High Resolution Full Spherical Videos

Frank Nielsen
Sony Computer Science Laboratories Inc., Tōkyō, Japan

Abstract

We describe algorithms for authoring and viewing high resolution immersive videos. Given a set of cameras designed to be aligned more or less at the same nodal point, we first present a process for stitching seamlessly synchronized streams of videos into a single immersive video corresponding to the video of the abstract multi-head camera. We describe a general registration technique onto geometric envelopes based on minimizing a novel appropriate objective function, and detail our compounded image synthesis algorithm of multi-head cameras. Several new environment maps with low discrepancy are presented. Finally, we give details on the viewer implementation. Experimental results on both immersive real and synthetic videos are shown.

1. Introduction

In the past years, two main trends of research in computer graphics have been focused on. Namely, on one hand, one has investigated algorithms for handling huge-size high-fidelity 3d scenes that incorporate not only geometry and textures but also physical attributes for realistic renderings, and on the other hand one has developed techniques for manipulating images for creating virtual environments allowing stunning photorealistic walkthroughs but lacking of interactivities. Both directions of research are closely used today jointly in the latest movie productions using computer vision and graphics.

1.1. Previous Works

Recently, there has been a strong industrial push into the mass production of cheap graphics hardware that can render and shade hundreds of millions of polygons in real-time. However, even if the graphics pipeline power has been unleashed to some extent, one of the challenges remains certainly into modeling (often by a laborious manual process) those complex scenes. Although 3d scanning technologies have been considerably improved to obtain fine detailed realistic models, one of the core problem remains the scene object composition. A key feature of this "geometric approach" is that the processing cost depends only on the intrinsic complexity of the geometric entities describing the scene. Therefore efficient algorithms for visibility, culling strategies and progressive model refinements are of primal importance. Another successful approach to navigate into virtual environments is to manipulate "real data" captured from our surrounding world. That stream of research is often coined as *image-based rendering* or *warping*. A very early example is the movie map developed by Lippman [5] where ≥ 54000 shots of the city of Aspen were stored on an analog video-disc. A simple player retrieved the corresponding location image by doing nearest image query from a database. The widespread use of photorealistic virtual environments was certainly introduced by Chen [4] and his team in the Apple Quicktime VR® format. Both panorama movie and object movie formats were introduced. A key difference with previous branching-movie works, was that the 360-degree panorama movie was accomplished not by a set of pictures taken at discrete rotations but stored into a single seamless "panoramic image" (*ie.*, environment map). The process of creating that panoramic image is today often called "cylindrical stitching". Recently, Aliaga and Carlbom [6] presented a promising scalable image-based approach for smooth interactive walkthrough based on a manually robot-driven omniview capturing system.

It is worth observing that even on the research line of fast 3d model-based walkthroughs, there is a trend of using image/video impostors (computed offline) to make the online rendering at interactive rates [12].

1.2. Main Contributions

In this paper, we propose several novel processes to stitch, store and view immersive videos. Namely,

- We introduce in Section 3 the notion of geometric envelopes as a support for the registration process and environment maps allowing tailored registrations.
- We propose in Section 4.2 a registration technique by minimizing a suitable objective function onto a given

Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'02)
0-7695-1506-1/02 $17.00 © 2002 IEEE

environment map for computing the "less visually" defective maps.

- We describe in Section 4.5 an image blending technique where rays (and not as usual pixels) are interpolated and blended according to their respective camera attributes. This scheme is used both for generating output video and inside the registration process where non integral pixel values are required.

- We introduce in Section 5 several full view environment maps as alternative to the equi-rectangular, cube or dual paraboloid maps. Namely, the stratified random, compressed latitude-longitude and Hammersley/Halton maps which have proven low discrepancy (*ie.*, good sampling in any direction).

2. Imaging Sensing Device

Imaging full view video can be done in several ways depending on how many cameras and lens combinations we use, or whether mirrors are used or not. A common approach is to use catadioptric systems [14] (*ie.,* using mirrors) to acquire a large vertical field of view over a complete 360-degree horizontal field of view. Although the monoblock recording system (only one CCD used so that even real-time system can be easily implemented) makes it easy to record video, we notice several undesirable drawbacks such as blurring, non-uniform image density and partial view acquisition (that is not full 4π steradian; This last point can be overcome by using two back-to-back such devices). Note that theoretically a single lens can capture close to 4π angular view but in practice the image quality degrades significantly.

