On Representing Spherical Videos

Frank Nielsen
Sony Computer Science Laboratories Inc., T$\bar{o}$ky$\bar{o}$

Abstract

Given a set of video camera units sharing the same optical center, one can stitch their images seamlessly into a single full view video coding for the abstract multi-head camera. Traditionally, equirectangular, cubic and dual paraboloidal environment maps have been used for storing panorama pictures. We present several new formats well-suited to immersive video that improve image quality and discuss briefly on our viewer implementation.

1. Introduction

Imaging full view video can be done in several ways depending on whether mirrors are used or not (so-called catadioptric systems), and on how many cameras and lens types are used [1, 2]. Multi-head camera[1] design is about combining a set of cameras, lens and eventually mirrors so that the imaging units obtained from the individual cameras share as close as possible a same nodal point. Figure 1 shows our 10-head prototype. One key difference with full view static mosaicing (eg., Apple Quicktime cubic VR®), is that the media (digital versatile disc or streaming through a broadband network, etc.) and memory bandwidths (PCI bus, graphics processor unit bandwidth, etc.) are limiting factors for playing full view video at 30p/60i frames per second. Therefore data need to be efficiently represented and compressed.

2. Environment mappings

An environment map $\mathcal{R}$ is a mapping function $m(\cdot,\cdot)$ so that for any given "pixel" coordinates (x, y) of a picture, it returns its $(\theta, \phi) = m(x, y)$ spherical coordinates. (Since pixels code for rays, we may call those elements *rayels*.) We can either interpret pixels as 0-dimensional elements: *rays* or as 2-dimensional objects: *solid angles*. The basic idea of environment mapping [4, 5] is that if a reflecting object is small compared to its distance from the surrounding world (modelled as a backdrop) the incoming illumination on its surface depends only on the reflected ray. Conventional maps are: spherical (orthographic

[1] *Dodeca* of Immersive Media (1989) is one of the very first prototypes. Swaminathan and Nayar [3] called them *polycameras* (1999).

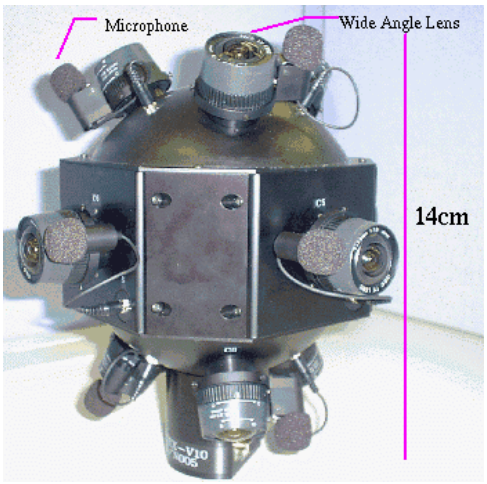

Figure 1: Multi-head camera: Ten CCD cameras with mounted wide-angle lens providing a complete field of view.

view of a centered mirroring ball, see OpenGL® specifications), latitude-longitude, cubical, cylindrical and dual paraboloidal maps. They all store the surrounding space of directions but their sampling properties are different. Indeed, using traditional cubic map, it is known that the difference of sampling is bounded by $3\sqrt{3}$. Let $\Omega_{x,y}$ be the solid angle subtended by the pixel with coordinates (x, y) of the map $\mathcal{R}$. Then $\frac{\max_{i,j}\Omega_{i,j}}{\min_{i,j}\Omega_{i,j}} = 3\sqrt{3}, 4, \infty$ for cubic, dual paraboloid and for latitude-longitude[2] maps respectively. Environment maps and immersive videos both need to discretize and store the space of directions efficiently. (Note that environment maps are often constrained by the fact that they need to be hardware accelerated [6].)

3. Alternative environment maps

Environment maps (say of size $W \times H$) are the texture attributes of a unit sphere. Besides the widely used latitude-longitude, cubic, dual parabolic maps, we present alternative new ones below and detail their properties.

[2] Due to degenerated oversampling at the poles.

