A Full-View Spherical Image Format

Shigang Li and Ying Hai
Graduate School of Engineering, Tottori University, Japan
{li@ele.tottori-u.ac.jp}

Abstract

This paper proposes a full-view spherical image format which is based on the geodesic division of a sphere. In comparison with the conventional 3D array representation which consists of five parallelograms, the proposed spherical image format is a simple 2D array representation. The algorithms of finding the neighboring pixels given a pixel of a spherical image and mapping between spherical coordinate and spherical image pixel are given also.

1. Introduction

While cameras with full FOV (field of view) are developed for map building [1], visual surveillance [2], mobile robots and vehicles [3], it seems that the problem of how to manage the full-view image data for storage and processing in an efficient way is bypassed. Figure 1 shows a pair of fisheye images which have a wider FOV than a hemisphere, respectively, and a full-view image can be synthesized from these two images. Obviously, there is a lot of redundant data in the captured raw images. The data redundancy is caused by using a physically rectangular image plane to capture a circular image and the overlapping regions among multiple views. This fact implies that managing the captured raw data directly is not efficient. Using a camera cluster to capture a full-view image has the same problem because there must be overlapping regions between the neighboring cameras so that a full-view image can be stitched from the images captured from the cameras with different orientations.

To cope with the above problem we propose a full-view spherical image format which only contains the information of the full FOV without data redundancy and is independent of concrete imaging devices. Captured full-view images are mapped to a spherical image, and the data management is carried out on the mapped spherical image.

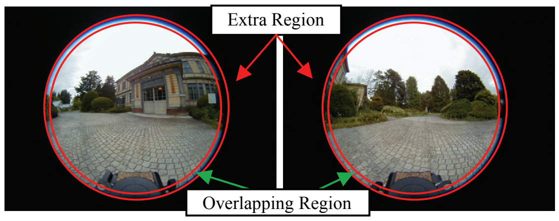

Figure 1 There are extra region and overlapping region in the raw full-view image captured by a fisheye camera.

The proposed spherical image format is based on the geodesic division of a sphere [4], which results that each pixel contains about the same area of view. The proposed spherical image format is a simple 2D array representation. Algorithms of finding the neighboring pixels given a pixel of a spherical image and mapping between spherical coordinate and spherical image pixel are presented also.

This paper is organized as follows. The related research is introduced in section 2. The details of the proposed spherical image format are described in section 3. Finally, we give the conclusions of this paper in section 4.

2. Related research

To manage a spherical image in a digital computer we need to tessellate a sphere. How to tessellate a sphere is a basic topic in computer vision and computer graphics. In computer graphics the art-of-the-state technology of representing a sphere is the cube map [5] which projects a cube onto a sphere [6]. The disadvantage of this method is the unevenness of the resulting tessellations.

2.1. Geodesic division of a sphere

It is known that geodesic-division-based tessellation of a sphere results in approximately uniform cells. Figure 2 shows an icosahedron, 1-level subdivision and 2-level subdivision of the

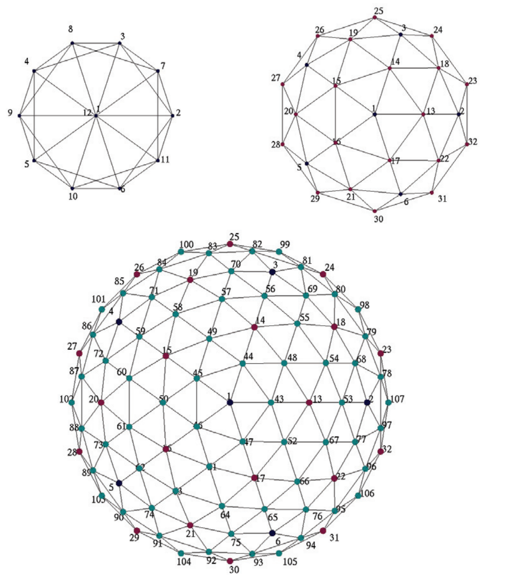
Figure 2. The process of the geodesic division of an icosahedron.

icosahedron; the vertices of the initial icosahedron are indicated in black, the vertices appeared newly during the 1st–level subdivision are indicated in red, and that appeared newly during the 2nd–level subdivision are indicated in green. As shown in the most left figure of Fig. 2, each vertex has six neighboring vertices, except for twelve vertices that have five neighboring vertices. These twelve vertices correspond to that of the original icosahedron. This means that almost all the pixel cells at the vertices are hexagonal in shape. This representation is called a SCVT (Spherical Centroidal Voronoi Tessellation) [7]. In this paper the proposed spherical image format is based on a SCVT image.

