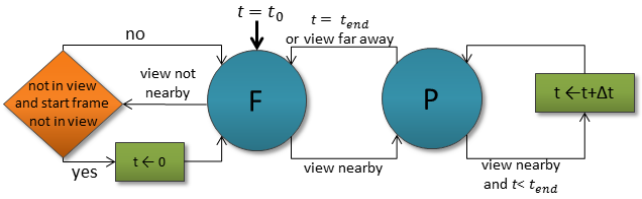

Figure 3: *State machine diagram for the interactive exploration mode. The start state is indicated by a thick arrow.*

The details of the modes is formally described with a state machine diagram (Figure 3). Every video is in one of two states: playing (P) or frozen (F). In addition there is a time parameter t. Videos begin frozen on their first frame, i.e., $t = t_0$. During the course of panning and zooming if a video is determined to be *nearby*, then it begins to play and continues until the end or the view moves sufficiently far away. Time is reset to the start only when the current and first frames of the video are out of view. We have defined *nearby* to mean there is at least some spatial overlap of the current view and video frame, and the field-of-view of the current view is no more than 0.4 radians larger than the video's field-of-view.

5.2 Automatic Transitions

A more convenient way to explore the dynamic aspects of the gigapixel+video panorama is to simply select a thumbnail of a video and have the system automatically transition the pan and zoom to best view that clip. To create a smooth camera path from an initial view position to follow a video, we optimize three curves through time, t, as shown in Figure 5. The first curve is through the two dimensional space of the panorama which we notate as x for the view center, and v for the video's center. The view begins at $x(-2)$, two seconds before the video begins to play, and proceeds to match up with the view at $x(1) = v(1)$, one second after the start of the video. The video's path through space is defined by the alignment process above. In the case of a locked down camera filming a single location in the scene the path is simply a constant.

Optimizing in Space

We define an optimal path to be one that matches the start and end points, matches the velocity of the video's path at $t = 1$, and does not intersect the video's frame at $t = 0$ to avoid *popping*. For smoothness we determine a Bezier cubic curve defined by four control points, $c_0..c_3$. The first, c_0, is located at the start point, $x(-2)$, and the last, c_3, is located to coincide with $v(1)$. c_1 is free to roam, providing two degrees of freedom. c_2 is constrained to lie along the tangent of $v(t)$ at $t = 1$, thus has one degree of freedom. In the case of a still camera we constrain c_2 to lie along a line between c_0 and c_3. To optimize the curve through space we satisfy several high level goals:

1. the path should be smooth and continuous,
2. it should be connected with the video along the preferred direction,
3. while avoiding intersecting the video on the first frame (popping)

Part of these goals are satisfied by the choice of a Bezier cubic and the degrees of freedom. However, practically, for the path to appear smooth we also wish to avoid high curvatures ("bends"). We can express the smoothness using an energy term

$$E^{smooth} = \int_{-2}^{1} \kappa\left(s\left(t\right)\right) \left| \frac{\mathrm{d}s(t)}{\mathrm{d}t} \right| \mathrm{d}t, \qquad (1)$$

where $\kappa(s)$ is defined as the reciprocal radius of the osculating circle at position s.

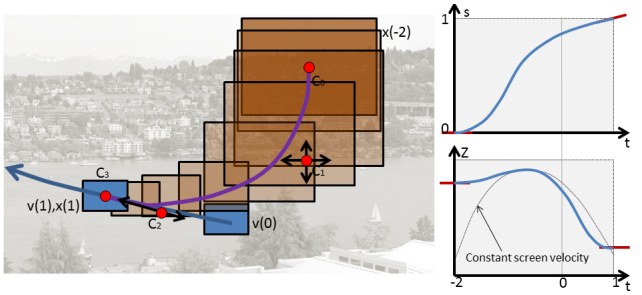

Figure 5: *Optimizing a pan and zoom path from two seconds before video starts (t=-2) to one second after (t=1). Three curves are determined: First, a Bezier cubic path in space (left) that creates a smooth transition from the start position of the view, x(-2), the view position 2 seconds before the video starts, to the position and velocity of the video one second after it begins, x(1) = v(1). Second, the time along the path, top right, a Hermite cubic curve s(t) that matches the start and end positions and velocities indicated in red. Third, bottom right, zoom as a function of time, z(t), that matches position and velocities at start and end, but also tries to keep the visual flow on the screen within bounds as defined by Igarashi et al. [2000].*

We define another energy term

$$E^{popping} = \frac{R(t)}{D(t)^2}, \qquad (2)$$

that measures how well we avoid the popping.

Here, $R(t)$ denotes the ratio of the view at time t that is filled by the video (a value of zero at time 0 means that we avoid popping). Minimizing $R(t)$ alone is not sufficient because it has a local minimum when the view is centered on the video due to the distortion of perspective projection. We thus divide by $D(t)$, the distance of the center of the video to the center of the view at time t. Dividing by the distance to the center adds a strong desire to move the video "out of the way".

We use both error terms for a combined optimization problem,

$$\min_{c_i} E^{smooth} + \lambda E^{popping}, \qquad (3)$$

where $\lambda = 1000$ is a balancing coefficient.

We minimize the energy with the Nelder-Mead method as implemented in the GNU Scientific Library[1]. Since the number of DOFs is only 3 (two for c_1, one for c_2), the algorithm converges rapidly to a local minimum. To better approximate the global minimum, we start the optimization from 100 random initial states and retain the result that yields the lowest energy.

Optimizing speed and zoom along the path

We still have two more aspects of the path to select. The first is how fast to move along the curve, c, that we just computed. A path length parameterization, $s(t)$, is used for the curve. Let l be the length of the curve. A function $s(t)$ maps from time to a location along the curve ($x(t) = c(s(t))$) with $s(-2) = 0$, and $s(1) = l$ according to the above given assumptions.

