Isocube: Exploiting the Cubemap Hardware

Liang Wan† Tien-Tsin Wong† Chi-Sing Leung‡
lwan@cse.cuhk.edu.hk ttwong@acm.org eeleungc@cityu.edu.hk

†The Chinese University of Hong Kong
‡City University of Hong Kong

***Abstract*— This paper proposes a novel six-face spherical map, *isocube*, that fully utilizes the cubemap hardware built in most GPUs. Unlike the cubemap, the proposed isocube uniformly samples the unit sphere (uniformly distributed) and all samples span the same solid angle (equally important). Its mapping computation contains only a small overhead. By feeding the cubemap hardware with the six-face isocube map, the isocube can exploit all built-in texturing operators tailored for the cubemap and achieve a very high frame rate. In addition, we develop an anisotropic filtering that compensates aliasing artifacts due to texture magnification. This filtering technique extends the existing hardware anisotropic filtering and can be applied to, not just the proposed isocube, but also other texture mapping applications.**

***Index Terms*— Isocube, Sampling on sphere, Cubemap, Equal solid-angle, Anisotropic filtering.**

I. Introduction

Environment mapping is a cost-effective way to raise visual richness and photorealism [1]. Millions of look-up operations may be required in rendering a frame. Hence the environment mapping functionality is normally hardwired in most GPUs. Most implementations adopt the six-face cubemap, due to its computational simplicity and memory-friendly rectilinear structure. However, the cubemap samples the spherical environment unevenly, more at the face corners while less at the face centers. Furthermore, each texel spans different solid angle (i.e. not equally important). Other common representations, the sphere map and the dual-paraboloid map [2], also do not sample the environment evenly.

Although there exist representations [3][4] that evenly sample the unit sphere (environment) and has the property of uniform solid-angle (all texels are equally important), their corresponding texture look-up processes require tailored shader implementations and they are not able to fully exploit the existing hardware. Unless they can exploit the hardware cubemap operators such as texture lookup and antialiasing operations, the speed performance of their shader implementations is rather low compared to that of the hardware cubemap. This motivates us to design a novel spherical representation that fits nicely into the *six-face* structure, so as to *pretend* itself to be a cubemap and exploit those operators originally designed for the six-face cubemap.

The proposed spherical representation is called, *isocube*, to indicate its properties of equal solid-angle and uniform sampling (low discrepancy). It is inspired by the *twelve-face* spherical representation, HEALPix [4], originated from astrophysics. Since it samples the environment uniformly and equally, the bias is minimized. From our experiments, its rendering quality is also slightly better than that of the cubemap. The isocube has a rather low computational cost. Furthermore, it can exploit the hardware cubemap operators and, hence, it achieves a very high frame rate.

In addition, we propose an anisotropic filtering that compensates the reconstruction aliasing due to *texture magnification*. The goal is to enforce filtering along the major edges in order to reduce the aliasing artifacts. This filter depends on both the texture gradient and the distance to the viewpoint. It exploits the existing hardware functionality and can be applied to, not just the isocube, but also other texture mapping.

In Section II, we first review common spherical parameterizations used in computer graphics. Section III describes the proposed isocube and the application of isocube environment mapping. Section IV discusses our anisotropic filtering for texture magnification. Implementation and results are presented in Section V. Section VI concludes our work.

II. Related Work

The cubemap is one of the most commonly used parameterizations of environment maps [1][5][6][7]. It represents the environment by a six-face cube, while each cube face is a perspective image as viewed from the cube center. The cubemap lookup process is extremely fast due to its efficient computation. While the cubemap is composed of six perspective images, effort has been made to design the tetrahedron map that requires only four perspective images [8][9]. A tetrahedron map requires less projections than a cubemap. However, its lookup process is more complex than that of a cubemap, since tetrahedron faces are triangular and not orthogonal to each other. Representing the sphere with eight perspective images, the octahedron map unfolds the sphere into a square other than the cross-structure of a cubemap [10]. Like the cubemap, the tetrahedron map or the octahedron map do not sample the environment evenly.

Another common parameterization is the sphere map [11], which resembles a mirror ball positioned in the environment. Fetching a texel from a sphere map is efficiently done by projective texture lookups. The sphere map however severely undersamples the environment near the sphere boundary. Because of the dependence on the viewing direction, the sphere map has to be updated whenever the viewing direction changes. The dual paraboloid map [2] extends the sphere map

so as to be independent of the viewing direction. Instead of a mirror ball, a paraboloid surface is adopted as the reflection object. The image seen by the orthographic camera that looks at the paraboloid comprises one hemisphere. As a result, a dual paraboloid map consists of two separate textures, one for each hemisphere.

Just like the dual paraboloid map tries to improve the sampling uniformity of the sphere map while maintaining the computational efficiency, the proposed isocube enhances the cubemap to make it equal solid-angle and uniformly sampled while achieving a very high frame rate.

III. Isocube Mapping

A. Constructing the Isocube Mapping

The proposed isocube is inspired by an allsky image *partitioning* scheme, Hierarchical Equal Area isoLatitude Pixelisation (HEALPix) [4][12]. The HEALPix starts by partitioning the spherical surface into 12 equal-area base quads, and recursively subdivides each quad to generate smaller sub-quads of equal area. The associated sampling pattern of this partitioning is found to be uniform (low discrepancy) on the sphere [12]. Figure 1 shows the HEALPix map by unrolling and concatenating the 12 base quads. Each base quad in this example is subdivided into 4×4 elements. Given a HEALPix map, the lookup and antialiasing operations have to be implemented with tailored shaders. Hence the speed performance is rather low. This motivates us to design a novel spherical representation so as to fully exploit the built-in cubemap hardware while maintaining the properties of equal solid angle and sampling uniformity.

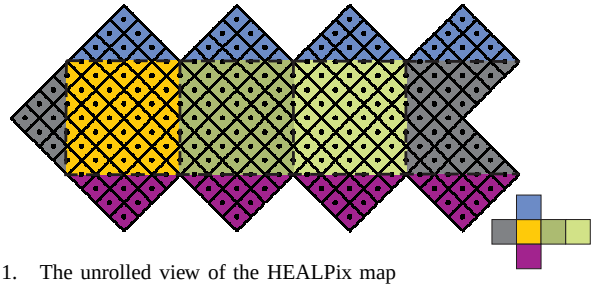


Fig. 1. The unrolled view of the HEALPix map

Initially we tried to pack the pixels of the HEALPix map into the six-face cube structure. Regions with the same color are packed into one cube face as suggested by the colors in Figure 1. However, a close inspection reveals that those pixels are not in a grid structure. Instead, we design a new set of partitioning equations on the sphere. This equation set partitions the spherical surface into 6 equal-area base faces instead of 12. Recursive subdivision generates smaller elements while maintaining the equal solid-angle property. As the resultant partition forms a six-face structure (*cube*) and all elements are the same in size (*iso*), we call the partitioning scheme as *isocube*.

