Graph-based deinterlacing

Jérôme Roussel, Pascal Bertolino

GRAPH-BASED DEINTERLACING

Jérôme ROUSSEL

ST Microelectronics S.A.
12 Rue Jules Horowitz B.P. 217
GRENOBLE - France

Pascal BERTOLINO

GIPSA-lab
Grenoble Institute of Technology
GRENOBLE - France

ABSTRACT

This article presents a new algorithm for spatial deinterlacing that could easily be integrated in a more complete deinterlacing system. The spatial interpolation process often fails to reconstruct edges that are diagonal or close to horizontal, leading to highly visible artifacts. Our solution aims at preserving the linear structure continuity. It connects pieces of horizontal edges coming from neighboring lines to build graphs which are then simplified in branches that represent the linear structure of the scene. These branches give the exact direction in which the interpolation has to be done. For the rest of the image, a traditional directional spatial interpolation gives satisfactory results already. Although the number of pixels interpolated with this method is relatively small, the overall image quality is subjectively well improved.

Index Terms— spatial deinterlacing, Laplacian extrema, graph, edge interpolation

1. INTRODUCTION

Interlaced video still is widely used by broadcasters. Interlacing has been chosen to reduce bandwidth in order to conciliate framerate and resolution. However, new flat panels like plasma or L.C.D are progressive ones and thus require the display of the whole image at time t. There is thus a high interest in deinterlacing methods that allow a conversion from interlaced to progressive. They can be classified in two major families, the methods without motion compensation and the ones with motion compensation [1, 2]. We will focus here on the methods without motion compensation and more precisely the adaptive methods. They consist in going towards the temporal method in areas where there is no movement and towards the spatial method in moving areas [3, 4, 5]. The spatial interpolation process has its own limitations, mainly on the rendering of edges close to horizontal [6, 7] and the technique proposed in this article addresses this particular point. In the first part, we are going to make a state of the art showing that the limitation mentioned cannot be easily overcome. Then, in a second part we will describe the different steps of our method, based on the construction and simplification of the Laplacian extrema. Finally, we will present our results.

2. EXISTING METHODS

Many methods and solutions have been proposed to perform spatial interpolation [8]. Majority of these methods try to make the interpolation along the direction of contours, using the so-called E.L.A method (Edge-based Line Averaging) [9]. This latter method detects the best direction Dir for interpolation within a window centered on the missing pixel and then makes the interpolation according to the found direction :

$$\tilde{f}(i,j) = \frac{f(i-1, j-Dir) + f(i+1, j+Dir)}{2}. \quad (1)$$

f represents the interlaced input image and $\tilde{f}$ the interpolated output image. (i,j) are the spatial coordinates where i represents the line position number and j the column position number. f is defined only for half of the lines, i.e. for i even or odd.

Although it leads to a better interpolation of contours, the method still has several limitations. Actually, the correlation is done at the local level and remains quite sensitive to noise. The edge direction thus happens to be wrong which can lead to very annoying artifacts since it disrupts the structure of thin lines or edges. Many alternatives of this method [10] make it possible to correct wrong direction interpolation for a majority of image pixels, for instance by computing the correlation between groups of pixels instead of pixel to pixel (figure 1). However, the results of these methods always remain dependent on the size of the window used, that determines the maximum angle allowed for the reconstruction of the contours. On the other side, the larger the window is, the higher the risk of bad interpolation [11]. Different existing features and metrics try to control an adequate window size and introduce weights to reduce the number of false directions [12, 13]. But methods used to calculate these weights significantly increase the complexity of the solution. Actually, all these alternatives only bring a final minor improvement and do not allow to reconstruct correctly lines and edges close to horizontal. This point is even more annoying in real time where moving horizontal lines do not only look disrupted, but also instable and highly flickering.

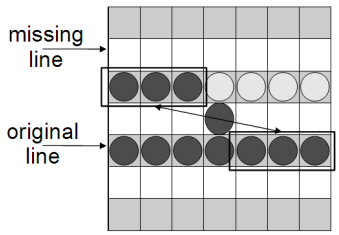

Fig. 1. Principle of modified E.L.A.

The method proposed here overcomes the limitation of the current directional spatial interpolation method since it is not based anymore on a searching window. Typically it overrules the wrong interpolation of edges close to horizontal, keeping the results of the traditional spatial interpolation where the results are already satisfactory. It can be added to a classical spatial deinterlacing method that can itself be integrated in a motion adaptive one.

