SIAM J. IMAGING SCIENCES
Vol. xx, pp. x

x–x

VIDEO INPAINTING OF COMPLEX SCENES

Alasdair Newson[†] [‡], Andrés Almansa [‡], Matthieu Fradet [†], Yann Gousseau [‡], and Patrick Pérez [†]

Abstract. We propose an automatic video inpainting algorithm which relies on the optimisation of a global, patch-based functional. Our algorithm is able to deal with a variety of challenging situations which naturally arise in video inpainting, such as the correct reconstruction of dynamic textures, multiple moving objects and moving background. Furthermore, we achieve this in an order of magnitude less execution time with respect to the state-of-the-art. We are also able to achieve good quality results on high definition videos. Finally, we provide specific algorithmic details to make implementation of our algorithm as easy as possible. The resulting algorithm requires no segmentation or manual input other than the definition of the inpainting mask, and can deal with a wider variety of situations than is handled by previous work.

Key words. Video inpainting, patch-based inpainting, video textures, moving background.

AMS subject classifications. 68U10, 65K10, 65C20,

1. Introduction.

Advanced image and video editing techniques are increasingly common in the image processing and computer vision world, and are also starting to be used in media entertainment. One common and difficult task closely linked to the world of video editing is image and video "inpainting". Generally speaking, this is the task of replacing the content of an image or video with some other content which is visually pleasing. This subject has been extensively studied in the case of images, to such an extent that commercial image inpainting products destined for the general public are available, such as Photoshop's "Content Aware fill" [1]. However, while some impressive results have been obtained in the case of videos, the subject has been studied far less extensively than image inpainting. This relative lack of research can largely be attributed to high time complexity due to the added temporal dimension. Indeed, it has only very recently become possible to produce good quality inpainting results on high definition videos, and this only in a semi-automatic manner. Nevertheless, high-quality video inpainting has many important and useful applications such as film restoration, professional post-production in cinema and video editing for personal use. For this reason, we believe that an automatic, generic video inpainting algorithm would be extremely useful for both academic and professional communities.

1.1. Prior work.

The generic goal of replacing areas of arbitrary shapes and sizes in images by some other content was first presented by Masnou and Morel in [29]. This method used level-lines to *disocclude* the region to inpaint. The term "inpainting" was first introduced by Bertalmio *et al.* in [7]. Subsequently, a vast amount of research was done in the area of image inpainting [6], and to a lesser extent in *video* inpainting.

Generally speaking, video inpainting algorithms belong to either the "object-based" or "patch-based" category. Object-based algorithms usually segment the video into moving fore-

---

[†] Technicolor, 35570 Cesson-Sévigné, France
[‡] Télécom ParisTech, CNRS LTCI, 75013 Paris, France

ground objects and background that is either still or displays simple motion. These segmented image sequences are then inpainted using separate algorithms. The background is often inpainted using image inpainting methods such as [13], whereas moving objects are often copied into the occlusion as smoothly as possible. Unfortunately, such methods include restrictive hypotheses on the moving objects' motion, such as strict periodicity. Some object-based methods include [12, 23, 27].

Patch-based methods are based on the intuitive idea of copying and pasting small video "patches" (rectangular cuboids of video information) into the occluded area. These patches are very useful as they provide a practical way of encoding local texture, structure and motion (in the video case).

Patches were first introduced for texture synthesis in images [17], and subsequently used with great success in image inpainting [8, 13, 16]. These methods copy and paste patches into the occlusion in a greedy fashion, which means that no global coherence of the solution can be guaranteed. This general approach was extended by Patwardhan *et al.* to the spatio-temporal case in [33]. In [34], this approach was further improved so that moving cameras could be dealt with. This is reflected by the fact that a good segmentation of the scene into moving foreground objects and background is needed to produce good quality results. This lack of global coherence can be a drawback, especially for the correct inpainting of moving objects.

Another method leading to a different family of algorithms was presented by Demanet *et al.* in [15]. The key insight here is that inpainting can be viewed as a labelling problem: each occluded pixel can be associated with an unoccluded pixel, and the final labels of the pixels result from a discrete optimisation process. This idea was subsequently followed by a series of image inpainting methods [21, 24, 28, 35] which use optimisation techniques such as graph cuts algorithms [9], to minimise a global patch-based functional. The vector field representing the correspondences between occluded and unoccluded pixels is referred to as the *shift map* by Pritch *et al.* [35], a term which we shall use in the current work. This idea was extended to the video case by Granados *et al.* in [19]. They propose a semi-automatic algorithm which optimises the spatio-temporal shift map. This algorithm presents impressive results on higher resolution images than are previously found in the literature (up to 1120×754 pixels). However, in order to reduce the large search space and high time complexity of the optimisation method, manual tracking of moving occluded objects is required. To the best of our knowledge, the inpainting results of Granados *et al.* are the most advanced to date, and we shall therefore compare our algorithm with these results.

We also note the work of Herling and Broll [22] whose goal is "diminished reality", which considers the inpainting task coupled with a tracking problem. This is the only approach of which we are aware which inpaints videos in a real-time manner. However, the method relies on restrictive hypotheses on the nature of the scene to inpaint and can therefore only deal with tasks such as removing a static object from a rigid platform.

Another family of patch-based video inpainting methods was introduced in the seminal work of Wexler *et al.* [37]. This paper proposes an iterative method that may be seen as an heuristic to solve a global optimisation problem. This work is widely cited and well-known in the video inpainting domain, mainly because it ensures global coherency in an automatic manner. This method is in fact closely linked to methods such as non-local denoising [10]. This link was also noted in the work of Arias *et al.* [3], which introduced a general non-local patch-based variational framework for inpainting in the image case. In fact, the algorithm of Wexler *et al.* may be seen as a special case of this framework. We shall refer to this general approach as the *non-local patch-based* approach. Darabi *et al.* have presented another variation on the work of Wexler *et al.* for image inpainting purposes in [14]. In the video inpainting case, the high dimensionality of the problem makes such approaches extremely slow, in particular due to the nearest neighbour search, requiring up to several days for a few seconds of VGA video. This problem was considered in our previous work [30], which represented a first step towards achieving high quality video inpainting results even on high resolution videos. Given the flexibility and potential of the non-local patch-based approach, we use it here to form the core of the proposed method. In order to obtain an algorithm which can deal with a wide variety of complex situations, we consider and propose solutions to some of the most important questions which arise in video inpainting.

1.2. Outline and contributions of the paper. The ultimate goal of our work is to produce an automatic and generic video inpainting algorithm which can deal with complex and varied situations. First of all, we tackle the problem of long execution times, which is a significant obstacle for research in video inpainting. This is due to the heavy computational load of the search for the *nearest neighbours* of video patches. In Section 2.1 we address this problem by proposing an extension of the PatchMatch algorithm [5] to the spatio-temporal case, which greatly accelerates this nearest neighbour search. Dealing with this problem is very important, as it is very difficult to test and experiment with the algorithm without reasonable execution times. Secondly, in Section 2.3, we look in detail at inpainting videos which contain dynamic video textures. Indeed, the reconstruction of textures and fine details is one of the main limitations of video inpainting methods. To deal with this, we propose a modification to the patch distance in order to identify these textured areas. We also create a multi-resolution texture feature pyramid to aid the correct reconstruction of textures at all resolutions. Thirdly, we deal with the problem of moving background, using a robust affine estimation of the dominant motion in the video (see Section 3.1). This turns out to be a crucial point to resolve, even in the case of relatively small motions, such as the shaking of a hand-held camera. We also consider the important question of how to initialise the inpainting solution (Section 3.2), and provide precise implementation details which are often left out in the literature (§3).

As shown in Section 4, the proposed algorithm produces high quality results in an automatic manner, in a wide range of complex video inpainting situations: moving cameras, multiple moving objects, changing background and dynamic video textures. No pre-segmentation of the video is required. One of the most significant advantages of the proposed method is that it deals with all these situations in a single framework, rather than having to resort to separate algorithms for each case, as in [19] and [18] for instance. In particular, the problem of reconstructing dynamic video textures has not been previously addressed in other inpainting algorithms. This is a worthwhile advantage since the problem of synthesising video textures, which is usually done with dedicated algorithms, can be achieved in this single, coherent framework. Finally, our algorithm does not need any manual input other than the inpainting mask, and does not rely on foreground/background segmentation, which is the case of many other approaches [12, 19, 23, 27, 34].

2. Main steps of proposed algorithm. As we have stated in the introduction, our video inpainting algorithm takes a non-local patch-based approach. At the heart of such algorithms lies a global patch-based functional which is to be optimised. We optimise this with an iterative algorithm, as is commonly found in algorithms using such a formulation of the problem [3, 14, 38]. The central machinery of the algorithm is based on the alternation of two core steps: a search for the nearest neighbours of patches which contain occluded pixels, and a reconstruction step based on the aggregation of the information provided by the nearest neighbours.

This iterative algorithm is embedded in a multi-resolution pyramid scheme, similarly to [16, 38]. The multi-resolution scheme is vital for the correct reconstruction of structures and moving objects in large occlusions. In our work, we also propose an additional multi-resolution pyramid which reflects textural attributes of the video, so as to correctly reconstruct video textures. This second pyramid is reconstructed in parallel to that of the video colour information. It is important to point out that both pyramids contribute to the nearest neighbour search, and therefore jointly determine the reconstruction of the colour and texture attributes at each iteration.