Figure 2 illustrates the construction process of isocube base faces. We first partition the spherical surface into the equatorial zone and two polar zones with arctic/antarctic circles at $|z| = 2/3$ (Figure 2(a)). The area of the equatorial zone is four times that of each polar zone. The equatorial zone is further

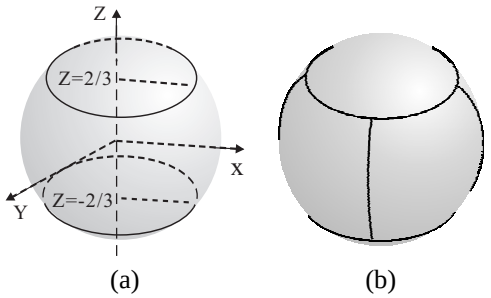


Fig. 2. Constructing isocube base faces: (a) the sphere is first partitioned into three zones, the equatorial zone in the middle, and two polar zones besides; (b) then the sphere is partitioned into six equal area base faces.

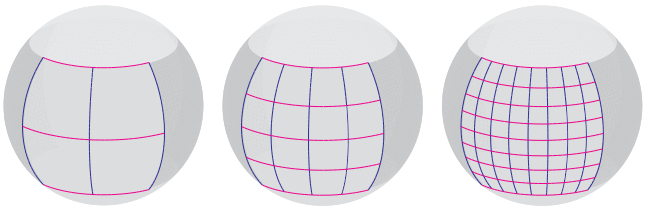


Fig. 3. Further partition one equatorial base face at resolutions (a) $N = 2$, (b) $N = 4$, and (c) $N = 8$.

subdivided into four symmetric equal regions. The resultant six base faces, as shown in Figure 2(b), have the same area (Appendix A).

Next we partition each base face into $N \times N$ elements. Denote ϕ as the azimuth angle, and θ as the polar angle while $z = \cos\theta$. Within the equatorial zone, we develop a set of k-curves and l-curves,

$$k\text{-curves: } \phi = \frac{\pi}{2}\left(\frac{k}{N} - \frac{1}{2}\right),\ k = 0, \ldots, 4N \qquad (1)$$

$$l\text{-curves: } z = \frac{2}{3}\left(\frac{2l}{N} - 1\right).\quad l = 0, \ldots, N \qquad (2)$$

Figure 3 shows the partitions at three resolutions on one equatorial base face, with k-curves indicated in blue color and l-curves in magenta color.

In the polar zones, we adopt a different set of partitioning equations. Due to the symmetry within the polar base faces, we only present the subdivision equations within the region where $(\phi, z) \in [\,0, \frac{\pi}{4}\,] \times [\frac{2}{3}, 1]$,

k-curves:

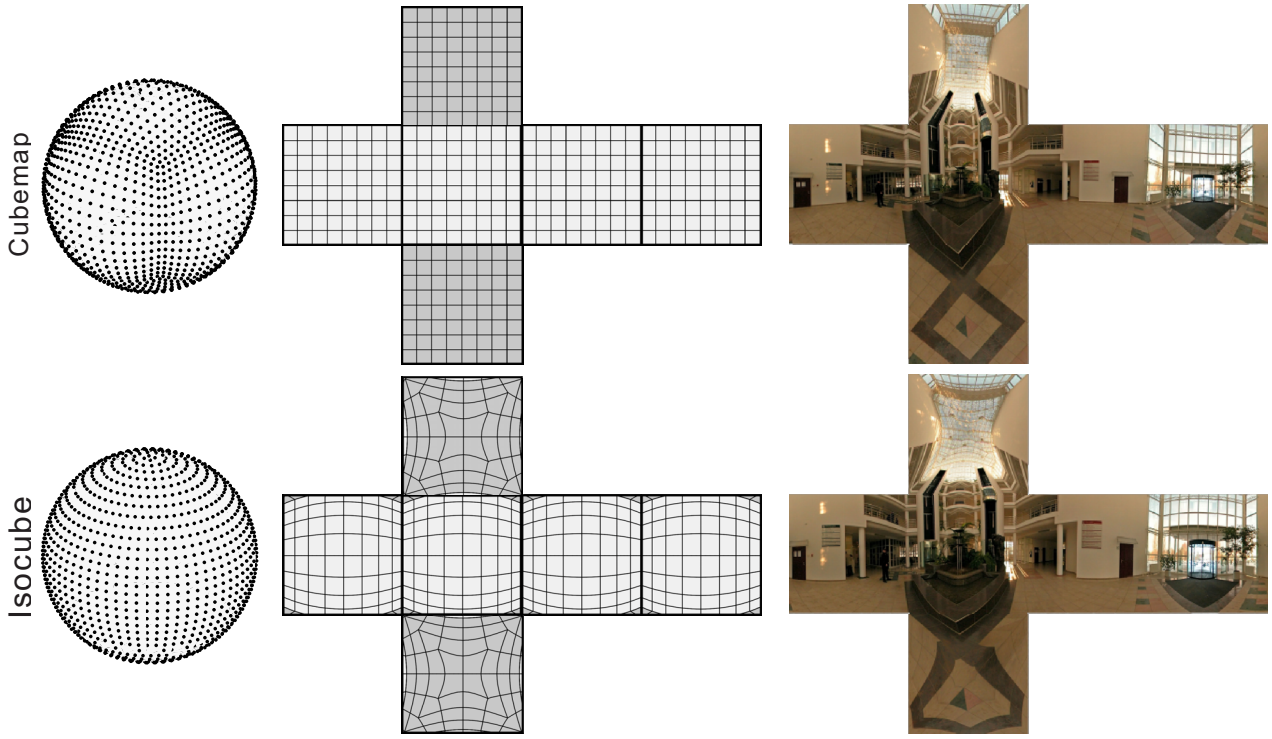
$$\phi = \frac{\pi}{2}\frac{2k+\varepsilon}{2N} \cdot \frac{1}{\sqrt{3(1-z)}}, k = 0, \ldots, \left\lfloor \frac{N}{2} \right\rfloor - 1 \qquad (3)$$

l-curves:

$$z = 1 - \frac{1}{3}\left(\frac{2l+\varepsilon}{N}\right)^2, \qquad l = 0, \ldots, \left\lfloor \frac{N}{2} \right\rfloor \qquad (4)$$

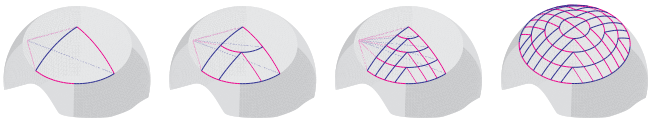
where $\varepsilon = N \bmod 2$. As shown in Figure 4, k-curves and l-curves in the polar base faces are not as regular as those in the equatorial zone, and may form corners. Figure 4(d) may remind careful readers the elevated concentric map [3]. Unlike the elevated concentric map which covers a hemisphere, our polar faces only span the arctic/antarctic regions.

The two sets of partitioning curves (for the equatorial and polar zones) guarantee the equal solid-angle property in the generated partition. Readers are referred to Appendix A for the



(a) sampling pattern (b) unrolled view (c) environment map

Fig. 5. (a) compares the associated sampling patterns of the cubemap and isocube. (b) shows how the cubemap is distorted on the isocube map. Basically, the straight lines in the cubemap may become curves in the isocube. (c) show the same environment in both maps (environment map courtesy of Tomáš Feltl).



