

Efficient object based video inpainting

M. Vijay Venkatesh, Sen-ching Samson Cheung and Jian Zhao¹

*Center for Visualization and Virtual Environments
Department of Electrical and Computer Engineering
University of Kentucky
Lexington, KY 40507, USA*

Abstract

Video inpainting is the process of repairing missing regions (holes) in videos. Most automatic techniques are computationally intensive and unable to repair large holes. To tackle these challenges, a computationally-efficient algorithm that separately inpaints foreground objects and background is proposed. Using Dynamic Programming, foreground objects are holistically inpainted with object templates that minimizes a sliding-window dissimilarity cost function. Static background are inpainted by adaptive background replacement and image inpainting.

Key words: Video inpainting, sliding window, template matching, dynamic programming, background replacement, object segmentation, object tracking

1. Introduction

Image inpainting, a technique to complete missing areas of a static image with information surrounding the hole has gained much popularity from the image processing community in recent years. Video inpainting started off as a natural extension of image inpainting algorithms. While still in early stages of development, it has garnered a great deal of attention due to its potential applications in video error concealment (Rane et al., 2003), multimedia editing, visualization, video stabilization (Matsushita et al., 2006) and new applications such as video modification for privacy protection (Cheung et al., 2006; Wickramasuriya et al., 2004; Zhang et al., 2005a). A straightforward extension of image inpainting algorithms to video inpainting is to treat the underlying video data as a set of distinct images and apply image inpainting algorithms to them individually. This approach to video inpainting proves inadequate as it fails to take advantage of the high temporal correlation that exists in video sequences. While recent video inpainting al-

gorithms have addressed this issue with relative success, many technical challenges still remain. The most notable ones are their algorithmic complexity and the limited hole size they can handle.

In this paper, we propose an efficient object-based video inpainting algorithm based on a modular approach – using simple background replacement and occasional image inpainting for static background inpainting and introducing a novel sliding-window based dissimilarity measure in a dynamic programming framework to inpaint moving objects. This technique can effectively handle large regions of occlusions, inpaint objects that are completely missing for several frames, and has minimal blurring and motion artifacts. Furthermore, our object-based approach is significantly faster than the patch-based scheme – with an unoptimized MATLAB implementation, our scheme takes minutes, rather than hours as required by many existing schemes, to obtain comparable inpainting results. Our proposed scheme also offers a unified framework to address video inpainting under both static and limited moving camera conditions.

The rest of this paper is organized as follows: In Section 2, we discuss existing video inpainting techniques and summarize the key contributions of our approach. We present an overview of our system in Section 3, and discuss the technical details of each major component in Sections 4, 6 and 5. We demonstrate the results of our method under dif-

Email address: {mvijay,cheung}@engr.uky.edu, Jian.Zhao@uky.edu (M. Vijay Venkatesh, Sen-ching Samson Cheung and Jian Zhao).

URL: <http://www.vis.uky.edu/mialab> (M. Vijay Venkatesh, Sen-ching Samson Cheung and Jian Zhao).

¹ The authors would like to acknowledge the support of Department of Justice under the grant 2004-IJ-CK-K055.

ferent operating conditions in Section 7 and conclude the paper in Section 8.

2. Related work

This section analyzes existing video inpainting algorithms, most of which fundamentally evolved from image inpainting approaches. Image inpainting techniques can be broadly classified into two categories: Partial Differential Equation (PDE) based approaches and Texture synthesis based approaches. One of the earliest efforts in extending the Partial Differential Equation (PDE) based image inpainting (Bertalmio et al., 2000) to video was performed by Bertalmio et.al (Bertalmio et al., 2001). The focus of this method is to fill in the hole spatially by extending the edges and filling the hole with smoothed color information by a diffusion process. It does not take into effect the temporally continuous nature of the video and treats the video as individual images. Due to extensive smoothing, it does not reproduce the texture information and suffers from severe blurring artifacts. Consequently this method is effective only in restoring small scratches or spots occurring in archival footage.

An important class of image inpainting algorithms are inspired from the work on texture synthesis by nonparametric sampling (Efros and Leung, 1999). A Markov Random Field (MRF) is used to model the local distribution of the pixel and new texture is synthesized by querying the existing texture using a patch-based direct sampling process. This forms the basis of the space-time video completion scheme in which the inpainting is formulated as a global optimization problem (Wexler et al., 2004, 2007). The hole of the video is filled by using spatio-temporal patches sampled from the existing video. This work advocates an exhaustive searching strategy throughout the entire spatio-temporal domain for finding appropriate patches to fill the hole, thus making it computationally intensive. Cheung et.al introduced a space-time patch model based on probabilistic learning with applications to inpainting (V.Cheung et al., 2005). These condensed models called *epitomes* are learned by compiling large number of space-time patches drawn from input videos. Inpainting is treated as a reconstruction problem and the epitomes in this case are learned from the observed pixels. Inferring the missing pixels from the condensed epitomes leads to severe oversmoothing of the reconstructed pixels.

The nonparametric sampling based approach for images is improved by incorporating a priority based sampling algorithm (Criminisi et al., 2004). This scheme also uses local image patches to fill the hole. Instead of following a fixed ordering along the boundary of the hole to fill the patches, the fill order was driven by a priority-based mechanism. Regions along the hole boundary that have local structures and a large spatial support are favored over other regions. Patwardhan et al. extend this idea to video inpainting under constrained camera motion by first sepa-

rating foreground objects from background layer by using optical flow (Patwardhan et al., 2005, 2007). Foreground pixels are completed first before the background regions are completed. Holes in the moving foreground regions are inpainted by a priority-based exemplar process. Damaged patches around the boundary of the hole are filled by selecting candidates from the foreground mosaic that minimizes a distance metric consisting of the pixel color values and the optical flow vectors. While being effective in completing regions with sparsely distributed structural cues, this method, like its image inpainting counterpart, is susceptible to growing incorrect patches due to spurious local variations. More importantly, this technique cannot handle the case when a significant portion of the object is missing, and it has difficulty in inpainting curved structures.

A video completion scheme based on motion layer estimation followed by motion compensation and texture completion has been proposed (Zhang et al., 2005b). After removing a particular motion layer, motion compensation is used to complete moving objects and non-parametric texture synthesis is used to complete the static background regions. The inpainted layers are then warped into every video frame to complete the holes. Video completion by motion field transfer – transfer of spatio-temporal patches of motion field instead of direct color sampling has been introduced recently (Shiratori et al., 2006). This technique is extremely sensitive to noise as they involve local motion estimates by a derivative-based process. It has difficulty inpainting large motion as their motion estimation techniques focus solely on measuring small local movement. In addition, as the scheme transfers only motion information, it suffers from blurring artifact due to the use of a re-sampling process to estimate color information. A video completion algorithm for perspective camera under constrained motion has been proposed recently (Shen et al., 2006). The foreground and background layers are separated and objects in foreground volume are rectified to compensate for perspective projection. The pixels in the foreground are completed by modeling it as a graph labeling problem as described in (Sun et al., 2005) and a dynamic programming is used to solve it.

Deviating from the patch-based methods discussed above, Jiaya et.al introduced an object-based inpainting system which utilizes a user-assisted segmentation to inpaint holes in foreground regions that exhibit cyclic motions (Jia et al., 2006). To complete the missing foreground regions they explicitly estimate the periodicity of the moving foreground object and align them with the partially damaged pixels in the hole boundary to complete missing regions. Temporal consistency is achieved by a novel (moving pixel) wrapping and regularization process using tensor voting. A similar technique that utilizes mean shift tracking to limit the search space and nonparametric texture synthesis coupled with graph cuts has been proposed (Jia et al., 2005). This method currently does not have a mechanism to handle moving cameras and also reports artifacts at the boundaries of the hole region.

2.1. Approach and Key Contributions

A visible trend emerging from recent works is the use of a two-stage process: segmentation of the video into different regions followed by separate completion of the respective regions. Existing algorithms differ by the method they use to complete hole regions in the moving foreground. It is often the most challenging and time consuming step due to the extensive search involved. Algorithms following this approach often have better performances because of two main reasons: First, segmenting into different layers not only provides better matching results, but also significantly reduces the search space for finding appropriate matches used for inpainting. Second, the background completion process can be made much faster by using an adaptive background replacement scheme or an efficient image inpainting technique.

