Machine Vision and Applications (2017) 28:819–837
DOI 10.1007/s00138-017-0874-y

ORIGINAL PAPER

Optimal seamline detection in dynamic scenes via graph cuts for image mosaicking

Li Li[1] · Jian Yao[1] · Haoang Li[1] · Menghan Xia[1] · Wei Zhang[2]

Received: 3 October 2015 / Revised: 13 June 2017 / Accepted: 10 August 2017 / Published online: 19 September 2017

Abstract In this paper, we present a novel method for creating a seamless mosaic from a set of geometrically aligned 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 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 segments, 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.

✉ Jian Yao
jian.yao@whu.edu.cn
http://cvrs.whu.edu.cn/

[1] Computer Vision and Remote Sensing (CVRS) Lab, School of Remote Sensing and Information Engineering, Wuhan University, Wuhan, Hubei, People's Republic of China

[2] School of Control Science and Engineering, Shandong University, Jinan, Shandong, People's Republic of China

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 [18] and satellite imagery mosaicking [27].

In an ideal static scene, 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 differences in exposure, 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 through color correction and smoothing transition [6,16,25,31], 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 there are large depth differences in the scene with respective to the camera, or captured from dynamic scene 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 seamline, but it cannot handle the obvious parallax caused by geometric misalignments or dynamic objects. One effective way to solve this problem is to detect

the optimal seamline between two images and ensure that the optimal seamline avoids crossing obvious dynamic objects and the regions with low image similarity and large object dislocation. If the 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, 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 regard the seamline detection as an energy optimization problem. They solved it by minimizing the specially designed energy functions which are defined to represent the differences along the seamline. For those 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 [13], Dijkstra's algorithm [10], dynamic programming [2], and graph cuts [4,5]. 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, and they will possibly fail in dynamic scenes. If the scene contains dynamic objects, ghosting effect would possibly appear in the last mosaicked images. Some algorithms [3,9,12,14,19,20,24,29,30] have been designed for dynamic scenes to remove ghosting effect. Boutellier et al. [3] 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 ghosts caused by moving objects, it detected the moving objects by simply thresholding the difference map between two images. López et al. [19] also applied this strategy to avoid the moving objects when mosaicking the images captured by mobile devices. Davis [9] 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 [10] 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 fail to find the optimal seamline using intensity difference alone. To solve this problem, Mills and Dudek [20] 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 to the background, the optimal seamline found by this algorithm may cross the dynamic objects and the last composite image would contain ghosting effect. To improve the algorithm proposed by Mills and Dudek [20], Zeng et al. [29] proposed a 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 [2], which is more efficient than Dijkstra's algorithm, to find the optimal seamline. These existing methods can avoid crossing dynamic objects and produce high-quality image mosaics for most scenes. However, these methods cannot remove dynamic objects from the finally mosaicked images and obtain the last background panoramic images, as shown in Fig. 1d–f, especially in (d), the moving object appears twice in the last panorama. To solve this problem, Zhi and Cooperstock [30] 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 images are captured without the moving objects in the scene. In addition, Kim and Hong [14] proposed a background estimation algorithm and applied it to stitch the images captured from 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 region to estimate the background, but we only use two images for each overlap region.

In this paper, 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, as shown in Fig. 1g. The motivations and applications of our proposed method are summarized as follows:

- First, our proposed method can construct the panoramic background of an image sequence and eliminate the moving objects from the dynamic scene. The panoramic images generated by our method have relatively clean backgrounds. In many cases, we only want to obtain the background of a scene, and the moving objects will occlude some important signs or information (e.g.,

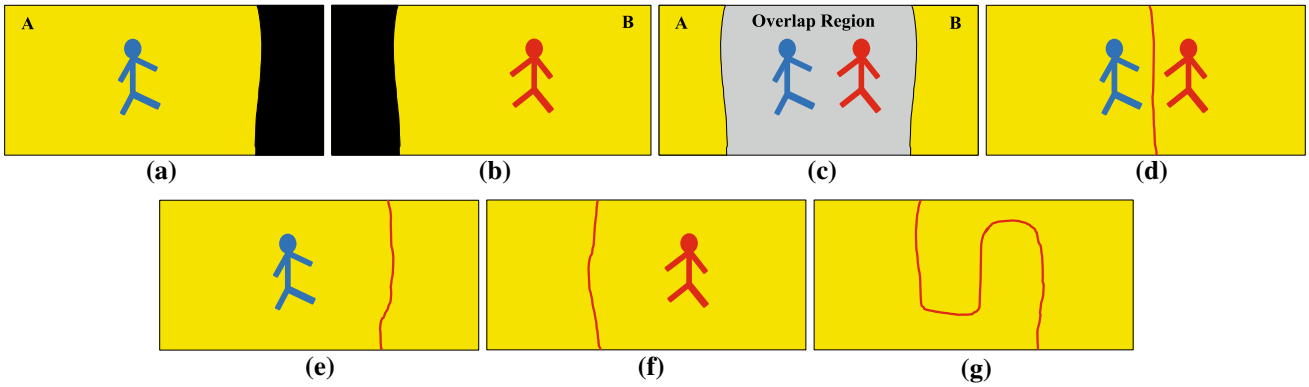


Fig. 1 Two geometrically aligned input images **A** and **B** in **a**, **b** with a moving object in the overlap region as shown in **c**. The image mosaics with different seamlines are presented in **d**–**g**, which avoid crossing the moving object, like the works completed in [9,20,29]. The moving 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

instruction of safe passage, poster, and beautiful scenery). For example, we want to take a panoramic photograph for a beautiful scenery, but there are always several people walking in this scene; our method can solve this problem very well and generate the last panoramic photograph free from moving objects.

- Second, our proposed method can avoid the same dynamic object appears twice or multiple times in the same scene. In many cases, particularly humans in an indoor scene, the moving object is only allowed to appear once at most; otherwise, there will be artifacts in the image mosaic.
- Last, the panoramic images generated by our method are free from the artifacts caused by geometric misalignments and dynamic objects. The ultimate standard to judge whether a mosaic is good or not, is if it is free from artifacts. Our proposed method can eliminate the artifacts caused by geometric misalignments and dynamic objects, so can be used to generate high-quality panoramic images for dynamic scenes.

We formulate the optimal seamline detection as an energy optimization problem and solve it in the graph cuts energy minimization framework [17]. In our proposed method, not only the intensity and gradient differences are considered into the cost of each pixel in the overlap regions between images, but also the texture complexity inspired by histogram of oriented gradient (HOG) [8] is integrated into the cost function. In addition, to create an image mosaic with a relatively clean background, 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 algorithm [7], and finally determined from which image they come in probability based on contour matching. All of these are integrated into the graph cuts framework for detecting the optimal seamline concealing the parallax and compensating the image regions covered by dynamic objects for image mosaicking, as shown in Fig. 1g.

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

2 Seamline detection via graph cuts

In the overlap region 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 algorithm [5] 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 [21,24]. For example, Rother et al. [23] provided a powerful semiautomatic algorithm for foreground object extraction based on iterated

Algorithm 1 Optimal Seamline Detection for 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 ones.
   (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 $\mathbf{I}_2$ according to Eq. (6).
      ii. Calculate the gradient difference $C_g(\mathbf{x})$ between $\mathbf{I}_1$ and $\mathbf{I}_2$ according to Eq. (7).
      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 $\mathcal{H}_2$ according to Eq. (8), respectively. The texture complexity energy $C_t(\mathbf{x})$ is the sum of $\Gamma_1(\mathbf{x})$ and $\Gamma_2(\mathbf{x})$, as defined in Eq. (9).
      iv. Calculate the last energy cost $C(\mathbf{x})$ by combining $C_c(\mathbf{x})$, $C_g(\mathbf{x})$ and $C_t(\mathbf{x})$, as defined in Eq. (5).
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 according to Eq. (10).
      ii. Compute the HOG distance $D_h(\mathbf{x})$ between two extracted HOG descriptors $\mathcal{H}_1$ and $\mathcal{H}_2$ according to Eq. (11).
      iii. Calculate the last distance $D(\mathbf{x})$ by combining $D_c(\mathbf{x})$ and $D_h(\mathbf{x})$ according to Eq. (12), and generate the distance map M_d.
   (c) Find all dynamic regions $\mathcal{O} = \{O_k\}_{k=1}^{K}$ from M_d.
   (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.
      iv. Calculate two probabilities $P_1(O_k)$ and $P_2(O_k)$ based on the results of contour matching.
3. ***Seamline detection via graph cuts***
   (a) Define the data energy term $E_{\text{data}}(\mathcal{I})$. For each pixel $\mathbf{x}$:
      i. Check whether $\mathbf{x}$ locates on the regions of dynamic objects found in step 2.
      ii. If not, define the data energy costs according to the `C2` case of Eq. (3). Otherwise, according to the `C1` case.
   (b) Define the smooth energy term $E_{\text{smooth}}(\mathcal{I})$. For each pixel pair $(\mathbf{x}, \mathbf{y})$, the smooth energy $E_{\text{smooth}}(\mathbf{x}, \mathbf{y})$ is defined as the sum of $C(\mathbf{x})$ and $C(\mathbf{y})$ computed in step 1.
   (c) Set the initial labels of all pixels as $\mathbf{I}_1$ or $\mathbf{I}_2$. Optimize the last defined energy function Eq. (2) via graph cuts, and find the last optimal seamline.

