Exemplar-Based Video Inpainting Without Ghost Shadow Artifacts by Maintaining Temporal Continuity

Timothy K. Shih, *Senior Member, IEEE*, Nick C. Tang, and Jenq-Neng Hwang, *Fellow, IEEE*

***Abstract*—Image inpainting or image completion is the technique that automatically restores/completes removed areas in an image. When dealing with a similar problem in video, not only should a robust tracking algorithm be used, but the temporal continuity among video frames also needs to be taken into account, especially when the video has camera motions such as zooming and tilting. In this paper, we extend an exemplar-based image inpainting algorithm by incorporating an improved patch matching strategy for video inpainting. In our proposed algorithm, different motion segments with different temporal continuity call for different candidate patches, which are used to inpaint holes after a selected video object is tracked and removed. The proposed new video inpainting algorithm produces very few "ghost shadows," which were produced by most image inpainting algorithms directly applied on video. Our experiments use different types of videos, including cartoon, video from games, and video from digital camera with different camera motions. Our demonstration at http://member.mine.tku.edu.tw/www/T_CSVT/web/shows the promising results.**

***Index Terms*—Digital inpainting, image completion, motion map segmentation, object removal, object tracking, video inpainting, video special effect.**

I. Introduction

IMAGE inpainting/image completion [1], [3], [5] is a technique to restore/complete the area of a removed object which is manually selected by the users. The technique produces a reasonably good quality of output on still images. Although there are earlier approaches that focus on removing only small well-selected areas on photographs, the work reported in [1] and [3] produces fairly good results in general cases, especially when applied to large continuous areas. Image inpainting techniques can complete holes based on both spatial and frequency features. Structural properties, such as edges of a house, are extracted from the spatial domain and used to complete an object with its structural property extended [1], [3]. In addition to [1] and [3], another image completion approach [5] uses automatic semantic scene matching to search for potential scenes in a very large image database. The mechanism fills the missing regions (i.e., scenes) using information usually not in the same picture and provides a diverse set of results for a given input. On the other hand, textural information can be propagated from the surrounding areas toward the center of hole such that a seamless natural scene can be recovered [8]. In an inpainting process, in general, the user has to select a target object to be removed (and thus the hole is created). Although object selection is the only step in which the user has to intervene in the completion procedure, many mechanisms suggest that human intelligence can be incorporated to produce a better result [11], [17]. The work discussed in [11] uses an interface to identify a source area, where texture information is used to inpaint another selected target area. The work discussed in [17] further suggests that most natural or artificial objects can be defined by a few main curves. The salient structure of an image should be completed before the texture characteristics are brought in. Therefore, by asking the user to draw a few curved lines, the algorithm proposed in [17] can produce excellent image inpainting results. In general, the problem of image completion can be defined as the following. Assuming that the original image I is decomposed into two parts, $\mathrm{I} = \Phi \cup \Omega$, where Ω is a *target area/hole* manually identified by the user, and Φ is a *source area* with information to be used to complete Ω. Also, there is no overlap between the target area and the source area. These terms (i.e., I, Φ, and Ω) are commonly used in most articles discussing inpainting algorithms. However, when dealing with removing objects from a video sequence, several issues should be further considered. First, manually selecting a target area is impossible due to the number of frames. Second, human recommended structural/textural information is difficult to obtain, even with edge detections. Therefore, the procedure of video inpainting needs to incorporate with a robust tracking mechanism and an effective structural/textural propagation mechanism.

One of the approaches to complete a removed object in video is to directly apply the techniques used in image completion [1], [3], i.e., treating each video frame as an independent image. Most image completion techniques [1], [3] are based on one assumption—the target area Ω has a similar texture and continuous structure from the source area Φ. Therefore, the source and target areas are divided into equal-size patches, with the size of a patch being small (e.g., 3×3 or 5×5 pixels). Patches from the source area, using a sophisticated matching mecha-

Manuscript received May 18, 2007; revised September 16, 2007 and December 12, 2007. First published February 13, 2009; current version published April 01, 2009. This paper was recommended by Associate Editor D. S. Turaga.

T. K. Shih is with the Department of Computer Science, National Taipei University of Education, Taipei 106, Taiwan (e-mail: tshih@cs.tku,edu.tw).

N. C. Tang is with the Department of Computer Science and Information Engineering, Tamkang University, Tamsui 251,Taiwan (e-mail: nickctang@gmail.com).

J. N. Hwang is with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: hwang@u.washington.edu).

Digital Object Identifier 10.1109/TCSVT.2009.2013519

nism, are selected to fill in holes in the target area. Two important measurements are used: priority and confidence values [1]. Priorities of patches that lay on the boundary of source and target areas are computed according to spatial properties of the patches. Confidence values indicate the degree of reliable information of a patch inpainted onto the target area. The fill-in process is repeated from the outer boundary of the target area toward the inner boundary, until the target area is completed. Video inpainting algorithms can be developed based on the above frame-by-frame inpainting approaches.

The video inpainting algorithm discussed in [12] adopts the image completion approach proposed in [1] with static camera assumption. A moving target in a stationary video can be removed by filling in exemplar patches in the same frame onto the missing background. Since the moving target can be partially occluded by another moving object, an inter-frame search is needed to find the best candidate patch. Only the portion of the moving foreground in the selected patch is copied. The priority of the remaining pixels in the background is adjusted to zero such that background is not changed. The video inpainting algorithm for still camera [12] is further extended to cope with nonstationary videos under restricted camera motions [13]. Background and foreground of video are separated, with optical-flow mosaics generated. A similar priority concept used in [12] is also used in [13] to find the highest priority filling-in patches in the foreground. After foregrounds of all frames are inpainted, the remaining background holes are filled using patches directly from adjacent frames and texture synthesis. A few assumptions are made in [13], e.g., camera motion is parallel to the plane of frames (i.e., no intrinsic camera rotations). Also, a moving object may not change its size (i.e., no zooming). Another paper [19] uses a motion layer segmentation algorithm to separate a video sequences to several layers according to the amount of motion. Each separate layer is completed by applying motion compensation and image completion algorithms. Except for the layer with objects to be removed, all of the remaining layers are combined in order to restore the final video. However, temporal consistency among inpainted areas between adjacent frames was not taken care of in [19]. The work discussed in [8] also removes objects from video frame by a priority scheme. Fragments (i.e., patches) on the boundary of the target area are selected with a higher priority. However, a fragment is completed using texture synthesis instead of copying from a similar source. A graph cut algorithm is used to maintain the smooth boundary between two fragments. To maintain a smooth temporal continuity between two target fragments, two continuous source fragments are selected with a preference. However, complex camera motion is not considered in [8].

Another interesting paper [10], which is less similar to our work, proposed detection and reconstruction of missing data in video based on a 3-D autoregressive model. Interpolation instead of patch duplication is used. Usually, only small missing regions can be repaired by interpolation techniques. The work [18] also looks at the problem from a 3-D perspective, which includes pixel positions (2-D) and frame numbers (time). The algorithm optimizes searching of patches at different resolution level. The inpainting results are visually pleasant. The only drawback is that the work [18] assumes the missing hole of every video frame is provided. Therefore, there is no tracking mechanism used to identify the object to be removed.

The inpainting algorithm discussed in [6] repairs static background as well as moving foreground. The algorithm takes a two-phase approach, sampling phase and alignment phase, to predict motion of moving foreground and to align the repaired foreground with the damaged background. Since the algorithm can be extended to use a reference video mosaic, with proper alignment, the algorithm discussed in [6] can also work for different intrinsic camera motions. However, the background video generated based on the reference video mosaic is important. Mistreatment of a generated background video will result in ghost shadows of the repaired video. In [7], the same group of authors [6] further extends their algorithm to deal with variable illumination. Although part of the process (i.e., layer separation and moving foreground sampling) is semi-automatic, the completion process is automatic.

In our earlier investigation [15], we analyze temporal continuities based on stationary and nonstationary video with slow or fast foreground. We use a modified exemplar-based image completion mechanism. More specifically, for stationary videos, patches can be searched from different frames and copied to fill in a target area, which is tracked by a simple computation of optical flow. For nonstationary video, tracking mechanism can be extended to deal with fast or slow moving objects. However, complicated video motions degrade our video inpainting performance, due to the problem of "ghost shadows." That is, due to the temporal discontinuity of inpainted area, flickers can be produced and lead to a visual annoyance. In [16], the authors partially solve this problem by using a motion segmentation mechanism and a modified video inpainting procedure. The inpainted video is improved with less ghost shadows. However, it is still hard to deal with more complicated camera motions.

