Content Adaptive Tiling Method Based on User Access Preference for Streaming Panoramic Video

Zhengzhong Tu
State Key Lab of ASIC and System
Fudan University
Shanghai, China
16210720199@fudan.edu.cn

Tongyu Zong
State Key Lab of ASIC and System
Fudan University
Shanghai, China
16210720212@fudan.edu.cn

Xueliang Xi
China North Industries Group Corporation
Beijing, China
xlxi_cnigc@163.com

Li Ai
China North Industries Group Corporation
Beijing, China
liai_cnigc@163.com

Yize Jin
State Key Lab of ASIC and System
Fudan University
Shanghai, China
14210720072@fudan.edu.cn

Xiaoyang Zeng
State Key Lab of ASIC and System
Fudan University
Shanghai, China
xyzeng@fudan.edu.cn

Yibo Fan
State Key Lab of ASIC and System
Fudan University
Shanghai, China
fanyibo@fudan.edu.cn

***Abstract*—Tiled streaming has been proposed for delivering ultra-high resolution videos such as zoomable online lectures or panoramas. In tiled streaming, the source video is first partitioned into grid of small rectangular tile groups. Each tile group is independently encoded and compressed. When an user asks for a certain viewport at a time, the server only streams the viewed tiles to save up bandwidth. However, not much work has been done on finding the best tiling method for streaming panoramic video. This paper proposes an effective tiling algorithm for tiled streaming by using both video content and user access preference history. Experimental results show that the proposed tiling method can save up to 32.4% and 69.8% of average streamed bitrate compared to conventional uniform tiling scheme and simply streaming the entire panorama respectively on equi-rectangular panoramic video.**

Keywords—Tiled streaming; panoramic video; tiling; streaming

I. Introduction

Recently, an increasing interest in study on Virtual Reality (VR) has emerged since many VR applications and techniques have been prevalent all over the world both in academia and industry. Virtual Reality (VR) involves a large range of fields including computer vision, computer graphics, video encoding, video streaming, human-machine interaction and so on. Generally, an interactive VR streaming system often contains the following four parts: Video capturing, Projective encoding, Interactive streaming and Stereo rendering.

This paper focuses on the streaming aspect of an interactive panoramic streaming system. Actually, some advanced online social multimedia service providers such as YouTube [1] and Facebook [2] are currently supporting VR video streaming for VR headsets. The main character of streaming VR video is the user only asks for some part of the whole video to display on screen at a time. Therefore, tiled streaming is introduced [5] to save the bandwidth for streaming ultra-high resolution video such as panoramas. As shown in Fig. 1, the source video is firstly divided into grid of video segments such as 32 tiles (4 × 8). Given the user's required viewport, the server delivers streams overlapped with the user's viewable region. The client side then retrieves partial view with the received bit stream.

However, there is a trade-off in tiled streaming. The streams delivered usually cannot exactly cover a certain viewport, so redundant data outside the viewport exists. Streaming these redundant parts definitely cause a large waste of bandwidth. If the video is partitioned into smaller tiles, less redundancy can be achieved whereas the compression efficiency drops. On the contrary, larger tile size results in better compression efficiency but more redundant data.

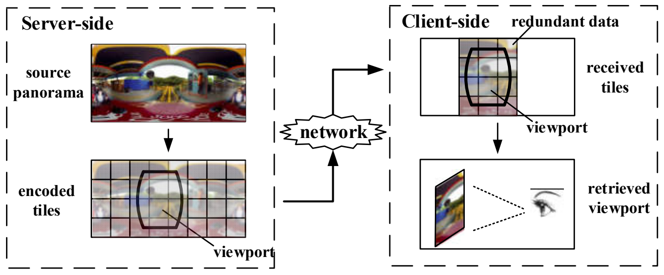


Fig. 1. Tiled streaming in an interactive VR streaming system.

Not much work has been done on optimizing streaming panoramic videos. However, there already existed some works studying on issues about delivering online pan-tilt-zoom (PTZ) video lectures. Some works utilize the concept of region-of-interest (ROI) which denotes the user access frequency to the video [3], [4]. In [3], Ngo et al. proposed two methods supporting zoomable video streaming called tiled streaming and monolithic streaming respectively. Their later work [4] exploited user access patterns and encode different regions of the video with different encoding parameters based on the

This work was supported in part by the National Natural Science Foundation of China under Grant 61674041, in part by the STCSM under Grant 16XD1400300, in part by the State Key Lab of ASIC & System under Grant 2015MS006.

popularity of the region. They show that their adaptive tiling method can reduce the expected bandwidth by up to 27%.