Multi-head camera[1] design is about combining a set of cameras, lens and eventually mirrors so that the imaging units obtained from their respective individual cameras share as close as possible a same nodal point. On one hand, using several CCD devices allow us to use different exposure settings to capture well-balanced sceneries (like sunsets, live concerts, etc.) and control the density of the images. But on the other hand, it makes the recording unit more cumbersome. Also since it is desirable to minimize the number of camera units, we are using lens with inherent large distortions and try to minimize the overlapping areas of the video streams. This means that the stitching process becomes far more delicate than the still imaging case. Since usually the focal plane is in front of the CCD this means that using standard cameras for imaging the full sphere will result in parallax errors that can be unnoticed during the shooting if objects are past a given threshold distance that we call *parallax clearance*. However, no-parallax full view multi-head cameras can be designed using systems of mirrors to simulate a unique virtual nodal point (See [3]). Also even if parallax is noticeable for a given omniview system, one can apply deghosting techniques [9] to compensate and minimize its effect.

[1] *Dodeca* of Immersive Media (1989) is one of the first prototypes. Swaminathan and Nayar [17] called them *polycameras* (1996).

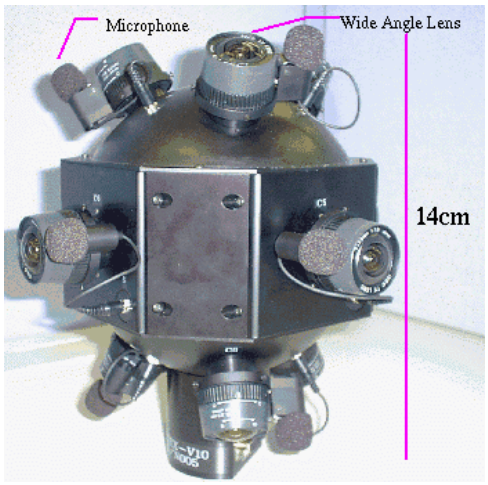

Figure 1. Our full spherical multi-head camera: Ten CCD cameras with mounted wide-angle lens.

Because multi-head cameras allow flexibility in controlling the resolution of the imaging and its nice exposure capabilities, we selected this approach (See also [15, 18]).

3. Envelopes

3.1. Definition

Informally speaking, an *envelope* $\mathcal{E}$ is defined by an origin O and a surface $\mathcal{S}$ so that any ray emanating from O intersects the surface $\mathcal{S}$ of $\mathcal{E}$ in at most one point. The locus of the points K such that $(K, \mathcal{S})$ defines an envelope is called the *kernel* (See [7], page 18). Spheres, cylinders, cubes, paraboloids, star-shaped polyhedra are examples of envelopes. (A torus is not an envelope.) Let $\rho(\theta, \phi)$ be the radial function ($(\theta, \phi) \in [-\pi, \pi[\times[-\frac{\pi}{2}, \frac{\pi}{2}[$), defined everywhere or partially, centered at O describing the 3d envelope $\mathcal{E}$. For a given point $(X \quad Y \quad Z)^T$ on $\mathcal{E}$, we can associate its spherical coordinates (ρ, θ, ϕ) as (see Figure 5):

$$\theta = \arctan\frac{X}{Z}, \phi = \arctan\frac{Y}{\sqrt{X^2+Z^2}}, \rho = \sqrt{X^2+Y^2+Z^2}.$$

3.2. Environment mappings

An environment map is a function $m(\cdot,\cdot)$ that for any given "pixel" coordinates (x,y) (indexing a ray) of a picture returns its spherical angular $(\theta,\phi) = m(x,y)$ attributes. We call such a picture a *raymap*. Conventional raymaps are: equi-rectangular (*ie.,* latitude-longitude), cubical, cylindrical and dual paraboloidal maps (See Figure 4). We detail density properties of those conventional environments as well as our new proposed ones in Section 5.

3.3. Relationship with Multi-Head Cameras

Images acquired by a set of ideal pinhole cameras sharing the same optical center O relate to each other by homographies, linear applications in the projective space. Since any pixel (x,y) of an image taken by a pinhole camera can be transformed into a ray $\vec{r}$, stitching can be seen as a process that registers those bundle of rays by pure rotation transformations. We can choose any envelope $\mathcal{E}$ to convert a ray $\vec{r}$ into its 3d Euclidean coordinates $\vec{r} \cap \mathcal{S}$. (Often the unit sphere is chosen. See Figure 2 and 5.) Actually, if we knew for each ray the distance where it first hit an object, we would have enough information to generate correctly any view from inside the visibility cell of O (included in the kernel [1, 7] of $\mathcal{E}$). This property is used both on synthetic computer graphics images and range camera sources.