While following this major trend of modular approach, our proposed technique differs from existing techniques by inpainting the entire objects instead of small spatio-temporal patches using available templates. The use of the entire object rather than individual patches provides a significant computational reduction which leads to a close-to-real-time implementation. The alignment and motion continuity are accomplished by using a sliding-window similarity metric in a dynamic programming setting. A preliminary version of our work has appeared in (Cheung et al., 2006). This earlier version lacked the ability to handle partial occlusions occurring when a moving object enters and exits a hole region. Partial occlusions were treated as complete occlusions by discarding the available information. As a result the earlier scheme had difficulties in establishing smooth transitions at the boundaries. We improve our previous approach by grouping consecutive, possibly incomplete object into a single entity under a sliding window mechanism. The process of grouping consecutive templates allows us to implicitly lock onto the global movement of the constituent objects and allows us to handle cases where the entire object is completely occluded. Succinctly put, our key contribution is a computationally-efficient object-based inpainting algorithm capable of inpainting partially- and completely-occluded objects and providing global motion consistency by using sliding-window registration and dynamic programming. We will demonstrate the effectiveness of our algorithm by inpainting different human subjects of varying pose, size and motion in video sequences captured through both static and moving cameras.

3. Overview of the Inpainting System

In this section we provide an overview of the entire system and describe the motivations behind the design. We begin by specifying the assumptions under which our system operates and note that these assumptions are in line with those made by the existing state-of-the-art systems. We assume that the scene consists of stationary background

and moving foreground regions. To inpaint the foreground objects that are occluded in the absence of model based or learning based methods, digital inpainting techniques require that the necessary information to fill in the hole be available in the video. For our focus on inpainting human motion in a short time window, this assumption is usually satisfied due to the fact that common human movements like walking or running are repetitive. We also limit that any camera movement present to be parallel to the image projection plane.

Figure 1 shows the schematic diagram of our proposed system. The input video is first fed into a background subtraction module. The output of this stage is a set of moving foreground blobs. In case of moving cameras, the camera motion is first estimated by a block matching scheme. The camera motion is then compensated by warping back all the video frames into the same coordinate system and estimate a background panorama. We then perform moving object segmentation that serves two purposes - it segments foreground blobs into constituent objects and links objects from frame to frame. The segmented objects, or object templates, are stored in a database. Target objects specified by external means are removed from each image forming the hole region to be inpainted.

The static portion of the hole can be filled by an adaptively updated background image if background information is available. Otherwise, image inpainting is performed based on the surrounding image statistics. The occluded moving foreground objects are inpainted by a two-stage process using the stored object templates. We first classify the frames in the hole as either partially-occluded or completely-occluded as shown in Figure 2. This is accomplished by comparing the size of the templates in the hole with the median size of templates in the database. The reason of handling these two cases separately is that the availability of partially-occluded objects allow direct spatial registration with the stored templates, while completely-occluded objects must rely on registration done before entering and after exiting the hole. The partial objects are first completed with the appropriate object templates selected by minimizing a window-based dissimilarity measure. Between a window of partially-occluded objects and a window of object templates from the database, we define the dissimilarity measure as the Sum of the Squared Differences (SSD) in their overlapping region plus a penalty based on the area of the non-overlapping region. The partially-occluded frame is then inpainted by alpha-matting with the object template that minimizes the window-based dissimilarity measure. Once the partially-occluded objects are inpainted, we are left with completely-occluded ones. They are inpainted by a Dynamic Programming based dissimilarity minimization process, but the matching cost is given by the dissimilarity between the available candidates in the database and the previously completed objects before and after the hole. The completed foreground and background regions are fused together using simple alpha matting. We now describe each component in details.

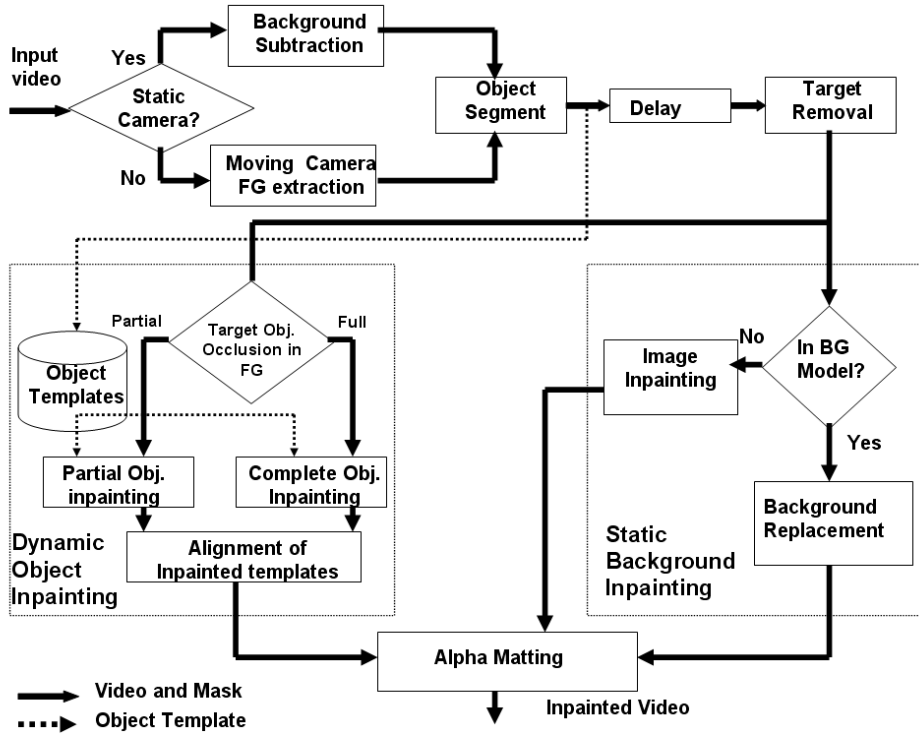


Fig. 1. Schematic diagram of the proposed object removal and inpainting system.

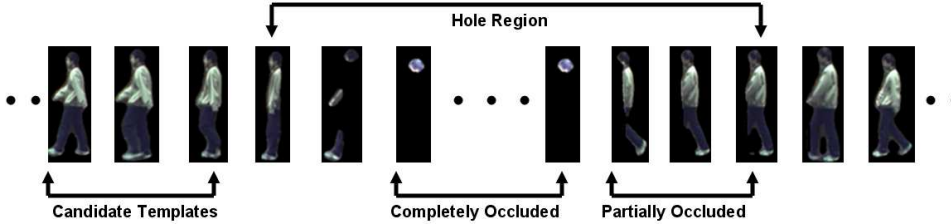


Fig. 2. Classification of the input frames into partially and completely occluded frames.

4. Foreground Extraction and Object Segmentation

To extract the foreground moving blobs and the shadows, we employ a background subtraction algorithm proposed in (Horprasert et al., 1999) which needs to be trained by a sequence with just a static background. In the training phase, each pixel in the background is modeled by a 4-tuple $(E_i, \sigma_i, a_i, b_i)$, where E_i is the expected color value, σ_i is the standard deviation of color value, a_i is the variation of the brightness distortion, and b_i is the variation of the chromaticity distortion. In the classification phase, every pixel in the incoming frame is classified either as a foreground pixel if the chromaticity exceeds a color threshold, or as a shadow pixel if they have similar chromaticity but lower brightness than those of the same pixel in the background image. The threshold values for the classification are computed in the training phase by setting the detection rate at 99.9%. Shadow separation is important as objects extracted in this step may be used for object interpolation in other locations where the old shadow is unlikely to be cor-

rect. We also apply a two-threshold hysteresis to prevent over-segmentation of foreground blobs.