A. Our Contributions

In this paper, we further improve our earlier results in [15] and [16] by allowing more complicated camera motions in the video inpainting, including zooming, panning and their combinations. The types of video we tested include cartoons from TV, videos generated by computer games, and videos taken using digital camera. From the technical perspective, several achievements are realized:

- a revised image completion algorithm based on edge detection and a new concept called patch template;
- a fast motion search algorithm based on 4SS [14] originally adopted in video compression, due to the characteristics of temporal continuities;
- a new motion segmentation algorithm for object tracking and the separation of video layers;
- an improved video inpainting algorithm dealing with different camera motions under diversified temporal continuities.

The improved video inpainting algorithm can be further extended to deal with more complicated intrinsic camera rotations (i.e., roll, pitch, and yaw), if an effective patch matching mechanism is developed accordingly (e.g., matching skew or rotated patches). Practical applications of our research include special effect production in movie industry, restoring corrupted video

frames in aged movies, and removing and replacing selected objects/actors in videos.

The remainder of this paper is organized as follows. We discuss the modified image completion algorithms in Section II. Temporal continuities are analyzed in Section III, followed by the proposed video inpainting algorithms. Experimental results and analyses are discussed in Section IV before we conclude our discussion and point out possible extensions in Section V.

(a) (b) (c)

Fig. 1. An example of producing edge map. (a) Original Image I. (b) Image Segment I′. (c) Edge Map ***BI***.

II. Image Inpainting Methods

Many image inpainting techniques restore the holes in images by propagating linear structures into the target region via diffusion, which is inspired by the partial differential equations of physical heat flow. The main drawback of these techniques is that the diffusion process introduces some blur, which becomes noticeable when filling larger regions.

Exemplar-based image inpainting [1] is introduced to overcome this drawbacks and can produce a reasonably good quality of output for larger regions on still images. Similar techniques can be adopted to remove an object from a video sequence by combining with an object tracking mechanism to fit the need of video inpainting. Two important techniques introduced in [1] for image inpainting are: *priority map* [i.e., $\mathrm{P(p)}$] and *confidence term* [i.e., $\mathrm{C(p)}$]. The priority map, $\mathrm{P(p)} = \mathrm{C(p)D(p)}$, is defined to be the product of a data term (i.e., $\mathrm{D(p)}$ and the confidence term $\mathrm{C(p)}$. The data term is based on the product of the isophote and a unit vector orthogonal to the front of an inpainted area. The confidence term favors outpointing regions (e.g., sharp bands) in the inpainted area. On the other hand, the data term gives a high priority to inpainted area which has a potential to copy structural information from the source area. The data term and the confidence term together results in a strategy to select the best patch to be inpainted. When the inpainting algorithm selects a patch from the source area, the sum of squared differences (SSD) criterion based on the CIE Lab color space is used. In Section II-A, we introduce a new matching strategy, based on edge detection and a new concept of patch template, to make sure that structural similarity can be achieved.

A. Exemplar-Based Image Inpainting Revised

Our discussion strictly follows the notations introduced in [1]. Let $\boldsymbol{I}$ be the original image (or a frame in a video) which includes a target area, denoted by Ω, to be inpainted and a source area, denoted by Φ, where patches are searched and used. Hence, $\boldsymbol{I} = \Phi \cup \Omega$. However, our approach is different from [1] by starting from using a mean shift region segmentation algorithm based on [2]. Given the resolution of the color frames, the mean shift procedure takes $\boldsymbol{I}$ as input, and produces $\boldsymbol{I'}$ as the result of segmentation. We use parameters $\mathrm{h_s} = 7, \mathrm{h_r} = 6.5, \mathrm{M} = 20$ (see the algorithm discussed in [2]). The edge detection algorithm (not used in [1]) is then used to convert $\boldsymbol{I'}$ to a binary image ***BI***, which represents an edge map (i.e., 0 for background and 1 for edge). Let $\Phi\varepsilon \subset \boldsymbol{BI}$ be the area corresponding to $\Phi \subset \boldsymbol{I}$. $\Phi\varepsilon$ is the edge map of the source region. As an example, Fig. 1(a) is the original image, with segmented image and edge map in Fig. 1(b) and (c), respectively.

After the edge map is obtained, we modify the algorithm discussed in [1]. Let $\delta\Omega$ be a front contour on Ω and adjacent to Φ. We assign the initial confidence term of each pixel in I according to [1]:

$\mathrm{C(p)} = 1.0$ iff $\mathrm{p} \in \Phi$ and $\mathrm{C(p)} = 0.0$ iff $\mathrm{p} \in \Omega$.

Let Ψ_p be a patch centered at pixel $\mathrm{p} \in \delta\Omega$. We also use the same concept to compute the confidence term $\mathrm{C(p)}$

$$\forall \mathrm{p} \in \delta\Omega, \mathrm{C(p)} = \left(\sum \mathrm{q} \in (\Psi\mathrm{p} \cap \Phi)\mathrm{C(q)}\right)/|\Psi\mathrm{p}| \quad (1)$$

where $|\Psi_\mathrm{p}|$ is the area of Ψ_p. The area is equal to nine pixels, i.e., a 3×3 patch is used in our experiments. Essentially, the confidence term in (1) computes the percentage of useful pixels in a patch $\Psi\mathrm{p}$. Useful pixels mean the coverage of source pixels in Φ. These useful pixels are used in the original algorithm proposed in [1] to find the best matched patch in the source area.

Note that the data term proposed in [1] uses an isophote and a unit vector. Data term $\mathrm{D(p)}$ seems to favor in-pointing areas in Ω. However, the confidence term $\mathrm{C(p)}$ seems to favor outpointing areas. However, it is time consuming to compute these two terms for each patch. We propose a new approach which modifies the data term, based on the observation on continuous structure derivation, which we believe to be the most important factor. Our assumptions follow. First, the confidence term summarizes the percentage of useful pixels in a patch. Structural information is not included. For instance, a vertical line and a horizontal line in two different patches (with the same number of useful pixels) could have the same confidence term. We believe the use of confidence term only indicates how much information in a target patch can be used as a reference while we look for similar patches in the source area. This also motivates our second effort in the development of a better matching strategy (i.e., *patch template*).

Instead of computing the isophote, we compute the percentage of edge pixels in the patch, by obtaining information from the edge map of the source region (i.e., $\Phi\varepsilon$). Since edge map of the source region is computed only once for the entire image, computation time is saved, as compared to computing the isophote and the unit vector for each patch at each iterated step. In computing the data term, color variation in the patch is also used

$$\forall \mathrm{p} \in \delta\Omega, \mathrm{D(p)} = \max\left(1, \left(\sum_{\mathrm{q} \in (\Psi\mathrm{p} \cap \Phi\varepsilon)} c\right)\right) * \mathrm{var}(\Psi_\mathrm{p})/|\Psi_\mathrm{p}| \quad (2)$$

where the max function is used to assure a nonzero summation of pixel count in the edge map

$$\mathrm{var}(Pi,j) = \frac{\sqrt{\sum_{\forall i}\sum_{\forall j}(xij-\bar{x})^2}}{i\times j-1}$$
$$\bar{x} = \frac{\sum_{\forall i}\sum_{\forall j}x_{ij}}{i\times j}. \quad (3)$$

Note that the constant c represents the weight (set to 1 in our experiment) of a pixel. The priority map can now be computed based on the modified $\mathrm{D(p)}$

$$\mathrm{P(p)} = \mathrm{C(p)} * \mathrm{D(p)}.$$

After the confidence term $\mathrm{C(p)}$, the modified data term $\mathrm{D(p)}$, and the priority map $\mathrm{P(p)}$ are available, we further adopt the *patch template* as our searching strategy. More specifically, let $\Psi_{\mathrm{p}\wedge}$ be a patch of the highest priority

$$\Psi_{\mathrm{p}\wedge} = \mathrm{argmax}(\mathrm{P(p)}, \mathrm{p}\in\delta\Omega). \quad (4)$$

In general, we believe the larger the patch, the better chance to find a good match. Thus, it is necessary to consider useful surrounding pixels in addition to only using the useful pixels within a patch. Let $\Gamma_{\mathrm{p}\wedge}$ be a patch template of $\Psi_{\mathrm{p}\wedge}$:

$\Gamma_{\mathrm{p}\wedge} = \cup_{\pm k\Psi\mathrm{p}\wedge}(\Psi_{\mathrm{p}\wedge}\cap\Phi) \neq \phi$, where ϕ is an empty set.