Note that the geodesic-division-based triangle cell is the dual of hexagonal cell of the SCVT. The vector of a vertex of a geodesic division sphere corresponds to the principle direction of a pixel of a SCVT image.

2.2. Data structure of SCVT image

Two kinds of data structure have been proposed for representing a SCVT image. One is a hierarchical tree structure [8][9]; however, the drawback of this representation is that the adjacency relationship between neighboring cells is not preserved. The other is a spherical array data structure proposed in [10]; this array makes it easier and more efficient to find the corresponding cell given a view direction, and to find the neighboring cells given a cell on a tessellated sphere. Since [10] is the closest research to our best knowledge, we give more detailed introduction on this spherical array data structure.

In [10], an icosahedron is flattened out and five connected parallelograms are obtained, each

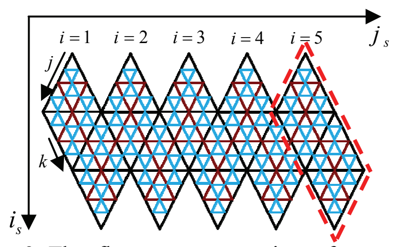

Figure 3 The flatten representation of a geodesic division sphere. One of the parallelograms is indicated as red lines.

parallelogram consisting of four triangular faces. The sketch of the above representation is shown in Figure 3 where one of the parallelograms is indicated as dot red lines. The flattened-out representation is referred to as the spherical array. Since a parallelogram can be represented by a 2D array, the spherical array is represented as five 2D arrays except for the two missing vertices referred to as the zenith and the nadir. Let index i specify a parallelogram, and let j and k specify a vertex within the parallelogram, respectively. The spherical array corresponds to a 3D array indexed by (i,j,k) . In [10] the operation of finding neighboring cells and mapping between spherical coordinate and spherical cell is carried out in this 3D spherical array.

In this paper we propose a method of representing a SCVT image as a rectangular 2D array. The contribution of this paper is as follows.

- A SCVT image is represented as a rectangular 2D array. The proposed representation is understood easily by intuition.
- The simplicity of the data structure results in efficiency of operations in a tessellated sphere.

3. Rectangular 2D spherical image format

3.1. 2D array of a SCVT image

The 2D array of the corresponding tessellated cells of the 2-level subdivision of Fig.2 is shown in Figure 4 by arranging the cells in rows along the longitudinal direction and in column along the latitudinal direction, respectively; cells 1 and 12 correspond to the zenith and the nadir, respectively. The 2D array can be obtained also from the flatten representation of a geodesic division sphere, shown in Fig. 3, by arranging the cells of the five parallelograms along i_s and j_s without considering the shared ones. Note that in Fig. 4 the vertices in the same row do not always have the same latitude degree.

Suppose the number of levels of subdivision is L. The resultant array of the vertices would have $3\cdot 2^L+1$ rows, but the number of vertices in rows is variable from 1 to $5 \cdot 2^L$. Let i be the row number and let N_i be the number of vertices of i th row, as shown in Figure 4. We have the following equation indicating the number of vertices in a row.

$$N_i = \begin{cases} 1, (i = 0) \\ 5i, (0 < i \le 2^L) \\ 5 \times 2^L, (2^L < i \le 2^{L+1}) \\ 5 \times (R_L - i), (2^{L+1} < i < R_L) \\ 1, (i = R_L) \end{cases} \qquad (1)$$

where $R_L = 3 \cdot 2^L$. The above array is called a SCVT array in this paper.

Let the position of a L_v th-level vertex be (i_v, j_v). According to the rule of the geodesic subdivision its new position at a L th-level ($L > L_v$) subdivision geodesic array, (i, j), is determined as follows.

$$i = 2^{(L-L_v)} i_v, j = 2^{(L-L_v)} j_v. \qquad (2)$$

Using (2) we can determine that a vertex appears from which level subdivision.

3.2. Finding neighboring cells

In a SCVT array the adjacency of cells between rows is corrupted apparently although the adjacency of cells in a row is preserved. However, the neighboring cells between rows can be computed simply according to the rule of geodesic division.

The vertices of geodesic division can be classified into three types: the twelve 0-level vertices which has five neighboring vertices (such as vertex 3), ones which have one neighboring cell in one neighboring row and three neighboring cells in the other neighboring row (such as vertex 43), and ones which have two neighboring cells in the two neighboring rows (such as vertex 48), respectively.