(a) $N = 2$ (b) $N = 4$ (c) $N = 8$ (d) $N = 8$

Fig. 4. Partition the region where $(\phi, z) \in [\ 0, \frac{\pi}{4}\ ] \times [\frac{2}{3}, 1]$ at three resolutions (a) $N = 2$, (b) $N = 4$, and (c) $N = 8$. (d) shows the complete partition on the polar base face with N=8.

detail mathematical proof. Since the isocube composes of six faces with each face containing the same amount of samples, it fits naturally in the classical cubemap structure.

We now compare the sampling uniformity of the isocube to that of the cubemap. Suppose that one sample is positioned at the center of each element. The left column in Figure 5 shows the sampling patterns of the cubemap and isocube respectively. At the same face resolution, the isocube distributes samples evenly on the sphere, while the cubemap places more samples at the face corners. The uniformities of the two sampling patterns are quantitatively measured using the generalized discrepancy metric [13]. The discrepancy of the isocube is much lower (more uniformly distributed) than that of the cubemap. The detailed discrepancy analysis among the isocube, cubemap, HEALPix and other common spherical mappings can be found in Appendix B.

The middle column in Figure 5 visualizes how the cubemap (upper) is distorted on the isocube by mapping the cubemap subdivision lines onto the isocube map (lower). The central regions of the cube faces are stretched on the isocube map, while the corner regions get suppressed. Consequently, the isocube places samples more evenly on the sphere than that of cubemap.

The trade-off and drawback of isocube is the anisotropic distortion of the partitioned cells, especially in the polar regions. As evidenced in Figure 4, the rectangular texels (in texture domain) at the corners of isocube polar faces are mapped to triangular cells on the spherical surface. Such distortion may cause aliasing especially when the texture is magnified. In Section IV, an anisotropic filtering is proposed to attenuate such artifact.

B. Isocube Environment Mapping

As the isocube fits nicely into the six-face cube structure, it can be naturally loaded into the cubemap hardware. To further exploit the cubemap hardware, we need to convert the reflection vector $R = [x, y, z]^{\mathrm{T}}$ to the texture look-up vector $Q = [s, t, q]^{\mathrm{T}}$ that satisfies the requirements of the hardware cubemap operations. With Q, the isocube can "steal" the hardware cubemap operators such as the texture look-up and anti-aliasing operations.

To map $R \rightarrow Q$, we introduce an intermediate index I for the isocube map and decompose the mapping into two successive steps *conceptually*. The first step computes the index I of the pixel (in the isocube map) in which the reflection vector $R = [x, y, z]^{\mathrm{T}}$ points to. The second step converts the index I to the texture look-up vector $Q = [s, t, q]^{\mathrm{T}}$ fed to the hardware cubemap operations, that is

$$[x, y, z]^{\mathrm{T}} \rightarrow I \rightarrow [s, t, q]^{\mathrm{T}}. \tag{5}$$

The notation I refers to an indexing scheme of samples on the isocube. It composes of a tuple $[c, o]^{\mathrm{T}}$. As shown in

Figure 6, the samples on the isocube map form a set of rings (indicated as dashed circles) parallel to the equator. We define c to be the index of the ring, where the current pixel lies on, counting from the north pole. We then define o as the index of the pixel counting from the sample located at/near to $\phi = 0$ on the ring c, where $[\theta, \phi]^{\mathrm{T}}$ are spherical coordinates of R. Given a reflection vector R, $[c, o]^{\mathrm{T}}$ can be computed efficiently according to Equations (1) - (4).

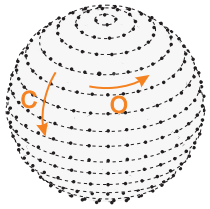


Fig. 6. Indexing the samples on the isocube.

We further extend the index I to the homogeneous form $I = [c, o, 1]^{\mathrm{T}}$. The mapping $I \rightarrow Q$ can be written in the matrix form,

$$\begin{bmatrix} s \\ t \\ q \end{bmatrix} = \mathbf{A} \begin{bmatrix} c \\ o \\ 1 \end{bmatrix}, \tag{6}$$

where

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & 0 & a_{23} \\ a_{31} & a_{32} & 0 \end{bmatrix}. \tag{7}$$

The values of elements in $\mathbf{A}$ are listed in Table I. The experimental results (refer to Figure 9 in Section V) show that the isocube map samples the environment more evenly.

TABLE I
VALUES OF ELEMENTS IN $\mathbf{A}$

	a_{11}	a_{12}	a_{31}	a_{32}
$\phi \in [\ 0,\ \frac{1}{2}\pi\ )$	1	0	0	2
$\phi \in [\ \frac{1}{2}\pi,\ \pi\ )$	2	-2	1	0
$\phi \in [\ \pi,\ \frac{3}{2}\pi\ )$	-1	0	4	-2
$\phi \in [\ \frac{3}{2}\pi,\ 2\pi\ )$	-6	2	-1	0
	a_{21}	a_{23}		
$z \in (\ \frac{2}{3},\ 1\ ]$	0	1		
$z \in [-\frac{2}{3},\ \frac{2}{3}\ ]$	-1	1		
$z \in [-1, -\frac{2}{3}\ )$	0	-1		

IV. ANISOTROPIC FILTERING FOR TEXTURE MAGNIFICATION

Although the isocube samples the environment more evenly than the cubemap, the isocube-based environment mapping, like other texture mapping techniques [14], also suffers from aliasing problems. Figure 7 shows such an example. The aliasing problems appear when the texture is over-minified or over-magnified, as shown in Figures 7(b) and (d), which are rendered by simple bilinear interpolation. There are several anti-aliasing techniques proposed to counteract the aliasing due to texture minification [15], [16], [17], [18], [19]. Among them, mip-mapping [20] or anisotropic filtering are usually realized on graphics hardwares due to their computational efficiency. Figure 7(c) shows the anti-aliased result using hardware anisotropic filtering. On the other hand, although effort has been made to counteract artifacts due to texture magnification [21], [22], [23], [24], existing techniques are seldom hardware-friendly except [25]. No customized hardware features are available to counteract this kind of aliasing. Note that hardware anisotropic filtering has only a minimal effect on jaggy edges caused by texture magnification (Figure 7(e)). In this paper, we propose an efficient and hardware-friendly filtering technique for counteracting the aliasing due to texture magnification.

As the artifacts are usually more apparent at the edge areas than at the non-edge areas (as evidenced by Figures 7(d) and (e)). The artifacts occur as the result of naively magnifying the rectilinear pixels, without considering the actual content in the texture. Intuitively speaking, we smooth ("smear") the pixels along a dominant direction so as to attenuate the objectionable visual artifacts. In particular, we take the edge direction as the dominant direction (Figure 8). Considering a texel p in the edge region, an elliptical filter with the major axis aligned with the edge direction is constructed to control the filtering. The proposed "smearing" method is efficient and hardware-friendly by naturally extending the usage of hardware anisotropic filtering.

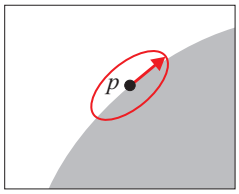


Fig. 8. Anisotropic filtering for texture magnification. The elliptical filter is enforced to align with the edge direction.