In case of moving cameras, we extract the moving object in each frame by comparing it with the following frame using a block matching technique. Each frame is partitioned into non-overlapping blocks and the motion vector is estimated by searching in the neighborhood. Assuming that the camera moves parallel to the image projection plane, the median shift of all the blocks gives us a reliable estimate of the camera motion. The estimated camera motion is used to align all the available frames and the median value of each pixel position over time is chosen to create a background panorama. All the frames are then registered within this panorama by using the estimated motion vectors. We then segment the moving foreground from the background by modeling the distribution of the background pixels as a Gaussian Mixture and identifying the foreground pixels as those that significantly deviates from the background model.

The background subtraction process has a direct bearing on the quality of the captured Target object templates which are

later used to perform foreground inpainting. The classification threshold is conservatively chosen based on confidence level so that static background is rarely mistaken as moving objects. Such an approach is usually adequate for indoor environment. There are other more sophisticated background subtraction schemes such as (Elgammal et al., 2002) that can handle outdoor dynamic background. The main focus of our work, however, is to demonstrate the robust and efficient performance of our inpainting algorithm to complete missing foreground regions when compared with the existing techniques. As such, we have not incorporated the most advanced background subtraction scheme in our experiments.

Following this step, it is imperative to segment foreground blobs into constituent objects during occlusion. Due to the large variation of object pose and movement, moving object segmentation is a very challenging problem. However, we would like to emphasize that the performance of our inpainting algorithm does *not* depend heavily on this process. By checking the variation in the object bounding boxes, we can usually detect whether the resulting segmentation is reasonable. If we cannot accurately segment the objects during occlusion, we can consider them as complete occlusions and proceed to inpaint them using our complete object inpainting process. Furthermore, in creating the hole for inpainting, we use a more relaxed classification threshold in our background subtraction to remove all traces of the foreground objects. By using our background replacement scheme, we can recreate the background texture that is erroneously removed by the background subtraction process.

As such, we have implemented a relatively simple tracking-based segmentation algorithm that is computationally very efficient. The object with ID i observed at time t , denoted as O_t^i , is represented by a normalized spatial color histogram $h_t^i(\mathbf{p}, c)$ at each pixel position \mathbf{p} and color c . This histogram is adaptively updated throughout the lifetime of this object. Different quantities can be computed based on this histogram. For example, we can define the bounding region B_t^i to include all pixel location \mathbf{p} with $h_t^i(\mathbf{p}) = \sum_c h_t^i(\mathbf{p}, c) > \tau$ for some small $\tau > 0$. We can also define the location of the object centroid $\mathbf{c}_t^i = \sum_{\mathbf{p}} \mathbf{p} \cdot h_t^i(\mathbf{p})$. To track an object, we adaptively maintain its velocity vector \mathbf{v}_t^i using the following recursive update:

$$\mathbf{v}_t^i = \alpha \mathbf{v}_{t-1}^i + (1 - \alpha)(\mathbf{c}_t^i - \mathbf{c}_{t-1}^i) \quad (1)$$

where $\alpha > 0$ is an empirical parameter that dictates the rate of adaptation. With the velocity vector, we predict the spatial color histogram at time $t + 1$ as $\tilde{h}_{t+1}^i(\mathbf{p}, c) = h_t^i(\mathbf{p} - \mathbf{v}_t^i, c)$. To determine the object label of foreground pixel $I_{t+1}(\mathbf{p})$, we adopt the following decision rule:

$$I_{t+1}(\mathbf{p}) = c \in O_{t+1}^i \quad \text{if} \quad \frac{\tilde{h}_{t+1}^i(\mathbf{p}, c)}{\tilde{h}_{t+1}^j(\mathbf{p}, c)} > \frac{f_t^j}{f_t^i} \quad \forall j \quad (2)$$

f_t^i is the ‘‘depth score’’ for object O_t^i . For all our test sequences, we take advantage of the simple scene geometry

by assuming that object O_t^i is closer to the camera than object O_t^j if the baseline of the bounding region B_t^i is lower than that of B_t^j . The depth scores essentially act as priors for different objects in our moving object segmentation algorithm. The idea is to give bias towards objects that are closer to the camera as indicated by the baseline of their corresponding bounding boxes. The true prior would base on the part of the object (more likely to observe the occluded objects through the legs than the body of the occluding objects) and the amount of overlap between the occluding and occluded objects. As it is beyond the scope of this paper to precisely model all the different aspects of occlusion, we adopt the heuristic of assigning one as the depth score to the object farthest away from the camera, and doubling the object depth score as we move closer to the camera. After segmentation, the spatial color histogram of the *foreground* object is updated as follows:

$$h_{t+1}^i(\mathbf{p}, c) = \begin{cases} \alpha h_t^i(\mathbf{p} - \mathbf{v}_t^i, c) + (1 - \alpha) & I_{t+1}(\mathbf{p}) = c \in O_{t+1}^i \\ \alpha h_t^i(\mathbf{p} - \mathbf{v}_t^i, c) & \text{otherwise} \end{cases} \quad (3)$$

For all occluded objects, we ignore any partial observation and use the update rule $h_{t+1}^i(\mathbf{p}, c) = h_t^i(\mathbf{p} - \mathbf{v}, c)$

5. Static Background Inpainting

To perform background replacement, we cannot directly use the background image from Section 4, which represents the long-term average of the time-varying background. Instead, we need to use background pixels that are most compatible with the current frame to fill the hole. As a result, we maintain a separate, fast-adaptive background image based on Kalman filter (Karmann and Brandt, 1990) and use it for background replacement. We further apply an edge-sensitive filter similar to the de-blocking filter used in H.263 to smooth the boundaries of the region (H.263, 1998). In our implementation, we use a 5×5 filter with its strength determined by the dynamic range of the original pixels under the filter mask.

Occlusions cause by stationary objects cannot be completed by adaptive background replacement scheme as they lack background information in the occluding regions. For completing these regions, we use the exemplar-based image inpainting algorithm (Criminisi et al., 2004). This patch based method first computes a priority for each pixel in the damaged region by assuming a patch size, which is set to 9×9 in our implementation. The priority is determined by two factors, extent of the support of undamaged pixels and edge strength. Consequently narrow or edge regions are assigned with higher priority and are completed before attempting to fill smooth and homogenous regions. This method is capable of filling large areas by first propagating linear structure and then synthesizing the image textures. It performs well for a wide range of images with good tex-

ture and structure replication but has difficulties in handling curved structures.

6. Dynamic Object Inpainting

In this section, we describe the sliding-window based technique for foreground object inpainting. Stationary or moving target objects that occlude the object of interest in the foreground are removed from the video. Removal of these target objects creates a hole which could contain partially or completely-occluded object of interest. We first identify the partially-occluded frames and inpaint them before proceeding to the completely-occluded ones. The partially-occluded frames are not used to inpaint the completely-occluded ones. They are only used in the registration process and at the final stage as the alpha matting to composite with the inpainted objects to synthesize natural object movements.

To simplify the explanation of the process, we assume that there is only one moving object that needs to be inpainted. If there are multiple objects, one can apply the same technique to each object in the order of their depth. For our test sequences, this order is computed based on the baselines of the objects. We denote the object template in the database as O_{t_i} where t_i indicates the time of the frame when the template is captured, and i indicates the time of *the frame inside the hole where this template is a candidate for inpainting*. We use \tilde{O}_i to denote the actual object inside or near the boundary of the hole at frame index i .

6.1. Sliding window based dissimilarity measure

The basic idea is to use a combined set of successive object templates extracted at other time instances as candidates for interpolation. In our previous work, we used individual object candidates in a sliding-window framework to complete the hole (Cheung et al., 2006). This process is further improved by grouping successive object templates and treating them as a single entity under a sliding-window mechanism. Let w be the number of consecutive templates in a group. Our algorithm works for any odd numbered group size w . We first define an appropriate distance measure which computes the dissimilarity in shape, color, texture between the available candidates in the database and the partial or completely-occluded foreground objects present in the hole.