3. EXTREMA EXTRACTION

With the existing methods, the objects that suffer a wrong reconstruction are the ones for which the contours are ramps or thin edges that are diagonally oriented, and the problem becomes stronger as the contour direction approaches the horizontal. Such contours are disconnected or aliased by the deinterlacing process. This shortcoming is all the more visible that the visual system is very sensitive to continuity. To detect the corresponding contour pixels, a vertical Laplacian filter (i.e since only the vertical direction is involved, this is the second derivative of the intensity in the vertical direction) is applied to the image : It removes the low frequencies and keeps the high frequencies in the vertical direction. The detection of the extrema is carried out locally on the known lines of the image by comparing the Laplacian value of a pixel $\Delta f(i,j)$ with its two neighboring values $\Delta f(i-2,j)$ and $\Delta f(i+2,j)$. Any resulting pixel has one of the three types : extremum (either minimum or maximum) or not extremum (figure 2, middle). A threshold T allows to reject extrema whose absolute value is too weak to be relevant. Let $\mathcal{H}$ be the set of maxima pixels and $\mathcal{L}$ the set of minima pixels :

$$\mathcal{H} = \left\{ \begin{array}{l} \Delta f(i,j) > 0 \\ \Delta f(i,j) > \max(\Delta f(i-2,j), \Delta f(i+2,j)) + T \end{array} \right\}$$

$$\mathcal{L} = \left\{ \begin{array}{l} \Delta f(i,j) < 0 \\ \Delta f(i,j) < \min(\Delta f(i-2,j), \Delta f(i+2,j)) - T \end{array} \right\}$$

On the same line, the extrema of the same type can form connected components (called segments in the sequel) according to the horizontal 2-connectivity. Since the continuation of the method is not founded on the traversing and the processing of pixels but on the traversing and the processing of segments, the traditional two-dimensional image structure is not suitable any more and is replaced with a higher level structure : namely the segment. Each segment will be a vertex of a graph and will be characterized by its coordinates (line, starting column), length, type (minimum or maximum) and neighborhood.

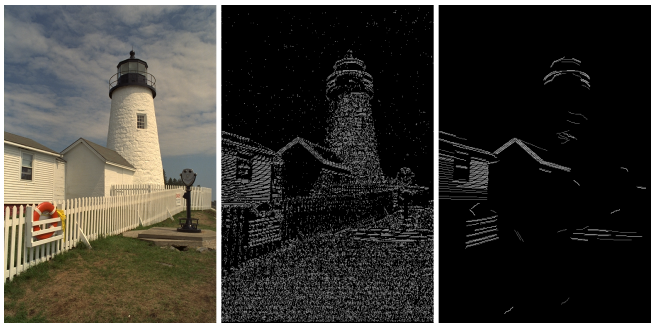

Fig. 2. Laplacian Extrema detection (middle) and the resulting branches (right)

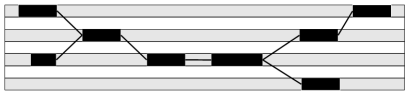

Fig. 3. Example of graph

4. GRAPH CONSTRUCTION

The chosen data structure is a compromise between the memory size needed and the complexity to traverse $\mathcal{H}$ and $\mathcal{L}$. From a data-processing point of view, it is important to note that the data structure used to store the graph is an array of rows (one row per known line of the image). Each row contains the chained list of the segments extracted in the corresponding line. Using chained lists is an easy and flexible way to handle any number of segments in each line, but a 2D array of segments could also be used.

In a first step, the image is browsed line by line and the data structure is filled with the segments and each segment is linked to its neighbors (figure 3). These links are the edges of the graph and are actually materialized by pointers.

A segment on a line i has at most 4 neighbors of the same type : NorthWest and NorthEast on line $i-2$ and SouthWest and SouthEast on line $i+2$. By definition of the Laplacian, neighbors of the same type situated North or South of the segment cannot exist. Neighbors situated on the same line as the segment are not taken into account because no interpolation will be needed between them since the content of line i already exists. A consequence is that the link between two segments has either the direction NorthWest to SouthEast (or $(\backslash)$) or NorthEast to SouthWest (or $(/)$).

The distance between two neighboring segments of the same type S_1 and S_2 is the Euclidean distance $d(S_1, S_2)$, cal-

culated between the closest extremities of S_1 and S_2. Two neighboring segments S_1, S_2 are linked if they are close enough compared to their respective length L_1, L_2, i.e if they satisfy the following condition :

$$d(S_1, S_2) < \lceil \min(L_1, L_2) * k \rceil \quad (2)$$

In our experiments, k is fixed to 0.7 which is the value of Kell factor.