Let us start by the case of 2D texture mapping. Suppose G is the normalized gradient vector at the texel p. The edge direction E can be approximated by the vector $[-G.y, G.x]^{\mathrm{T}}$ which is perpendicular to the gradient vector. In Cg shading language [26], the command to perform non-projective texture lookup with derivatives is,

$$\texttt{tex2D(sampler2D } tex, \\ \texttt{float2 } p, \texttt{float2 } dpdx, \texttt{float2 } dpdy),$$

where tex is a 2D texture map, p is the texture coordinate, and the derivatives $dpdx$ and $dpdy$ determine an anisotropic filtering footprint. We then define a footprint along the edge direction E, that is

$$\texttt{tex2D(sampler2D } tex, \\ \texttt{float2 } p, \texttt{float2 } \Delta_x, \texttt{float2 } 0),$$

where,

$$\Delta_x = \xi \cdot \begin{bmatrix} -G.y, \\ G.x \end{bmatrix}.$$

The parameter ξ controls how large the footprint spans. It is related to the distance to the viewpoint. When the object comes close to the viewpoint, a large footprint is expected to compensate more severe aliasing artifacts due to a large texture magnification ratio. Rather than computing the distance, we use level-of-detail [11], and define ξ as,

$$\xi = a \cdot \lambda^b, \tag{8}$$

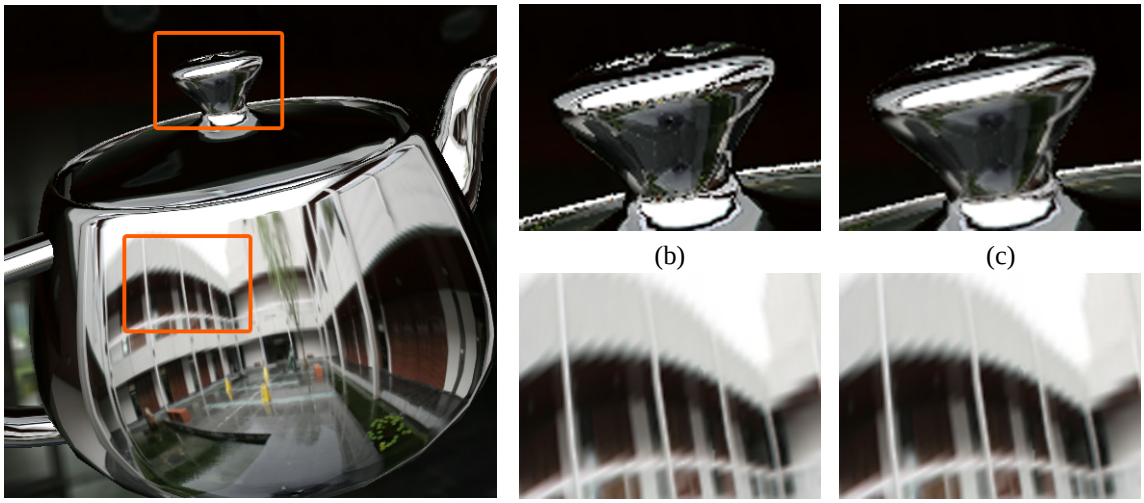


Fig. 7. Aliasing artifacts due to texture minification (the right upper row) and in magnification (the right bottom row): (b) and (d) rendered by simple bilinear interpolation; (a), (c) and (e) are rendered with bilinear interpolation, mipmapping and hardware anisotropic filtering enabled.

where

$$\lambda = -\log2(\max(|dpdx|, |dpdy|)). \quad (9)$$

a and b are two user-defined parameters. Parameter a has a positive value. The larger a is, the more blurry the rendered image will be. Parameter b suppresses or stretches the range of λ.

The proposed anisotropic filtering does not limit to 2D texture mapping. We now describe the steps on how to apply it to the isocube environment mapping:

1) Compute the gradient fields of the isocube environment map.
2) Bilinearly interpolate the gradients during run-time.
3) Compute the derivative vector Δ_x for the texture lookup vector Q.
4) Look-up the environment map by

```
texCUBE(samplerCUBE tex,
        float3 Q,float3 Δx,float3 0).
```

Note that `texCUBE` requires a $3D$ derivative vector. We have to convert the 2D edge direction E to an appropriate 3D derivative vector Δ_x. For instance, if the lookup vector Q points to the positive-x or negative-x isocube face, we compute Δ_x as,

$$\Delta_x = \xi \cdot (-G.y, \ \ 0, \ \ \text{sign}(Q.x)\ G.x).$$

The similar conversions hold for other lookup vectors. Readers are referred to the experimental results in Figure 12 in the next section.

Since both texture magnification and minification may happen in the same image and our anisotropic filtering only works for magnification, we selectively perform the proposed filtering in our shader only when texture magnification is detected. The detection is based on the LOD value. If texture minification is detected instead, the shader utilizes the hardware anti-aliasing functionality by explicitly calling texCUBE(tex, Q) in the shader. Obviously, this shader implementation of anti-aliasing has an overhead in comparing to fully automatic hardware anti-aliasing.

V. Experimental Results and Discussion

In this section, we first evaluate the proposed isocube-based environment mapping, then validate the extended anisotropic filtering. Finally, we demonstrate a dynamic reflection approximation using the isocube.

A. Isocube vs. Cubemap

To evaluate the proposed isocube, we first visually compare it with the classical cubemap. The isocube maps and cubemaps used in the experiments have the same resolution of $128 \times 128 \times 6$. We generate both of them by resampling a high-resolution environment cubemap ($768 \times 768 \times 6$). Figure 9 compares the rendering results (512×512) of the isocube and cubemap environment mapping. The blow-ups in Figures 9(b)-(e) demonstrate that the isocube samples the environment more balanced than the cubemap. In this specific example, the three vertical lines at the center of Figure 9(d) and the poster hanged on the wall in Figure 9(b) are better preserved in the isocube than that in the cubemap. Figure 9(f) shows one more example. In comparison, the grid structure reflected from the spacecraft in Figure 9(g) is better preserved in the isocube.

Besides the visual comparison, we also measured the image quality in terms of PSNR. In the experiment, we first generate 50 random orientations between the object and the viewpoint. Then for each orientation, we render an image of size 1600×1600 and measure the PSNR. The control images are the ones rendered with a cubemap of resolution $768 \times 768 \times 6$.

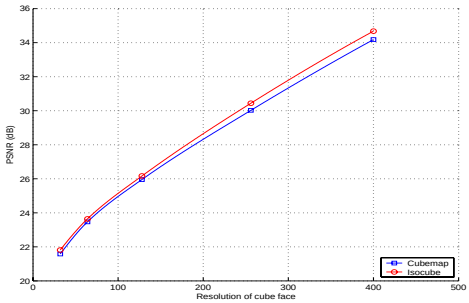


Fig. 10. PSNR achieved by isocube maps and cubemaps against the resolution of cube face.