graph cuts, named as "GrabCut". Kolmogorov and Zabih [15] 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 or min-cut algorithm [4]. 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:

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

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 are 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 we can detect the optimal seamline globally. The time complexity of graph cuts is $O(n^2 E)$, 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 Fig. 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 $E_{\text{data}}(\mathcal{I})$ and the smooth energy term $E_{\text{smooth}}(\mathcal{I})$. The data energy term represents all energy costs for individual pixels with one of input source images. The smooth energy term represents all energy costs between adjacent pixels. The total energy cost $E(\mathcal{I})$ is defined as:

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

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

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

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 as the following two cases:

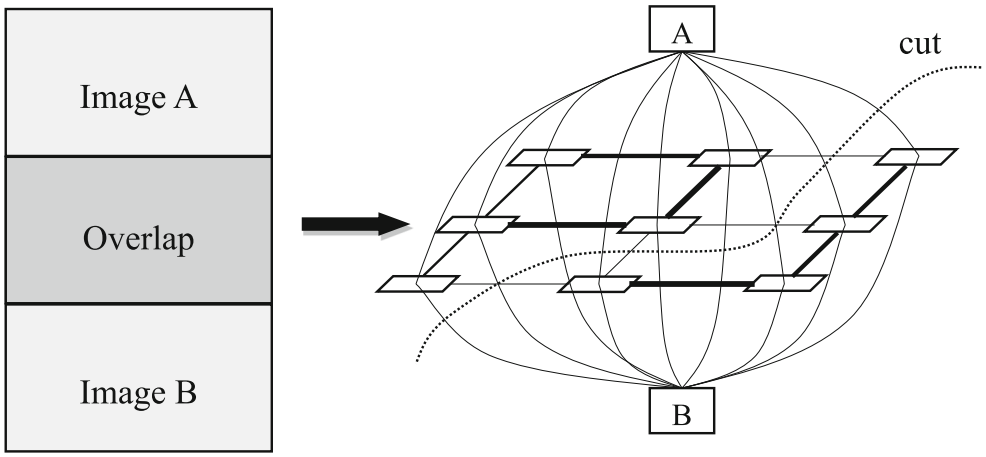


Fig. 2 An illustration example of the optimal seamline detection method via graph cuts. The thickness of lines between adjacent pixels represents the value of the energy cost and the "cut" denotes the minimum cut, which means the optimal seamline

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 Sect. 4.

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

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

where $\mathcal{N}(\mathcal{I})$ denotes the set of all adjacent pixel pairs 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, $\sigma(\mathbf{x},\mathbf{y}) = 1$. $E_{\text{smooth}}(\mathbf{x},\mathbf{y})$ represents the smooth energy between two adjacent pixels $\mathbf{x}$ and $\mathbf{y}$, which is defined as $E_{\text{smooth}}(\mathbf{x},\mathbf{y}) = C(\mathbf{x}) + C(\mathbf{y})$, where $C(\mathbf{x})$ and $C(\mathbf{y})$ are the energy costs of the pixels $\mathbf{x}$ and $\mathbf{y}$, respectively, and will be defined in Sect. 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:

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

3.1 Intensity and gradient difference

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

$$C_c(\mathbf{x}) = |\mathbf{I}_1^g(\mathbf{x}) - \mathbf{I}_2^g(\mathbf{x})|, \tag{6}$$

where $\mathbf{I}_1^g(\mathbf{x})$ and $\mathbf{I}_2^g(\mathbf{x})$ 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 terms of intensity and gradient differences defined above, in theory, the optimal seamlines detected via graph cuts can locate on 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 also suitable to be crossed by the seamline due to that the image differences in these regions are not easy to be observed. However, we find that sometimes the seamlines detected based on above two differences fail to cross those regions. The major reason of this problem is that those regions also have large intensity and gradient differences in some cases. But, we observe that texture complexity of those regions is poor. Thus, to solve this problem, we propose a new texture

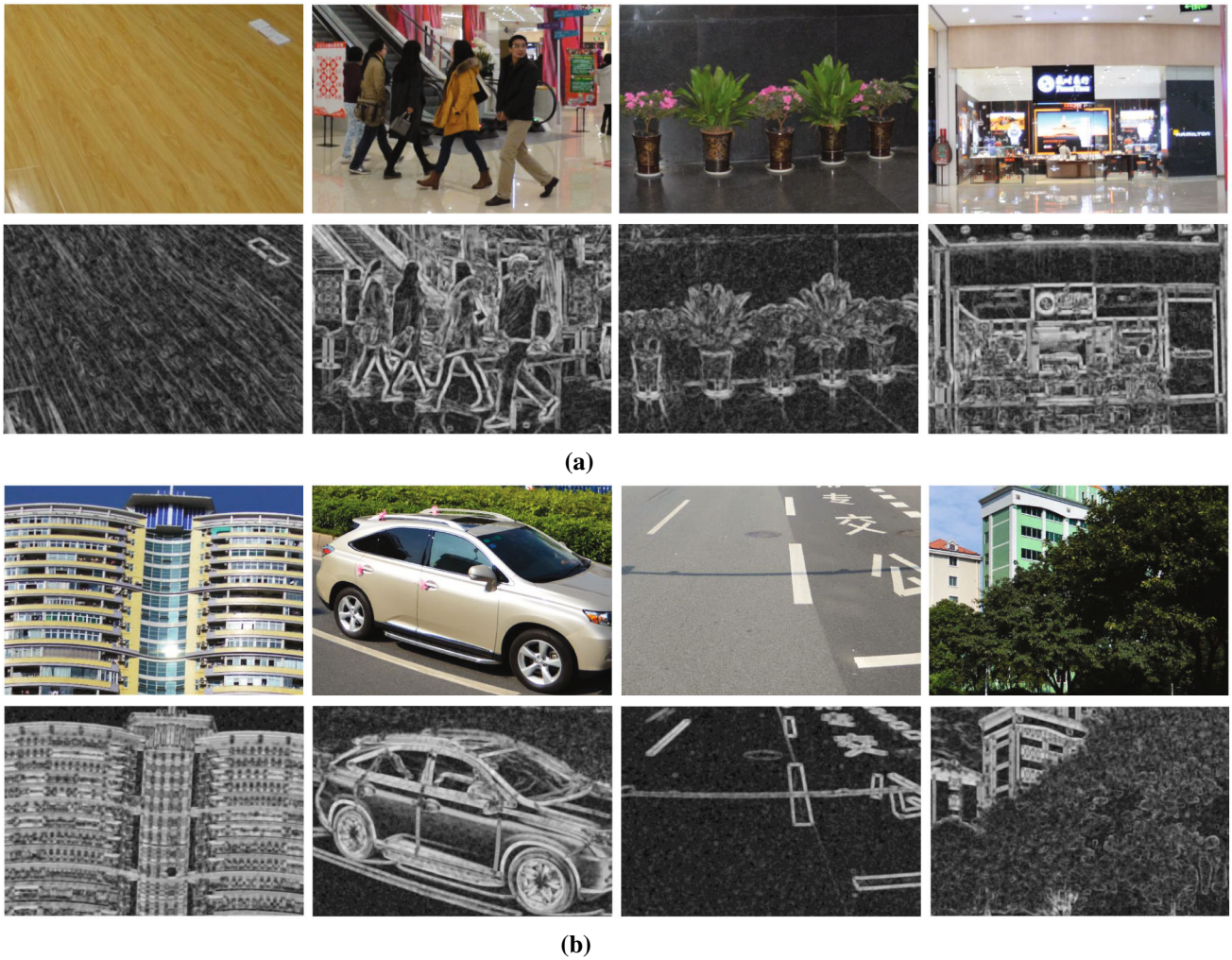


Fig. 3 Some typical regions (top) existed in indoor/outdoor scenes and their corresponding normalized texture complexity maps (bottom) where the lighter regions indicate higher texture complexities. **a** Indoor scenes. **b** Outdoor scenes

complexity measurement to distinguish those regions, which is inspired by histogram of oriented gradient (HOG) feature descriptor. HOG has been widely and successfully used in computer vision and image processing for the purpose of object detection, such as human detection [32].

