Video Repairing: Inference of Foreground and Background Under Severe Occlusion*

Jiaya Jia Tai-Pang Wu Yu-Wing Tai Chi-Keung Tang

Vision and Graphics Group, Computer Science Department
Hong Kong University of Science and Technology
{leojia, yuwing, pang, cktang}@cs.ust.hk

Abstract

In this paper, we propose a new method, video repairing, *to robustly infer missing static background and moving foreground due to severe damage or occlusion from a video. To recover background pixels, we extend the image repairing method, where layer segmentation and homography blending are used to preserve temporal coherence and avoid flickering. By exploiting the constraint imposed by periodic motion and a subclass of camera and object motions, we adopt a two-phase approach to repair moving foreground pixels: In the sampling phase, motion data are sampled and regularized by 3D tensor voting to maintain temporal coherence and motion periodicity. In the alignment phase, missing moving foreground pixels are inferred by spatial and temporal alignment of the sampled motion data at multiple scales. We experimented our system with some difficult examples, where the camera can be stationary or in motion.*

1 Introduction

Damage to a video may be caused by natural deterioration of old celluloid films, or deliberate object removal by interactive segmentation tool (such as [13, 14]). These damages may create large image holes in many consecutive frames, exposing a previously occluded background and moving foreground objects. Currently, restoring these videos is still a challenging problem, in both situations that missing pixels can be interpolated from some frames and are totally unknown. Take Fig. 1 as an example, which is extracted from a video. We want to remove the large sculpture which occludes both the background scene and the moving person. Since the camera is static, the missing background is absent in all frames while the moving person can be seen from some of them. Many issues, such as temporal consistency and faithful pixels inference, arise. They should be addressed properly in order to achieve a visually plausible restoration.

In this paper, we propose a synthesis approach to infer missing background and foreground to achieve a visually acceptable video restoration, where the camera can be stationary or moving. By exploiting a large subclass of camera and object motions, we show that this difficult problem can be solved to some extent, if certain conditions to be described are satisfied.

*This work is supported by the Research Grant Council of Hong Kong Special Administration Region, China: HKUST6171/03E.

Our overall approach consists of background layer inference and moving foreground repairing. To faithfully generate missing background pixels, we maintain temporal coherence based on the layer representation. Hence, large holes in video can be filled by naturally extending the still image repairing method [9]. As for the missing foreground pixels, we use a sampling and alignment strategy, by assuming periodic motion. We first sample moving elements or *movels* (section 3.2) from the input video. Then, movel that contains missing motion pixels can be regarded as a damaged movel, which can be repaired by aligning sample movels. Finally, the repaired foreground and background are composited together to restore a damaged frame. Experiments on difficult examples show that our restored frames are nearly seamless when they are viewed as individual images, or played together as a video sequence in integration aspect (we invite reviewer to view our submitted video accompanying this paper). Little spatial or temporal artifacts are observed, which are mainly caused by the abrupt change of light and shadow.

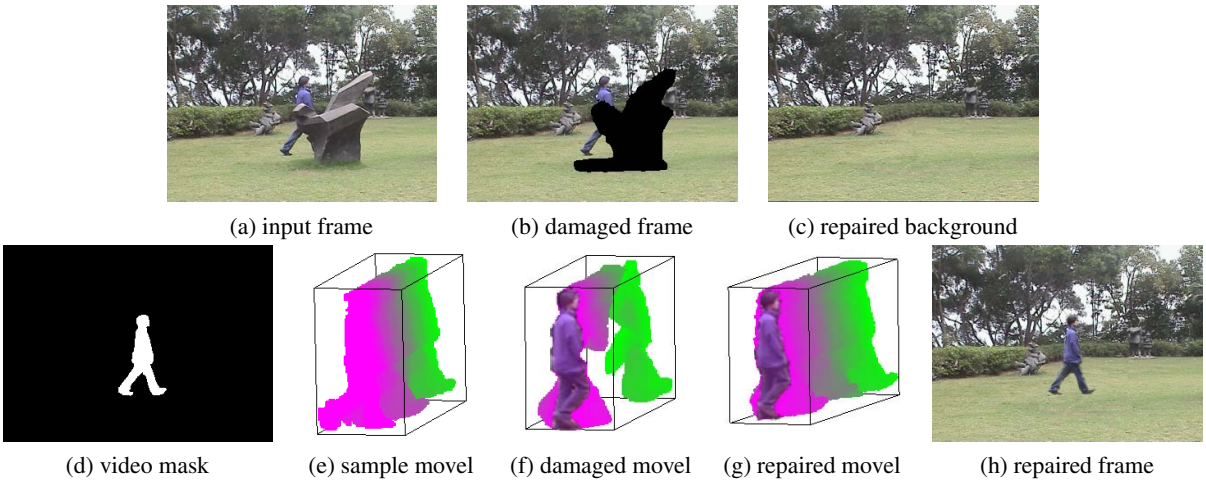
The organization of this paper is as follows: the static background repairing and moving foreground repairing are described in sections 2 and 3 respectively, where related works are also reviewed. In section 4, we summarize the video repairing system. Results are presented and discussed in section 5, followed by concluding remarks in section 6.

2 Static background repairing

In this section, we first motivate our extension to image repairing method [9] to infer a static background in a moving video, by describing a few plausible approaches. In many cases, it is found that layer representation is the key to maintain temporal consistency without distortion, and homography blending should be used to avoid flickering.