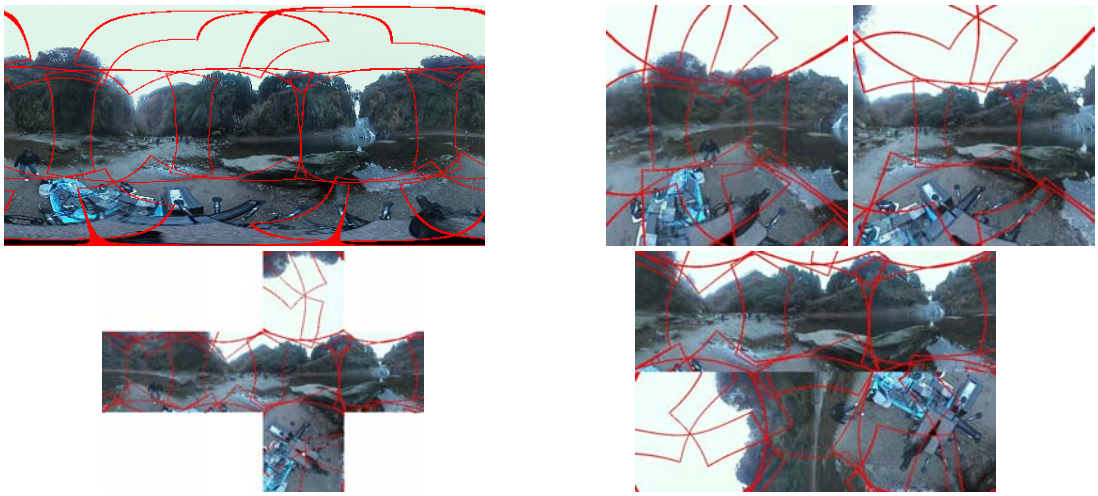

Figure 2: Conventional environment maps: Equirectangular, front and back dual paraboloid, cubic and rearranged cubic (packed into a rectangle) maps . Borders delimiting the respective fields of view of the camera units of Figure 1 are shown in red.

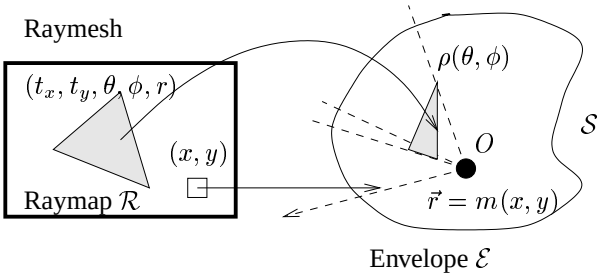

Figure 3: Ray mesh: the envelope $\mathcal{E}$ and its corresponding map $\mathcal{R}$.

Ray mesh. The ray mesh format is a 3d mesh envelope[3] where each vertex has 5 components: (ρ, θ, ϕ) its spherical coordinates and (t_x, t_y) the corresponding texture coordinates inside the map (see Figure 3). Technical considerations[4] when implementing efficiently those ray meshes are left in [1].

Often, to get a good sampling of the sphere, one use a icosahedron or a surface refinement of it. For example, B. Fuller's map is a special unfolding of an icosahedron onto a map and can be considered as a ray mesh (See Figure 4, 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, remains a challenging open problem in itsef. Ray mesh can be controlled easily dynamically with time. Using a GUI, we set more precision on people's faces, etc.

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 this oversampling of information at the poles by sampling proportionally the latitude circles. Let $\phi_h = -\frac{\pi}{2} + \pi\frac{h}{H}$, 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[5] of $1 - \frac{2}{\pi} \simeq 0.36$. Also, we keep the advantage that rays are stored in a 2d lexicographic order (θ, ϕ) so that ray interpolation (see below) and texture can be used without any indexing problem (See Figure 4, top right).

Stratified random. Yet another property of the sphere is that any z-slice (portion of the sphere sandwiched between to planes perpendicular to the z-axis) 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 map 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 coordinate of the samples so that it remains in its ϕ-slice) that we sort and store (See Figure 4, bottom left).

[3] An envelope $\mathcal{E}$ is a surface centered at O such that any ray emanating from O intersects at most once $\mathcal{E}$.

[4] Mipmap and interpolation filterings, triangle stripping, etc.

[5] In practice this means that we obtain better image quality from the video coded at a constant bite rate.

