Journal of Information Hiding and Multimedia Signal Processing ©2016 ISSN 2073-4212
Ubiquitous International Volume **7**, Number **2**, March **2016**

Novel Occlusion Object Removal with Inter-frame Editing and Texture Synthesis

Chuan Qin[1,2*], Haoran Ren[1], Chin-Chen Chang[3], and Qingkui Chen[1]

*Corresponding Author
[1]School of Optical-Electrical and Computer Engineering,
University of Shanghai for Science and Technology, Shanghai 200093, China
E-mail: qin@usst.edu.cn, hr2e14@soton.ac.uk, chenqingkui@usst.edu.cn

[2]Shenzhen Key Laboratory of Media Security
Shenzhen University, Shenzhen 518060, China
E-mail: qin@usst.edu.cn

[3]Department of Information Engineering and Computer Science,
Feng Chia University, Taichung 40724, Taiwan
E-mail: alan3c@gmail.com

Received August, 2015; revised November, 2015

Abstract. *In this paper, an effective occlusion object removal scheme based on inter-frame editing and texture synthesis is proposed, which can be used to realize the seamless filling for the undesired objects during taking photograph. The principle idea of the proposed scheme is to fully taking advantage of the information of recorded frames during the preparing time before pressing the shutter. First, by inter-frame editing, the moving region can be detected and extracted with the frame difference and OR operation. Since the desired foreground object is often stationary, thus, the extracted moving region, i.e., the undesired moving object, can be basically filled with the inter-frame information. The post-processing consisting of the inter-frame and intra-frame texture synthesis can be further utilized to remove residual moving region and undesired stationary object, respectively. Experimental results demonstrate the effectiveness of the proposed scheme.*
Keywords: Occlusion removal, Moving detection, Inter-frame editing, Texture synthesis.

1. **Introduction.** As signal processing technology has rapidly progressed, image acquisition by digital camera has become more and more convenient when we need to record every aspect of our lives [1-2]. However, we sometimes get in trouble when taking a photo that includes something we do not expect and do not want in the photo [3-4]. For example, at a tourist destination, we always seem to get someone we do not know in our photos of the landscape. But, it is not practical to make image capturing scene without any occluding objects. Thus, how to process the acquired data during or after image capturing to remove the occlusions deserves in-depth investigations [5-6].

In recent years, many related works have been proposed to solve the problem of object removal for digital images. Bertalmio et al. developed an image inpainting method based on a partial differential equation (PDE) model [7-8], which can be used to recover images from damaged film, and to remove the texts that were superimposed on the desired images, as well as deleting moving objects from images. The basic idea of this method was to smoothly diffuse the information in the neighborhood of the repairing regions along

the isophote direction. During the process of information diffusion, a third-order PDE should be solved, which involved complex computation. Although this PDE-based image inpainting method can effectively recover the structural information in relatively thinner repairing regions, it cannot deal with the recovery for the larger repairing regions with abundant textural information. Criminisi et al. proposed an exemplar-based method for region filling and object removal [9], which can recover the missing of large textures by sampling the known area. The results obtained by this repairing method, which was based on the sampling texture synthesis, were closely related with the filling order. This method had the following advantages: (1) high efficiency; (2) diffusing the texture information accurately and not producing useless texture information; (3) accurately expanding the information concerning the structure of the line traits. However, it also had the following disadvantages: (1) under the premise of no similar blocks, this method may produce unreasonable results; (2) it cannot expand the information about the curved traits structure; (3) it cannot process the images whose depth information was not clear.

The traditional image inpainting algorithm based on total variation (TV) can remove the occlusions, such as spots, scratches, and scene coverage [10]. By using this algorithm, which was known as the linear filling algorithm, a linear structure (also called as the contour) was used to diffuse from the outer area to the target area. The development of this algorithm was inspired by the recovery algorithm of the PDE in physical heat flow when dealing with certain situations in engineering. But, the main disadvantage of this algorithm was that, when it was used to fill a larger area, the resulting area may be blurred or have other defects. Chan and Shen proposed a new image inpainting algorithm based on the curvature driven diffusion (CDD) [11]. This algorithm can repair the broken edges in the joint regions, but cannot process the textural regions. Heeger and Bergen developed an algorithm called the pyramid-based texture analysis/synthesis [12]. They used an image pyramid and matched the histogram of random noise in image source texture. The advantage of this algorithm was that, it can produce satisfactory results in stochastic textures. Nevertheless, if the histogram matching failed, it may capture more structural textures [13]. Patwardhan et al. presented an approach to separate objects in the foreground from the background with the help of optical flow [14]. According to the reference of optical flow and the pixel values, the selected candidates were utilized to fill the damaged image patches on the boundary of the hole. This approach produced very few ghost shadows, and it can deal with different camera motions. However, it had two disadvantages, i.e., (1) it cannot handle the cases in which a significant portion of the object was missing and (2) it had the difficulty in repairing the curve structures.

