# EXAMPLAR-BASED VIDEO INPAINTING WITH MOTION-COMPENSATED NEIGHBOR EMBEDDING

*Mounira Ebdelli, Christine Guillemot*

INRIA
Campus Universitaire de Beaulieu
35042 Rennes Cedex France

*Olivier Le Meur*

University of Rennes 1
Campus Universitaire de Beaulieu
35042 Rennes Cedex France

## ABSTRACT

This paper describes a video inpainting algorithm based on motion-compensated neighbor embedding. The unknown pixels are estimated as a linear combination of the $K$ closest patches using motion-compensated neighbor embedding. The algorithm is first assessed by assuming the motion information of the masked pixels to be known. This assumption is not realistic in video editing (object removal) applications. It however helps isolating the various problems for the sake of analysis. Different approaches are then assessed in the context where the motion information of missing pixels is unknown. Experiments on several videos show the benefits of the proposed approach which lead to natural looking videos with less annoying artefacts than when using a template matching technique.

## 1. INTRODUCTION

Image inpainting refers to methods which consist in filling-in missing regions (holes) in an image [1],[2]. Existing methods can be classified into two main categories. The first category concerns diffusion-based approaches which propagate level lines or linear structures (so-called isophotes) via diffusion based on partial differential equations [1], [3] and variational methods [4]. Diffusion-based methods tend to introduce some blur when the hole is large. The second type of approach concerns examplar-based methods which sample and copy best match texture patches from the known image neighborhood [5]-[6].

In this paper, we focus on the examplar-based family of inpainting methods. First extensions of examplar-based image inpainting method to video has been proposed in [7], [8] and [9]. The algorithm in [7]-[8] first estimates the motion for each pixel in the image in order to determine whether a pixel $p$ belongs to a moving object ($M_c(p) = 1$) or it belongs to the static background ($M_c(p) = 0$). It then proceeds by first inpainting the moving objects. In this step, the priority data term of [10] is adapted in order to give more priority to patches for which the motion is orthogonal to the hole front line. The most similar patch to the input patch is then searched in neighboring frames, by using a template matching which computes the SSD between vectors of five components (R, G, B, $V_x$, $V_y$), where $V_x$ and $V_y$ denote the horizontal and vertical components of the motion vector, of the known samples in the input patch and the corresponding samples in the candidate patches. Each time a patch is filled, the motion information $M_c(p)$ of the filled pixels is updated with the $M_c$ values of the copied pixels. The priority of non moving pixels is set to $0$, and the algorithm iterates until all the pixels to be filled have their priority equal to $0$. The algorithm then proceeds by inpainting the stationary background. Patches with the closest temporal locality with the highest priority are copied first until no more temporal information is available. The algorithm then terminates by inpainting the remaining holes using the image inpainting technique of [10].

Here, we extend the examplar-based video inpainting algorithm of [7] along several directions:

- The search for the best matching patch is performed within a motion-compensated window rather than in the entire images of the sequence. This first allows reducing the computational time. This also favours the exploitation of self similarity in a local spatio-temporal neighborhood.
- The simple copy strategy inherent to the template matching is also replaced by more elaborate neighbor embedding techniques (i.e. LLE [11] and NMF [12]) which are locally chosen in order to minimize the approximation error of the known pixels in the input patch [13].
- The use of these techniques instead of the simple TM has required to adapt the strategy for computing the similarity between the input patch and its K-NN, as well as for updating the $M_c$ information of the pixels to be filled.

The proposed inpainting technique has first been assessed assuming this information known also for the masked pixels. This assumption is obviously not realistic in a context of video editing. However, it can be valid in other applications, like loss concealment, provided the motion vectors of given patches (or blocks) can be transmitted in separate packets from the ones conveying the texture information, hence be known (correctly received) for the missing pixels. The method has then been evaluated in the context where the motion information ($M_c$) is unknown for the areas to be filled in, in which case the corresponding $M_c$ values need to be estimated for the unknown pixels from the ones of the $K$-NN patches.