2.1 Motivation

Suppose that we are capable of repairing large holes in a single damaged frame [9]. The straightforward, frame-by-frame repairing approach for video will fail because there is no temporal consistency among the repaired frames, unless the holes are sufficiently small. Take FLOWER GARDEN as an example, where the foreground tree has been removed. Although we cannot observe obvious visual artifact when each individually restored frame is viewed independently, we can immediately notice a phantom tree truck drifting horizontally, when the frames are consecutively played as a sequence. Such



(a) input frame (b) damaged frame (c) repaired background

(d) video mask (e) sample movel (f) damaged movel (g) repaired movel (h) repaired frame

Figure 1: Summary of video repairing, using SCULPTURE as a running example. (a) is the input frame. (b) is the frame damaged by removing the sculpture. (c) is the result of static background repairing. (h) is the result of moving foreground repairing, where (d)–(g) are intermediate results to be explained in the paper. The camera is stationary.

alias is due to the sole use of spatial information for frame restoration.

An alternative approach is to construct a video mosaic. A single layer can be used [19, 8] to fill the missing background, and then warp back the restored frames from the video mosaic. This approach works well if the scene is distant or contains very few occlusions (hence the entire imaging can be well approximated by a single homographic or planar perspective transform). However, for scenes with significant depth discontinuities, a single video mosaic introduces unacceptable distortion.

To simultaneously restore the missing background for a static/moving camera without visible distortion, and to enforce the necessary temporal consistency if the camera motion is smooth, we adopt the layered mosaics approach. Inspired by geometric proxies used in image-based rendering systems for rendering anti-aliased novel views when the scene is cluttered and complex [20, 6, 3, 10, 1, 17, 21], we propose to segment the scene into layers. As our input is a video, the inherent challenge is an efficient method to perform multi-image segmentation. Therefore, in this paper, we adopt layer propagation: a layer in a video frame can be regarded as a 3D image patch with similar features. By segmenting the background into similar layers with depth ordering in each frame, each layer has respective optical flow for achieving the temporal consistence in a complex scene.

In the rest of this section, we describe each step of background repairing method in more detail.

2.2 Input preparation

Note that this preparation step is not required if the input is already a damaged video clip with holes labeled. However, in situation that users want to manually remove some objects, we provide a convenient method to generate masks in all frames. First, a few key frames are chosen to mask off the object(s) to be removed. Afterwards, the resulting holes in the key frame are tracked and located automatically in all

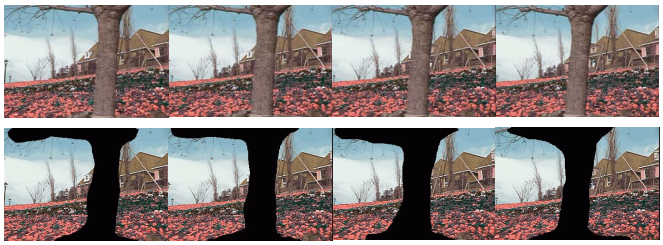


Figure 2: Sample frames from the original and damaged video.

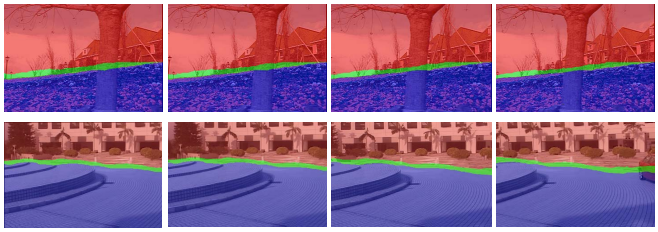


Figure 3: Video layers with overlapping boundaries. The in-between green stripe represents overlapping layer boundaries.

remaining frames by the mean shift tracking algorithm [5]. Fig. 2 shows some sample frames from the original and damaged video. The layer boundary needs not be very accurate. A rough segmentation of the static object is adequate. The video with propagated holes, or an already damaged video, is the input to our system.

2.3 Layer segmentation and propagation

Similar to object removal, a few key frames are chosen to specify layer boundaries. Note that the specified layer boundaries need not be exact, just like the foreground object boundaries in Fig. 2.

The segmentation boundaries in key frames will be automatically propagated to other frames by the mean shift tracking algorithm [5] again. We reduce the error introduced in this process by imposing a smoothness filter. Two examples are shown in Fig. 3. Here, we have the foreground layer (the

flower bed), the background layer (the house and the sky), and the overlapping boundary layer for each frame. The overlapping boundary layer is used in homography blending (next section) to eliminate visual artifacts along layer boundaries when the frames are restored from the mosaic.

After layer propagation, different layers in frames are stitched together to generate layer mosaics to guarantee temporal coherence among frames. In [19], a semi-automatic approach was proposed for image alignment. An approximate translation is first given by the user. Then an iterative approach is used to estimate perspective transform. In this work, we modify this scheme to make it fully automatic by making use of phase correlation in [7] to produce a good translation matrix. Alternatively, a Gaussian pyramid can be used to compute the initial translation. Both methods work well.

