Proc. VIIth Digital Image Computing: Techniques and Applications, Sun C., Talbot H., Ourselin S. and Adriaansen T. (Eds.), 10-12 Dec. 2003, Sydney

Edge-based Non-uniform Light Field Mapping

Wang Wendong, Kong Dehui, Yin Baocai

Multimedia and Intelligent Software Technology Laboratory
Beijing University of Technology, Beijing 100022, China
Email: wwd@emails.bjut.edu.cn

Abstract. In this paper, an image-based method---non-uniform light field mapping is proposed which combines edge-based surface light field partition and non-uniform decomposition of light field matrixes for real-time rendering. Through this method view-dependent appearance of scenes with complex reflectance properties can be rendered in real-time. This method obtains surface light field by resampling the data which are captured from vantage points. To acquire interactive rendering rate we propose to partition surface light field over bi-triangles and arrange it into 2-D matrixes, then we approximate surface light field data by non-uniformly factoring these matrixes into textures. Finally, we present an image-based rendering method which improves rendering performance through textures sorting and rendering directly from these compact data with the support of graphic card.

1. Introduction

Many interactive 3D applications such as Virtual Reality, computer animation, game and education involve capturing and reproducing the appearance of real objects. Real objects, however, exhibit complex surface properties such as subsurface scattering and refraction. General representation must model these properties and render them photorealistic with interactive rate. In spite of the rapid progress in computer graphics over the past three decades, rendering complex scene with such reflectance properties in real-time is still a challenge. Image-based methods [1, 2, 3] are proposed to solve the problem, and many of these techniques are based on 4-D plenoptic function [1, 2, 15] ---parameterize the rays in 4-D empty space using two parallel planes.

Unfortunately, Image-based rendering (IBR) requires dense samples of the scene and uses thousands of images to ensure view-dependent photorealistic results. Geometry-less image-based methods [1, 2, 3, 6] take a collection images captured from vantage points as input, and construct the plenoptic function of the scene which can be used to synthesis novel images from arbitrary viewpoints. Levoy [1] developed a method to parameterize the rays using two parallel planes and constructed a 4-D plenoptic function. Novel images are synthesized by reverse searching in the sample database and interpolating between acquired ray samples. In this method, the representation restricts the viewpoint to lie outside of the convex hull of the object. View-dependent texture mapping (VDTM) [3, 4] is a kind of geometry-based IBR where multiple reference images (through visibility computation) are weighted and

composed together to render a novel image. This approach uses geometry information to re-project original images into the desired camera viewpoint and compute the visibility. VDTM supports rendering outside the convex hull of the object for using visibility computation and requires less images than geometry-less IBR.

Chen *et al.* [8, 10] developed a compact representation for surface light field (SLF) which was proposed by Wood *et al.* [5] and developed an algorithm to partition surface light field through mesh-ring (Vertex-based method) and proposed to approximate the discrete 4-D surface light field function as a sum of products of lower-dimensional functions which are called textures. The uniform compressed textures can be effectively decompressed and the object can be rendered on-the-fly by using commodity graphics hardware.

IBR makes a unique compression algorithm for its light field capture methods and random access requirements [1, 7, 15]. Levoy [1] applied vector quantization (VQ) algorithm to obtain compressed representations of light field. Magnor [7] proposed a MPEG-like compression algorithm for light field that produces better compression ratios than those obtained by Levoy. Nishino [13] developed a compression approach for dense images in Eigen-space. They constructed a set of texture maps for each cell by projecting each image onto the mesh. Compression is achieved by performing a principal component analysis on each set of textures.

Non-uniform light field mapping method which utilizes geometry of the object to reduce the data size and achieves interactive rendering through texture sorting and texture mapping can be used to solve these problems. First, the method obtains SLF by resampling the input images which are captured from vantage points. Then we propose to partition surface light field over bi-triangles and arrange them into 2D matrixes to achieve interactive rendering rate. Third, we approximate surface light field data by non-uniformly factoring these matrixes into textures according to their individual properties. Finally, an image-based rendering method is presented which improves rendering performance by textures sorting and rendering directly from these compact data with the support of graphic card. The pipeline of edge-based non-uniform light field mapping is shown in Fig. 1.

