Cube2Video: Navigate between Cubic Panoramas in Real-Time

Qiang Zhao, Liang Wan, *Member, IEEE,* Wei Feng, *Member, IEEE,* Jiawan Zhang, *Member, IEEE,* and Tien-Tsin Wong, *Member, IEEE*

***Abstract*—Online virtual navigation systems enable users to hop from one 360^o panorama to another, which belong to a sparse point-to-point collection, resulting in a less pleasant viewing experience. In this paper, we present a novel method, namely Cube2Video, to support navigating between cubic panoramas in a video-viewing mode. Our method circumvents the intrinsic challenge of cubic panoramas, i.e., the discontinuities between cube faces, in an efficient way. The proposed method extends the matching-triangulation-interpolation procedure with special considerations of the spherical domain. A triangle-to-triangle homography-based warping is developed to achieve physically plausible and visually pleasant interpolation results. The temporal smoothness of the synthesized video sequence is improved by means of a compensation transformation. As experimental results demonstrate, our method can synthesize pleasant video sequences in real time, thus mimicking walking or driving navigation.**

***Index Terms*—Cubic panoramas, virtual navigation, video-viewing mode, triangle-to-triangle homography-based warping, temporal smoothness**

I. Introduction

PAnorama-based virtual navigation systems become prevalent to ordinary people via the Internet, including Google Street View [1], Bing Maps Streetside [2], etc. Due to the bandwidth limit, current virtual navigation systems usually provide a *sparse point-to-point* collection of panoramas. The common navigation control is to enable users navigate within one panorama (panning, zooming, and rotating) [3][4], and hop from one panorama to another. The hopping manner is simple and fast, yet may bring users apparent visual discontinuity. Although this problem can be alleviated by image transition like cross-fading, as existing navigation systems did, it is often difficult to distinguish the scene clearly during the transition.

A more physically plausible solution is generating intermediate panoramas from reference ones [5][6][7][8][9][10]. Some early works [5][6][11][8][12] operate on cylindrical panoramas, while some recent works [9][13][10] operate on cubic panoramas, which suffer less non-linear deformation and have native hardware support. The most challenging problem in using cubic panoramas is to tackle the discontinuity between cube faces. For instance, Shi et al. [9] developed a pixel-based method tracing the optical ray of each pixel. Kolhatkar and Laganière [13] estimated the optical flow fields for each cube face extended by a boundary projected from adjacent faces. Zhang et al. [10] divided cube faces into central and boundary regions for triangulation, and synthesized new views via face-to-face homography transformation. Although these methods try to solve the discontinuity problem, they still have obvious artifacts, and/or are tedious for real-time navigation.

In this paper, we present a novel method that can circumvent the cube discontinuity problem in an easy and effective way. Our method is based on the simple concept that the cube is a representation of the sphere. Accordingly, we can project the cubic panorama on the sphere, and perform manipulations in the spherical domain.

The major contributions of our paper is three-fold. Firstly, we extend the matching-triangulation-interpolation procedure [7][14][10] with special considerations of the spherical domain (Section III). Specifically, we employ an angular error metric in the spherical domain to get reliable sparse correspondences between two cubic panoramas. Convex hull triangulation is then applied to triangulate the panorama normalized on the unit sphere. In the interpolation, a new warping scheme is performed between pairs of spherical triangles. The second contribution is proposing a (spherical) triangle-to-triangle warping, which combines a homography transformation and an affine transformation. Using this warping, we are able to achieve physically plausible and visually pleasant interpolation results. As the third contribution, we describe a compensation transformation to improve the temporal smoothness of the synthesized video (Section IV). Finally, parallel implementation for the new view generation is developed to achieve real time performance (Section V). As experimental results demonstrate, our method can synthesize satisfied videos for both in-door and out-door panorama sequences (Section VI).

II. Related work

A. View Interpolation

The introduction of view interpolation can date back to the mid-90s. A main stream of existing methods are based

Q. Zhao, W. Feng and J. Zhang are with Tianjin University, P. R. China (E-mail: qiangzhao_tju@163.com, wfeng@tju.edu.cn, jwzhang@tju.edu.cn).

L. Wan (Corresponding author) is with the School of Computer Software, Tianjin University and with Tianjin Key Lab for Advanced Signal Processing, Civil Aviation University of China, P. R. China (E-mail: lwan@tju.edu.cn).

T. T. Wong is with The Chinese University of Hong Kong, P. R. China (E-mail: ttwong@acm.org).

on optical flow, disparity or depth information [15][16][17], which need to establish dense correspondences between reference images. Another type of view interpolation is based on projective geometry. Seitz and Dyer [18] introduced view morphing that interpolates along the base line of an image pair and obtains physically plausible images. Their method was further extended by several works [19][20][21]. There also exist triangulation-based interpolation techniques. Lhuillier and Quan [22] presented joint view triangulation, which generates novel views by warping the matched triangles. This work was improved by Siu and Lau [23][14]. After constructing overlap-solved triangular meshes for three reference images, they developed a transformation-based warping for arbitrary view synthesis. This work is at the basis of our interpolation algorithm.

B. Panorama Interpolation and Navigation

Panorama interpolation considers a large field of view [24], e.g. 360 degrees in our work. According to the underlying spherical mapping representations, most existing techniques fall into two categories, using cylindrical panoramas [5][6][11][8][12], and using cubic panoramas [9][13][10]. For cylindrical panoramas, there are pixel-based solutions [5][8], and triangle-based warping methods [11][6]. In addition, the view synthesis for catadioptric omni-directional images can be done through rectification to cylindrical panoramas [12].

For cubic panoramas, Shi et al. [9] performed a search guided by color consistency for each pixel along its optical ray to find the correspondences in reference panoramas. Kolhatkar and Laganière [13] estimated optical flow fields on cube faces extended by boundaries projected from adjacent faces, and interpolated intermediate frames using view morphing. As the interpolation only involves blending, they achieved real-time navigation using GPU. The above two methods are rather simple, but they may suffer severe visual artifacts. There existed several works following the matching-triangulation-interpolation procedure [7][10], yet with detailed approaches different from ours. Yeung [7] segmented panoramas into regions, and established region correspondences. After triangulating the regions individually, special schemes are employed to get equal-sized triangular meshes in matched regions. Finally, hardware texture mapping [6] is applied to do interpolation. Zhang et al. [10] proposed a three-step triangulation method, in which the central regions of cube faces are first triangulated, then the meshes of the horizontal four faces are connected, and finally the regions surrounding up and down faces are processed. A new view is synthesized by performing face-to-face homography transformation [14], with the cube face discontinuity considered. Their method gets better interpolation results, however, is rather complicated for real-time navigation. Our method, on the other hand, can circumvent the cube discontinuity problem effectively, and achieve real-time navigation.

Based on the panorama collections, there are also works generating limited field-of-view videos for navigation purpose. For instance, Chen et al. [25] automatically varied the speed and field of view of the video to highlight turns and landmarks. Their method selects frames, without interpolation, from a sequence of panoramas captured at close intervals. Peng et al. [26] only considered a 60^o view facing the heading direction and simulated moving forward simply by the zoom-in effect. In our work, we can synthesize a 360^o panoramic video.

III. Interpolation of Cubic Panoramas

We now present how to synthesize an arbitrary view between two cubic panoramas.

