Optimal Seamline Detection in Dynamic Scenes via Graph Cuts for Image Mosaicking

Li Li, Jian Yao*, Haoang Li, Menghan Xia

Computer Vision and Remote Sensing (CVRS) Lab, School of Remote Sensing and Information Engineering, Wuhan University, Wuhan, Hubei, P.R. China

Wei Zhang

School of Control Science and Engineering, Shandong University, Jinan, Shandong, P.R. China

Abstract

In this paper, we present a novel method for creating a seamless mosaic from geometrically aligned dynamic images captured from the scene with dynamic objects at different times. The artifacts caused by dynamic objects and geometric misalignments can be effectively concealed in our proposed seamline detection algorithm. In addition, we simultaneously compensate the image regions of dynamic objects based on the optimal seamline detection in the graph cuts energy minimization framework and create the mosaic with a relatively clean background. To effectively ensure the high-quality of the optimal seamline, the energy functions adopted in graph cuts combine the pixel-level similarities of image characteristics, including intensity and gradient, and the texture complexity. To successfully compensate the image regions covered by dynamic objects for creating a mosaic with a relatively clean background, we initially detect them in overlap regions between images based on pixel-level and region-level similarities, then refine them based on segmentation and determine their image source in probability based on contour matching. We finally integrate all of these into the energy minimization framework to detect optimal seamlines. Experimental results on different dynamic scenes demonstrate that our proposed method is capable of generating high-quality mosaics with relatively clean backgrounds based on the detected optimal seamlines.

*Corresponding author.
Email address: jian.yao@whu.edu.cn (Jian Yao)
URL: http://cvrs.whu.edu.cn/ (Jian Yao)

Keywords: Image Mosaicking, Seamline Detection, Dynamic Object Compensation, Graph Cuts, Image Parallax

1. Introduction

Image mosaicking is an important and classical problem in the fields of image processing, photogrammetry, remote sensing and computer vision, which is used to blend two or multiple geometrically aligned images into a single composite image as seamlessly as possible, e.g., panorama stitching and satellite imagery mosaicking.

In ideally static scenes, both the photometric inconsistencies and the geometric misalignments are not existed or not obviously visible in overlap regions between images, the composite image mosaicked from multiple ones always looks very perfect. However, due to illumination variation and exposure differences, there always exist photometric inconsistencies to some extents between images captured from different viewpoints. The artifacts caused by photometric inconsistencies can be solved very well by color correction and smoothing transition (Levin et al., 2004; Brown and Lowe, 2007; Xiong and Pulli, 2009; Tao et al., 2013; Zhou et al., 2014), which try to conceal stitching artifacts by smoothing color differences between input images. In addition, due to the fact that input images are captured without the precisely common projection center, and the scenes have large depth differences with respective to the camera, or captured from dynamic scenes with moving objects at different times by a single camera mounted on a rotation platform, the geometric positions of corresponding pixels from different images may be different, especially for those pixels covered by dynamic objects. The smoothing transition technique can deal with the color differences along the seamlines, but can't handle the obvious parallax caused by geometric misalignment or dynamic objects. One efficient way to solve this problem is to detect the optimal seamlines between two images, and ensure that the optimal seamlines avoid crossing obvious dynamic objects and the regions with low image similarity and large object dislocation. If the seamlines and stitching artifacts are still visible due to color differences, the smoothing transition technique can be further applied to solve it easily. In this paper, our work focuses on the optimal seamline detection despite the presence of dynamic objects and geometric misalignments. In addition,

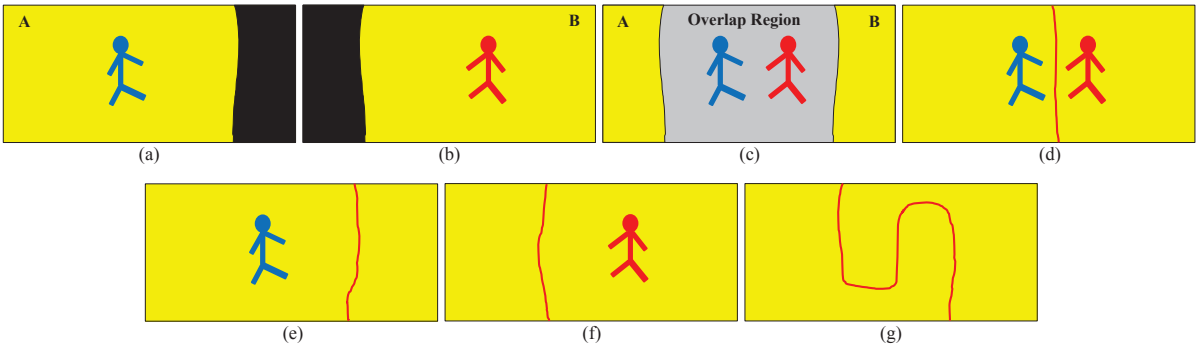


Figure 1: Two geometrically aligned input images **A** and **B** in (a) and (b) with a moving object in the overlap region as shown in (c). The image mosaics with different seamlines are presented in (d)-(g), which avoid crossing the moving object, like the work completed in (Davis, 1998; Mills and Dudek, 2009; Zeng et al., 2014). The moving object is visible in (d)-(f), especially in (d), the person appears twice in the last composite image. However, it is invisible in (g) by the compensation with the background image regions, which is the goal of our method.

by compensating the image regions of dynamic objects existing in one image with the background regions in another image, we can generate visually pleasant image mosaics with relatively clean backgrounds.

Optimal seamlines are generally located in overlap regions between images where their intensity or gradient differences are minimal. Many methods regarded the seamline detection as an energy optimization problem. They solved it by minimizing the specially designed energy function which is defined to represent the difference along the seamlines. For these methods, the key issues concentrate on how to define effective energy functions and how to guarantee the optimality of the solution. The energy functions are often defined by considering color, gradient and texture, and are optimized via different optimization algorithms, e.g., snake model (Kass and Witkin, 1988), Dijkstra's algorithm (Dijkstra, 1959), dynamic programming (Bellman, 1957), and graph cuts (Boykov et al., 2001; Boykov and Kolmogorov, 2004). Our proposed algorithm will apply the graph cuts algorithm to solve the problem of optimal seamline detection.

However, most stitching algorithms are designed for static scenes, maybe failed in dynamic scenes. If the scenes contain dynamic objects, ghosting effect would possibly appear in the last mosaicked images. Some algorithms (Davis, 1998; Hsieh, 2004; Boutel-

lier et al., 2007; Kim and Hong, 2008; Mills and Dudek, 2009; Tao et al., 2011; Zhi and Cooperstock, 2012; López et al., 2014; Zeng et al., 2014) have been designed for dynamic scenes to remove ghosting. Boutellier et al. (2007) proposed an image stitching algorithm to generate high-quality image mosaics from both video and separate images on mobile phones. To avoid the appearance of ghost caused by moving objects, it detected the moving objects by simply thresholding the difference map between two images. López et al. (2014) also applied this strategy to avoid the moving objects when mosaicking the images captured by mobile devices. Davis (1998) presented a system for creating pleasing mosaics in the presence of moving objects. It first generated the cost image by computing the intensity differences between corresponding pixels. Then it applied the Dijkstra's algorithm (Dijkstra, 1959) to search for an optimal seamline (minimum cost path) based on this cost image. However, if there are large exposure differences between source images, this method may be failed to find the optimal seamlines using intensity difference alone. To solve this problem, Mills and Dudek (2009) proposed to first compensate for exposure differences, and then find the optimal seamline via the Dijkstra's algorithm in the cost image defined by combining the image information of intensity and gradient. However, if the dynamic objects are similar with the background, the optimal seamline found by this algorithm may cross the dynamic object and the last composite image would contain ghosting effect. To improve the algorithm proposed by (Mills and Dudek, 2009), Zeng et al. (2014) proposed an new optimal searching criterion that combines gradient difference with edge-enhanced weighting intensity difference, which provides an effective mechanism for avoiding problems caused by moving objects. Then this algorithm applied dynamic programming (Bellman, 1957), which is more efficient than the Dijkstra's algorithm, to find the optimal seamline. This method could find the optimal seamlines in most dynamic cases. These existing methods can avoid crossing dynamic objects and produce high-quality image mosaics for most scenes. However, these methods can't remove dynamic objects from the finally mosaicked images and obtain the last background panorama, as shown in Figures 1(d)-(f), especially in (d), the moving object appears twice in the last panorama. To solve this problem, Zhi and Cooperstock

(2012) proposed an image stitching algorithm which is comprised of two stages. In the first stage, it discovered motions between input aligned images and extracted the moving region pairs through a multi-seed based region growing algorithm. In the second stage, with prior information provided by the extracted regions, it detected the optimal seamline by performing the graph cut optimization in gradient-domain, and simultaneously ensured that the pixels in each extracted region only come from one image. Since it used information from only one image for each extracted region pair, it can avoid that the moving object appears twice in the last composite image, and make it just like as if the image are captured without the moving objects in the scene. In addition, Kim and Hong (2008) proposed a background estimation algorithm and applied it to stitch the images captured form the dynamic scenes. This algorithm can generate the background as completely as possible, and also guarantee the completeness of moving objects if they move very slowly in the scene. The goal of our method is similar to this background estimation algorithm. However, this algorithm used multiple images for each regions to estimate the background, but we only use two images for each overlap regions.