Hammersley sequence. This is a deterministic sequence of points distributed on the sphere with proven low discrepancy, i.e. good sampling. It has been used before in computer graphics for Monte-Carlo sampling algorithms. Each nonnegative integer k can be written using a prime base p as $k = a_0 + a_1 p + a_2 p^2 + ... + a_r p^r$, where each $a_i \in \{0, ..., p-1\}$. Define function $\Phi_p(k) = \frac{a_0}{p} + \frac{a_1}{p^2} + ... + \frac{a_r}{p^{r+1}}$. The sequence $\Phi_2(k)$ for $k = 0, 1, 2, ...$ is called the Van der Corput sequence. The mapping to points (or equivalently directional vectors) on the spherical surface can be done by mapping $(\frac{k}{n}, \Phi_p(k)) \rightarrow (\theta, t) \rightarrow (\sqrt{1-t^2}\cos\theta, \sqrt{1-t^2}\sin\theta, t)$. We refer the reader to [7] for details concerning the efficient implementations of those sequences. One inconvenient is that we do not have an easy 2d lexicographically indexing order of the rotations. We overcome this problem when retrieving image from a virtual camera using 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 neighbouring directional vectors obtained from the first step. (We used them for interpolation).

4. Viewer

We can either draw per-pixel or per-texture elements. Because of widely available hardware capabilities, we chose to render using 2d textured triangle primitives. For each triangle $t = \{p_1, p_2, p_3\}$ with p_i associated to (θ_i, ϕ_i) of the ray mesh, we compute the corresponding position of the corresponding triangle in the xy-screen as $x_i \sim \mathbf{KR}\vec{r}_{\theta_i,\phi_i}$, where $\mathbf{R}$ is a rotation matrix and K is the pinhole projection matrix: $\mathbf{K} = \begin{pmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix}$, where (p_x, p_y) is the principal point, f_x and f_y are the x-focal and y-focal lengths respectively (obtained from horizontal and vertical fields of view).

Using matrix formulations, we do the intersection of a unit sphere with a line passing through the origin (instead of a ray), so we need to remove the ambiguity[6] 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 ray mesh not through an ideal perspective camera but from any kind of camera (exotic models like cameras with 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 $r(\phi) = f_1\phi + f_2\phi^2 + ...$ for a fisheye camera C_F using the equi-distance projection model (See Figure 5).

[6] The line intersects the sphere into two antipodal points.

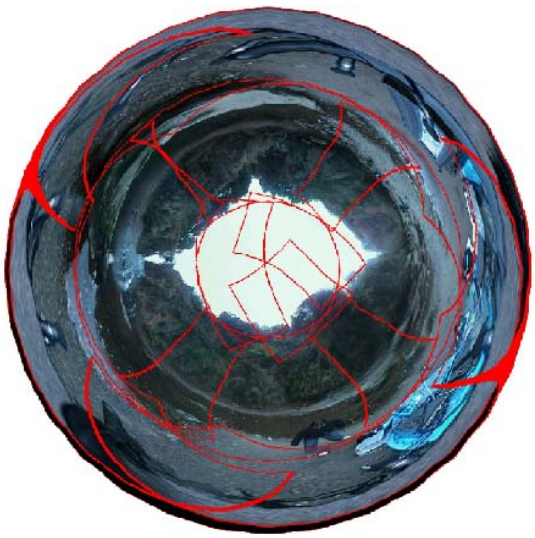

Figure 5: A 360-degree fisheye view created from an environment map.

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 neighbour, 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 ray interpolation becomes pixel interpolation when the focal tends to infinity (orthographic projection). 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). Let $L(\cdot)$ be a bijective function used for going from the ideal pinhole to the original image (Figure 6). We select a neighborhood of (x', y') and using $L^{-1}(\cdot)$, we find their respective coordinates in the ideal image (see Figure 6). Finally we can use our ray interpolation

