Towards fast, generic video inpainting

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

[*]Technicolor, 975 Avenue des Champs Blancs, 35570 Cesson-Sévigné, France
alasdair.newson@technicolor.com, matthieu.fradet@technicolor.com, patrick.perez@technicolor.com

[†]Télécom ParisTech - CNRS LTCI, 43 rue Barrault, 75013 Paris, France
alasdair.newson@telecom-paristech.fr, andres.almansa@telecom-paristech.fr, yann.gousseau@telecom-paristech.fr

ABSTRACT

Achieving globally coherent video inpainting results in reasonable time and in an automated manner is still an open problem. In this paper, we build on the seminal work by Wexler *et al.* to propose an automatic video inpainting algorithm yielding convincing results in greatly reduced computational times. We extend the *PatchMatch* algorithm to the spatio-temporal case in order to accelerate the search for approximate nearest neighbours in the patch space. We also provide a simple and fast solution to the well known over-smoothing problem resulting from the averaging of patches. Furthermore, we show that results similar to those of a supervised state-of-the-art method may be obtained on high resolution videos without any manual intervention. Our results indicate that globally coherent patch-based algorithms are feasible and an attractive solution to the difficult problem of video inpainting.

Keywords

Video inpainting, PatchMatch, exemplar-based, patches, space-time occlusion

1. INTRODUCTION AND PRIOR WORK

The goal of *image inpainting* [1], also known as *disocclusion* [2, 3] is to convincingly replace a region with some other image content. The vast majority of inpainting methods find this information in the image itself. One of the main difficulties of this task is the correct reproduction of both geometric structures and textures. Video inpainting is the problem of inpainting a spatio-temporal hole in a video. This adds new technical challenges such as inpainting the foreground, background and moving objects. Additionally, the execution time becomes a critical aspect, as certain video inpainting algorithms which do not deal specifically with this aspect may take days or even weeks to execute.

Generally speaking, video inpainting algorithms belong to either the "object-based" or "patch-based" category. Object-based algorithms usually segment the video into moving foreground 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 [4], whereas moving objects may be copied into the occlusion as smoothly as possible. Unfortunately, such methods often include restrictive hypotheses on the moving objects' motion, such as strict periodicity. Some such object-based methods include [5, 6, 7].

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. The first patch-based method to ensure temporal coherency in video inpainting was described by Wexler *et al.* in [8]. This is an iterative method that may be seen as a heuristic to solve a global optimisation problem. The high dimensionality of the problem makes the algorithm very slow, requiring up to several days for a few seconds of VGA video. Patwardhan *et al.* [9] also use a patch-based approach. This is a greedy algorithm, and therefore cannot guarantee global coherency.

Pritch *et al.* [10] first proposed to use the discrete optimisation algorithm called "graph cuts" [11] for the purpose of image inpainting. This method optimises an energy functional of the *shift map* to inpaint an image. The shift map is the mapping from the set of pixels in the occlusion to a subset of the unoccluded pixels, which corresponds to the nearest neighbours from a patch-based point of view. Other image inpainting methods such as [12, 13] have also used this sort of strategy. The idea of using graph cuts for video inpainting was recently introduced by Granados *et al.* in [14]. 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 1120x754 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 convincing to date, and we shall therefore compare our algorithm with these results.

Another recent method by Facciolo *et al.* [15] deals with video editing in the gradient domain. However, the goal of this method is to enforce temporal consistency given a certain inpainting result, rather than producing the inpainting result itself. Nevertheless, such post-processing methods could be used to refine the results of video inpainting algorithms.

The seminal work of Wexler *et al.* [8] is widely cited and well-known, mainly because it ensures global coherency in an automatic manner. However, due to extremely long execution times, it is difficult to implement and experiment with, making the setting of implementation details and parameters especially tedious. This is in fact the greatest obstacle preventing the progress of research in this direction. The goal of this work is to produce an algorithm which is able to achieve the global coherence of the algorithm presented in [16], while maintaining execution times which are not prohibitive for experimentation and practical use. For this, we build on Wexler's central idea of iterative aggregation of nearest neighbours in the patch space to obtain an automatic algorithm with greatly reduced execution times. We extend the *PatchMatch* algorithm to the spatio-temporal domain in order to accelerate the search for approximate nearest neighbours. Furthermore, we propose a simple solution to the well-known over-smoothing problem due to the averaging of patches, and also provide specific implementation details to make our work reproducible. The resulting algorithm yields similar results on the low-resolution examples from [16], with a speedup of up to 50 times, and, furthermore, is able to successfully inpaint high resolution videos in an automatic manner, which has not been done before. We reduce execution times by an order of magnitude in comparison with the most recent *supervised* method [14], with visually similar results.

2. PATCH-BASED GLOBAL OPTIMISATION

Our video inpainting approach builds on the foundations laid out by Wexler *et al.* in [16]. This algorithm fills a spatio-temporal volume using the information in the unoccluded parts of the video. The solution is obtained by the minimisation of a global patch-based functional. To achieve this, the algorithm alternates between the search for the nearest neighbours (NNs) of spatio-temporal patches in the occluded region and the reconstruction of the inpainted volume using these NNs. This process is iterated several times in order to converge to a solution. As in many optimisation problems, a multi-resolution spatio-temporal pyramid is used in order to avoid local minima. We shall first of all recall the algorithmic structure of [16], after which we present our extension of PatchMatch to the spatio-temporal case (for NN the search) and finally our initialisation and modified reconstruction schemes.