The tangent at the start point $\frac{ds}{dt}(-2)$ is given by the current speed of the view (i.e. zero if the view is resting, or positive if the user is panning). The tangent at the end point, $\frac{ds}{dt}(1)$, is given by the motion of the video. We define $s(t)$ as a cubic Hermite spline that

[1] http://www.gnu.org/software/gsl

Figure 4: *Two gigapixel images with integrated sparse video clips.* Left: *Zurich Airport;* Right: *Constance Bridge.*

Dataset	GP	#Images	#Clips (Used)	Size (GB)
Lake Union	1,09	324	134 (12)	33,6
Lake Constance	1,98	180	18 (5)	6,2
Zurich Airport	2,53	189	34 (13)	7,8

Table 1: *Summary of the acquired data.*

matches the given start and end values and tangents (see the plot of $s(t)$ in the upper right of Figure 5).

Next, we define a function $z(t)$ that maps from time to a zoom level (see Figure 5 lower right). The simplest mapping would be to derive it analogously to $s(t)$ as a cubic Hermite spline from the start/end values and tangents. However, if the current view is zoomed in and spatially distant from the video this leads to a very high apparent velocity, because the view appears to move rapidly while being zoomed in. This rapid visual flow is avoided by zooming out and then zooming back in. Igarashi *et al.* [2000] describe a method to set the zoom level automatically depending on the speed.

Since we cannot simultaneously satisfy the start and end constraints and the "desired zoom level" as defined by Igarashi *et al.*at all times, we instead add one more constraint, to examine the "desired zoom level" only at the middle of the transition. If the desired zoom level is zoomed further in than the zoom we would get from the naive Hermite spline we are fine because the apparent speed will not be too large. However, if the desired zoom level is zoomed further out than the spline we change our spline by forcing it to interpolate the desired zoom level in the middle of the transition. This additional fifth constraint leads to an analytically defined quartic spline in this case.

6 Results

We have captured and processed three scenes. In each case, multiple videos were captured along with the gigapixel panorama stitched from multiple individual photos. Videos were recorded over a period of a few hours each. Examples of the results can be seen in the accompanying video and static representations are in Figures 1 and 4. Some statistics of the data is shown in Table 1.

Similarly to Kopf *et al.* [2007] we store the Gigapixel images as pyramids and use a multi-threaded architecture to fetch and cache image tiles. They are internally represented as cube maps. We store the embedded videos as jpeg sequences on a RAM disk for fast random access. All results were produced on an Intel Core i7-2600K CPU at 3.40 GHz with 16GB memory and a NVIDIA GTX 580 GPU with 1.5 GB GPU memory. The user interface for the viewer uses standard panning and zooming controls. Imagery is rendered using a perspective projection.

7 Conclusion

We have extended static gigapixel panoramic images by embedding video clips within them to create a more dynamic medium for exploration. We have shown how to align the videos both spatially and photometrically and blend them with the background panorama. Most importantly, we demonstrated two user interfaces for moving between the embedded videos and triggering them to play. A simple interactive mode triggers the videos when the view approaches them. Panning and zooming to the videos in an automated user interface provides an optimized path to each video that avoids artifacts due to the sparse nature of the videos in space and time. We have found the dynamic nature of the results quite exciting to interact with.

There is clearly a lot more that can be done both at the detailed level and for the broader goal of developing this dynamic medium. At the detail level, we need to do a better job with aligning videos especially in areas where the background does not provide sufficient features, such as boats on the water, or airplanes in the sky. We also are working on means to automate turn taking for the display of overlapping videos. At a higher level, we wish to integrate multiple time scales of video. In the photograph of the setup in the accompanying video, there are two additional cameras, one of which took a medium resolution photograph every second. This provides a time lapse over a number of hours. We are exploring ways to integrate this with the real-time videos, and/or use the time lapse to help solve the lower level issue listed above of registering the detailed videos.

Even static gigapixel images provides serendipitous surprises as one pans and zooms around the scene. The addition of dynamic elements truly brings the immersive experience to life.

References

AGARWALA, A., ZHENG, K. C., PAL, C., AGRAWALA, M., COHEN, M., CURLESS, B., SALESIN, D., AND SZELISKI, R. 2005. Panoramic video textures. *Proc SIGGRAPH 2005 24*.

CHEN, S. E. 1995. Quicktime VR: an image-based approach to virtual environment navigation. *Proc SIGGRAPH '95*, 29–38.

KOPF, J., UYTTENDAELE, M., DEUSSEN, O., AND COHEN, M. F. 2007. Capturing and viewing gigapixel images. *Proc SIGGRAPH 2007 26*, 3.

LUAN, Q., DRUCKER, S., KOPF, J., QING XU, Y., AND COHEN, M. 2008. Annotating gigapixel images. *UIST 2008*.

RAV-ACHA, A., PRITCH, Y., LISCHINSKI, D., AND PELEG, S. 2005. Dynamosaics: Video mosaics with non-chronological time. *Proc CVPR '05*, 58–65.

SARGENT, R., BARTLEY, C., DILLE, P., KELLER, J., NOURBAKHSH, I., AND LEGRAND, R. 2010. Timelapse gigapan: Capturing, sharing, and exploring timelapse gigapixel imagery. *Fine International Conference on Gigapixel Imaging for Science*.

SZELISKI, R. 2006. Image alignment and stitching: a tutorial. *Found. Trends. Comput. Graph. Vis. 2*, 1 (Jan.), 1–104.

TAKEO IGARASHI, K. H. 2000. Speed-dependent automatic zooming for browsing large documents. *UIST 2000*, 139–148.