The Rhombic Dodecahedron Map: An Efficient Scheme for Encoding Panoramic Video

Chi-Wing Fu, *Member, IEEE*, Liang Wan, *Member, IEEE*, Tien-Tsin Wong, *Member, IEEE*, and Chi-Sing Leung, *Member, IEEE*

***Abstract*—Omnidirectional videos are usually mapped to planar domain for encoding with off-the-shelf video compression standards. However, existing work typically neglects the effect of the sphere-to-plane mapping. In this paper, we show that by carefully designing the mapping, we can improve the visual quality, stability and compression efficiency of encoding omnidirectional videos. Here we propose a novel mapping scheme, known as the *rhombic dodecahedron map* (RD map) to represent data over the spherical domain. By using a family of skew great circles as the subdivision kernel, the RD map not only produces a sampling pattern with very low discrepancy, it can also support a highly efficient data indexing mechanism over the spherical domain. Since the proposed map is quad-based, geodesic-aligned, and of very low area and shape distortion, we can reliably apply 2D wavelet-based and DCT-based encoding methods that are originally designated to planar perspective videos. At the end, we perform a series of analysis and experiments to investigate and verify the effectiveness of the proposed method; with its ultra-fast data indexing capability, we show that we can playback omnidirectional videos with very high frame rates on conventional PCs with GPU support.**

***Index Terms*—Spherical mapping, rhombic dodecahedron, panoramic video, omnidirectional video, video encoding, graphics processing unit (GPU)**

I. Introduction

Panoramic video, or omnidirectional video [1]–[5], represents time-varying 360^o environments. It can be played back via wide field-of-view or immersive VR displays [6]–[11], and presents the captured environment in a highly immersive and dynamic fashion. Large immersive display applications require high resolution of video frames in order to maintain reasonable visual quality. Hence efficient representation of omnidirectional video without wasting pixels (or storage) is crucial. Existing video formats are, however, mostly designed for planar perspective videos, while omnidirectional videos are spherical in nature. Therefore spherical parameterization [12] (or sphere-to-plane mapping) is needed to map the spherical video frames to the planar domain in order to facilitate video compression methods originally designed for rectangular videos. Typical mapping schemes that researchers have explored so far include cylindrical projection [13]–[16] and Cube-map projection [17]. Cylindrical projection unfolds the video frames using a longitude and latitude grid, whereas the Cube-map projects the video frames onto the six faces of an enclosing cube. In this way, we can warp each video frame into a single 2D image (cylindrical projection) or six 2D square images (Cube-map projection), stack them up, and pass the image sequence(s) to standard video encoders like MPEG.

However, the *efficiency* of these mapping schemes has not been studied in depth. Most previous research simply ignores the effect of mapping on coding efficiency. Inappropriate mapping schemes may excessively and/or insufficiently sample the omnidirectional video frames and lead to waste and/or aliasing. In this paper, we focus on the study of mapping schemes with respect to omnidirectional video encoding, and propose a novel mapping scheme, called the *rhombic dodecahedron map* (RD map). It is based on the rhombic dodecahedron model [18], [19]. Compared with existing mapping schemes, this novel mapping scheme offers the following advantages:

- *Uniform Distribution:* The RD map offers a uniformly distributed sampling pattern. Each pixel in the RD map spans almost the same solid angle and hence importance. It offers an efficient and unbiased platform, without oversampling nor undersampling, for DCT-based and wavelet-based [20], [21] video encoding. This improves the compression efficiency as compared to the Cube-map based video encoding.
- *Small Distortion:* The shape distortions of subdivided pixels are small and very similar. This is important to achieve high-quality playback of the video frames.
- *Fast Retrieval:* The geodesic property of RD map offers an ultra-fast data indexing and retrieval algorithm. It can be realized and implemented on GPUs in ordinary PCs. The fast indexing algorithm allows a high-speed playback of omnidirectional video.

The paper is organized as follow: First of all, Section II reviews the existing works in applying various spherical maps to encode omnidirectional videos. Then, Section III presents the detail of our proposed rhombic dodecahedron map, whereas Section IV derives the mathematics for fast indexing on the map. Next, Section V contains three parts: 1) the implementation of our omnidirectional video system (including capturing, encoding, decoding, and playback) based on the rhombic dodecahedron map; 2) three quantitative analysis to compare the efficiency of the rhombic dodecahedron map against other mapping schemes; and 3) a series of experiments to verify the effectiveness of our proposed mapping scheme. Finally, Section VI draws the conclusion.

C.-W. Fu is with the School of Computer Engineering, Nanyang Technological University, Singapore; email: cwfu@ntu.edu.sg;

L. Wan was with the Department of Computer Science & Engineering, the Chinese University of Hong Kong; email: lwan@cse.cuhk.edu.hk, the work was started during her time at the Chinese University of Hong Kong; and is with Department of Electronic Engineering, the City University of Hong Kong;

T.-T. Wong is with the Department of Computer Science & Engineering, the Chinese University of Hong Kong; email: ttwong@cse.cuhk.edu.hk;

C.-S. Leung is with the Department of Electronic Engineering, the City University of Hong Kong; email: eeleungc@cityu.edu.hk

Manuscript received June 6, 2007; accepted January 30, 2009.

II. Related work

Omnidirectional Images and Videos

Chen [22] developed the QuickTime VR system, which models a 3D environment in the form of a 360-degree cylindrical or cubical panoramic image. Users can pan, zoom in/out, and link to different sites through the hotspots in the image. Szeliski and Shum [23] later proposed a warping and stitching algorithm to combine multiple perspective images into a single panoramic image. Shum et al. [24], [25] further extended the idea and proposed the concentric mosaic model to make panoramic images navigatable. Wong et al. [26] incorporated image-based relighting techniques into panoramic images and make the panoramic images relightable in real-time. Agarwala et al. [27] developed an efficient algorithm to create panoramic video textures so that we can add persistent motions over static panoramic images.

Majumder et al. [6] developed a teleconferencing system using panoramic videos; images were captured from a cluster of video cameras, and registered and blended to form a panoramic view of the environment. Other teleconferencing applications that involve panoramic videos include [13] and [28]. Rather than using a large projection surface for presenting panoramic videos, Neumann et al. [7] employed a head-mounted display (HMD), in which the renderings change naturally with the users' head. Foote [8] developed a practical and inexpensive capturing system called the FlyCam [10], and applied it to create spatially-indexed panoramic videos [9]. Tang et al. [11] developed a video-based tele-immersive system, called the Immersive Cockpit. The proposed system displays streamed and stitched video sequences on a hemispherical display. Other than using camera clusters, omnidirectional cameras [3], [5] that make use of fisheye lens or hyperboloidal mirror reflection provide another efficient mean for capturing panoramic videos [15], [29], [30] without image stitching.

Mapping for Omnidirectional Videos

Bauermann et al. [15] were among the first who investigated the problems in encoding omnidirectional videos. Rather than directly encoding a raw omnidirectional video freshly captured through a fisheye lens or a hyperboloidal mirror, they first warped and re-sampled the raw video through a cylindrical projection before the encoding. More specifically, Smolić and McCutchen [16] studied some mapping schemes in cartography and employed equal-area cylindrical projection rather than a basic cylindrical map to further enhance the encoding efficiency. Other than cylindrical maps, Ng et al. [17] employed the Cube-map projection to encode omnidirectional videos. They dissected an omnidirectional video frame into six Cube-based tiles for their specially-designed MPEG encoding.

Problems with Existing Mappings

Except [15], [16], which explored area ratio in the mapping according to encoding efficiency, in all the other previous work we noticed, the purpose of mapping is simply to warp the captured video into encodable form, where the compression efficiency of the mapping are ignored.

As pointed out in [15], standard video encoding schemes such as MPEG are originally designed for sequence of images that are planar perspective in nature. The motion estimation and compensation mechanism assume translational motion of small image blocks in 2D rectangular domain. Omnidirectional videos are, however, non-rectangular in nature. Hence, the encoding assumptions in MPEG may no longer valid. Therefore, Bauermann et al. [15] applied cylindrical projection to warp omnidirectional videos into cylindrical form before the encoding. Though this warping improves the encoding efficiency for MPEG, the top and bottom areas around the poles, however, have to be ignored.