In this paper, in order to avoid that the same dynamic object appears twice or multiple times in the same scene, and due to the demands of privacy protection, background subtraction and panorama scenery, we propose a novel optimal seamline detection method to create a seamless composite image with a relatively clean background (i.e., with dynamic objects as few as possible) and free from artifacts from geometrically aligned images, as shown in Figure 1(g). We formulate this problem as an energy optimization to find optimal seamlines in the graph cuts energy minimization framework. In our proposed method, not only the intensity and gradient differences are considered into the cost of each pixel in overlap regions between images, but also the texture complexity inspired by HOG (Histogram of Oriented Gradient) (Dalal and Triggs, 2005) is integrated into the cost function. In addition, to create image mosaics with relatively clean backgrounds, the dynamic objects are detected initially by computing the pixel-level similarity in the CIELAB color space and the region-level similarity between HOG feature descriptors, then refined by segments generated by the Mean Shift (Comaniciu and Meer, 2002), and

finally determined from which image they come in probability based on contour matching. All of these strategies are integrated for detecting optimal seamlines concealing the parallax and compensating the image regions covered by dynamic objects for image mosaicking, as shown in Figure 1(g).

The remaining part of this paper is organized as follows. Section 2 introduces the graph cuts energy minimization framework for finding optimal seamline between two images. The energy cost of each pixel by considering the intensity, gradient and texture complexity is defined in Section 3. Dynamic objects detection between two images is demonstrated in Section 4. Experimental results on indoor or outdoor scenes with dynamic objects are presented in Section 5 followed by the conclusion drawn in Section 6.

2. Seamline Detection via Graph Cuts

In overlap regions between two images, we regard each pixel as a node in the graph and assume that each node has four cardinal neighbors in the 4-neighborhood. The link between two adjacent nodes is regarded as an edge in the graph and its weight cost is defined as the sum of energy costs of these two nodes. The graph cuts algorithm is used to associate the label of each pixel to one of the input source images with the minimum energy cost.

2.1. Graph Cuts

Graph cuts (Boykov et al., 2001) is an efficient energy optimization algorithm to solve labeling problems. It has been widely used in many applications of image processing and computer vision, such as image matting, image segmentation, stereo matching, and image blending (Tao et al., 2011; Philip et al., 2015). For example, Rother et al. (2004) provided an powerful semi-automatic algorithm for foreground object extraction based on iterated graph cuts, named as "GrabCut". Kolmogorov and Zabih (2001) presented an energy minimization formulation of the correspondence problem with occlusions, and provided a fast approximation algorithm based on graph cuts. The basic idea is to first construct a weighted graph where each edge weight cost represents the corresponding cost energy value, and then find the minimum cut in this graph based on the max-flow

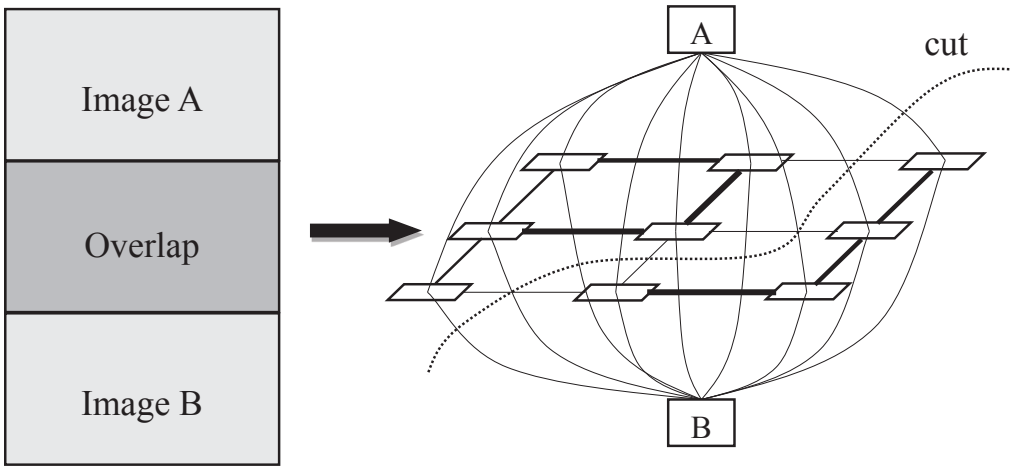


Figure 2: An illustration example of the optimal seamline detection method via graph cuts. The thickness of lines between adjacent pixels represents the value of the energy cost and the "cut" denotes the minimum cut, which means the optimal seamline.

or min-cut algorithm (Boykov and Kolmogorov, 2004). Let $\mathcal{P}$ be the set of all elements (i.e., vertexes in the graph), $\mathcal{N}$ be the set of all element pairs $\{p, q\}$ in the neighborhood (i.e., edges in the graph), and $\mathcal{L}$ be the set of all labels. The objective is to find a labeling map f assigning a label $f_p \in \mathcal{L}$ to each element $p \in \mathcal{P}$ by minimizing the following cost energy function:

$$E(f) = \sum\nolimits_{p\in\mathcal{P}} D_p(f_p) + \sum\nolimits_{(p,q)\in\mathcal{N}} V_{p,q}(f_p, f_q), \tag{1}$$

where $D_p(f_p)$ denotes the cost of assigning the label f_p to the element p and $V_{p,q}(f_p, f_q)$ defines the cost of assigning the labels f_p and f_q to the element pair p and q, respectively, which are often called as *the data energy term* and *the smooth energy term*, respectively. If f_p and f_q is equal, the value of $V_{p,q}(f_p, f_q)$ would be equal to 0, namely, the Potts model. The graph cuts algorithm can guarantee the global optimality of the last solution, which means that we can detect the optimal seamline globally. The time complexity of graph cuts is $O(n^2E)$, where n is the number of nodes and E is the number of edges.

2.2. Labeling via Graph Cuts

We formulate the optimal seamline detection as an energy minimization problem and use graph cuts to find the solution between pixels, as shown in Figure 2. Major steps of the proposed algorithm are simply described in Algorithm 1. For a composite image $\mathcal{I}$ from an input pair $(\mathbf{I}_1, \mathbf{I}_2)$ with an overlap, the energy cost $E(\mathcal{I})$ is comprised of the data energy term $E_{data}(\mathcal{I})$ and the smooth energy term $E_{smooth}(\mathcal{I})$. The data energy term represents all energy costs for individual pixel with one of input source images. The smooth energy term represents all energy costs between adjacent pixels. $E(\mathcal{I})$ is defined as:

$$E(\mathcal{I}) = E_{data}(\mathcal{I}) + E_{smooth}(\mathcal{I}), \tag{2}$$

where the data energy term $E_{data}(\mathcal{I})$ is defined as:

$$E_{data}(\mathcal{I}) = \sum\nolimits_{\mathbf{x}\in\mathcal{I}} \left(D_l^1(\mathbf{x}) + D_l^2(\mathbf{x})\right), \tag{3}$$

where $D_l^1(\mathbf{x})$ and $D_l^2(\mathbf{x})$ represent the costs of assigning the label of the pixel $\mathbf{x}$ to $\mathbf{I}_1$ and $\mathbf{I}_2$, respectively. Given a pixel point $\mathbf{x}$ in the composite image $\mathcal{I}$, the data cost will be separately calculated for the following two cases:

C1: $\mathbf{x} \in O_k$, $D_l^1(\mathbf{x}) = P_1(O_k) \times T$ and $D_l^2(\mathbf{x}) = P_2(O_k) \times T$, where T is the penalty coefficient ($T = 100$ was used in this paper).

C2: $\mathbf{x} \notin \mathcal{O}$, $D_l^1(\mathbf{x}) = 0$ if $\mathbf{x} \in \mathbf{I_1}$ otherwise $D_l^1(\mathbf{x}) = \infty$ if $\mathbf{x} \notin \mathbf{I_1}$ and similar for $D_l^2(\mathbf{x})$.

where $\mathcal{O} = \{O_k\}_{k=1}^K$ represents a set of dynamic objects, and $P_1(O_k)$ and $P_2(O_k)$ are the probabilities of the dynamic object O_k belonging to $\mathbf{I}_1$ and $\mathbf{I}_2$, respectively. Detailed description will be introduced in Section 4.

The smooth energy term $E_{smooth}(\mathcal{I})$ represents all energy costs of adjacent pixel pairs, which is defined as:

$$E_{smooth}(\mathcal{I}) = \sum\nolimits_{(\mathbf{x},\mathbf{y})\in\mathcal{N}(\mathcal{I})} \sigma(\mathbf{x},\mathbf{y}) \cdot E_{smooth}(\mathbf{x},\mathbf{y}), \tag{4}$$

where $\mathcal{N}(\mathcal{I})$ denotes the set of all adjacent pixel pairs in the 4-neighborhood in $\mathcal{I}$, and the coefficient $\sigma(\mathbf{x},\mathbf{y}) = 0$ if the labels of the pixels $\mathbf{x}$ and $\mathbf{y}$ are the same, otherwise

Algorithm 1 Optimal Seamline Detection in Dynamic Scenes

Input: The aligned image pair $\mathbf{I}_1$ and $\mathbf{I}_2$.
Output: The last composite image $\mathcal{I}$.