Next, we give a detailed explanation on how to find the neighboring cells of a given cell under the subdivision level L. According to the structure of a SCVT array the cells can be classified into three groups, G_A, G_B and G_C, as shown in Fig. 4. In the following formulas, $J_u = N_{i+1}/5 * \lfloor j/(N_i/5) \rfloor + j\%(N_i/5)$ and $J_d = N_{i-1}/5 * \lfloor j/(N_i/5) \rfloor + j\%(N_i/5)$.

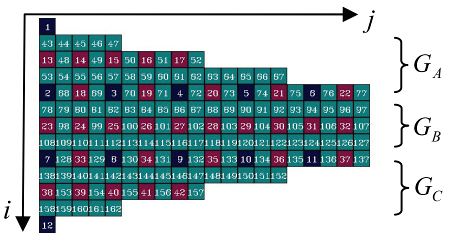

Figure 4. The corresponding 2D array of vertices of the geodesic division sphere.

- For the zenith, $(i = 0)$, and the nadir, $(i = R_L)$, their five neighboring vertices are follows, respectively.
  $(1,0), (1,1), (1,2), (1,3), (1,4)$.
  $(R_L - 1,0), (R_L - 1,1), (R_L - 1,2), (R_L - 1,3), (R_L - 1,4)$.
- For the other 0-level vertices, its five neighboring vertices are follows in terms of (2).
  - If $i = 2^L$ and $j\%2^L = 0$
    $(i-1, N_{i-1}/5 * (j/(N_i/5))), (i, (j-1+N_i)\%N_i), (i, j+1),$
    $(i+1, (j-1+N_{i+1})\%N_{i+1}), (i+1, j)$.
  - If $i = 2^{L+1}$ and $j\%2^L = 0$
    $(i-1, j), (i-1, j+1), (i, (j-1+N_i)\%N_i), (i, j+1),$
    $(i+1, N_{i+1}/5 * (j/(N_i/5)))$.
- For the vertices belonging to group G_A, $(0 < i \le 2^L)$, and being on the edge joining the zenith and one of the 0-level vertices, $j\%(N_i/5) = 0$, the neighboring vertices are follows.
  $(i-1, J_u), (i, (j-1+N_i)\%N_i), (i, j+1),$
  $(i+1, (J_d - 1 + N_{i+1})\%N_{i+1}), (i+1, J_d), (i+1, J_d + 1)$.
  For the vertices which satisfy with $i = 2^L$, $j\%2^L \ne 0$,
  $(i-1, J_u), (i-1, (J_u - 1 + N_{i-1})\%N_{i-1}), (i, (j-1+N_i)\%N_i), (i, (j+1)\%N_i),$
  $(i+1, j), (i+1, (j-1+N_{i+1})\%N_{i+1})$.
  For the other vertices of group G_A, the neighboring vertices are follows.
  $(i-1, J_u), (i-1, (J_u - 1 + N_{i-1})\%N_{i-1}), (i, (j-1+N_i)\%N_i), (i, (j+1)\%N_i),$
  $(i+1, J_d), (i+1, (J_d + 1)\%N_{i+1})$.
- For the vertices belonging to group G_C, $(2^{L+1} \le i < R_L)$, and being on the edge joining the nadir and one of the 0-level vertices, $j\%(N_i/5) = 0$, the neighboring vertices are follows.
  $(i-1, (J_u - 1 + N_{i-1})\%N_{i-1}), (i-1, J_u), (i-1, J_u + 1),$
  $(i, (j-1+N_i)\%N_i), (i, j+1), (i+1, J_d)$.
  For the other vertices of group G_C, the neighboring vertices are follows.
  $(i-1, (J_u + 1)\%N_{i-1}), (i-1, J_u), (i, (j-1+N_i)\%N_i), (i, j+1),$
  $(i+1, J_d), (i+1, (J_d - 1 + N_{i+1})\%N_{i+1})$.
- For the vertices belonging to group G_B, $(2^L < i < 2^{L+1})$, the neighboring vertices are follows.
  $(i-1, (j+1)\%N_{i-1}), (i-1, j), (i, (j-1+N_i)\%N_i), (i, (j+1)\%N_i),$
  $(i+1, j), (i+1, (j-1+N_{i+1})\%N_{i+1})$.

Note that in the above operation "%" means the computation of the remainder. This operation is used to cope with the vertices at the beginning or end of a row which are adjacent to each other at the tessellated sphere.