Furthermore, a robust estimation of affine, dominant motion in the video is used for a pre-processing step which allows us to deal with the case of moving background and cameras. This step is crucial in order to obtain repetitive spatio-temporal patches, even in the case of motion with small amplitude.

All algorithmic choices are made explicit (such as those concerning the use of multi-resolution pyramids), so that the algorithm is readily useable. A summary of the different steps in our algorithm can be seen in Figure 1.

Notation. Before describing our algorithm, let us first of all set down some notation. A diagram which summarises this notation can be seen in Figure 2. Let $u : \Omega \rightarrow \mathbb{R}^3$ represent the colour video content, defined over a spatio-temporal volume Ω. In order to simplify the notation, u will correspond both to the information being reconstructed inside the occlusion, and the unoccluded information which will be used for inpainting. We denote a spatio-temporal position in the video as $p = (x, y, t) \in \Omega$ and by $u(p) \in \mathbb{R}^3$ the vector containing the colour values of the video at this position.

Let $\mathcal{H}$ be the spatio-temporal occlusion (the "hole" to inpaint) and $\mathcal{D}$ the data set (the unoccluded area). Note that $\mathcal{H}$ and $\mathcal{D}$ correspond to spatio-temporal *positions* rather than actual video content and that they form a partition of Ω, that is $\Omega = \mathcal{H} \cup \mathcal{D}$ and $\mathcal{H} \cap \mathcal{D} = \emptyset$.

Let $\mathcal{N}_p$ be a spatio-temporal neighbourhood of p. This neighbourhood is defined as a rectangular cuboid centred on p. The video *patch* centered at p is defined as vector $W_p^u = (u(q_1) \cdots u(q_N))$ of size $3 \times N$, where the N pixels in $\mathcal{N}_p$, $q_1 \cdots q_N$, are ordered in a predefined way.

Let us note $\tilde{\mathcal{D}} = \{p \in \mathcal{D} : \mathcal{N}_p \subset \mathcal{D}\}$ the set of unoccluded pixels whose neighborhood is also unoccluded (video patch W_p^u is only composed of known color values). We shall only use patches stemming from $\tilde{\mathcal{D}}$ to inpaint the occlusion. Also, let $\tilde{\mathcal{H}} = \cup_{p \in \mathcal{H}} \mathcal{N}_p$ represent a dilated version of $\mathcal{H}$.

Given a distance $d(\cdot, \cdot)$ between video patches, a key tool for patch-based inpainting is to define a correspondence map that associates to each pixel $p \in \Omega$ (notably those in occlusion)

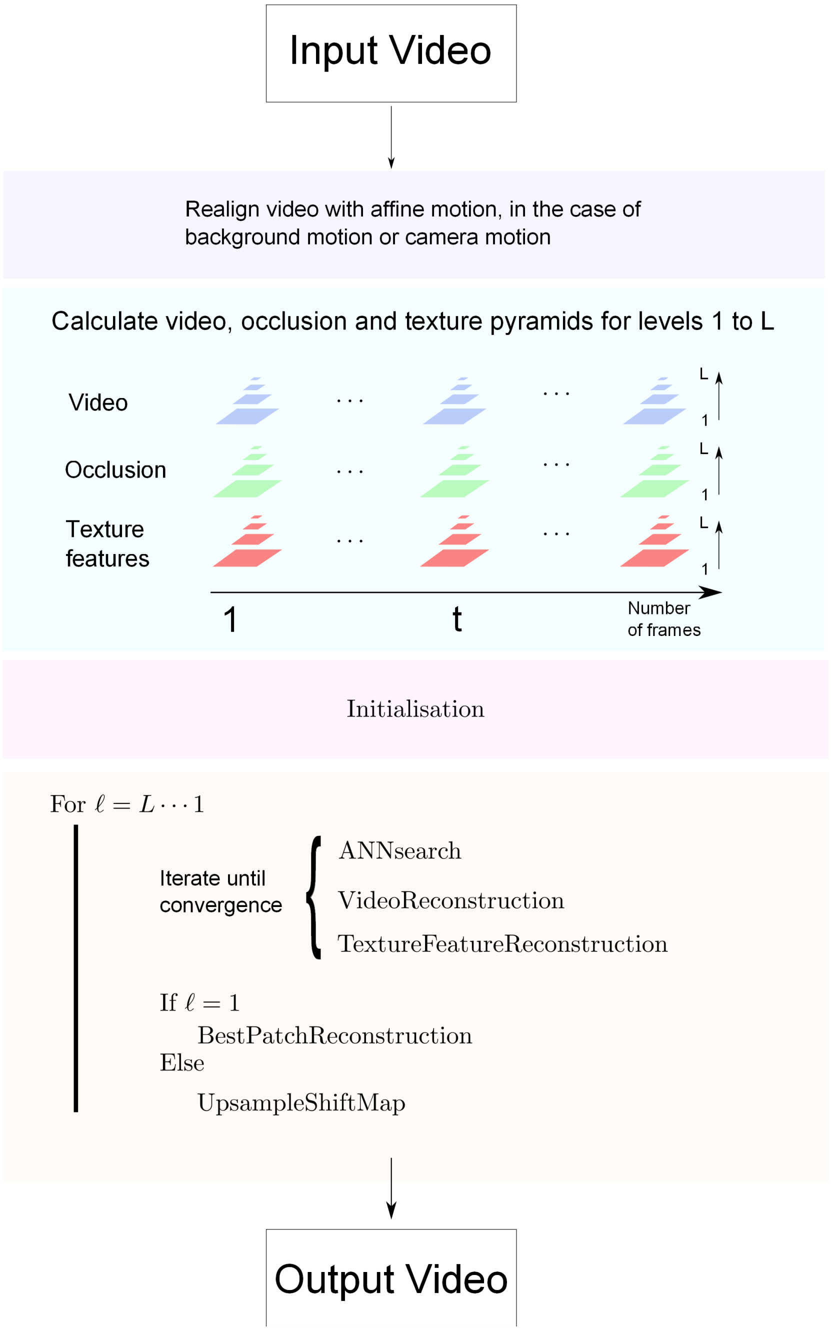


Figure 1. *Diagram of the proposed video inpainting algorithm.*

another position $q \in \tilde{\mathcal{D}}$, such that patches W_p^u and W_q^u are as similar as possible. This can be formalized using the so-called *shift map* $\phi : \Omega \rightarrow \mathbb{R}^3$ that captures the shift between a position and its correspondent, that is $q = p + \phi(p)$ is the "correspondent" of p. This map must verify that $p + \phi(p) \in \tilde{\mathcal{D}},\ \forall p$ (see Figure 2 for an illustration).

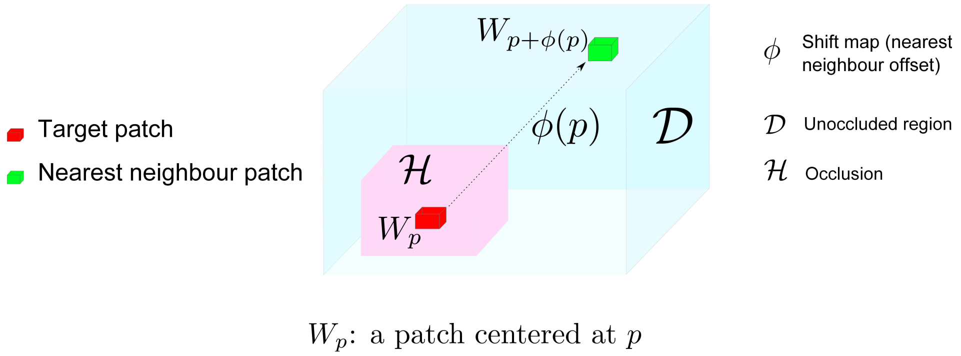


Figure 2. *Illustration of the notation used for proposed video inpainting algorithm.*

Mimizing a non-local patch-based functional. The cost function which we use, following the work of Wexler *et al.* [38], has both u and ϕ as arguments:

$$E(u, \phi) = \sum_{p \in \mathcal{H}} d^2(W_p^u, W_{p+\phi(p)}^u), \tag{2.1}$$

with

$$d^2(W_p^u, W_{p+\phi(p)}^u) = \frac{1}{N} \sum_{q \in \mathcal{N}_p} \| u(q) - u(q + \phi(p)) \|_2^2. \tag{2.2}$$

In all that follows, in order to avoid cumbersome notation we shall drop the u from W_p^u and simply denote patches as W_p.

We show in Appendix A, that this functional is in fact a special case of the formulation of Arias *et al.* [3]. As mentioned at the beginning of this Section, this functional is optimised using the following two steps:

Matching Given current video u, find in $\tilde{\mathcal{D}}$ the *nearest neighbour* (NN) of each patch W_p that has pixels in inpainting domain $\mathcal{H}$, that is, the map $\phi(p),\ \forall p \in \Omega \setminus \tilde{\mathcal{D}}$.

Reconstruction Given shift map ϕ, attribute a new value $u(p)$ to each pixel $p \in \mathcal{H}$.

These steps are iterated so as to converge to a satisfactory solution. The process may be seen as an alternated minimisation of cost (2.1) over the shift map ϕ and the video content u. As in many image processing and computer vision problems, this approach is implemented in a multi-resolution framework in order to improve results and avoid local minima.

Now that we have given a general outline of our algorithm, we proceed to address some of the key challenges in video inpainting. The first of these concerns the search for the nearest neighbours of patches centred on pixels which need to be inpainted.

