Video Inpainting

Sean Moran (sean.moran@cantab.net)
April, 2009

1 Introduction

Video Inpainting refers to a field of Computer Vision that aims to remove objects or restore missing or tainted regions present in a video sequence by utilizing *spatial* and *temporal* information from neighbouring scenes. The overriding objective is to generate an inpainted area that is merged seamlessly into the video so that visual coherence is maintained throughout and no distortion in the affected area is observable to the human eye when the video is played as a sequence.

It is important at this stage to distinguish video and image inpainting from the related field of "texture synthesis". The main difference between these approaches is in the size and characteristics of the region to be corrected. For texture synthesis the region can be much larger with the main focus being the filling in of two-dimensional repeating patterns that have some associated stochasticity i.e. textures. In contrast, inpainting algorithms concentrate on the filling in of much smaller regions that are characterised by linear "structures" such as lines and object contours. There has been some recent work by Criminisi et al. [1] to bridge this gap and to provide a single algorithm to handle both of these cases given both the presence of textures and structures in images.

Despite still being a relatively young field compared to the related area of image inpainting, some rather encouraging results have already been obtained by the research community. Figures 1 and 2 demonstrate a video inpainting algorithm being employed to edit two different video sequences. In Figure 1 video inpainting techniques are used to completely erase a person in a typical office surveillance scene. In Figure 2 we use a video inpainting algorithm to repair a damaged video.

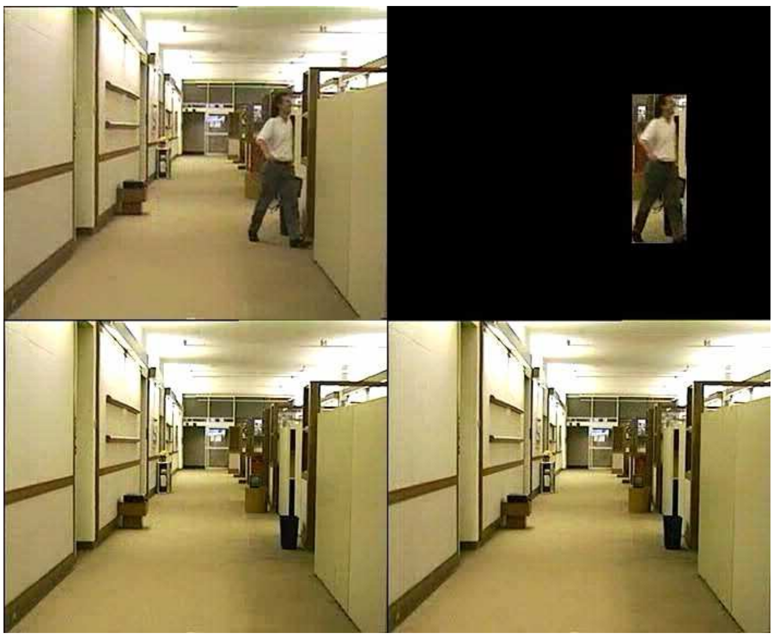


Figure 1: *In this surveillance scene, a video inpainting algorithm is applied to completely remove the person from the video. On the top left we see a frame from the original untouched video containing the person. On the right, the region containing the person is isolated from the video, leading to removal of the person on the bottom left image using a video inpainting algorithm. On the bottom right a hidden watermark of the person is embedded back into the image (note this last step is not commonly performed as part of a typical video inpainting algorithm).* Image source: http://www.vis.uky.edu/~jameszhao/video_inpainting.html