Our approach depends on both geometry and dense images of the scene thus lead to compact representation and efficient rendering. It removes the convex hull restriction of the two-plane light field parameterization and admits a progressive representation which can be rendered in interactive rate through textures sorting algorithm and texture mapping technology.

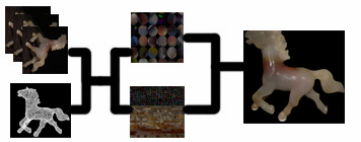


Fig. 1. The pipeline of edge-based non-uniform LFM. Left: acquired images and geometry of the object. Middle: view textures and surface textures. Right: rendering result

The following sections describe these algorithms in detail. We begin by describing the representation of surface light field through edge-based partitioning in section 2. Next, we discuss our resampling algorithm for surface light field in section 3.1 and introduce our factoring algorithm based on bi-triangle in section 3.2. Then we discuss the rendering algorithm in section 4. Finally, we illustrate the experiments results and present ideas for future research in section 5 and section 6 respectively.

2. Representation

A SLF [5] is also a 4-D function that completely defines the outgoing radiance of every point on the surface of an object in every viewing direction:

$$L(r,s,\theta,\phi) \rightarrow (r,g,b) \quad (1)$$

The first pair of parameters of this function (r,s) describes the surface location and the second pair of parameters (θ,ϕ) describes the viewing direction. In practice, a surface light field function is normally stored in discrete samples, which requires much storage thus leads to impractical direct manipulation. Chen *et al.* [8, 10] proposed to partition the 4-D surface light field based on mesh vertex as shown in Equation (2):

$$L^{e^k}(r,s,\theta,\phi) = L(r,s,\theta,\phi) \bullet e^k(r,s) \quad (2)$$

We assume the geometry of the scene is represented by mesh triangles. Through experiments, we find that the partition function $e^k(r,s)$ should satisfy all of the following criteria simultaneously to overcome rendering discontinuous introduced by resampling through mesh triangles (see section 3.1 for details):

- The function should divide surface light field data in the (r, s) domain without altering the original data.
- The function should be continuous within one primitive.
- To reduce blurring artifacts, the function should be zero in the edge of the primitives.

Through these analyses, we propose to partition surface light field based on edge of mesh triangles as illustrated in Fig. 2. They can be described in Equation (2).

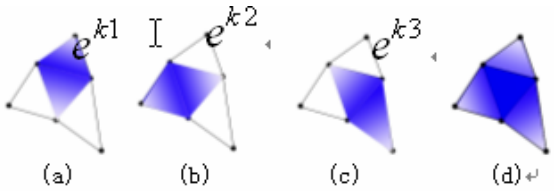


Fig. 2. Edge-based partition function: e^{k1}, e^{k2}, e^{k3}, which add to unity in centric triangle Δt

Here, we define three partition functions for centric triangle Δt: e^{k1} for edge k_1, e^{k2} for edge k_2 and e^{k3} for edge k_3. They all satisfy the three criteria and add to unity in centric triangle, so this partition method preserves the original surface light field for all mesh triangles. And because the partition functions reduce to zero at triangle edge, they ensure surface light field continuous in triangle edge. To accelerate calculation speed, a predefined lookup table based on barycentric coordinate is used for edge-based surface light field partition. The same process can be adopted for all edges in scene mesh and they add to unity in any triangle. We refer to the surface light field unit corresponding to each

edge $L^{e_k}(r,s,\theta,\phi)$ as *edge light field*.

Finally, we define local coordinate system for edge-based surface light field partition: the surface normal at the middle point of the common edge is labeled as Z-axis and the common edge is defined as X-axis. Each edge light field is re-parameterized in their individual local coordinate systems. To reduce notation complex, we use the same letter to denote edge light fields in their local coordinate frames. All processing in non-uniform light filed mapping is identical for each bi-triangle, so we explain algorithms for only one bi-triangle in the following section.

3. Encoding Surface Light Field

The processing of edge-based non-uniform light field mapping can be divided into two main steps: Encoding surface light field into compact textures and rendering directly from these textures.

Encoding surface light field for a real object requires the following steps: (1) range scanning the object and build a triangle mesh to represent its geometry; capture a collection of images of the object from vantage points and register them to the mesh [5, 14, 8], (2) resample the raw data to obtain surface light field through visibility computation, (3) partition the surface light field based on bi-triangles, (4) non-uniformly factor 4-D surface light field into a sum products of 2D function (also called textures).