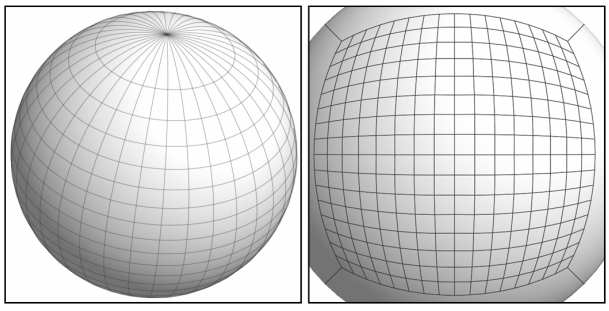

Fig. 1. Equal-area cylindrical map (left) and Cube-map (right).

Owing to the need to setup a mapping for video encoding, Smolić and McCutchen [16] employed the equal-area cylindrical projection, which brought in an equal-area projection from the spherical domain to the warped cylindrical surface. Though the mapped pixels on the sphere all have equal area, this mapping fails to address sampling uniformity and stretch distortion (aspect ratio or shape) on pixels, which in turn could heavily distort moving objects in the warped videos. This problem is especially severe around the poles or high latitude areas because pixels in these areas are largely stretched, see Figure 1(left). Other than driven by the equal-area objective, the Cube-map approach [17] gets rid of the pole problem in the mapping by having six planar perspective projections, each corresponding to a cube face. However, pixels near the center of cube faces occupy relatively larger solid angles (spherical areas) as compared to pixels near the face corners (Figure 1(right)). Moreover, pixels near the face corners are heavily distorted. Hence, objects moving in the scene could appear to have different sizes and shapes in the image space even they move at a fixed distance from the camera. In existing methods, the (pixel) sampling uniformity in the spherical domain is ignored, even it is a crucial factor in the encoding.

III. The Rhombic Dodecahedron Map

This paper presents a novel mapping scheme aiming at low area, shape, and stretch distortion, highly uniform pixel sample distribution, as well as fast data indexing on the spherical domain. The base geometric model employed in the mapping is a rhombic dodecahedron. In this section, part A first describes the geometric model, and part B presents the basic great circle subdivision scheme for mapping creation. Then, part C introduces the normal curve idea, which forms the basis for the skew subdivision scheme proposed in part D.

A. Rhombic Dodecahedron

A rhombic dodecahedron is a convex polyhedron with twelve identical rhombi on its surface. Figure 2 presents one

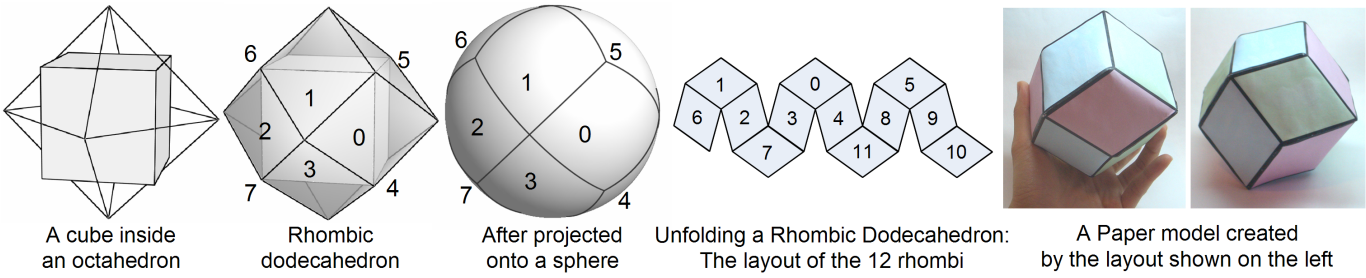

Fig. 2. The geometric structure of a rhombic dodecahedron.

typical way to construct its geometry. First, we inscribe a cube in an octahedron with the cube's edges just touching the edge mid-points of the octahedron. Then, we can join the cube's eight vertices with the octahedron's six vertices in a way shown in the second sub-figure, so that $3 \times 8 = 24$ edges are created to form the twelve identical rhombi in the rhombic dodecahedron. Next, if we project these rhombi onto a sphere surface using a gnomic projection, we can create a spherical rhombic dodecahedron model with twelve identical spherical rhombi, all with equal area on the sphere. Note that the 4π sphere surface is now segmented into twelve identical regions, where the segmented boundaries are the geodesic edges from the spherical rhombic dodecahedron model.

Furthermore, it is worth to note that not all kinds of geometric model supports efficient signal processing. The given model has to be quad-based in nature so that off-the-shelf signal processing techniques can be directly employed. Other than the longitude-latitude grid and the Cube-map, rhombic dodecahedron also satisfies this criteria.

B. Great Circle Subdivision Scheme

After segmenting the sphere surface using a spherical rhombic dodecahedron, our next task is to subdivide each spherical rhombus so that we can create pixel areas on the sphere surface and map the 4π spherical domain into 2D rectilinear grids. The very first strategy we applied here is great circle subdivision.

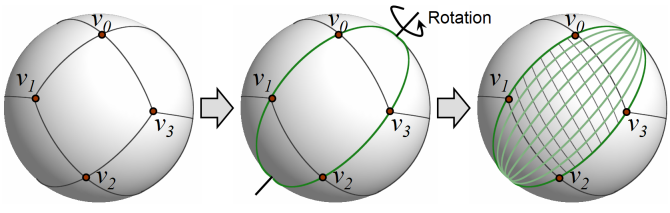

Fig. 3. The great circle subdivision scheme.

Figure 3 illustrates our great circle subdivision scheme on a spherical rhombus with on-sphere vertices v_0, v_1, v_2, and v_3. Since edges of spherical rhombi are geodesics on sphere, we can construct great circles to pass through them, see the middle sub-figure: The two constructed great circles through v_0v_1 and v_3v_2 intersect at two opposite points on sphere, where these two points can form a rotational axis that turns the great circle through v_0v_1 to the great circle through v_3v_2. By applying a uniform speed rotation about this axis, we can create a family of great circles to partition the spherical rhombus into vertical strips. Then, by repeating this process for the other pair of opposite edges of the rhombus, we can partition the rhombus into a grid of spherical areas (pixels) as shown in the right sub-figure. Since all the twelve spherical rhombi are identical in size and shape, this subdivision method can be applied to all of them to create the same subdivision.

C. Normal Curve

Though the great circle subdivision looks straightforward, it indeed implicitly embeds an interesting geometric structure, which in turn enables us to explore more efficient subdivision schemes. In essence, given any great circle on sphere, we can uniquely define an axis through the sphere center in such a way that the axis is perpendicular to the plane containing the great circle. If we specify a certain spinning direction to the great circle, we can uniquely define a normal vector perpendicular to the great circle plane according to the right-hand rule. As an example, we could compute $\hat{v}_0 \times \hat{v}_1$ and $\hat{v}_3 \times \hat{v}_2$ (see Figures 3 and 4) to obtain two normal vectors, $\hat{n}_{01}$ and $\hat{n}_{32}$, for the rhombus edges, v_0v_1 and v_3v_2, respectively. Just like vertices of the spherical rhombus, these normal vectors are also points on the sphere. In addition, note also that we have taken $\hat{v}_0$ to $\hat{v}_1$ and $\hat{v}_3$ to $\hat{v}_2$ as the common spinning direction for the two great circles through v_0v_1 and v_3v_2, respectively.

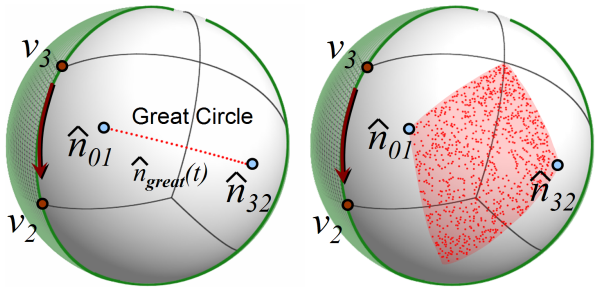

Fig. 4. A normal curve defines a subdivision scheme.