After the layered mosaics of the background have been constructed, large holes may exist. We repair the mosaics by image repairing [9]. To reconstruct all frames from the repaired mosaics, the straightforward approach of projecting the layers onto the reference view does not work when the entire scene cannot be approximated by a single layer. The reason is that the projected layer boundaries may not be consistent with each other (right of Fig. 4). Specifically, the projected boundaries have two problems: small holes and overlapping pixels.

The small holes along the projected layer boundaries are caused by the lack of the needed color information in all frames. On the other hand, after projection, some pixels receive color information from two or more layers when overlapping occurs. In the latter case, to disambiguate this situation, we may choose one color as its value (right of Fig. 4), or blend the projected colors. Although these methods work well for still images, it produces large temporal inconsistency in video sequence.

2.4 Homography blending

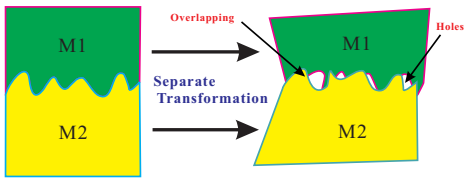


Figure 4: Overlapping and small holes are obtained by warping two layers, using their respective homographic transform.

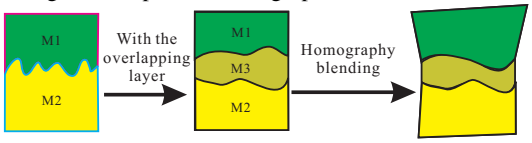


Figure 5: Homography blending by creating an overlapping boundary layer.

To reduce boundary artifacts, misregistration among frames, and to achieve better temporal coherence, we process the layer boundaries as follows (Fig. 5): instead of directly blending color values in overlapping areas, we blend the homography matrices between two adjacent layers, and apply them to the projection of pixels in the overlapping area (and the pixels around it). The detailed algorithm is as follows:

1. Mosaics are constructed for all layers. After that, a layered reference mosaic K is constructed by choosing a reference frame, to which all other layers are warped. Each layer has its own optical flow to maintain temporal consistency.
2. Let M_1 (red) and M_2 (blue) be the foreground and background layers, in a frame respectively, which are adjoined by the overlapping boundary layer M_3 (green), as depicted in Fig. 3 and Fig. 5. Let H_1^{-1} and H_2^{-1} be the homography that respectively warps $M_1 \cup M_3$ and $M_2 \cup M_3$ to the focal plane of the reference mosaic K. Blend the homographies H_1 and H_2 for M_1 and M_2 respectively, and create $H_3 = \alpha H_1 + (1-\alpha)H_2$ for M_3, where α is the blending coefficient.
3. To warp the repaired frame back, project mosaic K to layer M_i by transformation matrix H_i, $i = 1, 2, 3$.

Note that homography blending should not be applied when two layers are physically disjoint in depth, where edge discontinuity is not a problem any more.

By blending the homographies instead of pixel colors, we reduce the abrupt color value change along the occlusion boundary. α can be further described as a function of the distance to the layer boundary on the reference mosaic. If there are more than two layers, we generalize the above by processing two layers at a time, and merge with others. A hierarchical structure is used to store video layers.

In summary, we perform the following steps in order to maintain the spatial and temporal coherence of the restored video:

1. *Layer propagation*. Layer information is propagated to the entire video by the mean shift algorithm.
2. *Layered mosaics*. We adopt the method in [19] to construct layered video mosaics and the reference mosaic K. This semi-automatic technique is upgraded into a fully automatic alignment algorithm by our use of phase correlation.
3. *Frame repairing*. To achieve spatial coherence, image repairing [9] is performed on the constructed layered reference mosaic to fill holes that cannot be filled by the video mosaic construction [19]. The restored background is warped back to restore repaired frames.
4. *Homography blending*. Warp the layers from the reference mosaics back to each frame and blend the homography matrices in the overlapping area.

In the next section, based on the reconstructed background, we recover the occluded moving objects.

3 Moving foreground repairing

If the objects users want to remove also occlude moving foreground, which is quite common in reality, we need to reconstruct motion pixels and overlay them onto the repaired background. In the following, we first motivate our two-phase approach for repairing moving foreground. Then, details of the two phases, sampling and alignment, will be described.

3.1 Motivation

When repairing large motion, spatial consistency in each frame should be maintained. If the camera motion is smooth, temporal consistency should be preserved as well. The method suggested by [2] can be extended to the 3D spatio-temporal space to infer missing pixels. However, only small holes can be repaired, since texture or structure information can not be easily used.

More constraint should be enforced in order to repair very large missing motion. One reasonable constraint is motion periodicity [15]. We encode this knowledge by sampling periodic motion in a video: this corresponds to the *movel sampling phase* (section 3.2). To repair motion, we observe that the missing element inside a hole can be predicted if we know which part of the cycle it belongs to. It translates into our *movel alignment phase* to perform this prediction (section 3.3). Our algorithm is somewhat similar to [4], which also adopts an alignment scheme, processing the data in a coarse-to-fine warping framework. A *sample movel* is a video which contains at least one cycle of the periodic motion. If some frames in a movel are damaged, we call it a *damaged movel*. Our problem is thus reduced to one of aligning the sample movel with the damaged movel in order to repair the latter.