Raw data acquire and register step can be done use algorithms developed by Chen [8]. Our method can be described with Fig.1 when the raw data has been acquired. The upper-left photos in Fig.1 describe the raw images acquired use camera from different view positions and bottom-left photo in Fig.1 denote object geometry acquired by range-finder. The middle of the figure presents the compact textures---view maps and surface maps which obtained from our non-uniform matrixes factoring. Resampling algorithm is adopted between left and middle components of this figure. The right photo is a result rendered from a viewpoint using our edge-based light filed mapping algorithm.

3.1 Resampling

Surface light field [5, 8, 10] requires dense samples as input, but we can only acquire scattered samples of actual surface light field function in practice. Resampling, therefore, is an intuitive method to approximate the surface light field function. We assume the edge sizes of the triangles are relatively small compared to their distance to the cameras, through which the resampling process which is considered a 4D function reconstruction problem traditionally can be realized in two steps: resampling parameters (r,s) and (θ,ϕ) respectively.

Before data resampling, visible cameras for each mesh triangle should be computed to determine which camera position can be used for the triangle resampling. We

define full-visible list and part-visible list for each mesh triangle, which corresponds to a set of irregular size texture patches captured from various camera positions, depending on its area shadowed by other triangles.

$$\text{If } K = \frac{\tilde{A}^{V}_{\Delta t}}{A^{V}_{\Delta t}} > k_1 \qquad \text{camera V is full-visible to } \Delta t$$
$$\text{If } k_2 < K < k_1 \qquad \text{camera V is part-visible to } \Delta t \tag{3}$$

Where $A^{V}_{\Delta t}$ denotes the area of Δt and $\tilde{A}^{V}_{\Delta t}$ represents the visible area of Δt from viewpoint V. k_1 is a predefined constant to determine whether the viewpoint can be added to the full-visible list of this triangle. k_2 is another predefined constant to determine whether this viewpoint can be added to part-visible list of this triangle. We take $k_1 = 0.9$ and $k_2 = 0.3$ in our experiments respectively.

Let V_i^f denotes full-visible viewpoint in camera i for Δt, V_i^p denotes part-visible viewpoint in camera i for Δt. And let list $\mathbf{F}_{\Delta t} = \{\mathbf{F}_{\Delta t,1}, \mathbf{F}_{\Delta t,2}, \cdots, \mathbf{F}_{\Delta t,\tilde{N}_t}\}$ denote full-visible views of Δt and list $\mathbf{P}_{\Delta t} = \{\mathbf{P}_{\Delta t,1}, \mathbf{P}_{\Delta t,1}, \cdots, \mathbf{P}_{\Delta t,P_t}\}$ denote part-visible views of Δt, which are sorted by K defined in Equation (3). Each element in the two lists is vector which consists in corresponding pixel values in raw image that re-projected to its geometry mesh from computed camera point $V_i^l \ (l = f, l = p)$.

Before view resampling, we should convert $\mathbf{F}_{\Delta t}$ and $\mathbf{P}_{\Delta t}$ to bi-triangle form. This can be done simply by compute the union list of two common-edge triangles. Let $\mathbf{F}_{e^k}$ and $\mathbf{P}_{e^k}$ represent full-visible view list and part-visible view list respectively. We use M_{e^k} to represent the number of points in bi-triangle e^k and $\tilde{N}_{e^k}$ to denote full-visible viewpoints number for bi-triangle e^k. $V_i^l \ (l = f, l = p)$ is converted to $V_{e^k,i}^l \ (l = f, l = p)$ by the same method.

Then parameters (r, s) resampling process can be done directly through bi-interpolation and a $M_{e^k} * \tilde{N}_{e^k}$ matrix $\mathbf{M}$ for bi-triangle e^k can be obtained.