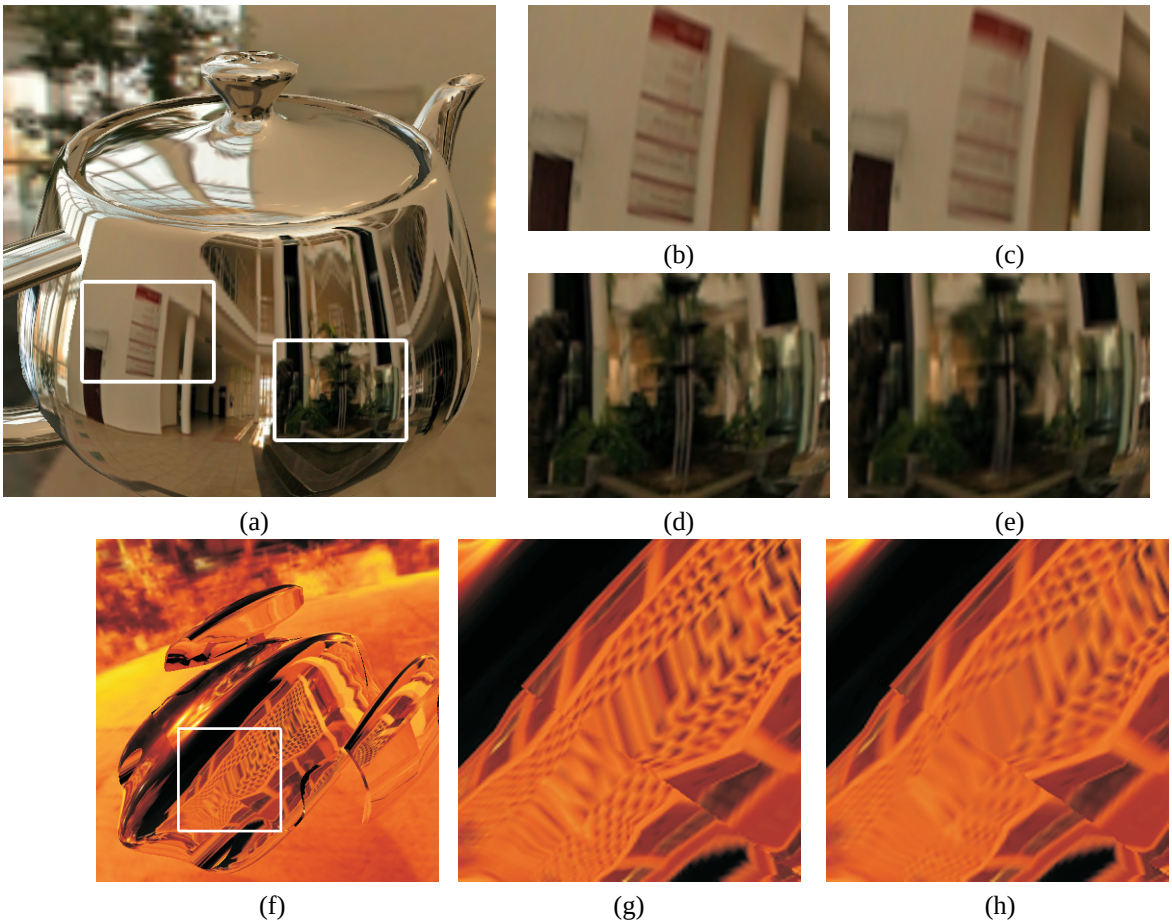


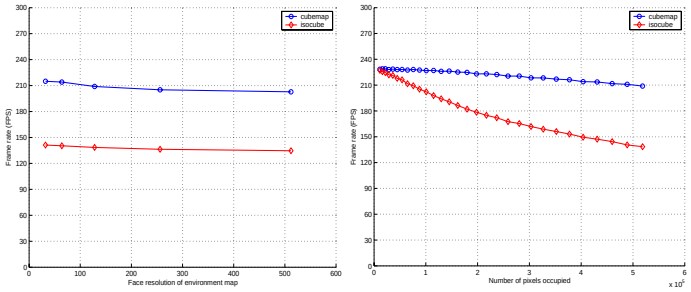
Fig. 9. Environment mapping with the cubemap and isocube: (a) and (f) are the rendered images in a size of 512×512; (b), (d), and (g) are blowups of the isocube environment mapping; (c), (e), and (h) are blowups of the cubemap environment mapping. Note that the isocube maps sample the environment more evenly than the cubemaps. (Environment map (a) courtesy of Tomáš Feltl)

Only foreground pixels (occupied by the environment-mapped object) are counted in the PSNR measurement because the background pixels must be always the same. To minimize the interference due to anti-aliasing, we disable most anti-aliasing functionalities (including mipmapping and anisotropic filtering introduced in previous section) and enable only the bilinear interpolation in the experiment. Figure 10 plots the average PSNR (over 50 images) of both isocube and cubemap against the face resolution. Since the control images are generated from a cubemap of $768 \times 768 \times 6$, we choose to compare the PSNR up to the resolution of $400 \times 400 \times 6$ as the isocube is resampled from that $768 \times 768 \times 6$ cubemap. In this particular experiment, the PSNR of isocube is slightly and consistently better than that of cubemap. Nevertheless, we have to emphasize that the major advantage of isocube is the balance instead of quality.

Despite of the two-step decomposition, the actual computational cost of the complete mapping $R \rightarrow Q$ is rather low. The demonstrative shader code in Cg shading language can be found in Appendix C. This small code already contains *all* necessary computation in our mapping. Although it has been further optimized for speed, only the unoptimized version is listed in Appendix C for clarity.

To measure the speed performance of isocube (optimized version of shader) in comparing to the hardware cubemap, we set up two tests. For both tests, we rendered an environment-mapped object with 106,466 vertices. The mip-map anti-aliasing is enabled. In the first test, we measure the frame rate against the resolution of environment texture. In the second test, we measure the frame rate against the number of visible pixels occupied by the object, in other words, the number of execution of fragment shader. The experiment is achieved by increasing the screen resolution. All experiments are conducted on a Pentium IV 2.6 GHz CPU installed with nVidia GeForceFX 6800 Ultra. Figure 11(a) plots the frame rates of isocube and cubemap when the resolution of texture increases. Obviously, the shader implementation of isocube ($\sim$140 fps) is not as fast as the hardware cubemap ($\sim$210 fps). The increase of texture resolution only *slightly* decreases the frame rate. Both maps drop similarly. In Figure 11(b), we compare how the frame rates drop when the number of visible pixels increases. The curve of the isocube drops significantly while that of the cubemap drops steadily. This significant drop of isocube is mainly due to the computation overhead of its shader implementation.

Compared to cubemaps, the acquisition or generation of isocube map is less convenient than that of the cubemap. Unlike the cubemap, each face of isocube is not a perspective image. Hence it cannot be obtained by rendering as in the cubemap generation. Similar to the spherical map and the dual paraboloid map, the isocube map is generated by resampling a higher resolution cubemap. For static environment, this may



(a) (b)

Fig. 11. Speed performance against (a) the texture resolution; and (b) the number of pixels rendered.

not be a concern. However, it inevitably increases the computational cost when the environment (hence the environment map) is dynamic.

B. Anisotropic Filtering for Texture Magnification

Figure 12 compares the results of isocube environment mapping with (right column) and without the proposed anisotropic filtering (middle column). It can be seen that artifacts are attenuated when using the proposed anisotropic filtering. Figures 12(a) and (f) are rendered with $a = 0.024$ and $b = 0.18$. In addition, a frame rate of 164.4 fps is obtained in rendering these images. By filtering along the edge direction, our approach may not retain the sharpness of the edges. The filtered results look a bit blurry compared to the bilinearly interpolated results. Sharp texture magnification is reported by preventing interpolation across the boundaries defined in silhouette maps [25].