2.1 The Space-Time Completion algorithm [16]

We present here the algorithm of [16] in greater detail. As far as possible, we retain the notation found in [16]. Let $\mathcal{H}$ be the spatio-temporal occlusion and $\mathcal{D}$ the data set (unoccluded area). Furthermore, let $\tilde{\mathcal{D}}$ be the region in which all patches are completely outside of $\mathcal{D}$. The ANN search is restricted to patches belonging to $\tilde{\mathcal{D}}$. Let $p = (x,y,t)$ be a position in the video and $\mathcal{N}_p$ be the spatio-temporal neighbourhood of p. This neighbourhood is defined as a rectangular cuboid centred on p. A *patch* is simply a vector of the values (grey-level, colour etc...) which are found in the video content at the positions belonging to $\mathcal{N}_p$. We shall refer to such a patch as W_p. Let us also define the NN *shift map* $\phi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, which indicates that the NN (for a given patch distance) of W_p is centred at the position $p+\phi(p)$. The NN patch of W_p shall therefore be denoted as $W_{p+\phi(p)}$. Wexler *et al.* use 5x5x5 patches, with each position in a patch being associated with the following vector : $(R,G,B,\beta u,\beta v)$, where R,G and B are the colour values, u and v are roughly estimated optical flow components, and β is a scaling factor. The distance $d(W_p, W_{p+\phi(p)})$ is the sum of squared differences (SSD) of each of these components, for all positions in the patch.

The inpainting algorithm of Wexler *et al.* and, indeed, many inpainting algorithms in general, contain three main components:

- The search for NNs
- The reconstruction of a video using the NNs
- The implementation of the spatio-temporal pyramid.

As previously mentioned, the first two steps are iterated several times, at each pyramid level.

An exhaustive search for exact NNs is obviously far too time-consuming, especially in the spatio-temporal case. Therefore, in order to make the algorithm feasible, the work of Arya *et al.* [17] is used in [16] to find *approximate nearest neighbours* (ANNs), rather than exact nearest neighbours.

Concerning the reconstruction of the solution, Wexler *et al.* propose a scheme in which the colour value c of a pixel $p \in \mathcal{H}$ is obtained with a *weighted mean* of all the colours given by the ANNs of the patches which contain p :

$$c_p = \frac{\sum_{i \in \mathcal{N}_p} \alpha_p^i s_p^i c^i}{\sum_{i \in \mathcal{N}_p} \alpha_p^i s_p^i} \tag{1}$$

with

$$s_p^i = \exp\left(-\frac{d(W_i, W_{i+\phi(i)})}{2\sigma^2}\right),$$

where c^i is the colour value at the position $p+\phi(i)$. The value α_p^i is a weight given to the pixel i which reflects the distance from i to the occlusion boundary. More precisely, $\alpha_p^i = \gamma^{dist(p)}$, where $dist(p)$ is the distance from p to the occlusion boundary, and γ is set to 1.3. Finally, σ is defined as the 75th percentile of all the distances $d(W_i, W_{i+\phi(i)}), i \in \mathcal{N}_p$. One drawback of the previous weighted mean is a blurred inpainting result. Therefore, Wexler *et al.* propose another reconstruction method which is a robust estimation of the value of c_p using the *mean-shift* algorithm. This mean-shift is carried out on points in the RGB colour space, provided by c^i. Each point is weighted by $\alpha_p^i s_p^i$, and the dominant mode (the cluster with the most votes) is found in this space. Only the colours which belong to the dominant mode are used for the reconstruction of the pixel. During the algorithm, the bandwidth used for the mean shift is gradually reduced. The goal of this scheme is to gradually decide on the best ANN for each W_p. The mean shift algorithm is used in order to make the reconstruction robust to poor ANNs found by the ANN search, and also reduces the blurring effect in the final result.

Finally Wexler *et al.* use a spatio-temporal Gaussian pyramid in order to improve the optimisation results, as is common in inpainting algorithms. The downsampling scheme is not precisely defined in [16]. The upsampling of a solution from one level to another is done by propagating the current ANNs to the finer level. The finer solution is then produced with the same reconstruction scheme, using the new ANNs.

We now proceed to explain the details of the proposed algorithm.