## 2. EXAMPLAR-BASED VIDEO INPAINTING: BACKGROUND

This section summarizes the main steps of the examplar-based video inpainting algorithm proposed in [7]-[8]. Motion information is first estimated for each pixel in each image of the input video sequence, by using an optical flow estimation algorithm or by using a simple 4x4 block matching algorithm. A threshold applied on the horizontal ($V_x$) and vertical ($V_y$) components allows determining whether the pixel belongs to the moving foreground object ($M_c = 1$) or to the stationary background ($M_c = 0$). The process then follows three main steps: (i) hole filling in the foreground moving object; (ii) hole filling in the stationary background; (iii) filling in the remaining holes with a spatial image inpainting technique.

### 2.1. Inpainting the moving foreground object

Once the fill front $\delta\Omega$ within the image $I_t$ has been identified, for each patch $\Psi_p$ to be filled, centered at pixel p (unknown pixel) located near the front line, computation of the priority $P(p) = D(p)C(p), \forall p \in \delta\Omega$, where the confidence term $C(p)$ and the data term $D(p)$ are defined as in [8]. The data term aims at giving more priority to patches for which motion direction is perpendicular to the front line.

The algorithm then searches for the patch $\Psi_{\hat{p}}$ (centered on a pixel $p$ located on the fill front) having the highest priority, i.e. $\hat{p} = \text{argmax}_{p\in\delta\Omega}P(p)$. Given an input patch $\Psi_{\hat{p}}$ to be filled, it finds the most similar patch to the known region in $\Psi_p$ in a given search window, centered on the spatial position of the pixel $p$, within the known parts of the previous or next frames, by computing the sum of squared differences (SSD) between the corresponding 5 components vectors (R,G,B, $V_x$,$V_y$). The unknown pixels of the input patch are then approximated by a copy of their co-located pixels in the most similar patch (this is a so-called Template Matching (TM)). The confidence term $C(p)$ is updated by computing the new ratio of known versus unknown pixels in the patches, and the motion term $M_c(p)$ is updated by setting its value to the value of the corresponding copied pixel from the most similar candidate patch. We will see in Section 4 that errors in the update of the values of the missing pixels can induce dramatic inpainting errors at the next filling steps.

### 2.2. Inpainting the stationary background

The priority of all patches to be filled in is computed as $P(p) = C(p)D(p)$, where $C(p)$ and $D(p)$ are also defined in [8]. Here again, the data term $D(p)$ measures the amount of information available in previous and next frames at the location of the pixel $p$. Patches with the closest temporal locality and the highest priority, taken from a search window centered on the position of the pixel $p$, are copied first until $D(p) = 0$ which means that no more temporal information is available to fill in the remaining patches. The remaining holes are then filled in by a method simalar to [10]. The major difference concerns the computation of the best patches which is chosen between the best patches stemming from TM, LLE and NMF (this is explained at the end of Section 3).

## 3. MOTION-COMPENSATED NEIGHBOR EMBEDDING

Neighbor embedding methods refer to a range of methods which aim at approximating a given input data point by its $K$ nearest neighbors. Instead of searching for the best matching patch within a search window centered on the spatial position of the input patch in previous and/or next frames, the search is performed within a window centered on the position indicated by the motion vector $(V_x(p), V_y(p))$ of the pixel $p$. The SSD between the 5 components vectors (R,G,B,$V_x$,$V_y$) is first computed, and the $K$ nearest neighbors are kept. Here we focus on LLE and NMF-based neighbor embedding to combine the first $K$ candidates. Only the first steps of these approaches are used, which find constrained least squares approximation to the input patches from their $K$ nearest neighbors, hence the name of neighbor embedding.

**Locally linear embedding (LLE)** consists in approximating each input data point by a linear combination of its $K$-nearest neighbors under the constraint that the sum of the weights is equal to 1. One thus searches to approximate the known pixels of the patch $\Psi_p$ to be filled in by a linear combination of the co-located pixels in the K-nearest neighbors patches $\Psi_i, i = 1 \dots K$.