Most of the above mentioned methods are time-consuming, which often cannot process the images in real-time during taking photograph, and they are not suitable to deal with the removal operation for occlusion object with larger area. Therefore, in this paper, we propose an occlusion object removal scheme using inter-frame editing and texture synthesis, which can effectively solve the problem of undesired object occlusion in real-time during taking photograph. The principle idea of the proposed scheme is to fully take advantage of the information of recorded frames during the preparation time before pressing the shutter. Thus, the relationship between the recorded frames is fully considered and exploited, and also the continuity in each frame is also utilized for the post-processing to produce a visually satisfactory processed image.

This rest of this paper is organized as follows. Section 2 first gives the framework of the proposed scheme. Section 3 describes the proposed occlusion removal method based on the inter-frame editing. The post-processing based on texture synthesis is presented in Section 4. Section 5 gives the experimental results and analysis, and Section 6 concludes the paper.

2. **Framework of the Proposed Scheme.** The two main procedures of the proposed occlusion object removal scheme are the inter-frame editing and the post-processing based on texture synthesis. Detailedly, the inter-frame editing consists of detection of moving region, extraction of moving region, and filling of undesired moving region. The post-processing based on texture synthesis includes two processes with the uses of inter-frame and intra-frame information, respectively. Figure 1 shows the basic flowchart of the proposed scheme.

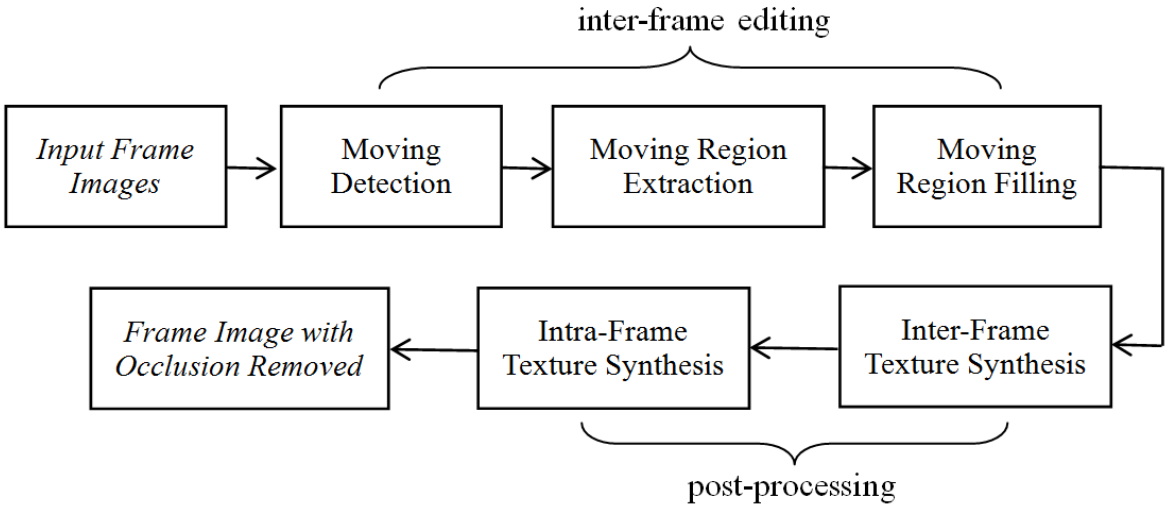

FIGURE 1. Flowchart of the proposed scheme

The initial input of the proposed scheme is a series of frames recorded by the users camera before and just at the time of pressing the shutter. The moving regions of the last frame are first detected and extracted using the frame-difference oriented method. After that, detected moving regions are filled with the available inter-frame information. Then, according to whether the occlusion residuals exist or not, the texture synthesis technique based on the inter-frame and intra-frame information is determined to conduct the post-processing. The final output image is the last recorded frame at the time of pressing the shutter without undesired occlusion objects.

3. **Occlusion Removal by Inter-Frame Editing.** In this Section, the occlusion removal method by using inter-frame editing technique is described. Many undesired occlusions are moving objects during photo taking. Thus, the first thing we should do is to locate the moving region roughly in the scene. Then, the accurate contour of moving object should be quickly extracted. At last, the detected, moving occlusion object is further filled by the inter-frame information.

3.1. **Moving Detection by Frame Difference.** Currently, the popular techniques for moving detection are frame-difference based method, background-estimation based method, and light-flow based method. Due to the extensive computation required, the background-estimation based method and the light-flow based method are not suitable for the real-time situation, such as taking photograph described in this paper. In contrast, in the proposed scheme, we utilize the frame-difference based method for moving detection with the static background.