Also, $\pm k\Psi_{\mathrm{p}\wedge}$ represents the patch $\Psi_{\mathrm{p}\wedge}$ plus its surrounding pixels outside $\Psi_{\mathrm{p}\wedge}$ and within a distance of k. We empirically set $k = 2$, resulting in the patch template size to be 7×7. However, the size of patch is equal to 3×3 as we have discussed. Note that the intersection with the source region Φ is computed to ensure that no "empty patch" is used. With the patch template defined, the search procedure follows. Let $\Gamma_{\mathrm{q}\wedge}$ be the best-matching patch template against all candidate patch templates in the source image

$$\Gamma_{\mathrm{q}\wedge} = \min_{\Gamma\mathrm{q}\in\Phi\mathrm{q}r} d(\Gamma_{\mathrm{p}\wedge},\Gamma_{\mathrm{q}}) \quad (5)$$

where $\Phi_{\mathrm{q}}^{\mathrm{r}}\subset\Phi$ represents a region of source image centered at q by a distance of r pixels (empirically set to 10 in our experiments). Note that, the search iteration looks for patch templates in pixel-by-pixel steps. We define $\mathrm{d}(\Gamma_{\mathrm{p}\wedge},\Gamma_{\mathrm{q}})$ as a distance function of two patch templates

$$d(\Gamma_{\mathrm{p}\wedge},\Gamma_{\mathrm{q}}) = \mathrm{SSD}_{CIE\,Lab}(\Gamma_{\mathrm{p}\wedge},\Gamma_{\mathrm{q}}) * \max\left(1,\left(\sum_{\mathrm{q}\in(\Gamma\mathrm{q}\cap\Phi\varepsilon)} c\right)\right). \quad (6)$$

The distance function considers two factors: the sum of squared distance (SSD) and the number of useful pixels in the patch template. We want to find a best patch $\Psi_{\mathrm{q}\wedge}$, where $\Psi_{\mathrm{q}\wedge}\subset\Gamma_{\mathrm{q}\wedge}$. Hence, the inpainting algorithm copy $\Psi_{\mathrm{q}\wedge}$ to $\Psi_{\mathrm{p}\wedge}, \forall\mathrm{p}\in\Psi_{\mathrm{p}\wedge}\cap\Omega$ (i.e., only copy pixels to the empty positions in patch $\Psi_{\mathrm{p}\wedge}$, without changing the area already inpainted). The constant, $c = 1$, represents the weight of a useful pixel. The last step in the inpainting algorithm is to update the confidence map $\mathrm{C(p)}$ in [1]

$$\mathrm{C(p)} = \mathrm{C(p^{\wedge})}, \forall\mathrm{p}\in\Psi_{\mathrm{p}\wedge}\cap\Omega. \quad (7)$$

Essentially, a copied pixel p has its confidence map updated by the confidence term computed from the patch $\Psi_{\mathrm{p}\wedge}$, that is, all copied pixels in a patch have the same confidence value, which reveals the percentage of useful pixels in $\Psi_{\mathrm{p}\wedge}$. We modify the strategy as

$$\mathrm{C(p)} = \mathrm{C(p^{\wedge})} * (d(\Psi_{\mathrm{p}\wedge},\Psi_{\mathrm{q}\wedge})/\alpha)\forall\mathrm{p}\in\Psi_{\mathrm{p}\wedge}\cap\Omega \quad (8)$$

where α is a normalization factor such that $(d(\Psi_{p\wedge},\Psi_{\mathrm{q}\wedge})/\alpha)$ is within (0, 1]. Thus, we incorporate the degree of similarity into the confidence map. Note that the same distance function for two patch templates can be applied to two patches $\Psi_{\mathrm{p}\wedge}$ and $\Psi_{\mathrm{q}\wedge}$. In conclusion, our proposed modified image inpainting algorithm can be summarized as follows (based on the modified $\mathrm{D(p)}$, $\mathrm{C(p)}$ and the use of $\Gamma\mathrm{q}$):

Inpaint$(\Omega) =$

Repeat until region Ω is empty.

1) Compute boundary $\delta\Omega$ and $\mathrm{P(p)}, \forall\mathrm{p}\in\delta\Omega$.
2) Propagate texture and structure information:
   a) Find $\Psi_{\mathrm{p}\wedge} = \max(\forall\Psi_{\mathrm{p}}, \mathrm{p}\in\delta\Omega)$, and define $\Gamma_{\mathrm{p}\wedge}$ for $\Psi_{\mathrm{p}\wedge}$.
   b) Find $\Gamma_{\mathrm{q}\wedge} = \min_{\Gamma\mathrm{q}\in\Phi\mathrm{q}r} d(\Gamma_{\mathrm{p}\wedge},\Gamma_{\mathrm{q}})$, and define $\Psi_{\mathrm{q}\wedge}$.
   c) Copy $\Psi_{\mathrm{q}\wedge}\cap\Omega$ to $\Psi_{\mathrm{p}\wedge}\cap\Omega$.
3) Update $\mathrm{C(p)} = \mathrm{C(p^{\wedge})} * (d(\Psi_{\mathrm{p}\wedge},\Psi_{\mathrm{q}\wedge})/\alpha), \forall\mathrm{p}\in\Psi_{\mathrm{p}\wedge}\cap\Omega$.

B. Ghost Shadows

The modified image inpainting algorithm is applied to all frames in a video clip. Objects to be removed are tracked by a tracking algorithm to be discussed in Section III-C. Each inpainted frame has a visually pleasant result when viewed individually. However, when we combine all inpainted frames to a video, ghost shadows are created. Ghost shadow causes a visually unpleasant effect, which is due to *temporal discontinuity*. To solve this problem, we have to study movements in video and camera motions. We analyze temporal continuity in Section III, followed by our solution to remove ghost shadows.

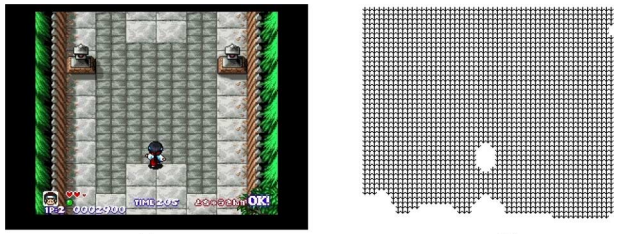
III. Video Inpainting Without Ghost Shadows

Motions in video can be created by an extrinsic moving object or the intrinsic movement of a video camera, such as panning, zooming, camera rotations (i.e., roll, pitch, and yaw) and other special effects[1] between scene transitions (e.g., fade out). With a constrained camera motion [13], the inpainted results are successful when the camera is parallel to the plane of frames. However, when dealing with zoom-in or zoom-out, patch matching and the computation of optical-flow mosaics [13] could be difficult. We will present solutions to deal with several types of camera motions, with ghost shadows either completely removed (for panning) or greatly reduced (for perspective video and tilting). Before we discuss the proposed

[1]Special camera effects such as fade in/fade out, flash, and others are not considered in this paper.

(a) (b)

Fig. 2. Inpainting of stationary video (courtesy of Benesse Taiwan Inc.) (a) Original video. (b) Inpainted background.



(a) (b)

Fig. 3. Video from video game and the motion map (courtesy of Nintendo). (a) Video from video game. (b) Motion map.

algorithms, we first explain various types of video inpainting under different motions.

A. Various Types of Video Inpainting

The problem of video inpainting can be divided into the following categories:

Type I: Stationary video with moving objects

Type II: Nonstationary video with still objects

Type III: Nonstationary video with moving objects (could be occluded), including all camera motions.

To deal with Type-I video inpainting, image inpainting algorithm can be extended. The focus is on how to inpaint the background properly. However, similar to other works, our image inpainting algorithm could not rebuild the semantic information lost in the background (e.g., structure of windows and door in Fig. 2). Thus, we found it makes more sense to take another inpainting strategy. Assume that the entire video has x frames. We use the middle frame (or any frame) as the base and identify the target area Ω. To select target patches on $\delta\Omega$, priority of selection is not considered. Instead, a target patch template $\Gamma_{q\wedge}$ with respect to each source patch Ψ_p is selected from frames $x/2\pm k$, where k is given from 1 to $x/2$. In the iteration

$$\Gamma_{q\wedge} = \min_{\Gamma q \in \Phi qr} d(\Gamma_p, \Gamma_q)$$

is used again and Γ_p is the source patch template of source patch Ψ_p. The target patch template Γ_q is selected from the relative position of Ψ_p among different frames, instead of searching on the same frame. The computation may result in a hole if the foreground moving object is almost fixed in the same position. Therefore, the modified exemplar-based image inpainting procedure is applied to the remaining hole. The inpainted background is shown in Fig. 2(b). In type-I video inpainting, there should be no ghost shadow, unless the inpainted patches at the same position among different frames are not the same (i.e., computed separately).

Type-II video inpainting has still objects with relative positions unchanged with respect to its background. However, the background is moving. To deal with this problem, the target area Ω is usually selected on the first frame. The modified exemplar-based image inpainting algorithm is applied to Ω, and the inpainted area is recorded. This recorded area is shifted and copied when the background is moving. Thus, a motion vector of the moving background (to be discussed in Section III-B) is computed and applied to the shifting. The inpainting procedure fills the shifting background area before other foreground areas are inpainted. If the still object is partially occluded, only the portion not occluded is duplicated. In case that the still object is occluding another object, the still object can be treated as the area called Ω_E (to be discussed later in Section III-D).