**Non-negative Matrix Factorization (NMF)** searches for two lower dimensional nonnegative matrices whose product gives a good approximation of the input data matrix [10]. The problem is once again a constrained least squares (LS) problem with constraints of nonnegativity of the component matrices. One matrix is fixed (it actually contains in its columns the vectorized $K$-NN patches, and only the other matrix (which is here a vector containing the weights of the linear combination) needs to be found. The most widely used solution for solving this constrained LS problem is the multiplicative update procedure [12], where the NMF problem is solved by iteratively updating the elements of each component matrix of the product. Here, we update only the matrix containing the weights $\alpha_{\mathbf{k}}$ of the linear combination as

$$\alpha_k \leftarrow \alpha_k \frac{\left(\mathbf{\Psi}^{\mathrm{T}}\psi_p\right)_k}{\left(\mathbf{\Psi}^{\mathrm{T}}\mathbf{\Psi}.\alpha\right)_k}. \quad (1)$$

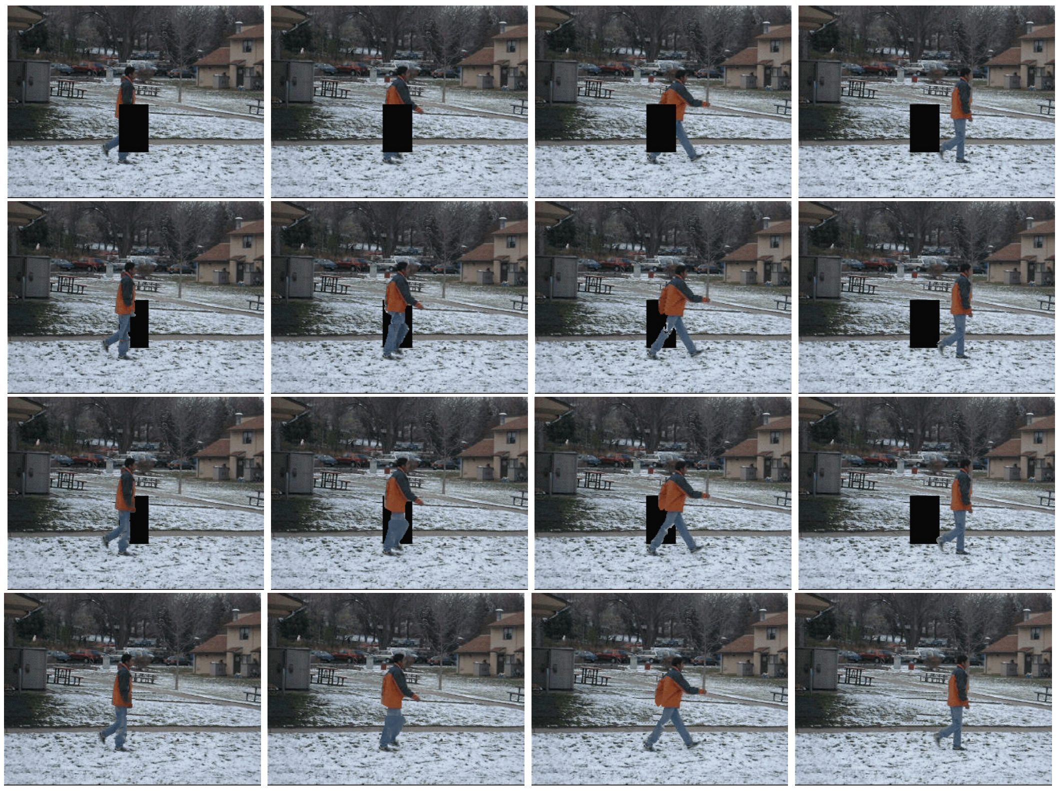

**Fig. 1**. Images from the original sequence to be inpainted (first row); Moving object inpainted with the TM-based algorithm of [7] (second row); Inpainted with the proposed algorithm after the moving object processing step (third row); Inpainted with the proposed algorithm after the three steps (last row). Here $M_c$ is assumed to be known.

where, $k$ represents the $k^{th}$ element of the vectors corresponding to the input or $K$-NN patches stored in the matrix $\Psi$.