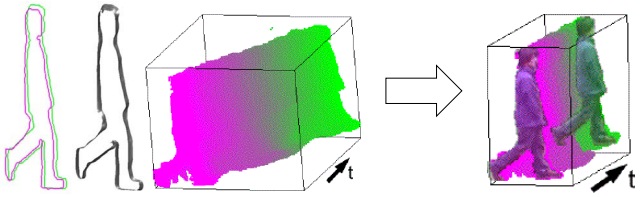
After we have collected the sample movel and located the damaged movel, we need to pre-processing them. The pre-processings are called *movel wrapping*, *movel regularization*, and *movel normalization*. Movel wrapping removes the motion discontinuity in sample movels. Movel regularization helps to maintain the temporal coherence of the repaired movel. Movel normalization reduces the search space and eliminates the velocity mismatch problem (to be described).

3.2 Phase 1: movel sampling

One requirement of the synthesized motion is to exhibit the necessary spatio-temporal consistency of the periodic motion. To achieve this, we sample a consecutive set of frames directly from the input video. As mentioned before, the two types of movels are sample movels and damaged movels. Sample movel is used to repair a damaged movel. They are illustrated in Fig. 1.

To sample a movel, the scene background should be learned first. A static or moving video camera is turned on for a few seconds to collect the mean μ and variance σ^2 of the luminance of all pixels (x, y) in the field of view. For moving camera, (x, y) refers to the image coordinates of the resulting video mosaic. After learning the μ and σ^2, moving pixels are detected by comparing the luminance of the current frame with the collected statistics. Connected components are then constructed for the moving pixels. After removing isolated noise, a video mask (Fig. 1(d)) is produced, which is used to sample a movel (Fig. 1(e)). If there are multiple motions, multiple movels will be extracted. More sophisticated and robust background model and updating rules can be used, which are not the focus of this paper.

Movel wrapping. If the first and the last frame of the sample movel do not appear the same, a "pop-up" effect will be observed when they are concatenated in the motion synthesis process. To "wrap" up a movel, we use the last 5 frames and first 5 frames of the sample movel to infer their in between frames. Using the video mask of these few frames, we first extract the 2D boundaries of the moving character. Let P_t be the set of moving pixels in frame t. We construct the connected component of P_t in each frame (and thus remove isolated noise). Let ∂P_t be the boundary pixels of the connected component. Denote the volume resulting by stacking the boundary pixels of all frames along the temporal axis by



(a) (b) (c)

Figure 6: (a) the movel boundary pixels from the first and last frames being stacked together. (b) the movel boundary surface obtained by 3D tensor voting. This spatio-temporal surface represents the in-between movel boundary for wrapping up the movel. (c) Sample movel after wrapping and normalization.

$$Boundary = \cup_{t=n-4\cdots n,1\cdots 5}\partial P_t \quad (1)$$

where n is the total number of frames in the sample movel. Thus, Eqn. (1) represents a subset of spatio-temporal movel boundary. It is used as input to the 3D tensor voting [11] to vote for a *surface* in the spatio-temporal domain, which completes the in-between movel boundary to connect the first and the last frame (Fig. 6(a), (b)):

$$Surf = SurfaceExtract(TensorVoting(Boundary)) \quad (2)$$

where $SurfaceExtract(\cdot)$ is a surface extraction procedure to produce an implicit surface representation $Surf$, from the dense tensor map produced by tensor voting. $Surf$ is used to represent the in-between movel boundary that wraps the first and last frame of the movel. Finally, view morphing [16] is applied to infer the color of the pixels of the resulting in-between frames.

Movel regularization. To preserve temporal consistency, movel regularization is performed. First, the centroid of each connected component P_t of the sample movel in a frame t is computed, by $c_t = \frac{1}{N_t}\sum(x, y)_t$, where N_t is the size of the connected component, and $(x, y)_t$ is the set of moving pixels in frame t (top left of Fig. 7 shows the centroids of all connected components in a movel). In a damaged movel, we only compute centroids for the frames where the holes do not cover (bottom left of Fig. 7). Missing centroids will be inferred by movel regularization.

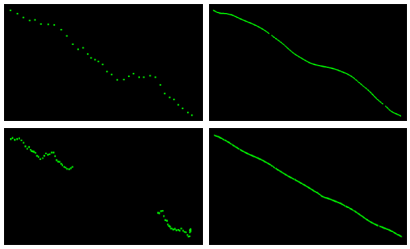
Since the centroids are found individually in each frame, the corresponding path along their temporal axes is not smooth (the curves in Fig. 7(a)). We want the repaired movel to maintain temporal coherence (the curves in Fig. 7(b)) after regularization, and call the procedure movel regularization.

Let $Centroid = \{(x, y, t)\}$ be the set of all centroids in the wrapped sample movel. 3D tensor voting [11] is used to vote

for a smooth trajectory in the spatio-temporal domain, which implicitly enforces the desired spatio-temporal coherence:

$$Curve = CurveExtract(TensorVoting(Centroid)) \quad (3)$$

where $CurveExtract(\cdot)$ is a curve extraction procedure which produces a smooth curve $Curve$, from the same dense tensor field obtained after tensor voting. As shown in the bottom of Fig. 7(b), in a damaged movel, a large portion of centroids is simply missing. To regularized a damaged movel, we perform hierarchical tensor voting in Eqn. (3), and vote for the curve in multiple scales. This is done by prefiltering and subsampling the centroids using a factor of 2. Large gap can therefore be filled. New centroids are introduced after voting, which will be used to adjust the frames of the repaired movel for maintaining temporal coherence.