Type-III video inpainting is the most complicated problem, which has moving foreground and changing background due to different camera motions. Extending the exemplar-based image inpainting algorithm and applying the algorithm on each frame will produce ghost shadows. The phenomena are due to the discontinuous temporal characteristic of inpainted results among frames. Theoretically, an inpainted background or an inpainted moving object occluded by another object should maintain a temporal continuity. An example of temporal continuity can be illustrated by a *motion map* in Fig. 3. In Fig. 3(b), the arrows indicate that the background is moving down. The white area represents nonmoving objects. Assuming the avatar at the bottom center of the screen is removed, the hole should be inpainted by using the background. Therefore, the inpainted hole [at the location of the white hole in Fig. 3(b)] should have a consistent motion vector (moving down) with the background.

However, the computation of temporal continuity for inpainted objects will become much more involved when occluded and occluding objects are considered. Thus, segmentation of the motion map is necessary before we address our video inpainting algorithm.

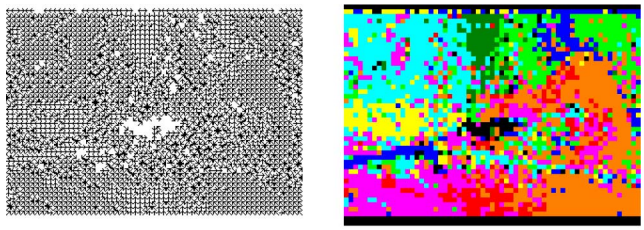
B. Segmentation of Motion Vector Map

Since objects to be removed may be occluded or can occlude other objects, it is necessary to separate a video frame into several regions with patches in each region moving in a similar direction. We adopt a motion estimation algorithm (i.e., 4SS) [14], which was originally proposed for block-based motion estimation in video coding, to compute a map of motion vectors. This map is further partitioned into several *motion segments*, which can represent moving objects at different layers and the nonstationary background. We further extend the motion estimation algorithm proposed in [14] by merging blocks with similar motion to separate a video into different layers. As soon as objects in different layers are tracked and removed, video inpainting algorithm can be applied from the bottom layer to the top layer step by step to fill-in holes due to object removal. Our motion segmentation algorithm is based on the following steps.

1) Use a modified 4SS algorithm [14] to compute Motion Map based on HSI color space and edge detection (not used in [14]).

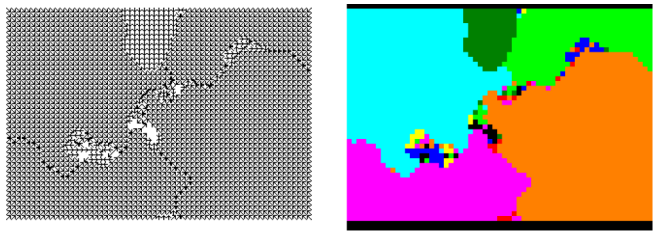
(a) (b)

Fig. 4. Original frame and its edge map. (a) Original frame. (b) Edge map.



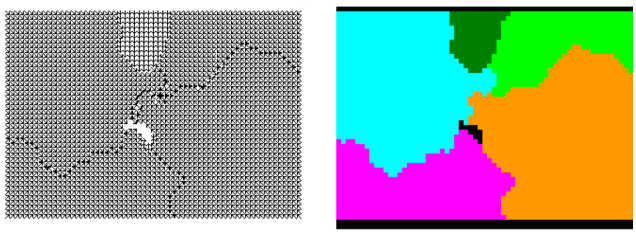
(a) (b)

Fig. 5. Motion map after Step 1). (a) Motion vectors. (b) Motion segments.



(a) (b)

Fig. 6. Motion map after Steps 2) and 3). (a) Motion vectors. (b) Motion segments.



(a) (b)

Fig. 7. Motion map after Step 3). (a) Motion vectors. (b) Motion segments.

2) Closing and Opening are used to merge segments (they have the same effect to remove isolated noisy regions as media filtering).
3) For those remaining blocks not processed in Step 2) above, we merged blocks based on similar average motion vectors.

In Step 1), edge detection algorithm is used. Fig. 4(a) shows an original frame with its corresponding edge map shown in Fig. 4(b). We use a 3×3 Sobel convolution kernel.

To estimate the similarities among blocks of size 5×5 (fixed in our algorithm), we use the sum of absolute differences (SAD) based on the HSI color components separately and the differences are accumulated as follows:

$$H_{\mathrm{SAD}_{(i,j)}}(dx,dy) = \sum_{a=0}^{p-1}\sum_{b=0}^{q-1} |H_n(i+a,j+b) - H_{n+1}(i+a+dx,j+b+dy)|$$

$$S_{\mathrm{SAD}_{(i,j)}}(dx,dy) = \sum_{a=0}^{p-1}\sum_{b=0}^{q-1} |S_n(i+a,j+b) - S_{n+1}(i+a+dx,j+b+dy)|$$

$$I_{\mathrm{SAD}_{(i,j)}}(dx,dy) = \sum_{a=0}^{p-1}\sum_{b=0}^{q-1} |I_n(i+a,j+b) - I_{n+1}(i+a+dx,j+b+dy)|$$

$$\mathrm{dis}(B_i,B_j) = H_{\mathrm{SAD}_{(i,j)}} + S_{\mathrm{SAD}_{(i,j)}} + I_{\mathrm{SAD}_{(i,j)}}.$$

Let B_i be the source block and let B_j be the target block. B_j is distributed according to the 4SS searching topology. The comparison of block similarity takes two steps:

1) Compare the edge maps of two blocks (i.e., B_i and B_j). If more than 70% of pixels in the edge maps match at the corresponding positions, B_i and B_j are called edge similar. Store all B_j that is edge similar to Bi in a block set BS.
2) If BS is not empty, then

Find B_j in BS for B_i, where $\mathrm{B}_j = \arg\min \mathrm{dis}(\mathrm{B}_i, \mathrm{B}_j)$

else

Find B_j in all 4SS searching blocks for Bi,

where $\mathrm{B}_j = \arg\min \mathrm{dis}(\mathrm{B}_i, \mathrm{B}_j)$).

After Step 1), a motion map is computed. Fig. 5(a) shows the motion vectors and Fig. 5(b) shows the vectors in different colors.

The motion segments produced after Step 1) does not necessary form sufficiently large contiguous segments. One reason for such a discontinuity is due to mismatches of contiguous blocks. With the additional refining operations in Step 2), we can thus eliminate some isolated regions, with the improved results shown in Fig. 6.

The motion map shown in Fig. 6 is further refined by merging similar motion blocks based on majority count in Step 3. After Step 3), the resulting motion map produced in Fig. 7 reveals the zoom-out effect (motion vectors point toward a center direction) of the ping-pong video sequence. The average motion vector of blocks grouped in the same segment is then used in our video inpainting algorithm.

C. Target Identification

After we compute the motion segments, the user has to select an object to be removed. This is the only step involving the user in our video inpainting algorithm. To precisely extract objects from video is a challenging problem [4], [9]. However, we use a new approach by using our motion segmentation and image inpainting algorithm. The new tracking algorithm, given as follows, is quite efficient and effective.

1) The user manually select the bounding box, Box_i, of the object to be removed (instead of selecting a shape).

Fig. 8. Using image inpainting technique for tracking. (a) Bounding box. (b) Inpainted box. (c) Differences shown in red. (d) Tracked object.

2) Find the motion segment which has the largest number of overlapping pixels with Box_i, and compute the average motion vector of the motion segment.
3) Use the average motion vector to identify another bounding box, Box_{i+1}, of the same size in the next frame.
4) Use image inpainting algorithm to inpaint each bounding box Box_x, to obtain the inpainted box Box_y, in the entire video sequence.
5) Compute the difference of pixels in $\mathrm{Box_x}$ and $\mathrm{Box_y}$, by using the Y component from YUV color transform of each pixel.
6) If the difference of Y value is above a threshold 15, the pixel belongs to a tracked target; otherwise, disregard the pixel.

An example of tracking is illustrated in Fig. 8. The bounding box [in 8(a)] is inpainted [in 8(b)] and the difference [in 8(c)] is obtained before the final object is tracked [in 8(d)].

After target object is identified and tracked through the entire video sequence and the motion segments are all computed, we integrate these techniques in the video inpainting algorithm.

D. Video Inpainting Based on Multilayered Segmentation

Motion segments can be used to represent the separation of moving objects from a nonstationary video background. If the object in a segment is removed, the inpainted areas should retain a similar motion with respect to its surrounding segments. If image inpainting is applied to each segment without considering the relationship of motions among surrounding segments, the temporal continuity may not sustain. Thus, to inpaint an object, the inpainted area in the previous frame needs to be incorporated. Since objects may be occluded by another object, the relationship among segments becomes more complicated. It is important to define characteristics of segments from both spatial and temporal aspects.