The frame-difference method for moving object detection is based on the principle that the gray values and the positions of background pixels are unchanged. The frame-difference method takes advantage of the strong correlation between adjacent frames in

frame sequence. The merit of this method is that, it is so fast that it can be suitable for the real-time application. Also, it is not very sensitive to the light changes in the environment. Making use of the gray difference between two consecutive frame images, the motion characteristics of a frame sequence can be obtained. Through comparing the corresponding regions in the frame images at two different times, the regional difference caused by object movement can be detected. During actual calculation, the difference between the two consecutive frame images are computed pixel by pixel, and then, the difference image can be produced. If the absolute value corresponding to one pixel in the difference image is greater than the pre-determined threshold value, set the corresponding pixel value in the resulting image for moving detection as 1, otherwise as 0. In this way, the roughly detected moving region based on the two consecutive frame images can be labeled by the non-zero area in B_k, see Eqs. (1-2).

$$D_k(x, y) = f_k(x, y) - f_{k-1}(x, y) \tag{1}$$

$$B_k(x, y) = \begin{cases} 0, |D_k(x, y) < T| \\ 1, |D_k(x, y) \geq T| \end{cases} \tag{2}$$

where $f_k(x, y)$ denotes the pixel value of the kth frame image at the coordinate $(x,\ y)$, k is the serial number of the frame images, D_k is the difference between the kth and the $(k-1)th$ frame images, B_k is the binary resulting image for moving detection based on f_k and f_{k-1}, and T is the pre-determined threshold.

Although the advantage of the above algorithm is high real-time, the disadvantage is that, it is very sensitive to environmental noise. Thus, the threshold value T is important to this algorithm. If T is set too low, it is insufficient to suppress the noises in frame images. Conversely, if T is set too high, it will ignore useful changes within the frame images. Furthermore, especially for some large moving object that has a consistent color, there may be some holes inside the detected moving region, thus, the binary image Bk for moving detection is just a rough result and cannot extract the moving objects completely. In the following subsection, we will describe how to make the detected moving region more accurately.

3.2. **Extraction of Moving Region.** Since the time interval of the two consecutive frame images are very small, the movement range of the moving object is not particularly large and the roughly detected moving region in B_k may have hollow portions. Thus, after the frame-difference processing in subsection 3.1, we conduct the OR operation for B_k using the consecutive detected binary images to fill the hollow portions. During the process of actual calculations, the OR operation is conducted on the corresponding pixels at the same positions of several consecutive frame-difference binary images. That is to say, as long as only one of the pixels at the same position of several consecutive frame-difference images is equal to 1, the corresponding pixel in the output image after OR operation is equal 1; Otherwise, if all pixels at the same position of several consecutive frame-difference images are equal to 0, the output pixel value is 0, see Eq. (3). This method can effectively fill the hollow portions in the detected moving region.

$$R_k(x, y) = B_{k-p}(x, y) \cup \ldots \cup B_{k-1}(x, y) \cup B_k(x, y) \cup B_{k+1}(x, y) \cup \ldots \cup B_{k+q}(x, y) \tag{3}$$

where $\cup$ denotes the OR operation for two binary values, and R_k denotes the result image after conducting OR operation for B_k by using all corresponding pixels in the $(p + q + 1)$ consecutive frame-difference binary images. In order to further improve the result of Eq.

(3), the median filter with larger window is utilized to filter R_k and remove the residual small holes effectively.

Figure 2 gives an example of moving region extraction. Figure 2(a) shows one frame image recorded by camera before pressing the shutter. Figure 2 (b) is the binary frame-difference image B_k by using Eqs. (1-2) with $T = 3$. Figure 2(c) is the result image R_k by using the OR operation in Eq. (3) with $p = 2$ and $q = 0$, and Figure 2(d) is also the result R_k after OR operation with $p = 14$ and $q = 0$. It can be observed from Figure 2 that, the simple frame-difference method cannot produce satisfactory moving detected result, and the hollow portions in B_k can be basically filled through the OR operation on consecutive frame-difference images.