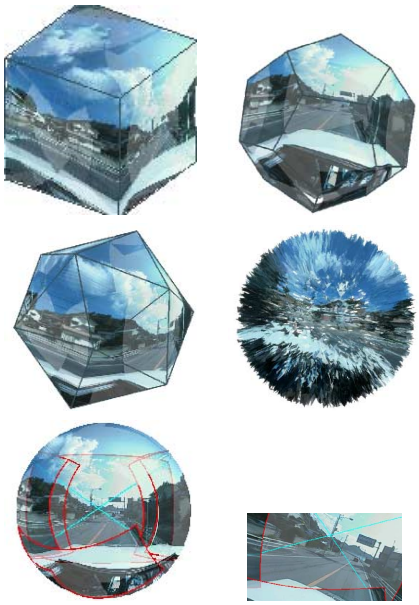

Figure 2. Snapshots from the GUI. Several envelopes: cube, dodecahedron, icosahedron and random triangle soup. Bottom left: Outside the spherical envelope. We can select image(s) and slide them onto the envelope. Bottom right: Envelope viewed from the origin O: they all view the same.

4. Multi-Head Camera

4.1. Notations

Let $\mathcal{C} = \{C_1, ..., C_n\}$ be a set of n cameras (see Figure 1 for the 10-head camera). Each camera C_i is the assembly of a lens L_i and a pinhole sensing device S_i. (All cameras are hardware genlocked). In our settings, we choose $n \in \{2, 8, 9, \mathbf{10}\}$ cameras and each frame buffer size is an NTSC interlaced signal. The frame buffer are capturing the rays of light passing through the lens by means of a CCD. Let $\vec{r_i}(x,y)$ be the geometric ray captured by a pixel of coordinates (x,y) of camera C_i. If the focal and iris settings are set up to remained fixed during the shooting, the geometry of the ray does not change over time but only its incoming luminance and chrominances. Classically in computer vision, the relationship between $\vec{r}$ and $(x_i(\vec{r}), y_i(\vec{r}))$ has been modeled for a pinhole camera as follows:

$$(x_i(\vec{r}), y_i(\vec{r})) = RayToImage(\vec{r}, C_i) = L_i(\tilde{x}_i(\vec{r}), \tilde{y}_i(\vec{r})),$$

where $(\tilde{x}_i(\vec{r})\ \tilde{y}_i(\vec{r})) \sim \mathbf{K}_i\mathbf{R}_i\vec{r}$, with $\mathbf{K}_i$ the camera intrinsic parameters of C_i and $\mathbf{R}_i$ the rotation matrix. We use homogeneous coordinates so that the operator $\sim$ means $(x\ y) = (x'/w\ y'/w) \sim (x' \quad y' \quad w)$. $\mathbf{K}_i$ is usually handled as the 3x3 matrix:

$$\mathbf{K}_i = \begin{pmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix},$$

where (p_x, p_y) is the principal point, f_x and $f_y = a_y f_x$ are the x-focal and y-focal lengths respectively (can be obtained from horizontal and vertical fields of view) and s is the skewness parameter (we set it to 0). $L_i(\cdot)$ is a bijective function used for going from the ideal pinhole to the original image. In other words, we work in the $\tilde{x}$ space of ideal images and we look up corresponding pixels in the original image via a lens distortion function L_i. Note that we can initialize (p_x, p_y) to $L^{-1}(\frac{w}{2}, \frac{h}{2})$, where w and h are the width and height respectively of the image. A ray $\vec{r}$ can be coded internally in different ways set purposely according to its context. For example, conventional representations include: an anchored point O and either (i) a unit vector $\vec{v}$, or (ii) two angles (θ,ϕ) coding for the unit direction of the ray emanating from O:$\vec{r} = \begin{pmatrix} \sin\theta\cos\phi \\ \sin\phi \\ \cos\theta\cos\phi \end{pmatrix}$, or (iii) a

unit quaternion $(\ q_1 \quad q_2 \quad q_3 \quad q_4\ )$. Quaternions are well-suited for optimization while one can easily understand/edit interactively Euler angles.

4.2. Objective Function

Registration is the process of finding rotational parameters (roll, pitch and yaw angular attributes) of each camera C_i so that all rays share a common[2] nodal point O and that their bundles of rays match. We need to find n unit orthogonal matrices $\mathbf{R}_i$ so that for all directions $\vec{r}$, the luminance/chromacies of the rays match well. Our objective function differs from previous approaches [9, 11, 10] with respect to two aspects:

First, the 'score atom' of the matching of ray $\vec{r}$ is defined according to density functions w_i's (important for wide angle lens, fisheye lens, etc.). Denote by $l(\vec{r})$ the luminance attribute of $\vec{r}$. We define:

$$s_{\vec{r}} = \sum_{i=1}^{n} |w_i(\vec{r}) l_i(\vec{r}) - \frac{1}{\sum_{j=1}^{n} w_j(\vec{r})} \sum_{j=1}^{n} w_j(\vec{r}) l_j(\vec{r})|^2.$$

The $w_i(\cdot)$'s are related to the density of the mapping $L_i(\cdot)$ from the ideal to original image (see Figure 3, third picture). More precisely, we choose:

$$w_i(\vec{r}) = \frac{\Gamma_i(\vec{r})}{\frac{\partial^2 L_i(x(\vec{r}), y(\vec{r}))}{\partial x \partial y}}.$$

where $\Gamma_i(\vec{r})$ is defined as the solid angle spanned[3] by the pixel $(\tilde{x}_i(\vec{r}), \tilde{y}_i(\vec{r}))$ of ideal image i. That is the bigger the derivative of $L_i(\cdot)$, the less precision from the original images we have in the ideal image. (Instead of taking the $\sum$ in $s_{\vec{r}}$ we can also use the $\max$ operator.)