2.1. Approximate Nearest Neighbour (ANN) search. When considering the high complexity of the NN search step, it quickly becomes apparent that searching for exact nearest neighbours would take far too long. Therefore, an *approximate nearest neighbour* (ANN) search is carried out. Wexler *et al.* proposed the k-d tree based approach of Arya and Mount [4] for this step, but this approach remains quite slow. For example, one ANN search step takes about an hour for a video containing $120 \times 340 \times 100$ pixels, with about $422,000$ missing pixels, which represents a relatively small occlusion (the equivalent of a 65×65 pixel box in each frame). We shall address this problem here, in particular by using an extension of the PatchMatch algorithm [5] to the spatio-temporal case. We note that the PatchMatch algorithm has also been used in conjunction with a 2D version of Wexler's algorithm for *image* inpainting, in the Content-Aware Fill tool of Photoshop [1], and by Darabi *et al.* [14].

Barnes *et al.*'s PatchMatch is a conceptually simple algorithm based on the hypothesis that, in the case of image patches, the shift map defined by the spatial offsets between ANNs is piece-wise constant. This is essentially because the image elements which the ANNs connect are often on rigid objects of a certain size. In essence, the algorithm looks randomly for ANNs and tries to "spread" those which are good. We extend this principle to the spatio-temporal setting. Our spatio-temporal extension of the PatchMatch algorithm consists of three steps: (i) initialisation,(ii) propagation and (iii) random search.

Let us recall that $\tilde{\mathcal{H}}$ is a dilated version of $\mathcal{H}$. Initialisation consists of randomly associating an ANN to each patch W_p, $p \in \tilde{\mathcal{H}}$, which gives an initial ANN shift map, ϕ. In fact, apart from the first iteration, we already have a good initialisation: the shift map ϕ from the previous iteration. Therefore, except during the initialisation step (see Section 3.2), we use this previous shift map in our algorithm instead of initialising randomly.

The propagation step encourages shifts in ϕ which lead to good ANNs to be spread throughout ϕ. In this step, all positions in the video volume are scanned lexicographically. For a given patch W_p at location $p = (x, y, t)$, the algorithm considers the following three candidates : $W_{p+\phi(x-1,y,t)}$, $W_{p+\phi(x,y-1,t)}$ and $W_{p+\phi(x,y,t-1)}$. If one of these three patches has a smaller patch distance with respect to W_p than $W_{p+\phi(p)}$, then $\phi(p)$ is replaced with the new, better shift. The scanning order is reversed for the next iteration of the propagation, and the algorithm tests $W_{p+\phi(x+1,y,t)}$, $W_{(p+\phi(x,y+1,t)}$ and $W_{p+\phi(x,y,t+1)}$. In the two different scanning orderings, the important point is obviously to use the patches which have already been processed in the current propagation step.

The third step, the random search, consists in looking randomly for better ANNs of each W_p in an increasingly small area around $p + \phi(p)$, starting with a maximum search distance. At iteration k, the random candidates are centred at the following positions:

$$q = p + \phi(p) + \lfloor r_{\max} \rho^k \delta_k \rfloor, \tag{2.3}$$

where $r_{\max}$ is the maximum search radius around $p + \phi(p)$, δ_k is a 3-dimensional vector drawn from the uniform distribution over unit cube $[-1, 1] \times [-1, 1] \times [-1, 1]$ and $\rho \in (0, 1)$ is the reduction factor of the search window size. In the original PatchMatch, ρ is set to 0.5. This random search avoids the algorithm getting stuck in local minima. The maximum search parameter $r_{\max}$ is set to the maximum dimension of the video, at the current resolution level.

The propagation and random search steps are iterated several times to converge to a good solution. In our work, we set this number of iterations to 10. For further details concerning the

PatchMatch algorithm in the 2D case, see [5]. Our spatio-temporal extension is summarized in Algorithm 1.

Algorithm 1: ANN search with 3D PatchMatch

```
Data: Current inpainting configuration $u$, $T$, $\phi$, $\mathcal{H}$
Result: ANN shift map $\phi$
$\tilde{\mathcal{H}} \leftarrow$ DilateOcclusion($\mathcal{H}$);
for $k = 1$ to 10 do
    for $p = p_1$to $p_{|\tilde{\mathcal{H}}|}$ (pixels in $\tilde{\mathcal{H}}$ lexicographically ordered) do
        $a = p - (1,0,0)$, $b = p - (0,1,0)$, $c = p - (0,0,1)$;
        $q = \arg\min_{r \in \{p,a,b,c\}} d(W_p^u, W_{p+\phi(r)}^u)$;
        if $p + \phi(q) \in \tilde{\mathcal{D}}$ then $\phi(p) \leftarrow \phi(q)$;
    end
    for $p = p_{|\tilde{\mathcal{H}}|}$ to $p_1$ do
        $a = p + (1,0,0)$, $b = p + (0,1,0)$, $c = p + (0,0,1)$;
        $q = \arg\min_{r \in \{p,a,b,c\}} d(W_p^u, W_{p+\phi(r)}^u)$;
        if $p + \phi(q) \in \tilde{\mathcal{D}}$ then $\phi(p) \leftarrow \phi(q)$;
    end
    for $p = p_1$ to $p_{|\tilde{\mathcal{H}}|}$ do
        $q = p + \phi(p) + \lfloor r_{\max}\rho^k \mathrm{RandUniform}([-1,1]^3) \rfloor$;
        if $d(W_p^u, W_{p+\phi(q)}^u) < d(W_p^u, W_{p+\phi(p)}^u)$ and $p + \phi(q) \in \tilde{\mathcal{D}}$ then $\phi(p) \leftarrow \phi(q)$;
    end
end
```

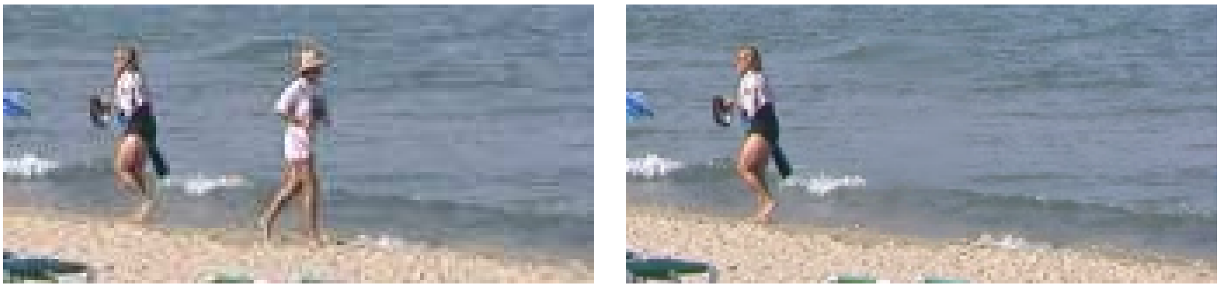
We note here that other ANN search methods for image patches exist which outperform PatchMatch [20, 25, 32]. However, in practice, PatchMatch appeared to be a good option because of its conceptual simplicity and nonetheless very good performance. Furthermore, to take the example of the “TreeCANN” method of Olonetsky and Avidan [32], the reported reduction in execution time is largely based on a very good ANN shift map initialisation followed by a small number of propagation steps. In our case, we already have a good initialisation (from the previous iteration), which makes the usefulness of such approaches questionable. However, further acceleration is certainly something which could be developed in the future.

2.2. Video reconstruction. Concerning the reconstruction step, we use a a weighted mean based approach, inspired by the work of Wexler *et al.*, in which each pixel is reconstructed in the following manner:

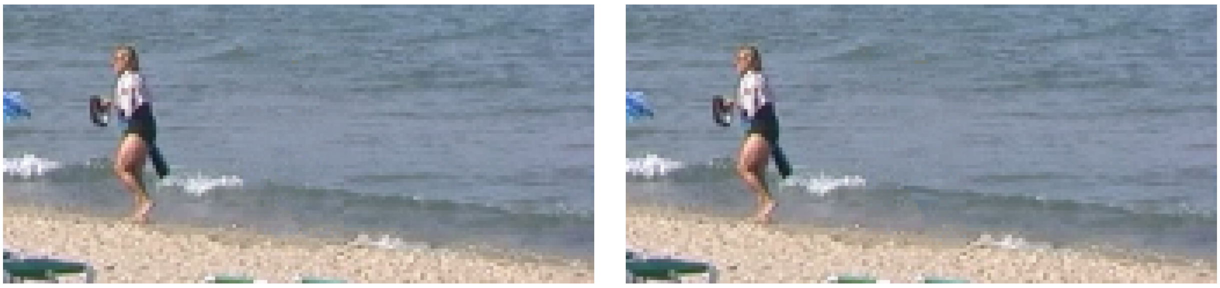
$$u(p) = \frac{\sum_{q \in \mathcal{N}_p} s_p^q \, u(p + \phi(q))}{\sum_{q \in \mathcal{N}_p} s_p^q}, \quad \forall p \in \mathcal{H}, \tag{2.4}$$

with

$$s_p^q = \exp\left(-\frac{d^2(W_q, W_{q+\phi(q)})}{2\sigma_p^2} \right). \tag{2.5}$$



Original frame : “Crossing ladies” [38]   Weighted average



Mean shift   Best patch

Figure 3. Comparison of different final reconstruction methods. *We observe that the proposed reconstruction using only the best patch at the end of the algorithm produces similar results to the use of the mean shift algorithm, avoiding blur induced by weighted patch averaging, while being less computationally expensive. Please note that the blurring effect is best viewed in the pdf version of the paper.*

Wexler *et al.* proposed the use of an additional weighting term to give more weight to the information near the occlusion border. We dispense with this term, since in our scheme it is somewhat replaced by our method of initialising the solution which will be detailed in Section 3.2. Parameter σ_p is defined as the 75th percentile of all distances $\{d(W_q, W_{q+\phi(q)}),\ q \in \mathcal{N}_p\}$ as in [38].