A. Finding Reliable Correspondences

We begin with the description of epipolar geometry for the sphere [27]. For one world point $\mathbf{X}$, an epipolar plane is given by $\mathbf{X}$ and two camera centers $\mathbf{C}$, $\mathbf{C}'$ (illustrated in Figure 1). The epipolar constraint requires the corresponding image points $\mathbf{x}$ and $\mathbf{x}'$ lie on this plane, i.e., satisfying the condition

$$\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0, \tag{1}$$

where $\mathbf{F}$ is the fundamental matrix that can be decomposed as a combination of camera motion and calibration matrices.

After extracting feature points on cubic panoramas via SIFT [28], we estimate $\mathbf{F}$ using the eight-point algorithm [29], and eliminate correspondence outliers by the RANSAC framework [30]. In the RANSAC iteration, we should determine whether a match is complied with current $\mathbf{F}$. This is often realized by evaluating the geometric error metric [29], given by

$$\varepsilon_{geo} = \frac{(\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x})^2}{\|\mathbf{F}\mathbf{x}\|^2 + \|\mathbf{F}^{\mathrm{T}}\mathbf{x}'\|^2}, \tag{2}$$

where $\|\cdot\|^2$ is the normal-2 distance. The geometric error is effective to remove outliers departing from the epipolar plane, but may fail when the wrong candidate matches lie on or close to the epipolar plane. For example, in Figure 1 $-\mathbf{x}'$ is antipodal to $\mathbf{x}'$. In case that $-\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x}$ is close to zero, ε_{geo} is small, and then $-\mathbf{x}'$ will be wrongly accepted as an inlier. We found this problem is not seldom in our application, and may cause serious overlaps in the followed triangulation process.

To deal with this problem, we introduce an angular error metric under the assumption that the angels between different correspondence pairs are close. To avoid complicated reprojection computation, we define the angle between correspondences in a simplified way. As shown in Figure 1, we translate $\mathbf{x}$ into the second panorama sphere (see the red dot), and calculate the angle (denoted as α) between the resulting $\mathbf{x}$ and its possible match $\mathbf{x}'$. The angular error metric is defined as the extend of how the angle departs from the main angle distribution, given by

$$\varepsilon_{ang} = \frac{|\alpha - \mu_S|}{\sigma_S}, \tag{3}$$

where μ_S and σ_S are the mean and standard deviation of the angles between the correspondences in the consensus

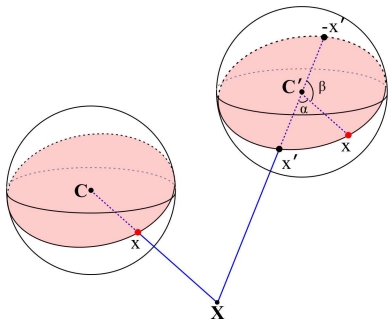


Fig. 1. The world point $\mathbf{X}$ gives a point correspondence, $\mathbf{x}$ and $\mathbf{x}'$. However, based on the geometric error, wrong candidate matches close to the epipolar plane (in light red color) may be mistaken for inliers, for example $-\mathbf{x}'$ which is antipodal to $\mathbf{x}'$. Our angular error metric can remove such outliers with abnormal angular offsets.

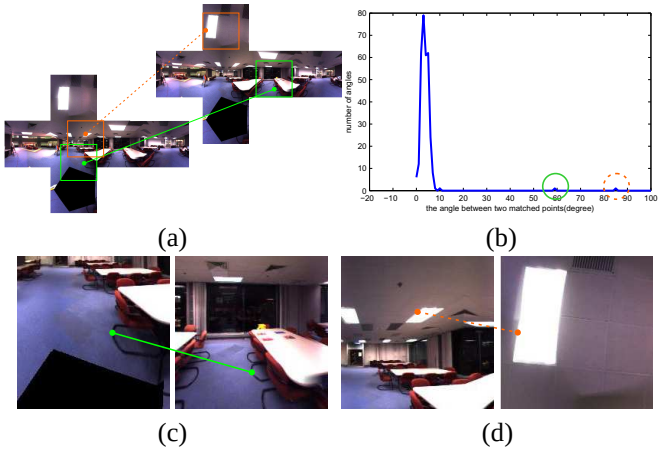


Fig. 2. Correspondence outliers removed by our angular error metric. (a) shows two correspondence outliers with small distances in geometric error, with blowups displayed in (c) and (d). (b) plots the angle distribution. The peaks marked with circles correspond to the two outliers, respectively.

set S determined by the geometric error. We empirically set the threshold for the angular error to be 0.95. One real example is shown in Figure 2. From our experiments, we found that the angular error metric is applicable for both indoor and street-level panorama sequences.

B. Triangulating Panoramas

Unlike the work of Zhang et al. [10], which adopts different triangulation schemes for face central regions and boundaries, we do triangulation in a unified way. Specifically, since the feature points have been projected on the sphere, triangulating a panorama can be done by applying convex hull triangulation algorithm [31]. As a sphere is completely symmetric around the center, there is no discontinuity problem when triangulating the sphere, and hence the cube face discontinuity can be naturally ignored.

Given two neighboring panoramas, we perform triangulation on the first sphere and apply the mesh connectivity to the second sphere. Due to moving objects/cameras and/or mismatched points, the triangular mesh on the second panorama may have overlaps inside. For instance, in Figures 3(a) and 3(b), the feature points in the red colored regions are correctly matched, however, the moving car with relatively large disparity results in an overlap in the second sphere.

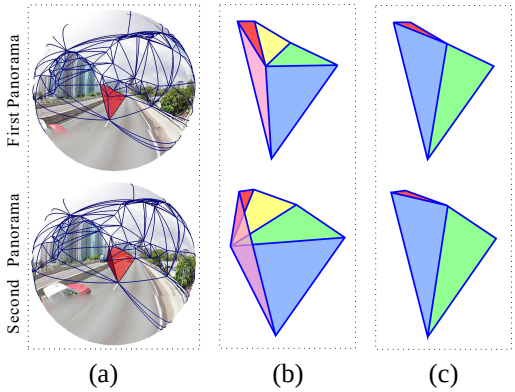


Fig. 3. Triangulating panoramas: (a) The triangulation results for two neighboring panoramas. The overlap on the second panorama is indicated as the crimson region. (b) The triangular meshes of the overlapped regions. (c) The meshes after the rematching.

To solve such overlaps, we first conduct checking of the topology consistency, which requires no triangle intersection in the second triangular mesh. Specifically, we begin with one triangle in the second mesh and include it in the matched region. We then check whether a new triangle, adjacent to the matched triangles, arises intersections. If an intersection occurs, the new triangle is rejected as unmatched. This checking process separates the triangular mesh into matched regions and unmatched regions. Next, we rematch the unmatched regions. Since the unmatched region is a curved surface, for which the convex hull triangulation is not applicable, we project the boundary of each unmatched region to a 2D plane using gnomonic projection, and perform 2D edge constrained Delaunay triangulation [14] as in Figure 3(c).

The checking-and-rematching process continues until there is no unmatched regions or the unmatched regions remain unchanged. We found the later case occurs when the unmatched region on the first sphere have convex boundary, while the corresponding region on the second sphere have concave boundary. Therefore, we use a ping-pong technique. That is, we first do the rematching on the first sphere, and check the topology consistency for the second one. If there still exist unmatched regions, we do the rematching on the second sphere, and check the topology consistency for the first one. This process is fast to converge, which usually needs no more than 5 iterations, and after this process there are rarely unmatched regions in our experiments.