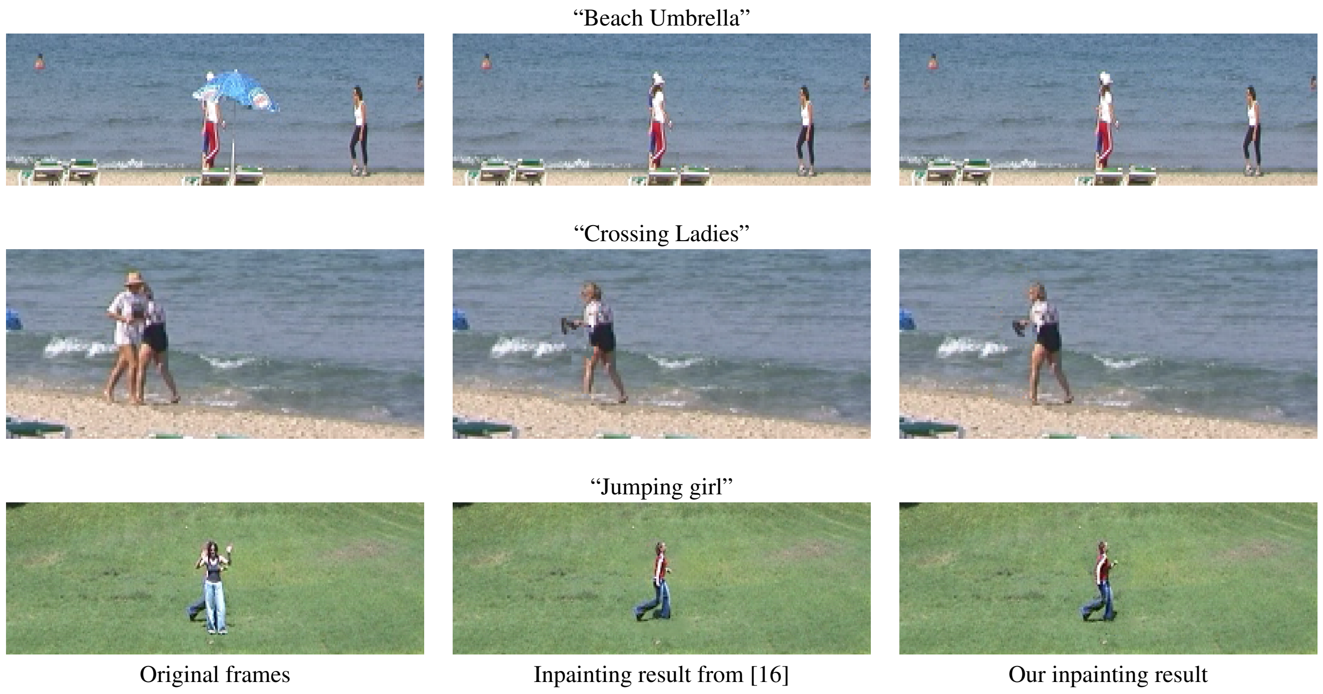


Figure 1: Comparison of our results with the those of [16]. Visually, the obtained inpainted is very similar, but we are able to reduce the ANN search time by a factor of up to 50 times. Result videos are viewable at http://www.enst.fr/~gousseau/videoinpainting_cvmp

2.2 Approximate nearest neighbour search

The search for ANNs is the most time-consuming part of the algorithm, and thus the most important to work on to obtain faster execution times. Relatively recently, a very efficient ANN search algorithm named "PatchMatch" was proposed by Barnes *et al.* [18], specifically for the task of finding perceptually close patches in images. PatchMatch's efficiency comes from the observation that ANN shift maps are often very regular between images (in fact they are piecewise constant), if the same image content is present. Barnes *et al.* suggested the use of PatchMatch for image inpainting purposes, and this algorithm was subsequently used in the "Content Aware Fill" tool by Photoshop CS5 [19], and more recently by Darabi *et al.* [20]. To the best of our knowledge it has never been proposed for video inpainting. We propose to extend the PatchMatch algorithm to the spatio-temporal setting, in order to reduce the search times obtained with the ANN search described in [17]. It seems quite natural to use PatchMatch for video inpainting, as videos are even more repetitive than images, and this suits the piecewise constant nature of the ANN shift maps found by PatchMatch very well.