(a) (b)

Figure 7: Regularizing a sample movel (top) and a damaged movel (bottom) by 3D tensor voting. The temporal axis is horizontal, and the spatial axis is vertical. (a) shows the centroids of the connected components in all frames before regularization. (b) shows the regularized centroids obtained by curve extraction using 3D tensor voting. When a large gap exists in a damaged movel (bottom left), hierarchical 3D tensor voting is run.

Movel normalization. This process translates the pixels of all frames in a movel such that each centroid is at the image center after the translation. Movel normalization provides the following benefits: (i) in movel alignment (section 3.3), Levenberg-Marquardt minimization converges much faster if we normalize both the sample and damaged movels to the image center. (ii) the velocities of the sample movel and the damaged movel to be repaired are not the same in general, thus making the initial guess difficult. Movel normalization reduces the search space in the Gaussian pyramid or phase correlation [7], allowing for good initial guess be made readily, (iii) the total number of voxels (or the spatio-temporal volume) to be processed in movel alignment can be largely reduced.

Using the first frame at time t_0 as reference, all other frames in a movel are translated such that their regularized centroids (existing or inferred) lie on a straight line parallel to the temporal axis at the frame center (Fig. 6(c)). The 2D translation for respective frame is simply $(\delta x, \delta y) = c_t - c_{t_0}$, where c_k is the centroid of frame k. Both the sample movel[1] and damaged movel[2] will be normalized using the centroids, before the estimating the alignment parameters in the next phase.

[1]The sample movel is now wrapped and regularized

[2]The damaged movel is now regularized.

Note that movels are normalized to increase efficiency in phase 2. After the alignment parameters have been estimated, the movels are *de*normalized, by applying $-(\delta x, \delta y)$ at each pixel for outputting the restored frames.

3.3 Phase 2: movel alignment

This section presents a hierarchical algorithm which aligns a sample movel with a damaged movel in order to repair the latter. No knowledge of the movel/scene is assumed. However, this algorithm can only support a subclass of camera and object motions. For example, we cannot recover a rotated face that has not been sampled in a sample movel. The subclass of camera motion we can handle consists of transformation that can be expressed by homography in the spatio-temporal space: let (x, y, t) and (x', y', t') be the sample and aligned movel coordinates respectively. They can be related by a 4×4 homographic transform (Fig. 8):

$$\begin{bmatrix} x' \\ y' \\ t' \\ 1 \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ h_8 & h_9 & h_{10} & h_{11} \\ h_{12} & h_{13} & h_{14} & h_{15} \end{bmatrix} \begin{bmatrix} x \\ y \\ t \\ 1 \end{bmatrix} \quad (4)$$

$$\mathbf{x}' = \mathbf{H}\mathbf{x} \quad (5)$$

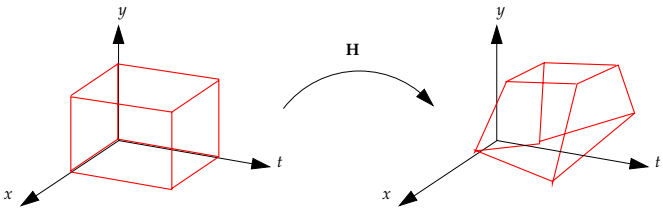


Figure 8: Spatio-temporal alignment of the sample movel to align with the damaged movel.

So, our algorithm supports a more general spatial and temporal alignment (c.f. [4]) that are 3D projective transformations. The problem has now reduced to the estimation of $\mathbf{h} = \{h_k\}, 0 \le k \le 15$. To speed up the estimation, we may turn off some parameters because rotation is not allowed. For instance, the upper 3×3 submatrix is an identity matrix if the motion to be repaired only involves translation. As usual, we minimize the squared intensity errors in the volume. Let $\mathbf{I}$ be the damaged movel, and $\mathbf{I}'$ be the aligned sample movel. Define the error term in the overlapping volume of $\mathbf{I}$ and $\mathbf{I}'$.

$$E = \sum [\mathbf{I}'(x', y', z') - \mathbf{I}(x, y, z)]^2 \quad (6)$$

We perform the optimization by Levenberg-Marquardt iterative minimization algorithm. The intensity gradient $(\frac{\partial \mathbf{I}'}{\partial x'}, \frac{\partial \mathbf{I}'}{\partial y'}, \frac{\partial \mathbf{I}'}{\partial z'})^T$ is computed at each voxel (x', y', z'). A Hessian matrix $\mathbf{A} = [a_{kl}]$ and weighted gradient vector $\mathbf{b}$ are computed:

$$a_{kl} = \sum \frac{\partial e}{\partial h_k} \frac{\partial e}{\partial h_l} \quad (7)$$

$$b_k = -\sum e \frac{\partial e}{\partial h_k} \quad (8)$$

Update to $\mathbf{h}^{(m+1)}$ is given by $\delta\mathbf{h}$:

$$\delta\mathbf{h} = (\mathbf{A} + \alpha\mathbf{I})^{-1}\mathbf{b} \quad (9)$$

$$\mathbf{h}^{(m+1)} \leftarrow \mathbf{h}^{(m)} + \delta\mathbf{h} \quad (10)$$

where α is a time-varying parameter [18].