Let O_{t_i} be an object template in the database and \tilde{O}_i be the same foreground object near the boundary of or inside the hole. All object templates are modeled as functions that map the 2D coordinates \mathbf{p} to its gray-scale pixel value in the range of $[0,255]$. A dissimilarity measure $d(\tilde{O}_i, O_{t_i})$ must take into account of the proper alignment and the difference in shapes. Let \tilde{M}_i and M_{t_i} be the binary masks that define the shape of \tilde{O}_i and O_{t_i} respectively. We define the distance $d(\tilde{O}_i, O_{t_i})$ as follows:

$$d(\tilde{O}_i, O_{t_i}) \triangleq \min_{\mathbf{m}} E_O(\tilde{O}_i, O_{t_i}; \mathbf{m}) + E_N(\tilde{O}_i, O_{t_i}; \mathbf{m}) \quad (4)$$

where the score $E_O(\tilde{O}_i, O_{t_i}; \mathbf{m})$ between the overlapping areas and the score $E_N(\tilde{O}_i, O_{t_i}; \mathbf{m})$ between the non-overlapping areas are defined as follows:

$$E_O(\tilde{O}_i, O_{t_i}; \mathbf{m}) \triangleq \sum_{\mathbf{p} \in \tilde{M}_i} \left[\tilde{O}_i(\mathbf{p}) - O_{t_i}(\mathbf{p} + \mathbf{m}) \right]^2 \tilde{M}_i(\mathbf{p}) M_{t_i}(\mathbf{p} + \mathbf{m}) \quad (5)$$

and

$$E_N(\tilde{O}_i, O_{t_i}; \mathbf{m}) \triangleq 255^2 \sum_{\mathbf{p} \in \tilde{M}_i} \tilde{M}_i(\mathbf{p}) [1 - M_{t_i}(\mathbf{p} + \mathbf{m})] \quad (6)$$

$E_O(\tilde{O}_i, O_{t_i}; \mathbf{m})$ measures the SSD between the overlapping region of \tilde{O}_i and the candidate template O_{t_i} shifted by vector \mathbf{m} . The penalizing score $E_N(\tilde{O}_i, O_{t_i}; \mathbf{m})$ counts the number of pixels, weighted by 255^2 , within the object \tilde{O}_i not covered by O_{t_i} shifted by vector \mathbf{m} . This score acts as a balancing factor in helping to choose a candidate which not only has good similarity in overlapping areas but also is structurally similar to the partial template in the hole. For example, a candidate template similar in color and texture as well as structure (by the way of alignment with the partial template in the hole) would be chosen over a template which is similar in color and texture but does not align properly with the original partial template in the hole. The distance $d(\tilde{O}_i, O_{t_i})$ is the minimum combined score over all possible alignment vectors. We denote the optimal alignment vector as \mathbf{m}^* .

If partial templates are available in a window of consecutive frames, we can increase the robustness of the matching by simultaneously matching the entire window. Let G_{t_i} be a group of w consecutive object templates $O_{t_i}, \dots, O_{t_i+w-1}$ of the same foreground object from the candidate database. Let \tilde{G}_i be a group of partial objects $\tilde{O}_i, \dots, \tilde{O}_{i+w-1}$ of the same object in the hole. We now define a window-based distance $d_w(\tilde{G}_i, G_{t_i})$ as follows:

$$d_w(\tilde{G}_i, G_{t_i}) \triangleq \min_{\mathbf{m}} \sum_{f=0}^{w-1} E_O(\tilde{O}_{i+f}, O_{t_i+f}; \mathbf{m}) + E_N(\tilde{O}_{i+f}, O_{t_i+f}; \mathbf{m}) \quad (7)$$

Note that a single alignment vector \mathbf{m}^* is used to minimize the distance between all the objects in the window. As such, \mathbf{m}^* is robust enough to be directly used in the registration and inpainting of partial objects in the hole. As for complete occlusion, no partial templates are available to anchor the entire window and we have to resort to a dynamic programming framework to derive the proper registration throughout the entire duration of the hole. This process is explained in the next section.

6.2. Dynamic programming based optimization

Let h be the number of frames we need to interpolate and w be the size of the object template window. We define our time index i ranged from $i = 0$, corresponding to the time instance $w - 1$ frame before the hole, to $i = h + 2w - 3$ or $w - 1$ frames after the hole. Thus, the hole ranges from frame $i = w - 1$ to $i = h + w - 2$.

Partial templates are usually available at the beginning and towards the end of the hole region. If there are more than w consecutive partial templates, they can be directly used to identify the proper object templates in the database for inpainting. For example, assume that we have half a window of partial objects present starting from the first frame (frame index w) of the hole. As the partial object \tilde{O}_w is in the middle of the window \tilde{G}_0 , \tilde{O}_w can be inpainted by the middle frame of the template window $G_{t_0^*}$ that minimizes the window distance with \tilde{G}_0 as defined by Equation (7). Specifically, $G_{t_0^*}$ is defined as

$$G_{t_0^*} = \arg \min_{t_0} d_w(\tilde{G}_0, G_{t_0}) \quad (8)$$

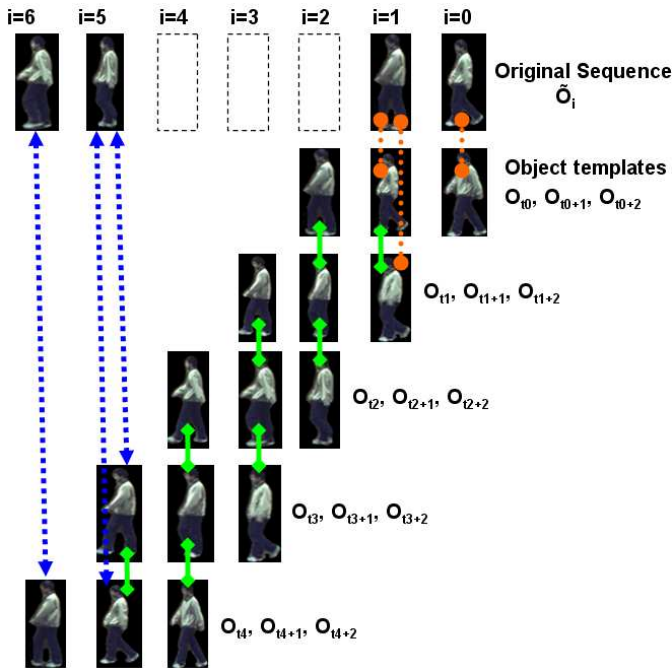


Fig. 3. Illustration of object interpolation in complete occlusion: the top row shows the original sequence with completely occluded templates in the hole. The subsequent rows show the best 3-frame sliding windows of object templates that minimize the recursive cost function defined in (9). Different color lines indicate different components in (9).

On the other hand, if there are completely-occluded objects in the hole, we cannot directly compute the window based dissimilarity measure. This is due to the lack of partial objects over the span of the window length. Furthermore, in this condition, the registration must be done using the available object templates before and after the hole.

Figure 3 shows an example of such a sequence in which $h = 3$ and $w = 3$. To define the matching cost function for the completely-occluded objects, we consider all possible w -frame windows of object candidates in our database such that when we slide them across the hole, the total distances between overlapping object templates from successive windows are minimized. This total distance is recursively computed as follow:

$$C_{t_i} \triangleq \sum_{j=0}^{w-1} d(\tilde{O}_{i+j}, O_{t_i+j}) \cdot 1_{\tilde{O}_{i+j}} + \min_{t_{i-1}} \left(C_{t_{i-1}} + 1_{\{i>0\}} \cdot \sum_{j=0}^{w-2} d(O_{t_i+j}, O_{t_{i-1}+j-1}) \right) \quad (9)$$

for $i = 0$ to $i = h + 2w - 3$, spanning the entire duration of the hole. We set the boundary condition $C_{t_{-1}} = 0$. The first term of Equation (9) is the cost between the candidate templates O_{t_i+j} and the partial objects \tilde{O}_{i+j} near the boundary of the hole, provided that they are present and hence the use of an indicator function $1_{\tilde{O}_{i+j}}$. These distance measurements are indicated by the red and blue arrows in Figure 3. The second term in (9) represents the cumulative cost across the complete occlusion where we maintain the motion continuity by computing the overlap between successive candidate windows. These measurements are indicated by the green arrows in Figure 3. By using a dynamic program to minimize the ending cost function $C_{t_{h+2w-3}}$, we obtain a series optimal template windows $G_{t_i^*} = \{O_{t_i^*}, \dots, O_{t_i^*+w-1}\}$ for $i = 0$ to $i = h + 2w - 3$. To inpaint frame i which is in the middle of the window $G_{t_{i-(w-1)/2}^*}$, the middle object template $O_{t_{i-(w-1)/2}^*+(w-1)/2}$ is used.