Observe that in order to minimise (2.1), the natural approach would be to do the reconstruction with the non-weighted scheme ($s_p^q = 1$ in Equation 2.4) that stems from $\frac{\partial E}{\partial u(p)} = 0$. However, the weighted scheme above tends to accelerate the convergence of the algorithm, meaning that we produce good results faster.

An important observation is that, in the case of regions with high frequency details, the use of this mean reconstruction (weighted or unweighted) often leads to blurry results, even if the correct patches have been identified. This phenomenon was also noted in [38]. Although we shall propose in Section 2.3 a method to correctly identify textured patches in the matching steps, this does not deal with the *reconstruction* of the video. We thus need to address this problem, at least in the final stage of the approach: throughout the algorithm, we use the unweighted mean given in Equation 2.4 and, at the end of the algorithm, when the solution has converged at the finest pyramid level, we simply inpaint the occlusion using the best patch among the contributing ones. This corresponds to setting σ_p to 0 in (2.4-2.5) or, seen in another light, may be viewed as a very crude annealing procedure. Final reconstruction at

Original image Image melding [14] Liu and Caselles [28] Proposed approach

Approximate nearest neighbours without texture features

Approximate nearest neighbours with texture features

Figure 4. **Illustration of the necessity of texture features for inpainting**. *Without the texture features, the correct texture may not be found. In this case, the algorithm mistakes the hat for the correct zone, since the hat has a similar average colour to the beard, over the size of a patch.*

position $p \in \mathcal{H}$ reads:

$$u^{(\text{final})}(p) = u(p + \phi^{(\text{final})}(q^*)), \quad \text{with } q^* = \arg\min_{q \in \mathcal{N}_p} d(W_q^{(\text{final}-1)}, W_{q+\phi^{(\text{final})}(q)}). \tag{2.6}$$

Another solution to this problem based on the mean shift algorithm was proposed by Wexler *et al.* in [38], but such an approach increases the complexity and execution time of the algorithm. Figure 3 shows that very similar results to those in [38] may be obtained with our much simpler approach.

2.3. Video texture pyramid. In order for any patch-based inpainting algorithm to work, it is necessary that the patch distance identify “correct” patches. This is not the case in several situations. Firstly, as noticed by Liu and Caselles in [28], the use of multi-resolution pyramids can make patch comparisons ambiguous, especially in the case of textures, in images and videos. Secondly, it turns out that the commonly used ℓ^2 patch distance is ill-adapted to comparing textured patches. Thirdly, PatchMatch itself can contribute to the identification of incorrect patches. These reasons are explored and explained more extensively in Appendix B. A visual illustration of the problem may be seen in Figure 4. We note here that similar ambiguities were also identified by Bugeau *et al.* in [11], but their interpretation of the phenomenon was somewhat different.

We propose a solution to this problem in the form of a multi-resolution pyramid which reflects the textural nature of the video. We shall refer to this as *the texture feature pyramid*. The information in this texture feature pyramid is added to the patch distance in order to identify the correct patches.

In order to identify textures, we shall consider a simple, gradient-based texture attribute. Following Liu and Caselles [28], we consider the absolute value of the image derivatives, averaged over a certain spatial neighbourhood ν. Obviously, many other attributes could be considered, however they seemed too involved for our purposes.

More formally, we introduce the two-dimensional texture feature $T = (T_x, T_y)$, computed at each pixel $p \in \Omega$:

$$T(p) = \frac{1}{\text{card}(\nu)} \sum_{q \in \nu} (|I_x(q)|, |I_y(q)|), \tag{2.7}$$

where $I_x(q)$ (resp. $I_y(q)$) is the derivative of the image intensity (grey-level) in the x (resp. y) direction at the pixel q. The squared patch distance is now defined as

$$d^2(W_p, W_q) = \frac{1}{N} \sum_{r \in \mathcal{N}_p} \Big(\|u(r) - u(r-p+q))\|_2^2 + \lambda \|T(r) - T(r-p+q)\|_2^2 \Big), \tag{2.8}$$

where λ is a weighting scalar.

The feature pyramid is then set up by subsampling the texture features of the full-resolution input video u. We note here that each level is obtained by subsampling the information contained at the *finest pyramid resolution*, and *not* by calculating T^ℓ based on the subsampled video at the level ℓ:

$$\forall (x, y, t) \in \Omega^\ell, \; T^\ell(x, y, t) = T(2^\ell x, 2^\ell y, t), \quad \ell = 1 \cdots L. \tag{2.9}$$

This is an important point, since the required textural information *does not exist* at coarser levels. Features are not filtered before subsampling, since they have already been averaged over the neighbourhood ν. In the experiments done in this paper, this neighbourhood is set, by default, to the area to which a coarsest-level pixel corresponds, which is a square of size 2^{L-1}, as is done in [28]. However, in a more general setting, the size of this area should be independent of the number of levels, so care should be taken in the case where few pyramid levels are used.

A notable difference with respect to the work of Liu and Caselles [28] is the fact that we use the texture features at *all* pyramid levels. Liu and Caselles do not do this, since they perform graph cut based optimisation at the coarsest level, and at the finer levels only consider small relative shifts with respect to the coarse solution.

A final choice which must be made when using the texture features is how they are *themselves* reconstructed. In shift maps based algorithms, this is not a problem, since by definition an occluded pixel takes on the characteristics of its correspondent in $\mathcal{D}$ (colour, texture features or anything else).

In our case, we inpaint the texture features using the same reconstruction scheme as is used for colour information (see Eq. 2.4):

$$T(p) = \frac{\sum_{q \in \mathcal{N}_p} s_p^q T(p + \phi(q))}{\sum_{q \in \mathcal{N}_p} s_p^q}, \ \forall p \in \mathcal{H}. \tag{2.10}$$

Conceptually, the use of these features is quite simple, and easily fits into our inpainting framework. To summarise, these features may be seen as simple texture descriptors which help the algorithm avoid making mistakes when choosing the area to use for inpainting.

The question naturally arises of why this problem has not been more discussed in the inpainting literature. Indeed, to the best of our knowledge, only Bugeau *et al.* [11] and Liu and Caselles [28] have clearly identified this problem in the case of image inpainting. This is due to the fact that most other inpainting algorithms *restrict the ANN search space* to a local neighbourhood around the occlusion. Unfortunately, this restriction principle does not hold in video inpainting since the information can be found anywhere in the video volume, in particular when a complex movement must be reconstructed. Here, the introduction of the texture feature pyramids leads to greatly improved patch comparisons, meaning that we can use the whole video as the search space.

The methodology which we have proposed for dealing with dynamic video textures is important for the following reasons. Firstly, to the best of our knowledge, this is the first inpainting approach which proposes a global optimisation and which can deal correctly with textures in images and videos, without restricting the search space (contrary to [13, 14, 19, 28, 35]). Secondly, while the problem of recreating video textures is a research subject in its own right and algorithms have been developed for their synthesis [26, 36], ours is the first algorithm to achieve this within an inpainting framework. Finally we note that algorithms such as that of Granados *et al.* [18], which are specifically dedicated to background reconstruction, cannot deal with background video textures (such as waves), since they suppose that the background is rigid. This hypothesis is clearly not true for video textures. An example of the impact of the texture features in video inpainting may be seen in Figure 7.

3. Additional components.

3.1. Inpainting with mobile background.

We now turn to another common case of video inpainting, that of mobile backgrounds. This is the case, for example, when hand-held cameras are used to capture the input video.

There are several possible solutions to this problem. Patwardhan *et al.* [34] segment the video into moving foreground and background (which may also display motion) using motion estimation with block matching. Once the moving foreground is inpainted, the background is realigned with respect to a motion estimated with block matching, and the background is filled by copying and pasting background pixels.

Granados *et al.* [18] propose a homography-based algorithm for this task. They estimate a set of homographies between each frame and choose which homography should be used for each occluded pixel belonging to the background.

Both of these algorithms require that the background and foreground be segmented, which we wish to avoid here. Furthermore, they have the quite strict limitation that pixels are simply copied from their realigned positions, meaning that the realignment must be extremely accurate. Here, we propose a solution which allows us to use our patch-based variational

Original frame : “Jumping man” [18]

Inpainting without realignement

Inpainting with realignement

Figure 5. **A comparison of inpainting results with and without affine realignment**. *Notice the incoherent reconstruction on the steps, due to random camera motion which makes spatio-temporal patches difficult to compare. These random motions are corrected with affine motion estimation and inpainting is performed in the realigned domain.*

framework for both tasks (foreground and background inpainting) simultaneously, without any segmentation of the video into foreground and background.