Spatio-temporal Gaussian pyramid. Estimation efficiency can be greatly enhanced by constructing a Gaussian pyramid on the sample movel and the damaged movel, which are first converted into grey levels. The highest resolution is the sampling resolution of the movels. Lower resolution levels are obtained by prefiltering the 3D spatio-temporal data, followed by sub-sampling by a factor of 2, along the spatio-temporal axes.

After automatic initialization (by phase correlation, see the algorithm summary below) we run the Levenberg-Marquardt to refine the warping transform. Typically, $\mathbf{H}$ converges within 20 iterations, if rotation is not considered.

The algorithm.

1. Construct two spatio-temporal 3D Gaussian pyramids, one for the damaged movel: $\mathbf{S}_0 = \mathbf{I}, \mathbf{S}_1, \cdots, \mathbf{S}_L$, and one for the sample movel $\mathbf{S'}_0 = \mathbf{I'}, \mathbf{S'}_1, \cdots, \mathbf{S'}_L$, both of which are normalized. Using $\mathbf{S}_L, \mathbf{S'}_L$, initialize $\mathbf{H}$ by phase correlation proposed in [7]. Alternatively, we can estimate the initial $\mathbf{H}$ as a pure translation, by using brute force, which is not time consuming as we know the location of the hole.

2. For every resolution, $l = L \cdots 0$:

   (a) For each voxel i at location (x_i, y_i, t_i)

   i. Compute (x', y', t') by Eqn. (5).

   ii. Compute E by Eqn. (6).

   iii. Compute intensity gradient $(\frac{\partial \mathbf{I'}}{\partial x'}, \frac{\partial \mathbf{I'}}{\partial y'}, \frac{\partial \mathbf{I'}}{\partial z'})^T$.

   iv. Compute the partial derivative of $e_i = \mathbf{I'}(x', y') - \mathbf{I}(x, y)$ with respect to $h_k, 0 \leq k \leq 15$:

$$\frac{\partial e_i}{\partial h_k} = \frac{\partial \mathbf{I'}}{\partial x'}\frac{\partial x'}{\partial h_k} + \frac{\partial \mathbf{I'}}{\partial y'}\frac{\partial y'}{\partial h_k} + \frac{\partial \mathbf{I'}}{\partial z'}\frac{\partial z'}{\partial h_k} \quad (11)$$

   v. Compute $\mathbf{A}$ and $\mathbf{b}$ by adding up the contribution of pixel i, as computed by Eqn. (11).

   vi. Compute $\mathbf{h}^{(m+1)}$ using Eqn. (9) and (10).

   vii. Continue step (2) until the computed error E is below a threshold.

3. If $l \geq 1$, propagate $\mathbf{H}$ to level $l - 1$, and repeat step (2). Else, the algorithm terminates and outputs $\mathbf{H}$.

Generation of a repaired movel and video. To generate the repaired movel, we make use of our estimated $\mathbf{H}$ in the previous section to warp the sample movel. In doing so, the warped frames in the sample movel can be aligned with the frames of the damaged movel, thus achieving the purpose of repairing it.

Recall that movel normalization is performed to handle different velocities between the sample and the damaged movels. Having repaired the damaged movel, we *de*normalize the repaired movel by translating the frames back to their original positions, using the regularized centroids. Since the centroids (existing or inferred) have been regularized, and the repaired frames are restored from the sample movel, the resulting repaired frames exhibit the necessary spatio-temporal coherence.

4 The video repairing system

Now, we are ready to put together the previous sections 2 and 3 into a working system. Fig. 9 shows the overall approach of video repairing, which is comprised of background and foreground repairing.

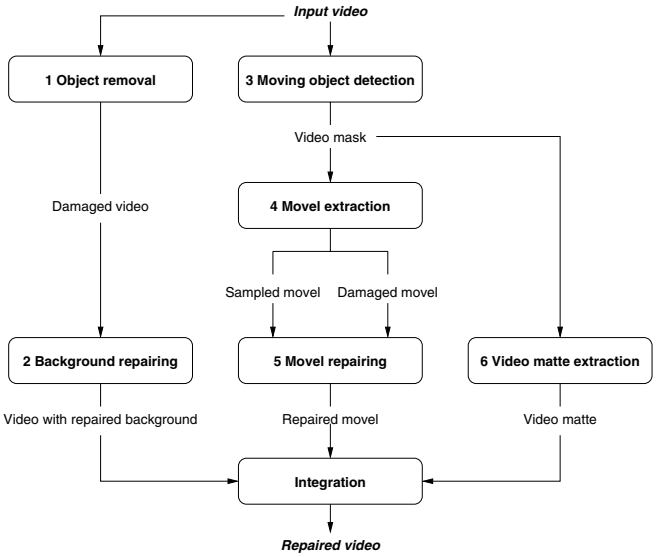


Figure 9: Flowchart of video repairing: static background repairing and moving foreground repairing.

Figure 10: Repairing the background of BRIDGE: sample input frames (left) and the restored frames (right), after bridge removal and car insertion.