Since normal vectors and great circles are one-to-one correspondence, we can interpret the great circle subdivision scheme we established in previous subsection in terms of normal vectors. Taking a family of N partitioning great circles generated by the great circle subdivision scheme, we can plot their corresponding normal vectors on the sphere. Since these great circles are generated by a uniform speed rotation, their corresponding normal vectors thus form a geodesic curve between $\hat{n}_{01}$ and $\hat{n}_{32}$ on the sphere, see Figure 4 (left). Any point on this geodesic curve corresponds to a unique partitioning great circle between v_0v_1 and v_3v_2.

This interpretation enables us to have a different view on subdivision scheme; the problem of creating a family of partitioning great circles between opposite rhombus edges can

be interpreted as a problem of creating a valid spherical curve, called the *normal curve*, joining $\hat{n}_{01}$ and $\hat{n}_{32}$. Any point on this designated curve corresponds to a partitioning great circle between v_0v_1 and v_3v_2. In essence, a valid normal curve, say $\hat{n}(t)$ ($t \in [0,1]$), has:

- $\hat{n}(0) = \hat{n}_{01}$ and $\hat{n}(1) = \hat{n}_{32}$,
- $\forall t \in [0,1], G(\hat{n}(t))$ should cut through the other two opposite edges (i.e., v_0v_3 and v_1v_2) of the given rhombus, and
- $\forall t_1, t_2 \in [0,1]$, if $t_1 \neq t_2$, $G(\hat{n}(t_1))$ and $G(\hat{n}(t_2))$ should not intersect within the given rhombus,

where $G(\hat{n}(t))$ is the great circle corresponding to point $\hat{n}(t)$ on the normal curve. The first constraint is the end-point condition for the partitioning, while the second constraint ensures that all generated great circles cut through the given rhombus properly; the highlighted red region in Figure 4 (right) denotes a region of valid normal vectors obeying this constraint. The last constraint guarantees no intersection between partitioning great circles within the given rhombus.

D. Skew Great Circle Subdivision Scheme

In looking for a good normal curve that satisfies the above constraints as well as can improve the data encoding efficiency, we find that we can apply a small circle to join $\hat{n}_{01}$ and $\hat{n}_{32}$ rather than just a geodesic, see Figure 5. Note that on sphere, the radius of a great circle always equals the radius of the sphere, but the radius of a small circle can take any value smaller than that.

Since small circles on sphere can have different radii, we can optimally choose a small circle that maximizes the quantitative measures to be presented in Section V (with detailed analysis and comparisons). Based on the optimization, we found the small circle curve, $\hat{n}_{small}(t)$, shown in Figure 5; its corresponding subdivision pattern is shown in Figure 6. The family of great circles generated by this subdivision kernel forms a curve stitching pattern on the sphere as they do *not* rotate about the pole. In Section V, we will show that this off-pole rotation can lead to a much better pixel distribution that has higher data uniformity, and lower area and stretch distortion than that of straightforward on-pole rotation (great circle subdivision). We call the subdivision method the *skew great circle subdivision scheme*, which is the core of our rhombic dodecahedron map (RD map).

IV. Data Indexing

This section continues our discussion on the rhombic dodecahedron map and derives the mathematical detail for the fast indexing method of this map.

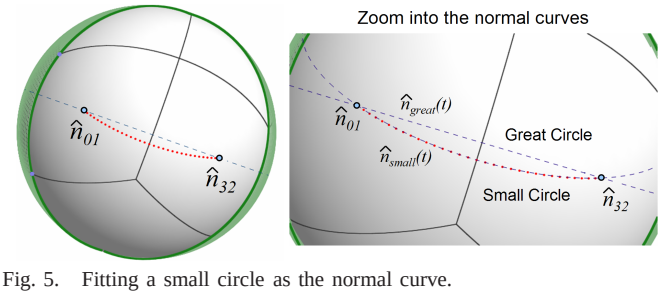

Fig. 5. Fitting a small circle as the normal curve.

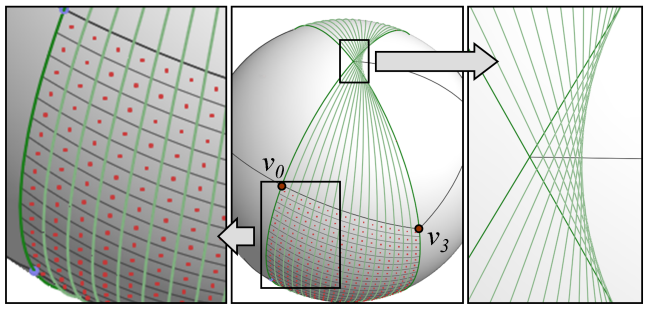

Fig. 6. The skew great circle subdivision scheme forms a curve stitching pattern at the pole of rotation in the great circle subdivision scheme; left: the partitioning; middle: $\hat{n}_{small}(t)$; Right: zoom at the pole.

A. The Data Retrieval Problem

Systematically, the retrieval of data in the RD map can be formulated as: *Given a query point $\hat{q}$ on the sphere, what is the associated data value at $\hat{q}$?* In answering this question, we have to locate the pixel area that contains $\hat{q}$ on the sphere, and retrieve or interpolate the pixel value at $\hat{q}$.

The process of locating the pixel area that contains $\hat{q}$ can be reduced to two sub-steps: 1) locate the base spherical rhombus that contains $\hat{q}$, and 2) locate the corresponding pixel on the rhombus. The first sub-step is relatively simple. Since edges of a rhombic dodecahedron follow a system of six great circles, we can perform six dot products between $\hat{q}$ and the great circle normals, create a 6-bit integer with zeros and ones corresponding to the dot product results (positive and negative), and quickly locate the spherical rhombus that contains $\hat{q}$ by a table lookup. Furthermore, since the six great circles are not arbitrarily orientated, we can further speedup this sub-step by simplifying the dot product multiplications. As a result, this sub-step requires just eight fragment program instructions in our GPU implementation.

B. On Deriving the Indexing Equation

The normal curve concept is useful not only for the design of subdivision scheme, it is also useful for the second sub-step in the data indexing problem. The underlying reason is that any valid normal curve, say $\hat{n}(t)$, corresponds to a family of non-intersecting great circles sweeping through a given spherical rhombus from one side to the other. Thus, every single location on a spherical rhombus is associated with a certain great circle (for each set of partitioning great circles) defined with a certain t along a normal curve. In this way, if $\hat{q}$ locates inside a given rhombus, we can always find a certain great circle passing through it; and since the associated normal vector of this great circle is perpendicular to the great circle plane, we have

$$\hat{q} \cdot \hat{n}(t) = 0 \text{ for some } t \in [0,1].$$

Hence, the data indexing problem can be reduced to the above simple equation: given $\hat{q}$, solve for t. Furthermore, it is worth to note that since we employ a uniform-speed sampling along the small circle arc $\hat{n}_{small}(t)$, we can quantize the resultant t value, and quickly locate the subdivided strip that contains $\hat{q}$ on the spherical rhombus. Finally, by solving this simple equation again for the other set of partitioning great circles corresponding to the other pair of opposite edges on the

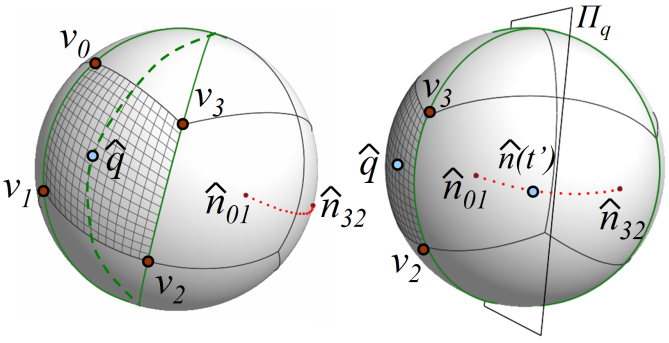

Fig. 7. Solving $\hat{q} \cdot \hat{n}(t) = 0$ by the intersection of Π_q and $\hat{n}(t)$.

spherical rhombus, we can analytically determine the specific pixel that contains $\hat{q}$ on the RD map.