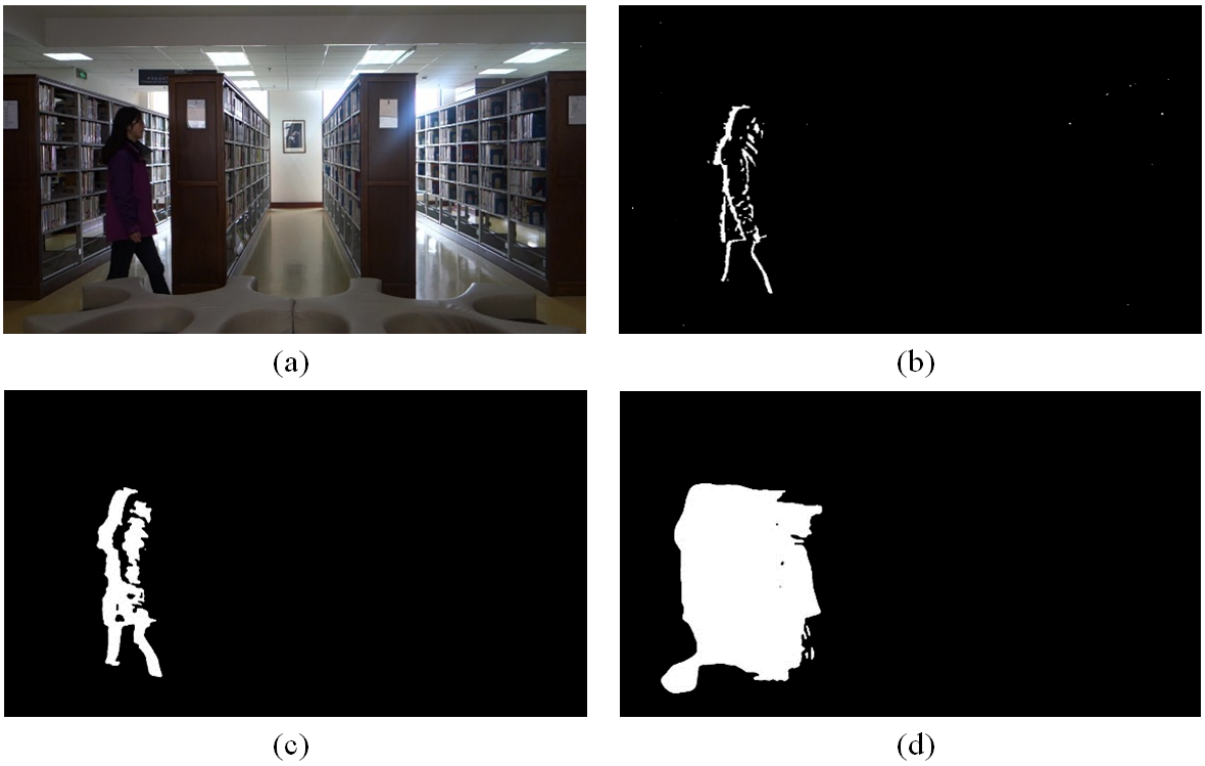

FIGURE 2. An example of moving region extraction. (a) frame image f_k, (b) frame-difference image B_k with $T = 3$ (c) the result image R_k by using OR operation in Eq. (3) with $p = 2$ and $q = 0$, (d) the result image R_k by using OR operation in Eq. (3) with $p = 14$ and $q = 0$

3.3. **Moving Region Filling Using Inter-frame Information.** Based on the processing in subsections 3.1 and 3.2, the resulting image R_k for moving region extraction is composed of the values of 1 and 0, which represent the foreground (moving object) corresponding to the white part and the background (stationary object) corresponding to the black part, respectively. Next, the important problem we are going to address is how to seamlessly remove the white part, i.e., undesired moving object.

After obtaining the result images of moving region extraction for all frame images recorded by the camera, the last frame image, i.e., the frame image captured on pressing the shutter, and its corresponding result image of moving region extraction are both chosen and denoted as f_c and R_c (c is the total number of recorded frame images). The non-zero part in R_c corresponds to the region that we want to remove. The user can also select his/her preferable region for occlusion removal in f_c or R_c. Then, all the other

images R_k $(k = 1, 2, ..., c-1)$ of moving region extraction and f_k are exploited in the descending order of k (k starts from c-1). Detailed steps are described as follows:

Step 1: If the region in R_k corresponding to the non-zero part in R_c has some part with zero values, the region in f_k corresponding to the zero part of R_k is used to replace the corresponding region in f_c.

Step 2: The non-zero part in R_c corresponding to the replaced region of f_c with *Step*1 is then updated to zero.

Step 3: If there is still non-zero part existing in R_c, and k is no less than 2, let $k = k-1$ and go back to *Step*1 for iterations.

Step 4: If there is no non-zero part existing in R_c or k is equal to 1, the iteration of Steps 1-3 stops and the final f_c is the processed image with moving object removal by inter-frame editing.

Figure 3 illustrates a result of moving region filling by using inter-frame information. Figure 3(a) shows the frame image f_c captured on pressing the shutter, and the moving person on the left part of f_c, which occludes the photographing scene, is the object for removal. After using the moving object extraction method with $p = 2$ and $q = 0$, the moving object can be filled using inter-frame information, see Figure 3(b). It can be seen from Figure 3(b) that, most region of the moving object is removed in the processed image. However, some smaller parts of the moving object still remain, which is caused by the incompleteness of the result for moving object extraction. In the following, the post-processing method based on texture synthesis is utilized to solve this problem and achieve better visual effects of object removal.

(a) (b)

FIGURE 3. Result of moving region filling by inter-frame editing. (a) frame image f_c captured on pressing the shutter, (b) processed image after removing the moving object by inter-frame editing

4. **Post-processing Based on Texture Synthesis.** After conducting the occlusion removal operation by inter-frame editing described in Section 3, we can basically remove the undesired moving objects from the captured image. However, in the periphery or inner region of the moving object, some residues and chromatic aberrations may be left. Thus, the post-processing is required to make the result better. In the following, the texture synthesis using inter-frame and intra-frame information is given to complete the whole occlusion object removal procedure.