In the final step, we need to put the optimal object candidate at the correct location within each interpolated frame to achieve smooth object movement. To accomplish that, we utilize the alignment vectors from the object distances defined in (4) and (7). When partially-occluded objects are present in the entire window, solving the minimization in (8) provides the optimal template window $G_{t_i^*}$ with corresponding alignment vector $\mathbf{m}_{t_i^*}$. As we use the frame in the middle of the window for inpainting, we denote the centroid of that object template as $\mathbf{c}_{t_i^*}$. The centroid of the inpainted object $\hat{\mathbf{c}}_i$ is simply:

$$\hat{\mathbf{c}}_i = \mathbf{c}_{t_i^*} + \mathbf{m}_{t_i^*} \quad (10)$$

For the duration of the hole where complete occlusion occurs, we do not have a direct measurement of the actual alignment vector for each optimal inpainting template. The alignment vector $\mathbf{m}_{t_i^*}$ is measured with respect to the optimal inpainting template of the previous frame rather than the actual object in the scene. As such, to compute $\hat{\mathbf{c}}_i$, we need to sum all the alignment vectors starting from the first frame when complete occlusion occurs:

$$\hat{\mathbf{c}}_i = \mathbf{c}_{t_i^*} + \sum_{j=w-1}^i \mathbf{m}_{t_j^*} \quad (11)$$

As error can accumulate throughout the hole, we need to introduce an adjustment vector \mathbf{d} to each frame and then compute the actual centroid location $\hat{\mathbf{c}}_i$ by adding this adjustment vector:

$$\mathbf{d} = \frac{1}{h} (\mathbf{c}_{h+w-1} - \hat{\mathbf{c}}_{h+w-1}) \quad \text{and} \quad \hat{\mathbf{c}}_i = \hat{\mathbf{c}}_i + \mathbf{d} \quad (12)$$

7. Experimental Results and Complexity Analysis

In this section we present the results of our algorithm on six video sequences and compare the inpainting quality and complexity with other algorithms. Each of the sequences highlights the flexibility of our technique in addressing challenging scenarios including inpainting of partial or completely occluded objects, inpainting under moving camera, and inpainting of moving objects with complex motion, changing pose and perspective. Table 1 summarizes the details of the video sequences used in this section. To compare the effectiveness of our approach with existing schemes we apply our technique on the data provided by (Patwardhan et al., 2007; Wexler et al., 2007) in Section 7.3 and 7.5. Our results and the original video sequences are available in our project website at [http://vis.uky.edu/mialab/Video Inpainting.html](http://vis.uky.edu/mialab/Video%20Inpainting.html).

Table 1
Video sequences and their attributes (POF=partially occluded frames, FO=fully occluded frames)

Fig	Name	Resolution	length	POF	FOF
4	Three Person	320 × 240	75	15	0
5	One Board	320 × 240	100	14	14
6	Moving Camera	320 × 240	40	9	0
7	Spinning Person	320 × 240	140	23	0
9	Perspective change	320 × 240	64	12	0
10	Jumping Girl	300 × 100	240	25	21

7.1. Multiple occlusion handling using tracking

We first show our results on a sequence which has multiple occlusions. The “three person” sequence listed in Table 1 is a typical indoor surveillance sequence captured by a stationary camera. As shown in Figure 4, the object to be inpainted is being occluded by two other moving foreground objects at different time instances. Also notice that the object to be inpainted is occluded by a stationary object at the start of the sequence. As a result, there are incomplete templates of the object of interest in the database. We do not assume any a priori information about the presence of incomplete templates due to this occlusion. As discussed in Section 4, we first extract the moving foreground regions by background subtraction and employ tracking and object

segmentation to identify individual objects. In Figure 4, the first column shows four frames from the input sequence and the second column shows the results of object segmentation. The cyan and yellow colored regions represent the hole region. As the object of interest is only partially occluded, we use the partial object templates for registration and inpainting as described in Section 6. The window size used in the dissimilarity measurement is set to five. The last column in Figure 4 shows the inpainted results. As we can observe, the system is able to fill the holes in a seamless manner without using a priori knowledge about the incomplete templates.

7.2. Objects with complete occlusions

In this section we highlight the efficacy of our inpainting algorithm in handling complete occlusion over a number of frames. The top row of Figure 5 shows four input frames from the “One Board” sequence in which the moving foreground is occluded by a stationary board in the front. This sequence presents an extremely challenging task as it suffers near complete occlusion for 14 frames when the person is walking behind the board. Our system first identifies the hole subsequence that contains partial templates and employs the partially-occluded object inpainting scheme. The system then proceeds to complete the remainder of the holes using fully-occluded object inpainting. The window size is again set at five. The inpainted video frames are shown in the second row of Figure 5. The static regions in the background are inpainted by an image inpainting scheme (Criminisi et al., 2004). On close inspection, we can observe some artifacts on the ground below the inpainted person. This is caused by transferring part of the moving shadows due to imprecise segmentation.



Fig. 5. Inpainting of completely occluded objects: a) First row shows the unmodified input sequence with a stationary occlusion. Note the heavy occlusion and also the fact that very little foreground information is available in third frame. b) The second row shows the inpainted sequence. Notice that the completely occluded object has been effectively inpainted with smooth transitions in the entry and exit. The stationary background is inpainted using the image inpainting the technique in (Criminisi et al., 2004).