From a geometric point of view, the above dot product equation can be further transformed to a plane-curve intersection problem. That is, given any $\hat{q}$ on sphere, we can construct a 3D plane, say Π_q, through the sphere center and perpendicular to $\hat{q}$, see Figure 7. If $\hat{q} \cdot \hat{n}(t) = 0$ has a solution, Π_q should also intersect $\hat{n}(t)$. As a result, we can formulate the mathematical procedure for solving t using Π_q, refer to the Appendix.

When we introduce normal curves, readers may question that small circle arcs cannot cover the space of all possible non-intersecting great circles in terms of the normal curve concept, so we may have some better normal curves than just small circle arcs. This could be true, but the use of small circle arcs does provide us with a fast data indexing that is implementable on the GPU, and also a promising subdivision pattern, see Section V. Taking these advantages into account, a more complicated normal curve hence may not be worthy.

V. Implementation, Analysis, and Experiments

This section contains three parts: The first part details the implementation of our omnidirectional video system (including capturing, encoding, decoding, and playback); the second part presents three complementary metrics (sampling uniformity, area deviation, and shape distortion) to quantitatively analyze and compare different mapping schemes; finally, the third part shows a series of experiments to analyze and evaluate the playback performance, the visual quality, the PSNR in data representation, the stability in rendering quality, and the video encoding efficiency.

A. Implementation: the Omnidirectional Video System

A.1 Video Capture We capture omnidirectional videos with Ladybug2, a spherical digital video camera from Point Grey Research [30]. Ladybug2 contains six cameras (five positioned in a horizontal ring and one pointing straight up). Each camera captures a perspective image at a resolution of 1024×768. The camera system covers approximately 75% of a full sphere and its 1394b transfer interface enables transfer rates of up to 30 fps. Figure 8 shows the system setup. It consists of a PC (An AMD Athlon 64x2 Dual Core PC with Processor 3800+, 2.01GHz, 2.0GB RAM, and Geforce 7800) and the Ladybug2 camera mounted on a tripod that sits on a movable platform.

Although Ladybug2 comes with a proprietary software that stitches the six views into a cylindrical panorama, it does not fully utilize the captured pixels for stitching a high-quality omnidirectional frame; resampling of the cylindrical panorama is needed to produce the RD map. Therefore, we developed our own stitching program to directly generate high-quality RD maps (frames) by resampling the raw frames from the six cameras, based on a user-specified subdivision number (the number of subdivisions on the rhombus faces). Each of the six cameras is carefully calibrated to obtain its intrinsic and extrinsic parameters in order to minimize the image distortion. Subsequently, we can efficiently encode and store each omnidirectional image in a highly uniform manner, visualizable as the fish-like form in Figure 9 (left).

Fig. 8. Left: Ladybug2 camera. Right: the setup of our capture system: a PC, the Ladybug2 camera device, and a tripod attached on a movable platform.

A.2 Video Compression For compatibility with the existing MPEG framework, we proposed a four-strip encoding scheme to pack the image blocks like the six-tile encoding scheme in [17]. Given the twelve rhombi in the fish-like form, we rearrange them as four consecutive strips shown in Figure 10. Using this scheme, four video streams are formed and each is encoded with the standard MPEG2.

A.3 Video Decoder and Renderer Our fast video playback system involves an MPEG2 decoder and an OpenGL-based renderer. In the decoding of an MPEG2-encoded video, we have a sequence of 2D textures (RD maps) in the form of the four-strip format on the texture memory of the GPU. The renderer we developed then handles this texture stream, and takes advantage of the parallel processing power on the GPU to lookup color values from them. In detail, the graphics hardware first interpolates the viewing direction for each pixel fragment on the screen using its rasterization engine. Then, a Cg fragment program addresses each pixel fragment encountered, and computes the RD map indexing algorithm so as to retrieve the color values stored on the RD map. Since fragment programs are executed in parallel on an array of pixel processing pipelines, we can largely boost up the RD map indexing performance and render the omnidirectional videos at a speed far greater than real-time requirement.

B. Analysis on the Mappings: Three Quantitative Metrics

B.1 Sampling Uniformity The classical approach we employed to measure the uniformity of sample point distribution is the discrepancy analysis method [31]–[33]. Its basic idea is to measure the variation of pairwise distances among sample

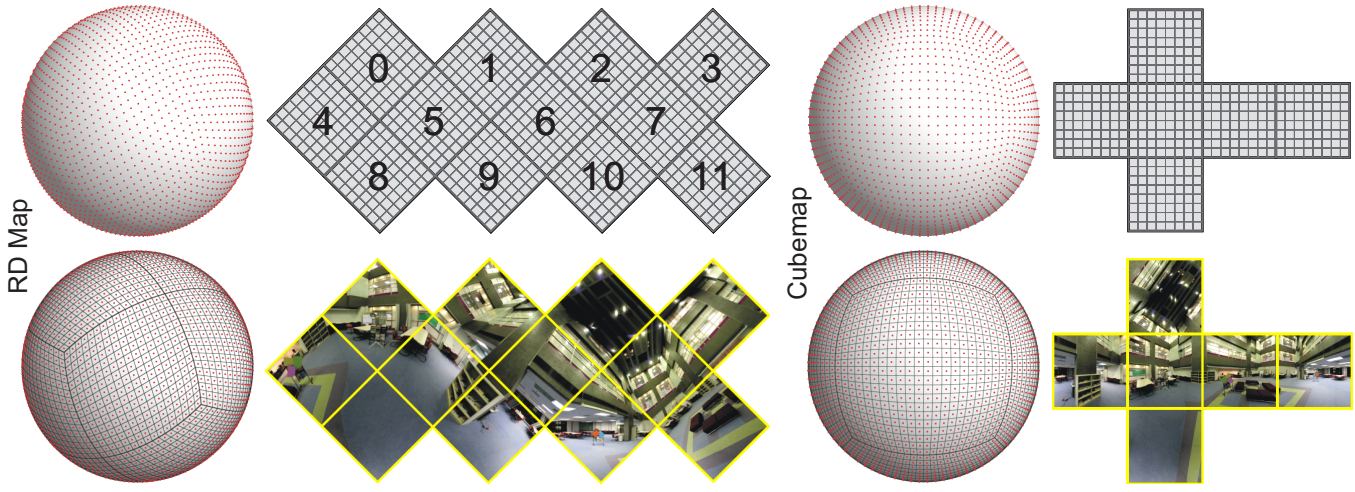

Fig. 9. Visual Comparison: the rhombic dodecahedron map (left) and Cube-map (right).

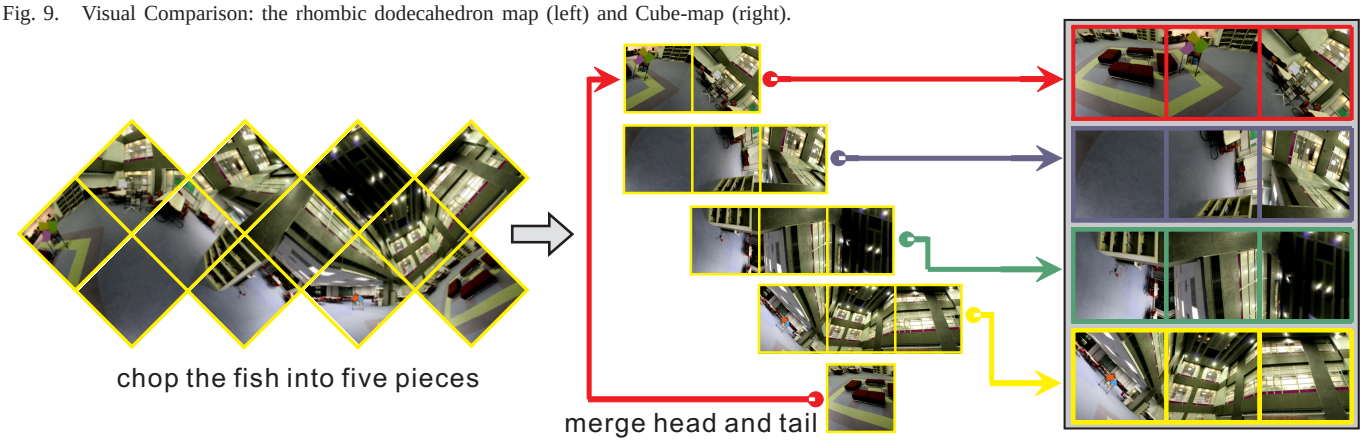

Fig. 10. Turning the fish-like form into four strips for MPEG-compatible encoding.

point locations; a smaller distance variation (known as the discrepancy) implies a more uniform sampling distribution. Based on this concept, Cui et al. [33] devised the *generalized discrepancy* formulae below for measuring sampling uniformity on the sphere:

$$D = \frac{1}{2\sqrt{\pi}N}\Big[\sum_{ij}^{N}\Big[1-2\ln\Big[1+\sqrt{\frac{1}{2}(1-\hat{x_i}\cdot\hat{x_j})}\Big]\Big]\Big]^{\frac{1}{2}},$$

where $\hat{x_i}$, i from 1 to N, is the location of the ith sample point on a unit sphere.