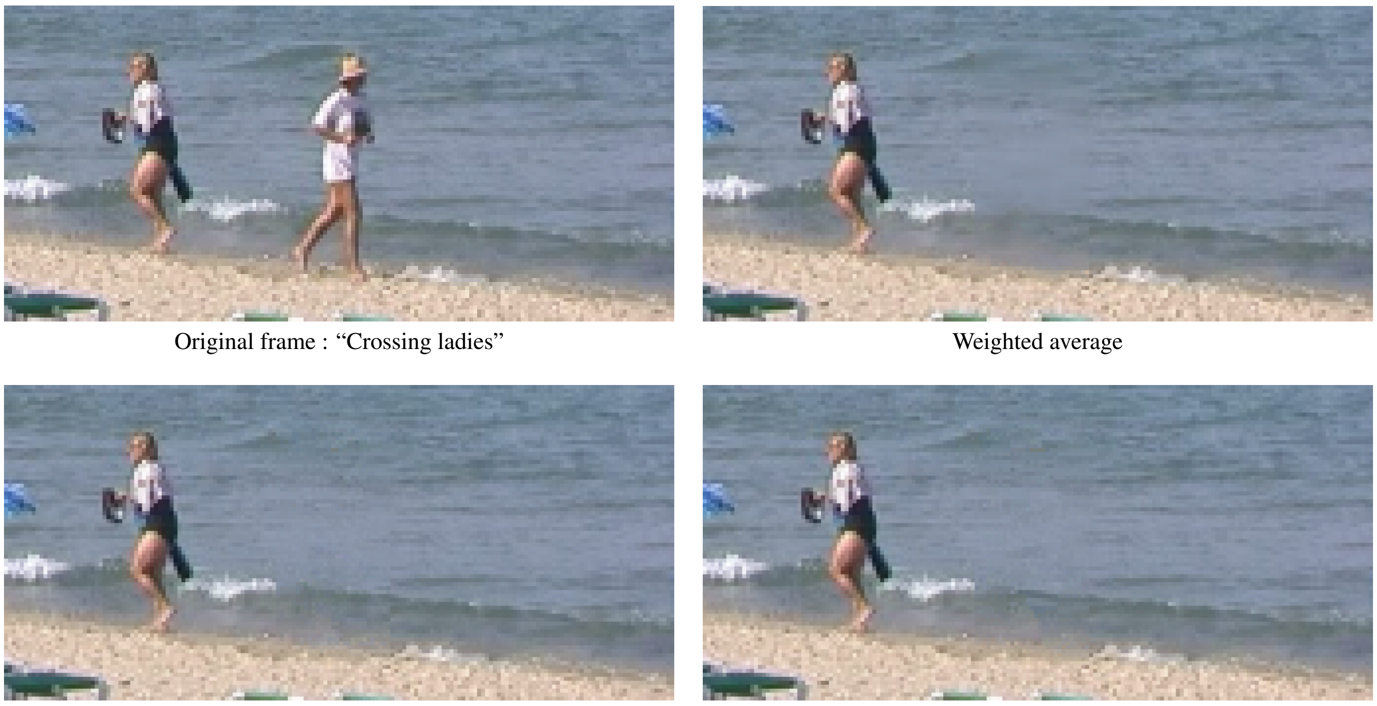
The original PatchMatch contains three steps : initialisation, propagation and random search. Note that we keep the same notation as in the spatio-temporal case. In particular, p represents a position (x,y) in the image, and $\phi : \mathbb{R}^2 \to \mathbb{R}^2$ represents the shift map. We use the same notation in the image and video cases to make clarify our presentation, and assume that it will be clear which case is being discussed. The initialisation is done by randomly associating an ANN to each patch W_p. During propagation, the patches are sequentially scanned from low to high indices, first in the x and then in the y dimension. For a given patch $W_{(x,y)}$, the algorithm considers the following ANNs : $W_{(x,y)+\phi(x-1,y)}$ and $W_{(x,y)+\phi(x,y-1)}$. If one of these ANNs has a smaller patch distance with respect to $W_{(x,y)}$ than $W_{(x,y)+\phi(x,y)}$, then $W_{(x,y)+\phi(x,y)}$ is replaced with the best new ANN. The scanning order is reversed for the next iteration of the propagation (from high to low), and the algorithm tests $W_{(x,y)+\phi(x+1,y)}$ and $W_{(x,y)+\phi(x,y+1)}$. In the two different scanning orderings, the important point is obviously to use the neighbours which have already been tested in the current propagation step. The motivation for this step is the idea that objects in images are spatially coherent, and therefore that offsets which lead to good ANNs for a given patch are likely to produce good ANNs in the proximity of this patch. The third step, the random search, consists of randomly looking for better ANNs of each W_p in an increasingly small area around $W_{p+\phi(p)}$, starting with a maximum search distance. The random ANNs are centred at the following positions :

$$q_j = p + \phi(p) + w\rho^j \mathbf{R}_j, \tag{2}$$

for all $j \in \mathbb{N}$ such that $w\rho^j > 1$, where w is the maximum search radius around the $W_{p+\phi(p)}$, $\mathbf{R}_j$ is a pair of random variables in $[-1,1] \times [-1,1]$ and ρ 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.

We now detail our extension of PatchMatch to the video setting. Let $W_{(x,y,t)}$ be a spatio-temporal patch centred at (x,y,t). We initialise the ANNs randomly in any position $p \in \tilde{\mathcal{D}}$.

For propagation, we need to define a scanning order of the patches. As in [18], we scan in the x dimension and then the y dimension. Lastly, we scan in the temporal dimension. For each $W_{(x,y,t)}$, we test $W_{(x,y,t)+\phi(x-1,y,t)}$, $W_{(x,y,t)+\phi(x,y-1,t)}$ and $W_{(x,y,t)+\phi(x,y,t-1)}$. As in [18], the scanning order is reversed between propagation iterations, and $W_{(x,y,t)+\phi(x+1,y,t)}$, $W_{(x,y,t)+\phi(x,y+1,t)}$ and $W_{(x,y,t)+\phi(x,y,t+1)}$ are


Original frame : "Crossing ladies" Weighted average

Mean shift Best patch

Figure 2: Comparison of different reconstruction methods. We observe that the reconstruction using the best patch at the end of the algorithm produces similar results to the use of the mean shift algorithm. Please note that the blurring effect is best viewed in the pdf version of the paper.

tested. For this propagation step, it is important to know whether the scanning order of the video dimensions has any influence on the quality of the resulting ANNs. In particular, the regularity properties (upon which the PatchMatch algorithm is based) may be different for different dimensions. For example, if we suppose that there is greater information redundancy along the t axis, then it is preferable to analyse the ANNs along this axis before analysing the others. This sort of consideration was not taken into account in [18], since there is no reason *a priori* to suppose that the x and y axes present different degrees of coherence. For a visual illustration of the neighbours used during the propagation step, see Figure 3.