4.1. **Texture Synthesis Using Inter-frame Information.** Denote the processed result of the inter-frame editing in Section 3 for the frame image captured on pressing the shutter as f_c. The user can choose some residually unprocessed moving regions due to

the missing detection of moving objects in f_c sequentially from left to right and from up to down, and then adopts the texture synthesis mechanism to operate them one by one.

For simplicity, it is assumed that each chosen region in f_c for texture synthesis processing is a block sized 4×4. Denote the number of chosen blocks in f_c for texture synthesis as s. Then, we build one L-shape neighborhood L_c^i for each chosen block $b_c^i (i = 1, 2, .., s)$. Figure 4 gives an illustration, in which the chosen block b_c^i is represented by the block with the red-line border. The L-shape neighborhood L_c^i consists of four 4×4 blocks, which locate on the left, upper-left, up, and upper-right of the chosen block b_c^i for texture synthesis. For each chosen block b_c^i in f_c, we collect all the c-1 blocks b_k^i at the same position in $f_k (k = 1, 2, ..., c-1)$, and their corresponding L-shape neighborhoods L_k^i are also found. The mean square error (MSE) between each of the c-1 L-shape neighborhoods L_k^i and L_c^i is sequentially calculated $(k = 1, 2, .., c-1)$. Note that there are $4 \times 16 = 64$ pixels in each L-shape neighborhood, thus, the MSE calculation is conducted on two corresponding groups of 64 pixels coming from L_k^i and L_c^i, respectively. The L-shape neighborhood L_{k*}^i with the smallest MSE is located, see Eq. (4), and its corresponding block b_{k*}^i is then utilized to replace the block b_c^i for texture synthesis in f_c.

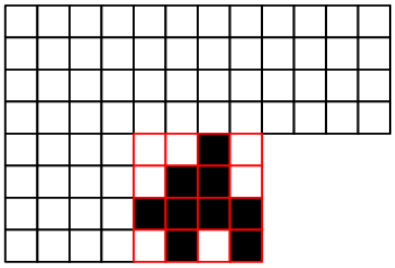

FIGURE 4. An illustration of the L-shape neighborhood for texture synthesis

$$k^* = \arg\min_k \frac{1}{64} \sum \left[L_k^i(x, y) - L_c^i(x, y) \right]^2, \quad k = 1, 2, \ldots, c-1, \tag{4}$$

where the summation $\sum$ is conducted for all 64 pixels with the coordinates (x, y) in the L-shape neighborhoods, i.e., L_k^i and L_c^i. After all chosen blocks b_c^i (i = 1, 2, ..., s) in the frame image f_c are conducted through the above described processing in the raster-scanning order, the texture synthesis using the inter-frame information for f_c finishes, and all the other frames except the processed f_c can be deleted to release the camera memory.

4.2. **Texture Synthesis Using Intra-frame Information.** Although inter-frame editing in Section 3 and inter-frame based texture synthesis in subsection 4.1 can basically remove the occlusion regions caused by moving objects, however, sometimes, some stationary objects existing in captured image are also undesired. Thus, as an effective supplement for occlusion object removal, the texture synthesis using intra-frame information can be utilized to seamlessly remove the stationary objects, which cannot be processed through inter-frame editing and inter-frame based texture synthesis for moving regions.

Different with the texture synthesis using inter-frame information, the texture synthesis using intra-frame information builds all candidate L-shape neighborhoods for searching just within f_c itself. Detailedly, the user first chooses some undesired stationary regions in f_c sequentially from left to right and from up to down, and then adopts the texture synthesis mechanism to operate them one by one. Similar with the previous subsection, we also assume each chosen region in f_c is a block sized 4×4. Then, the L-shape neighborhood of the chosen block is built, and all the other candidate L-shape neighborhoods in f_c are

collected. Through calculating the MSEs between the L-shape neighborhood of the chosen block and all the candidate L-shape neighborhoods in f_c, the 4×4 block corresponding to the candidate L-shape neighborhood with the smallest MSE is used to replace the chosen block for texture synthesis. Due to the characteristic of Markov random field (MRF) for image textural information, the neighborhood similarity can guarantee the seamlessness of the replacement operation.

After all chosen blocks for intra-frame based texture synthesis finish the above processes, the entire procedure for the occlusion object removal (including both moving and stationary objects) completes, and the final produced f_c is the output photographic image without occlusions.

5. **Experimental Results and Analysis.** Experiments were conducted on different frame images from both indoor and outdoor scenes. The resolution of the camera for image capturing had two settings, i.e., 1280×720 and 640×480. The programming environment of the experiments was Visual C++ and Intel OpenCV library on a computer with the 3.3 GHz processor, 4 GB memory, and Window 7 operating system.