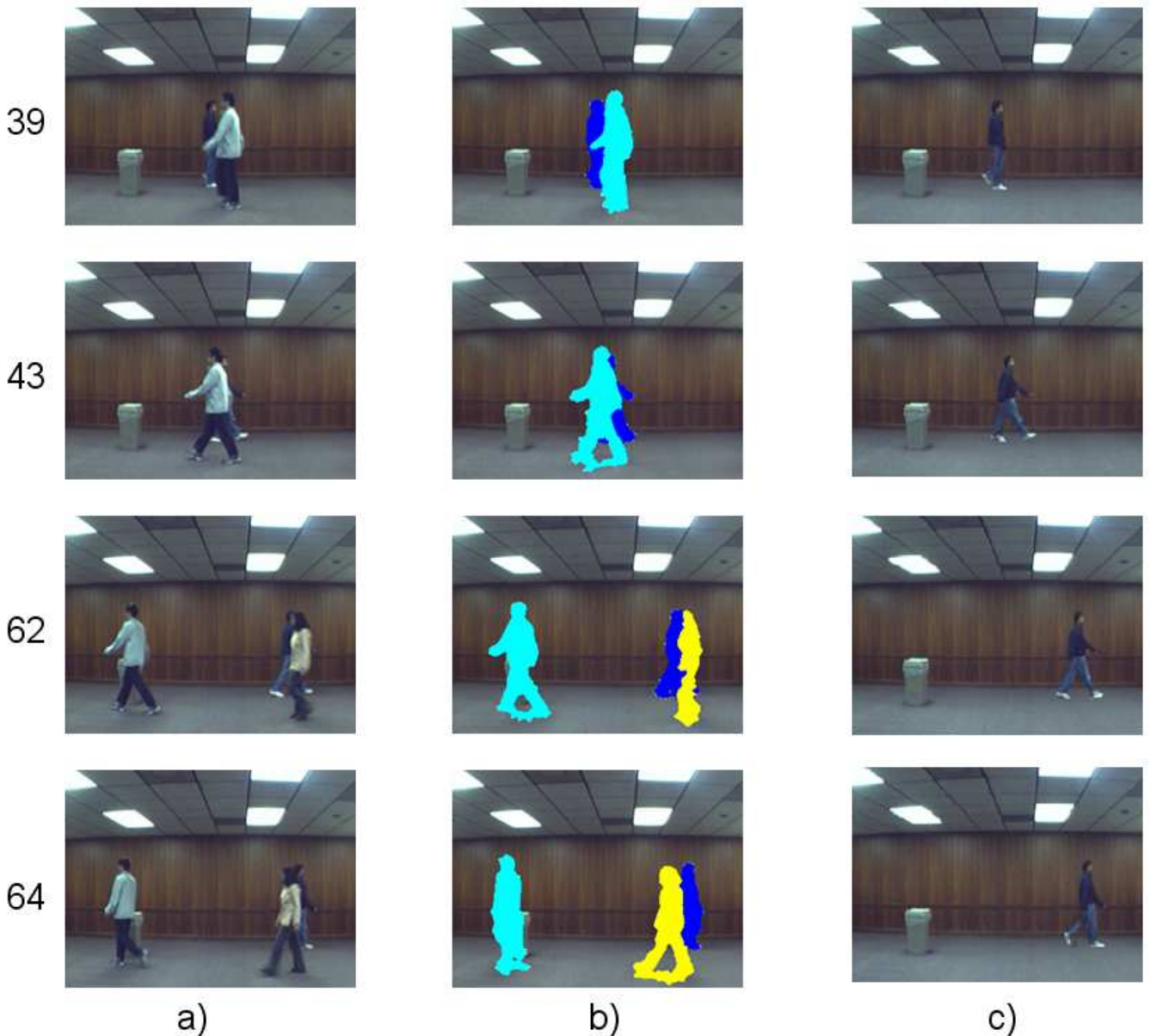


Fig. 4. Illustration of object interpolation: a) The first column shows the original input sequence along with the frame number. b) The second column shows results of the tracking and foreground segmentation stage in which objects are classified and segmented. c) The third column shows the inpainted result in which the moving foreground target objects are eliminated and the object of interest located at the back is inpainted. Also notice that the stationary object which is occluded by moving foreground in frame 62 and 64 has been inpainted back by adaptive background substitution.

7.3. Moving camera sequence

Figure 6 shows the video sequence “Moving Camera” taken from a moving camera originally used in (Patwardhan et al., 2007). The goal is to remove the foreground person and inpaint the background person during the brief occlusion. We use this sequence to demonstrate the capability of our system in addressing inpainting under restricted camera motion. We use the ground truth of the mask of the target object to be removed, which lies in front of the object of interest, which were provided by the authors of (Patwardhan et al., 2007). It should be noted that our appear-

ance based segmentation technique would not yield better results in this case as the interfering objects share similar appearances. The hole region extends over nine frames and has near complete occlusions when the objects cross each other. We employ our moving camera foreground extraction scheme explained in Section 4 to extract the moving foreground regions. The occluded regions are inpainted using our dynamic object inpainting scheme. The background regions are inpainted using a background replacement scheme based on the estimated background panorama. A closeup look at the inpainted frames using our method and that of (Patwardhan et al., 2007) are shown in the second and third columns of Figure 6 respectively. Our approach clearly pro-

vides better spatial details due to the use of actual texture rather than texture synthesis. A potential shortcoming of our object-based approach over patch-based approaches is that it might occasionally select a template with a wrong pose due to the similarity of the visible regions of the partial template and the lack of depth information. An example can be found in the top image of Figure 6b which shows the wrong leg moving forward as it becomes apparent in the subsequent images.

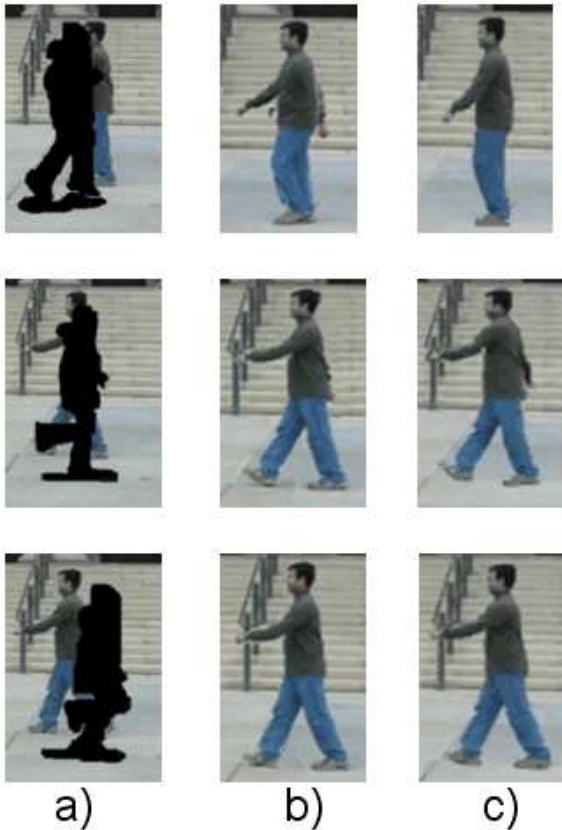


Fig. 6. Inpainting under Camera motion: a) First column shows the magnified version of the input sequence with the hole. b) The second column shows a close-up look of the final inpainting results using our algorithm. c) The third column shows the final inpainting result using the algorithm in. (Patwardhan et al., 2007).

7.4. Objects that change in pose

In this subsection, we illustrate the inpainting of a foreground object that undergoes a significant change in pose in the hole region. The “Spinning Person” sequence shows a person spinning from the right of the scene to the left. The first row of Figure 7 shows the closeup view of the partially occluded foreground object inside a synthetic hole. Figure 7b shows the results of our inpainting algorithm. It is clear that the change of pose of the head relative to the body inside the hole has been inpainted realistically. By using the partial templates inside the hole region in a sliding window based measure, we are able to lock onto the global change in pose occurring at a different time instance

to inpaint the hole in a holistic manner. Note that the assumption of repetitive movement is not strictly observed as the pose of the spinning person varies greatly throughout the sequence. Existing video inpainting sequences including patch based methods may encounter difficulty in inpainting such a sequence because of the greedy approach they pursue in filling the patches.



Fig. 7. Inpainting of foreground object with varying pose: a) The first row shows the magnified version of the input sequence with the hole. b) The second row shows a close-up look of the final inpainting results using our algorithm.

7.5. Inpainting under perspective change

The motivation behind this section is to illustrate that our video inpainting scheme can be extended to moving objects under perspective changes. Here we consider a case in which the person is walking at a constant velocity but the trajectory of the motion is not parallel to the camera plane. Due to this condition, the foreground object undergoes a perspective change. Figure 8a shows a montage of the foreground objects increasing in size as it moves closer towards the camera. We perform a normalization procedure to rectify the foreground templates so that the motion trajectory is parallel to the camera plane. Two points, at the top of the head and at the bottom of the feet, are extracted from the foreground object at different time instances. We obtain a series of vertical lines by connecting the head and feet points at different frames, and two horizontal lines by connecting all the head points as well as all the feet points. The vertical vanishing point is then computed as the intersection of all the vertical lines while the horizontal vanishing point is formed by intersecting the two horizontal lines. A metric rectification is then performed with the help of these vanishing points and the resulting rectified foreground volume is shown in Figure 8b (Hartley and Zisserman, 2004).

We then apply our inpainting algorithm on the rectified foreground volume. Figure 9a and 9b show the damaged and the inpainted foreground volume. The blocking artifact is more pronounced in this case due to the brightness variations of the objects templates in the database and the blurring effect is a main consequence of the scaling involved during the interpolation of candidates in the metric rectification process.