Table 1 compares the average ANN error (in terms of the sum of square differences) per patch component, for all the scanning orderings (nine in total). The three standard sequences of Wexler *et al*. [16] are analysed in this manner. We observe that the patch error is very stable with respect to the scanning order. Therefore, we have chosen (arbitrarily) to maintain the original scanning order of PatchMatch, and add the t dimension afterwards.

The extension of the random search step is straightforward. Instead of searching in a square around current ANNs, we search in a cube, for all $p \in \tilde{\mathcal{D}}$. We keep the original window size reduction parameter $\beta = 0.5$. In the current work, we allow the random search to search the entire video, however this could possibly be tuned to accelerate the algorithm further. We set the number of iterations of propagation and random search to 10, which gives good results on all the videos we tested. Between iterations of ANN search and reconstruction, we initialise PatchMatch with the previous ANNs.

In [16], the patches include rough estimations of an optical flow, u and v. Having experimented with and without these elements, we found that their influence was very small. Therefore, we only use the colour components in our patch comparisons.

2.3 Reconstruction

As mentioned in Subsection 2.1, two reconstruction schemes are presented in [16]: weighted averaging and a robust mean shift colour estimation. While the mean shift has advantages such as avoiding blurring in the final results, it complicates the algorithm and adds new parameters, such as the speed at which the mean shift's band width is reduced, for which details are not given in [16].

Dealing with this blurring problem is particularly important if video textures are present, such as the waves in the "Beach Umbrella" and "Crossing Ladies" examples of Wexler *et al*. An important question is whether the gradual reduction of the band-width of the mean shift throughout the algorithm is necessary, or if this may be delayed until the end of the algorithm. In terms of Equation (1), the mean shift band-width reduction is very similar to reducing the parameter σ throughout the iterations of ANN search and reconstruction.

After testing and comparing various reconstruction schemes, we propose the following method. Throughout the algorithm we use the weighted mean of Equation (1). Then, after convergence of this scheme at the finest pyramid level, we reconstruct the video using only the colour value c^i indicated by the ANN in $\mathcal{N}_p$ with the smallest patch distance. This reconstruction correctly "deblurs" the result, and is far simpler to implement and use than the mean shift. With respect to execution time, the mean shift algorithm

	Mean ANN patch error, per component						
Propagation order	t,y,x	y,t,x	t,x,y	x,t,y	y,x,t	x,y,t	Full search
Beach Umbrella	9.22	9.61	9.49	9.54	9.57	9.11	6.83
Crossing Ladies	7.53	7.44	7.42	7.51	7.50	7.35	6.14
Jumping Girl	6.48	6.52	6.40	6.49	6.50	6.45	4.80

Table 1: Comparison of propagation scanning ordering, in terms of average component error between a patch and its approximate nearest neighbour. On the last row is the average error of the true nearest neighbour.

has a time complexity of $O(T|W_p|^2 n_o)$, where T is the number of iterations of each mean shift, $|W_p|$ is the size of a patch, and n_o is the number of occluded pixels. This mean shift is repeated at each iteration within each pyramid level. The proposed reconstruction, on the other hand, has a time complexity of $O(|W_p|n_o)$, the same as the weighted mean. Even though the ANN search represents the majority of computation time, it is obviously preferable to use a reconstruction scheme which is linear, rather than quadratic, with respect to the patch size.

Figure 2 shows some visual comparisons of the inpainting results using different reconstruction schemes. The weighted mean, the mean shift and the proposed reconstruction are compared. We observe that the reconstruction using the best ANN (the proposed reconstruction) is extremely similar to the result using the mean shift. The proposed reconstruction is able to recreate which corresponds to video textures.

Another crucial part of the reconstruction is the initialisation of the inpainting solution at the coarsest pyramid resolution. This initialisation is left unspecified in [16]. To initialise the solution, we inpaint at the coarsest level using an onion peel approach, with a layer thickness of one pixel. More formally, let $\mathcal{H}'$ be the current occlusion, and $\partial\mathcal{H}'$ the current layer to inpaint. This layer is obtained by morphological erosion of $\mathcal{H}'$ with a 26-neighbourhood (a 3x3x3 cube). We define the *unoccluded* neighbourhood $\mathcal{N}'_p$ of a pixel p, with respect to the current occlusion $\mathcal{H}'$. These are all the pixels in the neighbourhood of p which are not currently occluded.