The texture complexity $\Gamma(\mathbf{p})$ of the pixel $\mathbf{p}$ in the overlap region is calculated as follows. At first, all 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 gradient, 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 bth 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 $\mathbf{x}$ locates on 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 are 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 complexities 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 with-

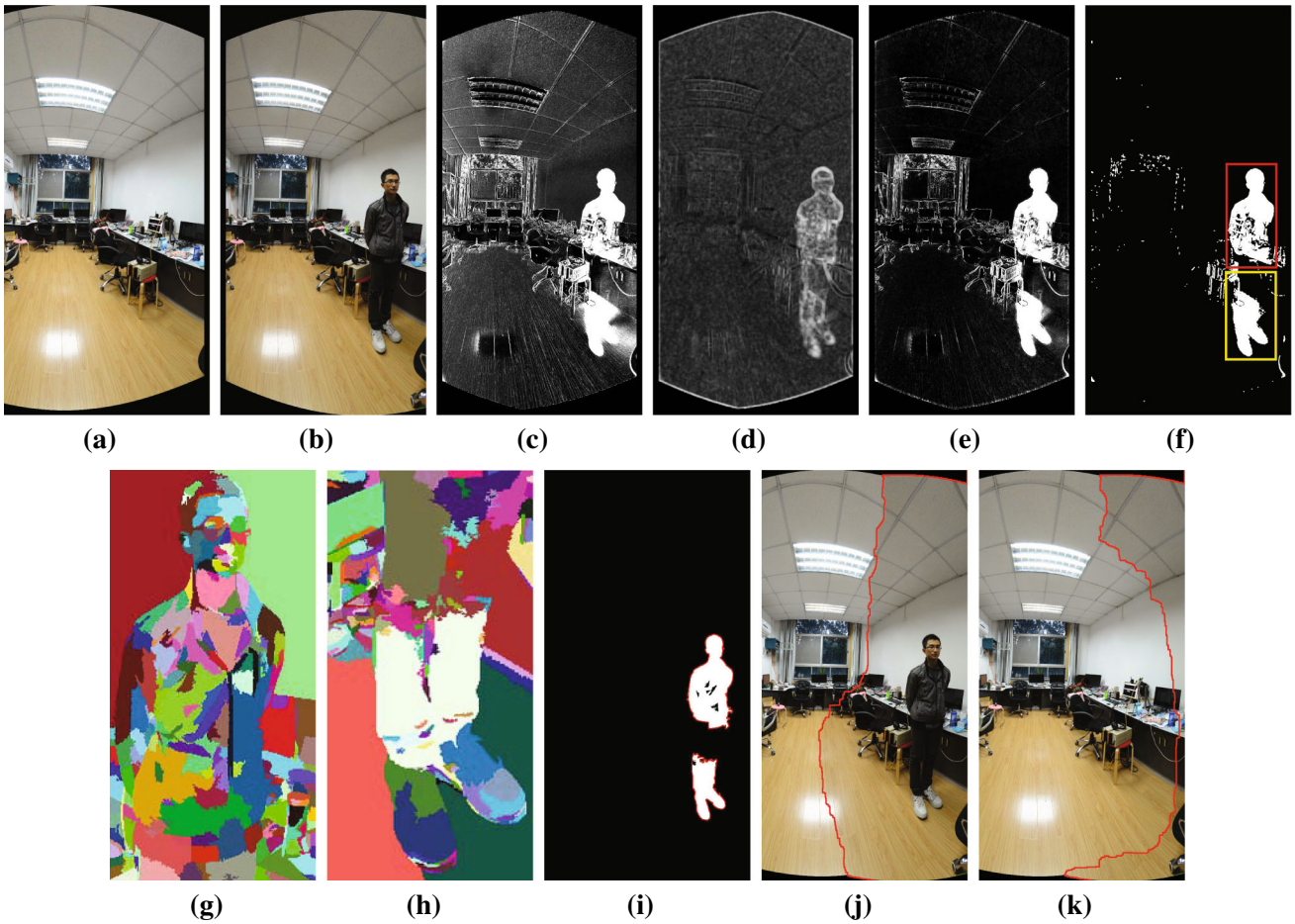


Fig. 4 **a**, **b** Two geometrically aligned input source images with a dynamic object (a pedestrian); **c** the color distance map in CIELAB; **d** the HOG distance map; **e** the final distance map; **f** the dynamic objects binary map after mathematical morphology; **g**, **h** the corresponding 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 (color figure online)

out 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 difference in the overlap region 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 [1]. 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 to clearly detect the dynamic objects from the image distance map only based on the above defined color distances, as shown in Fig. 4c. 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 Sect. 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 follows:

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

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 Fig. 4d. By combining $D_c(\mathbf{x})$ with $D_h(\mathbf{x})$, 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 Fig. 4e, we can simply get the binary map with a distance threshold. We further filter out small noise regions via mathematical morphology, as shown in Fig. 4f. 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 [7], respectively. We then find the contour of O_k, as shown in Fig. 4i. 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 the 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 $P_1(O_k) = \frac{M_1(O_k)}{M_1(O_k)+M_2(O_k)}$, $P_2(O_k) = \frac{M_2(O_k)}{M_1(O_k)+M_2(O_k)}$, respectively, and $P_1(O_k) + P_2(O_k) = 1$. For example, in Fig. 4f, there are two dynamic regions, masked by red and yellow rectangles, respectively. We observe 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 Fig. 4g, h. In this way, we obtain the refined dynamic objects map as shown in Fig. 4i.

5 Experimental results and analysis

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. Those images were first aligned and warped into a common coordinate system via the popularly used open-source library *Panorama Tools*,[1] which also serves as the underlying core engine for many image stitching softwares, such as *PTGui* [2] and Hugin.[3] Here, we only used the modules and functions of image alignment and image projection of *Panorama Tools* to generate the aligned images. And the color or another features of input fish-eye images are not changed. Of course, there always exist geometric misalignments between those 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. In addition, we also applied the structural similarity (SSIM) index [26] to evaluate the quality of the detected seamline. Our algorithm was implemented with C++ under Windows and tested in a computer with an Intel Core i7-4770 at 3.4GHz.

5.1 Quality assessment

After image mosaicking with the found optimal seamline, we need to evaluate the quality of the mosaicked image. In the last several decades, a great deal of effort has gone into the development of the image quality assessment methods that take advantage of known characteristics of the human visual system (HVS). Wang et al. [26] proposed a noticeable quality metric named as the structural similarity (SSIM) index. In last several years, the SSIM index has been widely used to evaluate the quality of color correction [28] and image mosaicking methods [11,22]. In this paper, we also applied the SSIM index to evaluate the quality of image mosaicking along the detected seamline. Let $\mathbf{S}$ be the found optimal seamline between the image pair $\mathbf{I}_1$ and $\mathbf{I}_2$, and $\mathcal{I}$ be the last mosaicked image with the seamline $\mathbf{S}$. The quality measurement SSIM($\mathcal{I}$) used in this paper is defined as:

$$\text{SSIM}(\mathcal{I}) = 2 \times \frac{1}{N}\sum_{i=1}^{N} \min(\text{SSIM2}_k(A_i, B_i)) - 1; k = 1, 2, \tag{13}$$

where $\text{SSIM2}_k(A_i, B_i)$ denotes the normalized SSIM index between two local window blocks A_i and B_i, centered at the ith point of $\mathbf{S}$, from two images $\mathbf{I}_k$ and $\mathcal{I}$, and N is the point number of the seamline $\mathbf{S}$. Since SSIM index $\text{SSIM}_k(A_i, B_i)$

[1] http://panotools.sourceforge.net/.

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

[3] http://hugin.sourceforge.net/.