Definitions (see Fig. 9):

Ω^{t}	hole to be completed at frame t;
$\Omega'^{\mathrm{t+1}}$	hole of Ω^t completed at frame $t+1$ (in red);
Ω_E:	hole to be completed at an upper layer (in blue);
ω^{t}	surrounding block of $\Omega^{\mathrm{t}}, \Omega^{\mathrm{t}} \subseteq \omega^{\mathrm{t}}$;
$\omega^{\mathrm{t+1}}$	surrounding block of $\Omega'^{\mathrm{t+1}}, \Omega'^{\mathrm{t+1}} \subseteq \omega^{\mathrm{t+1}}$;
Δ_{xy}:	average motion vector of background;
δ_{xy}	average motion vector of foreground object (or Ω^t);
$\Omega^{\mathrm{t+1}}$	hole to be completed at frame $t+1$ (in green);

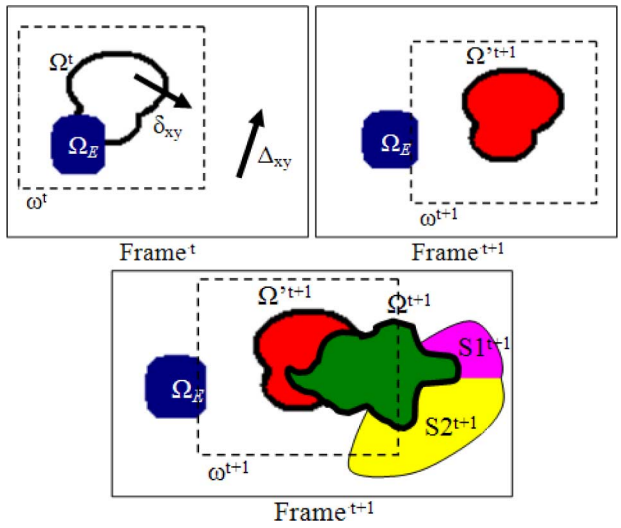


Fig. 9. Separation of holes and completed holes in consecutive video frames.

As illustrated in Fig. 9, each item given in the definition is self-explanatory. The surrounding block (shown in dotted lines) of the hole (shown in red) can be obtained by proportionally enlarging the bounding box of the hole to 225% (i.e., 1.5 times enlarged horizontally and vertically). The average motion vectors are derived from the motion vector maps computed in the previous section. The goal is to analyze how to complete the hole at frame $\mathrm{t}+1$ (i.e., $\Omega^{\mathrm{t+1}}$).

We define $\mathrm{T}(\Omega, \Delta)$ as a transformation which takes a hole Ω and an average motion vector Δ. The transformed object is located in the next frame. For instance, in a panning video, $\mathrm{T}(\Omega, \Delta)$ could be a translation such that $\Omega^{\mathrm{t+1}} = \mathrm{Translate}(\Omega \mathrm{t}, \delta \mathrm{xy})$, where $\Omega^{\mathrm{t+1}}$ is relocated in the next frame. For the background of a stationary video, $\mathrm{T}(\Omega, \Delta)$ is a unit function (i.e., Δ is set to zero).

In summary, we modify the image inpainting function discussed earlier for video inpainting which takes a hole and completes it. In general, $\Omega' = \mathrm{Inpaint}(\Omega)$, where Ω' represents the completed hole. The two functions can be expressed as

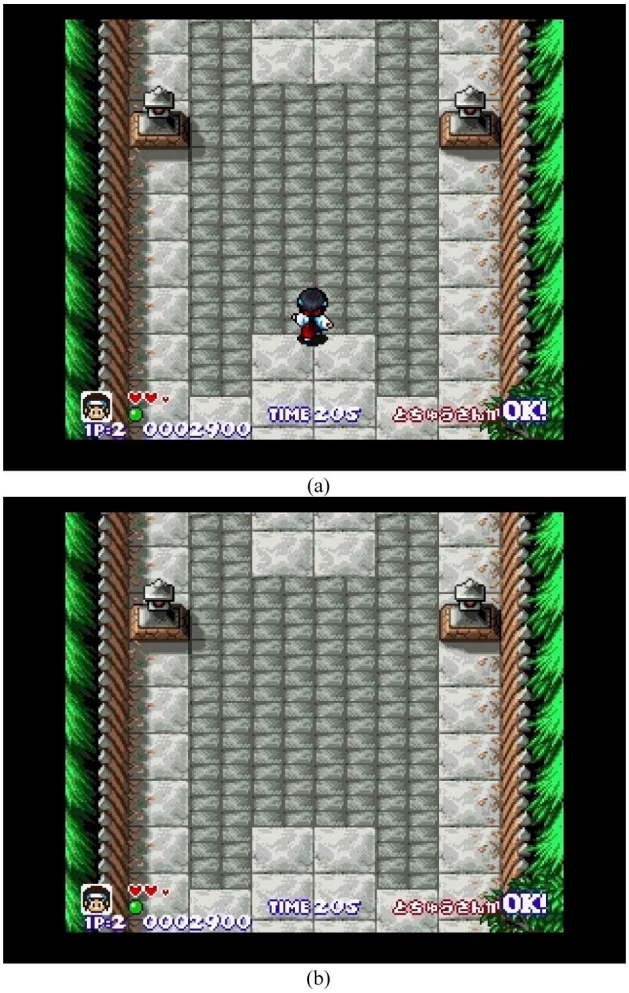
$$\begin{aligned}\mathrm{Translate}(\mathrm{Inpaint}(\Omega^{\mathrm{t}}), \Delta_{\mathrm{xy}} + \delta_{\mathrm{xy}}) &= \mathrm{Translate}(\Omega'^{\mathrm{t}}, \Delta_{\mathrm{xy}} + \delta_{\mathrm{xy}}) \\ &= \Omega'^{\mathrm{t+1}}\end{aligned}$$

where the hole Ω^{t} is completed at frame $t+1$ (i.e., $\Omega'^{\mathrm{t+1}}$ is produced). Note that the transformation can be applied to a surrounding block as well as follows:

$$\mathrm{Translate}(\omega^{\mathrm{t}}, \Delta_{\mathrm{xy}}) = \omega^{\mathrm{t+1}}$$

which means that the surrounding block ω^{t} is moved along with the background to $\omega^{\mathrm{t+1}}$ at frame $\mathrm{t}+1$.

The transformation $\mathrm{T}(\Omega, \Xi)$ can be generalized to cope with different camera motions, such as tilting and perspective video. For instance, $\mathrm{T}(\Omega, \Xi)$ could be a function transform $(\Omega, \Delta_{\mathrm{xy}}, \mathrm{S_{xy}}, \theta)$, where $\mathrm{S_{xy}}$ is a scaling factor, and θ is a rotation angle. Scaling factor is used in camera zooming and rotation is used in tilting. However, to precisely compute the motion vectors of a nonstationary video with perspective effects is difficult but doable to some extent (as shown in [15]).



(a)

(b)

Fig. 10. Image inpainting—video game. (a) Source video frame. (b) Inpainted video frame.

(a)

(b)

Fig. 11. Image inpainting—engineering building. (a) Source video frame. (b) Inpainted video frame.

The area to be inpainted at frame $t+1$ (i.e., Ω^{t+1} in green) contains several regions. Different strategies of inpainting procedure are used. The decomposed regions include

$$\begin{aligned}\Omega_A &= \Omega'^{t+1} \backslash \Omega^{t+1} \\ \Omega_B &= \Omega'^{t+1} \cap \Omega^{t+1} \\ \Omega_C &= (\Omega^{t+1} \backslash \Omega'^{t+1}) \cap \omega^{t+1} \\ \Omega_D &= \Omega^{t+1} \backslash \omega^{t+1}\end{aligned}$$

Region Ω_{A} at frame $t+1$ should be disregarded since the original surrounding area of Ω^{t+1} is kept. To inpaint region Ω_{B}, patches from Ω'^{t+1} are used to maintain temporal continuity. Region Ω^{C} is the new area to be inpainted thus patches from ω^{t+1} are used. In case that region Ω_{C} is surrounded by two or more motion segments, patches from the corresponding regions are used. To inpaint region Ω_{D}, patches outside ω^{t+1} and inside frame $t+1$ of the same motion segment (i.e., $\mathrm{S1}^{t+1}$ or $\mathrm{S2}^{t+1}$) are used. Note that, the empty hole to be completed at an upper layer (denoted as Ω_{E}) is left intact. Thus, we generalize the Inpaint(Ω) function as $\mathrm{Inpaint}(\Omega) = \mathrm{Inpaint} + (\Omega, \mathrm{A})$, where A is an area to search for patches. Therefore, the video inpainting algorithm takes the following options:

$$\begin{aligned}&\mathrm{Inpaint}(\Omega_A) = \omega^{t+1} \cap \Omega'^{t+1} \\ &\mathrm{Inpaint}(\Omega_B) = \mathrm{Inpaint} + (\Omega_B, \Omega'^{t+1}) \\ &\mathrm{Inpaint}(\Omega_C) = \mathrm{Inpaint} + (\Omega_C, \omega^{t+1}) \\ &\mathrm{Inpaint}(\Omega_D) = \mathrm{Inpaint} + (\Omega_D, (\mathrm{S1}^{t+1} \cup \mathrm{S2}^{t+1}) \backslash \omega^{t+1}).\end{aligned}$$

In summary, our proposed video inpainting algorithm is given as follows.