C. Generating a Novel View

After the panoramas are triangulated, we develop a triangle-based scheme for the novel view generation. It contains three steps: 1) determining the destination triangular mesh of the novel view; 2) computing a triangle-to-triangle homography-based transformation between a pair of source and destination triangles; 3) synthesizing the destination triangles by the backward warping. Since all

the triangles are operated in the spherical domain, we need not to differentiate the triangles that cover more than one cube face as Zhang et al. did [10].

1) Determining the Destination Triangular Mesh: To determine a destination triangle, we only need to compute the coordinates of its vertices. Given a pair of correspondences, $\mathbf{x}$ and $\mathbf{x}'$, we first recover the corresponding 3D scene point $\mathbf{X}$ using linear triangulation [29], then reproject $\mathbf{X}$ to the novel view based on the view's position $\mathbf{C}''$ and orientation $\mathbf{R}''$, giving the reprojected vertex $\mathbf{x}''$ as

$$\mathbf{x}'' = \frac{\mathbf{R}''(\mathbf{X} - \mathbf{C}'')}{\|\mathbf{R}''(\mathbf{X} - \mathbf{C}'')\|}. \tag{4}$$

$\mathbf{C}''$ and $\mathbf{R}''$ can be computed via (13) and (14) respectively. Thanks to the gnomonic projection associated with the cubic panorama, we can no longer differentiate a spherical triangle and its projective counterpart against the sphere center. Without introducing ambiguity, we mainly refer to the projective triangle in the following discussion.

2) Triangle-to-Triangle Homography Transformation: Given one triangle ΔEDF as shown in Figure 4, we set up a virtual camera located at the sphere center, with the image plane π defined by the projective triangle ΔEDF. Here, we choose $\overrightarrow{DE}$ to be the x-axis of the image plane π. The local homogeneous coordinates of vertices D, E, F in the image plane π can be expressed as

$$[\mathcal{X}_D \ \mathcal{X}_E \ \mathcal{X}_F] = \begin{bmatrix} 0 & |\mathbf{E}-\mathbf{D}| & \cos\theta|\mathbf{F}-\mathbf{D}| \\ 0 & 0 & \sin\theta|\mathbf{F}-\mathbf{D}| \\ 1 & 1 & 1 \end{bmatrix}, \tag{5}$$

where θ is the angle between edges $\overline{DE}$ and $\overline{DF}$.

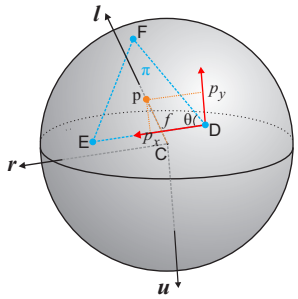


Fig. 4. Setting up of the virtual camera for ΔEDF, with the coordinate system specified by ***rCul***. See text for more details.

For the virtual camera, the line passing through the sphere center C, with the direction perpendicular to the image plane defines the principal axis ***l***. The intersection point p is the principal point, specifying the camera calibration matrix $\mathbf{K}_t$ as follows,

$$\mathbf{K}_t = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{6}$$

where f is the focal length; p_x, p_y are 2D offsets of principal point in the image plane. The camera orientation respective to the sphere is given by

$$\mathbf{R}_t = [\boldsymbol{r}, \boldsymbol{u}, \boldsymbol{l}]^T, \tag{7}$$

where ***r*** is the direction of the positive x-axis in the image plane π; $\boldsymbol{u} = \mathrm{cross}(\boldsymbol{l}, \boldsymbol{r})$. By taking the global position ($\mathbf{C}$) and rotation ($\mathbf{R}$) information for the sphere into account, we have the final camera matrix given by

$$\mathbf{P}_t = \mathbf{K}_t\mathbf{R}_t\mathbf{R}[\mathbf{I} \ -\mathbf{C}]. \tag{8}$$

For a pair of source (e.g. on the first panorama) and destination triangles, we first set up the virtual camera for each triangle, and get their 2D local coordinates and camera matrices using (5) and (8). We then compute the homography matrix $\mathbf{H}_h$ based on three point correspondences $\mathcal{X}_i \leftrightarrow \mathcal{X}_i''$ and two camera matrices $\mathbf{P}_t \leftrightarrow \mathbf{P}_t''$, i.e.

$$[\{\mathcal{X}_i \leftrightarrow \mathcal{X}_i''\}, \mathbf{P}_t \leftrightarrow \mathbf{P}_t''] \rightarrow \mathbf{H}_h. \tag{9}$$

Details about the computation can be found in [14][29].

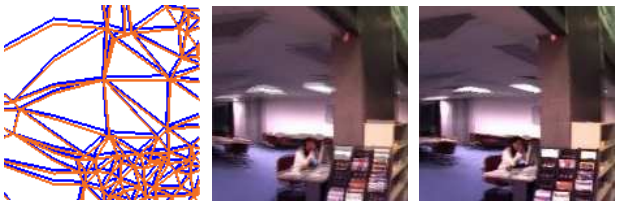
The above transformation is physically plausible, however, it is affected by small reconstruction errors in recovering 3D scene points, which is an over-determined problem. This will make $\mathcal{X}_i'' \neq \mathbf{H}_h\mathcal{X}_i$ (see Figure 5(a)), and may lead to blurring in the interpolated results (see Figure 5(b)). To tackle this problem, we enforce the position constraint that requires the transformed point coincide with the original point. Specifically, we get an intermediate point $\widehat{\mathcal{X}}_i = \mathbf{H}_h\mathcal{X}_i$, and compute an affine matrix $\mathbf{H}_a$ from the new point correspondences $\widehat{\mathcal{X}}_i \leftrightarrow \mathcal{X}_i''$, i.e.

$$\{\widehat{\mathcal{X}}_i \leftrightarrow \mathcal{X}_i''\} \rightarrow \mathbf{H}_a. \tag{10}$$

The final warping matrix is computed as

$$\mathbf{H} = \mathbf{H}_a\mathbf{H}_h. \tag{11}$$

As shown in Figure 5(c), the refined homography transformation can improve the interpolation quality.



(a) (b) (c)

Fig. 5. Enforce the position constraint that requires the transformed point coincide with the original point. (a) The triangles transformed using $\mathbf{H}_h$ (in red color) may not coincide with the triangles of the first reference panorama (in blue color). (b) Interpolated result using $\mathbf{H}_h$ as the warping matrix. (c) Interpolated result using $\mathbf{H} = \mathbf{H}_a\mathbf{H}_h$ as the warping matrix.

3) Synthesizing the Interpolated Panorama: To generate the interpolated cubic panorama, the backward warping is used. For every pixel on the novel view, we find its corresponding pixels on the two reference panoramas by means of estimated warping matrices, and do color blending.