Some choices are in order 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:

$$d(W_p, W_{p+\phi(p)}) = \frac{\sum_{i\in\mathcal{N}'_p}(||I(i) - I(i)||_2^2)}{|\mathcal{N}'_p|}. \quad (3)$$

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(1):

$$c_{p,p\in\partial\mathcal{H}'} = \frac{\sum_{i\in\mathcal{N}'_p}\alpha_p^i s_p^i c^i}{\sum_{i\in\mathcal{N}'_p}\alpha_p^i s_p^i}. \quad (4)$$

The pixels within a given layer of the occlusion are inpainted in

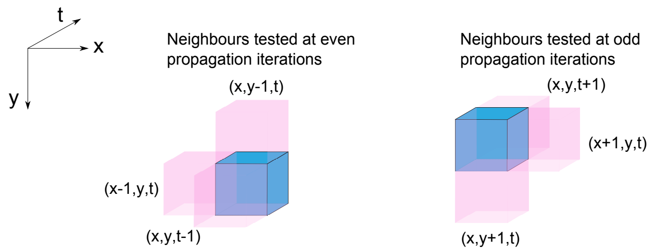

t
x
y
Neighbours tested at even propagation iterations
(x,y-1,t)
(x-1,y,t)
(x,y,t-1)
Neighbours tested at odd propagation iterations
(x,y,t+1)
(x+1,y,t)
(x,y+1,t)

Figure 3: A visual illustration of the neighbour whose shift maps are tested during the propagation step of the spatio-temporal PatchMatch. PatchMatch changes the neighbours used for testing between odd and even iterations.

parallel, not sequentially.

Finally, in [16] the number of iterations in a pyramid level is fixed. In the interest of robustness, we use the average pixel colour difference in each channel 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.

2.4 Spatio-temporal subsampling

The last part of the algorithm concerns the implementation of the multi-resolution spatio-temporal pyramid. As noted in [14], temporal subsampling may produce undesirable effects. We do not temporally subsample either, apart from cases where objects are occluded for a long time, which was done on only one video ("Jumping girl") in this paper. Before downsampling, we filter the video by averaging 2x2 image blocks.

The use of a spatio-temporal pyramid implies two further details: the upsampling of the current solution and whether or not to pass an upsampled version of the current ANNs as an initialisation to the PatchMatch at the finer level. We keep the same solution upsampling procedure as in [16], which is able to preserve fine details. However, we do not initialise PatchMatch at a finer level using the coarse level ANNs. This avoids biasing the finer ANNs towards certain spatio-temporal positions. Since certain details may not be visible at coarser levels, we prefer to let PatchMatch find the best ANNs at each level independently.

3. RESULTS

Since the main goal of our work is to achieve generic video inpainting with good results in reduced time, we evaluate our algorithm in terms of both execution time and visual quality. In this section, we shall compare our work to that of Wexler *et al.* [16] and the most recent video inpainting method of Granados *et al.* [14].