1) Users manually identify an object to be removed in a bounding box.
2) The object is tracked (see Section III-C).
3) Motion segments are computed (see Section III-B).
4) Apply Inpaint(Ω) to complete holes (see image inpainting in Section II and this section).

IV. Experimental Results

We use several types of video examples for experiments, including videos captured from video games and digital cameras. Different camera motions including panning, zooming (zoom-in

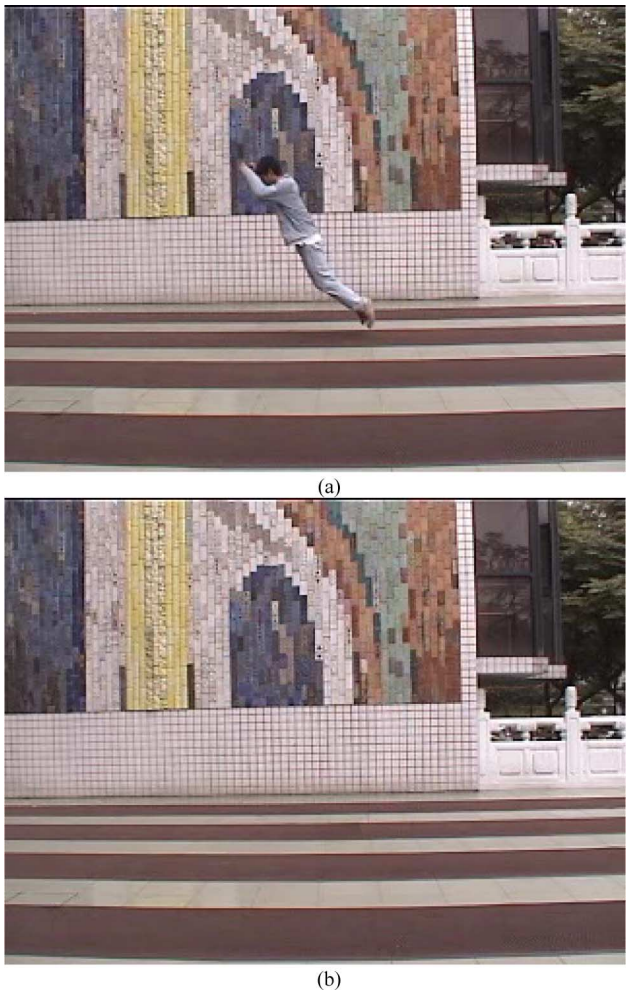


Fig. 12. Image inpainting—jump. (a) Source video frame. (b) Inpainted video frame.

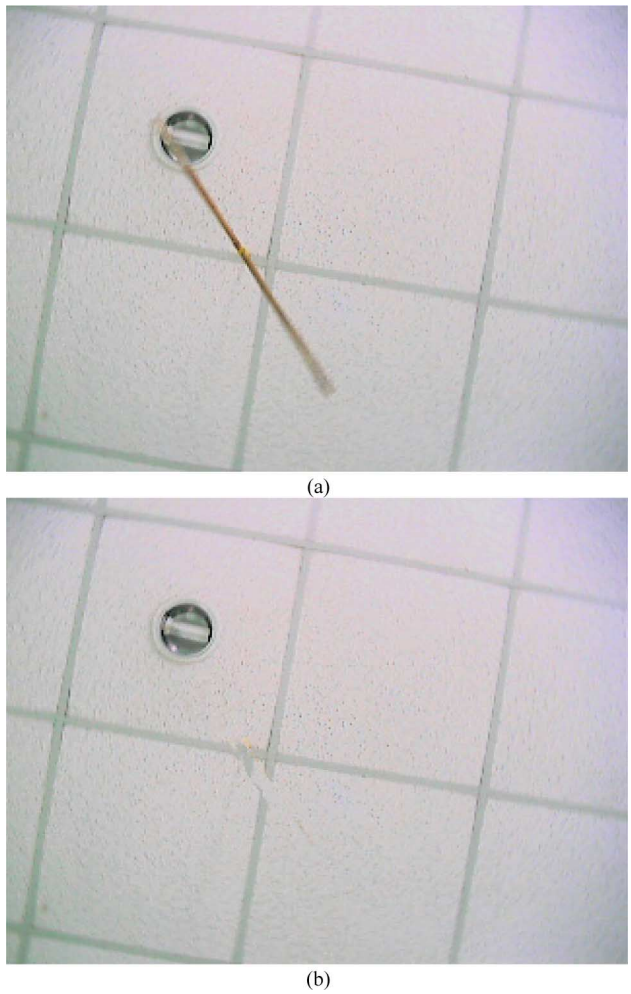


Fig. 13. Image inpainting—a failure case. (a) Source video frame. (b) Inpainted video frame.

and zoom-out), perspective, and tilting are used. We show some examples in Sections IV-A and IV-B. Analyses with comparisons to related works are given in Section IV-C. Limitations are discussed in Section IV-D.

A. Examples of Image Inpainting

Figs. 10–12 illustrate video frames to be inpainted frame-by-frame based on our proposed modified image inpainting technique discussed in Section II. All images have resolution of 320 by 240 pixels and 24 b of RGB colors. The patch size is 3×3 and the size of the patch template is 7×7. Fig. 10 is an image from a video game which has a constant background moving toward the upper direction. Fig. 11 is from a camera with panning motion. Fig. 12 is a stationary video. The three examples show reasonable results.

B. Examples of Video Inpainting

Examples of video inpainting are shown in Fig. 14. Each video sequence example includes a source video and an inpainted video, except the examples with several video sequences (Fig. 14(k)–(q), with the source and inpainted videos in removing multiple objects). These examples illustrate results of our video inpainting algorithm, under the following camera motions.

1) **Stationary video (a) and (b):** the inpainting strategy discussed in Section III-A produces an inpainted video sequence in (b). The marked red areas are completed step-by-step in 0.332 s (average time per frame based on a Pentium IV CPU, 3 GHz, with 1GB RAM). Only the time for inpainting is counted.
2) **Nonstationary video from video games (c) and (d):** the video game has its background moving up in a constant speed. Thus, motion segments are easily computed. The background objects are inpainted perfectly (see details in Fig. 10). The entire process includes time for tracking, motion segmentation, and inpainting. The total time used is 0.627 s on average per frame.
3) **Zoom-in video (e) and (f):** the video zooms-in slowly, with the walking man removed. The total time used is 0.562 s on average per frame.
4) **Zoom-out video (g) and (h):** the video zooms-out slowly, with another walking man removed. The total time used is 0.575 s on average per frame.

Fig. 14. Video inpainting under different camera motions.

5) **Panning video with scenery objects (i) and (j):** multiple objects are removed. The total time used is 0.855 s on average per frame.
6) **Panning video with objects in different layers (rows (k) and (m):** video (k) is the source, with multiple persons in different layers removed [in (l) and (m)]. This example shows a panning camera motion from right to left. The total time used is 1.184 s on average per frame.
7) **Panning video with objects in different layers (n)–(q)** This is another example of object removal in different layers. Video (n) is the source and the rest are results. The total time used is 1.835 s on average per frame.