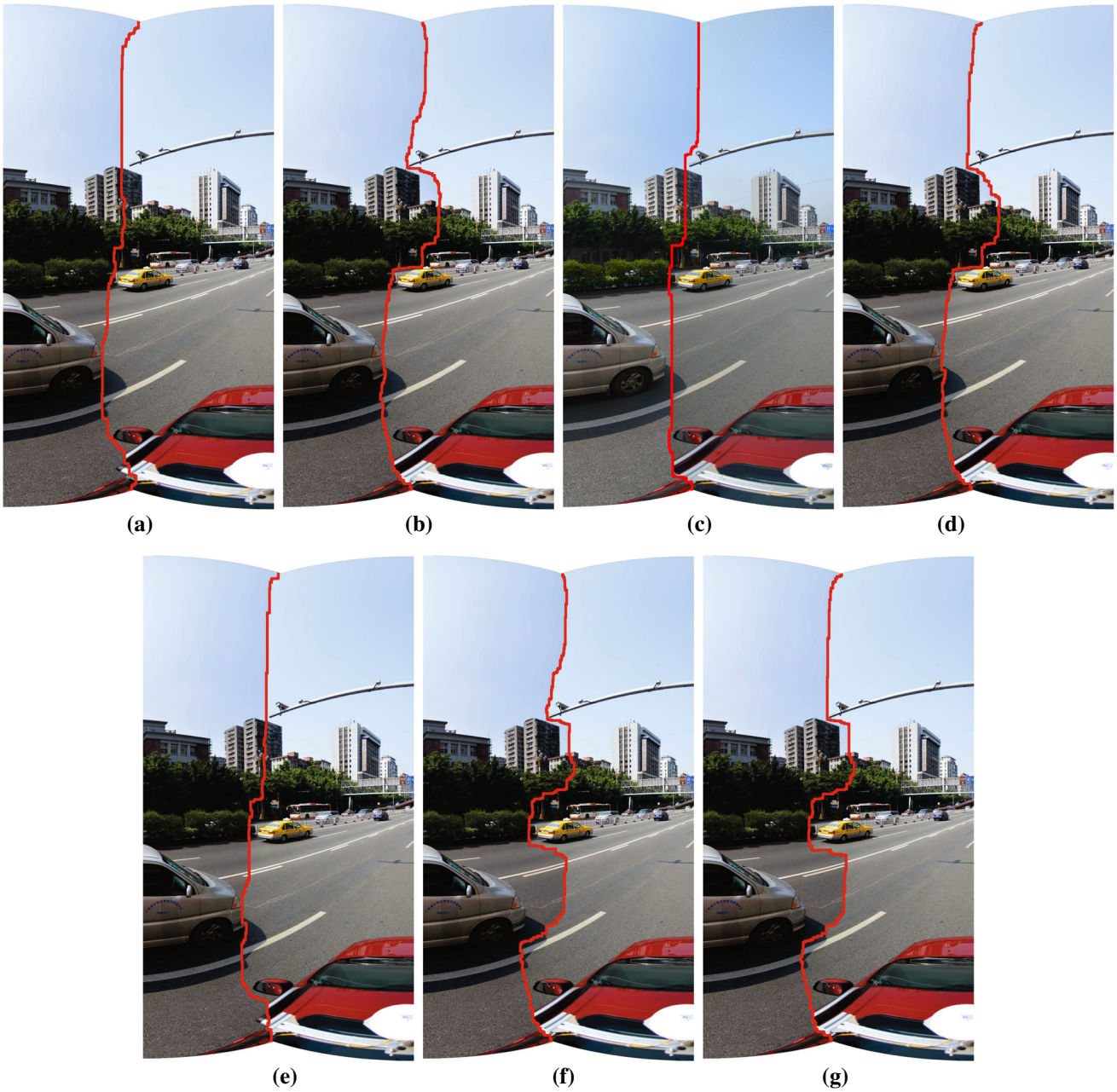


Fig. 5 The seamline detection results with the different combination of intensity (A), gradient (G), and texture complexity (T): **a** A; **b** G; **c** T; **d** $A+G$; **e** $A+T$, **f** $G+T$; **g** $A+G+T$. The computational times (consisting of all elapsed times in the energy cost computation and the graph cuts optimization) of seven combinations in **a**–**g** are 6.4840, 8.3790, 7.0290, 8.3280, 6.5910, 7.3190, and 7.3710 s, respectively. **a** A (6.4840 s). **b** G (8.3790 s). **c** T (7.0290 s). **d** $A+G$ (8.3280 s). **e** $A+T$ (6.5910 s). **f** $G+T$ (7.3190 s). **g** $A+G+T$ (7.3710 s)

belongs to $[-1, +1]$, so we modify the classical SSIM index to $\mathrm{SSIM2}_k(A_i, B_i) = (\mathrm{SSIM}_k(A_i, B_i) + 1)/2$. In addition, the SSIM index is computed independently in each channel of the image in the RGB color space, and then the final SSIM index $\mathrm{SSIM}_k(A_i, B_i)$ is computed by averaging them. The SSIM index itself is defined as a combination of luminance, contrast, and structure components:

$$\mathrm{SSIM}(A, B) = [l(A, B)]^{\alpha} \cdot [c(A, B)]^{\beta} \cdot [s(A, B)]^{\gamma}, \quad (14)$$

where $l(A, B) = \frac{2\mu_a\mu_b+C_1}{\mu_a^2+\mu_b^2+C_1}$, $c(A, B) = \frac{2\sigma_a\sigma_b+C_2}{\sigma_a^2+\sigma_b^2+C_2}$, $s(A, B) = \frac{\sigma_{ab}+C_3}{\sigma_a\sigma_b+C_3}$, μ_a and μ_b are the mean luminance values of the local window blocks A and B, respectively, and σ_a and σ_b are the standard variances of the blocks A and B, respectively. σ_{ab} is the covariance between the blocks A and B. Here, the small constants C_1, C_2, and C_3 are included to avoid the divide-by-zero error, α, β, and γ are three parameters used to adjust the relative importance of three

components. According to the method presented in Wang et al. [26], we used the default settings: $C_1 = (0.01 \times L)^2$, $C_2 = (0.03 \times L)^2$, $C_3 = C_2/2$, $L = 255$ for images of the intensity range [0, 255] and $\alpha = \beta = \gamma = 1$. The higher value of SSIM($\mathcal{I}$) indicates the higher quality of the detected seamline.

5.2 Results

To illustrate whether our energy criteria defined in Sect. 3 is effective, we compared the seamline detection results by using different combinations of intensity (A), gradient (G), and texture complexity (T), as shown in Fig. 5. Thus, there are seven kinds of combinations of features. We observed from Fig. 5 that the seamlines detected by using A, G and $A + G$ without considering texture complexity (T) cannot avoid crossing the gray car. However, when the texture complexity (T) was considered, both $G+T$ and $A+G+T$ can generate high-quality seamlines which avoided crossing the obvious objects and crossed the regions with high similarity, and only $A + T$ failed. This is mainly due to the fact that the intensity of the gray car is similar to the intensity of the road. However, we also found that the seamline detected by using T cannot avoid crossing the building as A did, but it can avoid crossing the gray car, as shown in Fig. 5c. We also observed that the seamlines produced by using $G + T$ and $A + G + T$ are similar, but according to the previous works completed by Mills and Dudek [20] and Zeng et al. [29], the combination of intensity and gradient information is more robust. In the aspect of computational cost, when the texture complexity was considered, the times of $G+T$ and $A+G+T$ are 7.3190 and 7.3710 s, respectively. All of those times are less than the times of G and $A + G$, which are 8.3790 and 8.3280 s. And the times of $A + T$ and A are 6.5910 and 6.4840 s, respectively, which are almost the same, but they are less than the time of T, 7.0290 s. Although the energy computation time has increased due to the use of texture complexity, the optimization computational time of the seamline detection has decreased. This is mainly due to that the optimal solution can be more easily found via graph cuts when the texture complexity was used.

Two geometrically aligned images captured in a shopping center with multiple dynamic objects, as shown in Fig. 6a, b were used to test our algorithm. In this experiment, we compared our proposed algorithm with the popularly used commercial software *PTGui* and the Zeng et al.'s approach [29]. From Fig. 6a, b, we observed that there are many dynamic objects in the overlap region, 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 Fig. 6c. This problem can be solved well via the optimal seamline detection algorithm, as shown in Fig. 6d, e. However, the moving objects always appear in the final image mosaics. But, it disappeared when the proposed dynamic objects compensation was applied, as shown in Fig. 6f. From Fig. 6e, f, we also observed that the seamlines can always pass through the regions with high image similarity and simultaneously avoid crossing dynamic objects; this is mainly due to the constraint of our proposed energy criteria defined in Sect. 3. Another example is shown in Fig. 7. The main dynamic object in this indoor scene is a walking man. And the similar results can be observed as shown in Fig. 6. In addition, we presented the results of the quantitative assessment with the SSIM index for the seamlines detected by different methods in Table 1. We found that the qualities of the seamlines detected by our proposed method without dynamic objects compensation and Zeng et al.'s approach are almost the same. In Fig. 7, the seamline detected by our proposed method without dynamic objects compensation is the best. And, in Fig. 6, the seamline detected by Zeng et al.'s approach is the best; the score of our seamline detected without dynamic objects compensation is only a little less than the score of Zeng et al.' approach. But, with the use of dynamic objects compensation, the qualities drop a little due to the fact that the seamlines need to cross some regions with low image similarity for compensating the dynamic regions. From the visual comparison, we proved the superiority of our proposed algorithm which can generate 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 scene 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 Fig. 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 avoids crossing this dynamic object, as shown in Fig. 8d.