As for panoramic videos, article [5] addressed different tiling method in a real-time interactive panoramic video system based on different predictive models of movement of users' viewport. As for optimizing on tile size, [7] proposed a method to find the most efficient tile size for cylinder panoramic video coding subject to the target bandwidth. Yu et al. [11] proposed a content adaptive tiling method for cinematic virtual reality. Experiments showed average bitrate savings of over 18% relative to the baseline equal-area representation on an image dataset. But Yu's tiling method is not good enough to be used in tiled streaming because they only considered tiling in latitude. Zare et al. [6] proposed to store two versions of the same video content at different resolutions in order to solve the latency problem. The results indicated bitrate saving from 30% to 40% when compared to streaming the entire video content.

This paper points at how to find the best tiling method to reduce the streamed bitrate. In this paper, we propose a new tiling method for tiled streaming using both video content and user access preference. Experiments show that our scheme can save up to 32.4% of the average bitrate compared to uniform tiling method on equi-rectangular videos. To begin with, we would like to present the tiling problem to be solved in this paper in the following section.

The rest of this paper is organized as follows. Section II addresses the tiling problem we would like to solve in this paper. Section III presents the proposed tiling algorithm. Experimental results are shown in section V and finally the conclusion is given in section VI.

II. Tiling Problem

In this section, we will formulate the tiling problem studied in this paper as well as some background knowledge and notations related to the problem.

Suppose a web-based panoramic video delivering system providing video-on-demand (VoD) service for large quantities of users. Given a panoramic source video, e.g. one in equi-rectangular format, we recorded all the user historical viewpoint (center of a user's view) access frequency for every frame. Then we got frame-level user viewpoint access distribution which was called access probability map.

Consider the simplest case, for example, one frame instead of a video. We denote the frame we are interested in tiling as I. The probability map P has been obtained previously according to the overall historical access records of all the users. A tile map T of I consists of a set of non-overlapping rectangles, called tiles $t_1, t_2, \dots t_n$. Each tile t_i is contained in I and all the tiles collectively cover exactly I. The probability of every viewpoint v_i or the view window of viewpoint v_i in probability map P can be directly calculated through function $p(v_i)$ with the probability map P. A set of tiles in tile map T overlapped with the field-of-view (FOV), i.e. view-window/viewport, of viewpoint v_i will be streamed when a user chooses viewpoint v_i. The cost function $c(t)$ is assigned to get the bitrate of a tile. Streamed bitrate to user i who chooses viewpoint v_i can be computed by adding the bitrate of all the overlapped tiles. Assume an ideal case that the client chooses some viewport and instantly receives the overlapped tiles to retrieve the viewport. So for every viewpoint in the probability map, we can compute the total bitrate of required tiles as streamed bitrate. The tiling problem is to minimize the average streamed bitrate of all the possible viewpoints in probability map P, i. e. to minimize:

$$\sum_{v_i \epsilon P} \left\{ p(v_i) \sum_{FOV(v_i) \cap t_i \neq \emptyset} c(t_i) \right\} \tag{1}$$

It can be easily proofed that this expression is equivalent to :

$$\sum_{t_i \epsilon T} \left\{ c(t_i) \sum_{FOV(v_i) \cap t_i \neq \emptyset} p(v_i) \right\}$$

Where t_i denotes each tile contained in the tile map T, and $c(t_i)$ indicates its bitrate. $\sum_{FOV(v_i) \cap t_i \neq \emptyset} p(v_i)$ adds the probability of all the viewpoints whose view window overlaps with tile t_i to calculate the access probability of a tile t_i according to viewpoint access probability map P.