In all of our experiments, we have retained the parameters as they

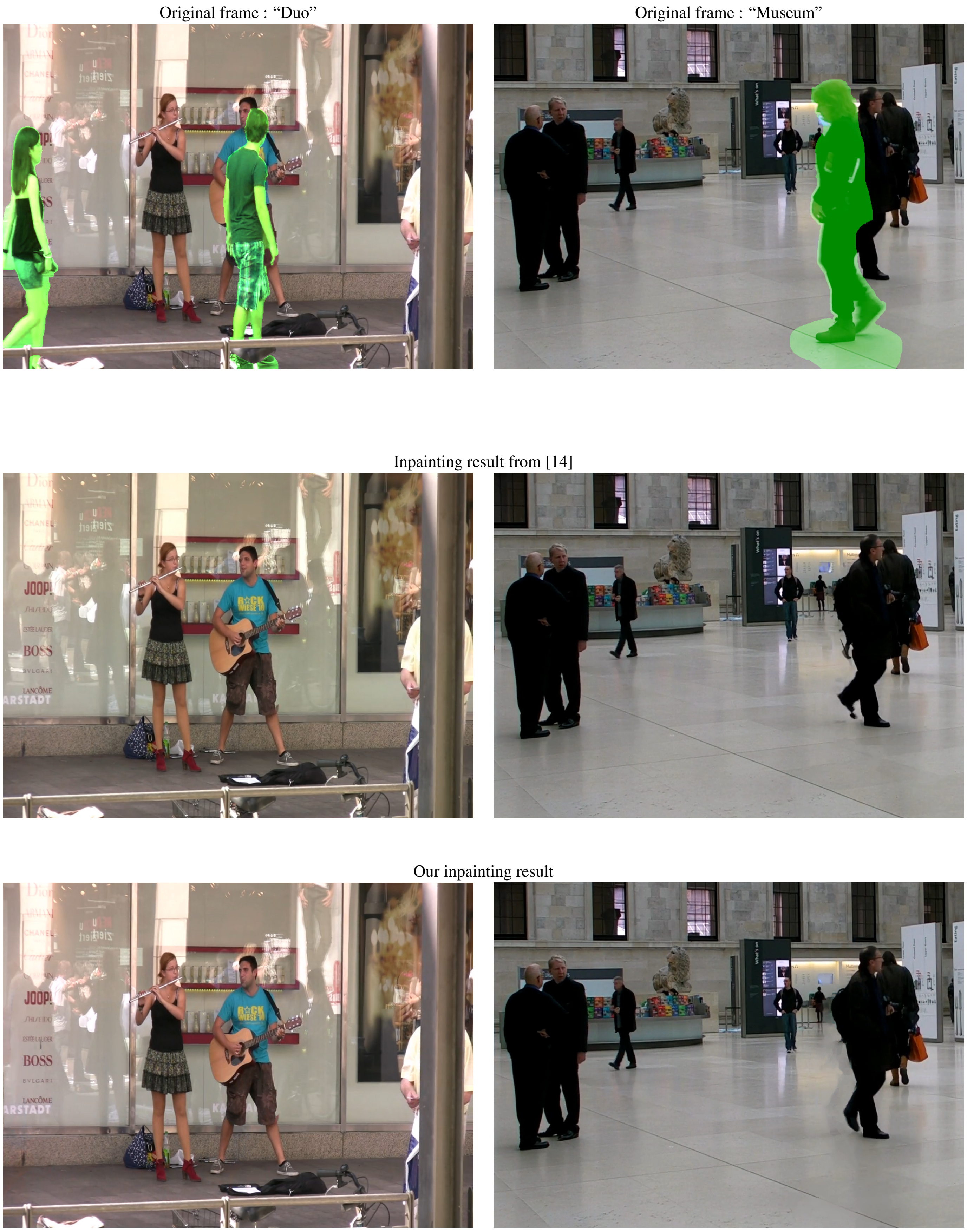


Figure 4: We achieve similar results to those of [14] in an order of magnitude less time, without user intervention. The occlusion masks are highlighted in green. Result videos are viewable at http://www.enst.fr/~gousseau/videoinpainting_cvmp

Algorithm	Approximate nearest neighbour execution times, for all occluded pixels at full resolution.				
	Beach Umbrella 264x68x200	Crossing Ladies 170x80x87	Jumping Girl 300x100x239	Duo 960x704x154	Museum 1120x754x200
Wexler	985 s	942 s	7877 s	-	-
Ours	50 s	28 s	155 s	2515 s	3958 s

Algorithm	Total execution time				
Granados	11 hours	-	-	-	90 hours
Ours	21 mins	6 mins	62 mins	7.1 hours	8.64 hours

Table 2: Partial and total inpainting 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.

are presented in the main body of the paper. In particular, we keep the patch size of 5x5x5 as used in [16]. We use ten iterations of propagation/random search during the PatchMatch algorithm and set the window size reduction factor β to 0.5. During the construction of the spatio-temporal pyramid, we do not subsample temporally in any of the videos, apart from "Jumping girl".

In [16], neither the initialisation of the occluded area nor the number of iterations per pyramid level are precisely defined. Since total execution time greatly depends on these details, we shall only compare our algorithm to Wexler's in terms of ANN search time. This represents the majority of total time (in [16], it takes up 95 percent of execution time), thus it makes sense to compare these timings. While our reconstruction scheme also reduces execution time, it will not represent as great a speed-up as the ANN search time, so we do not compare reconstruction timings. We also provide our total execution times, for reference.

Comparisons with Granados's results are more difficult, as it is semi-automatic. However, several execution times are provided in [14] and we compare our timings to these.

We recall that Wexler uses the method of [17] to search for ANNs. This method uses a parameter, ε, which determines the accuracy of the ANNs. As this parameter is not given in [16], we set it to produce the same average error per component as the spatio-temporal PatchMatch. This is the same evaluation method used by Barnes *et al.* in [18]. Specifically, we set ε to 10. In Table 2, the ANN search time comparisons may be seen. We obtain a speedup of 20-50 times over the method of [17].

In comparison to the work of [14] our method took 8h38m on the longest example ("Museum"), while Granados *et al.* report a total execution time of 90 hours, with a similar computer architecture to ours. We used a 64-bit machine with a 2.67 GHz Intel Xeon processor. For fairness, only one core of the processor was used (as in [14]). However, in Granados's work each occluded object and the background are inpainted separately, in parallel, whereas we treat all objects simultaneously. In the "Museum" example Granados reports [21] seven objects plus the background, meaning that the real workload is roughly several times greater.

We now compare the time complexity of our algorithm with that of [21]. The patch match algorithm runs in $O(6Qn_od_p + Qn_olog_2(N))$ time, where (as above) n_o is the number of occluded pixels, d_p is the dimension of the patches, N is the maximum random search space size, and Q is the number of iterations of propagation/random search. The reconstruction runs in $O(n_od_p)$ time. In comparison, Granados *et al.* report a time complexity of $O(n_o^3N)$. This explains the quite long execution times reported in [14] and in particular the need to restrict the search space manually.

It is interesting to note that the search space restriction used in [14] may not only reduce execution times, but might in fact improve inpainting results. Liu *et al.* [13], who used graph cuts for the purpose of image inpainting, have reported that inpainting results are often degraded when the entire image is used as a search space. In the case of image inpainting, restricting the search space is not a significant problem, since in the vast majority of cases the necessary information is situated around the occlusion. On the other hand, this is not the case for video inpainting. For example, if an object displays periodic motion with a particularly large period, then the video information may be situated at a correspondingly large spatio-temporal distance. In the proposed algorithm, we do not restrict the search space and we are able to produce coherent results. Finally, in the case of graph cut based inpainting schemes such as [10, 13, 14], it is not possible to use patches of arbitrary size, due to execution time issues. This is an extremely limiting factor, and is a strong argument in favour of inpainting schemes such as that which has been presented in this work.