In terms of omnidirectional video encoding, a small discrepancy value corresponds to a more uniform mapping scheme, which further implies a more uniform data sampling and encoding over the sphere surface. Figure 11 (left) plots the generalized discrepancy of various mapping schemes against different number of sampling points. Since discrepancy has a large range when we have Equal Lattice and Cube-map in the plotting, we have to use logarithmic scale in the vertical axis. Note also that Equal Lattice refers to a uniform angular subdivision in a longitude and latitude grid, while Hammersley [33]–[35] is a sampling point sequence on the sphere, known to have very low discrepancy. Moreover, we can also see from the figure that our skew great circle scheme (the RD Map) has the smallest discrepancy among all; It outperforms Cube-map, equal-area cylindrical map, as well as the basic great circle subdivision.

B.2 Area Deviation Area deviation aims at measuring the pixel area utilization on the sphere. Quantitatively, we first census the spherical area of all subdivided pixels on the sphere surface, and determine the corresponding standard deviation, A_{stdev}. Thus, a small A_{stdev} implies a more uniform area mapping between the data space (sphere) and the parametric space (2D rectilinear domain).

However, it is also worth to note that a low A_{stdev} does not always imply a uniform sampling. As in the case of equal-area cylindrical projection, though its A_{stdev} is always zero, its discrepancy is no better than the RD map. Figure 11 (middle) plots the analysis results. Since equal-area cylindrical is an equal area projection, it is ignored for proper scaling in the plot. And for Hammersley, since it is merely a point sequence on sphere without any pixel structure, it could not be tested by area deviation (nor by the following stretch measure).

B.3 Shape Distortion A spherical mapping locally transforms an infinitesimal spherical region to a planar quadrilateral. It is important to measure how much the spherical pixel deviates from a 2D square. Researchers have proposed different stretch metrics for spherical mapping [36]–[39]. Snyder et al. defined the stretch using the larger singular value of the mapping's Jacobian [39]. Praun et al. computed the spherical L^2-stretch norm by L^2-integrating over the spherical surface [37]. Unlike the previous methods, we develop a stretch measure using the ratio of the two singular values and

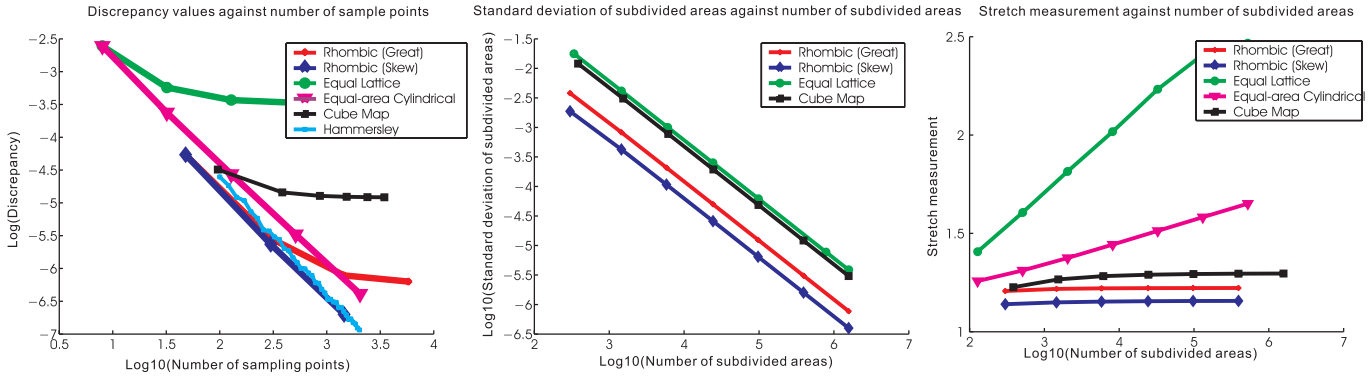

Fig. 11. Analyzing different mapping schemes: discrepancy (left), area deviation (middle), and stretch (right).

the ratio of a spherical pixel area and a 2D square,

$$S = \frac{1}{M}\sum_{i=1}^{M}\left((1+|1-\frac{A_i}{4\pi}\cdot M|)*(\frac{\Gamma_i}{\gamma_i})\right) \ ,$$

where M is the total number of subdivided pixel areas on the sphere, A_i is the area of the ith pixel on a unit sphere, and Γ_i and γ_i are its maximal and minimal singular values, respectively, after singular value decomposition (SVD). It is clear that the optimal stretch value should be 1.0 which happens when the pixels are equal-area and close-to-square. While the more the stretch is deviated from 1.0, the more the stretch or shape distortion a mapping produces. To be precise, we locally sample in each pixel area on the sphere by 2D grid points and project them onto the tangential plane so as to perform the SVD computation. From Figure 11 (right), we can see that the skew great circle scheme (the RD Map) also outperforms the other schemes, which implies that the RD Map introduces the least shape distortion for pixels on sphere.

C. Experiments

In this subsection, we carefully devise and conduct a series of experiments to study and verify the effectiveness of the proposed mapping scheme.

C.1 Playback Performance #1 Since the encoded data is stored according to the mapping scheme structure, in case we have intensive data query, for example, when playback an omnidirectional video, the speed of data retrieval is extremely crucial in the overall performance. For the RD map, since its analytical equation is simple and easy to implement, we can evaluate it by a fragment program running on GPU, and obtain over 140 fps in the video rendering.

TABLE I
PLAYBACK PERFORMANCE: VARYING RD MAP RESOLUTION.

	Number of pixels stored in the RD map			
	24300 (12×45^2)	97200 (12×90^2)	393132 (12×181^2)	1572528 (12×362^2)
Geforce 7800	207.92 fps	206.74 fps	205.60 fps	204.09 fps
Geforce 6800	160.97 fps	160.46 fps	160.62 fps	139.99 fps

Table I shows the frame rate in rendering omnidirectional videos with different video resolutions, i.e., RD map resolution. The timing statistics are recorded on an AMD Athlon 64x2 Dual Core PC with Processor 3800+, 2.01GHz, with 2.0GB of memory. Two different GPUs are employed in the experiment: nVidia GeForce 6800 and nVidia GeForce 7800 models. The performance results show that we can achieve a very high frame rate in the playback and such a frame rate is far enough to support real-time playback requirement (30 fps). Note that we use the texture rectangle format in OpenGL to represent 2D textures on GPU so that we can handle non-powers-of-two textures as in our RD maps, and the screen resolution is 980×980. This demonstrates the fact that we can always achieve very high framerate (playback speed) with varying screen resolution, even during video playback.

C.2 Playback Performance #2 In addition, we also compare the playback performance (rendering performance) against different number of on-screen pixels, i.e., screen resolution in the playback. Here we use the machine with Geforce 7800 as in experiment C.1, and the resolution of the RD map is 12×362^2. Furthermore, we try out two different configurations: The first one is a video stream, where we have to update the GPU texture per time frame (as in experiment C.1), whereas the second one is just a static texture on the GPU, see Table II for the performance results.