Second, since we store the result of the registration into a ray (environment) map $\mathcal{R}$, we want to obtain the less "visually" defective raymap (when displayed in the viewer). Therefore, we ask for minimizing:

$$\sum_{\vec{r_{i,j}} = m(i,j) | (i,j) \in \mathcal{R}} \Omega_{i,j} s_{\vec{r_{i,j}}},$$

where $\Omega_{i,j}$ is the solid angle subtended by the pixel with coordinates (i, j) of the raymap $\mathcal{R}$. For example using traditional cubic map, it is known that $\frac{\max_{i,j} \Omega_{i,j}}{\min_{i,j} \Omega_{i,j}} = 3\sqrt{3}$ (for dual paraboloid it is 4 and for equi-rectangular map it is infinite[4]). The bigger the solid angle, the most important we have to pay attention about finely optimizing this part as its display size is large. This amounts to give penalty of mismatches according to the local resolution of the environment map.

4.3. Camera Parameters

We used a simplified Tsai's model of camera [13] to modelize only radial distortions. That is, we considered the center of the lens (c_x, c_y), the aspect ratio a_y and two radial distortion coefficients κ_1 and κ_2 so far[5] can be used. The distortions of the lens mounted on the cameras are computed individually using a 2d per-pixel registration of a known observed pattern (See Figure 3, first and second pictures). Our calibration technique differs from usual ones that are said "stellar" because we do not use fiducials (see [17]). Focals can also be initially estimated using Tsai's calibration technique. They are then refined independently by a dense correlation technique described in the local optimization loop below.

We also add extra parameters to each camera like a region of interest (because of misalignments of CCD and lens that yield to black bands on the border of the image), color gamma coefficient, radial intensity drops (radiometric corrections), etc.

4.4. Local Numerical Optimization

Once all parameters have been properly initialized either by calibration, bundle adjustment methods on manually selected feature pairs, or by user manual input (using the GUI), we perform local numerical optimization to approach the global optimum. (Unfortunately we can still fall at a local optimum.) Basically, it requires to compute gradients, Hessians and Jacobians. Those are obtained from derivatives of the parameters according to the objective function. Therefore a plethory of schemes have been proposed so far. Starting from the early work of McMillan and Bishop [8] and Szeleski and Shum [9] (see also [11]) to the more recent work of Corg and Teller [10] which use quaternions for handling rotations in the registration process. Key differences with those methods are that (i) each camera has possibly different parameters and (ii) those parameters may change with time (eg., focus, computer-controlled, etc.) and most importantly (iii) our atom score is weighted according both to the densities for the ideal to original mappings and the density of the raymap. This means that the result of a registration may be different depending on whether we register on a cylinder, cube, sphere, etc. We do not take into account

[2] This is not true in absolute because of camera misalignments and complex optical lens systems made of several unit elements organized into groups. Also because of the dual wave/ray nature of light, diffraction occurs at the boundary of the pinhole and the light spread all over the image, etc.

[3] We consider pixels as surface elements. Solid angles can be approximated by $A \cos\alpha$ where A is the surface area spanned by a "pixel" and α is the incidence angle from O to the center of A.

[4] Oversampling at the poles.

[5] Alternatively, one can use more complex lens distortion models like decentering distortions [17], snakes, per-patch/per-pixel, etc.

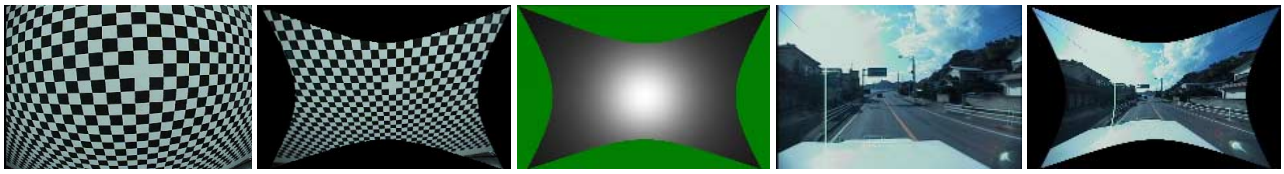

Figure 3. Lens parameters of camera C_5: center=$(360.8, 224.6)$, aspect ratio 1.083, radial coefficients $(2.29 \times 10^{-6}, 1.65 \times 10^{-11})$. The third image illustrates the ideal to original $L_5(\cdot)$ mapping density function. Observe the radial correction effect on the density of the ideal pinhole camera. The brighter, the more precise.

radiometric correction at this level (for example, Turkowski and Xiong [11] considered linear intensity correction) because we found that this can be done offline easily using imaging tools.