5. GRAPH SIMPLIFICATION

Once the segments have been interconnected, depending on the image content and the threshold T, many complex graphs with cycles can be obtained, each one with many segments and links. Each graph must be split into the most likely branches, i.e in such a way that only the longest linear branches are kept since they represent contours in the image rather than noise or texture. For a given segment type, a branch is either a sequel of consecutive segments linked by $(\backslash)$ or a sequel of consecutive segments linked by $(/)$.

To perform the simplification process, each link must be aware of the length of its branch so that any link must be removed if needed, just with a local decision. To weight each link with the right length, two browsing passes are necessary in the array of rows. It is much more simple than a true graph traversal that would complicated to handle and time consuming. The first pass browses the array from top to bottom while the second pass does it from bottom to top (note that doing it conversely would provide the same result). In a given row each segment is processed independently, since the only information needed is the kind of links it has and their weight.

For instance, the total length of a $(\backslash)$ branch is obtained by starting from a segment with no NorthWest link, by following the $(\backslash)$ branch, and from 1, adding 1 to the branch length each time a segment is crossed. When the end of the branch is reached, the total length is known and must be propagated back to the whole branch during a second ascending pass.

Now the graphs are ready to be simplified : A segment may have 0, 1, 2, 3 or 4 links (figure 4). If it has zero or one link (it is alone in a graph or this is the extremity of a branch), there is nothing to do. If it has 2 links of the same direction, it belongs to one branch and there is nothing to do. If it has 2 links of different directions or if has 3 or 4 links, this means that it belongs to two branches. In these cases, among the 2, 3 or 4 links, the one(s) corresponding to the shorter branch is (are) removed (figure 2, right). A thresholding on the branch length can also be done to process the contours at a given resolution.

6. INTERPOLATION

The interpolation is carried out using a forward (from West to East) traversing of each remaining branch. Let S_1 and S_2 be two connected segments described by their lengths

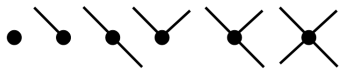

Fig. 4. The 6 link configurations (apart from a rotation)

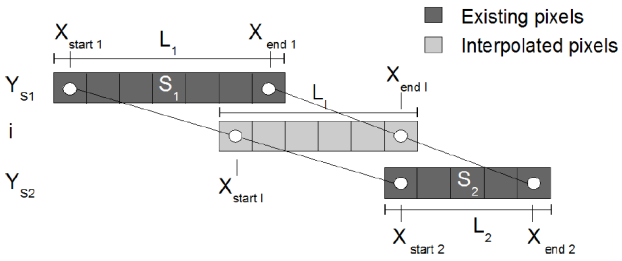

Fig. 5. Interpolation principle

L_1 and L_2, their starting coordinates (Y_{S_1}, X_{start_1}) and (Y_{S_2}, X_{start_2}). The pixels to be interpolated with our method correspond to the segment S_I whose extremities X_{start_I} and X_{end_I} are linearly interpolated from the extremities of the segments S_1 and S_2 (figure 5) :

$$X_{start_I} = X_{start_1} + \left| \frac{X_{start_2} - X_{start_1}}{2} \right| \quad (3)$$

$$X_{end_I} = X_{end_1} + \left| \frac{X_{end_2} - X_{end_1}}{2} \right| \quad (4)$$

$$\tilde{f}(i,j) = \frac{1}{2} f\left(Y_{S_1}, X_{start_1} + \left\lceil \frac{(j - X_{start_I}) \times (L_1 - 1)}{(L_I - 1)} \right\rceil\right) + \frac{1}{2} f\left(Y_{S_2}, X_{start_2} + \left\lceil \frac{(j - X_{start_I}) \times (L_2 - 1)}{(L_I - 1)} \right\rceil\right) \quad (5)$$
$$j \in [X_{start_I}, X_{end_I}]$$

L_I is the size of the segment to interpolate $X_{end_I} - X_{start_I} + 1$. Y_{S_1} and Y_{S_2} represent the ordinates $i - 1$ and $i + 1$ (or $i + 1$ and $i - 1$).