Figure 9 shows a representative example with the unsuccessful compensation. From Fig. 9a, b, we found that there are two moving objects denoted as **A** and **B** in this scene. In Fig. 9c, we used the geometric center line to replace the seamline, from which we found that one moving object has been crossed. This problem can be solved well by detecting the optimal seamline, as shown in Fig. 9d–f. However, we observed that our algorithm failed to compensate the regions of two moving objects, as shown in Fig. 9f. For the dynamic object **A**, the reason is because this man moved slowly and locally, and the background region cannot be compensated completely. For the dynamic object **B**, our algorithm should compensate the corresponding dynamic region in theory.

Fig. 6 A visual comparison of different algorithms stitching two 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 proposed algorithm without dynamic object compensation and with it, respectively. **a** Left image. **b** Right image. **c** *PTGui*. **d** Zeng et al.'s approach. **e** Without compensation. **f** With compensation

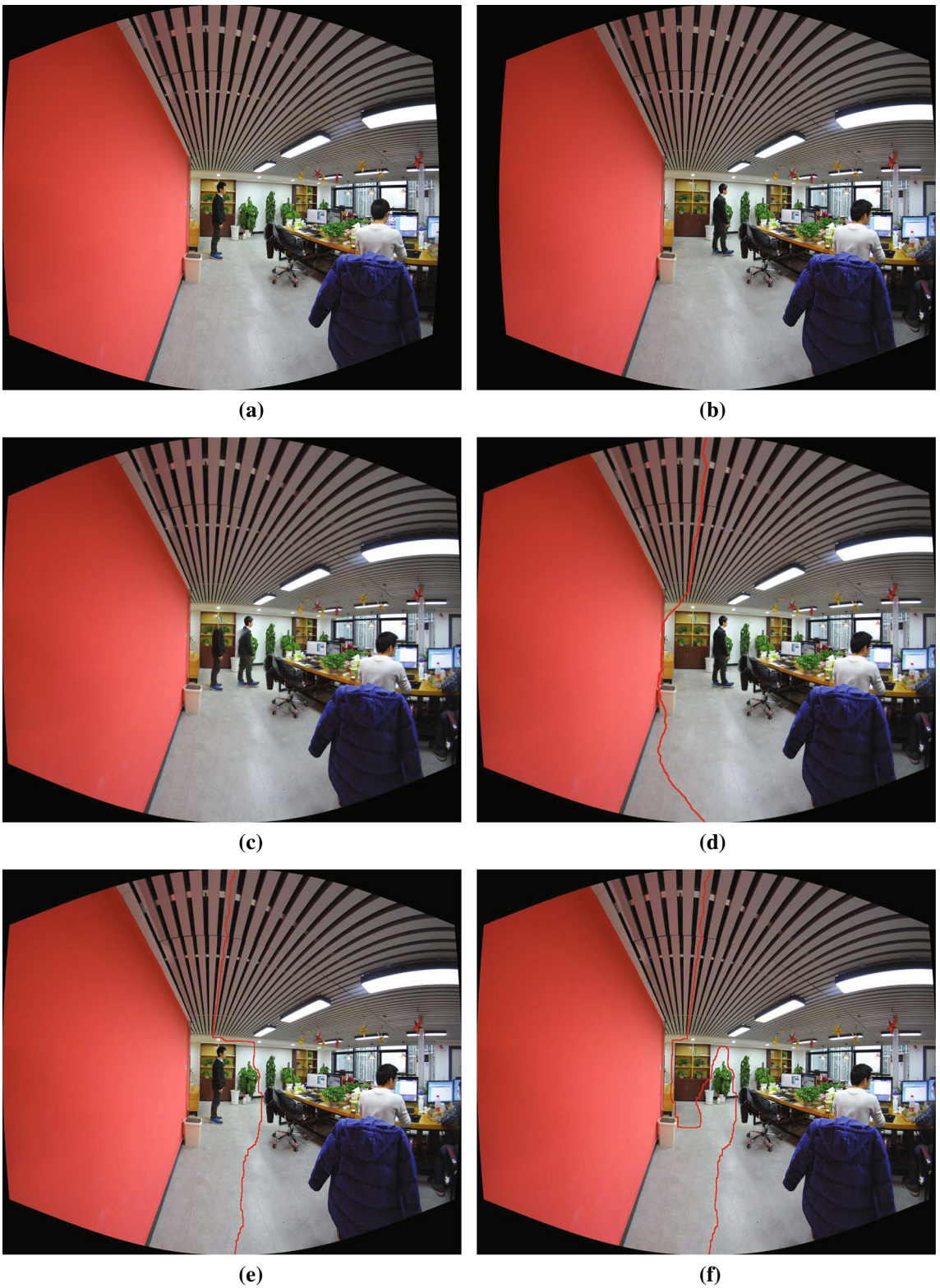


Fig. 7 A visual comparison of different algorithms stitching two input images captured in a small office: **a**, **b** two geometrically aligned input images; **c**, **d** the image mosaics generated by *PTGui* and the Zeng et al.'s approach, respectively; **e**, **f** the image mosaics generated by our algorithm without dynamic object compensation and with it, respectively. **a** Left image. **b** Right image. **c** *PTGui*. **d** Zeng et al.'s approach. **e** Without compensation. **f** With compensation

Table 1 The quantitative quality assessment with the SSIM index of seamlines detected by our proposed method and the Zeng et al.'s approach

	Zeng et al.'s approach	Without compensation	With compensation
Figure 6	0.862721	0.859037	0.80416
Figure 7	0.8737	0.889709	0.860047