We locally optimize our objective function using the Levenberg-Marquadt process since its use of second derivatives accelerate the convergence rate. The gradient $\mathbf{G}$ is computed by summing over the discretized raymap as follows:

$$\mathbf{G} = - \sum_{(\theta,\phi)=m(x,y)|(x,y)\in\mathcal{R}} s_{\theta,\phi} \frac{\partial s_{\theta,\phi}}{\partial \mathbf{P}},$$

where $\mathbf{P}$ are the parameter variables put in a vector.

The Hessian $\mathbf{H} = (\mathbf{H}_{i,j})_{i,j}$ is:

$$\mathbf{H}_{i,j} = - \sum_{(\theta,\phi)=m(x,y)|(x,y)\in\mathcal{R}} \frac{\partial^2 s_{\theta,\phi}}{\partial \mathbf{P}_i \partial \mathbf{P}_j}$$

The optimization loop starts from an estimate $\mathbf{P}$ of the parameters and update the solution as:

$$\Delta\mathbf{P} = -(\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{G}.$$

We set λ (stabilization parameter) initially as $1.0E-4$ and update it according to the variations of $\Delta\mathbf{P}$ (we keep in memory the parameters $\mathbf{P}$ that have yielded to the best result so far). Since we need to compute $(\mathbf{H}+\lambda\mathbf{I})^{-1}$, we may have unstable numerical matrix inverts. In that case, we use singular value decompositions[6] or pseudo-inverse matrices. Figure 4 shows our obtained raymaps. Once registered, we compute the raymap $\mathcal{R}$ by blending the pixels according to their weight function $w_i(\cdot)$'s. During the registration process, pixel intensity/color channel value is computed at non integer positions using a ray interpolation process described below.

[6]The invert of a SVD is easy to get by transpose and diagonal matrix invert operations.

4.5. Ray Interpolation

Often, we use pixel interpolation for computing the luminance/chrominances at non-integer coordinate values $P = (x, y)$. Usual schemes are nearest neighbor, bilinear, bicubic, Catmull-Rom spline and sinc interpolations to name just a few of them. In our framework, each pixel is handled as a ray. Therefore *ray interpolation*, which differs from pixel interpolation is presented below. In order to illustrate our idea, we first discard lens distortions and concentrate on ideal pinhole bilinear ray interpolation of a camera with focal f. Let $r_x = x - \lfloor x \rfloor$. If we used linear pixel interpolation, we can write P_x as $P_x = a_0 P_{\lfloor x \rfloor} + a_1 P_{\lceil x \rceil}$ where $a_0 = 1 - r_x$ and $a_1 = r_x$. Looking it as a ray, we get $P_x = \alpha_0 P_0 + \alpha_1 P_1$, where $\alpha_0 = 1 - \alpha_x$, $\alpha_1 = \alpha_x$, and $\alpha_x = \frac{\arctan\frac{x}{f} - \arctan\frac{\lfloor x \rfloor}{f}}{\arctan\frac{\lceil x \rceil}{f} - \arctan\frac{\lfloor x \rfloor}{f}}$. Therefore linearly interpolating the rays does not result in linearly interpolating the pixels! Note that when the focal tends to infinity, ray interpolation converges to pixel interpolation which is the common interpolation scheme correct for orthographic camera models.

When lens distortions are to be taken into account, we ask for the ideal ray $\vec{r}$. Using lens parameters this amounts to find the ray $\vec{r'}$ stored at (x', y') in the original image (containing distortions). We select a neighborhood of (x', y') and using $L_i^{-1}(\cdot)$, we find their respective coordinates in the ideal image. Finally we can use our ray interpolation onto those point coordinates. There is a lot of literature on interpolation of functions. Particularly relevant to the case of the sphere, is the use of natural neighbor interpolation [2] or quaternion interpolation.

5. Environment maps

Raymaps (say of size $W \times H$) are the texture attribute of the envelopes. Apart from the widely used spherical, cubic, dual parabolic ones, we present alternative ones below. (Those ones are well suited to the case of immersive videos were bandwidth is the limiting factor.)