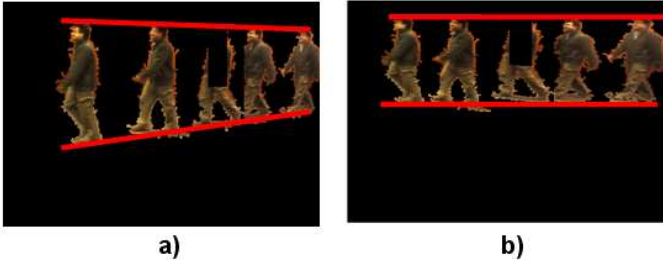


Fig. 8. Rectification of foreground objects: a) Foreground object under perspective change; b) Foreground object normalized by metric rectification.



Fig. 9. Inpainting foreground object under perspective change: a) A closeup of the foreground objects with the hole; b) Inpainted foreground objects using our algorithm.

7.6. Jumping girl sequence

This “Jumping Girl” sequence used in (Wexler et al., 2007) shows the moving object of interest exhibiting a repetitive jumping motion throughout the video. The object of interest is occluded by a stationary object which is cropped out using a user-supplied mask. Close-ups of the object of interest near the hole are shown in the first column of Figure 10. Using background subtraction algorithm, our system first extracts the moving foreground objects. Then, the frames with partial occlusion are identified and filled using the partial template inpainting. The remaining frames of the hole are filled by fully-occluded object inpainting scheme. Since we lack a background model in the hole regions, the background image is filled using an image inpainting scheme. The second column in Figure 10 shows our inpainting results. The corresponding results from (Patwardhan et al., 2007) and (Wexler et al., 2007) are shown in the third and fourth column of Figure 10 respectively. The results from (Patwardhan et al., 2007) are noisy and of poor resolution – this could be the limitation of their method in dealing with near complete occlusions as there is very little structural information present to do a reasonable inpainting. The foreground inpainting results

from (Wexler et al., 2007) are similar to ours but their results suffers from oversmoothing in the background regions.

7.7. Complexity Analysis

In this section, we analyze the asymptotic complexity of our algorithm and compare it with those of the methods described in (Patwardhan et al., 2007; Wexler et al., 2007). Let H be the number of frames in the hole, and within each frame, the hole area can be separated into N_f foreground pixels and N_b background pixels. Assume that the information needed to inpaint this hole can be found in a spatio-temporal volume of K frames, and each frame contains roughly $N \approx N_f + N_b$ useful pixels. We further assume that all the foreground pixels belong to the same object. Denote the window size and the search size of the alignment vectors in our algorithm as W and S respectively. Ignoring all the preprocessing tasks and focusing solely on the inpainting process, the following table compares the asymptotic complexities of three different variants of our scheme and those in (Patwardhan et al., 2007; Wexler et al., 2007): The first row of Table 2 shows the complexity of the

Table 2
Asymptotic Complexities of Three Schemes (PO=Partially-Occluded, FO=Fully-Occluded)

Schemes	Complexity
PO objects	$O(N_f HKWS + N_b H)$
FO objects with insufficient templates	$O(N_f HK^2 WS + N_b H)$
FO objects with sufficient templates	$O(N_f KWS + N_b H)$
Algorithm in (Patwardhan et al., 2007)	$O(N_f^2 HK + N_b H)$
Algorithm in (Wexler et al., 2007)	$O((N_f + N_b)^2 HK)$

partially-occluded object inpainting as described in Section 6. The complexity of the dissimilarity measurement in Equation (7) is $O(N_f WS)$. The dynamic programming is linear in terms of the number of frames H in the hole and thus the overall complexity for inpainting the foreground is $O(N_f HKWS)$. The complexity of background inpainting, regardless of whether background replacement or image inpainting is used, is simply $O(N_b H)$. The second row shows the complexity of the fully-occluded object inpainting based on Equation (9). Since there is no existing object template to compare with, this scheme requires each template to be compared with all other possible templates in order to deduce the optimal motion continuity. This results in a quadratic increase from K to K^2 , where K denotes the number of available templates. We notice that when the number of consecutive candidates available in the database is significantly greater than the number of the completely occluded templates in the hole, our algorithm always uses consecutive object templates to fill the hole. In other words, it is possible to have a significant speedup by heuristically testing only the strings of consecutive object templates slightly longer than the hole, and choosing the one that minimizes the dissimilarity at the boundaries.

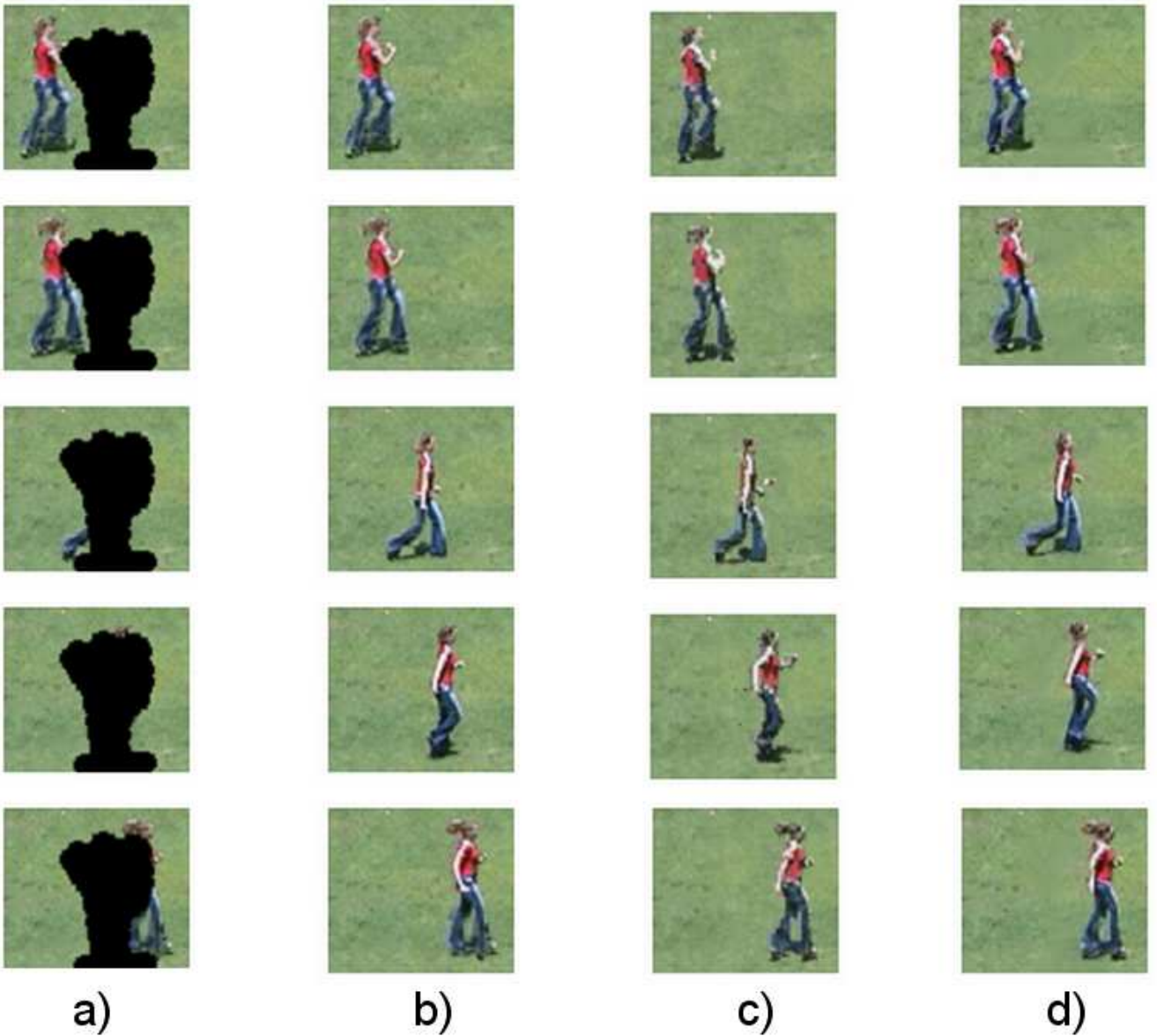


Fig. 10. Inpainting of complex motion: a) First column shows a magnified view of the input sequences adjoining the hole boundaries along with the corresponding frame number. b) Second column shows the inpainted result using our algorithm. Notice the visually consistent foreground inpainting and no blurring of the background. c) Third column shows the result of inpainting using the algorithm by Patwardhan et al. (Patwardhan et al., 2007). The inpainted foreground appears coarse and is not consistent throughout. d) Fourth column shows the result of inpainting using the algorithm by Wexler et al. (Wexler et al., 2007). The background appears to suffer from oversmoothing.