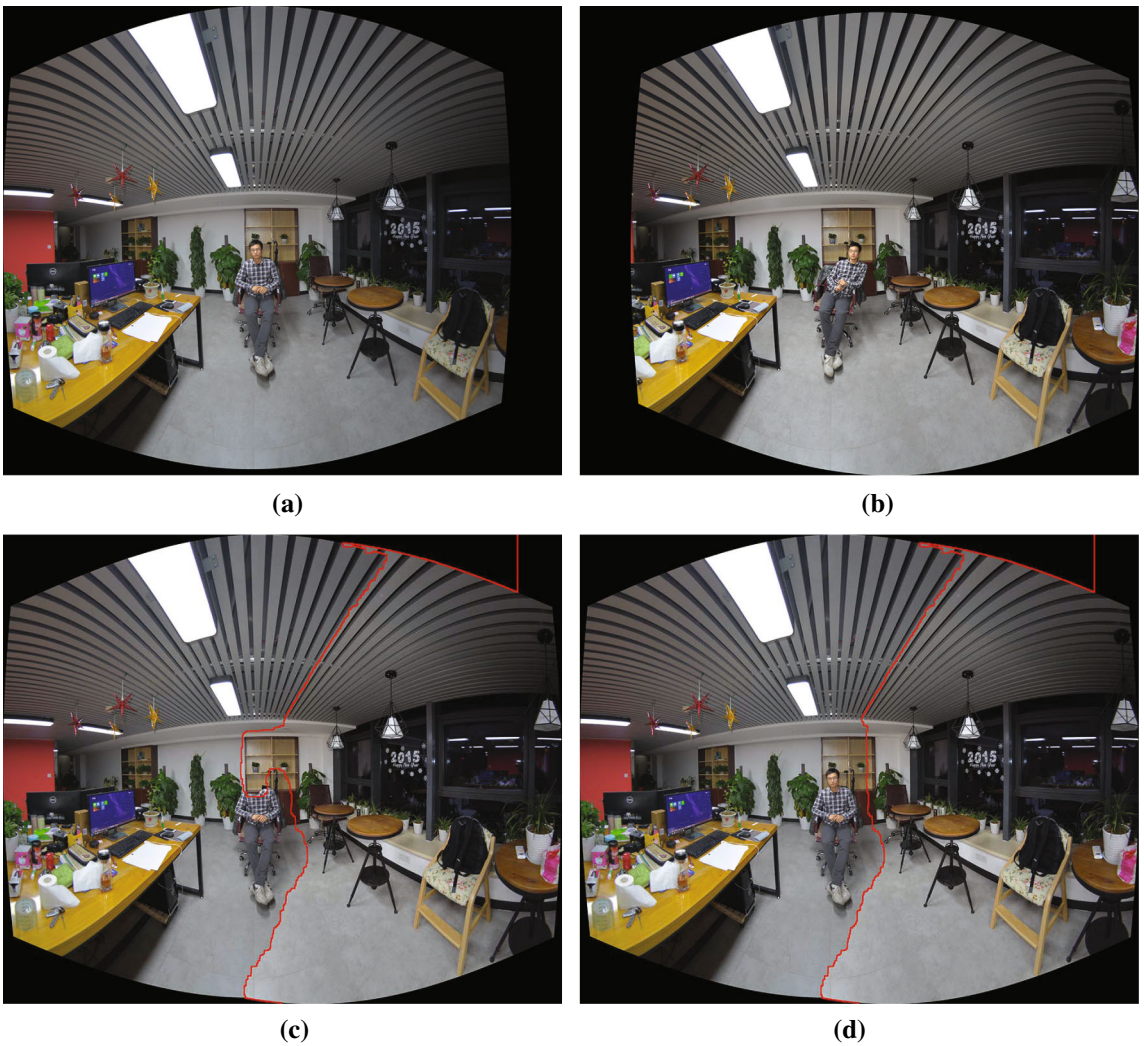


Fig. 8 An example of mosaicking two geometrically aligned images in **a**, **b** captured from a dynamic scene with a local motion of a human head based on the seamlines detected by the hard constraint in **c** and the soft one in **d**, respectively. **a** Left image. **b** Right image. **c** Mosaicked image with hard constraint. **d** Mosaicked image with soft constraint

However, due to that the color of background is similar to this dynamic object, the dynamic region is very difficult to be detected, and our algorithm cannot detect this dynamic object. Therefore, our algorithm failed to compensate the dynamic region covered by **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 did.

Figure 10 shows the 360°-view panoramas of an outdoor scene, which were generated by different algorithms. 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. From the original images, we can find that there is a moving man in this scene. The ghost caused by this moving man appears in the panorama generated by *PTGui*; at the same time, the man appears five times in this scene, as shown in Fig. 10b. In Fig. 10c, d, which are generated by the

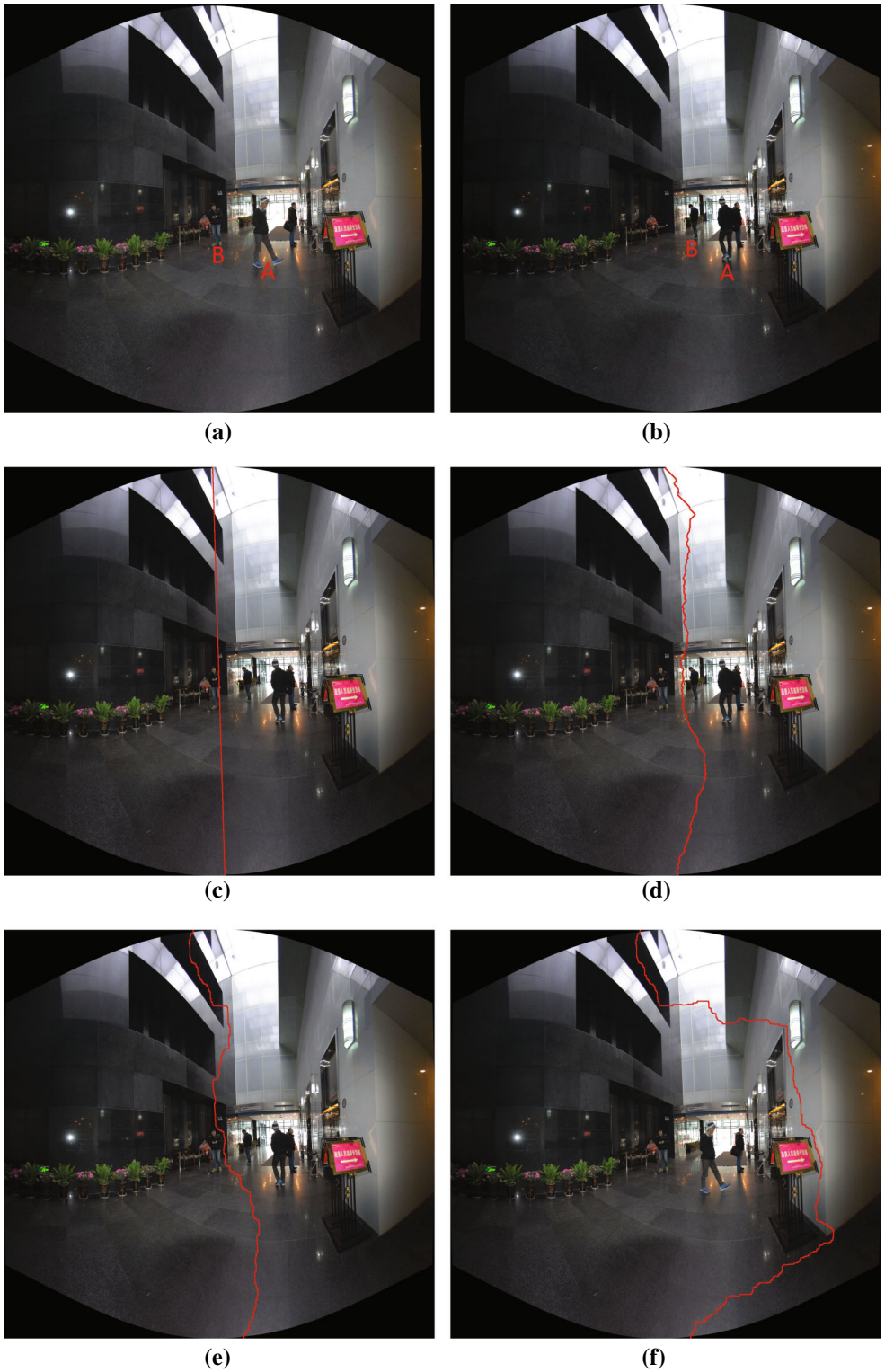


Fig. 9 An example with the unsuccessful compensation: **a**, **b** two geometrically aligned images with two moving objects **A** and **B**; **c** the image mosaicked based on the geometric center line; **d** the composition image generated by the Zeng et al.'s approach; **e**, **f** the image mosaics generated by our algorithm without dynamic objects compensation and with it, respectively. **a** Left image. **b** Right image. **c** Center line. **d** Zeng et al.'s approach. **e** Without compensation. **f** With compensation