Note that the warping matrices are specified for the triangles defined in a local image system. Given a pixel on the novel view, we should convert its 3D coordinates defined in the panorama's camera system to 2D coordinates defined in the local image system. Without loss of generality, suppose that pixel P falls inside $\triangle ABC$. We can compute its barycentric coordinates $[\alpha, \beta, \gamma]^\mathsf{T}$ according to its 3D

coordinate $\mathbf{x}_P$ and the triangle vertices $\mathbf{x}_i, i \in \{A, B, C\}$. Then the local 2D coordinates of pixel P can be represented as $\mathcal{X}_P = [\mathcal{X}_A, \mathcal{X}_B, \mathcal{X}_C][\alpha, \beta, \gamma]^\mathsf{T}$. These two steps can be substituted by matrix multiplications, i.e.,

$$\mathcal{X}_P = [\mathcal{X}_A, \mathcal{X}_B, \mathcal{X}_C][\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C]^{-1}\mathbf{x}_P. \quad (12)$$

After getting $\mathcal{X}_P$, the warping matrices can be used to find the corresponding pixels in the local image system of the triangles on the reference panoramas. An inverse procedure is performed to get 3D coordinates of the correspondences for texture lookup.

In the above procedure, we need to determine which triangle pixel P falls inside. The triangle-detecting operation is conducted for all the pixels. For speed up, we utilize both the neighboring triangle-detecting information for the pixels, which is stored as an index map, and the adjacency table for the triangular mesh that records for each triangle a list of adjacent triangles. Figure 6 demonstrates an example. When rendering a pixel p, we first check the triangles which its previously proceeded neighborhood pixels $N(p)$ fall inside, for instance, triangles 3, 4 for pixel p_1. If the check fails, for example for pixel p_2, we then take the adjacent triangles, saying triangles $0, 3, 4, 6$ adjacent to triangle 5, as the checking set. If both steps fail, we finally take all the remaining triangles as the checking set. Also note almost all the triangle-detecting operations can be determined using the first two steps. This hierarchical scheme can speed up the triangle-detecting operation significantly, e.g. from 130.507 seconds without the speedup to 8.305 seconds, for a triangular mesh with 950 triangles and a cubic face resolution of 512×512 pixels.

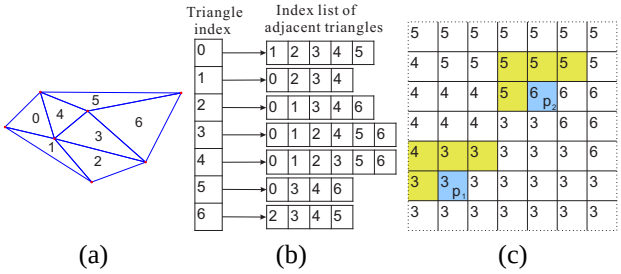


Fig. 6. The data structures used in the triangle-detecting operation. (a) An example triangular mesh. (b) The adjacency table for the mesh in (a). (c) A part of the index map of triangles which the pixels fall inside, and the preceding pixels are colored yellow. By using the index map and the adjacency table, the triangle-detecting operation can be accelerated.

IV. Panoramas to Video

We now describe how to synthesize a smooth video sequence from sparsely-collected reference panoramas. Intuitively, a series of novel viewpoints are to be interpolated between pairs of reference panoramas. Suppose two neighboring panoramas have positions given by 3D vectors $\mathbf{C}$ and $\mathbf{C}'$, and orientations given by 3×3 rotation matrices $\mathbf{R}$ and $\mathbf{R}'$. We linearly interpolate the position as

$$\mathbf{C}'' = t\mathbf{C}' + (1-t)\mathbf{C}, \quad (13)$$

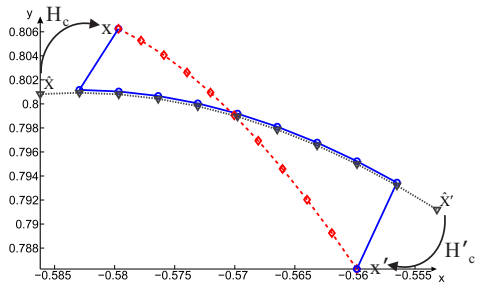


Fig. 7. Interpolating the panoramas by simply projecting the scene points causes temporal shaking in the generated video. The blue curve shows the position track of one reprojected scene point using Equation 4, interpolated at $t = 0.1, \ldots, 0.9$ and enclosed by two reference panoramas. We compute two compensation matrices $\mathbf{H}_c$ and $\mathbf{H}'_c$ that transform the reprojected feature points (the gray curve) to make the video more smooth (see the red curve). Please note in order to illustrate the tracks more clearly, the figure only shows the view of x-y plane.

where $t \in [0, 1]$ is the interpolation variable. The orientation interpolation is accomplished using spherical linear interpolation of quaternion (Slerp), i.e.,

$$\mathbf{q}'' = \mathrm{Slerp}(\mathbf{q}, \mathbf{q}'; t) = \frac{\sin(t\theta)}{\sin\theta}\mathbf{q}' + \frac{\sin[(1-t)\theta]}{\sin\theta}\mathbf{q}, \quad (14)$$

where $\mathbf{q}$, $\mathbf{q}'$ and $\mathbf{q}''$ are quaternions corresponding to rotation matrices $\mathbf{R}$, $\mathbf{R}'$ and $\mathbf{R}''$, and $\theta = \arccos(\mathbf{q} \cdot \mathbf{q}')$. After the novel viewpoints are determined, we can synthesize a video using the proposed algorithm in Section III.

However, we found there may exist temporal shaking between the interpolated frames and the original panoramas. We investigate this problem by analyzing the position track of feature points. As shown in Figure 7, the 9 interpolated positions at $t = 0.1, \ldots, 0.9$ together with the positions in the reference panoramas, $\mathbf{x}$ and $\mathbf{x}'$, form the blue curve, where the positions are spherical coordinates on the unit sphere. Apparent changes occur between the first (last) two points on the curve, although the changes are rather small in absolute values. We also interpolated two panoramas at $t = 0, 1$ using (4), yielding gray points $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$. As we can see, the reprojected positions $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ deviate from the original positions $\mathbf{x}$ and $\mathbf{x}'$, while they together with the 9 interpolated positions compose the gray curve, which is smooth. Hence, this shaking problem is mainly caused by estimation errors in the reconstruction of 3D scene points.

We handle the shaking problem by transforming the gray curve to make its starting and end point coincide with $\mathbf{x}$ and $\mathbf{x}'$, respectively. This can be achieved by computing two compensation matrices which satisfy

$$\mathbf{x}_i = \mathbf{H}_c\hat{\mathbf{x}}_i, \text{ and } \mathbf{x}'_i = \mathbf{H}'_c\hat{\mathbf{x}}'_i, \quad (15)$$

based on three correspondences $\hat{\mathbf{x}}_i \leftrightarrow \mathbf{x}_i$ and $\hat{\mathbf{x}}'_i \leftrightarrow \mathbf{x}'_i$ in triangles. Then, like (13) we linearly interpolate the compensation transformation as

$$\mathbf{H}''_c = \mathbf{t}\mathbf{H}'_c + (1-\mathbf{t})\mathbf{H}_c, \quad (16)$$

and apply it to the reprojected positions. The newly generated red curve is used as the position track of the feature point, based on which the warping matrix will be estimated.

For the example shown in Figure 8(a), we count the average error between the reprojected points in image space with and without compensation, which is about 21.6 pixels for a cube face size 512×512. As the shaking phenomenon is apparent only in video playing, readers are referred to the accompanying video for the visual comparison.