We will use Fig. 3 to explain the view resampling algorithm. Before view resampling, vectors $\mathbf{F}_{e^k}$ and $\mathbf{P}_{e^k}$ should be converted to local coordinate system of each bi-triangle as defined in section 2. We use $\mathbf{F}^L_{e^k}$ and $\mathbf{P}^L_{e^k}$ to denote each vectors in local reference frame defined by edge e_k. And let $\mathbf{C}^L_{e^k}$ represent current ìvisible viewsî and should be initialized equal to $\mathbf{F}^L_{e^k}$. First, we project all full-visible viewpoint $V^f_{e^k,i}$ of bi-triangle e^k onto the *x-y* plane of local reference frame defined

by edge e_k using Equations (4) to form a projection map as shown in Fig. 3a.

$$x = (\mathbf{d} \bullet x' + 1) / 2$$
$$y = (\mathbf{d} \bullet y' + 1) / 2 \qquad (4)$$

Where $\mathbf{d}$ denotes corresponding view direction $V^{l}_{e^k,i}$ $(l = f, l = p)$. Each point in the maps denotes a texture coordinate (x, y).

Second, we perform the Delaunay-triangulation of these points (see Fig. 3b). Third, we take current element in part-visible list $\mathbf{P}^{L}_{e^k,i}$ and project corresponding view direction $V^{p}_{e^k,i}$ into *x*-*y* plane. Then we approximate the current project part-visible view $\mathbf{P}^{L}_{e^k,i}$ (red point in Fig. 3c) using linear combination of $\mathbf{C}^{L}_{e^k}$ and obtain $\tilde{\mathbf{P}}^{L}_{e^k,i}$. Fourth, we evaluate L according to Equation (5) to determine whether this part-visible view should be added to current ì visible viewsî $\mathbf{C}^{L}_{e^k}$ and used for further Delaunay-triangulation. If L is larger than a predefined threshold, we add $\mathbf{P}^{L}_{e^k,i}$ to $\mathbf{C}^{L}_{e^k}$ and perform a new circle Delaunay-triangulation as discussed in step 2 and 3 according to the order in $\mathbf{P}_{e^k}$. Finally, we compute the regular grid of views by the combination views in $\mathbf{C}^{L}_{e^k}$.

$$L = \sum_{j \in V} (| P^{L}_{e^k,i,j} - \tilde{P}^{L}_{e^k,i,j} |) / N \qquad (5)$$

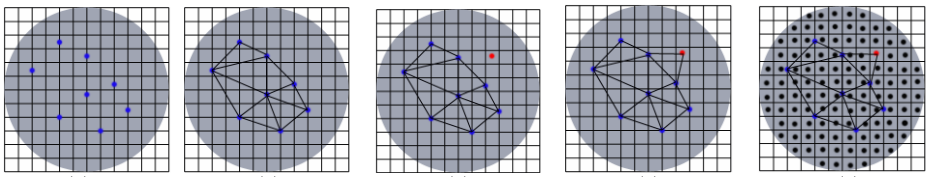


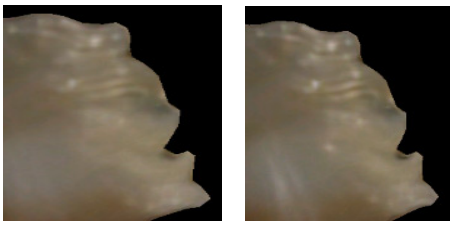
Fig.3 Resampling views: (a) Projection of original full-visible views, (b) and (d) Delaunay triangulation of current projected views, (c) Projection of current part-visible views, (e) Uniform grid of views computed by blending current ì visible viewsî

Where $P^{L}_{e^k,i,j}$ denotes the valid (not shadowed by other triangles in current viewpoint) pixel value for bi-triangle e^k from viewpoint $V^{p}_{e^k,i}$, N represents the valid points number and $\tilde{P}^{L}_{e^k,i,j}$ represents the same value computed by the combination of $\mathbf{C}^{L}_{e^k}$.

The rendering result of utilizing part-visible samples is shown in Fig. 4. As shown in this figure, the image edge would be less blurring by using part-visible

samples.

We have obtained approximate surface light field through data resampling algorithm. To build a computational matrix and eliminate the discontinuities across triangle boundaries, we propose to partition surface light field based on edges of triangles $e^k_{t1,t2}$, which is illustrated in Fig. 2 and described in section 2.