TABLE II
PLAYBACK PERFORMANCE: VARYING SCREEN RESOLUTION.

screen resolution	video playback	single image
400×400	243.81 fps	1499.16 fps
512×512	238.72 fps	1131.11 fps
640×640	231.84 fps	742.97 fps
800×800	217.17 fps	474.23 fps
980×980	204.15 fps	309.56 fps

C.3 Visual Comparison: RD Map and Cube-map Since Cube-map is a more popular and better choice of mapping than Equal lattice and equal-area cylindrical map, we visually compare Cube-map and RD Map: Figure 9 shows their basic structures with roughly the same number of sample points (located at center of pixels) distributed over the sphere. We can see from the grid representations that the RD map is less shape-distorted and has higher sampling uniformity as compared to the Cube-map. Instead of the 12-rhombus layout presented in Figure 2, we organize the twelve rhombi in a fish-like form shown in Figure 9 above. The face IDs shown on the fish figure match that shown in Figure 2.

Figure 12 visually compare the decoded results. For the top row, the same omnidirectional image is represented with the RD map in a face resolution of 181×181 and Cube-map in a face resolution of 256×256, while for the bottom row, we use a RD map of face resolution 362×362 and a Cube-map

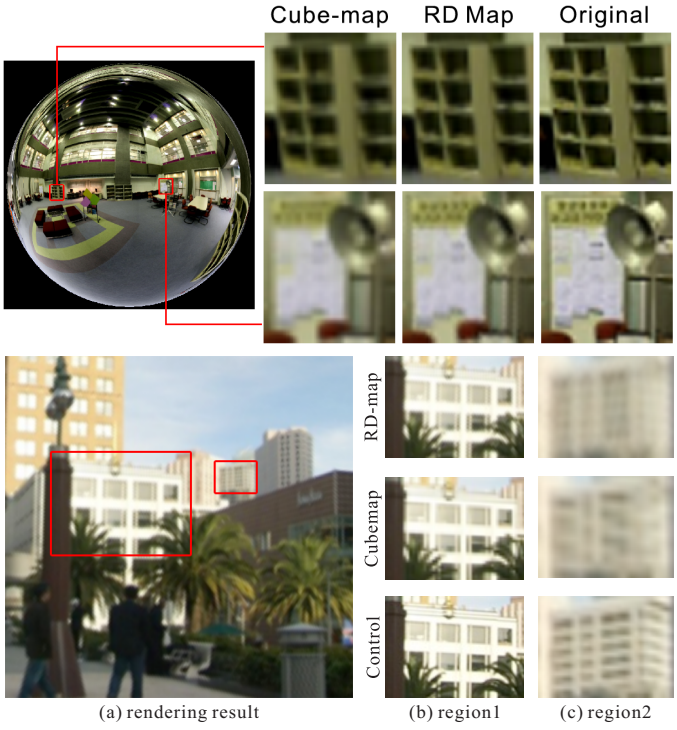

Fig. 12. Visual quality comparison of the RD map and Cube-map.

of face resolution 512×512; the control image is in Cube-map format with a resolution of 1024×1024. Thank to the sampling uniformity, low area deviation, and low stretch, the RD Map preserves more details of the bookshelf and the panel than the Cube-map.

C.4 Rendering Quality based on PSNR In addition, we also quantitatively compare the rendering quality in terms of PSNR. Here we measure how close the renderings are as compared to the control images. Note that we turn off MPEG encoding in this experiment (and also experiment C.5), just like what we did in Figure 12.

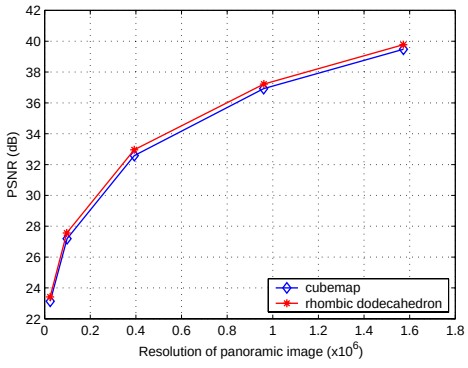

Fig. 13. PSNR achieved by Cube-maps and RD maps against the different map resolutions.

The same omnidirectional environment is encoded using Cube-map and RD map and peak-signal-to-noise (PSNR) ratio is used to measure the rendering quality against a control image, that is produced from a high resolution Cube-map. In detail, we randomly select 20 frames from the omnidirectional video sequence, which contains 322 frames. For each omni-directional frame, the scene (READINGROOM) is rendered at 60 different random viewing orientations and compared to the control images. The averaged PSNR of viewing directions is plotted against different resolutions of RD maps and Cube-maps, see Figure 13. From the figure, we can see that the rendering result of the RD maps have larger PSNR values than that of the Cube-maps. The improvement is consistent over all tested resolutions.

C.5 Stability in Rendering Quality Further than that, we also would like to point out that using the RD map, we can achieve more stable rendering quality. In this experiment, two viewing paths (the curves illustrated in Figure 14) on the unit sphere are arbitrarily selected to simulate how the user heads towards during the video playback. The camera is oriented so that it points along this viewing path, and the scene is rendered at 55 viewing orientations sampled on the path. These 55 viewing orientations are obtained by interpolating the quaternions at the six input key points that define the viewing path. We randomly select 20 omnidirectional frames from the READINGROOM omnidirectional video which is in a resolution of 6×256^2 for Cube-map and 12×181^2 for RD map. The averaged PSNRs are plotted against the 55 viewing orientations along the two user navigation paths.

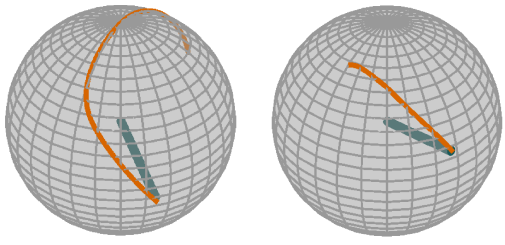

Fig. 14. Two viewing paths are arbitrarily selected; the tube pointing from the sphere center represents where the camera pointing to, initially.

Figure 15 shows that the PSNR varies with the content in the rendering and the PSNR resulted from RD map has the similar trend as that from Cube-map; however, the PSNR of Cube-map has a large fluctuation. It is due to the fact that Cube-map over-samples the corner regions and under-samples the central regions on its faces. On the other hand, our RD map achieves a more uniform sampling pattern over the spherical surface, and hence, results in a more stable visual quality.

C.6 MPEG2 Coding Efficiency Next, we evaluate the effect of the RD maps on MPEG2 coding efficiency. We use two omnidirectional video sequences, READINGROOM and WTY, that contains 322 and 564 frames, respectively, at frame rates of 30 fps. Both sequences are resampled in Cube-map at a resolution of 6×256^2 and RD map at a resolution of 12×182^2 (so that the two maps have similar resolutions). We then MPEG2-encode the videos with different target bit rates (bit per second) and compute the PSNR values of the decoded sequences against the original sequence. Note that the RD map has resolution 12×182^2 instead of 12×181^2 because the image resolution input to the employed MPEG encoder has to be even numbers. Unlike the previous two experiments (C.4 and C.5), which measure the PSNR of the rendered perspective views (planar in nature), the PSNR of the encoded *omnidirectional* video we measure this time is spherical in nature. The reason why we do not measure the PSNR in the "unfolded" rectilinear domain, where the

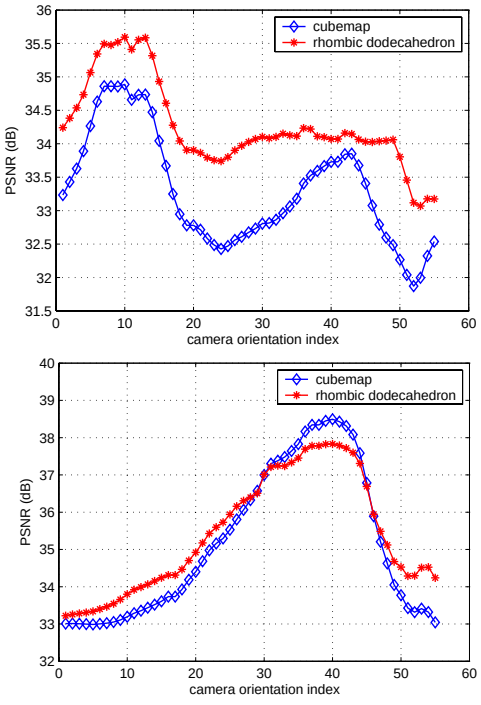

Fig. 15. PSNR achieved by Cube-maps and RD maps during the process of camera rotation; the top and bottom graphs correspond to the viewing paths shown on the left and right in Figure 14, respectively.

encoding is performed (i.e. the unfolded faces in Figure 9), is because it does not reflect the "physical" error as the video is actually spherical in nature. To correctly account for the contribution of each pixel over the spherical domain, when computing PSNRs, we have to multiply the error at each pixel by its corresponding solid angle (its spherical area). We called this weighted PSNR the *spherical PSNR* (sPSNR).