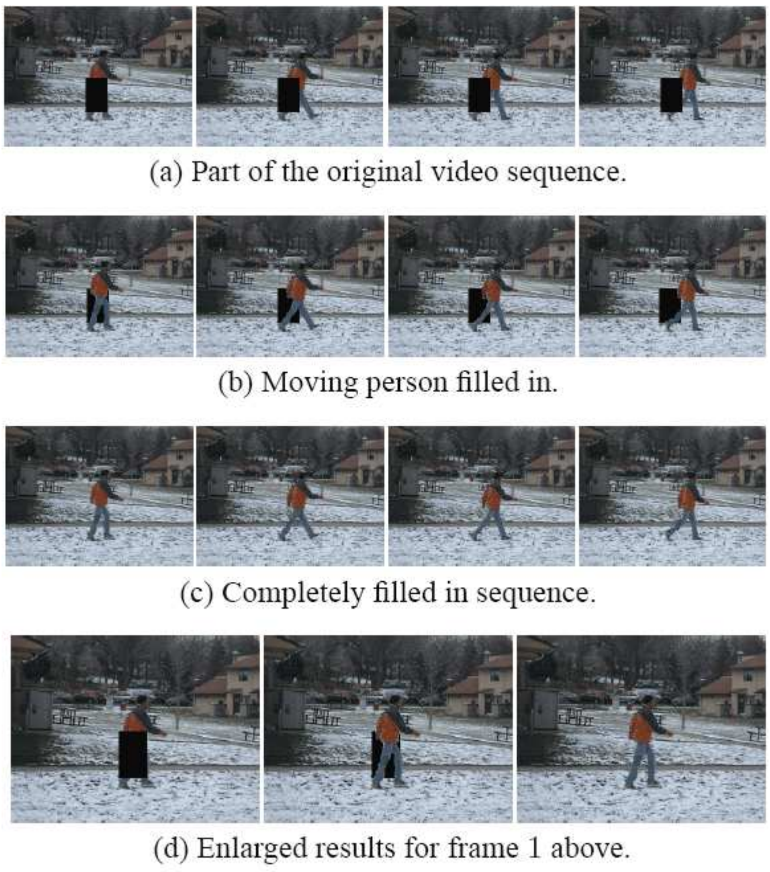

(a) Part of the original video sequence.
(b) Moving person filled in.
(c) Completely filled in sequence.
(d) Enlarged results for frame 1 above.

Figure 2: *Here we are employing a video inpainting algorithm to repair a damaged video. The damage consists of the large black region in the middle of the video sequence, possibly caused bitloss due to transmission across an unreliable network. Image (a) depicts the original video sequence - notice here how the damaged region completely occludes the bottom half of the person and the associated background. Image (b) shows the first step of the inpainting algorithm which fills in the missing portion of the person. Image (c) completes the process by filling in the missing background, with Image (d) providing a zoomed in view of the key frame with the person walking through the damaged region. Notice the excellent restoration of the video.* Image source: [5]

Interest in the video inpainting field has increased significantly amongst the research community over the past several years due to the varied and important applications of an automatic means of video inpainting. Some of the key applications include:

- **Undesired object removal:** Static or dynamic objects may not be wanted in a film, perhaps due to a change of heart by the director or simply that it was infeasible or unavoidable to exclude the objects during the initial recording (for example, a jet aeroplane flying past during the recording of an 18th century drama.)
- **Visual story modification:** Video inpainting can also be extended to change the behaviour of entities within a video. This may be required, for example, to censor an obscene gesture or action that is not deemed appropriate for the target audience, but for which it would be infeasible or expensive to re-shoot the scene.
- **Video restoration:** Videos can be damaged by scratches or dust spots or frames could simply be missing or corrupt. It is also possible that during transmission over unreliable networks, information pertaining to significant portions of video frames may be lost in transit. To view the video again in its former glory it is necessary to repair these damaged scenes in a manner that is visually coherent to the viewer.

These processes are frequently performed by restoration professionals in a manual fashion, which is not only painstaking and slow but also rather expensive. Therefore any means of automation would certainly be of benefit to both commercial organizations (such as broadcasters and film studios) and private individuals that wish to edit and maintain the quality of their video collection.