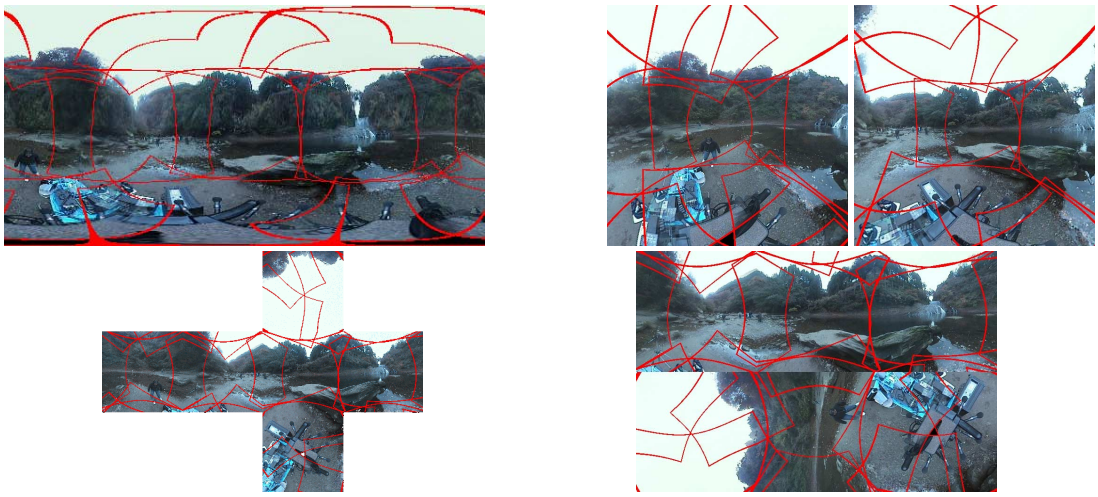

Figure 4. Result of the optimization for conventional environment maps: Equi-rectangular, front and back dual paraboloid, cubic and rearranged cubic (packed into a rectangle) maps. Borders delimiting the respective fields of view of the camera units of Figure 1 are shown in red.

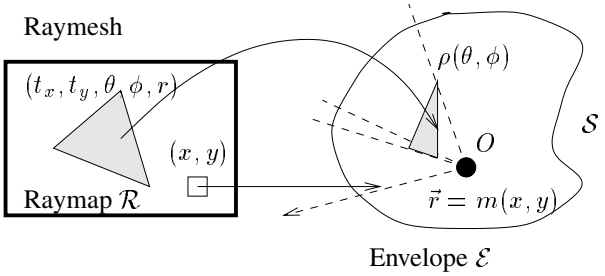

Figure 5. Raymesh: the envelope $\mathcal{E}$ and its corresponding raymap $\mathcal{R}$.

Raymesh. The raymesh format is a 3d mesh envelope where each vertex has 5 components: (ρ, θ, ϕ) its spherical coordinates and (t_x, t_y) the corresponding texture coordinates inside the raymap. Technical and practical considerations (time-varying meshes, etc.) when implementing efficiently those raymeshes are left in [1]. To obtain a simple sampling of the sphere, we can use a icosahedron or a surface refinement of it, etc. For example, B. Fuller's map (also called dymaxion) is a special unfolding of an icosahedron onto a raymap and can be considered as a raymesh (See Figure 7, bottom right). Note that seeking for point sets on the sphere uniformly triangulated in the sense that the ratio of a maximal length edge over a minimal length edge is minimal, is a challenging problem in itself (related to spherical codes).

Compressed spherical. One drawback of the equirectangular map is the non-uniformity of the sampling. Indeed at the equator, we have for each pixel an increment of $\Delta\theta = \frac{2\pi}{W}$. But at height h, it becomes $\Delta\theta = \frac{2\pi\sqrt{1-(\frac{2h-H}{H})^2}}{W}$. We can overcome partially this over amount of information at the poles by sampling proportionally the latitude circles. Let $\phi_h = \pi\frac{h}{H}$, then the width at row h is $w(h) = W\sin(\phi_h)$. We then sample proportionally at row h by $\Delta\theta_h = \frac{2\pi}{w(h)}$ increments. Doing so we can *compress* the standard equirectangular map without loosing quality. We save a factor[7] of $1 - \frac{2}{\pi} \simeq 0.36$. Also we keep the advantage that pixels are stored in a lexicographic order so that ray interpolation and texture can be used without any indexing problem (See Figure 7, top right). In cartography, a similar notion is the *sinusoidal equal area map*. However, a key difference with full view video formats is that we do not require the "maps" to be readable by human eyes but rather seek for efficient encoding/decoding computer processes.

Stratified random. Yet another property of the sphere is that any z-slice of a given width has the same area. This is often used in random distribution of point set on the sphere. We fix the row of the raymap to be between -1 and 1 and let $\phi = \arcsin\frac{2h-H}{H}$. Then we draw at random for each row W θ-positions (we jittered a bit the ϕ so that it remains in its ϕ-slice) that we sort and store (See Figure 7, bottom left).