1. ***Energy cost definition***
   (a) Convert color images into grayscale images.
   (b) Calculate gradient magnitudes via the Sobel operator.
   (c) For each pixel $\mathbf{x}$:
      i. Calculate the intensity difference $C_c(\mathbf{x})$ between $\mathbf{I}_1$ and $\mathbf{I}_2$.
      ii. Calculate the gradient difference $C_g(\mathbf{x})$ between $\mathbf{I}_1$ and $\mathbf{I}_2$.
      iii. Extract the HOG descriptors $\mathcal{H}_1$ and $\mathcal{H}_2$ in $\mathbf{I}_1$ and $\mathbf{I}_2$, and calculate the texture complexity $\Gamma_1(\mathbf{x})$ and $\Gamma_2(\mathbf{x})$ based on $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively. The texture complexity energy $C_t(\mathbf{x})$ is the sum of $\Gamma_1(\mathbf{x})$ and $\Gamma_2(\mathbf{x})$.
      iv. Calculate the last energy cost $C(\mathbf{x})$ by combining the above three energy terms.
2. ***Dynamic object detection***
   (a) Convert the RGB color space into the CIELAB color space.
   (b) For each pixel $\mathbf{x}$:
      i. Calculate the color distance $D_c(\mathbf{x})$ in the CIELAB color space.
      ii. Compute the HOG distance $D_h(\mathbf{x})$ between two extracted HOG descriptors $\mathcal{H}_1$ and $\mathcal{H}_2$.
      iii. Calculate the last distance $D(\mathbf{x})$ by combining $D_c(\mathbf{x})$ and $D_h(\mathbf{x})$, and generate the distance map M_d.
   (c) Find all dynamic regions $\mathcal{O} = \{O_k\}_{k=1}^{K}$ from M_p.
   (d) For each dynamic region O_k:
      i. Segment the corresponding rectangular image regions in $\mathbf{I}_1$ and $\mathbf{I}_2$ via the Mean Shift algorithm.
      ii. Check which image O_k comes from via contour matching.
      iii. Refine the dynamic region and apply the contour matching again.
3. ***Seamline detection via graph cuts***
   (a) Integrate all of those into the graph cuts energy minimization framework to detect the optimal seamline, and obtain the last composite image.

$\sigma(\mathbf{x}, \mathbf{y}) = 1$. $E_{smooth}(\mathbf{x}, \mathbf{y})$ represents the smooth energy between two adjacent pixels $\mathbf{x}$ and $\mathbf{y}$ in $\mathcal{I}$, which is defined as $E_{smooth}(\mathbf{x}, \mathbf{y}) = C(\mathbf{x}) + C(\mathbf{y})$, where $C(\mathbf{x})$ and $C(\mathbf{y})$ are the energy costs of the pixels $\mathbf{x}$ and $\mathbf{y}$ will be defined in Section 3.

3. Energy Cost Definition

The energy cost $C(\mathbf{x})$ of the pixel point $\mathbf{x} = (x, y)^\top$ in $\mathcal{I}$ is comprised of three terms: the intensity difference term $C_c(\mathbf{x})$, the gradient difference term $C_g(\mathbf{x})$, and the texture

complexity term $C_t(\mathbf{x})$, which is defined as:

$$C(\mathbf{x}) = (C_c(\mathbf{x}) + C_g(\mathbf{x})) \times C_t(\mathbf{x}). \tag{5}$$

3.1. Intensity and Gradient Difference

The intensity difference for the pixel $\mathbf{x}$ between two images is computed in the grayscale space, which is defined as:

$$C_c(\mathbf{x}) = |Gray_1(\mathbf{x}) - Gray_2(\mathbf{x})|. \tag{6}$$

where $Gray_1(\mathbf{x})$ and $Gray_2(\mathbf{x})$ denote the intensity values of $\mathbf{x}$ in $\mathbf{I}_1$ and $\mathbf{I}_2$, respectively.

The gradient magnitudes of each pixel in the horizontal and vertical directions are calculated via the Sobel operator in the grayscale space. The gradient difference cost term $C_g(\mathbf{x})$ of the pixel point $\mathbf{x}$ is defined as:

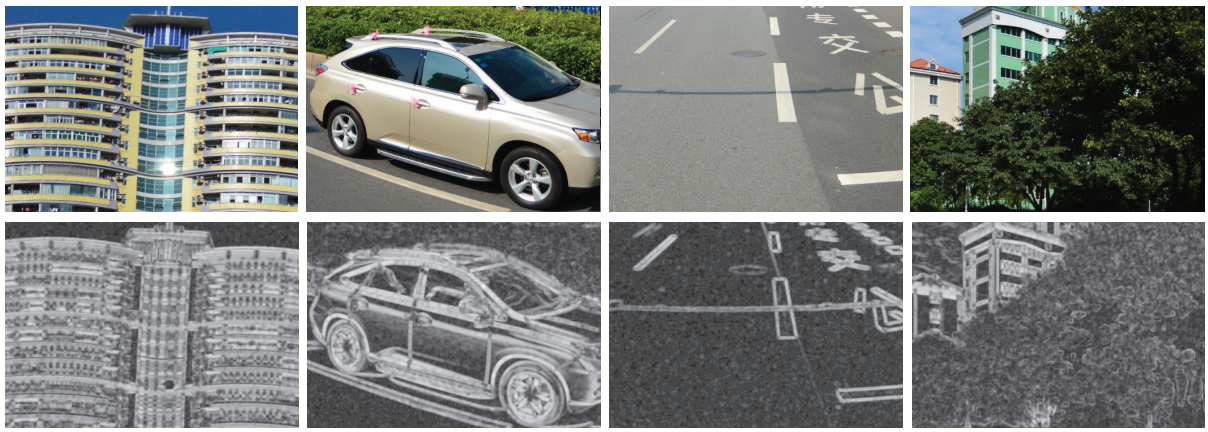
$$C_g(\mathbf{x}) = |G_1^x(\mathbf{x}) - G_2^x(\mathbf{x})| + |G_1^y(\mathbf{x}) - G_2^y(\mathbf{x})|, \tag{7}$$

where $G_1^x(\mathbf{x})$ and $G_1^y(\mathbf{x})$ denote the horizontal and vertical gradient magnitudes of $\mathbf{x}$ in $\mathbf{I}_1$, respectively, and there are the same meanings for $G_2^x(\mathbf{x})$ and $G_2^y(\mathbf{x})$.

3.2. Texture Complexity

Based on two energy costs in term of intensity and gradient differences defined above, in theory, the optimal seamlines found via graph cuts can locate in the regions with high similarity and avoid passing through obvious foreground objects. As we known, some specific regions such as coarse floors and ceilings, and richly-textured trees and flowers are more suitable to be located in the seamlines due to that the image difference in these regions is not easy to be observed. However, we find that sometimes the seamlines detected based on above two differences failed to cross those regions. The major reason of this problem is that those regions maybe also have large intensity and gradient differences. But, we observe that texture complexity of those regions is poor. Thus, to solve this problem, we propose a new texture complexity measurement to distinguish those regions, which is inspired by HOG (Histogram of Oriented Gradient) feature descriptors. HOG has been widely and successfully used in computer vision and image processing for the purpose of object detection, such as human detection (Zhu et al., 2006).

(a) Indoor Scenes



(b) Outdoor Scenes

Figure 3: Some typical regions (Top) existed in indoor/outdoor scenes and their corresponding normalized texture complexity maps (Bottom) where the lighter regions indicate higher texture complexities.

The texture complexity $\Gamma(\mathbf{p})$ of the pixel $\mathbf{p}$ in overlap regions is calculated as follows. All the gradient orientations are computed and converted into the range of $[0, 2\pi]$. Then, for each pixel point $\mathbf{x}$ in an image $\mathbf{I}$, we create a $k \times k$ ($k = 11$ was used in this paper) size window region $\mathcal{N}_{k\times k}(\mathbf{x})$ centered at this pixel. Then we compute the histogram of oriented gradient $\mathcal{H}(\mathbf{x})$ comprised of B ($B = 12$ was used in this paper) bins over this window region. Based on the histogram of oriented gradients, the texture complexity at $\mathbf{x}$ is defined as:

$$\Gamma(\mathbf{x}) = 1 - \frac{\sum_{b=1}^{B} \min(H_b(\mathbf{x}), \bar{H}(\mathbf{x}))}{\sum_{b=1}^{B} H_b(\mathbf{x})}, \tag{8}$$

where $H_b(\mathbf{x})$ denotes the frequency of the b-th bin in $\mathcal{H}(\mathbf{x})$ and $\bar{H}(\mathbf{x})$ represents the mean of frequencies of all bins, i.e., $\bar{H}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} H_b(\mathbf{x})$. Obviously, if $\mathbf{x}$ is located in the

local region with poor texture like walls or strongly repetitive patterns like coarse floors or trees, the frequencies of different bins in the histogram is approximately equal, so $\Gamma(\mathbf{x})$ is close to 0. In contrast, $\Gamma(\mathbf{x})$ is close to 1 if the frequencies of few bins are high and the remaining are low.