5.1. **Results of Occlusion Object Removal in Different Testing Scenes.** First, the overall performance of occlusion removal for both indoor and outdoor scences were given. Figure 5 shows the occlusion removal result for the indoor scence. Figure 5(a) is the frame image that was captured on pressing the shutter. Figure 5(b) is the intermediate processed result by inter-frame editing described in Section 3 with the parameters $T = 3$, $p = 2$, and $q = 0$. It can be observed that, most of the moving object, i.e., red rectangle region, was removed, and only some small residues were left. Figure 5(c) is the final processed result for the removal of residual moving occlusion with texture synthesis by inter-frame information. Because the processed effect after the texture synthesis by inter-frame information was already visually satisfactory, the texture synthesis by intra-frame information was not required. Figure 6 shows the occlusion removal result for the outdoor scence. Figures 6(a-c) are the frame image that was captured on pressing the shutter, the intermediate result after inter-frame editing also with the parameters $T = 3$, $p = 2$, and $q = 0$, and the intermediate result with texture synthesis by inter-frame information, respectively. It can be seen from Figure 6(c) that, after inter-frame editing and texture synthesis by inter-frame information, the occlusions caused by the moving object were totally removed seamlessly. However, the stationary leaf on the road was also undesired in the captured image, which was labeled by the red rectangle in Figure 6(c), thus, the texture synthesis by intra-frame information was then conducted to remove this stationary undesired object, see Figure 6(d).

By comparing the results in Figure 5(b) and Figure 6(b), we can find that the processed result of inter-frame editing for the outdoor scene is better than that of indoor scene. The reason is that, due to the low light, there is more noise in indoor scene than outdoor scene, which leads to the inaccuracy of moving object detection and more residual occlusion regions during inter-frame editing for the indoor scene.

In terms of texture synthesis, the inter-frame based texture synthesis can quickly remove the residual moving region by searching the best matched L-shape block only in the corresponding positions of neighboring frames for fast region replacement, while the intra-frame based texture synthesis should search all L-shape blocks within the current processing frame to find the best matched one for region replacement. Due to the large number of L-shape blocks in the frame image, which is much more than the number of frame images, thus, the computational complexity of the intra-frame based texture synthesis is much higher than that of the inter-frame based texture synthesis. But, the

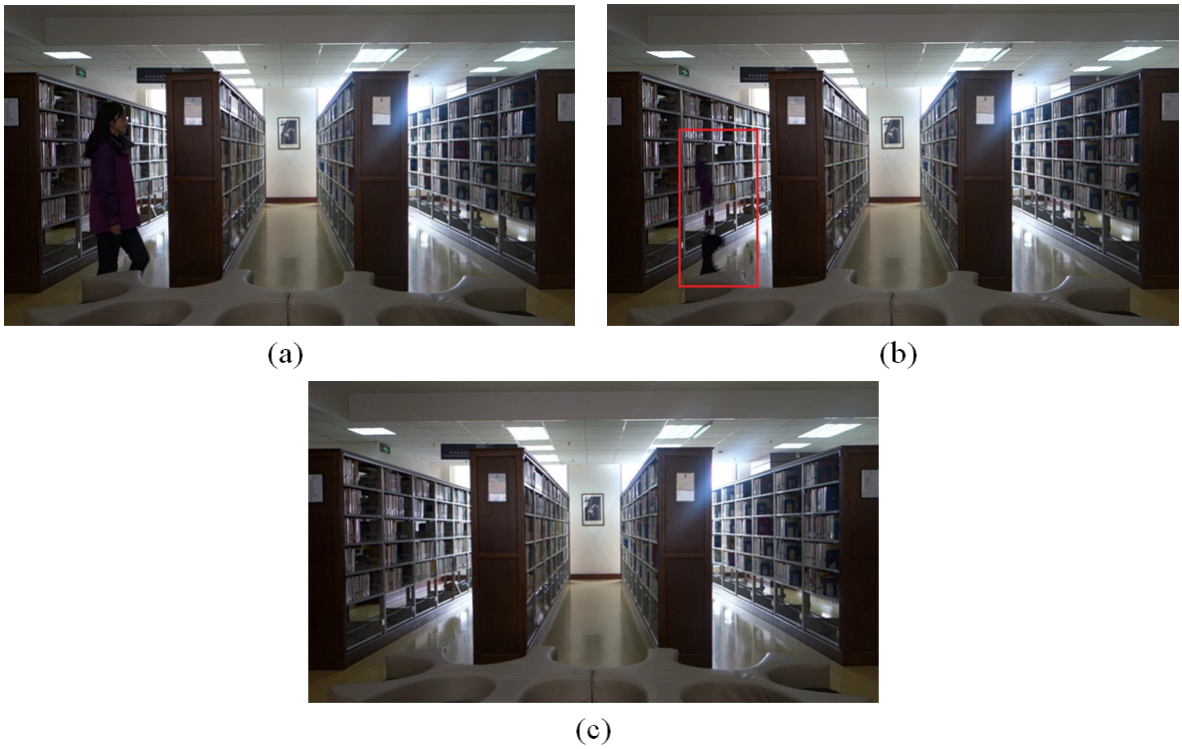

FIGURE 5. Result of occlusion removal for indoor scene. (a) frame image captured on pressing the shutter, (b) the intermediate result after inter-frame editing, (c) final processed result with texture synthesis by inter-frame information

advantage of the intra-frame based texture synthesis is that, it can effectively remove the undesired regions of the stationary objects, see the removal of the leaf on the ground in Figures 6(c-d).