Adaptive tiling is proposed by Ngo et al. [4] to solve the similar kind of problem but for zoomable online video lectures. They used a greedy heuristic to find a tile map to reduce the expected bandwidth. However, their method has several drawbacks. Firstly, they subjectively chose a traversing order (from top-left to bottom-right) to conduct their merging method, which is improper since the user's viewpoints tend to cluster around the central area of access probability map but exist sparsely near the edges of the map. Secondly, their merging method basically considered the cases of merging with right, bottom and diagonal neighbors. The one-sided growing direction cannot generate the optimal growing case of a tile. Furthermore, it is somewhat impractical to encode such small tiles on the fly by existing commercial encoders to obtain the bitrate of a tile. If they used an encoder that can truly encode a tile on the fly, the runtime of their algorithm was intolerably long.

III. Proposed Tiling Method

In order to get a more optimized solution, this paper proposes a new tiling method. We start with breaking the tile map into uniform tiles as small as possible and conduct a Bottom-Up growing procedure. The key idea of this algorithm is that the optimal growing case is always allowed to grow prior. To get the global optimum, the optimal growing case of each tile is defined by merging the tile with its omnidirectional neighbors. In order to enhance the processing efficiency of the program, we propose a function to estimate the bitrate of a tile. In order to elaborate the proposed tiling scheme more concretely, this section is divided into subsections A, B, C and D. Subsection A illustrates the bitrate-estimating function and B explains the probability-calculating function. Section C explains the nearest growing method of one tile. Finally, subsection D presents the whole tiling algorithm based on A, B and C.

A. Bitrate-estimating function

The proposed tiling method requires calculating the bitrate of various blocks of different size contained in the source frame.

We regard the blocks as intra-frame and use intra-frame encoding by HEVC Test Model (HM) to get the real bitrate of the blocks. Inspired by [8], the average gradient of a block is used to measure its complexity. We consider the relationship among bitrate and (block-size & block-complexity) as:

$$c(t) = F(s_t)H(g_t) \tag{2}$$

Where F is the function between bitrate and image area (s_t), and H denotes the relationship between bitrate and image complexity (g_t) which is measured by gradient. Large quantities of experiments were carried to fit the function F and H by decoupling fitting method. That is, we first find the relationship H and then utilize H to fit function F.

B. Probability-calculating function

To get the viewed probability of tile t, we use a probability function $p(t)$ to compute the access probability. With the user viewpoint probability map, access probability of tile t can be obtained by computing the sum of viewpoint probability whose view-window/field-of-view (FOV) overlaps with tile t. Note that if the probability map is in equi-rectangular format, the viewpoint should be projected on a sphere first and then the field-of-view (FOV) of the viewpoint is back-projected to planar FOV to judge if it overlaps with tile t. Fig. 2 shows the projecting as well as the back-projecting procedure. By adding all probabilities of viewpoint O' whose FOV overlaps with tile t, the viewed probability of tile t is acquired.

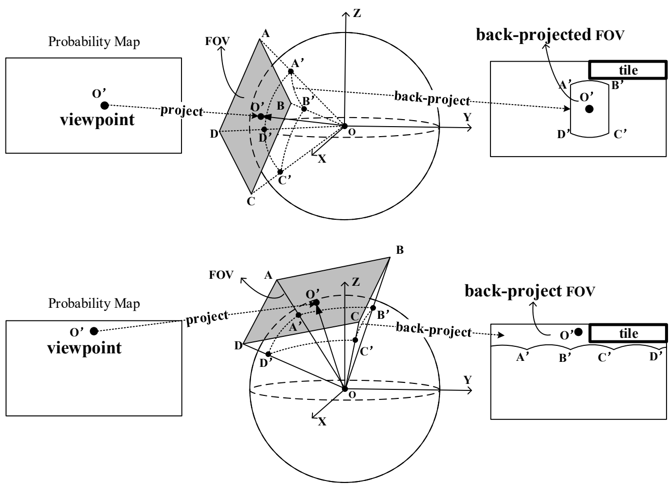


Fig. 2. Definition of Viewport/FOV/view window of viewpoint in equi-rectangular map

The top diagram in Fig. 2 presents a non-overlapping case while the bottom shows an overlapping case. It should be noted that $FOVs$ tangential to the sphere at points of different location could have disparate shapes of back-projected area in equi-rectangular map, which is obviously shown in Fig. 2. This projecting procedure is also adaptive to cube-map [11] if the probability-calculating function $p(t)$ is substituted with new one.