Based on the definition of the proposed texture complexity, the texture complexity term $C_t(\mathbf{x})$ for the pixel point $\mathbf{x}$ is defined as:

$$C_t(\mathbf{x}) = \Gamma_1(\mathbf{x}) + \Gamma_2(\mathbf{x}), \tag{9}$$

where $\Gamma_1(\mathbf{x})$ and $\Gamma_2(\mathbf{x})$ represent the texture complexity of $\mathbf{x}$ in $\mathbf{I}_1$ and $\mathbf{I}_2$, respectively. In this way, we can apply $C_t(\mathbf{x})$ to constrain the intensity and gradient differences in the regions with poor texture or strongly repetitive patterns without affecting other richly-textured regions. Figure 3 shows the normalized texture complexity maps of several typical regions existed in indoor and outdoor scenes.

4. Dynamic Object Detection

To compensate the image regions covered by dynamic objects in one image with background regions in another image, we should first detect the dynamic objects based on the image differences in overlap regions and then determine which image they come from in probability. The image difference $D(\mathbf{x})$ of the pixel point $\mathbf{x}$ from two images $\mathbf{I}_1$ and $\mathbf{I}_2$ is computed by combining the color distance $D_c(\mathbf{x})$ and the HOG distance $D_h(\mathbf{x})$. The color distance is computed in the CIELAB color space. The CIELAB color space also has been used to generate superpixels (Achanta et al., 2012). This color space includes all perceivable colors, which means that its gamut exceeds those of the RGB and CMYK color models. In addition, this color space is widely considered as perceptually uniform for small color distances. The color distance $D_c(\mathbf{x})$ is defined as:

$$D_c(\mathbf{x}) = (L_1(\mathbf{x}) - L_2(\mathbf{x}))^2 + (A_1(\mathbf{x}) - A_2(\mathbf{x}))^2 + (B_1(\mathbf{x}) - B_2(\mathbf{x}))^2, \tag{10}$$

where $L_1(\mathbf{x})$, $A_1(\mathbf{x})$ and $B_1(\mathbf{x})$ denote the intensity values of L, A and B channels of $\mathbf{x}$ in $\mathbf{I}_1$, respectively, and there are the same meanings for $L_2(\mathbf{x})$, $A_2(\mathbf{x})$ and $B_2(\mathbf{x})$. However, it is very difficult for clearly detecting the dynamic objects from the image distance

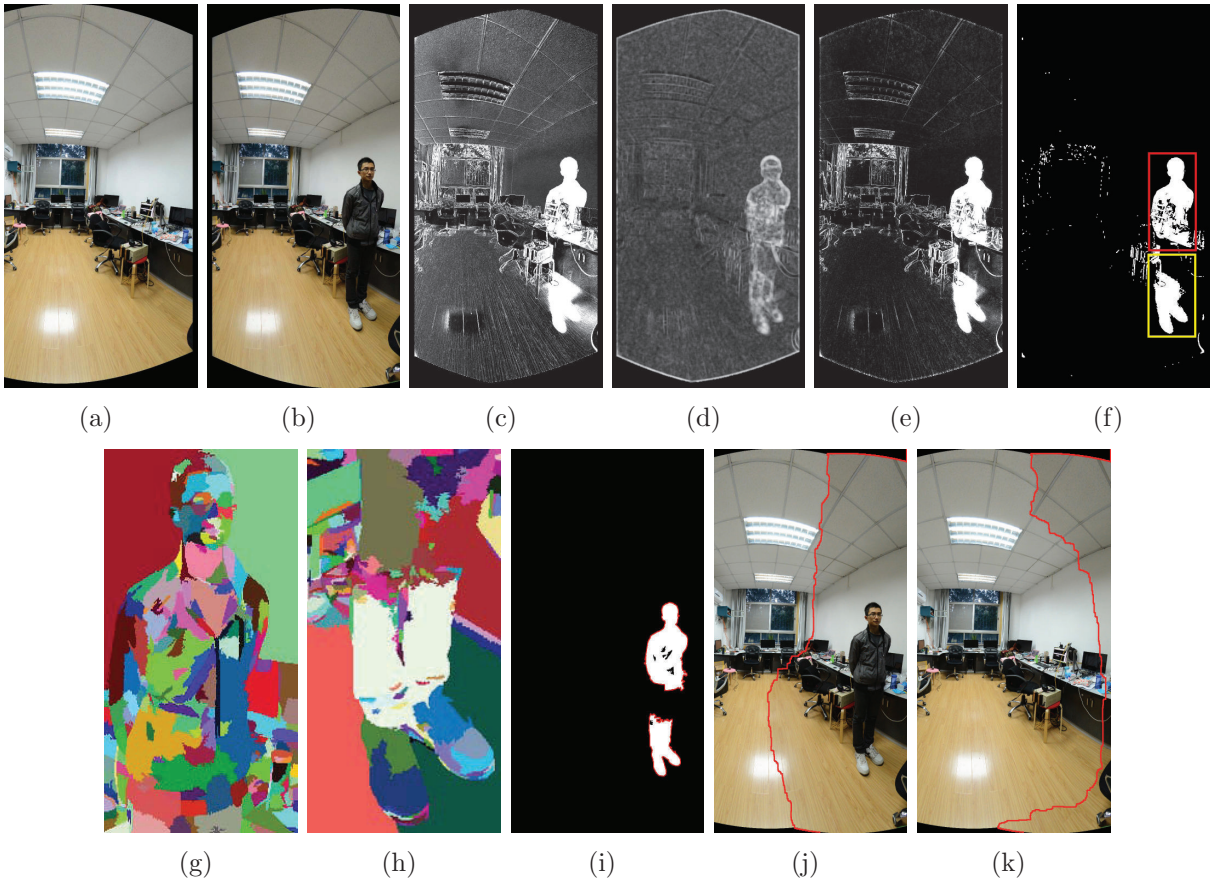


Figure 4: (a)-(b) two geometrically aligned input source images with a dynamic object (a pedestrian); (c) the color distance map in CIELAB; (d) the HOG distance map; (e) the final distance map; (f) the dynamic objects binary map after mathematical morphology; (g)-(h) the corresponding rectangle segmentation results in the second image of the Mean Shift algorithm for the regions masked by red and yellow rectangles in (f); (i) the dynamic objects binary map refined by the Mean Shift algorithm; (j)-(k) the finally mosaicked images with optimal seamlines without/with dynamic object compensation.

map only based on the above defined color distances, as shown in Figure 4(c). Because there are many noise and error regions caused by the geometric misalignments and the photometric inconsistencies. To overcome this shortcoming, we further modify the point distance based on the corresponding HOG distance. The HOG of each pixel in $\mathbf{I}_1$ or $\mathbf{I}_2$ has been calculated for computing the texture complexity defined in Section 3.2. Let $\mathcal{H}_1 = \{H_1^b\}_{b=1}^B$ and $\mathcal{H}_2 = \{H_2^b\}_{b=1}^B$ be HOGs of the pixel point $\mathbf{x}$ in $\mathbf{I}_1$ or $\mathbf{I}_2$, respectively. The distance between $\mathcal{H}_1$ and $\mathcal{H}_2$ is computed based on the Bhattacharyya distance as follow:

$$D_h(\mathbf{x}) = \sqrt{1 - \frac{1}{\sqrt{\bar{H}_1(\mathbf{x})\bar{H}_2(\mathbf{x})B^2}} \sum_{b=1}^{B} \sqrt{H_1^b(\mathbf{x})H_2^b(\mathbf{x})}}, \tag{11}$$

where $\bar{H}_1(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} H_1^b(\mathbf{x})$ and $\bar{H}_2(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} H_2^b(\mathbf{x})$. In this way, we obtain a HOG distance map as shown in Figure 4(d). Combining $D_c(\mathbf{x})$ and $D_h(\mathbf{x})$, the final distance $D(\mathbf{x})$ is defined as:

$$D(\mathbf{x}) = -\frac{1}{\ln(D_h(\mathbf{x}))} \times D_c(\mathbf{x}). \tag{12}$$

Based on the final distance map, as shown in Figure 4(e), we can simply get the dynamic objects map (the binary image) with a distance threshold. We further filter out small noise regions via mathematical morphology, as shown in Figure 4(f). Let $\mathcal{O} = \{O_k\}_{k=1}^{K}$ be all separated dynamic objects where K is the number of dynamic objects. For each dynamic object O_k, to decide which image it comes from, we first segment the corresponding rectangular image regions of O_k in $\mathbf{I}_1$ and $\mathbf{I}_2$ by the Mean Shift algorithm (Comaniciu and Meer, 2002). We then find the contour of O_k, as shown in 4(i). After that, we match the contours of dynamic objects with the boundaries of segments generated in $\mathbf{I}_1$ and $\mathbf{I}_2$, respectively. Furthermore we refine each dynamic object O_k based on the segments with a higher contour matching value. If more than half pixels of one segment belong to the foreground, we label this whole segment as a part of dynamic object, otherwise, we regard it as background. The contour matching values $M_1(O_k)$ and $M_2(O_k)$ of the refined dynamic object O_k with respective to $\mathbf{I}_1$ and $\mathbf{I}_2$ are re-computed at first. Then we compute the probabilities $P_1(O_k)$ and $P_2(O_k)$ of O_k belonging to $\mathbf{I}_1$ and $\mathbf{I}_2$ as $P_1(O_k) = \frac{M_1(O_k)}{M_1(O_k)+M_2(O_k)}$, $P_2(O_k) = \frac{M_2(O_k)}{M_1(O_k)+M_2(O_k)}$, and $P_1(O_k) + P_2(O_k) = 1$. For example, in Figure 4(f), there are two dynamic regions, masked by red and yellow rectangles, respectively. We found that they all come from the second image, because the contour matching values are higher than the first image. So, we further refine them by the segments generated in the second image, as shown in Figure 4(g) and Figure 4(h). In this way, we obtain the refined dynamic objects map as shown in Figure 4(i).