Our approach as discussed above depends on the gradient field of the texture map. The advantages are that it extends the existing hardware anisotropic filtering and it is efficient in computation. We can further improve storage efficiency by storing the gradient angle instead of the 2D gradient vectors. That is to store gradient angles in the α-channel of the isocube texture map. However, our approach may not work well for an extreme magnification. Since the current implementation interpolates the gradient map by hardware bilinear interpolation, and artifacts are unavoidable in the magnified gradient map. This problem could be fixed by using a user-defined vector edge map [25].

C. Glossy Reflection Approximation

As one more application, we now demonstrate how the isocube is applied to approximate glossy reflection with prefiltering techniques [27], [28], [29]. For simplicity, we assume the filtering kernel is a box filter. Since the mipmap textures are generated by box-filtering the original texture map, we could approximate the box-filtering results by looking-up the appropriate mipmap texture.

Suppose that the lookup vector Q points to the positive-x or negative-x isocube face, we do the approximation by,

$$\texttt{texCUBE(samplerCUBE } tex, \texttt{ float3 } Q, \texttt{float3 } \Delta_x, \texttt{float3 } \Delta_y),$$

where,

$$\Delta_x = \text{float3}(0,\ \ 0,\ \ \eta),$$
$$\Delta_y = \text{float3}(0,\ \ 0,\ \ \eta).$$

The parameter η controls the filter size. The larger the η value is, the more blurry the rendering result becomes. Figure 13 shows the approximation results of glossy reflection with isocube environment maps.

VI. Conclusion

This paper proposes a novel six-face spherical map, isocube, which not only fully utilizes the built-in cubemap hardware but also samples the environment evenly. The computational cost of the isocube mapping is low. By exploiting the hardware cubemap operators, the isocube achieves a very high frame rate. Although the isocube is currently implemented as a shader (hence, not as fast as the hardware cubemap), we believe its simplicity will encourage its future hardware implementation. In addition, we propose an anisotropic filtering to reduce aliasing artifacts due to texture magnification. The proposed filtering accounts for both the texture gradient and the distance to the viewpoint. It can be applied to our proposed isocube as well as other texture mapping techniques.

All cubemap hardwares, however, have the limitation which creates undesirable discontinuities along the cube face edges. Like the cubemap, the isocube map suffers from the seam problem. Users may utilize CubeMapGen [30], a preprocessing tool that generates a seam-free environment map or the mipmap chain, to alleviate this problem.

Besides environment mapping, isocube may benefit other applications requiring uniform representation of the environment, such as precomputed radiance transfer. It could also be applied to represent the view in the omnidirectional video, like iMAX.

Acknowledgments

We would like to thank Tomáš Feltl for granting us the use of "dole" environment map. We also thank Guangyu Wang for his help in capturing environment maps. This project is supported by the Research Grants Council of the Hong Kong Special Administrative Region, under RGC Earmarked Grants (Project No. CUHK417005 and CityU115606).

Appendix A: Equal Area Property

We prove the equal area property of the isocube using differential geometry. To do so, we first show the isocube mapping (from sphere to plane) is equiareal. Hence, by showing the mapped elements on plane are equal in area, we can prove the elements on sphere are also equal in area. We now review some basic theories of mappings in differential geometry [31]. Suppose that a curve on a surface $S \subset \Re^3$ has the parametric representation

$$\mathbf{x}(u_1, u_2) = (x_1(u_1, u_2), x_2(u_1, u_2), x_3(u_1, u_2)).$$

The first fundamental form of surface S is

$$ds^2 = \frac{\partial \mathbf{x}}{\partial u_1} \cdot \frac{\partial \mathbf{x}}{\partial u_1}(du_1)^2 + 2\frac{\partial \mathbf{x}}{\partial u_1} \cdot \frac{\partial \mathbf{x}}{\partial u_2} du_1 du_2 + \frac{\partial \mathbf{x}}{\partial u_2} \cdot \frac{\partial \mathbf{x}}{\partial u_2}(du_2)^2.$$

This quadratic form enables measurement of lengths of curves, angles, and areas on surface S. Denote its coefficients as,

$$E = \frac{\partial \mathbf{x}}{\partial u_1} \cdot \frac{\partial \mathbf{x}}{\partial u_1}, \quad F = \frac{\partial \mathbf{x}}{\partial u_1} \cdot \frac{\partial \mathbf{x}}{\partial u_2}, \quad G = \frac{\partial \mathbf{x}}{\partial u_2} \cdot \frac{\partial \mathbf{x}}{\partial u_2}.$$