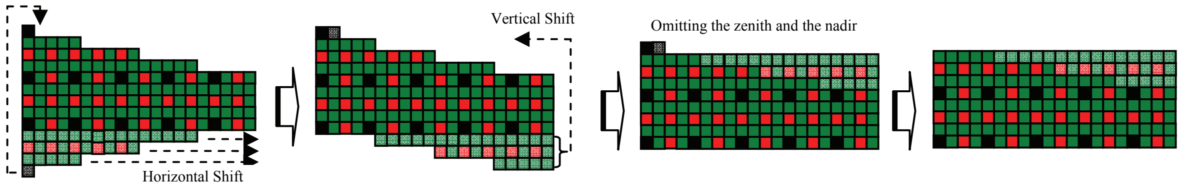

Figure 5 The compact representation is obtained by two operations, a horizontal shift and a vertical shift, and omitting the zenith and nadir cells.

3.3. Mapping between spherical coordinate and SCVT cells

The mapping from a discrete SCVT cell to spherical coordinate can be carried out quickly by using a look up table. On the other hand, since the angular interval between rows and that between the vertices in a row is not the same, we cannot find the corresponding vertex by only using the average of angular interval. This means that the inverse mapping from spherical coordinate to the nearest SCVT cell requires a searching.

In this paper a similar searching method to [10] is used. Given a spherical polar coordinate (θ, ϕ), an initial position is first estimated; then, the corresponding SCVT cell is found by iteratively searching a local maximum among the estimated cell and its neighboring cells.

3.4. Representing a geodesic array as a rectangular 2D array

In a SCVT array shown in Fig. 4 rows may include different number of vertices. However, by the simple operation shown in Figure 5 we can obtain a compact rectangular 2D array which contains all the SCVT cells except for that of the zenith and the nadir. In practice, omitting the zenith and the nadir does not almost give any influence to the information including in a SCVT image by considering an image with hundreds of thousands, or millions pixels.

Note that the algorithms of section 3.3 and 3.4 can be modified simply based on the above rectangular 2D array.

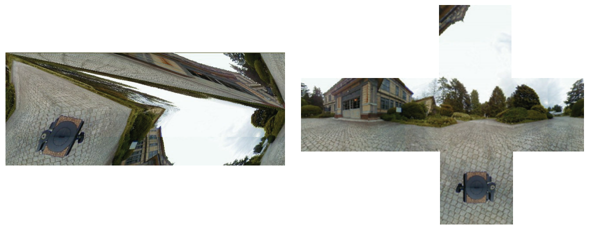

Figure 6 (a) A full-view image. (b) The proposed rectangular array representation of a tessellated sphere. (c) The flatten cubic image generated from the rectangular 2D array.

To test the proposed data structure we carried out an experiment using the full view image of Fig. 1. The converted image represented by the proposed spherical image format is shown in Fig. 6(a). A cube image is generated from the rectangular 2D array of Fig. 6(a) by the method of section 3.3, as shown in Fig. 6(b).

4. Conclusions

This paper proposes a novel full-view spherical image format which is based on the geodesic division of a sphere, i.e., a SCVT image. The proposed representation has the following characteristics.

- It is with approximately uniform cells in contrast with a cube-map representation used in computer graphics.
- It is a compact rectangular 2D array, which may be understood easily by intuition and be implemented simply.

How to use the proposed format for full-view image processing is our future work.

References

[1] S. Coorg and S. Teller. Spherical mosaics with quaternions and dense correlation. In *International Journal of Computer Vision*, vol. 37, pp. 259-273, 2000.

[2] N. D. Jankovic and M. D. Naish, "Developing a modular active spherical vision system", ICRA, 2005.

[3] S. Li, "Monitoring around a vehicle by a spherical image sensor", IEEE Trans. on Intelligent Transportation System, Vol.7, No.4, pp.541-550, 2006.

[4] B.K.P. Horn, Robot Vision, MIT Press, 1986.

[5] http://www.opengl.org

[6] C. Goad, "Special purpose automatic programming for 3D model-based vision", in Proc. DARPA Image Understanding Workshop, pp.94-104, 1983.

[7] Q. Du, M. Gunzburger, and L. Ju, Constrained centroidal Voronoi tessellations on general surfaces, *SIAM J. Sci. Comput.*, 2003.

[8] B. K. P. Horn, "Extended Gaussian image", Proc. IEEE, vol.72, no. 12, pp.1671-1686, 1984.

[9] K. Ikeuchi, "Generating an interpretation tree from a cad model for 3D-object recognition in bin-picking tasks", Int. J. Computer Vision, vol. 1, no. 2, pp.145-165, 1987.

[10] C.H. Chen and A.C. Kak, "A robot vision system for recognizing 3-D objects in low-order polynomial time", IEEE Trans. on Systems, Man, and Cybernetics, vol.19, no.6, pp.1535-1563, 1989.