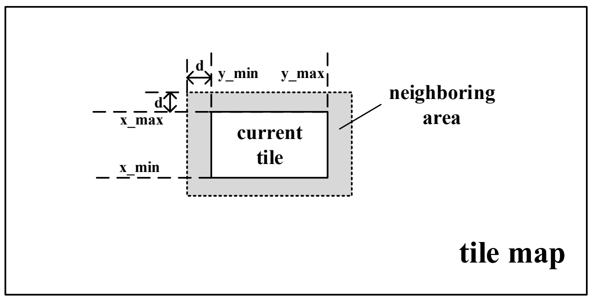
C. Nearest growing choice

The nearest growing neighbors should be defined first to illustrate the growing method. Given a tile $t_{current}$, we designate its feature coordinate as $\{x_{min}, x_{max}, y_{min}, y_{max}\}$. According to the above defined coordinates, we can find all the possible growing neighbors of tile $t_{current}$. Given any other tile t_{other} with its analogously defined feature coordinate $\{x'_{min}, x'_{max}, y'_{min}, y'_{max}\}$, if the coordinates of the above mentioned two tiles satisfy any one of the following four restrictions, t_{other} is said to belong to nearest growing neighbors of $t_{current}$:

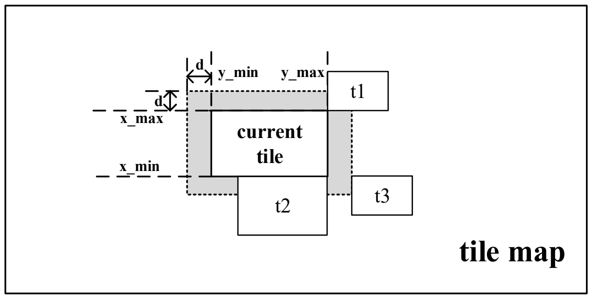
$$\begin{aligned} x_{min} - d \le x'_{min} \le x_{max} + d \\ x_{min} - d \le x'_{max} \le x_{max} + d \\ y_{min} - d \le y'_{min} \le y_{max} + d \\ y_{min} - d \le y'_{max} \le y_{max} + d \end{aligned} \tag{3}$$

Where d denotes the width of nearest neighboring area. Fig. 3(a) shows the definition of $\{x_{min}, x_{max}, y_{min}, y_{max}\}$ and d. The gray area represents neighboring area of $t_{current}$. Intuitively, if t_{other} overlaps with neighboring area of $t_{current}$, t_{other} is said to be one of nearest growing neighbors of $t_{current}$. Fig. 3(b) shows the neighboring and non-neighboring examples of $t_{current}$. t_1 and t_2 are members of nearest growing neighbors of $t_{current}$, whereas t_3 is not. If $t_{neighbor}$ belongs to neighbors of $t_{current}$, $t_{neighbor}$ as well as $t_{current}$ is able to constitute a big merged tile t_{merged}. The potential merged tile t_{merged}'s feature coordinate $\{x''_{min}, x''_{max}, y''_{min}, y''_{max}\}$ can be computed by coordinates of t_{other} and $t_{current}$ as shown in expression (4).

$$\begin{cases} x''_{min} = \min\{x_{min}, x'_{min}\} \\ x''_{max} = \max\{x_{max}, x'_{max}\} \\ y''_{min} = \min\{y_{min}, y'_{min}\} \\ y''_{max} = max\{y_{max}, y'_{max}\} \end{cases} \tag{4}$$



(a) Definition of neighboring area of current tile



(b) Examples of neighboring and non-neighboring tiles of current tile. t_1 and t_2 are neighboring tiles, while t_3 is not.

Fig. 3. Definition and relationship of neighboring tiles

We use function $c(t)$ and $p(t)$ introduced in section A and B respectively to compute the bitrate and probability of tile $t_{current}$, $t_{neighbor}$ and t_{merged}. Then the product of bitrate and probability of a tile is calculated as its expected bandwidth. After finding out all nearest neighbors of $t_{current}$, we can compute the Normalized Growing Speed (NGS) of each $t_{neighbor}$ by the following formula:

$$NGS = \frac{\left(\sum_{t_i \in t_{merged}} c(t_i) \cdot p(t_i) - c(t_{merged}) \cdot p(t_{merged})\right)}{area(t_{merged})} \quad (5)$$

Where $t_i \in t_{merged}$ denotes any tile that is totally contained in t_{merged}. After seeking out all the growing cases, we choose the case of largest NGS as the optimal one-step nearest growing choice of tile $t_{current}$. If the largest NGS of $t_{current}$ is negative, this kind of case is abandoned.