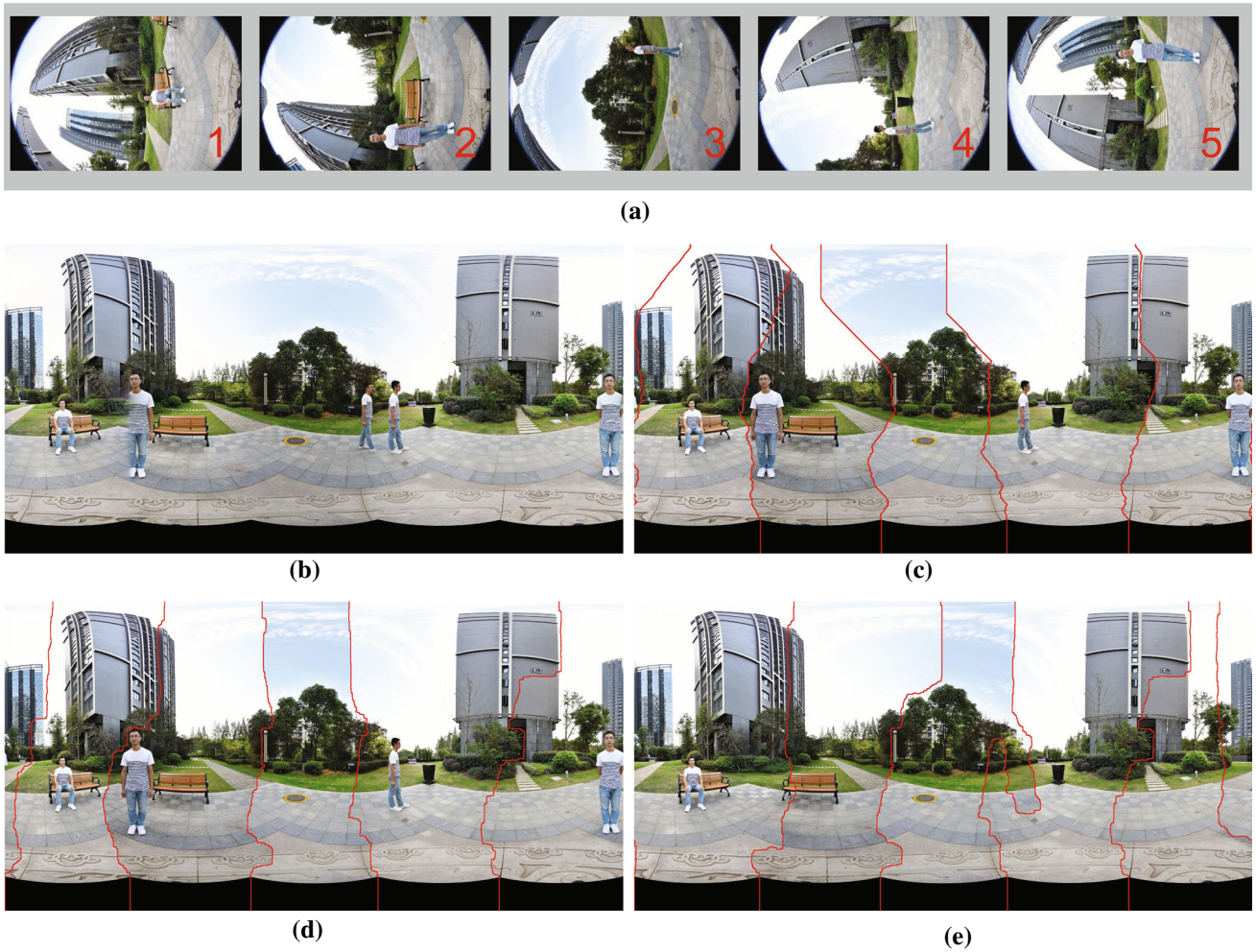
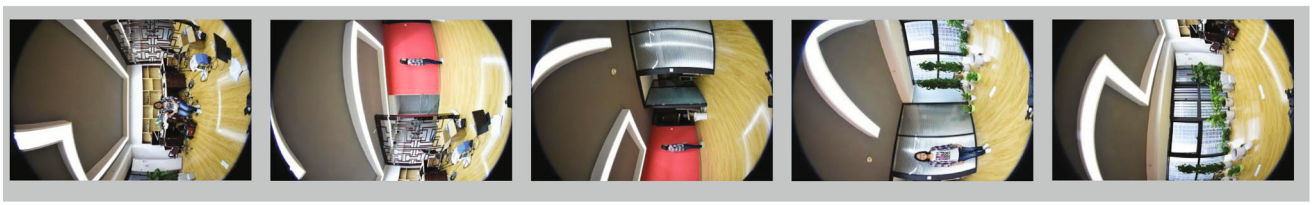


Fig. 10 The 360°-view panorama of an outdoor dynamic scene with a moving man: **a** 5 original images; **b** the image mosaicked by *PTGui*; **c** the image mosaicked by the Zeng et al.'s approach; **d** the image mosaicked by our proposed method without dynamic objects compensation; **e** the image mosaicked by our proposed method with dynamic objects compensation. The computation times of **c**–**e** are 31.4820, 48.213 and 57.129 s, respectively. **a** Original images (3200 × 4800). **b** *PTGui*. **c** Zeng et al.'s approach. **d** Without compensation. **e** With compensation

Table 2 The quantitative quality assessment with the SSIM index of seamlines detected by our proposed method and the Zeng et al.'s approach

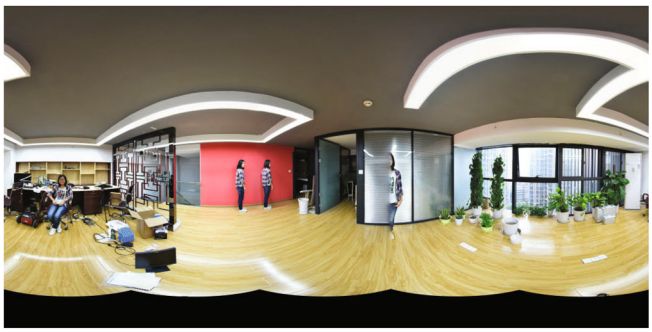
	Zeng et al.'s approach	Without compensation	With compensation
Figure 10	0.590076	0.685662	0.640019
Figure 11	0.674495	0.695654	0.646024

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 ghost disappears, but the man also appears four times in this same scene. This problem can be solved well by our proposed dynamic objects compensation strategy. As shown in Fig. 10e, this man only appears once when dynamic objects compensation was applied. With the use of dynamic objects compensation, the computation time increases to 57.129 s from the original 48.213 s without compensation. The increasing time was mainly spent on detecting the dynamic objects and determining which image they come from. In addition, we found that the computation time of the result shown in Fig. 10c is only 31.4820 s due to the fact that this algorithm applied 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. Particularly, the second seamline (from left

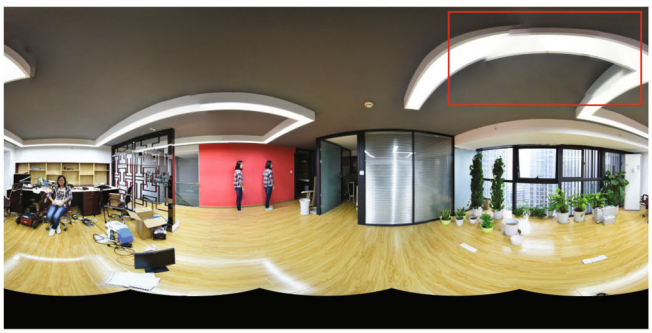


(a)

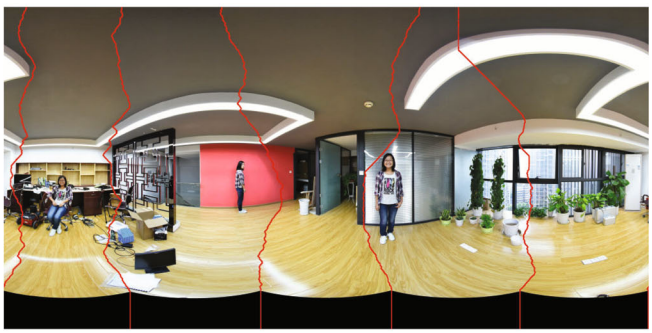
(b)



(c)



(d)



(e)

(f)

(g)

Fig. 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**–**g** are 31.5330, 41.910 and 55.385 s, 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

to right) passes through many edges of the building, but the seamline detected by our proposed algorithm avoids crossing them. And more importantly, the Zeng et al.'s approach cannot avoid this moving man appearing multiple times in the last image mosaic. The results of quantitative assessment with the SSIM index for the seamlines are presented in the second row of Table 2. From Fig. 10, we found that there are 5 seamlines on each stitched image. For each group of seam-

lines in each stitched image, we computed the quality value for each seamline independently, and then, the last quality value was computed by averaging them. We found that our proposed method outperforms than the Zeng el al.'s method, even with the use of dynamic objects compensation.

The 360°-view panorama of an indoor scene is presented in Fig. 11, and the similar conclusions can be drawn as the outdoor scene shown in Fig. 10. In addition, to prove that our algorithm can eliminate 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 Fig. 11b. We also show the image mosaic generated by the open-source software *Enblend* [4] in Fig. 11d, which also applied graph cuts to find the optimal seamlines for image mosaicking. From Fig. 11b, d and especially from 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 eliminate those artifacts caused by geometric misalignments by detecting the optimal seamlines, as shown in Fig. 11f, g. The results of quantitative assessment with the SSIM index for this experiment are presented in the third row of Table 2, and we found that the qualities of the seamlines detected by our proposed method without dynamic objects compensation are the best. With the use of dynamic objects compensation, the seamlines need to cross some regions with the low image similarity for compensating the dynamic objects, so the qualities decrease as we expected.

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 and geometric misalignments. One contribution in this paper is that we applied a new energy cost function combining the intensity difference, the gradient difference, and the texture complexity in the graph cuts energy minimization framework. Thus, our algorithm can ensure that the seamline optimally passes through the regions with high image similarity and avoids crossing obvious objects. Another contribution is that we proposed a novel approach to generate the composite images with clean backgrounds. It first detected 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 determined in probability which image those dynamic objects come from based on segments and contour matching. Finally, we integrated all of those into the seamline detection optimization process. Experimental results have proved that our algorithm can produce high-quality image mosaics free from artifacts and with relatively clean backgrounds (i.e., with dynamic object as few as possible).