2 Why not just use methods from the field of Image Inpainting?

It is important to clarify the reason as to why it is simply not possible to extend methods put forward for image inpainting (that is, the completion of damaged regions in single images using information from the surrounding areas) to the related field of video inpainting. After all a video can just be thought of as a sequence of still images. The key point to note is that image inpainting techniques only consider *spatial* information, and completely neglect the significant *temporal* component present in video sequences. Only considering the spatial component leads to quite severe image artefacts being produced in the resulting inpainted video. This is because the essential assumption of image inpainting, that edges should be interpolated in some smooth way, is not valid when we consider the dimension of time. Non-smooth temporal changes can arise, for instance, due to a particular pixel containing background in one frame, and foreground in a following frame. It may well be the case that when the inpainted video frames are taken separately no artefacts are visible, but only when played in a sequence do the artefacts (for example, a phantom object drifting steadily through the video) manifest themselves to the human eye.

3 Previous Work

Video inpainting is a particularly challenging problem. Difficulties arise due to the possibilities of camera motion, the requirement to handle both static and dynamic regions, as well as the pose and scale variation, lighting and shadows, background clutter, deformation and occlusions that are commonly present in a video. The problem is usually made tractable by imposing some limiting constraints and solving for a less complex subclass of the problem. Common constraints include a static camera, the handling of small ranges of motions only, and a limited size of region to be repaired (commonly this is quite small).

The literature is replete with a varied array of video inpainting proposals each advocating different approaches to the problem, some of which we will briefly mention hereunder. Bertalmio et al. [1] investigated the extent to which ideas from computational fluid dynamics could be applied to image and video inpainting. Here the authors only consider spatial information in a video and perform the inpainting on a frame by frame basis leading to limited applicability of the algorithm.

In contrast, Wexler et al. [2] take account of the spatial and temporal dimensions in their space-time video completion approach which enables the model to handle video sequences of complex dynamic scenes. Here the authors attempt to solve the inpainting problem by sampling a set of spatial-temporal patches (a set of pixels at frame t) from other frames to fill in the missing data. Global consistency is enforced for all patches surrounding the missing data so as to ensure coherence of all surrounding space time patches. This avoids artefacts such as multiple recovery of the same background object and the production of inconsistent object trajectories. This method provides decent results, however it suffers from a high computational load and requires a long video sequence of similar scenes to increase the probability of correct matches.

Patwardhan et al. [5] suggest a rather simpler method for inpainting stationary background and moving foreground in videos. To inpaint the stationary background a relatively simple spatio-temporal priority scheme is employed where undamaged pixels are copied from frames temporally close to the damaged frame, followed by a spatial filling in step which replaces the damaged region with a best matching patch so as to maintain a consistent background throughout the sequence.

In their paper [4], Jia et al. propose an inpainting method that aims to tackle the challenging issue of inpainting under the presence of severe occlusion. The authors segment a video into a moving object layer and a static background layer. The techniques of layer segmentation and homography blending are used to recover static background pixels. Moving foreground pixels are repaired by sampling motion data followed by the application of 3D tensor voting so as to maintain temporal coherence and motion periodicity. Missing dynamic foreground pixels are then inferred by spatial and temporal alignment of the sampled motion data. This work is significantly more complex than other proposals in the literature, but this added complexity enables the algorithm to handle a range of camera motions, including zoom-in, zoom-out, rotation about a fixed point, and panning.

It is quite common for most of the approaches in the literature to require the user to specify the damaged region to be repaired. However in [6], the authors manually show how it is possible to identify regions to be repaired in an automatic fashion using motion cues. This is certainly an important step for the widespread adoption of video inpainting technologies amongst the commercial and private communities.

In this article we will discuss the video inpainting proposal put forward by Patwardhan et al. [5] which describes a simple, fundamental approach to the problem making it ideal for the purposes of introducing and illustrating the core concepts in the field. The reader is encouraged to consult the further reading references at the end of this article for pointers to other papers describing the alternative methods of video inpainting.

4 Video Inpainting of Occluding and Occluded Objects

In their paper, *"Video Inpainting of Occluding and Occluded Objects"*, Patwardhan et al. suggest a basic technique to achieve the following two inpainting tasks:

1. Removal of non-stationary objects that occlude stationary background. The algorithm ensures that temporal consistency is maintained after the removal of the object.