V. Implementation Details

In the implementation, we separate the algorithm into a preprocessing stage, which includes the matching and triangulation, and an on-the-fly stage that is the novel-view generation. Note that all the three steps in the novel view generation are to be performed for every individual primitive (triangles or pixels). Hence, they are highly parallelizable, and we can improve the timing performance by utilizing the GPU's parallel computing power. Specifically, one CUDA pass computes the triangular mesh on the novel view, the warping matrices, and the compensation matrices. Another CUDA pass is taken for novel view synthesis, in which the multiple transformations, including the warping matrices and the coordinate mappings, can be accomplished easily by means of matrix multiplications. After that, the generated panorama is used for environment mapping in a DirectX rendering pass, and interactions such as zooming, panning, rotation are provided.

Note the triangle-detecting speed-up process introduced in Section III-C3 is not compatible with GPU processing, because of the irregular size of the adjacency table. Alternatively, we develop a GPU-based strategy to acquire the triangle index map. Since every triangle in the interpolated panorama has only one corresponding triangle in the reference panorama, the pixels within one destination triangle will share the same reference triangle's index. This reminds us of the hardware rasterization ability which can flat fill triangles with single colors. In the implementation, we add two passes to the pipeline. One general computing pass is to assemble the destination vertices to triangle elements with the color indicating the reference triangle's index, after the first CUDA pass. To use flat-filling capability, the vertices shared by different triangles are repeated, so that each vertex will have the correct triangle index. Since the cubemap is hardware supported, a rendering pass is then carried out to render the destination triangular mesh to the cubemap, generating the triangle index map.

VI. Experiments and Discussion

In this section, we evaluate our approach using different panoramas and make comparisons with existing techniques, then talk about the timing performance of our system, finally discuss our limitations.

A. Experimental Results

1) Results for Indoor and Street-level Imageries: In the first experiment, we tested on an indoor panorama sequence, which was captured using the Ladybug2 camera mounted on a manually movable platform. The sequence contains 33 frames, separated with small distances (roughly about 0.7 meter) and slight camera rotation. Figure 8 displays interpolation results at $\mathbf{t} = 0.25, 0.5$ and 0.75 between two pairs of neighboring panoramas. As Figure 8(a) shows, the scene is well interpolated, with a tendency to move rightward. In Figure 8(b), we can notice the lamp near the column becomes dark gradually. We also compose a video clip by interpolating 9 frames between pairs of adjacent panoramas (see the accompanying supplementary video). As the scene is static and the camera motion is relative small, the generated clip looks pleasing and temporally smooth to mimic walking navigation.

In the second experiment, we tested on street-level panoramas from Google Street View, which are taken from panoramic cameras mounted on specially adapted cars. There are many moving objects in the images and the displacement between adjacent panoramas is large, about 25 meters in our example. Figure 9 shows two interpolated sequences at $\mathbf{t} = 0.2, 0.4, 0.6, 0.8$. Because of the large depth range and moving objects in the outdoor scene, the occlusion phenomenon is more serious than that in the "Reading-room" indoor scene. Although our triangulation process can handle the occlusion to some extend, we may miss some cases. Despite this, as can be seen in Figure 9 and the supplementary video, our method can still get reasonable results for these sparsely collected imageries.

2) Comparison between Different Methods: We compare our method with three recent approaches, i.e. [9], [13] and [10]. In experiments, we implemented the first two methods, while due to the lack of technical details, we used the results from [10]. In Figure 10, 10(a), 10(b) and the first row in 10(c) are the panoramas captured in the laboratory of University of Ottawa with the camera moving forward. They are spaced with a distance of 1 meter. The first row in Figure 10(c), which is captured between Figures 10(a) and 10(b), is used as the ground truth.

The third row in Figure 10(c) is the result reported in [9] (note that this method uses four surrounding cubes as input views). We can see the result suffer from visible errors in the ceiling and the left wall shown in Figures 10(d) and 10(e), respectively. It is because the method adopts a pixel-based interpolation that is sensitive to the errors in estimated depths and color differences, and is difficult to guarantee the color smoothness over the entire image. The fourth row in Figure 10(c) is the result generated by the method of [13]. Since this method is based on optical flow estimation, the results may have many artifacts when the displacement is relatively large. The result at the last row in Figure 10(c), which was provided by the authors of [10], is better than previous two results. However, it still has apparent artifacts in the ceiling region (see Figure 10(e)). What's more, because this method triangulates the cubic panorama in form of the cube, there are black lines between cube faces in the result (see Figure 10(d)). Thanks to the triangulation on the unit sphere and the triangle-to-triangle homography-based transformation, our method can generate results with less artifacts (the second row in Figure 10(c)), and is best of the four methods.

Next, we carry out objective evaluations in Table I

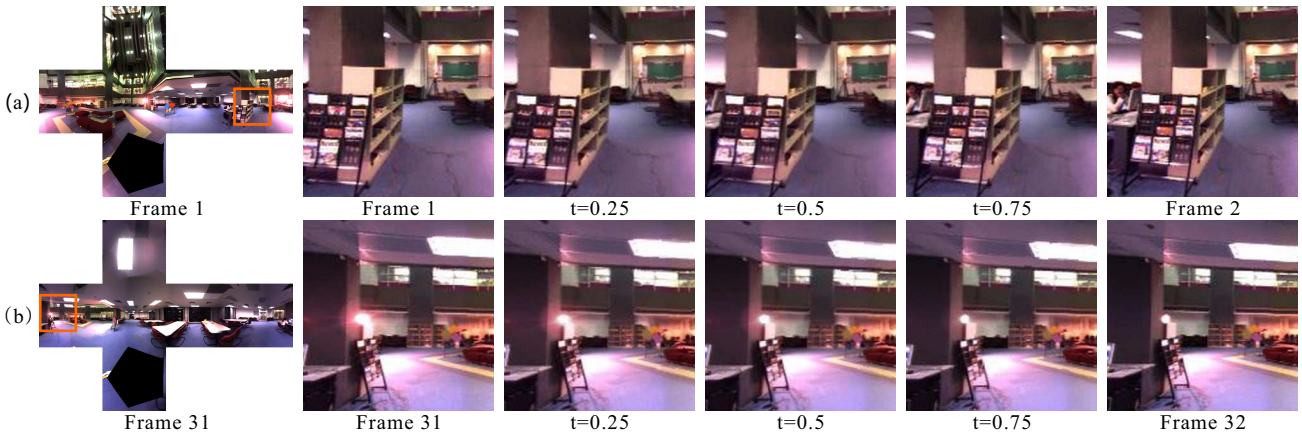


Fig. 8. The generated results for the "Reading-room" panorama sequence using our method. (a) Results interpolated from frame 1 to 2; (b) Results interpolated from frame 31 to 32.