5. Experimental Results

We tested our proposed method on the images captured from several dynamic scenes with moving objects. The images were captured by rotating a Nikon D7100 camera of 24 million pixels with a wide-angle lens on a tripod platform at different times. The faces in some pictures presented in this paper have been blurred due to privacy protection. These

images were first aligned and warped into a common coordinate system via the popularly used commercial software *PTGui* [1]. Of course, there always exist geometric misalignments between these images in different extents. We utilized those geometrically aligned images to test our proposed optimal seamline detection algorithm despite the presence of dynamic objects and geometric misalignments. Our algorithm was implemented with C++ under Windows and tested in a computer with an Intel Core i7-4770 at 3.4GHz.

To illustrate whether our energy criteria defined in Section 3 is effective, we compared the seamline detection results by different combination of intensity (A), gradient (G) and texture complexity (T), as shown in Figure 5. Due to that the effectiveness of texture complexity is similar with the balance coefficient, it can not be used alone to detect the seamlines. Thus, there are only 6 kinds of combination of features. We found that the seamlines detected based on A, G and $A+G$ without considering texture complexity (T) can't avoid crossing the car. However, when the texture complexity (T) is considered, $G+T$ and $A+G+T$ can generate high quality seamlines which avoid crossing the obvious objects and cross the regions with high similarity, only $A+T$ failed mainly due to that the intensity of the car is similar to the road. We also found that the seamlines produced by $G+T$ and $A+G+T$ are similar, but according to the previous work completed by Mills and Dudek (2009) and Zeng et al. (2014), the combination of intensity and gradient information is more robust. In the aspect of computational cost, the times of Figure 3(a)-(f) consisting of all the elapsed times in the energy cost computation and the graph cuts optimization are similar. When the texture complexity is considered, the times of $G+T$ and $A+G+T$ used are 7.3190s and 7.3710s, respectively. All of those times are less than the times of G and $A+G$ used, which are 8.3790s and 8.3280s. And the times of $A+T$ and A used are 6.5910s and 6.4840s, respectively, which are almost same. Although the energy computation time has increased due to the use of texture complexity, the computational time of seamline detection has reduced. This is mainly due to that the optimal solution maybe can be more easily found via graph cuts when the texture complexity is used.

[1] http://www.ptgui.com/

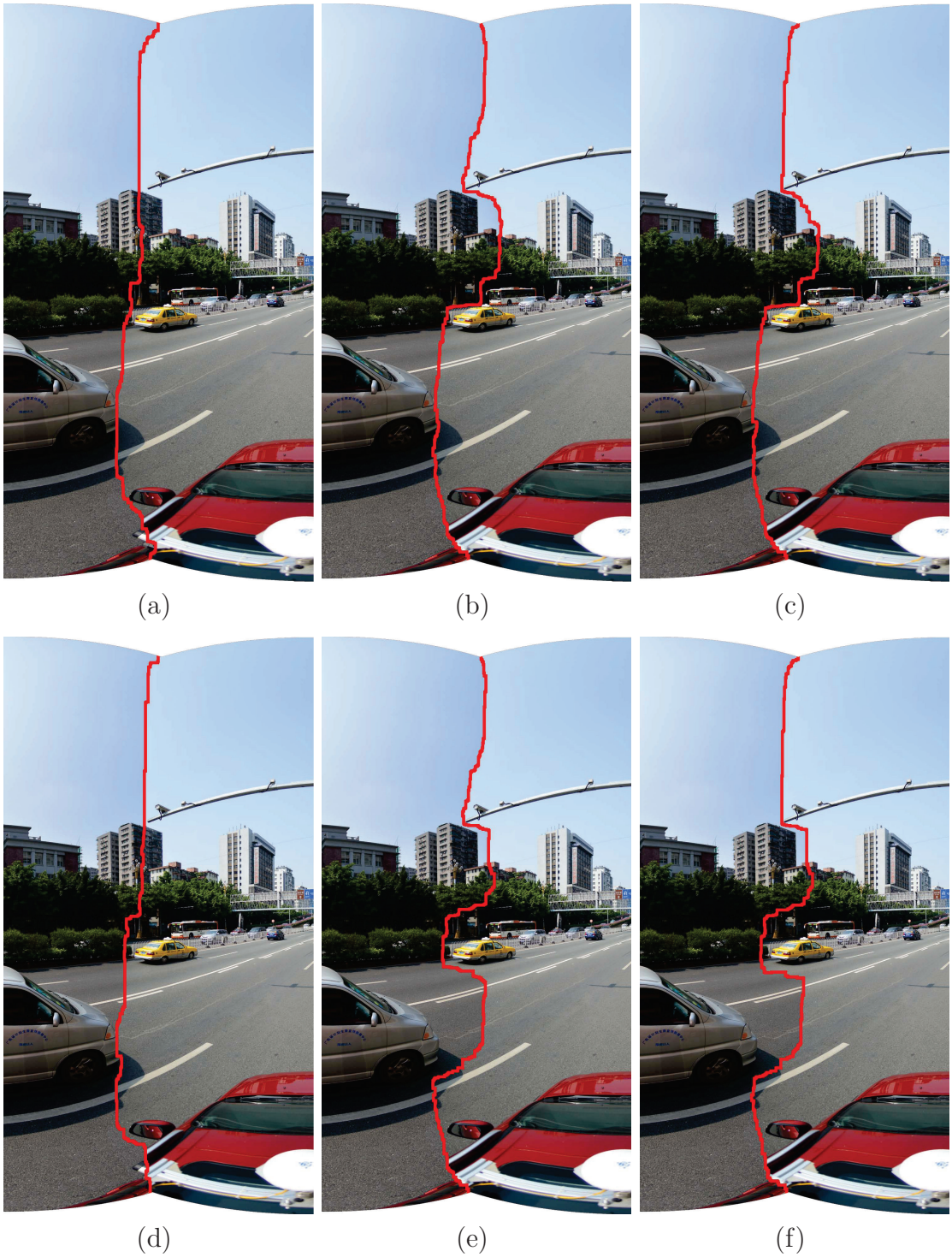
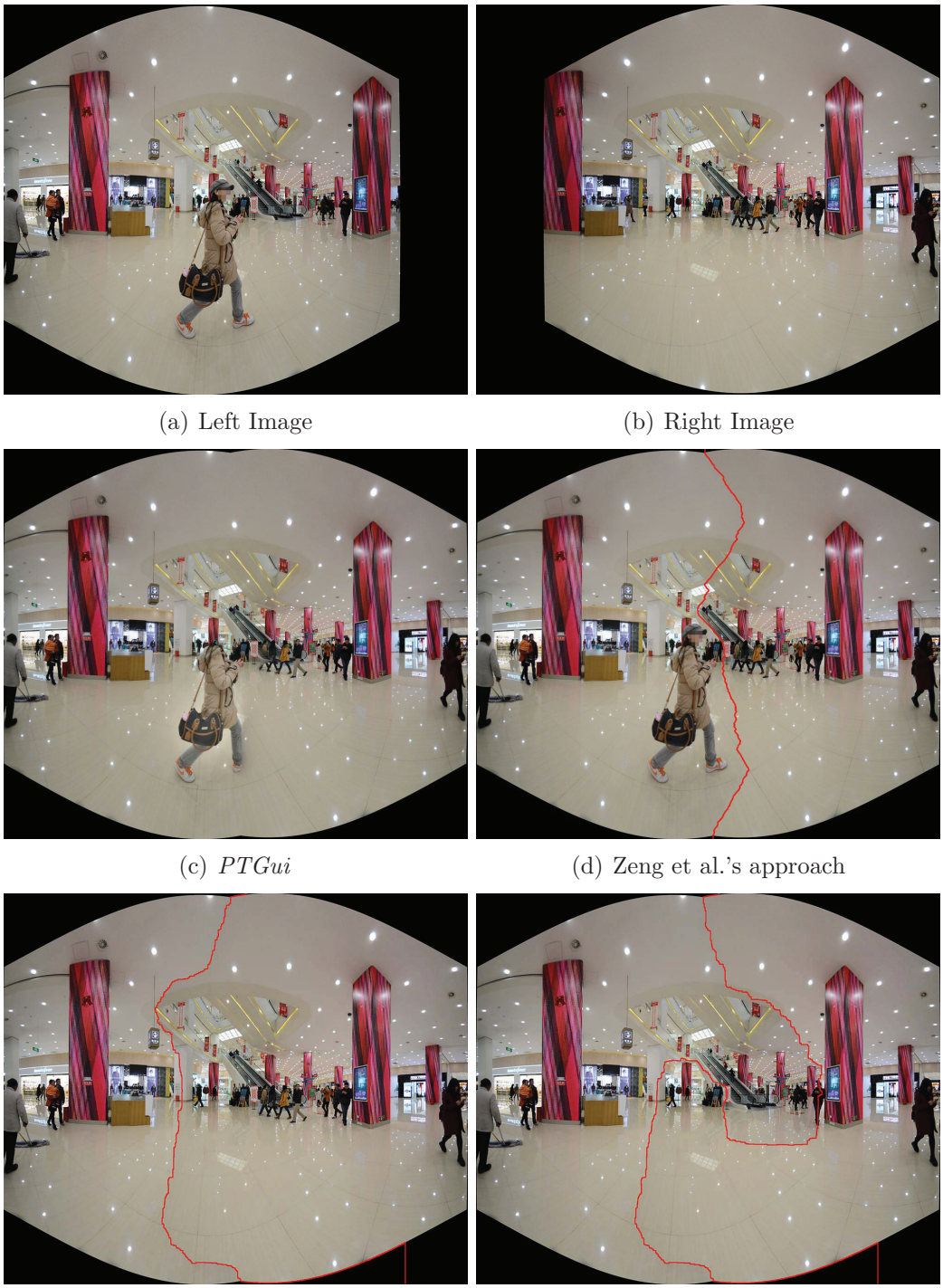