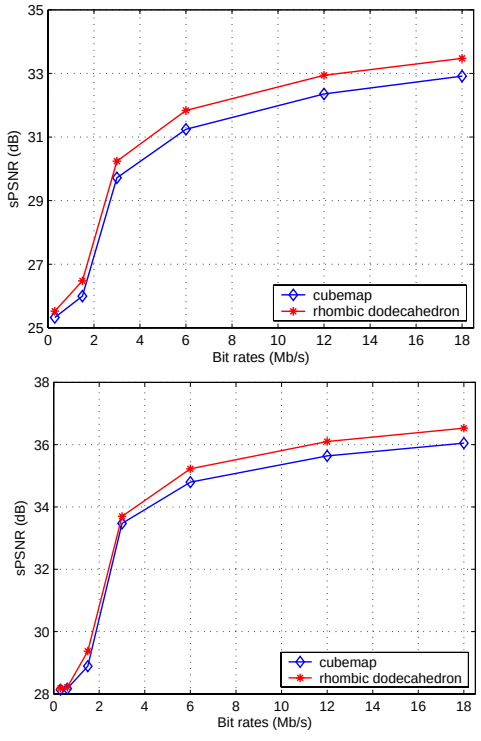

Fig. 16. sPSNR against target bit rates for Cube-map and RD map; top: READINGROOM and bottom: WTY.

Figure 16 compares the sPSNR values between Cube-map and RD map after video coding. The figure plots the curves of sPSNR values against the target bit rates. Here the target bit rates are the overall bit rates assigned to four-strip streams for RD map video, or similarly six-tile streams for Cube-map video. In general, the PSNR increases as the bit rate increases for both Cube map and RD map, and a bit rate of 3Mbps is sufficient to attain a decent sPSNR. In comparison, RD map has higher sPSNR values than Cube-map for both testing video sequences. From this experiment, we can see that the choice of mappings can affect the coding efficiency and that a uniform and low-distortion mapping, like the RD map, outperforms the efficiency of the traditional Cube-map.

VI. CONCLUSION

Sphere-to-plane mapping has a long history over two thousands of years [12]. Various mapping schemes have been proposed owing to different practical needs, such as cartography, navigation, etc. The RD map proposed in this paper is a novel mapping scheme favoring rectilinear-based encoding on the spherical surface. As the proposed mapping scheme is highly uniform, low in both area and stretch distortion, and supports ultra-fast data indexing, it can be used to map omnidirectional images and videos to the rectilinear domain and facilitate the encoding with the off-the-shelf image and video standards originally designed for rectangular domain.

We show that the effects of such mapping on omnidirectional video encoding cannot be neglected, as it affects the visual quality, stability, and compression efficiency. This novel mapping scheme provides a practical and effective platform for the development of an omnidirectional video system. Further than that, this mapping scheme can also lead to other future work yet to be explored. We shall investigate other potential applications of the RD map on different areas like allsky astrophysical imaging and shadow mapping in the future.

ACKNOWLEDGMENT

We would like to thank all reviewers for their constructive comment. We also thank Lai-Sze Ng for her contribution to the early development of this work. This project is partly supported by the Research Grants Council of the Hong Kong Special Administrative Region, under General Research Funds from Hong Kong Government (Project No. CUHK417107 and CityU116508), and the Nanyang Technological University Startup Grant (Project No. M58020007.500000).

REFERENCES

[1] E. H. Adelson and J. R. Bergen, "The Plenoptic function and the elements of early vision," in *Computational Models of Visual Processing*, M. S. Landy and J. A. Movshon, Eds. MIT Press, 1991, ch. 1, pp. 3–20.

[2] J. Meehan, *Panoramic Photography*. Amphoto Books, 1990.

[3] S. K. Nayar, "Catadioptric omnidirectional camera," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 1997, pp. 482–488.

[4] T. Svoboda, T. Pajdla, and V. Hlavác, "Epipolar geometry of panoramic cameras," in *Proceedings of the Computer Vision*, 1998, pp. 218–231.

[5] S. B. Kang, "Catadioptric self-calibration," in *Processings of the Computer Vision and Pattern Recognition*, 2000, pp. 201–207.

[6] A. Majumder, W. B. Seales, M. Gopi, and H. Fuchs, "Immersive teleconferencing: a new algorithm to generate seamless panoramic video imagery," in *Proceedings of the 7th ACM international conference on Multimedia*, 1999, pp. 169–178.

[7] U. Neumann, T. Pintaric, and A. Rizzo, "Immersive panoramic video," in *Proceedings of the 8th ACM international conference on Multimedia*, 2000, pp. 493–494.

[8] J. Foote and D. Kimber, "FlyCam: practical panoramic video," in *Proceedings of the 8th ACM international conference on Multimedia*, 2000, pp. 487–488.

[9] D. Kimber, J. Foote, and S. Lertsithichai, "FlyAbout: spatially indexed panoramic video," in *Proceedings of the 9th ACM international conference on Multimedia*, 2001, pp. 339–347.

[10] X. Sun, J. Foote, D. Kimber, and B. S. Manjunath, "Panoramic video capturing and compressed domain virtual camera control," in *Proceedings of the 9th ACM international conference on Multimedia*, 2001, pp. 329–347.

[11] W.-K. Tang, T.-T. Wong, and P.-A. Heng, "A system for real-time panorama generation and display in tele-immersive applications," *IEEE Transactions on Multimedia*, vol. 7, no. 2, pp. 280–292, 2005.

[12] J. P. Snyder, *Flattening the Earth : Two Thousand Years of Map Projections*. University of Chicago Press, 1993.

[13] D.-S. Lee, B. Erol, J. Graham, J. J. Hull, and N. Murata, "Portable meeting recorder," in *Proceedings of the 10th ACM international conference on Multimedia*, 2002, pp. 493–502.

[14] C. Grünheit, A. Smolić, and T. Wiegand, "Efficient representation and interactive streaming of high-resolution panoramic views," in *Proceedings of the International Conference on Image Processing*, vol. 3, 2002, pp. 209–212.

[15] I. Bauermann, M. Mielke, and E. Steinbach, "H.264 based coding of omnidirectional video," in *Proceedings of the International Conference on Computer Vision and Graphics*, 2004, pp. 22–24.

[16] A. Smolić and D. McCutchen, "3DAV exploration of video-based rendering technology in MPEG," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 3, pp. 348–356, 2004.

[17] K.-T. Ng, S.-C. Chan, and H.-Y. Shum, "Data compression and transmission aspects of panoramic videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 1, pp. 82–95, 2005.

[18] M. J. Wenninger, *Spherical Models*. Cambridge University Press, 1979, republished by Dover Publications, New York, 1999.

[19] E. W. Weisstein, "Rhombic dodecahedron," from MathWorld–A Wolfram Web Resource: http://mathworld.wolfram.com/RhombicDodecahedron.html.

[20] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG2000 still image compression standard," *IEEE Signal Processing Magazine*, vol. 18, pp. 36–58, Sep. 2001.

[21] B. Usevitch, "A tutorial on modern lossy wavelet image compression: Foundations of JPEG2000," *IEEE Signal Processing Magazine*, vol. 18(5), pp. 22–35, Sep. 2001.