Algorithm 2: Pre-processing step for realigning the input video.

```
Data: Input video $u$
Result: Aligned video and affine warps
$N_f$ = number of frames in video;
$N_m = \lfloor \frac{N_f}{2} \rfloor$;
for $n = 1$ to $N_f - 1$ do
    $\theta_{n,n+1} \leftarrow$ EstimateAffineMotion($u_n, u_{n+1}$);
end
for $n = 1$ to $N_f$ do
    if $n < N_m$ then $\theta_{n,N_m} = \theta_{N_m-1,N_m} \circ \cdots \circ \theta_{n,n+1}$;
    if $n > N_m$ then $\theta_{n,N_m} = \theta^{-1}_{N_m-1,N_m} \circ \cdots \circ \theta^{-1}_{n,n+1}$;
    $u_n \leftarrow$ AffineWrap($u_n, \theta_{n,N_m}$);
end
```

The fundamental hypothesis behind patch-based methods in images or videos is that content is redundant and repetitive. This is easy to see in images, and may appear to be the case in videos. However, the temporal dimension is added in video patches, meaning that a *sequence* of image patches should be repeated throughout the video. This is not the case when a video displays random motion (as with a mobile camera): even if the required content appears at some point in the sequence, there is no guarantee that the required spatio-temporal patches will repeat themselves with the same motion. Empirically, we have observed that this is a significant problem even in the case of motions with small amplitude.

To counter this problem, we estimate a dominant, affine motion between each pair of

(a) Occluded input image

(b) Random initialisation

(c) Smooth zero-Laplacian interpolation

(d) Onion peel initialisation

(a) Original image

(b) Inpainting after random initialisation

(c) Inpainting after zero Laplacian

(d) Inpainting after onion peel initialisation

Figure 6. **Impact of initialisation schemes on inpainting results**. *Note that the only scheme which successfully reconstructs the cardboard tube is the "onion peel" approach where first filling at the coarsest resolution is conducted in a greedy fashion.*

successive frames, and use this to realign each frame with respect to one reference frame. In our work, we chose the reference frame to be the middle frame of the sequence (this should be adapted for larger sequences). We use the work of Odobez and Bouthemy [31] to realign the frames. The occlusion $\mathcal{H}$ is obviously also realigned. Once the frames are realigned with the reference frame (Alg. 2), we inpaint the video as usual. Finally, when inpainting is finished, we perform the inverse affine transformation on the images and paste the solution into the original occluded area. Figure 5 compares the results with and without this pre-processing step, on a prototypical example. Without it, it is not possible to find coherent patches which respect the border conditions.

3.2. Initialisation of the solution. The iterative procedure at the heart of our algorithm relies on an initial inpainting solution. The initialisation step is very often left unspecified in work on video inpainting. As we shall see in this Section, it plays a vital role, and we therefore explain our chosen initialisation method in detail.

We inpaint at the coarsest level using an "onion peel" approach, that is to say we inpaint one layer of the occlusion at a time, each layer being one pixel thick.

More formally, let $\mathcal{H}' \subset \mathcal{H}$ be the current occlusion, and $\partial\mathcal{H}' \subset \mathcal{H}'$ the current layer to inpaint. We define the *unoccluded* neighbourhood $\mathcal{N}'_p$ of a pixel p, with respect to the current occlusion $\mathcal{H}'$ as:

$$\mathcal{N}'_p = \{q \in \mathcal{N}_p, q \notin \mathcal{H}'\}. \tag{3.1}$$

Some choices are needed to implement this initialisation method. First of all, we only compare the unoccluded pixels during a patch comparison. The distance between two patches W_p and $W_{p+\phi(p)}$ is therefore redefined as:

$$d^2(W_p, W_{p+\phi(p)}) = \frac{1}{|\mathcal{N}'_p|} \sum_{q \in \mathcal{N}'_p} \Big(\|u(q) - u(q+\phi(p))\|_2^2 + \lambda \|T(q) - T(q+\phi(p))\|_2^2 \Big). \tag{3.2}$$

We also need to choose which neighbouring patches to use for reconstruction. Some will be quite unreliable, as only a small part of the patches are compared. In our implementation, we only use the ANNs of patches whose centres are located outside the current occlusion layer. Formally, we reconstruct the pixels in the current layer by using the following formula, modified from Equation 2.4:

$$u_p = \frac{\sum_{q \in \mathcal{N}'_p} s_p^q u(p+\phi(q))}{\sum_{q \in \mathcal{N}'_p} s_p^q}. \tag{3.3}$$

The same reconstruction is applied to the texture features. A pseudo-code for the initialisation procedure may be seen in Algorithm 3.

Algorithm 3: Inpainting initialisation

```
Data: Coarse level inputs $u^L$, $T^L$, $\phi^L$, $\mathcal{H}^L$
Result: Coarse initial filling $u^L$, $T^L$ and map $\phi^L$
$B \leftarrow 3 \times 3 \times 3$ structuring element;
$\mathcal{H}' \leftarrow \mathcal{H}^L$;
while $\mathcal{H}' \neq \emptyset$ do
    $\partial\mathcal{H}' \leftarrow \mathcal{H}' \setminus$ Erosion($\mathcal{H}', B$);
    $\phi^L \leftarrow$ ANNsearch($u^L, T^L, \phi^L, \partial\mathcal{H}'$) ;   // Alg.1, with partial distance (3.2)
    $u^L \leftarrow$ Reconstruction($u^L, \phi^L, \partial\mathcal{H}'$) ;   // Eq.3.3
    $T^L \leftarrow$ Reconstruction($T^L, \phi^L, \partial\mathcal{H}'$) ;   // Eq.3.3
    $\mathcal{H}' \leftarrow$ Erosion($\mathcal{H}', B$);
end
```

Figure 6 shows some evidence to support a careful initialisation scheme. Three different initialisations have been tested: random initialisation, zero-Laplacian interpolation and onion peel. Random initialisation is achieved by initialising the occlusion with pixels chosen randomly from the image. Zero Laplacian (harmonic) interpolation is the solution of the Laplace equation $\Delta u = 0$ with Dirichlet boundary conditions stemming from $\mathcal{D}$. It may be seen (Figure 6) that the first two initialisations are unable to join the two parts of the cardboard tube together, and that the subsequent iterations do not improve the situation. In contrast, the proposed initialisation produces a satisfactory result.

3.3. Other important algorithmic details. In order to make our method as easy as possible to reimplement, we now present some further algorithmic details, which are in fact very important for achieving good results.

Our first remarks concern the implementation of the multi-resolution pyramids. Wexler *et al.* and Granados *et al.* both note that *temporal* subsampling can be detrimental to inpainting results. This is due to the difficulty of representing motion at coarser levels. For this reason we do not subsample in the temporal direction, as in [19]. The only case where we need to temporally subsample is when the objects spend a long time behind the occlusion, (this was only done in the "Jumping girl" sequence). This is quite a hard problem to solve since it becomes increasingly difficult to decide what motion an occluded object should have when the occlusion time grows longer, unless there is strictly periodic motion. We leave this as an open question, which could be investigated in further work.

A crucial choice when using multi-resolution schemes is the number of pyramid levels to use. Most other methods leave this parameter unspecified, or fix the size of the image/video at the coarsest scale, and determine the resulting number of levels [19, 28, 35]. In fact, when one considers the problem in more detail, it becomes apparent that the number of levels should be set so that the occlusion size is not too large in comparison to the patch size. This intuition is supported by experiments in very simple image inpainting situations, which showed that the occlusion size should be somewhat less than *twice* the patch size. In our experiments, we follow this general rule of thumb.


Original frame : "Waves" Inpainting without features Inpainting with features

Figure 7. Usefulness of the proposed texture features. *Without the features, the algorithm fails to recreate correctly the waves, which is a typical example of complex video texture.*

Another question which is of interest is how to pass from one pyramid level to another. Wexler *et al.* presented quite an intricate scheme in [38] to do this, whereas Granados *et al.* propose a simple upsampling of the shift map. This is conceptually simpler than the approach of Wexler *et al.* and, after experimentation, we chose this option as well. Therefore, the shift map ϕ is upsampled using nearest neighbours interpolation, and both the higher resolution video and the higher resolution texture features are reconstructed using Equation 2.4. One final note on this point is that we use the upsampled version of ϕ as an initialisation for the PatchMatch algorithm at each level apart from the coarsest (this differs to our previous work in [30]).

We also require a threshold which will stop the iterations of the ANN search and reconstruction steps. In our work, we use the average colour difference in each channel per pixel between iterations as a stopping criterion. If this falls below a certain threshold, we stop the iteration at the current level. We set this threshold to 0.1. In order to avoid iterating for too long, so we also impose a maximum number of 20 iterations at any given pyramid level.

The patch size parameters were set to $5 \times 5 \times 5$ in all of our experiments. We set the texture feature parameter λ to 50. Concerning the spatio-temporal PatchMatch, we use ten iterations of propagation/random search during the ANN search algorithm and set the window size reduction factor β to 0.5 (as in the paper of Barnes *et al.* [5]).

The complete algorithm is summarized in Alg. 4.

Original frame : “Girl” [18] Result from [18] Our result

Figure 8. A comparison of our inpainting result with the that of the background inpainting algorithm of Granados *et al.* [18]. *In such cases with moving background, we are able to achieve high quality results (as do Granados* et al.*), but we do this in one, unified algorithm. This illustrates the capacity of our algorithm to perform well in a wide range of inpainting situations.*

4. Experimental results.

The goal of our work is to achieve high quality inpainting results in varied, complex video inpainting situations, with reduced execution time. Therefore, we shall evaluate our results in terms of visual quality and execution time.