5.2. **Performance Evaluation under Different Parameters.** There are three parameters, i.e., T, p, and q, during inter-frame editing in the proposed scheme. In the practical photograph shooting, among all recorded frame images during one shooting process, we usually adopt the frame image on pressing the shutter as the final output, thus, the parameter q is always set to 0 in real applications. The performance evaluation for the proposed scheme under different T and p are given as follows.

Figure 7 shows the results of the binary frame-difference images B_k for the image in Figure 5(a) by Eqs. (1-2) with different thresholds T. Figure 7(a) is the result with $T = 3$, and (b) is the result with $T = 9$. It can be observed from the results that, smaller threshold T can help to detect more moving regions, but also leads to more noises as false alarms. Though greater threshold T can help to avoid some false alarms of moving detection, however, more actual moving regions are also undetected, which causes some hollow portions in the detected moving object and affects the following operations for occlusion removal. Therefore, in our experiments, the threshold T was set to 3 as the default, and the noises due to false alarms in Bk can be reduced by using the median filter.

As described in Section 3, after the moving detection by frame difference, the result image R_k for moving region extraction by OR operation in Eq. (3) with the parameter p can be obtained, and then, the moving region filling can be achieved through using R_k. Figure 8 shows the results of moving region filling with different thresholds p. Figure 8(a)

FIGURE 6. Result of occlusion removal for outdoor scene. (a) frame image captured on pressing the shutter, (b) the intermediate result after inter-frame editing, (c) the intermediate result with texture synthesis by inter-frame information. (d) final processed result with texture synthesis by intra-frame information

FIGURE 7. Frame-difference image B_k with different thresholds T. (a) $T = 3$, (b) $T = 9$

is the result with $p = 2$, and (b) is the result with $p = 14$. It can be seen from Figure 8 that, the result of moving region filling with p = 14 is better than that of $p = 2$, because

TABLE 1. Residual percentage of moving occlusion object in Figure 5(a) and Figure 6(a)

Parameter	**Figure 5(a)**	**Figure 6(a)**
$p = 1$	9.38%	8.61%
$p = 2$	6.01%	5.37%
$p = 4$	5.77%	5.04%
$p = 8$	2.67%	1.63%
$p = 10$	0.12%	0.04%
$p = 12$	0.01%	0
$p = 14$	0	0

more hollow portions in R_k can be filled to produce R_k through the OR operation with greater p. Table 1 presents the residual percentages in Figure 5(a) and Figure 6(a) after the region extracting and filling for moving occlusion objects with different parameters p. In order to conduct this testing, we utilized the classic image segmentation technique [15] to retrieve the whole region of moving occlusion object and calculated the residual percentages according to the extracted region with Eq. (3) and the segmented region with [15]. Although greater values of p can achieve lower residual percentages of moving occlusion objects, however, it leads to higher computational complexity during the OR operation on the consecutive frame-difference images. Thus, in order to increase the efficiency and avoid the influence on the subsequent photograph shooting, the value of p was set to 2 in the experiments.

(a) **(b)**

FIGURE 8. Result of moving region filling after inter-frame editing under different parameters p. (a) $p = 2$, (b) $p = 14$

5.3. **Analysis for Computational Complexity.** We also evaluated the computational complexity of the proposed scheme. First, the execution time of the moving region extraction during the process of inter-frame editing is given in Table 2. Two typical kinds of frame sizes, i.e., 640×480 and 1280×720 were utilized for testing, respectively. It can be found from Table 2 that, the moving region extraction with $p = 2$, i.e., the OR operation for three consecutive frame images, is significantly faster than that with $p = 14$, i.e., the OR operation for fifteen consecutive frame images. Note that, the time data listed in Table 2 are the average values for a large number of experiments. Therefore, in order to achieve lower computational complexity, the moving region extraction with $p = 2$ was applied though its completeness of the extracted moving region was not superior to that with $p = 14$.

TABLE 2. Average execution time of moving region extraction (unit: second)

Frame Size	**Time with $p = 2$**	**Time with $p = 14$**
640×480	1.093	2.516
1280×720	1.179	5.788

TABLE 3. Average execution time of the three main steps in the proposed scheme (unit: second)

Step Name	**Frame Size**	
	1280×720	**640×480**
Inter-frame editing	1.265	1.328
Inter-frame based texture synthesis	0.382	0.426
Intra-frame based texture synthesis	2.764	9.355

Table 3 gives the average execution time of the three main steps, i.e., inter-frame editing ($p = 2$), inter-frame based texture synthesis, intra-frame based texture synthesis, in the proposed scheme through a number of experiments. Detailedly, the main computation of inter-frame editing was consumed by the extraction of moving region. The main computation of inter-frame based texture synthesis was only the comparison of L-shape neighborhoods among c consecutive frames, thus, its execution speed was the fastest in these three steps. The intra-frame based texture synthesis was the most time-consuming step in the proposed scheme because the most suitable L-shape neighborhood should be searched among a large number of candidates in the current frame. But, in real application, the step of intra-frame based texture synthesis is not absolutely essential unless there are still some undesired stationary objects that are required for removal after the processing of the former two steps. In addition, currently, the proposed scheme was simulated only through relying on the software implementation. In the practical use, the scheme can be incorporated to the hardware by the embedded system programming, which can further improve the computational efficiency.