A running example SCULPTURE is used in this summary (Fig. 1):

1. We apply static background repairing in section 2 to repair the background, Fig. 1(c), and extract video mask by detecting moving pixels, Fig. 1(d). The camera can be stationary or moving.
2. Use the video mask to sample movels, which will be wrapped, regularized, and normalized (section 3.2). One wrapped, regularized, and normalized sample movel is shown in Fig. 1 (e). One regularized and normalized damaged movel is shown in Fig. 1(f).
3. Repair movels by spatio-temporal alignment (section 3.3). The repaired movel is shown in Fig. 1(g).
4. Optionally, we compute the matte to blend the repaired movel with the repaired background. A video matte is constructed by the Knockout's algorithm [12]. The video matte is used to blend the repaired movel and the repaired background (Fig. 1(c)) to generate the repaired frame Fig. 1(h).

5 Results

Fig. 10 shows one static background repairing result on BRIDGE, where the bridge has been removed and a car was inserted afterwards.

The accompanying videos we submitted show the results of static background and moving foreground repairing. Fig. 11 shows the result on the SCULPTURE sequence. Five sample frames from the input video, the damaged video (with the sculpture removed) and the repaired video produced by our system are shown. In particular, we are capable of repairing the motion in the 4th frame of Fig. 11, where the character is almost occluded by the sculpture in the input video. The camera is stationary in this example. All missing motions and the previously occluded background are also repaired. Note the continuity between the synthesized motion and the existing motion: when the restored video is played, no "pop-up" artifact is seen when the person enters and exits the hole of sculpture.

Fig. 12 shows a more challenging sequence CHAIRS, where the camera is moving and multiple motions at different speed exist. This result shows the strength as well as some limitations of our video repairing system, thus directing to future research. Here, the camera is moving. There are multiple motions in the input video. The near character walks faster than the far character. Thus, the far character is occluded by the chairs and the near character. Both motions are completely occluded in the 3rd frame. All background and motions are repaired by our video repairing with acceptable visual quality, except that the shadows of the characters cannot be synthesized, since we do not sample shadows in movels.

6 Discussion and conclusion

In this paper, we repair missing static background and moving foreground pixels due to severe occlusion or damage. These two technical contributions are deployed in our video repairing system to achieve a visually plausible restoration. Because a reference video mosaic is built, our system works for a large class of camera motions including zoom-in, zoom-out, rotation about a fixed point, and panning without too much parallax after layer segmentation has been performed. As a result, spatial as well as temporal consistency are maintained. In the future, for moving foreground repairing, we are interested in incorporating more knowledge to the movels, so that lighting and shadow can be handled.

References

[1] S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *CVPR'98*, pages 434–441, 1998.

[2] M. Bertalmio, G. Sapiro, C. Ballester, and V. Caselles. Image inpainting. In *SIGGRAPH'2000*, pages 417–424, 2000.

[3] M.J. Black and A.D. Jepson. Estimating optical-flow in segmented images using variable-order parametric models with local deformations. *PAMI*, 18(10):972–986, October 1996.

[4] Y. Caspi and M. Irani. A step towards sequence-to-sequence alignment. In *CVPR00*, pages II: 682–689, 2000.

[5] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR00*, pages II: 142–149, 2000.

[6] T.J. Darrell and A.P. Pentland. Cooperative robust estimation using layers of support. *PAMI*, 17(5):474–487, May 1995.

[7] J.E. Davis. Mosaics of scenes with moving objects. In *CVPR98*, pages 354–360, 1998.

[8] M. Irani and P. Anandan. Video indexing based on mosaic representations. In *IEEE Trans. on PAMI*, pages 86(5):905–921, May 1998.

[9] Jiaya Jia and Chi-Keung Tang. Image repairing: Robust image synthesis by adaptive nd tensor voting. In *CVPR2003*, 2003.

[10] Jed Lengyel and John Snyder. Rendering with coherent layers. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 233–242. ACM Press/Addison-Wesley Publishing Co., 1997.

[11] G. Medioni, M.S. Lee, and C.K. Tang. *A Computational Framework for Feature Extraction and Segmentation*. Elseviers Science, Amsderstam, 2000.

[12] Y. Mishima. Soft edge chroma-key generation based upon hexoctahedral color space, 1993.

[13] E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. In *SIGGRAGH'95*, pages 191–198, 1995.

[14] P. Perez and M. Blake, A.and Gangnet. Jetstream: Probabilistic contour extraction with particles. In *International Conference on Computer Vision (ICCV 2001)*, 2001.

[15] Arno Schodl, Richard Szeliski, David Salesin, and Irfan Essa. Video textures. *Siggraph*, pages 489–498, 2000.

[16] Steven M. Seitz and Charles R. Dyer. View morphing. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 21–30. ACM Press, 1996.

[17] Jonathan Shade, Steven Gortler, Li wei He, and Richard Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242. ACM Press, 1998.

[18] R. Szeliski. Video mosaics for virtual environments. In *IEEE Computer Graphics and Applications*, pages 22–30, March 1996.

[19] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 251–258. ACM Press/Addison-Wesley Publishing Co., 1997.

[20] J.Y.A. Wang and E.H. Adelson. Layered representation for motion analysis. In *CVPR93*, pages 361–366, 1993.

[21] Josh Wills, Sameer Agarwal, and Serge Belongie. What went where. *CVPR*, 2003.