[7] In practice this means that we obtain better image quality from the video at a given image size for a constant bit rate.

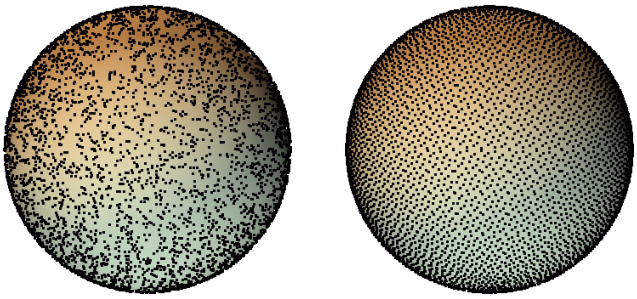

Figure 6. Left: 10000 points distributed uniformly using pseudo-random generators. Right: first 10000 points of the Hammersley sequence.

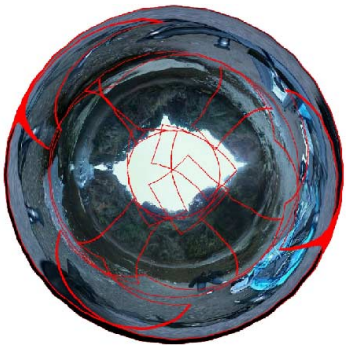

Figure 8. A 360-degree fisheye view created from a raymap.

Hammersley sequence. This is a deterministic sequence of points distributed on the sphere with low discrepancy [19]. It has been used before in computer graphics for Monte-Carlo algorithms. We use the Hammersley sequence with basis $p = 2$ (*ie.*, Van Der Corput sequence). One inconvenient is that we do not have an easy 2d lexicographically indexing order of the rays. However, there are several possibilities in implementing those point sequence map efficiently in retrieving image from a virtual camera. One of them is to consider a 2-pass algorithm: First we project all the point indices onto the virtual camera image plane (forward mapping of the indices). Then we do conventional backward mapping by retrieving the indices of the neighboring points obtained from the first step (See Figure 7, top left).

6. Viewer

There are several ways of visualizing the envelope: from inside to outside. For each in/out mode, we can either draw per-pixel or per-texture element. Because of widely available hardware capabilities, we chose to render using 2d textured triangles. For each triangle $t = \{p_1, p_2, p_3\}$ with p_i associated to (θ_i, ϕ_i) of the raymesh, we compute the corresponding position of the corresponding triangle in the xy-screen as $x_i \sim \mathbf{KR}\vec{r}_{\theta_i,\phi_i}$. Using matrix formulations, we do the intersection of a unit sphere with a line (instead of a ray), so we need to remove the ambiguity by checking whether $\vec{r}_{\theta_i,\phi_i} = \vec{r'}$ or not, where $\vec{r'} \sim (\mathbf{KR})^{-1} x_i$.

Also, we can choose to view the map not through an ideal perspective camera but from any kind of camera (or ray device-like cameras: wide angle lens or fish-eye lens). We simply need to define a $RayToImage(\theta, \phi, C)$ function that given a camera model C and spherical coordinates (θ, ϕ), returns the position, if it exists, on the xy-screen. For example, $RayToImage(\theta, \phi, C_F)$ can be written as $(x, y) = r(\phi)(\cos\theta, \sin\theta)$, where $r(\phi) = f_1\phi + f_2\phi^2 + \ldots$ for a fisheye camera C_F using the equi-distance projection model (See Figure 8). Also if the exact (or coarse) geometry of the envelope is given, we allow the user to move inside its visibility cell and produce the correct image [1].

7. Experiments

Our camera prototypes are mobile for indoor/outdoor shootings. We can control the camera CCDs individually through an IEEE 1394 link interface. We record each camera video onto DV tapes for offline processing. Finally we implemented our viewer onto a conventional PC and playback MPEG-2 streams (15 Mpbs CBR) at 30 fps from DVD.

8. Summary and Perspectives

In this paper, we described a system for stitching, storing and viewing immersive videos onto envelopes. When the geometry of the surrounding environment is available this allow ones to move freely inside its visibility cell enhancing the realism of the interaction. Immersive video is pushing traditional videos one step further towards spatial-temporal media which combines 3d audio and video, and give users freedom and a unique experience to interact with the contents.

References

[1] F. Nielsen, "Immersive Video Envelopes," 50 p., Sony CSL, *Technical Report*, 2001.

[2] D. Watson, "modemap: An implementation of natural neighbor interpolation on the sphere," xii, 171 p, 1998.