This scheme does not depend on the length of the hole and results in the complexity depicted in the third row of Table 2. The exemplar-based scheme in (Patwardhan et al., 2007) is similar to ours in terms of the separation of foreground and background inpainting. However, their scheme requires an extra factor of N_f as it needs to search the entire available foreground for filling each patch. The space-time completion scheme in (Wexler et al., 2007) does not separate foreground and background and as a result, has the highest computational complexity.

Our algorithm was implemented using MATLAB version 7.0 on a Xeon 2.1Ghz machine with 4 Gigabyte of memory. The actual running time to perform the inpainting for

the sequences used in this work are given in Table 3. Optimizing the code and porting it to C or C++ should significantly improve the performance. In addition, we do not exploit the preliminary alignment results from the tracking module and end up using a search size S close to the size of the entire foreground object. The comparison with the available templates can easily be parallelized which can further improve the performance.

8. Summary and Conclusions

We have presented a complete framework for an efficient video inpainting algorithm capable of addressing inpainting

Table 3

Actual Execution Time for the video sequences

Name	Inpainting $W = 3$	Inpainting $W = 5$	Pre-Process
Three Person	7.4 mins	10.2 mins	30 secs
One Board	3.5 mins	8.3 mins	40 secs
Moving Camera	2.6 mins	4.8 mins	3 mins
Spinning Person	12.6 mins	18.2 mins	35 secs
Perspective	3.6 mins	7.1 mins	35 secs
Jumping Girl	6.4 mins	11.4 mins	30 secs

under stationary camera and moving cameras. The stationary background region is filled by a combination of adaptive background replacement and image inpainting technique. Unlike other patch-based inpainting schemes, we inpaint the foreground object as a whole by formulating the problem as energy minimization. A new window-based dissimilarity measure is introduced to provide improved motion continuity within and at the boundaries of the hole. The final optimal candidates are selected by solving the minimization problem with dynamic programming. Our system offers several advantages over existing state-of-the-art methods in the following aspects: 1. Ability to handle large holes including cases where the occluded object is completely missing for several frames 2. Robust candidate selection from a set object templates provides significant speed up over existing patch-based schemes. 3. Object alignment process by the window-based scheme generates natural object movements inside the hole and provide smooth transitions at hole boundaries without resorting to any a prior motion model. 4. Our proposed scheme also provides a unified framework to address videos from both static and moving cameras and to handle moving objects with varying pose and changing perspective.

While we have made some progress in reducing the complexity of video inpainting, it remains a very challenging task – for instance, our scheme entirely ignores the presence of shadows. This results in unnatural appearance of objects. When the object to be inpainted exhibit complex, non-repetitive motion, the inpainting becomes significantly more challenging without assuming any a prior model. The variability in illumination conditions and arbitrary camera motion can also complicate the inpainting performance.

References

- Bertalmio, M., Bertozzi, A., Sapiro, G., 2001. Navier-stokes, fluid dynamics, and image and video inpainting. In: Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR). Vol. I. Hawaii, pp. 355–362.
- Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C., July 2000. Image inpainting. In: Proceedings of ACM Conf. Comp. Graphics (SIGGRAPH). New Orleans, USA, pp. 417–424.
- Cheung, S.-C. S., Zhao, J., Venkatesh, M. V., 2006. Efficient object-based video inpainting. In: Proceedings of International Conference on Image Processing (ICIP). IEEE, pp. 705–708.
- Criminisi, A., Perez, P., Toyama, K., September 2004. Region filling and object removal by exemplar-based inpainting. *IEEE Transactions on Image Processing* 13 (9), 1200–1212.
- Efros, A., Leung, T. K., September 1999. Texture synthesis by non-parametric sampling. In: IEEE International conference on Computer Vision (ICCV). Corfu, Greece, pp. 1033–1038.
- Elgammal, A., Duraiswami, R., Harwood, D., Davis, L. S., 2002. Background and foreground modeling using non-parametric kernel density estimation for visual surveillance. *Proceedings of the IEEE* 90, 1151–1163.
- H.263, 1998. Video Coding for Low Bitrate Communication. ITU-T Recommendation H.263 Version 2.
- Hartley, R., Zisserman, A., 2004. Multiple View Geometry in Computer Vision, 2nd Edition. Cambridge University Press.
- Horprasert, T., Harwood, D., Davis, L., 1999. A statistical approach for real-time robust background subtraction and shadow detection. In: Proc. IEEE Frame-Rate Applications Workshop.
- Jia, J., Tai, Y.-W., Wu, T.-P., Tang, C.-K., May 2006. Video repairing under variable illumination using cyclic motions. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. Vol. 28. pp. 832–839.
- Jia, Y. T., Hu, S. M., Martin, R. R., 2005. Video completion using tracking and fragment merging. In: *Proceedings of Pacific Graphics*. Vol. 21. pp. 601–610.
- Karmann, K. P., Brandt, A., 1990. Moving object recognition using an adaptive memory background. In: Cappellini, V. (Ed.), *Time-Varying Image Processing and Moving Object Recognition*, 2nd Edition. Elsevier Science Publishers, pp. 289–307.
- Matsushita, Y., Ofek, E., Ge, W., Tang, X., Shum, H.-Y., 2006. Full-frame video stabilization with motion inpainting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (7), 1150–1163.
- Patwardhan, K., Sapiro, G., Bertalmio, M., 2005. Video inpainting of occluded and occluding objects. In: *Proceedings of IEEE International Conference on Image Processing (ICIP)*. Vol. 2. pp. 69–72.
- Patwardhan, K. A., Sapiro, G., Bertalmio, M., Feb 2007. Video inpainting under constrained camera motion. *IEEE Transactions On Image Processing* 16 (2), 545–553.
- Rane, S., Sapiro, G., Bertalmio, M., 2003. Structure and texture filling-in of missing image blocks in wireless transmission and compression applications. In: *IEEE Transactions on Image Processing*. Vol. 12. pp. 296–303.
- Shen, Y., Lu, F., Cao, X., Foroosh, H., 2006. Video completion for perspective camera under constrained motion. In: *Proceeding of the International Conference on Pattern Recognition (ICPR)*. Vol. 3. pp. 63–66.
- Shiratori, T., Matsushita, Y., Kang, S. B., Tang, X., June 2006. Video completion by motion field transfer. In: Pro-

- ceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Vol. 1. pp. 411–418.
- Sun, J., Yuan, L., Jia, J., Shum, H.-Y., July 2005. Image completion with structure propagation. In: Proceedings of ACM Conf. Comp. Graphics (SIGGRAPH). Vol. 24. pp. 861–868.
- V.Cheung, Frey, B. J., Jojic, N., 2005. Video epitomes. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). Vol. 1. pp. 42–49.
- Wexler, Y., Shechtman, E., Irani, M., June 2004. Space-time video completion. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). Vol. 1. pp. 120–127.
- Wexler, Y., Shechtman, E., Irani, M., 2007. Space-time completion of video. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29 (3), 463–476.
- Wickramasuriya, J., Datt, M., Mehrotra, S., Venkatasubramanian, N., October 2004. Privacy protecting data collection in media spaces. In: *ACM International Conference on Multimedia*. New York, NY, pp. 48–55.
- Zhang, W., Cheung, S.-C., Chen, M., September 2005a. Hiding privacy information in video surveillance system. In: Proceedings of the 12th IEEE International Conference on Image Processing (ICIP). Vol. 3. Genova, Italy, pp. 868–871.
- Zhang, Y., Xiao, J., Shah, M., 2005b. Motion layer based object removal in videos. In: Proceedings of the Seventh IEEE Workshops on Application of Computer Vision. Vol. 1. pp. 516–521.