Figure 5: The seamline detection results with the different combination of intensity (A), gradient(G) and texture complexity (T): (a) A; (b) G; (c) $A+G$; (d) $A+T$, (e) $G+T$; (f) $A+G+T$. And the computational times of six combinations in (a)-(f) are 6.4840s, 8.3790s, 8.3280s, 6.5910s, 7.3190s and 7.3710s, respectively.



(a) Left Image (b) Right Image

(c) *PTGui* (d) Zeng et al.'s approach

(e) Without Compensation (f) With Compensation

Figure 6: An example pair of input images captured in a mall shopping center: (a)-(b) Two geometrically aligned input images; (c)-(d) The image mosaics generated by *PTGui* and the Zeng et al.'s approach, respectively; (e)-(f) The image mosaics generated by our algorithm without dynamic objects compensation and with it, respectively.

In Figure 6, we used two geometrically aligned images captured in a mall shopping center with multiple dynamic objects to conduct our experiment. In this experiment, we compared our proposed algorithm with the popularly used commercial software *PTGui* and the Zeng et al.'s approach (Zeng et al., 2014). From Figures 6(a)(b), we found that there are many dynamic objects in the overlap regions, including a girl wearing a hat and many walking peoples nearing the escalator. The problem of ghosts caused by those moving objects arose in the composition image generated by *PTGui*, as shown in Figure 6(c). This problem can be solved well via the optimal seamline detection algorithm, as shown in Figures 6(d)(e). However, the moving objects always appeared in the final image mosaics. But, it disappeared when the proposed dynamic objects compensation was applied, as shown in Figure 4(f). From Figures 6(e)(f), we also observed that the seamlines were always located in the regions with the high images similarity and the low object dislocation and simultaneously avoided crossing dynamic objects mainly due to the constraint of our proposed energy criterion defined in Section 3. Another example is shown in Figure 7 where the mainly dynamic object in this indoor scene is a walking man. And the similar results can be found as Figure 6. From the visual comparison, we proved the superiority of our proposed algorithm which can generated image mosaics with relatively clean backgrounds and free from artifacts.

Under the case that an object just moves locally (i.e., small part of this object moves) or in the scenes with the texture of dynamic objects very similar to the background, these dynamic objects cannot be fully and accurately detected. If we applied the hard constraint with the data energy defined as: $D_l^1(\mathbf{x}) = \infty$ and $D_l^2(\mathbf{x}) = 0$ if $P_1(\mathbf{x}) > P_2(\mathbf{x})$ otherwise $D_l^1(\mathbf{x}) = 0$ and $D_l^2(\mathbf{x}) = \infty$, the detected seamline may cross the dynamic objects as shown in Figure 8 from which we observed that only head locally moved in this scene and was identified as a dynamic object region. However, with our proposed soft constraint defined in the `C1` case of Eq. (3) , the detected seamline successfully avoided crossing this dynamic object.

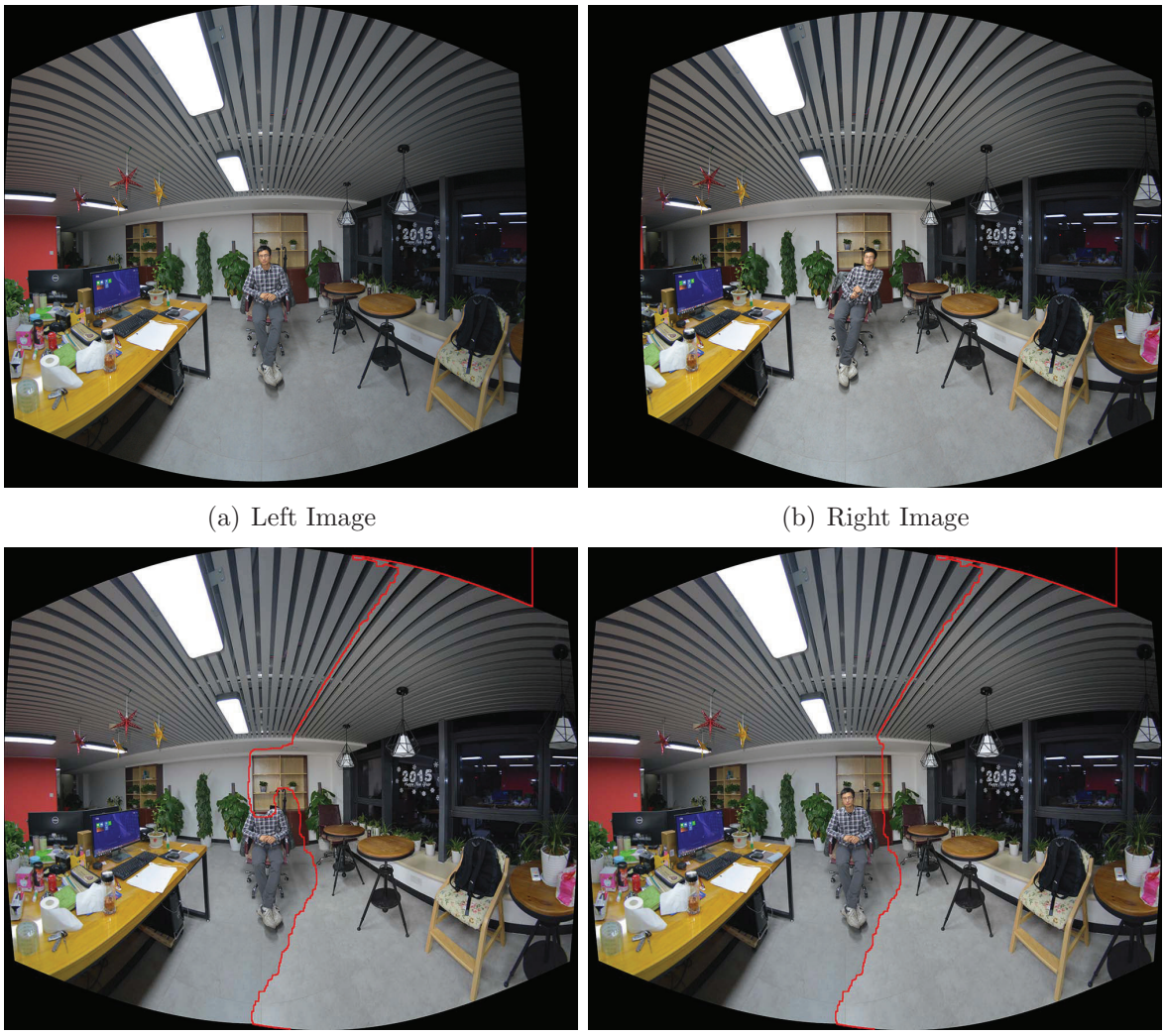
In Figure 9, we presented a representative failure example and discussed the key cause of this failure. From Figures 9(a)(b), we found that there are two moving objects in this

(a) Left Image (b) Right Image

(c) *PTGui* (d) Zeng et al.'s approach

(e) Without Compensation (f) With Compensation

Figure 7: An example pair of input images captured in a small office: (a)-(b) Two geometrically aligned input images; (c)-(d) The image mosaics generated by *PTGui* and the Zeng et al.'s approach, respectively; (e)-(f) The image mosaics generated by our algorithm without dynamic objects compensation and with it, respectively.