2. Filling in moving objects that may be partially occluded. This is performed whilst ensuring that the motion is globally consistent throughout the video.

To achieve these goals the authors build upon the previous 2-D texture synthesis work of [3] and [7]. To get an intuitive understanding of the video inpainting algorithm proposed by Patwardhan et al. it is instructive to take a brief detour to examine the 2-D texture synthesis work of Criminisi et al. [7], given that the video inpainting paper heavily relies on the suggested techniques in this paper. For this we will shift the analysis briefly to that of inpainting single *images* (rather than video inpainting).

4.1 Region Filling and Object Removal by Exemplar-Based Image Inpainting

The 2-D texture synthesis algorithm proceeds by a user manually selecting a region Ω in the image that is to be removed and filled. The source region or Φ is defined as the entire image minus the target region that is to be filled or $\Phi = I - \Omega$ where I denotes the image of interest. Figure 3 illustrates the main parameters used in the algorithm.

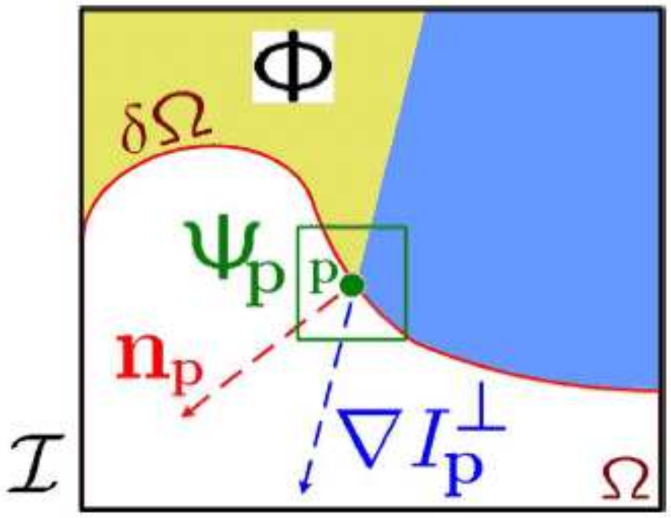


Figure 3: Ψ_p *denotes the patch to be filled.* $\mathbf{n_p}$ *is the normal to the region contour* $\delta\Omega$ *of the target region* Ω *and* $\Delta I_p^{\perp}$ *is the isophote at point* $\mathbf{p}$. *I denotes the entire image.* Image source: [7]

As the diagram illustrates, the target region to be inpainted is denoted by Ω with its contour given by $\delta\Omega$. This contour moves in an inward direction during the execution of the texture synthesis algorithm. A best first filling strategy is employed to fill the desired region based on calculated patch *priorities.* Two key ideas guide the calculation of these priorities:

1. Patches on the continuation of strong edges are preferred.
2. As well as those patches that are surrounded by high-confidence (in terms of their value) pixels.

Referring to Figure 3, given the patch Ψ_p, the priority $P(\mathbf{p})$ of this patch is defined as:

$$P(\mathbf{p}) = \mathbf{C}(\mathbf{p})\mathbf{D}(\mathbf{p}) \tag{1}$$

Where in this equation, $C(\mathbf{p})$ is the so-called *confidence* term and $D(\mathbf{p})$ is the *data* term. Intuitively, the confidence term attempts to represent the "reliability" of the information surrounding the pixel $\mathbf{p}$. Patches are deemed reliable if one or more of the following criterion are met:

1. More pixels are already filled in within the relevant patch.
2. Pixels in the patch were filled in early on in the execution of the algorithm.
3. The patch contains pixels that were not initially part of the target region.