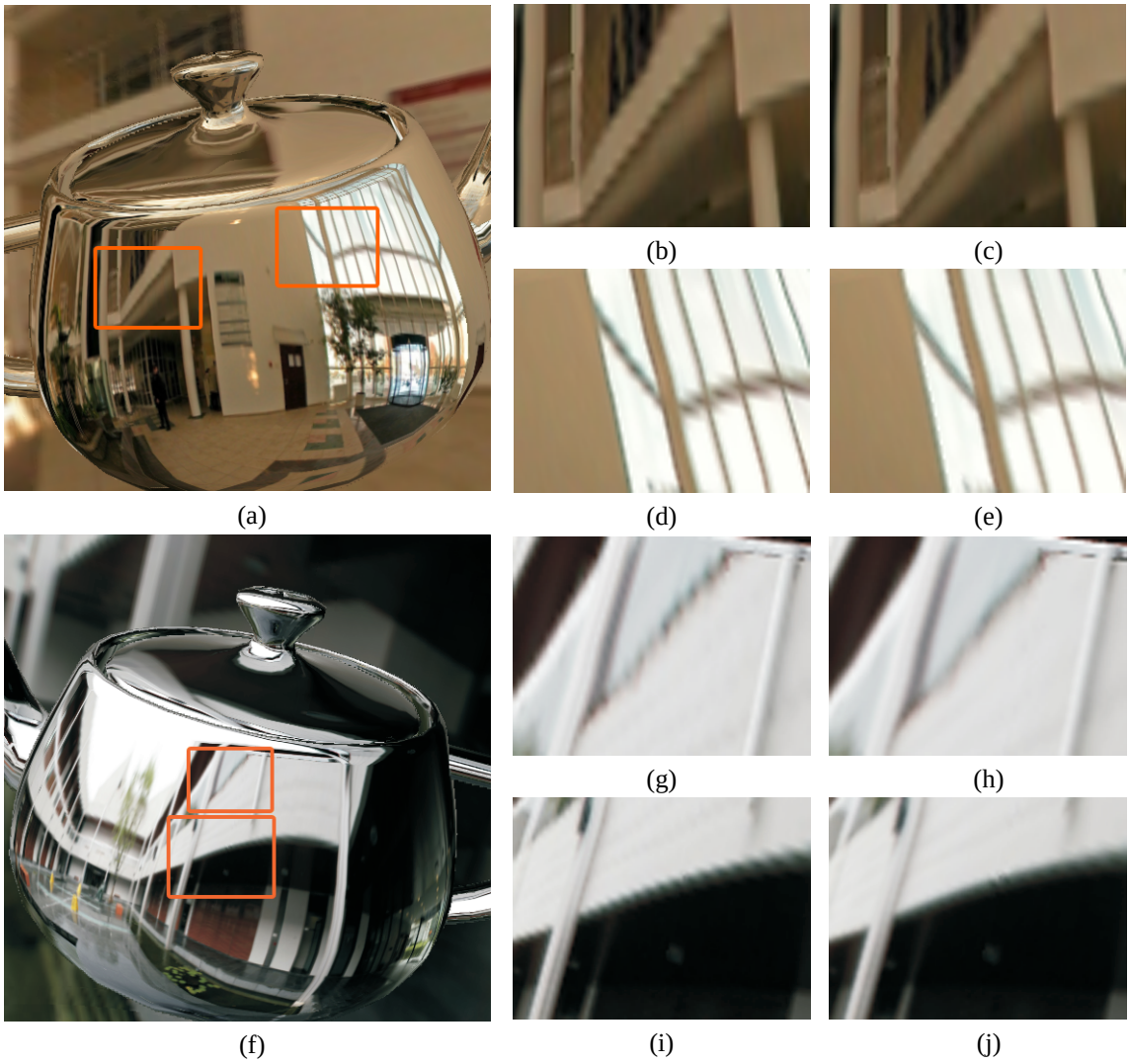


Fig. 12. Comparison between the isocube environment mapping (the middle column) and that with the proposed anisotropic filtering applied (the right column). The jaggy edges shown in blowups (b), (d), (g) and (i) have been smoothed in (c), (e), (h), and (j). (Environment map (a) courtesy of Tomáš Feltl)

Fig. 13. Example screenshots of glossy environment reflection.

Then the discriminant of the first fundamental form is defined as

$$g = EF - G^2.$$

Suppose now that S is spherical surface and S^* is the plane, and that both are parameterized in the same coordinates (u_1, u_2). An allowable mapping of surface S to S^* is called equiareal if every part of S is mapped onto a part of S^* which has the same area. The necessary and sufficient conditions of a mapping to be equiareal are as follows:

Theorem 1: An allowable mapping from surface S to S^* is equiareal if and only if the discriminants of the first fundamental forms of S and S^* are equal.

The proof can be found in differential geometry textbooks [31]. With the above theorem, we shall show that the isocube mapping maintains the equal solid-angle property. Suppose the spherical surface S has the following parametric form,

$$\begin{aligned} \mathbf{x} &= (x(\phi,\theta), y(\phi,\theta), z(\phi,\theta)) \\ &= (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta). \end{aligned}$$

Let e, f, g be the coefficients for the first fundamental form of S. The plane S^* has the form,

$$\mathbf{y} = (u(\phi,\theta), v(\phi,\theta)).$$

Let E, F, G be the coefficients for the first fundamental form

of S^*. We have the area ratio

$$\begin{aligned} k &= \sqrt{\frac{EG-F^2}{eg-f^2}} \\ &= \frac{1}{\sin\theta}\sqrt{EG-F^2}. \end{aligned} \quad (10)$$

If k is a constant, the mapping maps the regions of surface S with equal area to the regions of S^* with equal area in which the two are related by a constant. In particular, when $k=1$ the mapping preserves the same area for the corresponding regions in S and S^*.

We now transform the isocube partitioning equations into a mapping representation. To do so, we adopt a different parametrization (f,u,v) for isocube, where $f \in [0,\ldots,5]$ is the base face index, and $(u,v) \in [0,1]^2$ are the local coordinates in the face f. Given one equatorial base face, we have the mapping according to (Equations (1) and (2)),

$$\begin{aligned} u &= \frac{2}{\pi}\,\phi + 0.5, \\ v &= \frac{3}{4}\,\cos\theta + 0.5. \end{aligned}$$

Substituting them into the formula Equation (10),

$$\begin{aligned} k &= \frac{1}{\sin\theta}\sqrt{EG-F^2} \\ &= \frac{1}{\sin\theta}\sqrt{(\frac{2}{\pi})^2\cdot(\frac{3}{4}\sin\theta)^2 - 0} \\ &= \frac{3}{2\pi}. \end{aligned}$$

Similarly, for the north polar region we have the mapping according to Equations (3) and (4),

$$\begin{aligned} u &= \frac{1}{2}\sqrt{3-3\cos\theta} + 0.5, \\ v &= \frac{2}{\pi}\phi\sqrt{3-3\cos\theta} + 0.5. \end{aligned}$$

The coefficients of the first fundamental form are

$$\begin{aligned} E &= (\frac{2}{\pi}\sqrt{3-3\cos\theta})^2, \\ F &= -\frac{6\phi}{\pi^2}\sin\theta, \\ G &= \left[(\frac{3}{4})^2 + (\frac{3\phi}{\pi})^2\right]\cdot(\frac{\sin\theta}{\sqrt{3-3\cos\theta}})^2. \end{aligned}$$

Substituting them into Equation (10),

$$k = \frac{1}{\sin\theta}\sqrt{EG-F^2} = \frac{3}{2\pi}.$$

The above derivation holds for the rest of isocube regions. As the ratio is always constant, the isocube mapping maps the regions on the spherical surface with equal area to regions on the plane with equal area. In other words, isocube maintains the equal solid-angle property.

APPENDIX B: DISCREPANCY ANALYSIS

Although the equal solid-angle property guarantees all samples to be equally important, it is worth studying the uniformity of the isocube sampling pattern. Here the sampling pattern is formed by putting a sample at the center of each element. A quantitative metric, generalized discrepancy [13], is adopted for this purpose. It is defined as,

$$D(N) = \frac{1}{2\sqrt{\pi}N}\left[\sum_{i,j=1}^{N}\left(1 - 2\ln\left(1+\sqrt{\frac{1-\vec{\eta}_i\cdot\vec{\eta}_j}{2}}\right)\right)\right]^{\frac{1}{2}} \quad (11)$$

where $\{\vec{\eta}_1,\ldots,\vec{\eta}_N\}$ is a N-point sequence and $\vec{\eta}_i$ is a point on the sphere. The lower the D is, the more uniformly distributed the sampling pattern is.

Figure 14 shows the discrepancy curves of the isocube and three other parameterizations discussed in this paper. This graph plots the discrepancies against the number of samples (N). As the resolution (number of samples) increases, all parameterizations have improved uniformities. Among the four methods, the cubemap has much higher discrepancies. The discrepancy curve of the isocube almost coincides with that of HEALPix, and they both are slightly better than the elevated concentric map.

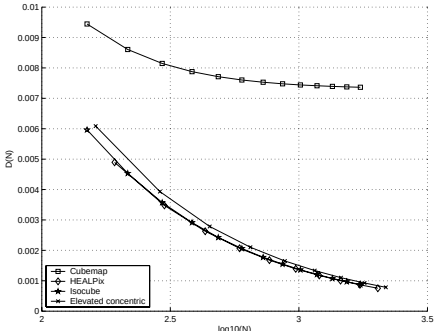


Fig. 14. Discrepancy of various sampling patterns.