(a) Left Image (b) Right Image

(c) Mosaicked Image with Hard Constraint (d) Mosaicked Image with Soft Constraint

Figure 8: Two geometrically aligned images in (a) and (b) captured from a dynamic scene with a local motion of a human head are mosaicked based on the seamlines detected by the hard constraint in (c) and the soft one in (d).

scene. We flagged them as **A** and **B**. In Figure 9(c), we used the geometric center line to replace the seamline, and we found that one moving object has been crossed. This problem can be solved well by detecting the optimal seamline, as shown in Figures 9(d)-(f). However, we observed that our algorithm failed to compensate the regions of two moving objects, as shown in Figure 9(f). For the dynamic object **A**, the reason is because this man moving slowly and locally, and the background regions cannot be compensated completely. For the dynamic object **B**, our algorithm should compensate those dynamic regions in theory. However, due to that the color of background is similar with the

dynamic objects, the dynamic regions are very difficult to detect, and our algorithm can not detect them. Therefore, our algorithm failed to compensate the dynamic regions covered by $\mathbf{B}$. Although our algorithm failed to compensate the dynamic regions with the information of background, it also can avoid passing through all dynamic objects as the Zeng et al.'s approach does.

In Figure 10, we present the 360°-view panorama of an outdoor scene. In many cases, the moving object is only allowed to appear once at most, otherwise there will be artifacts in the mosaicked images. To mosaic a panoramic image, we first captured five original images with the size of 3200×4800, and then warped them into the same coordinate system with the image size of 9512×4756 by *PTGui*. From the original images, we can found that there is a moving man in this scene. The problem of ghosts appeared in the panorama generated by *PTGui*, at the same time, the man appeared five times in this scene, as shown in Figure 10(b). In Figures 10(e)(f), which were generated by the Zeng et al.'s approach and our algorithm without dynamic objects compensation (i.e., the data energy is always calculated in the second case `C2`), the ghosts disappeared, but the man also appeared four times in the same scene. This problem can be solved well by our proposed dynamic objects compensation strategy. As shown in Figure 10(f), this man only appeared once when dynamic objects compensation was applied. With the dynamic objects compensation, the computation time increases to 57.129s from the original 48.213s without compensation. The increasing time is mainly spent on detecting the dynamic objects and determining which image they come from. We also found that the computation time of Figure 10(c) is only 31.4820s. This mainly due to that this algorithm used the dynamic programming method to find the seamline with the minimal cost, which is more efficient than graph cuts. But, we found that the quality of seamlines detected by this algorithm is lower than our proposed algorithm. Especially, the second seamlines (from left to right) pass through many edges of the building, but the seamlines detected by our proposed algorithm avoid crossing them. And more importantly, the Zeng et al.'s approach can not avoid the moving objects appearing many times in the last image mosaic. The 360°-view panorama of an indoor scene is presented in Figure 11, and the

similar conclusion can be drawn as the outdoor scene shown in Figure 10. In addition, to prove that our algorithm can handle the artifacts caused by geometric alignments and moving objects, we show the image mosaic generated based on the geometric center lines of the overlap regions in Figure 11(b). We also show the image mosaic generated by the open-source software *Enblend* [2] in Figure 11(d), which also applied graph cuts to find the optimal seamlines for image blending. From Figures 11(b)(d) and especially the local regions highlighted by red rectangles, we observed that there are many geometric misalignments between those input aligned images in different extents. But our proposed algorithm can avoid the artifacts caused by those geometric misalignments by detecting the optimal seamlines, as shown in Figures 11(f)(g). We also can found that our algorithm performed better than the Zeng et al.'s approach by comparing the panoramic images presented in Figures 11(e)(g). We observed that the first seamline from the left of Figure 11(e) passes through the desk and the chair, but the seamline in Figure 11(g) bypasses those regions and passes through nearby white wall and floor.

From the experiments conducted above, we observed that our proposed algorithm can generate high quality panoramic images in dynamic scenes. Simultaneously, our algorithm can compensate the regions occupied by dynamic objects based on our detected optimal seamlines. However, we also found that our algorithm is little time-consuming by comparing with the Zeng et al.'s approach. And, our algorithm can be used to produce high-quality panoramic maps for indoor and outdoor scenes.

6. Conclusion

This paper proposed a novel optimal seamline detection algorithm for image mosaicking via graph cuts despite the presence of dynamic objects. One contribution in this paper is that we applied a new cost energy function combining the intensity difference, the gradient difference, and the texture complexity in the graph cuts energy minimization framework. So our algorithm can ensure that the seamlines are optimally located in the regions with high image similarity and low object dislocation. Another contribution is

[2] http://enblend.sourceforge.net/

that we proposed a novel approach to generate the composition images with clean backgrounds. It first detects dynamic objects based on the image difference combining both the color distance in the CIELAB color space and the texture distance in HOG. Then, we determine in probability which image these dynamic objects come from based on segments and contour matching. Finally we integrate all of these into the seamline detection optimization process. Experimental results have proved that our algorithm can produce high-quality image mosaics without artifacts and with relatively clean backgrounds (i.e., with dynamic object as few as possible).

Nevertheless, the proposed algorithm may be improved in the future in the following ways. First, the superpixel segmentation can be introduced to greatly improve the optimization efficiency by decreasing the number of elements in graph cuts. Second, the scene understanding or parsing can be applied into our algorithm. For example, the floors and people can be detected out for guiding the seamlines.

Acknowledgment

This work was partially supported by the National Natural Science Foundation of China (Project No. 41571436), the National Natural Science Foundation of China under Grant 91438203, the Hubei Province Science and Technology Support Program, China (Project No. 2015BAA027), the Jiangsu Province Science and Technology Support Program, China (Project No. BE2014866), and the South Wisdom Valley Innovative Research Team Program.

References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S., 2012. SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (11), 2274–2282.

Bellman, R., 1957. Dynamic Programming. Princeton University Press, Princeton, NJ.

Boutellier, J. J., Bordallo-Lopez, M., Silvén, O., Tico, M., Vehviläinen, M., 2007. Creating panoramas on mobile phones. In: Electronic Imaging.

Boykov, Y., Kolmogorov, V., 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (9), 1124–1137.

Boykov, Y., Veksler, O., Zabih, R., 2001. Fast approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (11), 1222–1239.

Brown, M., Lowe, D. G., 2007. Automatic panoramic image stitching using invariant features. International Journal of Computer Vision 74 (1), 59–73.

Comaniciu, D., Meer, P., 2002. Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (5), 603–619.

Dalal, N., Triggs, B., 2005. Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).

Davis, J., 1998. Mosaics of scenes with moving objects. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).

Dijkstra, E. W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1 (1), 269–271.

Hsieh, J.-W., 2004. Fast stitching algorithm for moving object detection and mosaic construction. Image and Vision Computing 22 (4), 291–306.

Kass, M., Witkin, A., 1988. Snakes: active contour models. International Journal of Computer Vision 1 (4), 321C–331.

Kim, D.-W., Hong, K.-S., 2008. Practical background estimation for mosaic blending with patch-based markov random fields. Pattern Recognition 41 (7), 2145–2155.

Kolmogorov, V., Zabih, R., 2001. Computing visual correspondence with occlusions using graph cuts. In: IEEE International Conference on Computer Vision (ICCV).

Levin, A., Zomet, A., Peleg, S., Weiss, Y., 2004. Seamless image stitching in the gradient domain. In: European Conference on Computer Vision (ECCV).

López, M. B., Hannuksela, J., Silvén, O., Vehviläinen, M., 2014. Interactive multi-frame reconstruction for mobile devices. Multimedia Tools and Applications 69 (1), 31–51.

Mills, A., Dudek, G., 2009. Image stitching with dynamic elements. Image and Vision Computing 27 (10), 1593–1602.

Philip, S., Summa, B., Tierny, J., Bremer, P.-T., Pascucci, V., 2015. Distributed seams for gigapixel panoramas. IEEE Transactions on Visualization and Computer Graphics 21 (3), 350–362.

Rother, C., Kolmogorov, V., Blake, A., 2004. Grabcut: Interactive foreground extraction using iterated graph cuts. In: ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2004).

Tao, C., Sun, H., Yang, C., Tian, J., 2011. Efficient image stitching in the presence of dynamic objects and structure misalignment. Journal of Signal and Information Processing 2 (3), 205.

Tao, M. W., Johnson, M. K., Paris, S., 2013. Error-tolerant image compositing. International Journal of Computer Vision 103 (2), 178–189.