**Competition between TM, LLE and NMF candidates:** Three estimates of the input patch to be completed are first computed using TM, LLE and NMF methods. The estimated patch having the lowest SSD with respect to the known pixels of the input patch is then retained. In average, the selection rate is about 75% for LLE and 25% for NMF. Note that TM is almost never selected. Beside the textural information, it is required to compute the motion information $M_c$. For the template matching, this is simply given by the motion information of the best candidate patch. For LLE and NMF that combine several patches, the motion information of the best approximated patch is set to the motion information of the neighbour that has the highest correlation in terms of $M_c$ with the known pixels of $\Psi_p$.

## 4. RESULTS

To disentangle the multiple problems of video inpainting, the analysis is first performed by assuming that the motion information $M_c$ is known also for the missing part in the video. As said above, this is not realistic in video editing but it is feasible in loss concealment with adequate transmission strategies. Fig. 1 shows four images of the input sequence to be inpainted, as well as results of the inpainting when using the original algorithm described in [7] (second row of the figure). It also shows in the third and last rows the images obtained after the first step and the three steps of the proposed algorithm. The proposed approach using a motion-compensated window and a competition between TM, LLE and NMF gives better results as illustrated by Fig. 2. This is also visible on the first inpainted picture of Fig. 1. Fig. 4 shows estimated patches computed by using TM, LLE and NMF for a given patch (bottom-right of this figure). Execution time is also significantly reduced with the proposed algorithm. Using a matlab implementation of the algorithm proposed in [7], the inpainting of the sequence presented in Fig. 1 (50 frames) takes 19 hours and 24 minutes while it takes only 55 minutes with the proposed approach using a motion-compensated searching window of $31 \times 31$ pixels.

The performance of the algorithms has then been assessed in the context where the motion information $M_c$ is not known for missing pixels. This information thus needs to be estimated along the texture inpainting process. For each inpainted patch, the $M_c$ values of the inpainted pixels are updated as described in section 3. Figure 3 shows the images

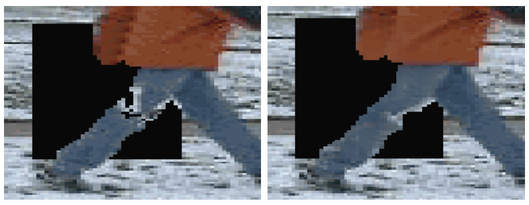

**Fig. 2**. Inpainted by using a template matching (left); inpainted with the proposed algorithm (right).

inpainted when the $M_c$ is unknown in the damaged region. One can observe that the $M_c$ update strategy is not sufficiently robust: an error in the motion information may lead to propagating the moving foreground in the stationary background.

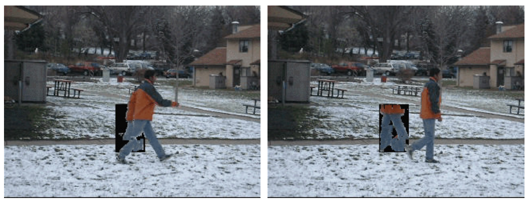

**Fig. 3**. Propagation effect when $M_c$ is unknown with the algorithm in [7] (left) and with the proposed algorithm (right).