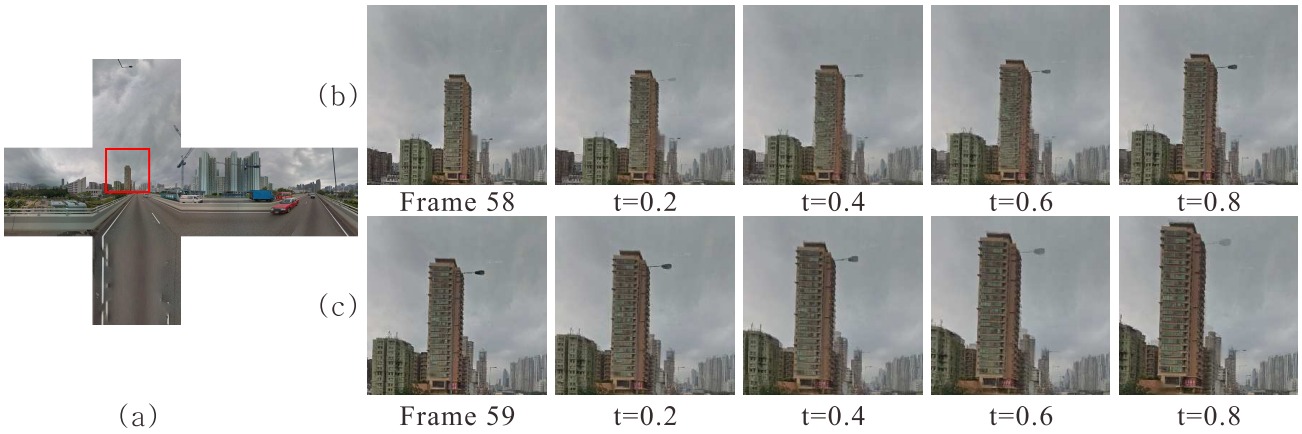


Fig. 9. The generated results for the Google Street View©data in Hong Kong using our method. (a) Reference panorama at frame 58, (b) and (c) are the generated results. The average distance between the reference panoramas is about 25 meters. As shown, the interpolated results are reasonable, though with some blurry artifacts.

and II by employing the PSNR and structural similarity index (SSIM) [32] measurement. Besides the example in Figure 10 (referred as "Ottawa"), we captured another two scenes (with a capturing distance of 0.5 meter) for comparison, which we refered as "SE Lab" and "MTS Lab". Due to space limitations, the visual results are not presented here. From the tables, we can see that our results have relatively higher PSNR values and SSIM scores. It is also noted that although the result from Zhang et al. [10] has only slightly lower scores than ours, it has severe artifacts in visual appearance.

3) Our Method v.s. Google Street View: We now make a comparison with Google Street View navigation service for the aforementioned Google Street-level imageries. In the Google Street View service, the hopping navigation is to jump from one panorama to another, in which a zooming cross-fading effect and other visual cues are made to give the user a sense of movement. However, the transition is over-blurred and has ghosting effects (see the first row in Figure 11), and it is not that easy to distinguish the scene objects in those intermediate frames. On the contrary, the video navigation provided by our method (the second row in Figure 11) makes view interpolation between adjacent panoramas and gives a more pleasing viewing experience. What's more, our video navigation can provide a more flexible control. Users can change the view direction when moving from one panorama to the adjacent one, which is not supported by Google Street View. Both the side-by-side comparison and flexible navigation control can be found in the supplementary video.

TABLE I
PSNR COMPARISON

	Our Method	[9]	[13]	[10]
Ottawa	**24.38**	24.06	19.93	24.35
SE Lab	**23.06**	22.39	19.93	N/A
MTS Lab	**20.43**	19.03	17.00	N/A

TABLE II
SSIM COMPARISON

	Our Method	[9]	[13]	[10]
Ottawa	**0.8259**	0.8225	0.6231	0.8209
SE Lab	**0.7891**	0.7563	0.6368	N/A
MTS Lab	**0.6703**	0.6244	0.4796	N/A

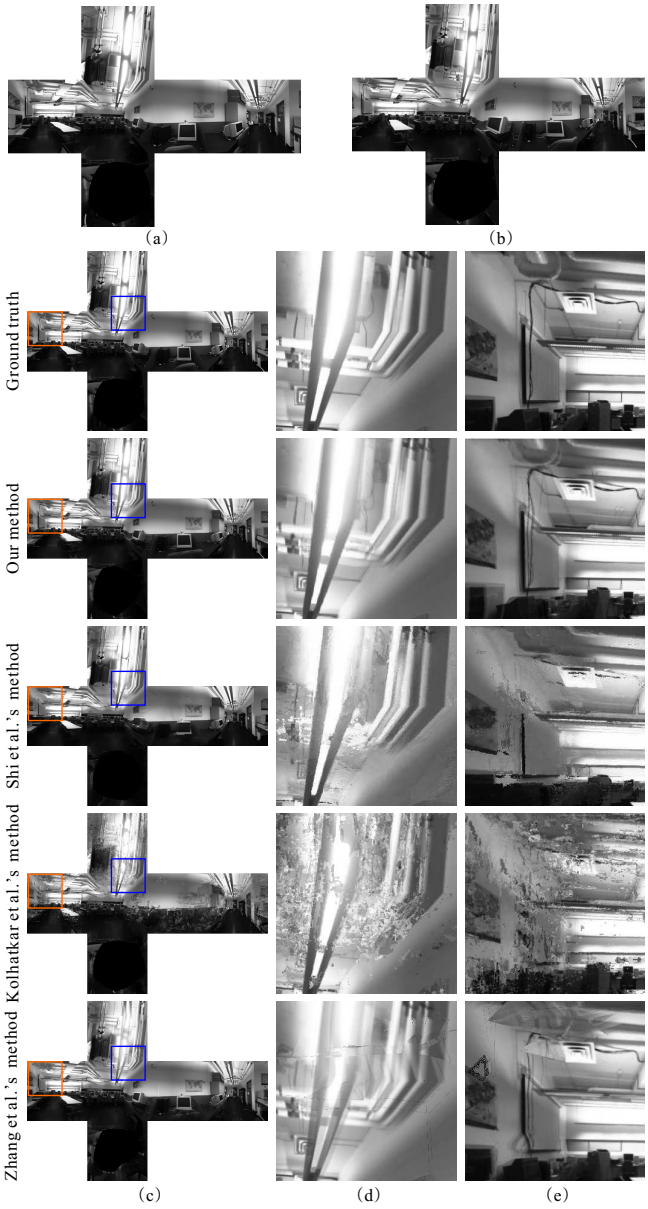


Fig. 10. Comparison between different methods. (a) and (b) are the reference panoramas; (c), (d), (e) Starting from the second row are the ground truth, our result, and those from [9], [13] and [10] at $\mathbf{t} = 0.5$, respectively. Obviously, our method generates a more pleasing result, and more visually similar to the ground truth.

A close inspection to Figure 11 and the video reveals that our method still suffers from some artifacts. This is due to two reasons. First, the street-level scene contains buildings with symmetric structures or repetitive elements, making it difficult to find correct matches. Second, moving objects on the street may cause large overlaps. Interestingly, although there are artifacts, our synthesized video still looks more temporally smooth in a global sense than the transition effect in Google Street View. This encourages us that the video navigation is a useful and promising control to current panorama-based navigation systems.

B. Timing Performance

Our system is built on a PC installed with Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz and NVIDIA GeForce GT 530. For the preprocessing stage implemented on CPU, we report the timing statistics for one typical case. It usually takes about 3 seconds to find reliable correspondences from two reference panoramas with a face size of 256×256, and about 16 seconds for the triangulation procedure, when there are about 950 triangles. The timing will grow with the increase of the panorama's resolution, the number of feature points, and/or the number of triangles.