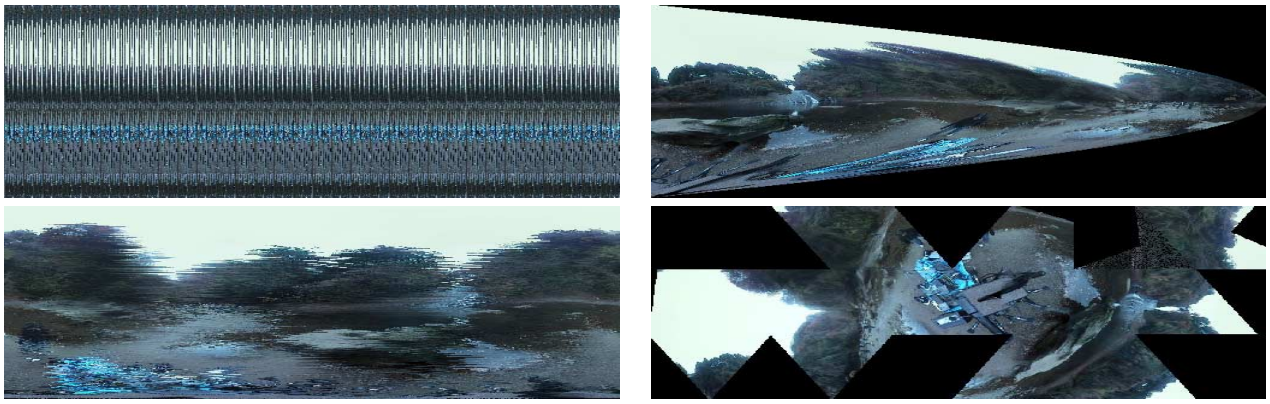

Figure 7. Top left: Hammersley map. Top Right: Compressed spherical map. Bottom left: Stratified random. Bottom right: B. Fuller's map

[3] V. S. Nalwa. "A true omnidirectional viewer," *Technical report*, Bell Laboratories, 1996.

[4] S. E. Chen, "Quicktime VR - An Image-Based Approach to Virtual Environment Navigation," *Computer Graphics SIGGRAPH*, pp. 29-38, 1995.

[5] A. Lippman, "Movie Maps: An Application of the Optical Videodisc to Computer Graphics," *Computer Graphics SIGGRAPH*, pp. 32-43, 1980.

[6] D. G. Aliaga and I. Carlbom, "Plenoptic Stitching: A Scalable Method for Reconstructing 3D Interactive Walkthroughs," *Computer Graphics SIGGRAPH*, pp. 443-450, 2001.

[7] F. P. Preprata and M. I. Shamos, "Computational Geometry: An Introduction," *Springer-Verlag*, 1988.

[8] L. McMillan and G. Bishop, "Plenoptic Modeling: An Image-Based Rendering System," *Computer Graphics SIGGRAPH*, pp. 39-46, 1995.

[9] R. Szeliski and H.-Y. Shum, "Creating Full View Panoramic Image Mosaics and Environment Maps," *Computer Graphics SIGGRAPH*, pp. 251-258, 1997.

[10] S. Coorg and S. Teller, "Spherical Mosaics with Quaternions and Dense Correlation," *International Journal of Computer Vision*, Vol. 37(3), pp. 259-273, 2000.

[11] Y. Xiong and K. Turkowski, "Registration, Calibration and Blending in Creating High Quality Panoramas," *IEEE Workshop on Algorithms of Computer Vision*, pp. 69-74, 1998.

[12] A. Wilson, M. C. Lin, D. Manocha, B.-L.Yeo and M. Yeung, "A Video-Based Rendering Acceleration Algorithm for Interactive Walkthroughs," *ACM Multimedia*, 2000.

[13] R. Y. Tsai, "A Versatile Camera Calibration Technique for High Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses," *IEEE Journal of Robotics and Automation*, Vol. RA-3 (4), 1987.

[14] S. Baker and S. Nayar, "Single Viewpoint Catadioptric Cameras," *Panoramic Imaging: Sensors, Theory, Applications,* Springer Verlag, 2000.

[15] A. Majumder, G. Meenakshisundaram, W. B. Seales and H. Fuchs, "Immersive Teleconferencing: A New Algorithm to Generate Seamless Panoramic Video Imagery," *ACM Multimedia 99*, pp. 169-178, 1999.

[16] S. K. Nayar and A. Karmarkar, "360x360 Mosaics," *Proc. IEEE Computer Vision and Pattern Recognition*, 2000.

[17] R. Swaminathan and S. K. Nayar, "Non-Metric Calibration of Wide-Angle Lenses and Polycameras," *Proc. IEEE Computer Vision and Pattern Recognition*, Vol. 2, pp. 413-419, 1999.

[18] "Panoramic Vision: Sensors, Theory and Applications," *Springer Verlag*, 2001.

[19] T.-T. Wong, W.-S. Luk and P.-A. Heng, "Sampling with Hammersley and Halton Points," *Journal of Graphics Tools*, Vol. 2(2), pp. 9-24, 1997.