(a) **(b)**

Fig. 4. Compare rendering result between not using part-visible samples (a) and using part-visible samples (b)

3.2 Factoring SLF matrix

Through resampling and partition, we acquire a matrix as shown in [8]. Edge-based partition of surface light field data allows us to approximate each *edge light field* independently while maintaining continuity. The codec invokes PCA [12, 13] or NMF [8, 11] algorithm to factor the matrix and obtain two types of 2-D function---surface maps and view maps. The factoring method can be described with Equation (6).

$$L^{e^k}(r,s,\theta,\phi) = \sum_{i=1}^{K} T_i(r,s) \bullet V_i(\theta,\phi) \qquad (6)$$

However, the decomposition term K should not be uniform for all mesh triangles. Nishino [13] proposed to determine the dimensions of Eigen-space according to whether the cells have highlights or self shadows. We separate all the mesh triangles into two groups based on its average brightness, and compute PSNR for each group. The result is shown in Table 1. From the table we can find that the decomposition term should be higher in highlights or self shadows areas to maintain uniform PSNR.

Approximation term K	Normal area PSNR	Highlights area PSNR (Uniform)	Highlights area PSNR (Non-uniform)
1	48	41	47
2	53	46	51
3	56	49	54

Table 1. PSNR of uniform decomposition and non-uniform decomposition

The simplest approach to maintain constant PSNR in PCA is to compute T according to Equation (7) where λ_i represents Eigen-values. When we maintain T to be constant, the approximation term k varies according to the properties of the bi-triangle. This method requires obtaining all the Eigen-values of each bi-triangle and therefore is not efficient.

$$T = \sum_{i=1}^{K} \lambda_i \; / \; \sum_{i=1}^{P} \lambda_i \qquad (7)$$

We adopt power iteration algorithm [9] to solve Equation (6) and propose a non-uniform decomposition algorithm to maintain PSNR constant accordingly. First, we

convert the resampled data from RGB to YC_rC_b , and determine the real decomposition term K for each mesh bi-triangle based on the average value of Y within it. Then, matrix **Y** is factored use computed term K and matrixes C_r and C_b are factored use term K/2. Finally, all YC_rC_b matrix factors are converted to RGB for rendering.

4. Rendering

Rendering is an inverse process of decomposition and can be intuitively achieved by applying Equation (8). The rendering performance, however, is not efficient. From the Equation (8) we can see that evaluating one complete approximation term is equivalent to multiplying pixel- by-pixel of the surface map and view map texture fragment pairs for three *edge light fields* of the triangle and adding the results together. This makes us to use texture mapping technology powered by current customer video card to accelerate the rendering rate.

$$\tilde{L}_{\Delta t}(r_p, s_p, \theta_q, \phi_q) = \sum_{i=1}^{K} \left(\sum_{j=1}^{3} T_{\Delta t,i}^{e^k}(r_p, s_p) \bullet V_i^{e^k}(\theta_q, \phi_q) \right) \qquad (8)$$

To avoid excessive texture swapping and improve rendering efficiency, we develop a textures sort algorithm to improve render performance. First, we predefine set of sizes for mesh triangles and tile the same size surface light field maps together. Since one triangle requires three surface maps per approximation term, we tile all these maps in the same collection. View maps, on the other hand, are independent to mesh triangles, so put all view maps of each term in one collection. Though each bi-triangle has one same view map for two triangles, we attach each triangle to one view map and sort these view maps (doubled number of bi-triangles) according to their attached triangle size in rendering process before filling to texture memory.

Chen *et al.* [8] propose an algorithm grouping view maps based on mesh triangle size. But because of the number within one ring is not constant, the algorithm is hard to implement in efficient way. For our edge-based partition approach, the algorithm can be efficiently realized. Let p denote the number of groups split from the triangle mesh according to predefined sizes and triangle size. We then produce one view map collection per group $[V_1, V_2, \cdots, V_p]$ and let $[S_1^i, S_2^i \cdots, S_{q_i}^i]$ be the collection of surface maps corresponding to view maps V_i (through textures sorting describe above). For each approximation term, the rendering algorithm can be described as follows:

```
for i = 1, … , p do
load sorted view maps V_i into texture unit 1
    for j=1, … , q_i do
```

load surface maps S_j^i into texture unit 2

render all triangles within S_j^i

endfor

endfor

5. Result

Data obtained from UNC are used to test our approach as shown in Fig. 5. From the figures we can see edge-based surface light field partition has advantages over vertex-based partition. First, non-uniform factoring has better reconstruction quality than uniform decomposition in same approximation term. Second, Non-uniform factoring has a bit smaller compression ratio than uniform factoring in same approximation term, but non-uniform factoring has smaller storage when the reconstruction quality is the same. Figure 5 (d) shows compress ratio of this data.