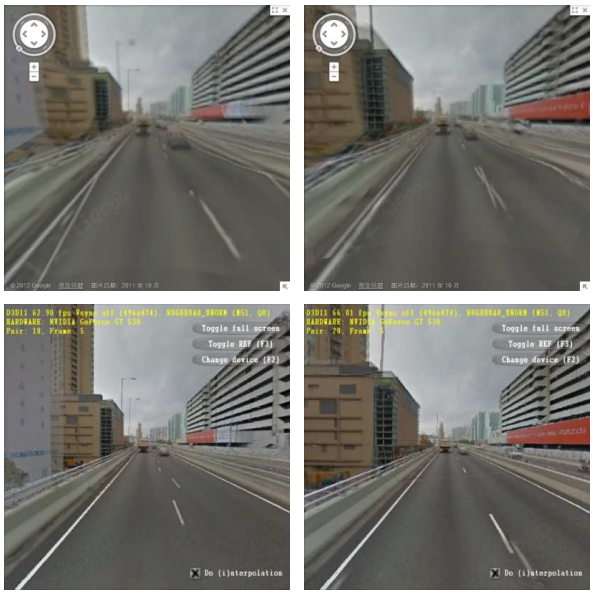


Fig. 11. Snapshots of the video navigation provided by Google Street View (the first row) and our method (the second row). Since we do not know the exact viewing parameters of Google Street View, there exist some differences between the fields of view.

For the online stage, we collect the timing statistics by accounting two major factors, i.e., the number of triangles and the face resolution of cubic panoramas. As shown in Figure 12, the timing curves are almost flat against the increase of the triangle number, and grow with the increase of the cube face resolution. For our CPU implementation, it takes about 8.305 seconds to generate a cubic panorama with a face size of 512×512 pixels. Our GPU implementation only needs 0.0172 seconds (58 fps). The speed up is about 480 times. It is obvious that our GPU implementation successfully achieves a real-time performance.

We now report the timing performance of previous methods. For the method of [13], which is based on optical flow, it can achieve interpolation at about 463 fps for cubic panoramas with a face size of 512×512. However, its quality is less satisfactory, as discussed previously. For the method of [9], our implementation takes about 239.47 seconds to generate a cubic panorama with a face size of 512×512 pixels. For the method of [10], it took several minutes for the interpolation step, which was reported by the authors, while their machine is slightly slower than ours.

C. Discussion

In our experiments, we find that the interpolation quality is related to the accuracy of correspondences, which is always challengeable. We also find several situations in

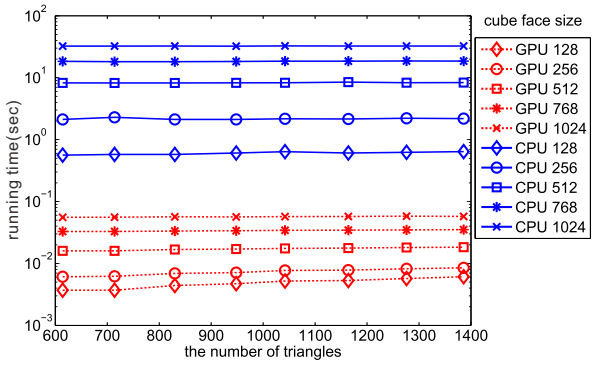


Fig. 12. Running time with the increasing number of triangles and panorama resolutions. The logarithmic scaling is used for the time axis due to the large value range.

which to get correct corresponding matches using SIFT will be quite difficult. For instance, there are many objects with self-similar structures; the panorama is covered by too many trees; or the change of the depth range in the pair of panoramas is too large. However, the comparison between several state-of-the-art feature detectors tells us that SIFT gets better interpolation results, as Figure 13 shows. Hence, it is one of our future work that investigates more advanced feature detectors.

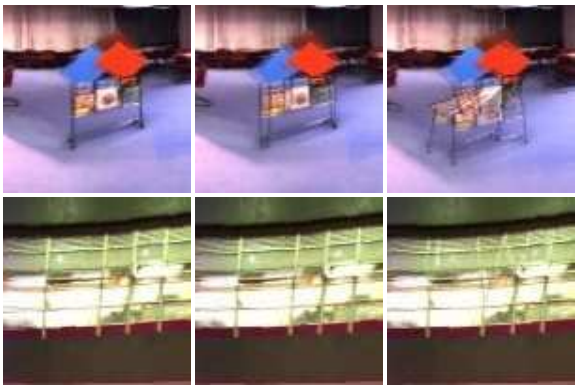


Fig. 13. Comparison between different feature detectors. (a) SIFT[28]; (b) SURF[33]; and (c) BRISK[34].

Secondly, although the rematching in the triangulation can solve most of overlaps, it sometimes mistakenly discard correct matches, which will cause blurry artifacts in the interpolated panoramas. One possible solution is to discard all the feature points covered by the unmatched regions and retriangulate the panoramas. This, however, may reduce the number of the feature points significantly. One issue that we do not tackle in this paper is the visibility between triangles [11]. Interestingly, we have not observed the visibility problem in our experiments. This may be because the detected feature correspondences are rather sparse.

Currently, when we generate the video, we interpolate the same number of panoramas between each pair of reference images. This may result in varying paces when watching the video. The problem can be alleviated by taking the real physical distance between reference images into account, and adjusting the number of interpolated frames adaptively.

VII. Conclusion

In this paper, we present a novel algorithm, namely Cube2Video, to provide users a video-viewing experience when navigating cubic panoramas on-the-fly. Different from existing methods, we tackle the challenging discontinuity problem in cubic panoramas in an easy and unified way. Since the cube is a projective representation of the sphere, we extends the matching-triangulation-interpolation procedure on the spherical domain. A new angular error is proposed to improve the matching accuracy. After triangulating on the unit sphere, we construct a virtual camera model and develop a triangle-to-triangle transformation scheme for the interpolation. The temporal smoothness of the synthesized video is improved via introducing a compensation transformation. In addition, we exploit the cubemap hardware support and the parallel computing power of GPU, thus achieving real-time video viewing. Experiments show that our method can get pleasant results, even for panoramas with large displacements.

Our system can be further improved in several ways. Firstly, line correspondences [35] can be included to further improve the accuracy of the feature correspondences. Such information may also serve as constraints and guide the triangulation procedure. Secondly, accounting the physical distances between panoramas may help us to obtain a more flexible control of the video navigation pace.

Acknowledgment

We would like to acknowledge Google, VIVA lab of University of Ottawa and SIAT for the panoramic data, we also thank Chunxiao Zhang for helpful discussion and experimental result. The work is supported by the National Natural Science Foundation of China (61100122 and 61100121), New Century Excellent Talents in University (NCET-11-0365), Tianjin Science Foundation for Youth (12JCQNJC00100), the research fund from The Tianjin Key Lab for Advanced Signal Processing, Civil Aviation University of China (TJKLASP-2012-2), General Research Fund of Hong Kong (CUHK 417411), and a research grant CUHK SHIAE 8115034.

References

[1] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver, "Google Street View: Capturing the world at street level," *Computer*, vol. 43, no. 6, pp. 32–38, 2010.

[2] M. Corporation, "Bing maps driving directions, traffic and road conditions," http://www.bing.com/maps/explore.

[3] S. E. Chen, "Quicktime vr-an image-based approach to virtual environment navigation," in *Siggraph*, August 1995, pp. 29–38.

[4] X. Guan, L. K. Shark, G. Hall, and W. Deng, "Distortion correction for immersive navigation in spherical image environment," in *International Conference on CyberWorlds*, 2009, pp. 96 – 101.

[5] L. McMillan and G. Bishop, "Plenoptic modeling: An image based rendering system," in *Siggraph*, August 1995, pp. 39–46.

[6] Y.-F. Chan, M.-H. Fok, C.-W. Fu, P.-A. Heng, and T.-T. Wong, "A panoramic-based walkthrough system using real photos," in *Proceedings of the 7th Pacific Conference on Computer Graphics and Applications (PG)*, 1999, pp. 231–240.