Nevertheless, the proposed algorithm can 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 seamline.

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

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

References

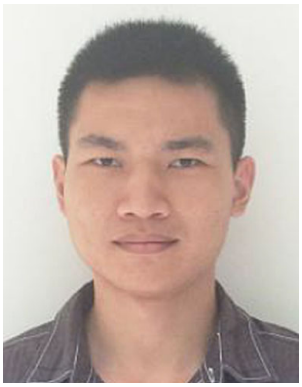
1. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S.: SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Trans. Pattern Anal. Mach. Intell. **34**(11), 2274–2282 (2012)
2. Bellman, R.: Dynamic Programming. Princeton University Press, Princeton (1957)
3. Boutellier, J.J., Bordallo-Lopez, M., Silvén, O., Tico, M., Vehviläinen, M.: Creating panoramas on mobile phones. In: Proceeding of SPIE Electronic Imaging, vol. 6498, issue 7 (2007)
4. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Trans. Pattern Anal. Mach. Intell. **26**(9), 1124–1137 (2004)
5. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23**(11), 1222–1239 (2001)
6. Brown, M., Lowe, D.G.: Automatic panoramic image stitching using invariant features. Int. J. Comput. Vis. **74**(1), 59–73 (2007)
7. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. **24**(5), 603–619 (2002)
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (2005)
9. Davis, J.: Mosaics of scenes with moving objects. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (1998)
10. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**(1), 269–271 (1959)
11. Dissanayake, V., Herath, S., Rasnayaka, S., Seneviratne, S., Vidanaarachchi, R., Gamage, C.: Quantitative and qualitative evaluation of performance and robustness of image stitching algo-

rithms. In: International Conference on Digital Image Computing: Techniques and Applications (DICTA) (2015)
12. Hsieh, J.-W.: Fast stitching algorithm for moving object detection and mosaic construction. Image Vis. Comput. **22**(4), 291–306 (2004)
13. Kass, M., Witkin, A.: Snakes: active contour models. Int. J. Comput. Vis. **1**(4), 321C–331 (1988)
14. Kim, D.-W., Hong, K.-S.: Practical background estimation for mosaic blending with patch-based markov random fields. Pattern Recognit. **41**(7), 2145–2155 (2008)
15. Kolmogorov, V., Zabih, R.: Computing visual correspondence with occlusions using graph cuts. In: IEEE International Conference on Computer Vision (ICCV) (2001)
16. Levin, A., Zomet, A., Peleg, S., Weiss, Y.: Seamless image stitching in the gradient domain. In: European Conference on Computer Vision (ECCV) (2004)
17. Li, L., Yao, J., Liu, Y., Yuan, W., Shi, S., Yuan, S.: Optimal seamline detection for orthoimage mosaicking by combining deep convolutional neural network and graph cuts. Remote Sens. **9**(7), 701 (2017a)
18. Li, L., Yao, J., Xie, R., Xia, M., Zhang, W.: A unified framework for street-view panorama stitching. Sensors **17**(1), 1 (2017b)
19. López, M.B., Hannuksela, J., Silvén, O., Vehviläinen, M.: Interactive multi-frame reconstruction for mobile devices. Multimed. Tools Appl. **69**(1), 31–51 (2014)
20. Mills, A., Dudek, G.: Image stitching with dynamic elements. Image Vis. Comput. **27**(10), 1593–1602 (2009)
21. Philip, S., Summa, B., Tierny, J., Bremer, P.-T., Pascucci, V.: Distributed seams for gigapixel panoramas. IEEE Trans. Vis. Comput. Graph. **21**(3), 350–362 (2015)
22. Qureshi, H., Khan, M., Hafiz, R., Cho, Y., Cha, J.: Quantitative quality assessment of stitched panoramic images. IET Image Proc. **6**(9), 1348–1358 (2012)
23. Rother, C., Kolmogorov, V., Blake, A.: Grabcut: Interactive foreground extraction using iterated graph cuts. In: ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2004) (2004)
24. Tao, C., Sun, H., Yang, C., Tian, J.: Efficient image stitching in the presence of dynamic objects and structure misalignment. J. Signal Inf. Proc. **2**(3), 205 (2011)
25. Tao, M.W., Johnson, M.K., Paris, S.: Error-tolerant image compositing. Int. J. Comput. Vis. **103**(2), 178–189 (2013)
26. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
27. Xia, M., Yao, J., Xie, R., Li, L., Zhang, W.: Globally consistent alignment for planar mosaicking via topology analysis. Pattern Recognit. **66**, 239–252 (2017)
28. Xu, W., Mulligan, J.: Performance evaluation of color correction approaches for automatic multi-view image and video stitching. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2010)
29. Zeng, L., Zhang, S., Zhang, J., Zhang, Y.: Dynamic image mosaic via SIFT and dynamic programming. Mach. Vis. Appl. **25**(5), 1271–1282 (2014)
30. Zhi, Q., Cooperstock, J.R.: Toward dynamic image mosaic generation with robustness to parallax. IEEE Trans. Image Process. **21**(1), 366–378 (2012)
31. Zhou, Z., Li, S., Wang, B.: Multi-scale weighted gradient-based fusion for multi-focus images. Inf. Fusion **20**(1), 60–72 (2014)
32. Zhu, Q., Yeh, M.-C., Cheng, K.-T., Avidan, S.: Fast human detection using a cascade of histograms of oriented gradients. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (2006)

Li Li received B.E. degree and M.E. degree from the School of Remote Sensing and Information Engineering at Wuhan University, China, in 2013 and 2016, respectively. He is pursuing a Ph.D. degree at School of Remote Sensing and Information Engineering, Wuhan University, China. He has published several international conference papers and in several journals and is the inventor of several patents. Currently, he mainly works on Image Mosaicking, LiDAR Data Processing, Robotics, Machine Learning, etc.

Jian Yao received B.Sc. degree in Automation in 1997 from Xiamen University, China, M.Sc. degree in Computer Science from Wuhan University, China, and Ph.D. degree in Electronic Engineering in 2006 from The Chinese University of Hong Kong. From 2001 to 2002, he worked as a Research Assistant at Shenzhen R&D Centre of City University of Hong Kong. From 2006 to 2008, he worked as a Postdoctoral Fellow in Computer Vision Group of IDIAP Research Institute, Martigny, Switzerland. From 2009 to 2011, he worked as a Research Grantholder in the Institute for the Protection and Security of the Citizen, European Commission–Joint Research Centre (JRC), Ispra, Italy. From 2011 to 2012, he worked as a Professor in Shenzhen Institutes of Advanced Technology (SIAT), Chinese Academy of Sciences, China. Since April 2012, he has been a Hubei "Chutian Scholar" Distinguished Professor with School of Remote Sensing and Information Engineering, Wuhan University, China, and the director of Computer Vision and Remote Sensing (CVRS) Lab (CVRS Website: http://cvrs.whu.edu.cn/), Wuhan University, China. He is the member of IEEE, has published over 100 papers in international journals and proceedings of major conferences and is the inventor of over 30 patents. His current research interests mainly include Computer Vision, Image Processing, Deep Learning, and LiDAR Data Processing, Robotics, SLAM, etc.

Haoang Li received B.E. degree from Wuhan University, China, in 2016. He is pursuing a M.E. degree at School of Remote Sensing and Information Engineering, Wuhan University, China. He has published several international conference papers. His current research interests include simultaneous localization and mapping (SLAM), structure from motion (SfM), Image Processing, etc.



Menghan Xia received B.E. degree in Remote Sensing Science and Technology in 2014 and M.Sc. degree in Pattern Recognition and Intelligent System in 2017 from Wuhan University, China. He is pursuing a Ph.D. degree at Department of Computer Science and Engineering, The Chinese University of Hong Kong, China. He has published several international conference papers and in several journals. Currently, he mainly works on Computer Vision, Computer Graphics, Deep Learning.

Wei Zhang is with School of Control Science and Engineering of Shandong University as an associate professor. He received his Ph.D. at The Chinese University of Hong Kong (CUHK). He previously worked as a Postdoc scholar at University of California, Berkeley (UC Berkeley). He is a member of IEEE and has published over 50 papers in computer vision, image processing and machine learning. He received several international awards from IEEE and served for many famous international conferences such as CVPR, ICCV, ECCV, ICIP, ROBIO, ICIA, ICAL.