D. *Proposed tile-growing method*

Based on the above explanations and notations, we are now to present the proposed tiling algorithm. The pseudocode of the proposed method is shown in Fig. 4. We begin with a tile map divided uniformly into grid of small tiles such as 10×20 tiles. In the first loop, each of the 200 tiles will be traversed to find the optimal growing choice of each tile. Then the NGS of every tile is compared to find the global optimal NGS as the consequential growing choice of the first solution, according to which we update the whole tile map. Afterwards, the big merged tile as well as other unmerged tiles is again traversed. The globally largest NGS is once again selected as the second growing solution. This iterative process will continue until no more possible growing cases can be found. The resulting tile map acquired is the optimized tile partition method for this sequence. This step-by-step growing algorithm guarantee that more optimized merging situations will always have priority to emerge than less optimized ones, which will result in a solution tile map closed to the optimal.

Algorithm: Proposed Tile-Growing Method

```
1.  Input: probability map P and image frame I
2.  Initialize tile map T with m × n tiles
3.  Do
4.      Find all tiles including grown tiles in tile map T
5.      For every tile $t_i$ do
6.          Find all the nearest growing neighbors of tile $t_i$
7.          Calculating NGS of each neighbor by Eq. (5)
8.          Find the largest NGS as the best growing case
9.          Judge if the largest NGS is positive
10.     End for
11.     Compare NGS of all the tiles in tile map T and
        select the largest NGS globally to update the tile
        map T
12. Until: tile map T remains unchanged
13. Output: resulting tile map T
```

Fig. 4. The pseudocode of proposed tiling algorithm.

IV. Experimental Results

A. *Experimental environment*

The proposed tiling algorithm was estimated using 6 high resolution 4K images from SUN360 database [9]. The 6 sequences used in our experiments in 3840x1920 equi-rectangular format are respectively called *Building, GoldenHall, RollerCoaster, SnowField, Street, Indoors* as shown in Fig. 8(a). We recorded over 100 users' viewpoint access distribution of each sequence and generated 6 different viewpoint access probability maps. Fig. 5 shows one example of the probability map. We conducted the proposed algorithm on the sequences accompanied with probability maps using MATLAB R2015b. The tile maps were initialized into 10x20 evenly divided tiles and then processed by the proposed tile-growing method.

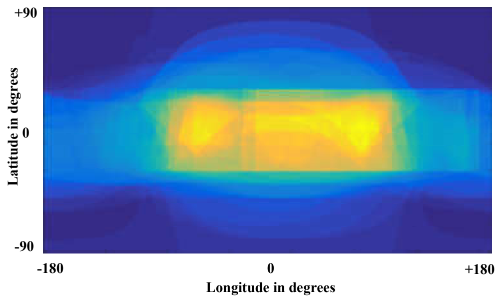


Fig. 5. The access probability map of *RollerCoaster*

B. *Resulting tile maps*

The resulting tile map are presented in Fig. 8(b). There are some observations from the tile maps that should be noted here. First, more popular regions tend to merge into bigger tiles, while less popular fields usually remain unchanged. Secondly, The frame content also influences tiled growing. Regions with convoluted textures are usually hard to form large tiles because that area is often not encoding-friendly. On the contrary, tiles with low image activity are more likely to merge with their neighbors.