In contrast, the data term attempts to give priority to the filling in of linear structures (e.g. people as opposed to textures) first, and this is quantified by measuring the strength of the isophotes (contours of equal luminance in an image) arriving at the contour $\delta\Omega$. Mathematically, these terms are defined as follows:

$$C(\mathbf{p}) = \frac{\boldsymbol{\Sigma}_{\mathbf{q}\in\boldsymbol{\Psi}_{\mathbf{p}}\cap(\mathbf{I}-\boldsymbol{\Omega})}\mathbf{C}(\mathbf{q})}{|\boldsymbol{\Psi}_{\mathbf{p}}|} \tag{2}$$

$$D(\mathbf{p}) = \frac{|\nabla\mathbf{I}_{\mathbf{p}}^{\perp}.\mathbf{n}_{\mathbf{p}}|}{\alpha} \tag{3}$$

Here $|\Psi_{\mathbf{p}}|$ is the area of $\Psi_{\mathbf{p}}$, α is a normalization constant, $\mathbf{n}_{\mathbf{p}}$ is a unit vector orthogonal to the boundary $\delta\Omega$ at point $\mathbf{p}$. A patch is associated with each point on the boundary.

Given these equations, the algorithm proceeds to propagate the data from a source patch to a target patch, where the target patch is that patch having the highest calculated priority $\mathbf{p}$ using Equation 1. The source patch is found by applying a distance measure (specifically the sum of squared distances measure operating on CIELab pixel values) to ascertain the similarity between patches from the source region and the target patch. The patch in the source region having the most "similarity" to a patch in the target region is the patch selected for copying to the target patch. Once the copying has been performed, the confidence values of all those pixels in the intersection of the target region and the region enclosed by the patch are then set to the confidence value of the highest priority pixel. The algorithm then proceeds in this manner until the desired region is inpainted. Figure 4 illustrates the operation of the inpainting algorithm on an example image.

Having now provided an overview of the 2-D texture synthesis algorithm of Criminisi et al. we are in a position to continue our discussion of the salient features of the video inpainting algorithm of Patwardham et al.

4.2 Video Inpainting of Stationary Background

We will firstly consider how the authors propose to inpaint *stationary background* in a video. Using the Equation 1, the proposed algorithm firstly assigns confidence values to every pixel in every frame of the video, with a value of 0 given to the moving foreground and damaged pixels (and 1 to the remaining pixels).