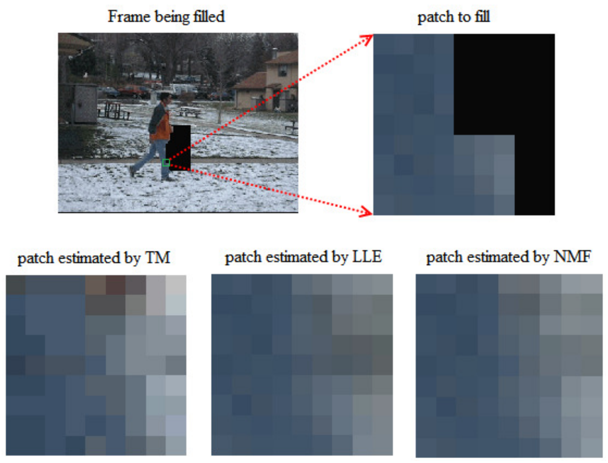

**Fig. 4**. Illustration of the proposed approach : frame being filled (top of the first column) and estimated patchs (bottom row) using template matching, LLE and NMF. Corresponding SSDs are 11064, 3295 and 4325 for TM, LLE, NMF respectively.

## 5. CONCLUSION

This paper describes a new examplar-based video inpainting algorithm based on motion-compensated neighbor embedding. The best approximation method among TM, LLE, and NMF is chosen locally according to an SSD measure. Results show improved performances when using the proposed inpainting scheme when the motion information $M_c$ is known. However, it has been observed that the algorithm, as its initial TM-based version, suffers from error propagation, especially in the first step of moving object inpainting. This propagation is due to uncertainties on the estimated motion information $M_c$. Future research needs to be dedicated to the design of a more robust update $M_c$ strategy.

## 6. REFERENCES

[1] M. Bertalmio, G. Sapiro, C. Ballester, and V. Caselles, “Image inpainting,” in *Proc. of SIGGRAPH*, pp. 417–424.

[2] G. Sapiro S. Rane and M. Bertalmio, “Structure and texture filling-in of missing image blocks in wireless transmission and compression applications,” vol. 12, no. 3, pp. 296–303, 2003.

[3] M. Bertalmio, A. Bertozzi, and G. Sapiro, “Navier-stokes, fluid dynamics, and image and video inpainting,” in *Proc. of Intl. Conf. on Computer Vision and Pattern Recognition, CVPR*, pp. 355–362.

[4] T. Chan and J. Shen, “Local inpainting models and tv inpainting,” *SIAM J. Appl. Math.*, vol. 62, no. 3, pp. 1019–1043, 2001.

[5] R. Bornard, E. Lecan, L. Laborelli, and J. Chenot, “Missing data correction in still images and image sequences,” in *ACM Intl. Conf. on Multimedia*, 2002.

[6] J. Sun, L. Yuan, J. Jia, and H.Y. Shum, “Image completion with structure propagation,” *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 861–868, 2005.

[7] G. Sapiro K. A. Patwardhan and M. Bertalmio, “Video inpainting of occluding and occluded objects,” in *Proc. of the IEEE Intl. Conf. on Image Processing, ICIP*, 2005, vol. 2, pp. 69–72.

[8] K. A. Patwardhan, G. Sapiro, and M. Bertalmio, “Video inpainting under constrained camera motion,” *IEEE Transactions On Image Processing*, vol. 16, no. 2, pp. 545–553, 2007.

[9] E. Shechtman Y. Wexler and M. Irani, “Space-time video completion,” in *Proc. of the Intl. Conf. on Computer Vision and Pattern Recognition, CVPR*, 2004, vol. 2, pp. 69–72.

[10] A. Criminisi, P. Perez, and K. Toyama, “Object removal by examplar-based image inpainting,” in *Proc. of Intl. Conf. on Computer Vision and pattern Recognition, CVPR*, 2003, pp. 721–728.

[11] S. Roweis and L. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *Science*, vol. 290, pp. 2323–2326, 2000.

[12] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” *Advances in Neural Information Process. Syst. (NIPS)*, 2000.

[13] M. Turkan and C. Guillemot, “Image prediction based on neighbor embedding methods,” *IEEE Trans. on Image Process.*, 2011.

## ACKNOWLEDGMENT

This work is supported by the ANR-09-VERS-019-02 grant (ARSSO project).