APPENDIX C: CG PROGRAM OF ISOCUBE MAPPING

The fragment shader below computes the texture lookup vector $Q = [s,t,q]^{\mathrm{T}}$ for the input reflection vector $R = [x,y,z]^{\mathrm{T}}$. To save the computational cost, we store the values of the upper four elements in Table I into a lookup table **signTBL**.

```
01  float3 R2Q( float3 R )
02  {
03    float2 I;
04    float3 Q;
05    float4 coef;
06    float  phi, y, ya, bequ, quar;
07
08    // compute azimuth angle and convert it in the range [0,4)
09    phi  = 2*atan2(R.z, R.x)/PI;
10    phi += step(phi, -0.5) * 4;
11
12    // decide whether the pixel is in the equatorial region
13    y    = R.y * 1.5;
14    ya   = abs(y);
15    bequ = step(ya, 1.);
16
17    // convert R → I
18    I.x  = sqrt(3 - 2*ya);
19    I.x  = lerp(I.x, 1, bequ);
20    I.y  = phi * I.x;
21
22    // map I → Q
23    quar = floor(phi + 0.5);
24    coef = texRECT(signTBL, float2(quar, 0));
25    Q.x  = dot(coef.xy, I);
26    Q.y  = lerp(sign(y), y, bequ);
27    Q.z  = dot(coef.zw, I);
28
29    return Q;
30  }
```

REFERENCES

[1] N. Greene, "Environment mapping and other applications of world projections," *IEEE Computer Graphics and Applications*, vol. 6, no. 11, November 1986.

[2] W. Heidrich and H.-P. Seidel, "View-independent environment maps," in *Proceedings of Graphics hardware*, 1998, pp. 39–45.

[3] P. Shirley and K. Chiu, "A low distortion map between disk and square," *J. Graph. Tools*, vol. 2, no. 3, pp. 45–52, 1997.

[4] K. M. Gorski, B. D. Wandelt, E. Hivon, F. K. Hansen, and A. J. Banday, "The HEALPix primer," Theoretical Astrophysics Center (TAC) Kopenhagen, Tech. Rep., Feb. 2003, astro-ph/9905275.

[5] R. Ng, R. Ramamoorthi, and P. Hanrahan, "All-frequency shadows using non-linear wavelet lighting approximation," in *SIGGRAPH'04*, 2004, pp. 376–381.

[6] J. L. Mitchell, "Motion blurring environment maps," in *ShaderX*^{4}, W. Engel, Ed. Charles River Media, 2006, pp. 263–268.

[7] D. Voorhies and J. Foran, "Reflection vector shading hardware," in *Siggraph '94*, 1994, pp. 163–166.

[8] T. Fortes, "Tetrahedron environment maps," Master's thesis, Chalmers University of Technology, Gothenburg, Sweden, 2000.

[9] I. Takashi, "Dynamic global illumination using tetrahedron environment mapping," in *ShaderX*^{4}, W. Engel, Ed. Charles River Media, 2006, pp. 157–169.

[10] E. Praun and H. Hoppe, "Spherical parametrization and remeshing," *Siggraph 2003*, vol. 22, no. 3, pp. 340–349, 2003.

[11] OpenGL_ARB, *OpenGL Specification, Version 1.1*, 1995.

[12] L. Wan, T.-T. Wong, and C.-S. Leung, "Spherical Q^2-tree for sampling dynamic environment sequences," in *Proceedings of Eurographics Symposium on Rendering '05*, 2005, pp. 21–30.

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

[14] P. S. Heckbert, "Survey of texture mapping," *IEEE Comput. Graph. Appl.*, vol. 6, no. 11, pp. 56–67, 1986.

[15] F. C. Crow, "Summed-area tables for texture mapping," in *SIGGRAPH '84*, 1984, pp. 207–212.

[16] N. Greene and P. Heckbert, "Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter," *IEEE Computer Graphics and Applications*, vol. 6, no. 6, pp. 21–27, 1986.

[17] J. McCormack, R. Perry, K. I. Farkas, and N. P. Jouppi, "Feline: Fast elliptical lines for anisotropic texture mapping," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 1999, pp. 243–250.

[18] L. Wang, S. Kang, H.-Y. Shum, and R. Szeliski, "Optimal texture map reconstruction from multiple views," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2001, pp. 347–354.

[19] K. Meinds and B. Barenbrug, "Resample hardware for 3d graphics," in *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, 2002, pp. 17–26.

[20] L. Williams, "Pyramidal parametrics," in *Siggraph 83*, 1983, pp. 1–11.

[21] J. Allebach and P. W. Wong, "Edge-directed interpolation," in *Proceedings of IEEE International Conference on Image Processing*, 1996, pp. 707–710.

[22] T. M. Lehmann, C. Gönner, and K. Spitzer, "Survey: Interpolation methods in medical image processing," *IEEE Transactions on Medical Imaging*, vol. 18, no. 11, pp. 1049–1075, 1999.

[23] X. Li and M. Orchard, "New edge-directed interpolation," *IEEE Transactions on Image Processing*, vol. 10, no. 10, pp. 1521–1527, 2001.

[24] D. Muresan and T. Parks, "Adaptively quadratic (AQua) image interpolation," *IEEE Transactions on Image Processing*, vol. 13, no. 5, pp. 690–698, 2004.

[25] P. Sen, "Silhouette maps for improved texture magnification," in *Proceedings of Graphics Hardware*, 2004.

[26] W. R. Mark, R. S. Glanville, K. Akeley, and M. J. Kilgard, "Cg: a system for programming graphics hardware in a C-like language," *ACM Transaction on Graphics*, vol. 22, no. 3, pp. 896–907, 2003.

[27] J. Kautz and M. D. McCool, "Approximation of glossy reflection with prefiltered environment maps," in *Proceedings of Graphics Interface*, 2000, pp. 119–126.

[28] M. Ashikhmin and A. Ghosh, "Simple blurry reflections with environment maps," *Journal of Graphics Tools*, vol. 7, no. 4, pp. 3–8, 2002.

[29] J. Kautz, K. Daubert, and H.-P. Seidel, "Advanced environment mapping in VR applications," *Computers & Graphics*, vol. 28, no. 1, pp. 99–104, 2004.

[30] J. Isidoro, "Filtering cubemaps: angular extent filtering and edge seam fixup methods," in *Siggraph sketch*, 2005. [Online]. Available: www.ati.com/developer/SIGGRAPH05/ Isidoro-CubeMapFiltering.pdf

[31] E. Kreyszig, *Differential Geometry*. University of Toronto Press, 1959.

Liang Wan received the B.Eng. and M.Eng. degrees in computer science and engineering from Northwestern Polytechnical University, China, in 2000 and 2003 respectively. She is currently a PhD candidate in computer science at the Chinese University of Hong Kong. Her research interests include computer graphics, precomputed lighting, image-based rendering, and image processing.

Tien-Tsin Wong received the B.Sci., M.Phil., and PhD. degrees in computer science from the Chinese University of Hong Kong in 1992, 1994, and 1998, respectively. Currently, he is a Professor in the Department of Computer Science & Engineering, Chinese University of Hong Kong. His main research interest is computer graphics, including image-based rendering, natural phenomena modeling, and multimedia data compression. He received *IEEE Transactions on Multimedia Prize Paper Award 2005* and *Young Researcher Award 2004*.

Chi-Sing Leung received the B.Sci. degree in electronics, the M.Phil. degree in information engineering, and the PhD. degree in computer science from the Chinese University of Hong Kong in 1989, 1991, and 1995, respectively. He is currently an Associate Professor in the Department of Electronic Engineering, City University of Hong Kong. His research interests include neural computing, data mining, and computer graphics. In 2005, he received the 2005 IEEE Transactions on Multimedia Prize Paper Award for his paper titled, "The Plenoptic Illumination Function" published in 2002.