[22] S. E. Chen, "QuickTime VR: an image-based approach to virtual environment navigation," in *SIGGRAPH 95 Conference Proceedings*. ACM SIGGRAPH, 1995, pp. 29–38.

[23] R. Szeliski and H.-Y. Shum, "Creating full view panoramic image mosaics and environment maps," in *SIGGRAPH 97 Conference Proceedings*. ACM SIGGRAPH, 1997, pp. 251–258.

[24] H.-Y. Shum and L.-W. He, "Rendering with concentric mosaics," in *SIGGRAPH 99 Conference Proceedings*, 1999, pp. 299–306.

[25] H.-Y. Shum, K.-T. Ng, and S.-C. Chan, "Virtual reality using the concentric mosaic: Construction, rendering and compression," in *Proceedings of the International Conference on Image Processing*, vol. 3, 2000, pp. 644–647.

[26] T.-T. Wong, P.-A. Heng, and C.-W. Fu, "Interactive relighting of panoramas," *IEEE Computer Graphics and Applications*, vol. 21, no. 2, pp. 32–41, 2001.

[27] A. Agarwala, K. C. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski, "Panoramic video textures," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 821–827, 2005.

[28] C. Liao, Q. Liu, D. Kimber, P. Chiu, J. Foote, and L. Wilcox, "Shared interactive video for teleconferencing," in *Proceedings of the 11th ACM international conference on Multimedia*, 2003, pp. 546–554.

[29] Y. Onoe, N. Yokoya, K. Yamazawa, and H. Takemura, "Visual surveillance and monitoring system using an omnidirectional video camera," in *Proceedings of International Conference on Pattern Recognition*, 1998, pp. 588–592.

[30] "Ladybug2," Point Grey Research, http://www.ptgrey.com/products/ladybug2/.

[31] D. P. Dobkin, D. Eppstein, and D. P. Mitchell, "Computing the discrepancy with applications to supersampling patterns," *ACM Transactions on Graphics*, vol. 15, no. 4, pp. 354–376, October 1996.

[32] P. Shirley, "Discrepancy as a quality measure for sample distributions," in *Proceedings of Eurographics*, 1991, pp. 183–193.

[33] J. Cui and W. Freeden, "Equidistribution on the sphere," *SIAM Journal on Scientific Computing*, vol. 18, no. 2, pp. 595–609, March 1997.

[34] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*. CBMS-NSF, SIAM, Philadelphia, 1992.

[35] T.-T. Wong, W.-S. Luk, and P.-A. Heng, "Sampling with hammersley and halton points," *ACM Journal of Graphics Tools*, vol. 2, no. 2, pp. 9–24, 1997.

[36] S. R. Buss and J. P. Fillmore, "Spherical averages and applications to spherical splines and interpolation," *ACM Transactions on Graphics*, vol. 20, no. 2, pp. 95–126, 2001.

[37] E. Praun and H. Hoppe, "Spherical parametrization and remeshing," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 340–349, 2003.

[38] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe, "Texture mapping progressive meshes," in *SIGGRAPH 2001 Conference Proceedings*, 2001, pp. 409–416.

[39] J. Snyder and D. Mitchell, "Sampling-efficient mapping of spherical images," Microsoft Research," Technical report, 2001.

APPENDIX

Solving $\hat{n}_{small}(t) \cdot \hat{q} = 0$

Mathematically, any small circle on a sphere can be interpreted as the cross-section of the sphere and a plane. Let Π_s be such a cross-section plane that contains $\hat{n}_{small}(t)$ and $\hat{a}$ be its normal, see Figure 17; note that any small curve arc can be uniquely defined by $\hat{n}_{01}$, $\hat{n}_{32}$, and $\hat{a}$. Now, we can define $\vec{h}$ as the projection of $\hat{n}_{01}$ along $\hat{a}$, and decompose $\hat{n}_{01}$ and $\hat{n}_{32}$:

$$\vec{h} \;=\; (\hat{n}_{01} \cdot \hat{a})\;\hat{a} \;\;\text{and}\;\; \left\{ \begin{array}{rcl} \hat{n}_{01} & = & \vec{h} \;+\; \vec{v}_{01} \\ \hat{n}_{32} & = & \vec{h} \;+\; \vec{v}_{32} \end{array} \right. ,$$

where $\vec{v}_{01}$ and $\vec{v}_{32}$ are the projections of $\hat{n}_{01}$ and $\hat{n}_{32}$ on Π_s, respectively.

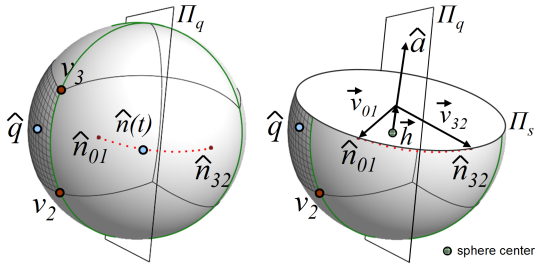

Fig. 17. Revealing the small circle plane, Π_s.

To accelerate the data indexing, we can pre-compute θ_s, l, $\hat{v}_{01}$, and $\hat{v}_{32}$ (note: these quantities are all on Π_s):

$$l \;=\; |\vec{v}_{01}|\;,\;\; \hat{v}_{01} \;=\; \vec{v}_{01}/l\;,\;\; \hat{v}_{32} \;=\; \vec{v}_{32}/l\;,\;\; \text{and}\;\; \theta \;=\; \cos^{-1}(\hat{v}_{01} \cdot \hat{v}_{32})\;.$$

Then, by using slerp interpolation from $\hat{v}_{01}$ to $\hat{v}_{32}$, we can represent the normal curve $\hat{n}_{small}(t)$ as

$$\hat{n}_{small}(t) \;=\; l \left[\frac{\sin((1-t)\theta)}{\sin\theta}\;\hat{v}_{01} \;+\; \frac{\sin(t\theta)}{\sin\theta}\;\hat{v}_{32} \right] \;+\; \vec{h}\;.$$

Since $\hat{v}_{01}$ and $\hat{v}_{32}$ are unit vectors, the slerp-interpolated result is always a unit vector. If $\hat{q}$ is found to be on the given base rhombus (from sub-step 1), the indexing equation becomes (put $l\hat{v}_{01} = \vec{v}_{01}$ and $l\hat{v}_{32} = \vec{v}_{32}$):

$$\hat{q} \cdot \left[\frac{\sin((1-t)\theta)}{\sin\theta}\;\vec{v}_{01} \;+\; \frac{\sin(t\theta)}{\sin\theta}\;\vec{v}_{32} \;+\; \vec{h} \right] \;=\; 0$$

$$(\hat{q} \cdot \vec{v}_{01})(\sin\theta\cos\alpha - \cos\theta\sin\alpha) + (\hat{q} \cdot \vec{v}_{32})\sin\alpha + (\hat{q} \cdot \vec{h})\sin\theta \;=\; 0\;.$$

By putting $A \;=\; [\;(\hat{q} \cdot \vec{v}_{32}) \;-\; (\hat{q} \cdot \vec{v}_{01})\cos\theta\;]$, $B \;=\; (\hat{q} \cdot \vec{v}_{01})\sin\theta$, and $C \;=\; (\hat{q} \cdot \vec{h})\sin\theta$, we have

$$A\sin\alpha \;+\; B\cos\alpha \;+\; C = 0\;.$$

Since A and B cannot be zero simultaneously (using proof by contradiction), the above equation can be further reduced by letting $\tan\phi = B/A$:

$$\begin{array}{rcl} \cos\phi\;sin\alpha \;+\; \sin\phi\;\cos\alpha & = & \dfrac{-C}{\sqrt{A^2+B^2}} \\ t & = & \dfrac{1}{\theta}\left[\;\sin^{-1}\left(\dfrac{-C}{\sqrt{A^2+B^2}}\right) \;-\; \phi\;\right]\;. \end{array}$$

Note that such an equation can be readily implemented (with proper transformations) as fragment program on the graphics processing unit (GPU).