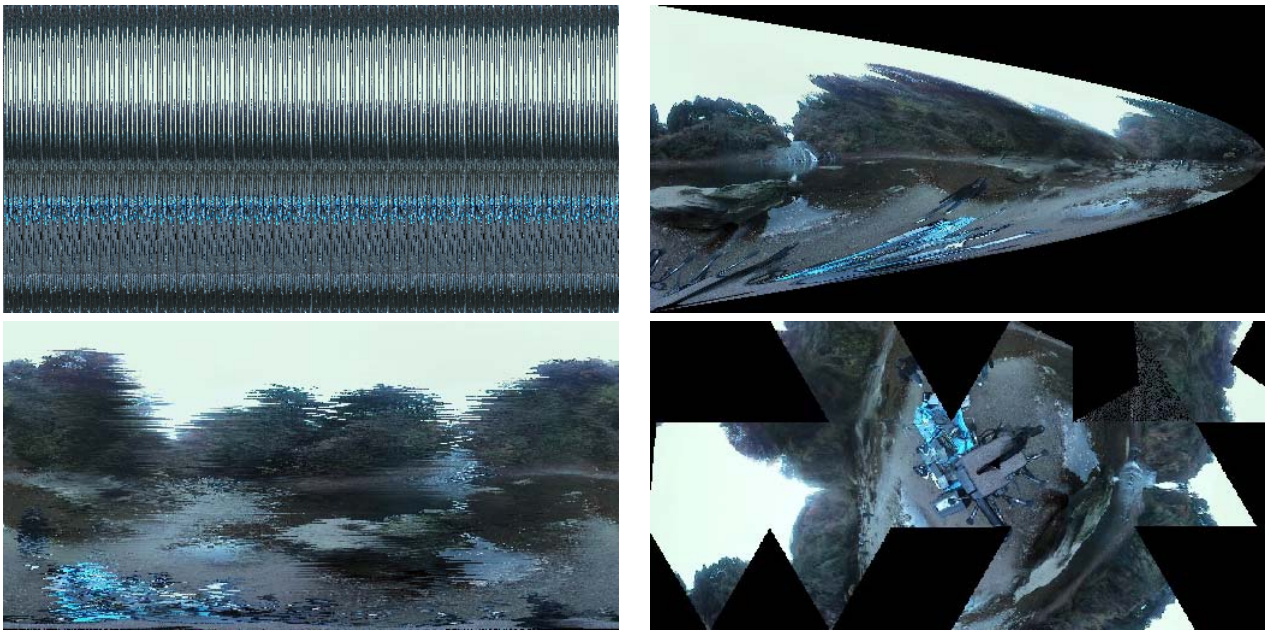

Figure 4: Top left: Hammersley map. Top Right: Compressed spherical map. Bottom left: Stratified random. Bottom right: B. Fuller's map

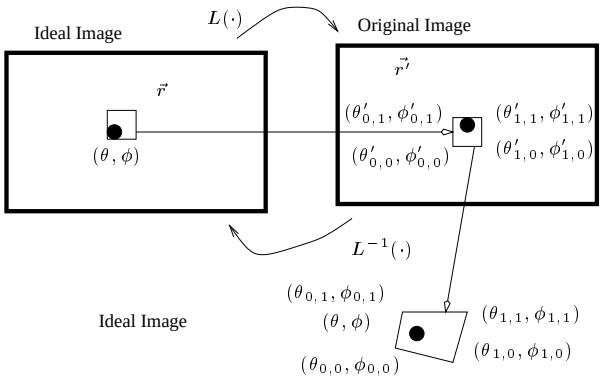

Figure 6: Ray interpolation.

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 or quaternion interpolation.

6. Experiments

Our camera prototype is mobile for outdoor shooting. We can control the camera CCDs individually through an IEEE 1394 link interface. We recorded each camera video onto DV tapes for offline processing. Finally we implemented our viewer on a low-cost PC and playback MPEG-2 streams (15 Mpbs CBR) at 30 fps from DVDs. We also added 3d surround sound (data are obtained from a micro array) to complete immersive experience. People can interact either by a joystick or by using a head-mounted display which orientation is tracked by a gyroscope. We present several full view immersive videos from indoor/outdoor shootings and some synthesized immersive videos computed by raytracing and radiosity renderings.

References

[1] "Immersive Video Envelopes," Sony CSL TR, 50 p., 2001.

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

[3] "Non-Metric Calibration of Wide-Angle Lenses and Polycameras," *IEEE trans. on Pattern Analysis and Machine Intelligence*, (22) 10 pp. 1172-1178, 2000.

[4] J. F. Blinn and M. E. Newell, "Texture and Reflection in Computer Generated Images," *Communications of the ACM*, Vol. 19(10), pp. 542-547, 1976.

[5] N. Greene, "Environment Mapping and Other Applications of World Projections," *IEEE Computer Graphics and Applications*, Vol. 6(11):21-29, 1986.

[6] W. Heidrich and H.-P. Seidel, "View-independent Environment Maps," *Eurographics/ACM SIGGRAPH Workshop on Graphics Hardware*, 1998.

[7] 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.