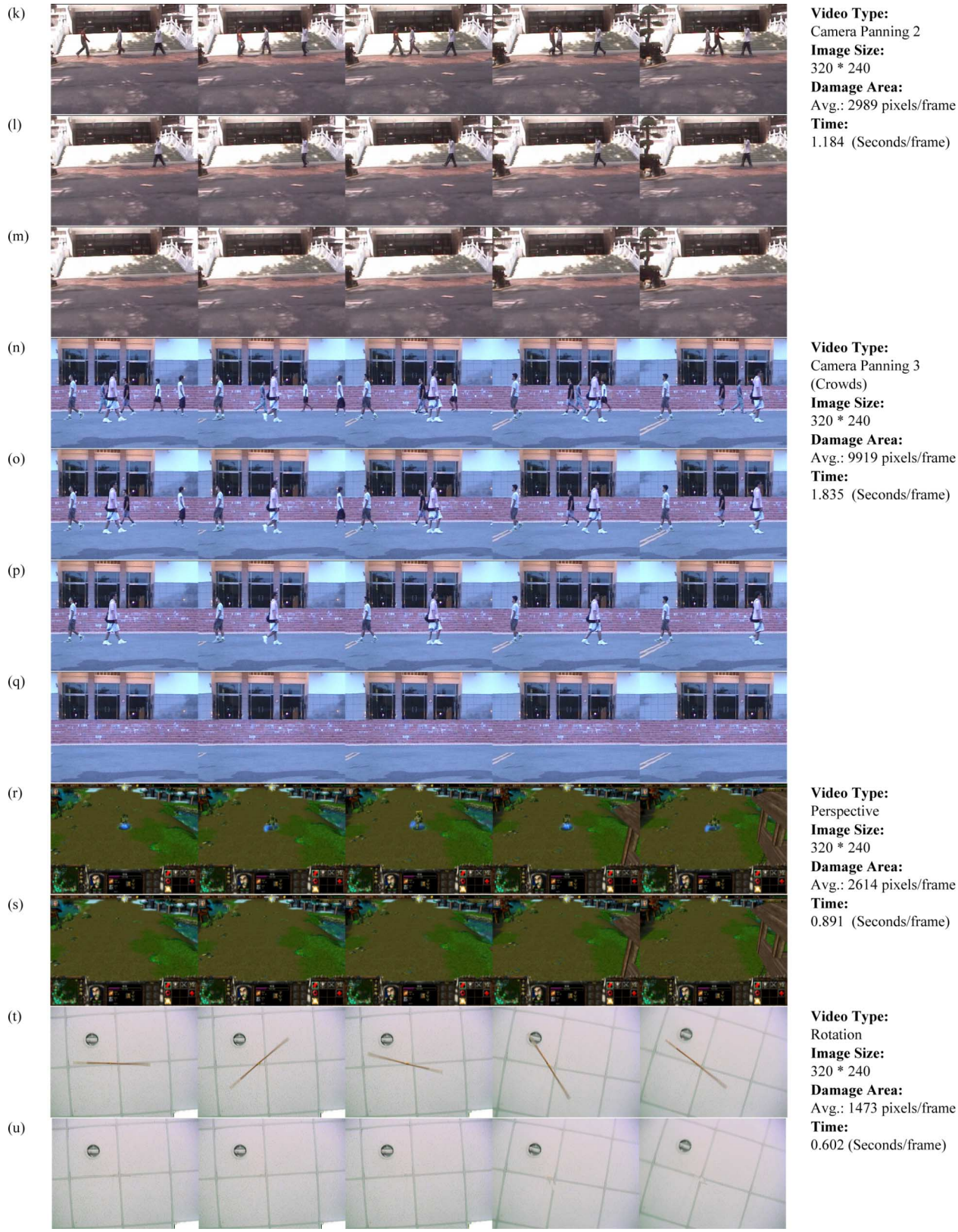


Fig. 14. (*Continued.*) Video inpainting under different camera motions.

8) **Video with perspective effect (r) and (s):** The video extracted from a video game has a pseudo perspective effect moving toward the top-center position on the screen. The avatar at the center of screen is removed. The total time used is 0.891 s on average per frame.
9) **Tilting video (t) and u):** The video is produced with a rotating camera. Also, the object to be inpainted is moving.

TABLE I
COMPARISON OF EXISTING METHODS WITH OUR NEWLY PROPOSED MECHANISM

Related Works	Human Interactions	Image Inpainting Methods	Type of videos	Camera Motions	Object Occluding	Ghost Shadows
Fragment Merging [8]	User provided mask as a shape	Texture synthesis and graph cut to merge patches	Taken from outdoor video camera, no fixed light source	Static Camera	Target Occludes other objects	Not removed (ghost shadows produced)
Occluding and Occluded Objects [12]	User provided mask as a shape	Exemplar-based image inpainting	Taken from outdoor video camera, no fixed light source	Static Camera	Occluded and occluding target	n/a (not discussed due to non-stationary video)
Under Constrained Camera Motion [13]	User provided mask as a shape	Exemplar-based image inpainting	Taken from outdoor video camera, no fixed light source	Non-static camera, with camera motion parallel to the plane of frames	Occluded and occluding target	Removed
Motion Layer Based [19]	User select a layer which contains a shape	Exemplar-based image inpainting (confidence values are not mentioned)	Taken from indoor and outdoor video camera	Static/non-static camera with panning	Target Occludes other objects	Not removed
Our Method	Bounding box give by the user, shape is tracked	Exemplar-based image inpainting (revised)	Video generated from computer, cartoons from TV, and video taken from outdoor video camera without fixed light source	Static/non-static camera with panning, zooming, perspective effect, and titling	Occluded and occluding target	Removed for non-static camera with panning and zooming and partially removed for perspective effect and titling

This is the most difficult example used in our experiments. The total time used is 0.602 s on average per frame.

Resolutions of videos are also 320 by 240 pixels with 24-b colors. It is difficult to show the effect of ghost shadows removed here.[2]

C. Comparison and Analysis

We qualitatively compare our approach with four related works [8], [12], [13], [19] in Table I. Criteria selected for comparison include the following:

- **Human interactions:** how much the user is involved in the video inpainting procedure;
- **Image inpainting methods:** the underlying image completion algorithm used (either based on texture synthesis or exemplar-based patch copy);
- **Type of videos:** indoor/outdoor video and computer generated video;
- **Camera motions:** stationary versus nonstationary video and how cameras are moved;
- **Object occluding:** whether the target object is occluding other objects or is occluded.
- **Ghost shadows:** whether ghost shadows are removed under nonstationary video with different camera motions.

In general, all inpainting algorithms listed in Table I require the user to manually select the object to be removed. However, depending on the tracking or layer separation algorithm, the amount of human interaction varies. Our method seems to be easier for a user to simply specify a bounding box of the target. Almost all video inpainting mechanisms use the exemplar-based image completion algorithm [1], except the work discussed in [8] where texture synthesis and graph cut are used. Most video inpainting projects use outdoor video. Shadows created by fixed light sources are not involved. Some projects focus on dealing with stationary video. Projects discussed in [13] and [19] use nonstationary videos under restricted camera motions. Only our proposed video inpainting algorithm considers a full set of camera motions. Moreover, almost all of our projects handle target that is partially occluded or is occluding another object.

Erasing an object from a stationary video should not produce ghost shadow in general, unless the same pixel in the static background is inpainted several times among different frames. However, to remove ghost shadows due to temporal discontinuity of nonstationary video inpainting, sophisticated block matching strategy and the separation of inpainting areas should be considered by taking into account of different camera motions.

D. Limitations

Our video inpainting algorithm includes several challenging procedures, such as object tracking, computing priority, and confidence values of patches, motion segmentation, and searching for best matched patches in different motion segments based on analysis of temporal continuity. The algorithm can deal with different camera motions. However, there are limitations:

1) **Excluding shadows in object tracking:** the novel tracking procedure based on motion map and image inpainting does not take advantage of shadows due to real world light sources.

[2]Interested readers should visit our website at http://member.mine.tku.edu.tw/www/T_CSVT/web/ to compare the video inpainting results with and without considering temporal continuity. The website contains several sets of demos in addition to the above examples.

2) **Patch selection on spatially continuous areas:** both the motion segmentation (and hence tracking) and inpainting procedures use a patch matching strategy. This match strategy should be treated differently from ordinary block matching algorithms in video coding. In video coding, a set of spatially continuous video blocks in a frame can be rebuilt from spatially discontinue blocks, if the predicted differences between two patches are below a threshold. The result still produces good video quality (with a price of higher bit-rate compression). However, to maintain spatial-temporal continuity in video in painting, it is critical to retrieve spatially continuous blocks/patches from other frames. Therefore, a much more sophisticated patch searching strategy which uses semantics of objects is worthy for further study.
3) **Patch matching on scaled and rotated objects:** even for patches selected from discontinuous areas, when dealing with tilting, perspective, and zooming motions of a camera, it is hard to match a block with another block which is rotated, skewed, and/or scaled. Thus, the examples discussed in Section IV-B can only remove part of ghost shadows in videos with perspective and tilting camera motions. In Fig. 13, we illustrate a failure case where an inpainted frame still has a problem due to block mismatching in a video with a rotated camera.
4) **Separation of video layers:** our motion segmentation algorithm only produces segments. But, the mechanism cannot tell the depth of segments/objects. Thus, in order to eliminate objects in each layer one by one, the user needs to decide which object to be removed first, followed by the second object. Object selection is subjective. Currently, our program only allows users to select one object to be removed each time. Therefore, if multiple objects are removed one by one, the user has to run the program several times [see examples in Fig. 14(k)–(q)].

V. Conclusion

Video inpainting under a full set of camera motions is a challenging task. The mechanisms proposed in this paper have several interesting contributions. First, motion map is segmented in order to precisely locate the movement of target inpainted area. The motion segmentation mechanism proposed in this paper is good enough for us to deal with the video inpainting problem. Second, we use edge map and patch template to estimate the underlying structure of blocks. By copying similar blocks, the resulting image completion procedure has a visually pleasant inpainting result. Finally, the proposed video inpainting procedure, based on the analysis of temporal continuities of video, is able to deal with different camera motions. Although video with a perspective effect and tilting video cannot be inpainted with all ghost shadows removed, our experiments shown in our demo website can effectively deal with panning and zooming of several types of video clips.