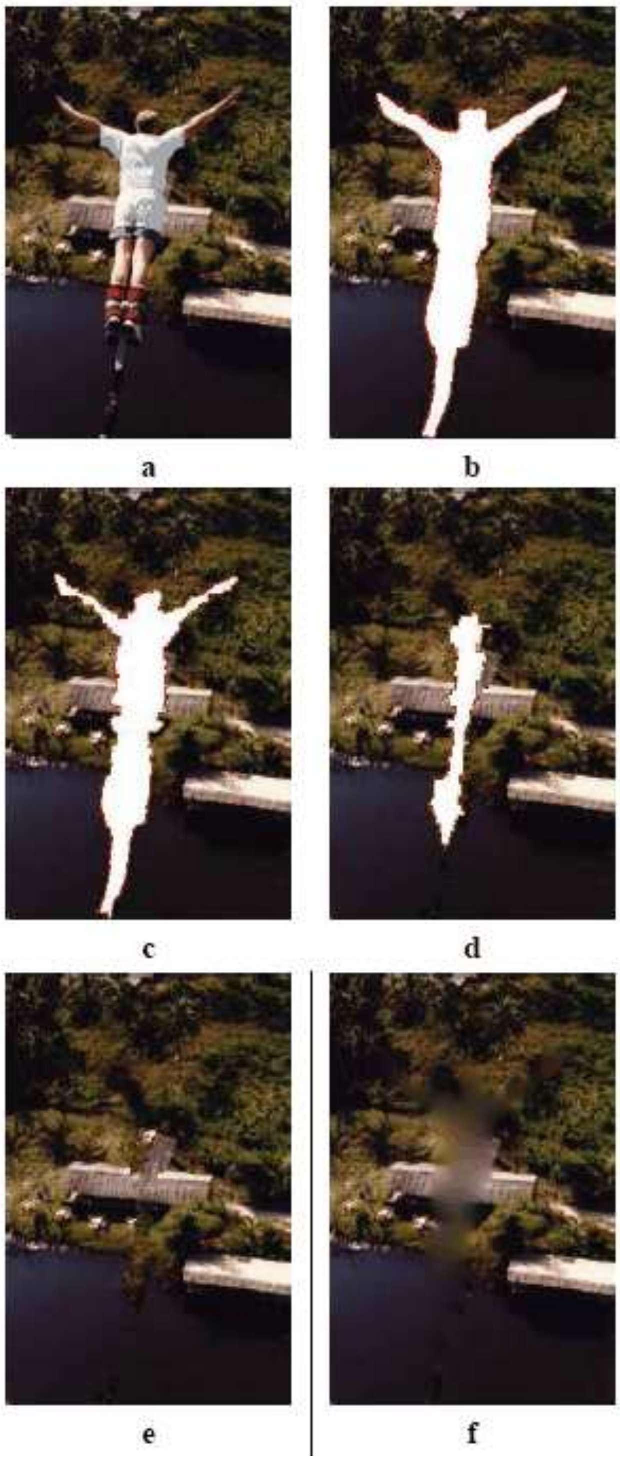


Figure 4: *Image (a) shows the original unedited image. (b) depicts the target region in white that is to be inpainted. (c) and (d) show different stages of the inpainting process with the final result being displayed in image (e). Image (f) shows a competing inpainting algorithm which performs less well due to the blurring and lack of consideration given to texture synthesis.* Image source: [7]

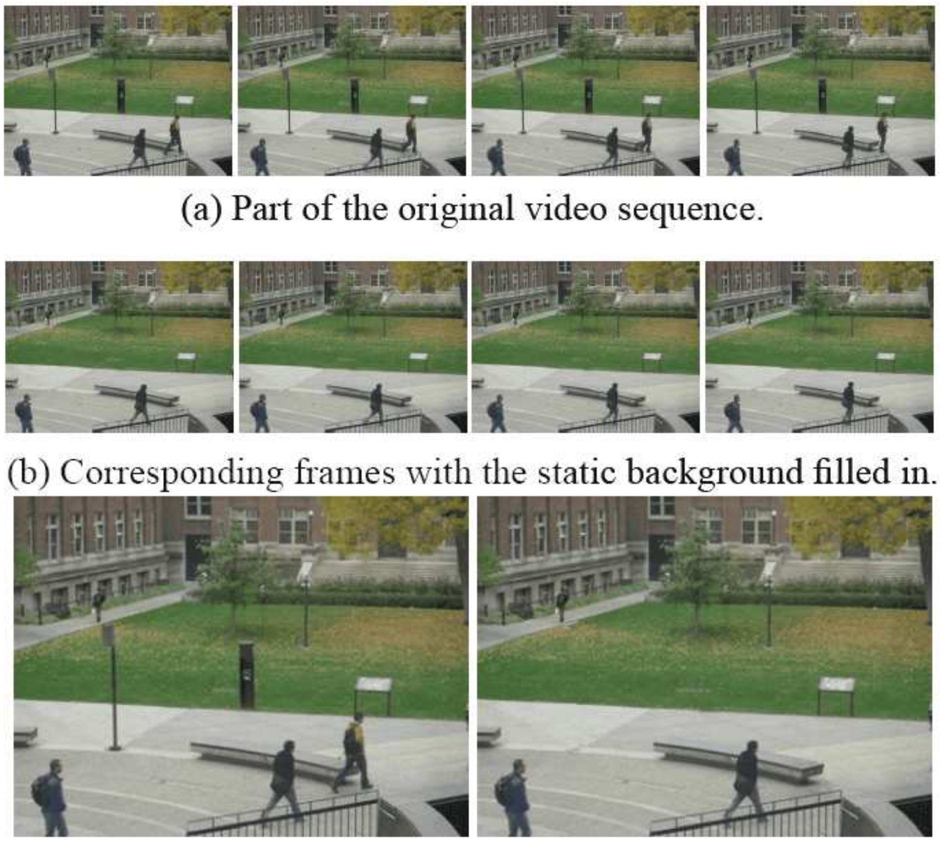
In contrast to the data term (Equation 3) proposed by Criminisi et al., the authors of this paper use a modified data term specifically designed to measure the abundance of temporal information at location p. This was defined as follows:

$$D(\mathbf{p}) = \frac{\sum_{\mathbf{p}\in\delta\mathbf{\Omega},\mathbf{t}=-\delta\mathbf{n}\ldots\delta\mathbf{n}} \mathbf{M_t}(\mathbf{p})}{\beta} \tag{4}$$

Where, in this equation M_t is set to zero if $\mathbf{p}$ is damaged or moving otherwise it is set to the value of 1. The parameter t measures the relative position of any frame from the current frame under consideration, β is a normalizing constant and n is the number of previous and next frames to be considered in the video.

Equation 1 is used to calculate the priority of the pixel $\mathbf{p}$ as before, with the resultant priority denoting the frame and pixel that will be filled in first with background information. Those patches within the closest temporal locality (i.e. frame) of the current frame with the highest confidence are copied to the location $\mathbf{p}$. This process is repeated until $D(\mathbf{p}) = 0$, which indicates that no more temporal information is available to copy.

At this stage a hole will be found at the same location in all of the video frames. A spatial filling in of the holes then proceeds according to Equation 3 - a highest priority location is found along with its best matching patch, and this patch is copied to replace the hole in *all* of the frames thereby ensuring a consistent background throughout the sequence. Figure 5 demonstrates the process of filling in the static background on an example video.



(a) Part of the original video sequence.

(b) Corresponding frames with the static background filled in.

(c) Enlarged results for frame 1 above.

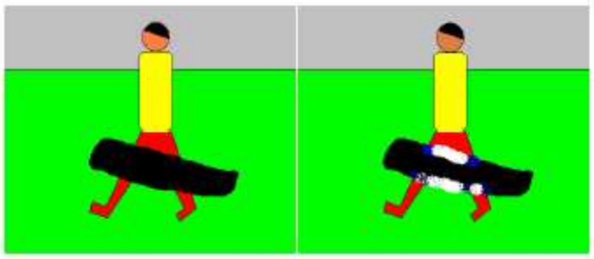
Figure 5: *This diagram illustrates the operation of the inpainting algorithm on inpainting a static background.* Image source: [5]

4.3 Video Inpainting of Moving Foreground

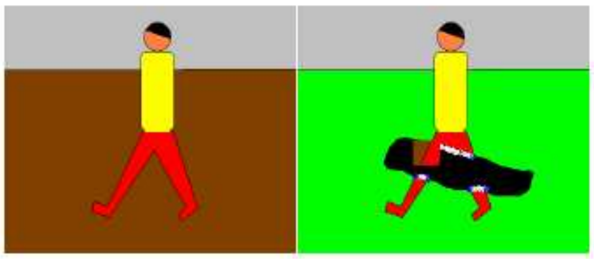
Figure 6 depicts the process adopted to fill in a moving object that is partially occluded by a stationary or moving second object. The algorithm proceeds (for the current frame) by firstly identifying the highest priority location for filling in the moving object. This priority is calculated again using Equation 1 but with an amended data term, defined as:

$$D(\mathbf{p}) = \frac{|(\nabla \mathbf{M}_{\mathbf{c}}^{\perp})_{\mathbf{p}}.\mathbf{n}_{\mathbf{p}}|}{\alpha} \tag{5}$$