7. RESULTS

Our algorithm has been tested on different sequences and still images which have been interlaced. On the lighthouse image, the ELA method [12] does not rebuild the horizontal edges and a strong aliasing appears, while our method reconstructs the continuity of these edges (figure 6). In the Barbara image, the ELA method makes a wrong interpolation by being mistaken about direction of the trouser strips, while our method reconstructs them as the original image (figure 7). Note that the method is not curvature dependent and taht it can rebuilt straight or curved contours. However, differences between ELA and our method are not high on PSNR because only few pixels are modified as we can note in table 1. As far as processing time is concerned, the method only performs very simple computations and the amount of chained list data to be analyzed is rather small.

Fig. 6. Comparison of the original image (left column), ELA (middle column) and our method (right column)

Fig. 7. Comparison of the original image (left column), ELA (middle column) and our method (right column)

8. CONCLUSION

Our method is not based on the principle of existing methods. It thus does not suffer from their limitations. Our method aims at correcting the most unpleasant artifacts for the human eye by detecting them directly. It is based on the continuity of object borders in order to reconstruct them. Finally, our method can be added to all the traditional methods to improve their weak point without a high added cost.

9. REFERENCES

[1] Kenji Sugiyama and Hiroya Nakamura, "A method of de-interlacing with motion compensated interpolation," *Consumer Electronics, IEEE Transactions on*, vol. 45, no. 3, pp. 611–616, 1999.

[2] Kenji Sugiyama, Yoshiyuki Yamada, and Naoya Sagara, "Improvement of motion compensated inter-field interpolation method for de-interlacing," in *TENCON 2006. 2006 IEEE Region 10 Conference*, Nov. 2006, pp. 1–4.

[3] Tero Koivunen, "Motion detection of an interlaced video signal," *Consumer Electronics, IEEE Transactions on*, vol. 40, no. 3, pp. 753–760, 1994.

Sequences	Size	% of detected pixels	Number of segments	Interpolated pixels	
				Number	% of the image
Lighthouse	768 × 512	7.44	14613	6927	1.76
Barbara	576 × 720	5.64	17161	11242	2.71
Starwars	576 × 720	1.94	1797	3125	0.75
Table Tennis	480 × 720	4.2	6035	8493	2.46
Means		4.8	9901	7446	1.92

Table 1. Number of pixels and segments interpolated by our method for one still image and several video sequences

[4] Shyh-Feng Lin, Yu-Ling Chang, and Liang-Gee Chen, "Motion adaptive interpolation with horizontal motion detection for deinterlacing," *Consumer Electronics, IEEE Transactions on*, vol. 49, no. 4, pp. 1256–1265, 2003.

[5] Lejun Yu, Jintao Li, Yongdong Zhang, and Yanfei Shen, "Motion adaptive deinterlacing with accurate motion detection and anti-aliasing interpolation filter," *Consumer Electronics, IEEE Transactions on*, vol. 52, pp. 712–717, 2006.

[6] Assaf Almog, Avi Levi, and Alfred M. Bruckstein, "Spatial de-interlacing using dynamic time warping," *Image Processing, IEEE Transactions on*, vol. 2, pp. II–1010–13, 2005.

[7] C. Ballester, M. Bertalmío, V. Caselles, L. Garrido, A. Marqués, and F. Ranchin, "An inpainting based deinterlacing method," in *accepted for publication in IEEE Transactions on Image Processing*, 2007.

[8] G. de Haan and E.B. Bellers, "Deinterlacing - an overview," *Proceedings of the IEEE*, vol. 86, no. 9, pp. 1839–1857, 1998.

[9] T. Doyle, "Interlaced to sequential conversion for edtv applications," in *2nd International Workshop Signal Processing of HDTV*, 1988, pp. 412–430.

[10] Tao Chen, Hong Ren Wu, and Zheng Hua Yu, "Efficient deinterlacing algorithm using edge-based line average interpolation," *Optical Engineering*, vol. 39, pp. 2101–2105, september 2000.

[11] Hoon Yoo and Jechang Jeong, "Direction-oriented interpolation and its application to de-interlacing," *Consumer Electronics, IEEE Transactions on*, vol. 48, pp. 954–962, 2002.

[12] Min Kyu Park, Moon Gi Kang, Kichul Nam, and Sang Gun Oh, "New edge dependent deinterlacing algorithm based on horizontal edge pattern," *Consumer Electronics, IEEE Transactions on*, vol. 49, no. 4, pp. 1508–1512, 2003.

[13] Min Byun, Min Kyu Park, and Moon Gi Kang, "Edi-based deinterlacing using edge patterns," *IEEE International Conference on Image Processing*, vol. 2, pp. 1018–1021, 2005.