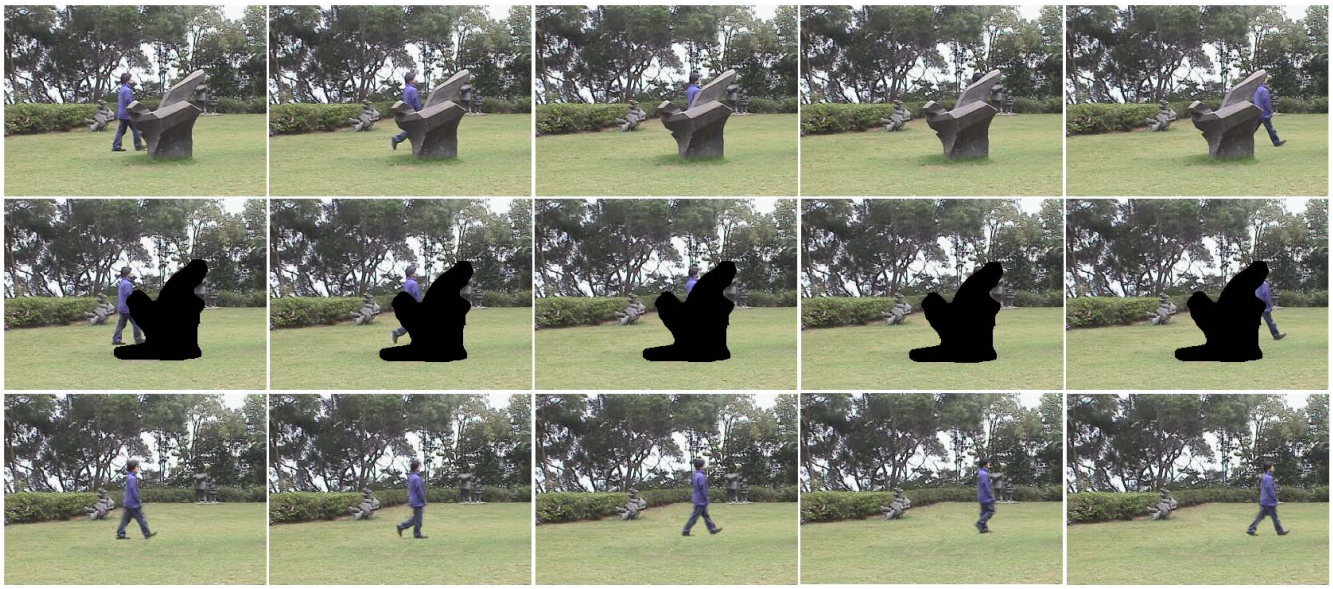


Figure 11: SCULPTURE sequence: sample frames from the input video, damaged video, and repaired video. The camera is static.

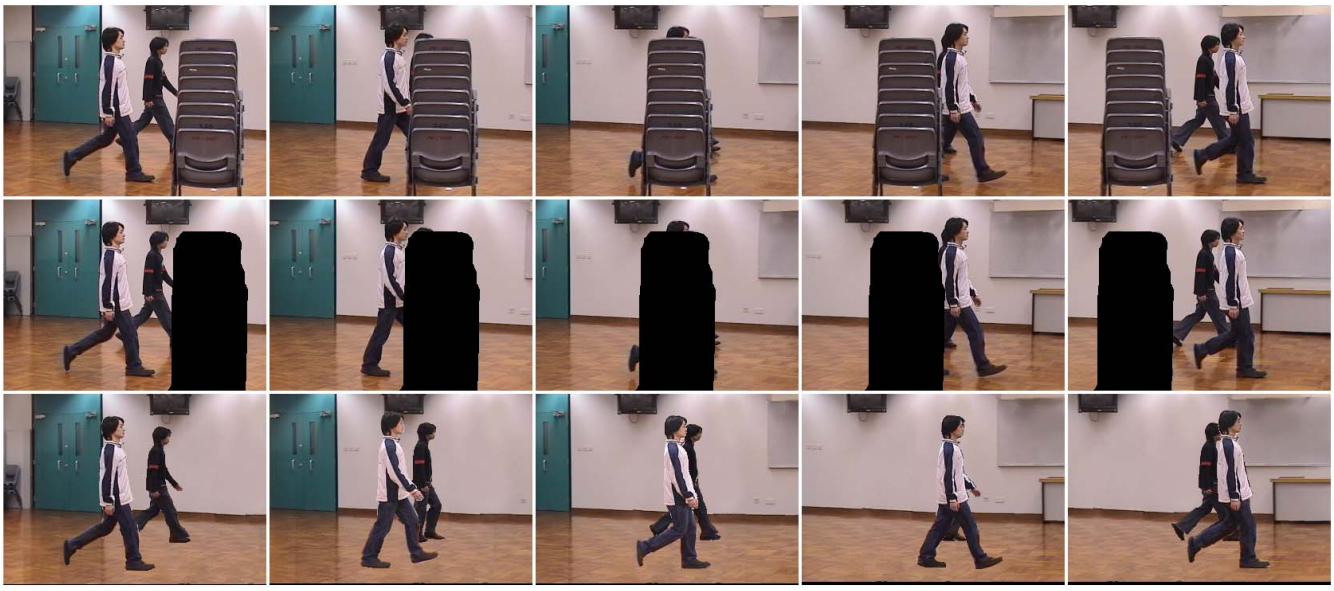


Figure 12: CHAIRS sequence: sample frames from the input video, damaged video, and repaired video. Moving camera with multiple motions at different velocities.