We compare our work to that of Wexler *et al.* [38] and to the most recent video inpainting method of Granados *et al.* [19]. All of the videos in this paper (and more) can be viewed and downloaded along with occlusion masks at `http://www.enst.fr/~gousseau/video_inpainting`.

4.1. Visual evaluations.

First of all, we have tested our algorithm on the videos proposed by Wexler *et al.* [38] and Granados *et al* [19]. The visual results of our algorithm may be seen in Figures 9 and 10. We note that the inpainting results of the previous authors on these examples are visually almost perfect, so very little qualitative improvement can be made. It may be seen that our results are of similarly high quality to the those of the previous algorithms. In particular we are able to deal with situations where several moving objects must be correctly recreated, without requiring manual segmentation as in [19]. We also achieve these results in at least an order of magnitude less time than the previous algorithms. We note that it is not feasible to apply the method of Wexler *et al.* to the examples of [19], whose resolution is too large (up to 1120×754×200 pixels).

Algorithm 4: Proposed video inpainting algorithm.

```
Data: Input video $u$ over $\Omega$, occlusion $\mathcal{H}$, resolution number $L$
Result: Inpainted video
$(u, \theta) \leftarrow$AlignVideo$(u)$ ;                                        // Alg.2
$\{u^\ell\}_{\ell=1}^L \leftarrow$ImagePyramid$(u)$;
$\{T^\ell\}_{\ell=1}^L \leftarrow$TextureFeaturePyramid$(u)$;                      // Eqs.2.7-2.9
$\{\mathcal{H}^\ell\}_{\ell=1}^L \leftarrow$OcclusionPyramid$(\mathcal{H})$;
$\phi^L \leftarrow$Random;
$(u^L, T^L, \phi^L) \leftarrow$Initialisation$(u^L, T^L, \phi^L, \mathcal{H}^L)$;   // Alg.3
for $\ell = L$ to 1 do
    $k = 0$, $e = 1$;
    while $e > 0.1$ and $k < 20$ do
        $v = u^\ell$;
        if $\ell < L$ and $k = 0$ then $\phi^\ell \leftarrow$UpSample$(\phi^{\ell+1}, 2)$;
        $\phi^\ell \leftarrow$ANNsearch$(u^\ell, T^\ell, \phi^\ell, \mathcal{H}^\ell)$ ;        // Alg.1
        $u^\ell \leftarrow$Reconstruction$(u^\ell, \phi^\ell, \mathcal{H}^\ell)$ ;             // Eqs.2.4-2.5
        $T^\ell \leftarrow$Reconstruction$(T^\ell, \phi^\ell, \mathcal{H}^\ell)$;
        $e = \frac{1}{3|\mathcal{H}^\ell|} \| u^\ell_{\mathcal{H}^\ell} - v_{\mathcal{H}^\ell} \|_2$;
        $k \leftarrow k + 1$;
    end
    if $\ell = 1$ then $u \leftarrow$FinalReconstruction$(u^1, \phi^1, \mathcal{H})$ ;  // Eq.2.6
end
$u \leftarrow$UnwarpVideo$(u, \theta)$
```

Next, we provide experimental evidence to show the ability of our algorithm to deal with various situations which appear frequently in real videos, some of which are not dealt with by previous methods. Figure 7 shows an example of the utility of using texture features in the inpainting process: without them, the inpainting result is quite clearly unsatisfactory. We have not directly compared these results with previous work. However it is quite clear that the method of [19] cannot deal with such situations. This method supposes that the background is static, and in the case of dynamic textures, it is not possible to restrict the search space as proposed in the same method for moving objects. Furthermore, the background inpainting algorithm of Granados *et al.* [18] supposes that moving background undergoes a homographic transformation, which is clearly not the case for video textures. By relying on a plain colour distance between patches, the algorithm of Wexler *et al.* is likely to produce results similar to the one which may be seen in Figure 7 (middle image). Finally, to take another algorithm of the literature, the method of Patwardhan *et al.* [34] would encounter the same problems as that of [18], since they copy-and-paste pixels directly after compensating for a locally estimated motion. More examples of videos containing dynamic textures can be seen at `http://www.enst.fr/~gousseau/video_inpainting`.

Our algorithm's capacity to deal with moving background is illustrated by Figure 8. We

"Beach Umbrella"

"Crossing Ladies"

"Jumping girl"

Original frames Inpainting result from [38] Our inpainting result

Figure 9. Comparison with Wexler *et al.* *We achieve results of similar visual quality compared to those in [38], with a reduction of the ANN search time by a factor of up to 50 times.*

do this in the same unified framework used for all other examples in this paper, whereas a specific algorithm is needed by Granados *et al.* [18] to achieve this. Thus, we see that the same core algorithm (iterative ANN search and reconstruction) can be used in order to deal with a series of inpainting tasks and situations. Furthermore, we note that no foreground/background segmentation was needed for our algorithm to produce satisfactory results. Finally, we note that such situations are not managed using the algorithm of [38]. Again, examples containing moving backgrounds can be viewed at the referenced website.

The generic nature of the proposed approach represents a significant advantage over previous methods, and allows us to deal with many different situations without having to resort to manual intervention or the creation of specific algorithms.

4.2. Execution times. One of the goals of our work was to accelerate the video inpainting task, since this was previously the greatest barrier to development in this area. Therefore, we compare our execution times to those of [38] and [19] in Table 1.

Comparisons with Wexler's algorithm should be obtained carefully, since several crucial parameters are not specified. In particular, the ANN search scheme used by Wexler *et al.* requires a parameter, ε, which determines the accuracy of the ANNs. More formally, if W_p is a source patch, W_q is the exact NN of W_p and W_r is an ANN of W_p, then the work of [4] guarantees that $d(W_p, W_r) \leq (1+\varepsilon)d(W_p, W_q)$. This parameter has a large influence on the computational times. For our comparisons, we set this parameter to 10, which produced

Original frame : “Duo” Original frame : “Museum”

Inpainting result from [19]

Our inpainting result

Figure 10. Comparison with Granados *et al.* *We achieve similar results to those of [19] in an order of magnitude less time, without user intervention. The occlusion masks are highlighted in green.*

Algorithm	ANN execution times for all occluded pixels at full resolution.				
	Beach Umbrella	Crossing Ladies	Jumping Girl	Duo	Museum
	$264 \times 68 \times 200$	$170 \times 80 \times 87$	$300 \times 100 \times 239$	$960 \times 704 \times 154$	$1120 \times 754 \times 200$
Wexler (kdTrees)	985 s	942 s	7877 s	-	-
Ours (3D PatchMatch)	50 s	28 s	155 s	29 min	44 min

Algorithm	Total execution time				
Granados	11 hours	-	-	-	90.0 hours
Ours	24 mins	15 mins	40 mins	5.35 hours	6.2 hours
Ours w/o texture	14 mins	12 mins	35 mins	4.07 hours	4.0 hours

Table 1

Partial and total execution times on different examples. *The partial inpainting times represent the time taken for the ANN search for all occluded patches at the full resolution. Note that for the "museum" example, Granados's algorithm is parallelised over the different occluded objects and the background, whereas ours is not.*

ANNs with a similar average error per patch component as our spatio-temporal PatchMatch. Another parameter which is left unspecified by Wexler *et al.* is the number of iterations of ANN search/reconstruction steps per pyramid level. This has a very large influence on the total execution time. Therefore, instead of comparing total execution times we simply compare the ANN search times, as this step represents the majority of the computational load. We obtain a speedup of 20-50 times over the method of [4]. We also include our total execution times, to give a general idea of the time taken with respect to the video size. These results show a speedup of around an order of magnitude with respect to the semi-automatic methods of Granados *et al* [19]. In Table 1, we have also added our execution times without the use of texture features to illustrate the additional computational load which this adds.

These computation times show that our algorithm is clearly faster than the approaches of [38] and [19]. This advantage is significant because not only is the algorithm more practical to use, but it is also much easier to experiment and therefore make progress in the domain of video inpainting.

5. Further work. Several points could be improved upon in the present paper. Firstly, the case where a moving object is occluded for long periods remains very difficult and is not dealt with in a unified manner here. The one solution to this problem (temporal subsampling) does not perform well when complex motion is present. Therefore, other solutions could be interesting to explore. Secondly, we have observed that using a multi-resolution texture feature pyramid produces very interesting results. Therefore, we could perhaps enrich the patch space with other features, such as spatio-temporal gradients. Finally, it is acknowledged that videos of high resolutions still take quite a long time to process (up to several hours). Further acceleration could be achieved by dimensionality-reducing transformations of the patch space.

6. Conclusion. In this paper, we have proposed a non-local patch-based approach to video inpainting which produces good quality results in a wide range of situations, and on high definition videos, in a completely automatic manner. Our extension of the PatchMatch ANN search scheme to the spatio-temporal case reduces the time complexity of the algorithm, so that high definition videos can be processed. We have also introduced a texture feature pyramid which ensures that dynamic video textures are correctly inpainted. The case of mobile cameras and moving background is dealt with by using a global, affine estimation of the dominant motion in each frame.

The resulting algorithm performs well in a variety of situations, and does not require any manual input or segmentation. In particular, the specific problem of inpainting textures in videos has been addressed, leading to much more realistic results than other inpainting algorithms. Video inpainting has yet not been extensively used, in a large part due to prohibitive execution times and/or necessary manual input. We have directly addressed this problem in the present work. We hope that this algorithm will make video inpainting more accessible to a wider community, and help it to become a more common tool in various other domains, such as video post-production, restoration and personal video enhancement.