[7] K. H. Yeung, "Panorama interpolation for novel view composition," Master's thesis, University of Hong Kong, August 2000.
[8] D.-H. Kim and J.-S. Choi, "View transition algorithm for the walkthrough on the panorama based navigation," in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2004, pp. 1109–1114.
[9] F. Shi, R. Laganiere, E. Dubois, and F. Labrosse, "On the use of ray-tracing for viewpoint interpolation in panoramic imagery," in *Proceedings of the Canadian Conference on Computer and Robot Vision (CRV)*, may 2009, pp. 200 –207.
[10] C. Zhang, Y. Zhao, and F. Wu, "Triangulation of cubic panorama for view synthesis," *Applied Optics*, vol. 50, no. 22, pp. 4286–4294, August 2011.
[11] C.-W. Fu, T.-T. Wong, and P.-A. Heng, "Computing visibility for triangulated panoramas," in *Proceedings of Eurographics Rendering Workshop*, 1999, pp. 169–182.
[12] W. Chen, W. Xu, Z. Xiong, and M. Zhang, "View synthesis for realistic virtual walk through based on omni-directional images," *The International Journal of Virtual Reality*, vol. 8, pp. 87–92, 2009.
[13] S. Kolhatkar and R. Laganière, "Real-time virtual viewpoint generation on the gpu for scene navigation," in *Proceedings of the Canadian Conference on Computer and Robot Vision (CRV)*, 2010, pp. 55–62.
[14] A. Siu and R. Lau, "Image registration for image-based rendering," *IEEE Transactions on Image Processing*, vol. 14, no. 2, pp. 241 –252, 2005.
[15] S. E. Chen and L. Williams, "View interpolation for image synthesis," in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques (Siggraph '93)*, 1993, pp. 279–288.
[16] C. L. Zitnick and S. B. Kang, "Stereo for image-based rendering using image over-segmentation," *International Journal of Computer Vision*, vol. 75, no. 1, pp. 49–65, 2007.
[17] M. Goesele, J. Ackermann, S. Fuhrmann, C. Haubold, R. Klowsky, D. Steedly, and R. Szeliski, "Ambient point clouds for view interpolation," in *ACM SIGGRAPH 2010 papers*, 2010, pp. 95:1–95:6.
[18] S. M. Seitz and C. R. Dyer, "View morphing," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (Siggraph '96)*, 1996, pp. 21–30.
[19] Y. Wexler and A. Shashua, "On the synthesis of dynamic scenes from reference views," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2000, pp. 576–581.
[20] R. Manning and C. Dyer, "Interpolating view and scene motion by dynamic view morphing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 1999, pp. 388–394.
[21] J. Xiao, C. Rao, and M. Shah, "View interpolation for dynamic scenes," in *Euro Graphics*, 2002, pp. 153–162.
[22] M. Lhuillier and L. Quan, "Image interpolation by joint view triangulation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1999, pp. 139–145.
[23] A. M. K. Siu and R. W. H. Lau, "Relief occlusion-adaptive meshes for 3d imaging," in *Proceedings of IEEE International Conference on Multimedia and Expo(ICME)*, 2003, pp. 101–104.
[24] M. J. Tsai, C. L. Kao, and J. Liu, "The gentle spherical panorama image construction for the web navigation system," in *International Conference on Acoustics Speech and Signal Processing(ICASSP)*, 2010, pp. 1578 – 1581.
[25] B. Chen, B. Neubert, E. Ofek, O. Deussen, and M. F. Cohen, "Integrated videos and maps for driving directions," in *Proceedings of the 22nd annual ACM symposium on User interface software and technology (UIST)*, 2009, pp. 223–232.
[26] C. Peng, B.-Y. Chen, and C.-H. Tsai, "Integrated google maps and smooth street view videos for route planning," in *2010 International Computer Symposium (ICS)*, December 2010, pp. 319 –324.
[27] J. Lim, N. Barnes, and H. Li, "Estimating relative camera motion from the antipodal-epipolar constraint," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 10, pp. 1907 –1914, oct. 2010.
[28] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.
[29] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge University Press, 2003.
[30] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[31] M. Domin, S. Langner, N. Hosten, and L. Linsen, "Direct glyph-based visualization of diffusion mr data using deformed spheres," in *Visualization in Medicine and Life Sciences*, 2008, pp. 185–204.
[32] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600 –612, april 2004.
[33] H. Bay, T. Tuytelaars, and L. Gool, "Surf: Speeded up robust features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, vol. 3951, 2006, pp. 404–417.
[34] S. Leutenegger, M. Chli, and R. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, November 2011, pp. 2548 –2555.
[35] B. Fan, F. Wu, and Z. Hu, "Line matching leveraged by point correspondences," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, june 2010, pp. 390 –397.

PLACE PHOTO HERE

Qiang Zhao received the B.S. degree in software engineering from the School of Computer Software, Tianjin University (TJU), P.R. China, in 2009. Currently he is pursuing the Ph.D. degree with the School of Computer Science and Technology, Tianjin University (TJU), P.R. China. His main research interests include view interpolation and feature detection.

PLACE PHOTO HERE

Liang Wan (M'08) received the B.Eng and M.Eng degrees in computer science and engineering from Northwestern Polytechnical University, P.R. China, in 2000 and 2003, respectively. She obtained a Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong in 2007. She is currently an Associate Professor in the School of Computer Software, Tianjin University, P. R. China. Her research interest is mainly on computer graphics, including image-based rendering, non-photorealistic rendering, pre-computed lighting, and image processing.

PLACE PHOTO HERE

Wei Feng (M'10) received the B.S. and M.Phil. degrees in Computer Science from Northwestern Polytechnical University, China, in 2000 and 2003 respectively, and the Ph.D. degree in Computer Science from City University of Hong Kong in 2008. From 2008 to 2010, he worked as research fellow at the Chinese University of Hong Kong and City University of Hong Kong, respectively. He is currently an associate professor in school of computer science and technology, Tianjin University. His major research interest is media computing, specifically including general Markov Random Fields modeling, discrete/continuous energy minimization, image segmentation, semi-supervised clustering, structural authentication, and generic pattern recognition. He got the support of the Program for New Century Excellent Talents in University, China, in 2011.

PLACE PHOTO HERE

Jiawan Zhang received the BSci, MPhil, and PhD degrees in computer science from Tianjin University in 1997, 2001, and 2004, respectively. Currently, he is working as a professor in the School of Computer Software, Tianjin University. His main research interests include computer graphics, visual analytics and digital culture heritage. He is a member of the IEEE and ACM, and a senior member of China Society of Image and Graphics, a senior member of China Computer Federation, and a co-chair of Tianjin Society of Image and Graphics.

Tien-Tsin Wong (M'94) received the BSci, M-Phil, and PhD degrees in computer science from the Chinese University of Hong Kong in 1992, 1994, and 1998, respectively. Currently, he is working as a professor in the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His main research interests include computer graphics, including perception graphics, computational manga, image-based rendering, GPU techniques, natural phenomena modeling, and multimedia data compression. He received the IEEE Transactions on Multimedia Prize Paper Award 2005 and Young Researcher Award 2004. He is a member of the IEEE and the IEEE Computer Society, a senior member of ACM, and a fellow of HKIE.