Figure 1 and Figure 4 present visual comparisons of our results with those of [16] and the high resolution results of [14]. We observe that our results are qualitatively very similar. These results were obtained in highly reduced execution times, and in a generic manner, without manual user intervention, validating the initial goal of our work. The comparison videos may be viewed at the following address : http://www.enst.fr/~gousseau/videoinpainting_cvmp.

4. FURTHER WORK

There are several points in the proposed algorithm which could be improved upon. In particular, the use of a spatio-temporal pyramid means that certain comparisons may be ambiguous at coarse resolutions. Liu *et al.* [13] introduce new components in the patch comparison to avoid this problem, and this could possibly be extended to the video context. Furthermore, we have not exploited the parallel aspect of both the random search step and the reconstruction. A parallel implementation of these steps could further decrease the execution times, making results available more quickly and increasing experimental possibilities. Finally, recent work [12] on nearest neighbour searches claims to accelerate PatchMatch by a factor of 10-20 in the 2D case, by using a kd-tree and dimensionality reduction. Extending these ideas to spatio-temporal patches presents new challenges, such as choosing a basis which is effective in terms of dimensionality reduction and computational cost.

5. CONCLUSION

We have shown that automatic video inpainting with global optimisation of a patch-based functional in reasonable execution times is possible, even for high resolution videos. By extending the PatchMatch algorithm to the case of spatio-temporal patches, we are able to provide a fast, useable video inpainting algorithm. Furthermore, we have proposed a fast, simple solution to the problem of over-smoothing of video inpainting results which is particularly problematic in the case of video textures. We have compared our results with those of [16] and [14] and found that our algorithm produces very similar results in an order of magnitude less time.

6. ACKNOWLEDGEMENTS

The authors would like to express their thanks to Miguel Granados for his kind help and for answering their questions concerning his work.

7. REFERENCES

[1] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," *SIGGRAPH*, pp. 417–424, 2000.

[2] S. Masnou and J.-M. Morel, "Level lines based dissoclusion," *ICIP*, vol. 3, pp. 259–263, 1998.

[3] S. Masnou and J.-M. Morel, "Disocclusion: a variational approach using level lines," *IEEE Trans. Image Processing*, vol. 11, no. 2, 2002.

[4] A. Criminisi, P. Perez, and K. Toyama, "Object removal by exemplar-based inpainting," *CVPR*, pp. 721–728, 2003.

[5] J. Jia, Y-W. Tai, T-P. Wu, and C-K. Tang, "Video repairing under variable illumination using cyclic motions," *PAMI*, vol. 28, no. 5, pp. 832–839, 2006.

[6] M. V. Venkatesh, S-C. S. Cheung, and J. Zhao, "Efficient object based video inpainting," *ICIP*, vol. 30, pp. 168–179, 2006.

[7] C-H. Ling, C-W. Lin, C-W. Su, Y-S. Chen, and H-Y. Liao, "Virtual contour guided video object inpainting using posture mapping and retrieval," *IEEE Trans. Multimedia*, vol. 13, no. 2, pp. 292–302, 2011.

[8] Y. Wexler, E. Shechtman, and M. Irani, "Space-time video completion," *CVPR*, vol. 1, pp. 120–127, 2004.

[9] K. A. Patwardhan, G. Sapiro, and M. Bertalmio, "Video inpainting of occluding and occluded objects," *ICIP*, vol. 2, pp. 69–72, 2005.

[10] Y. Pritch, E. Kav-Venaki, and S. Peleg, "Shift-map editing," *ICCV*, pp. 151–158, 2009.

[11] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. PAMI*, vol. 23, no. 11, pp. 1222–1239, 1999.

[12] K. He and J. Sun, "Computing nearest-neighbor fields via propagation-assisted kd-trees," *CVPR*, pp. 111–118, 2012.

[13] Y. Liu and V. Caselles, "Exemplar-based image inpainting using multiscale graph cuts," *IEEE Trans Image Process.*, vol. 22, pp. 1699–1711, 2013.

[14] 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," *Eurographics*, vol. 31, pp. 219–228, 2012.

[15] G. Facciolo, R. Sadek, A. Bugeau, and V. Caselles, "Temporally consistent gradient domain video editing," *EMMCVPR*, vol. 6819, pp. 59–73, 2011.

[16] Y. Wexler, E. Schechtman, and M. Irani, "Space-time completion of video," *IEEE Trans. PAMI*, vol. 29, no. 3, pp. 463–476, 2007.

[17] S. Arya and D. Mount, "Approximate nearest neighbor queries in fixed dimensions," *SODA*, pp. 271–280, 1993.

[18] C. Barnes, E. Schechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *SIGGRAPH*, vol. 28, no. 3, 2009.

[19] "http://www.photoshopessentials.com/photo-editing/content-aware-fill-cs5/," July 2013.

[20] S. Darabi, E. Schechtman, and C. Barnes, "Image melding : Combining inconsistent images using patch-based synthesis," *SIGGRAPH*, vol. 31, 2012.

[21] M. Granados, J. Tompkin, K. Kim, O. Grau, J. Katuz, and C. Theobalt, "How not to be seen – inpainting dynamic objects in crowded scenes," Tech. Rep., Max-Planck Institute für Informatik, http://domino.mpi-inf.mpg.de/internet/reports.nsf/, February 2011.