7. Acknowledgments. The authors would like to express their thanks to Miguel Granados for his kind help, for making his video content publicly available and for answering several questions concerning his work. The authors also thank Pablo Arias, Vicent Caselles and Gabriele Facciolo for numerous insightful discussions and for their help in understanding their inpainting method.

Appendix A. On the link between the non-local patch-based and shift map-based formulations.

In the inpainting literature, two of the main approaches which optimise a global objective functional include non-local patch-based methods [3, 38], which are closely linked to Non-Local Means denoising [10], and shift map-based formulations such as [19, 28, 35]. We will show here that the two formulations are closely linked, and in particular that the shift map formulation is a specific case of the very general formulation of Arias *et al.* [3, 2].

As far as possible we keep the notation introduced previously. In particular p is a position in $\mathcal{H}$, q is a position in $\mathcal{N}_p$ and r is a position in $\tilde{\mathcal{D}}$. In its most general version, the variational formulation of Arias *et al.* is not based on a one-to-one shift map, but rather on a positive weight function $w : \Omega \times \Omega \to \mathbb{R}^+$ that is constrained by $\sum_r w(p,r) = 1$ to be a probability distribution for each point p. Inpainting is cast as the minimisation of the following energy functional (that we rewrite here in its discretised version):

$$E_{\text{arias}}(u,w) = \sum_{p\in\mathcal{H}} \left[\sum_{r\in\tilde{\mathcal{D}}} w(p,r) d^2(W_p,W_r) + \gamma \sum_{r\in\tilde{\mathcal{D}}} w(p,r)\log w(p,r) \right], \tag{A.1}$$

with γ a positive parameter. The first terms is a “soft assignment” version of (2.1), while the second term is a regulariser that favors large entropy weight maps.

Arias *et al.* propose the following patch distance:

$$d^2(W_p,W_r) = \sum_{q\in\mathcal{N}_p} g_a(p-q)\varphi[u(q) - u(r+(p-q))], \tag{A.2}$$

where g_a is the centered Gaussian with standard deviation a (also called the intra-patch weight function), and φ is a squared norm. This is a very general and flexible formulation.

Arias *et al.* optimise this function using an alternate minimisation over u and w, and derive solutions for various patch distances. Let us choose the ℓ^2 distance for $d(W_p,W_r)$, as is the case in many inpainting formulations (and in particular the one which we use in our work). In this case, the minimisation scheme leads to the following expressions:

$$w(p,r) = \frac{1}{Z_p} \exp\left(-\frac{d^2(W_p, W_r)}{\gamma}\right), \tag{A.3}$$

$$u(p) = \sum_{q \in \mathcal{N}_p} g_a(p-q)\Big(\sum_{r \in \tilde{\mathcal{D}}} w(q,r)u(r+(p-q))\Big). \tag{A.4}$$

The parameter γ controls the selectivity of the weighting function w. In the limit case where $\gamma \to 0$, each weight function $w(p,.)$ must collapse to a single Dirac centered at a single match $p+\phi(p)$. If in addition, we consider that the intra-patch weighting is uniform, in other words $a = \infty$, the cost function E_{arias} reduces to:

$$E_{\text{arias}}(u,\phi) = \sum_{p \in \mathcal{H}} \sum_{q \in \mathcal{N}_p} \|u(q) - u(q+\phi(p))\|_2^2, \tag{A.5}$$

which is the formulation of Wexler *et al.* [38].

Rewriting (A.4) in the particular case just described, yields the optimal inpainted image

$$u(p) = \frac{1}{|\mathcal{N}_p|} \sum_{q \in \mathcal{N}_p} u(p+\phi(q)), \ \forall p \in \mathcal{H}, \tag{A.6}$$

as the (aggregated) average of the examples indicated by the NNs of the patches which contain p. Suppose that this aggregation can be reduced to a single shift-map

$$u(p) = u(p+\phi(p)), \ \forall p \in \mathcal{H}, \tag{A.7}$$

as is the case in the shift map-based formulations. Then the functional becomes[1]:

$$E_{\text{arias}}(u,\phi) = \sum_{p \in \mathcal{H}} \sum_{q \in \mathcal{N}_p} \|u(q+\phi(q)) - u(q+\phi(p))\|_2^2, \tag{A.8}$$

which effectively depends only upon ϕ.

If we look at the shift map formulation proposed by Pritch *et al.*, we find the following cost function over the shift map only:

$$E_{\text{pritch}}(\phi) = \sum_{p \in \mathcal{H}} \sum_{q \in \mathcal{N}_p} \Big(\|u(q) - u(q+\phi(p))\|_2^2 + \|\nabla u(q) - \nabla u(q+\phi(p))\|_2^2\Big). \tag{A.9}$$

Let us consider the first part, concerning the image colour values. Since we have $u(p) = u(p+\phi(p))$, we obtain again:

$$E_{\text{pritch}}(\phi) = \sum_{p \in \mathcal{H}} \sum_{q \in \mathcal{N}(p)} \|u(q+\phi(q)) - u(q+\phi(p))\|_2^2. \tag{A.10}$$

---

[1] We note that using the reconstruction of Equation A.7 poses problems on the occlusion border, but we ignore this here for the sake of simplicity and clarity.

Patch size	3×3	5×5	$3\times 3\times 3$	7×7	9×9	11×11	$5\times 5\times 5$
Probability	8×10^{-2}	10^{-2}	6×10^{-3}	4.1×10^{-4}	5.5×10^{-6}	3×10^{-7}	2×10^{-7}

Table 2

Probability of producing a random 2D or 3D patch that is closer to a random reference patch than to a constant one with same mean value. *Values are obtained through numerical simulations averaged over ten run for each experiment. Components of random patches are i.i.d. according to the centred normal law with a grey level variance of 25.*

Thus, the shift map cost function may be seen as a special case of the non-local patch-based formulation of Arias *et al.* under the following conditions : (i) $d^2(W_p, W_r) = \sum_{q\in\mathcal{N}_p} ||u(q) - u(r+(p-q))||_2^2$; (ii) $\gamma = 0$; (iii) The intra-patch weighting function g_a is uniform; (iv) $u(p) = u(p+\phi(p))$, that is, $u(p)$ is reconstructed using its correspondent according to a single shift map.

This short note on the link between the two formulations shows that the same functional is present in both, therefore it may be logical to expect similar results. However, one should keep in mind that they are certainly not equivalent, for reasons such as the difference in optimisation methodology, the presence of a gradient term in the formulation of Pritch *et al*, and the fact that the variational reconstruction (A.6) may be smoother than the shift-map reconstruction (A.7).

Appendix B. Comparing textured patches.

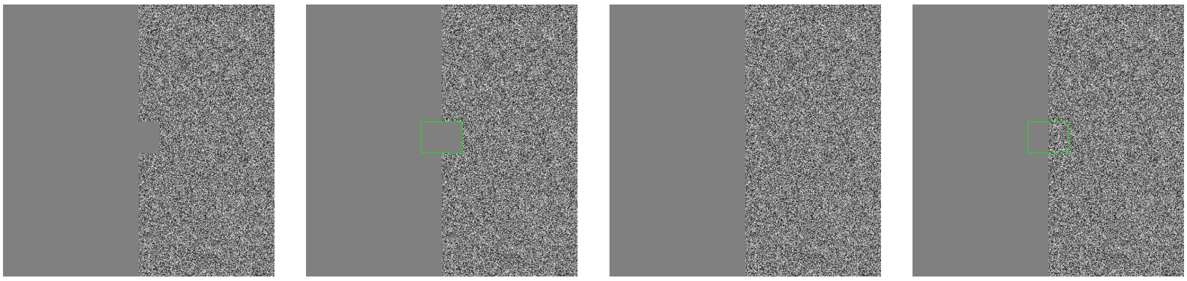
In this Appendix, we look in further detail at the reasons why textures may pose a problem when comparing patches for the purposes of inpainting. Liu and Caselles noted in [28] that the subsampling necessary for the use of multi-resolution pyramids inevitably entails a loss of detail, leading to difficulties in correctly identifying textures. In fact, we found that this difficulty may occur at *all* the pyramid levels in images and videos. Roughly speaking, we observed that textured patches are quite likely to be matched with smooth ones. The following simple computations quantify this phenomenon.

B.1. Comparing patches with the classical ℓ^2 distance.

The first reason concerns the patch distance. Let us consider a white noise patch, W, which is a vector of i.i.d. random variables $W_1 \cdots W_N$, where N is the number of components in the patch (number of pixels for grey-level patches), and the distribution of all W_i's is f_W. Let μ and σ^2 be, respectively, the average and variance of f_W. Let us consider another random patch V following same distribution, and the constant patch Z, composed of $Z_i = \mu$, $i = 1\cdots N$.

In this simple situation, we see that $\mathbb{E}[\|W-V\|_2^2] = 2\mathbb{E}[\|W-Z\|_2^2]$. Therefore, on average, the sum-of-squared-differences (SSD) between two patches of the same distribution is *twice* as great as the SSD between a randomly distributed patch and a constant patch of value μ.

The previous remark is only valid on average between three patches W, V and Z. In reality, we have many random patches V to choose from, and it is sufficient that one of these be better than Z for the least patch distance to identify a “textured” patch. Therefore, a more interesting question is the following. Given a white noise patch W, what is the probability that the patch V will be better than the constant patch Z. This is slightly more involved, and we shall limit ourselves to the case where W and V consist of i.i.d. pixels with a normal distribution.