Our future works are somewhat discussed in the limitations (Section IV-D). Shadows caused by fixed light sources can be removed by other techniques. However, it is possible to enlarge the target to some extent such that the shadow is covered. And, hopefully, the image completion algorithm will recover the lost information. The challenge is on block matching, which should allow a block to match another block which is scaled, rotated, or skewed. However, the degree of scaling and rotation is hard to predict based on the speed of zooming and rotation of camera. In addition, how to select continuous blocks in a continuous area to inpaint a target region is another challenging issue.

References

[1] A. Criminisi, P. Perez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Trans. Image Process.*, vol. 13, no. 9, pp. 1200–1212, Sep. 2004.
[2] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal.Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.
[3] I. Drori, D. Cohen-Or, and H. Yeshurun, "Fragment-based image completion," *ACM Trans. Graphics (SIGGRAPH)*, vol. 22, pp. 303–312, 2003, San Diego, CA.
[4] K. Hariharakrishnan and D. Schonfeld, "Fast object tracking using adaptive block matching," *IEEE Trans. Multimedia*, vol. 7, no. 5, pp. 853–859, Oct. 2005.
[5] J. Hays and A. A. Efros, "Scene completion using millions of photographs," *ACM Trans. Graphics (SIGGRAPH 2007)*, vol. 26, no. 3, Aug. 2007.
[6] J. Jia, T. P. Wu, Y. W. Tai, and C. K. Tang, "Video repairing: Inference of foreground and background under severe occlusion," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, Jun.–Jul. 2004, pp. 364–371.
[7] J. Jia, Y. W. Tai, T. P. Wu, and C. K. Tang, "Video repairing under variable illumination using cyclic motions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 5, pp. 832–839, May 2006.
[8] Y. T. Jia, S. M. Hu, and R. R. Martin, "Video completion using tracking and fragment merging," in *Proc. Pacific Graphics 2005, Visual Computing*, Sep. 2005, vol. 21, pp. 601–610.
[9] C. Kim and J. N. Hwang, "Video object extraction for object-oriented applications," *J. VLSI Signal Process.—Syst. Signal, Image, Video Technol.*, vol. 29, no. 1/2, pp. 7–22, Aug. 2001.
[10] A. C. Kokaram and S. J. Godsill, "Joint detection, interpolation, motion and parameter estimation for image sequences with missing data," in *Proc. Int. Conf. Image Process.*, Oct. 1997, vol. 2, pp. 191–194.
[11] F. Nielsen and R. Nock, "ClickRemoval: Interactive pinpoint image object removal," in *Proc. 13th Annu. ACM Int. Conf. Multimedia*, 2005, pp. 315–318.
[12] K. A. Patwardhan, G. Sapiro, and M. Bertalmio, "Video inpainting of occluding and occluded objects," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2005, vol. 2, pp. 69–72.
[13] K. A. Patwardhan, G. Sapiro, and M. Bertalmío, "Video inpainting under constrained camera motion," *IEEE Trans. Image Process.*, vol. 16, no. 2, pp. 545–553, Feb. 2007.
[14] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 2, pp. 313–317, Jun. 1996.
[15] T. K. Shih, N. C. Tang, W. S. Yeh, T. J. Chen, and W. Lee, "Video inpainting and implant via diversified temporal continuations," in *Proc. ACM Multimedia Conf.*, Oct. 2006, pp. 133–136.
[16] T. K. Shih, N. C. Tang, and J. N. Hwang, "Ghost shadow removal in multi-layered video inpainting," in *Proc. IEEE Int. Conf. Multimedia & Expo. (ICME 2007)*, Jul. 2007, pp. 1471–1474.
[17] J. Sun, L. Yuan, J. Jia, and H. Y. Shum, "Image completion with structure propagation," in *Proc. ACM SIGGRAPH*, 2005, vol. 24, pp. 861–868.
[18] Y. Wexler, E. Shechtman, and M. Irani, "Space-time completion of video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 3, pp. 463–476, Mar. 2007.
[19] Y. Zhang, J. Xiao, and M. Shah, "Motion layer based object removal in videos," in *Proc. 7th IEEE Workshop Appl. Comput. Vision*, 2005, pp. 516–521.

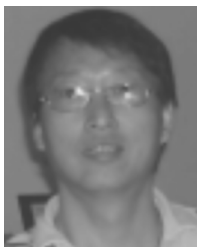
Timothy K. Shih (SM'08) received the B.S. degree from Tamkang University, Tamsui, Taiwan, in 1983, the M.S. degree from California State University in 1985, and the Ph.D. degree from Santa Clara University, Santa Clara, CA, in 1995.

He is a Professor with the Department of Computer Science, National Taipei University of Education, Taipei, Taiwan, and an Adjunct Professor with National Tsing Hua University, Taiwan. His current research interests include multimedia computing and distance learning. He has edited many books and published approximately 400 papers and book chapters, as well as participated in many international academic activities, including the organization of more than 50 international conferences and several special issues of international journals. He is the founder and co-editor-in-chief of the *International Journal of Distance Education Technologies* and an Associate Editor of *ACM Transactions on Internet Technology*.

Dr. Shih is a member of the Association for Computing Machinery. He is an Associate Editor of the IEEE TRANSACTIONS ON LEARNING TECHNOLOGIES and served as the Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA. He has received many research awards, including research awards from the National Science Council of Taiwan, the IIAS Research Award from Germany, the HSSS Award from Greece, the Brandon Hall Award from the United States, and several Best Paper Awards from international conferences. He was also the recipient of many funded research grants from both domestic and international agencies. He has been invited to give more than 25 keynote speeches and plenary talks in international conferences, tutorials in IEEE ICME 2001/2006 and ACM Multimedia 2002/2007, and talks at international conferences and overseas research organizations. He serves on the Educational Activities Board of the IEEE Computer Society.

Nick C. Tang received the B.S., M.S., and Ph.D. degrees from Tamkang University, Tamsui, Taiwan, in 2003, 2005, and 2008, respectively.

He is a Postdoctoral Fellow with the Institute of Information Science, Academica Sinica, Taiwan. His research interests include computer vision, video analysis, and their applications. He has published several important papers in video inpainting and video falsifying in the ACM International Conference on Multimedia and the Conference on Computer Vision and Pattern Recognition.



Jenq-Neng Hwang (F'01) received the B.S. and M.S. degrees from the National Taiwan University, Taipei, Taiwan, in 1981 and 1983, respectively, and the Ph.D. degree from the Signal and Image Processing Institute, University of Southern California, in 1988, all in electrical engineering.

After two years of obligatory military services, he enrolled as a Research Assistant in 1985 at the Signal and Image Processing Institute, Department of Electrical Engineering, University of Southern California. He was also a visiting student with Princeton University, Princeton, NJ, from 1987 to 1989. In the summer of 1989, he joined the Department of Electrical Engineering, University of Washington, Seattle, where he was promoted to Full Professor in 1999. He also served as the Associate Chair for Research & Development in the Department of Electrical Engineering from 2003 to 2005. He has published more than 200 journal and conference papers and book chapters in the areas of image/video signal processing, computational neural networks, multimedia system integration, and networking.

Dr. Hwang was the recipient of the 1995 IEEE Signal Processing Society's Annual Best Paper Award (with S.-R. Lay and A. Lippman) in the area of neural networks for signal processing. He has served as the Secretary of the Neural Systems and Applications Committee of the IEEE Circuits and Systems Society from 1989 to 1991 and was a member of Design and Implementation of Signal Processing Systems Technical Committee in IEEE Signal Processing Society. He is also a founding member of the Multimedia Signal Processing Technical Committee of IEEE Signal Processing Society. He served as the Chairman of the Neural Networks Signal Processing Technical Committee in the IEEE Signal Processing Society from 1996 to 1998, and was the Society's representative to the IEEE Neural Network Council from 1996 to 2000. He served as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING from 1992 to 1994 and an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS from 1992 to 2000. He is now an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and an Editor for the *Journal of Information Science and Engineering*. He is also on the editorial board of the *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*. He was the Conference Program Chair of 1994 IEEE Workshop on Neural Networks for Signal Processing held in Ermioni, Greece, September 1994. He was the General Co-Chair of the International Symposium on Artificial Neural Networks held in Hsinchu, Taiwan, in December 1995. He also chaired the tutorial committee for the IEEE International Conference on Neural Networks (ICNN'96) held in Washington DC, June 1996. He was the Program Co-Chair of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Seattle, 1998. He also served as the conference chairs for the IASTED Signal & Image Processing Conference and IASTED Internet Multimedia Systems and Applications in 2006. He was the Special Session Co-Chair of ISCAS 2008 and will be the Program Co-Chair of ISCAS 2009.