C. *Average streamed bitrate*

In section IV(A), we utilize a bitrate-estimating function Eq. (2) for consideration of practical and run-time aspects. But actually, the accuracy of the function does not have any effect on estimating the performance of the proposed tiling method. Therefore, we can assume the proposed bitrate-estimating function is accurate and then assess the streamed bitrate using the same function. Since the goal of this paper is to minimize the expression shown in Eq. (1), the metric adopted for estimating tile maps can be directly computing the average streamed bitrate of a tile map by Eq. (1). Fig. 6 compares the proposed tiling method with adaptive tiling [4] and naive 10x20 tiling. Fig. 7 compares the proposed tiling method with Yu's [11] as well as streaming the entire video. Only the resampling method in [11] is adopted here for simplicity. Yu's tiling only considered tiling at vertical direction, so it did not improve much compared to equi-rectangular map in terms of tiling streaming. Experimental results show that the proposed tiling method achieves: 1) up to 69.8% and average 66.2% streamed bitrate saving relative to streaming the entire panorama; 2) up to 32.4% and average 16.5%

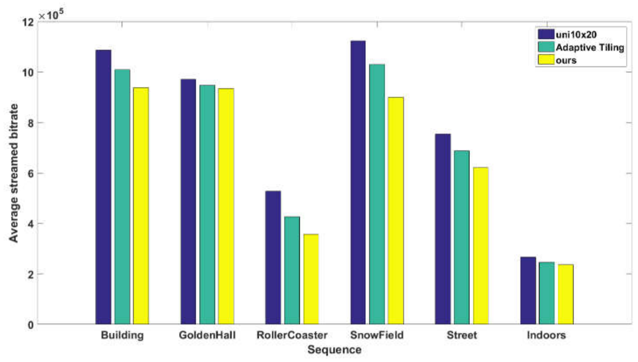


Fig. 6. Comparison of average streamed bitrate among: 1) Uniformly divided in 10x20 tiles; 2) Adaptive tiling in [4]; 3) The proposed tiling method.

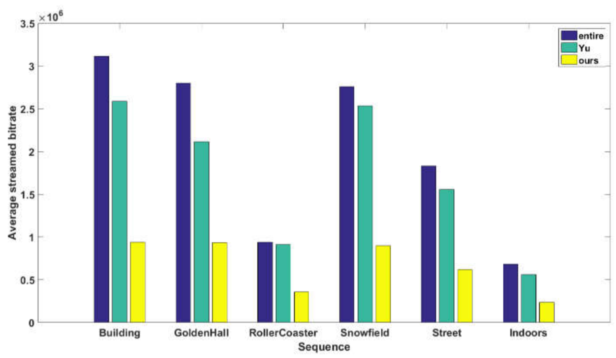


Fig. 7. Comparison of average streamed bitrate among: 1) Streaming entire panorama; 2) Yu's tiling method [11]; 3) The proposed tiling method.

bitrate saving compared to 10x20 uniform tile map; 3) up to 17.4% and average 8.5% bitrate saving compared to adaptive tiling [4].

V. Conclusion

In this paper, we propose a content adaptive tiling algorithm based on user view preference for server-client panoramic video streaming systems. Experimental results show that the proposed tiling method can save up to 32.4% and average 69.8% of average streamed bitrate when compared to naïve uniform tiling scheme and streaming the entire panorama respectively.

Future work targets on optimization of bitrate-estimating function for Group-of-Pictures and adaptive tiling method for panoramic video streaming. The effects of omnidirectional video on different formats are to be studied in further works.

References

[1] YouTube. [Online]. Available: https://www.youtube.com

[2] Facebook. [Online]. Available: https://www.facebook.com

[3] K. Q. M. Ngo, R. Guntur, A. Carlier, and W. T. Ooi, "Supporting zoomable video streams with dynamic region-of-interest cropping," in *Proceedings of the first annual ACM conference on Multimedia systems*. ACM, 2010, pp. 259–270.

[4] K. Q. M. Ngo, R. Guntur, and W. T. Ooi, "Adaptive encoding of zoomable video streams based on user access pattern," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 211–222.

[5] V. R. Gaddam, M. Riegler, R. Eg, C. Griwodz and P. Halvorsen, "Tiling in Interactive Panoramic Video: Approaches and Evaluation," in *IEEE Transactions on Multimedia*, 2016, *18*(9), pp. 1819-1831.

[6] A. Zare, A. Aminlou, M. Hannuksela, and M. Gabbouj, "HEVC-compliant Tile-based Streaming of Panoramic Video for Virtual Reality Applications," In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, October, pp. 601-605.

[7] F. Dai, Y. Shen, Y. Zhang, and S. Lin, "The most efficient tile size in tile-based cylinder panoramic video coding and its selection under restriction of bandwidth," In *International Conference on Multimedia and Expo*. IEEE, 2007, pp. 1355-1358.