Inpainting without features

Inpainting without features, with occlusion border outlined

Inpainting with features

Inpainting with features, with occlusion border outlined

Figure 11. **A toy example of the utility of the textures features**. *With them, we are able to distinguish between white noise (right) and the constant area (left), and thus recreate the noise.*

The SSD patch distance between W and V follows a chi-square distribution $\chi^2(0, 2\sigma^2)$, and that between W and Z follows $\chi^2(0, \sigma^2)$. With this, we may numerically compute the probability of a random patch being better than a constant one. Since the chi-squared law is tabulated, it is much the same thing to use numerical simulations.

In Table 2, we show the corresponding numerical values for both 2D (image) and 3D (video) patches. It may be seen that for a patch of size 9×9, there is very little chance of finding a better patch than the constant patch. In the video case, we see that in the case of $5 \times 5 \times 5$ patches, there is a 2×10^{-7} probability of creating a better patch randomly. This corresponds to needing an area of $170 \times 170 \times 170$ pixels in a video in order to produce on average one better random patch. While this is possible, especially in higher-definition videos, it remains unlikely for many situations.

B.2. ANN search with PatchMatch. We have indicated that the ℓ^2 grey-level/colour patch distance is problematic for inpainting in the presence of textures. Additionally, this problem is exacerbated by the use of PatchMatch. Indeed, the values of ϕ which lead to textures are not piecewise constant, and are therefore not propagated through ϕ during the PatchMatch algorithm. On the other hand, smooth patches represent on average a good compromise as ANNs and are the shifts which lead to them are piecewise constant and therefore well propagated throughout ϕ. Another problem which may lead to smooth patches being used is the weighted average reconstruction scheme. This can lead to blurry results which in turn means that smooth patches are identified.

One solution to these problems is the use of our texture feature pyramid (§2.3). This pyramid is inpainted simultaneously with the colour video pyramid, and thus helps to guide the algorithm in the choice of which patches to use for inpainting.

Figure 11 shows an interesting situation: we wish to inpaint a region which contains white noise. This toy example serves as an illustration of the appeal of our texture features. Indeed, it is quite clear that without them, there is no chance of inpainting the occlusion in a manner which would seem "natural" to a human observer, whereas with them it is possible, in effect, to "inpaint noise".

REFERENCES

[1] *http://www.photoshopessentials.com/photo-editing/content-aware-fill-cs5/.*

[2] Pablo Arias, Vicent Caselles, and Gabriele Facciolo, *Analysis of a variational framework for exemplar-based inpainting*, Multiscale Modeling & Simulation, 10 (2012), pp. 473–514.

[3] Pablo Arias, Gabriele Facciolo, Vicent Caselles, and Guillermo Sapiro, *A variational framework for exemplar-based image inpainting*, Int. J. Comput. Vision, 93 (2011), pp. 319–347.

[4] Sunil Arya and David Mount, *Approximate nearest neighbor queries in fixed dimensions*, in ACM-SIAM Symp. Discrete algorithms, 1993.

[5] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan Goldman, *Patchmatch: a randomized correspondence algorithm for structural image editing*, ACM Trans. on Graphics (Proc. SIGGRAPH), 28 (2009).

[6] Marcelo Bertalmio, Vicent Caselles, Simon Masnou, and Gillermo Sapiro, *Encyclopedia of Computer Vision*, Springer, 2011, ch. Inpainting.

[7] Marcelo Bertalmio, Guillermo Sapiro, Vicent Caselles, and Coloma Ballester, *Image inpainting*, in ACM SIGGRAPH, 2000.

[8] Raphaël Bornard, Emmanuelle Lecan, Louis Laborelli, and Jean Chenot, *Missing data correction in still images and image sequences*, in ACM Multimedia, 2002.

[9] Yuri Boykov, Olga Veksler, and Ramin Zabih, *Fast approximate energy minimization via graph cuts*, IEEE Trans. Pattern Anal. Machine Intell., 23 (2001), pp. 1222–1239.

[10] Anthoni Buades, Bartomeu Coll, and Jean-Michel Morel, *A non-local algorithm for image denoising*, in Int. Conf. Computer Vision and Pattern Recognition (CVPR), 2005.

[11] Aurélie Bugeau, Marcello Bertalmio, Vicent Caselles, and Guillermo Sapiro, *A comprehensive framework for image inpainting*, Image Processing, IEEE Transactions on, 19 (2010), pp. 2634–2645.

[12] Sen-Chin. Cheung, Jian Zhao, and M. Vijay Venkatesh, *Efficient object-based video inpainting*, in Int. Conf. Image Processing (ICIP, 2006.

[13] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama, *Object removal by exemplar-based inpainting*, in Int. Conf. Computer Vision and Pattern Recognition (CVPR), 2003.

[14] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan Goldman, and Pradeep Sen, *Image melding: combining inconsistent images using patch-based synthesis*, ACM Trans. Graph., 31 (2012).

[15] Laurent Demanet, Bing Song, and Tony Chan, *Image inpainting by correspondence maps: a deterministic approach*, tech. report, UCLA CAM, 2003.

[16] Iddo Drori, Daniel C. Or, and Hezy Yeshurun, *Fragment-based image completion*, ACM Trans. Graph., 22 (2003), pp. 303–312.

[17] Alexei Efros and Thomas Leung, *Texture synthesis by non-parametric sampling*, in Int. Conf. Computer Vision (ICVV), 1999.

[18] Miguel Granados, KwangIn Kim, James Tompkin, Jan Kautz, and Christian Theobalt, *Background inpainting for videos with dynamic objects and a free-moving camera*, in Europ. Conf. Computer Vision (ECCV), 2012.

[19] M. Granados, J. Tompkin, K. Kim, O. Grau, J. Kautz, and C. Theobalt, *How not to be seen: Object removal from videos of crowded scenes*, Comp. Graph. Forum (EUROGRAPHICS), (2012).

[20] Kaiming He and Jian Sun, *Computing nearest-neighbor fields via propagation-assisted kd-trees*, in Int. Conf. Computer Vision and Pattern Recognition (CVPR), 2012.

[21] ———, *Statistics of patch offsets for image completion*, in Int. Conf. Computer Vision (ECCV), 2012.

[22] Jan Herling and Wolfgang Broll, *Pixmix: A real-time approach to high-quality diminished reality*, in Int. Symp. Mixed and Augmented Reality (ISMAR), 2012.

[23] Jiaya Jia, Yu-Wing Tai, Tai-Pang Wu, and Chi-Keung Tang, *Video repairing under variable illumination using cyclic motions*, IEEE Trans. Pattern Anal. and Machine Intell., 28 (2006), pp. 832–839.

[24] Nikos Komodakis and George Tziritas, *Image completion using efficient belief propagation via priority scheduling and dynamic pruning*, IEET trans. Image Processing, 16 (2007), pp. 2649–2661.

[25] Simon Korman and Shai Avidan, *Coherency sensitive hashing*, in Int. Conf. Computer Vision (ICCV), 2011.

[26] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick, *Graphcut textures:*

image and video synthesis using graph cuts, ACM Trans. Graph., 22 (2003), pp. 277–286.

[27] Chih-Hung Ling, Chia-Wen Lin, Chih-Wen Su, Yong-Sheng Chen, and Hong-Yuan M. Liao, *Virtual contour guided video object inpainting using posture mapping and retrieval*, IEEE Trans. Multimedia, 13 (2011), pp. 292–302.

[28] Yunqiang Liu and Vicent Caselles, *Exemplar-based image inpainting using multiscale graph cuts*, IEEE Trans. Image Proc., 22 (2013), pp. 1699–1711.

[29] Simon Masnou and Jean-Michel Morel, *Level lines based disocclusion*, in Int. Conf. Image Processing (ICIP), 1998.

[30] Alasdair Newson, Andrés Almansa, Matthieu Fradet, Yann Gousseau, and Patrick Pérez, *Towards fast, generic video inpainting*, in Eur. Conf. Visual Media Production (CVMP), 2013.

[31] Jean-Marc Odobez and Patrick Bouthemy, *Robust multiresolution estimation of parametric motion models*, J. Vis. Comm. and Image Representation, 6 (1995), pp. 348–365.

[32] Igor Olonetsky and Shai Avidan, *Treecann - k-d tree coherence approximate nearest neighbor algorithm*, in Eur. Conf. Computer Vision (ECCV), 2012.

[33] Kedan Patwardhan, Guillermo Sapiro, and Marcelo Bertalmio, *Video inpainting of occluding and occluded objects*, in Int. Conf. Image Processing (ICIP), 2005.

[34] Kedar Patwardhan, Guillermo Sapiro, and Marcelo Bertalmio, *Video inpainting under constrained camera motion*, IEEE Trans. Image Processing, 16 (2007), pp. 545–553.

[35] Yael Pritch, Eitam Kav-Venaki, and Shmuel Peleg, *Shift-map image editing*, in Int. Conf. Computer Vision (ICCV), 2009.

[36] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa, *Video textures*, in ACM SIGGRAPH, 2000.

[37] Yonatan Wexler, Eli Shechtman, and Michal Irani, *Space-time video completion*, in Int. Conf. Computer Vision and Pattern Recognition (CVPR), 2004.

[38] ———, *Space-time completion of video*, IEEE. Trans. Pattern Anal. Machine Intel., 29 (2007), pp. 463–476.