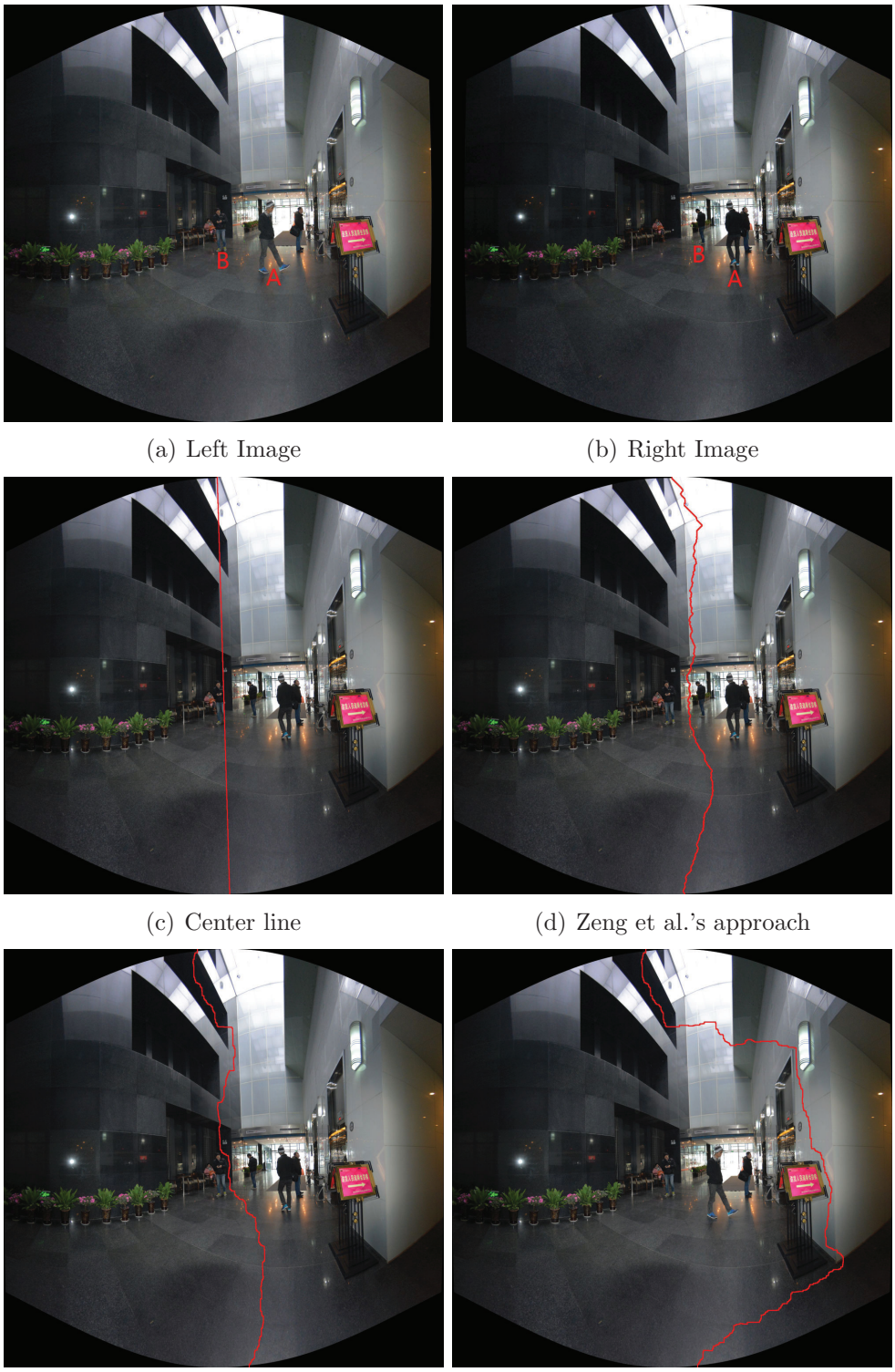
Xiong, Y., Pulli, K., 2009. Mask based image blending and its applications on mobile devices. In: Sixth International Symposium on Multispectral Image Processing and Pattern Recognition.

Zeng, L., Zhang, S., Zhang, J., Zhang, Y., 2014. Dynamic image mosaic via SIFT and dynamic programming. Machine Vision and Applications 25 (5), 1271–1282.

Zhi, Q., Cooperstock, J. R., 2012. Toward dynamic image mosaic generation with robustness to parallax. IEEE Transactions on Image Processing 21 (1), 366–378.

Zhou, Z., Li, S., Wang, B., 2014. Multi-scale weighted gradient-based fusion for multi-focus images. Information Fusion 20 (1), 60–72.

Zhu, Q., Yeh, M.-C., Cheng, K.-T., Avidan, S., 2006. Fast human detection using a cascade of histograms of oriented gradients. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).

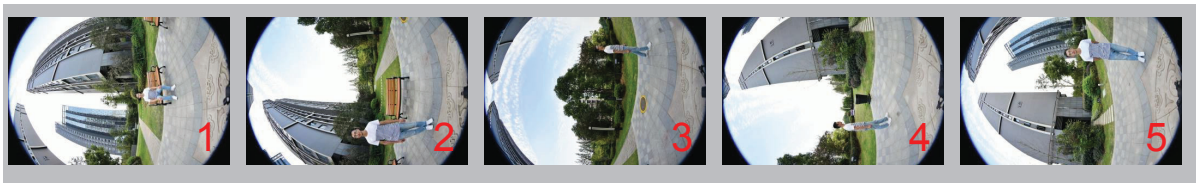


(a) Left Image (b) Right Image

(c) Center line (d) Zeng et al.'s approach

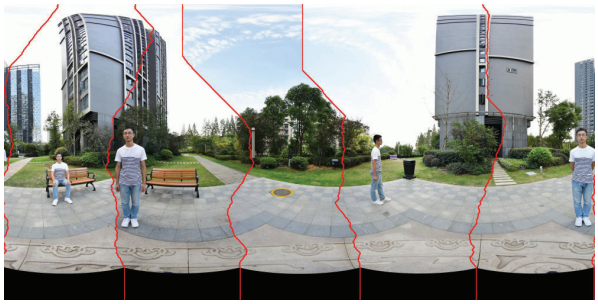
(e) Without compensation (f) With compensation

Figure 9: An example of failure case: (a)-(b) Two geometrically aligned images with two moving objects **A** and **B**; (c) The image mosaicked based on the geometric center line; (d) The composition image generated by the Zeng et al.'s approach; (e)-(f) The image mosaics generated by our algorithm without dynamic objects compensation and with it, respectively.

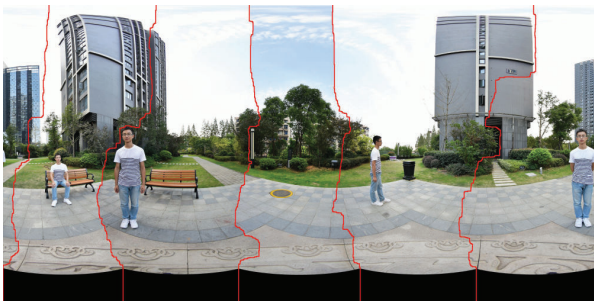


(a) Original Images (3200×4800)

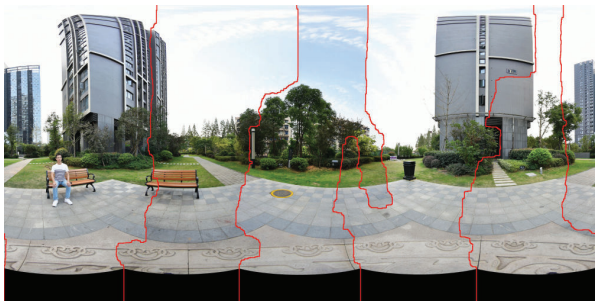
(b) *PTGui*



(c) Zeng et al.'s approach

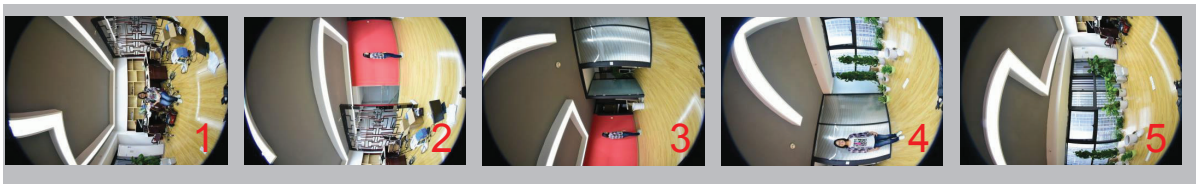


(d) Without Compensation



(e) With Compensation

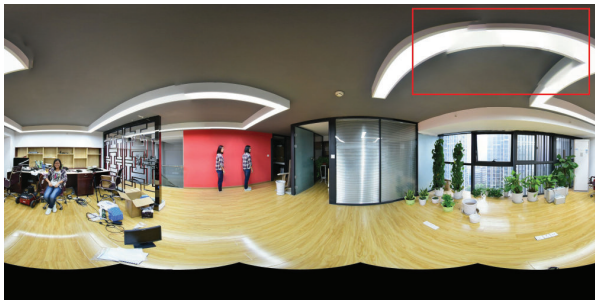
Figure 10: The 360°-view panorama of an outdoor dynamic scene with a moving man: (a) 5 original images; (b) the image mosaicked by *PTGui*; (c) the image mosaicked by the Zeng et al.'s approach; (d) the image mosaicked by our proposed method without dynamic objects compensation; (e) the image mosaicked by our proposed method with dynamic objects compensation. The computation times of (c), (d) and (e) are 31.4820, 48.213s and 57.129s, respectively.



(a) Original Images (3200×4800)

(b) Center lines

(c) *PTGui*



(d) *Enblend*

(e) Zeng et al.'s approach

(f) Without Compensation

(g) With Compensation

Figure 11: The 360°-view panorama of an indoor dynamic scene with a moving woman: (a) 5 original images; (b) the image mosaicked based on the geometric center lines; (c) the image mosaicked by *PTGui*; (d) the image mosaicked by *Enblend*; (e) the image mosaicked by the Zeng et al.'s approach; (f) the image mosaicked by our proposed method without dynamic objects compensation; (g) the image mosaicked by our proposed method with dynamic objects compensation. The computation times of (e), (f) and (g) are 31.5330s, 41.910s and 55.385s, respectively.