6. Conclusion and future work

This paper presents an edge-based non-uniform surface light field mapping algorithm. The compression ratio can be achieved up to 1000:1 through factoring edge-based surface light field matrixes and further invoking a normal still image compression algorithm (such as VQ). The approach can achieve an interactive rendering rate through textures sorting and texture mapping technology.

The non-uniform algorithm is not applied for NMF because it does not constraints the matrix factors to be orthogonal. The future work with non-uniform decomposition is to develop an orthogonal and negative factoring algorithm. 4-D surface light field can not represent dynamic scene. We are planning to extend this work to 6-D using EM-PCA algorithm.

Acknowledgments

We would like to thank Wei-Chao Chen at UNC for providing their raw data for our use. This work was supported by National ì863î Program (2001AA-114160) and Beijing Education Committee Program (P070701-01).

References

1. Marc Levoy and Pat Hanrahan. ìLight Field Renderingî. In *SIGGRAPH96,* pages31-42.

August, 1996.
2. Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski and Michael F. Cohen. ìThe Lumigraphî. In *SIGGRAPH96*, pages 43-54. August, 1996.
3. Paul E. Debevec, Camillo J. Taylor and Jitendra Malik. ìModeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approachî. In *SIGGRAPH96*, pages 11-20. August 1996
4. Paul E. Debevec, *et al.* ìEfficient View-Dependent Image-Based Rendering with Projective Texture- Mappingî. *Eurographics Rendering Workshop 1998*, pages 105ñ116, June 1998.
5. D. N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, Werner Stuetzle. ìSurface Light Fields for 3D Photographyî. In SIGGRAPH00, pages 287ñ296, July 2000.
6. Chris Buehler , Michael Bosse, Leonard McMillan, Steven Gortler and Michael Cohen. ìUnstructured Lumigraph Renderingî. In SIGGRAPH01, pages 612-619, August 2001.
7. M. Magnor and B. Girod. ìData Compression for Light Field Renderingî. IEEE Trans. Circuits and Systems for Video Technology, 10(3):338ñ343, April 2000.
8. Wei-Chao Chen， J. Bouguet , Michael H. Chu and Radek Grzeszczuk. ìLight Field Mapping: efficient Representation and Hardware Rendering of surface Light Fieldsî. In *SIGGRAPH02*, pages 447-456, July 2002.
9. Emilio Camahort Gurrea. ì4D Light-Field Modeling and Renderingî. *PhD thesis,* The University of Texas at Austin. 2001.
10. Wei-Chao Chen. ìLight Field Mapping: Efficient Repre- sentation of Surface Light Fieldsî. *PhD thesis*, The University of North Carolina at Chapel Hill. 2002
11. D. D. Lee and H. S. Seung. ìLearning the Parts of Objects by Non-Negative Matrix Factorizationî. *Nature*, 401:788ñ791, 1999.
12. Ralph Gross, Iain Matthews and Simon Baker. ìEigen Light-Field and Face Recongnition Across Poseî. *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition* (FGRí02).
13. K.Nishino, Y.Sato and K.Ikeuchi, "Eigen-Texture Method: Appearance Compression based on 3D Model". *In Proc. of Computer Vision and Pattern Recognition '99*, vol.1, pp618-624, Jun., 1999.
14. J-Y. Bouguet and P. Perona. ì3D Photography Using Shadows in Dual-Space Geometryî. *International Journal of Computer Vision,* 35(2):129ñ149, Dec- ember 1999.
15. Emilio Camahort, Apostolos Lerios and Donald Fussell. ìUniformly sampled light fieldsî *In Ninth Eurographics Workshop on Rendering*, pages 117ñ130, Vienna, Austria, JuneñJuly 1998.

(a) Photograph **(b) 2-term vertex-based** **(c)2-term edge-based** **(d) Compression ratio**

Fig. 5. Comparison between edge-based partition and vertex-based partition methods