6. **Conclusions.** In order to solve the occlusion problem caused by undesired objects during taking photograph, a novel occlusion object removal scheme by inter-frame editing and texture synthesis is proposed in this paper. By taking advantage of the preparation time before pressing the shutter, the moving region can be detected and extracted through the frame difference and OR operation. Since the desired foreground object is often stationary, thus, by filling the extracted moving region with inter-frame information, the undesired moving object can be basically removed. The post-processing based on inter-frame and intra-frame texture synthesis can be further utilized to remove the residual moving region due to the missing detection and also the undesired stationary object, respectively. Experimental results show the proposed scheme can efficiently realize the seamless occlusion removal of no matter moving and stationary objects.

Acknowledgements. This work was supported by the National Natural Science Foundation of China (61303203), the Natural Science Foundation of Shanghai, China (13ZR1428400), the Innovation Program of Shanghai Municipal Education Commission (14YZ087), Shanghai Engineering Center Project of Massive Internet of Things Technology for Smart Home (GCZX14014), Research Base Special Project of Hujiang Foundation (C14001), Hujiang Foundation of China (C14002), the Open Project Program of Shenzhen Key Laboratory

of Media Security, and the Open Project Program of the National Laboratory of Pattern Recognition (201600003).

REFERENCES

[1] K. Sharma, I. Moon and S. G. Kim, Extraction of Visual Landmarks Using Improved Feature Matching Technique for Stereo Vision Applications, *IETE Technical Review*, vol. 29, no. 6, pp. 473-481, 2012.

[2] A. Paul, K. Bharanitharan, and J. Wu, Algorithm and Architecture for Adaptive Motion Estimation in Video Processing, *IETE Technical Review*, vol. 30, no. 1, pp. 24-30, 2013.

[3] C. Guillemot and O. L. Meur, Image Inpainting: Overview and Recent Advances, *IEEE Signal Processing Magazine*, vol. 31, no. 1, pp. 127-144, 2014.

[4] C. Qin, S. Z. Wang and X. P. Zhang, Simultaneous Inpainting for Image Structure and Texture Using Anisotropic Heat Transfer Model, *Multimedia Tools and Applications*, vol. 56, no. 3, pp. 469-483, 2012.

[5] T. K. Shih, N. C. Tang and J. N. Hwang, Exemplar-based Video Inpainting without Ghost Shadow Artifacts by Maintaining Temporal Continuity, *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 19, no. 3, pp. 347-360, 2009.

[6] C. Qin, F. Cao and X. P. Zhang, Efficient Image Inpainting Using Adaptive Edge-Preserving Propagation, *The Imaging Science Journal*, vol. 59, no. 4, pp. 211-218, 2011.

[7] M. Bertalmio, A. Bertozzi and G. Sapiro, Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting, *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, Kauai, HI, vol. I, pp. 355-362, 2001.

[8] M. Bertalmio, G. Sapiro, V. Caselles and C. Ballester, Image Inpainting, *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, New Orleans, USA, pp. 417-424, 2000.

[9] A. Criminisi, P. Perez and K. Toyama, Region Filling and Object Removal by Exemplar-based Image Inpainting, *IEEE Trans. on Image Processing*, vol. 13, no. 9, pp. 1200-1212, 2004.

[10] T. F. Chan and J. Shen, Mathematical Models for Local Non-texture Inpaintings, *SIAM Journal on Applied Mathematics*, vol. 62, no. 3, pp. 1019-1043, 2001.

[11] T. F. Chan and J. Shen, Nontexture Inpainting by Curvature-Driven Diffusions, *Journal of Visual Communication and Image Representation*, vol. 12, no. 4, pp. 436-449, 2001.

[12] D. J. Heeger and J. R. Bergen, Pyramid-based Texture Analysis/Synthesis, *Proceedings of International Conference on Image Processing*, vol. 3, pp. 648-651, 1995.

[13] S. Ravi, P. Pasupathi, S. Muthukumar and N. Krishnan, Image Inpainting Techniques - A Survey and Analysis, *International Conference on Innovations in Information Technology*, pp. 36-41, 2013.

[14] K. A. Patwardhan, G. Sapiro and M. Bertalmo, Video Inpainting under Constrained Camera Motion, *IEEE Trans. on Image Processing*, vol.16, no. 2, pp. 545-553, 2007.

[15] M. Kass, A. Witkin and D. Terzopoulos, Snakes: Active Contour Models, *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321-331, 1988.