[8] W. J. Kim, J. W. Yi, and S. D. Kim, "A bit allocation method based on picture activity for still image coding," In *IEEE transactions on image processing*. IEEE, 1999, *8*(7), pp. 974-977.

[9] J. X. Xiao, K. A. Ehinger, A. Oliva and A. Torralba, "Recognizing scene viewpoint using panoramic place representation," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2695-2702.

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

[11] M. Yu, H. Lakshman and B. Girod, "Content adaptive representations of omnidirectional videos for cinematic virtual reality," in *Proceedings of the 3rd International Workshop on Immersive Media Experiences*. ACM, 2015, pp. 1-6.

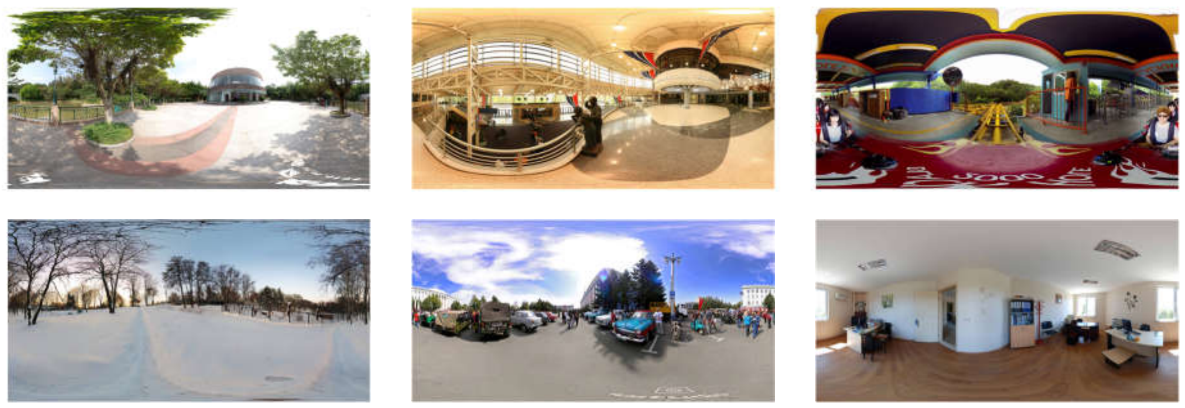


Fig. 8.   (a) Test sequences. Top row: *Building, GoldenHall, RollerCoaster*. Bottom row: *SnowField, Street, Indoors*.

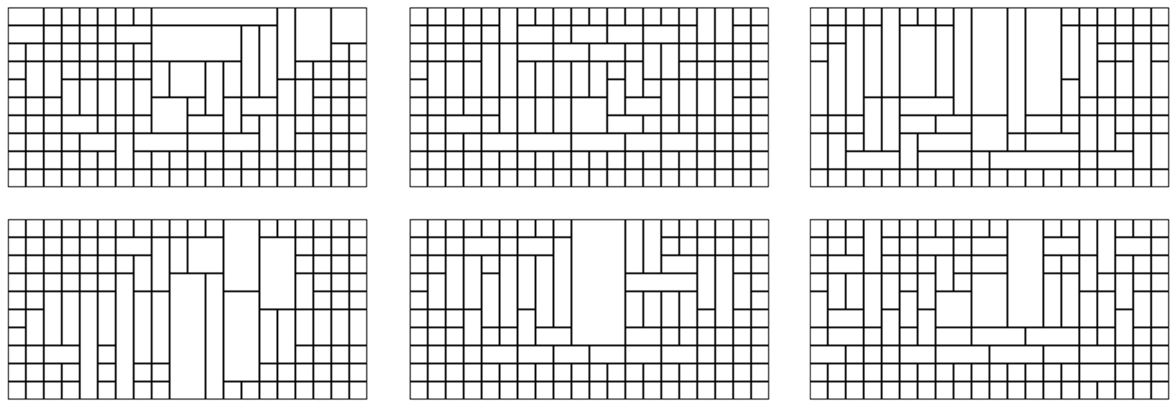


Fig. 8.   (b) Resulting tile maps. Top row: *Building, GoldenHall, RollerCoaster*. Bottom row: *SnowField, Street, Indoors*.