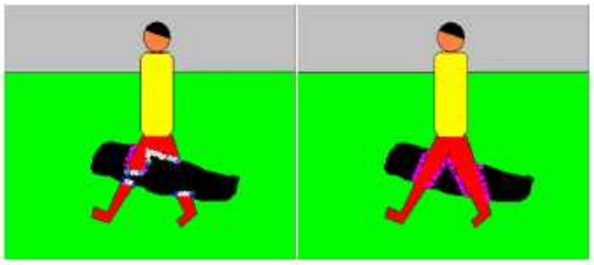
In this equation, M_c denotes a so-called *motion confidence image* which is zero if a pixel $\mathbf{p}$ belongs to the static background and 1 if the pixel belongs to the moving foreground. A sum of squared distances (SSD) metric is then used to find a best matching patch from another frame in the video. The SSD is computed for both the patch to be filled and on candidate replacement patches and consists of a 5 vector containing the RGB values and the horizontal and vertical velocities of the relevant patches. The moving part of the best matching patch is then copied to the current frame and the confidence values of the remaining pixels are updated as in Equation 2. Having performed this step the priority of non-moving pixels in the copied patch are set to zero (see Figure 6, image (c)). The algorithm iterates in this manner until all the pixels to be filled in have zero priority, at which point the object is deemed to have been inpainted.



(a) Damaged frame on the left and initial priorities indicated on the right. Blue and white indicate high and low priorities respectively.



(b) Best matching frame on the left and copied patch with different background shown on the right.



(c) Only moving part of the patch is copied (left) and the priority is constrained (purple indicates zero priority). After the moving object is completed there is no damaged pixel with high priority.

Figure 6: *This diagram illustrates the operation of the inpainting algorithm on inpainting a moving object in the foreground.* Image source: [5]

5 Discussion

This article has discussed one particular example of video inpainting, the paper by Patwardhan et al. [5], which proposes a relatively simply inpainting algorithm that builds upon the seminal 2-D texture synthesis work of Criminisi et al. [7]. This algorithm produced reasonably good results that were comparable to existing methods proposed in the literature. However this performance was traded off against the imposition of rather strong limiting constraints on the imaging conditions, in this case the requirement for a static camera. Clearly, in practice, many videos will have a camera that moves to some extent and so it is apparent from this short article that much work still needs to be performed in the field to realise the ultimate goal of a fully robust and reliable algorithm that can operate under a myriad of difficult imaging conditions, including camera motion and other conditions such as severe occlusion and cluttered background.

6 Further Reading

- [1] M. Bertalmio, A. L. Bertozzi, and G. Sapiro. "Navier-stokes, fluid dynamics, and image and video inpainting", *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, Vol. I, pp. 355-363, 2001.

- [2] Y. Wexler, E. Shechtman, and M. Irani. "Space-time video completion", *Proceedings. 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. I, pp. 120-127, 2004.

- [3] A. Efros and T. Leung. "Texture synthesis by non-parametric sampling", *IEEE International Conference on Computer Vision, Corfu, Greece*, pp. 1033-1038, 1999.

- [4] J. Jia, T. Wu, Y. Tai, and C. Tang. "Video repairing: Inference of foreground and background under severe occlusion", *Proceedings. 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. I, pp. 364-371, 2004.

- [5] K.A. Patwardhan, G. Sapiro, and M. Bertalmio. "Video inpainting of occluding and occluded objects", *in Proc. ICIP 2005.*, Vol. II, pp. 69-72.

- [6] C. Rasmussen and T. Korah. "Spatiotemporal Inpainting for Recovering Texture Maps of Occluded Building Facades", *IEEE Transactions on Image Processing*, 16(9), pp. 2262-2271, 2007.

- [7] A. Criminisi, P. Perez, and K. Toyama. "Region Filling and Object Removal by Exemplar-Based Image Painting", *IEEE Transactions on Image Processing*, 13(9), pp. 1200-1212, 2004.

7 Useful URLs

- Image and Video Inpainting:
  - http://mountains.ece.umn.edu/~guille/inpainting.htm
- Efficient Video Inpainting Based on Object Tracking and Interpolation:
  - http://www.vis.uky.edu/~jameszhao/video_inpainting.html
- Video Inpainting of occluding and occluded objects:
  - http://www.tc.umn.edu/~patw0007/icip2005