# High-Quality Spatial Interpolation of Interlaced Video

Alexey Lukin
Laboratory of Mathematical Methods of Image Processing
Department of Computational Mathematics and Cybernetics
Moscow State University, Moscow, Russia
lukin@graphics.cs.msu.ru

## Abstract

Deinterlacing is the process of converting of interlaced-scan video sequences into progressive scan format. It involves interpolating missing lines of video data. This paper presents a new algorithm of spatial interpolation that can be used as a part of more complex motion-adaptive or motion-compensated deinterlacing. It is based on edge-directional interpolation, but adds several features to improve quality and robustness: spatial averaging of directional derivatives, ”soft” mixing of interpolation directions, and use of several interpolation iterations. High quality of the proposed algorithm is demonstrated by visual comparison and PSNR measurements.

**Keywords:** *deinterlacing, edge-directional interpolation, intra-field interpolation.*

## 1 INTRODUCTION

Interlaced scan (or *interlacing*) is a technique invented in 1930-ies to improve smoothness of motion in video without increasing the bandwidth. It separates a video frame into 2 *fields* consisting of odd and even raster lines. Fields are updated on a screen in alternating manner, which permits updating them twice as fast as when progressive scan is used, allowing capturing motion twice as often. Interlaced scan is still used in most television systems, including certain HDTV broadcast standards.

However, many television and computer displays nowadays are based on LCD or plasma technologies. They cannot benefit from interlacing, as they are not using a raster scan to form an image. To display video on such systems, it has to be converted to progressive scan format. This process is known as *deinterlacing*.

In old-style CRT television displays, ”deinterlacing” is performed by the viewer’s vision smoothing properties. To achieve better display quality for the same video material on modern LCD or 100 Hz displays, deinterlacing should be performed in hardware.

Converting interlaced video to progressive-scan video requires *interpolating* a set of missing lines in every video field. Two trivial approaches exist in deinterlacing: *”Bob”* method interpolates every missing line by averaging 2 adjacent lines in the same field; *”weave”* method inserts missing lines from the previous field (that has a complementary parity). Problems with such simple methods are obvious: ”Bob” reduces vertical resolution of video, produces jagged edges, and suffers from flickering effect due to alternation of interpolated lines of different parity (Fig. 1a). ”Weave” produces a temporal mismatch of 2 fields combined in a single frame, resulting in ”comb” (or ”tearing”) artifacts on moving objects (Fig. 1b).

More advanced methods include *motion-adaptive* and *motion-compensated* deinterlacing [1]. Motion-adaptive methods detect presence of motion in video sequence and use temporal interpolation (”weave”) in still areas to retain full vertical resolution whereas a spatial interpolation (e.g. ”Bob”) is used near moving objects to prevent comb artifacts. Motion-compensated methods track the direction and speed of moving objects to interpolate missing lines in 3D spatiotemporal space. Since true motion data is usually unavailable, it is estimated from the video sequence.

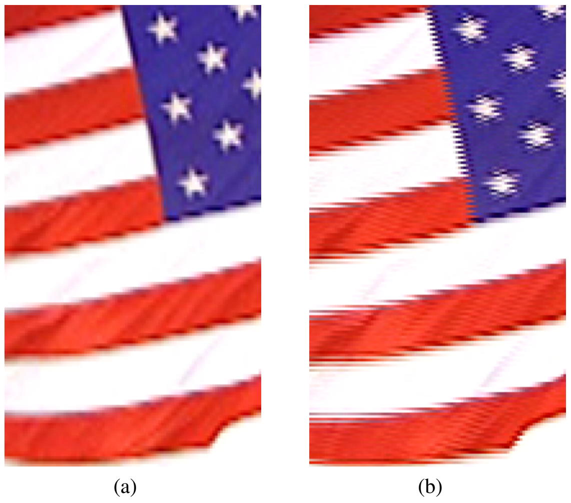


**Figure 1:** a: ”Bob” deinterlacing (line averaging),
b: ”weave” deinterlacing (field insertion).

In this paper, a new high-quality method of spatial interpolation of video frames in suggested. It is based on widely known ELA (edge-based line averaging) [2] methods, but introduces several improvements resulting in higher image quality and better robustness:

- spatial averaging of directional derivatives;
- ”soft” mixing of interpolation directions;
- use of several interpolation iterations with re-estimation of derivatives from a full-resolution frame.

This interpolation method can be used as a part of motion-adaptive or motion-compensated deinterlacing methods.

The rest of the paper is organized as follows. In Section 2, the prior art is reviewed and basic spatial interpolation algorithms are explained. In Section 3, the proposed algorithm is described. Section 4 shows image examples and PSNR figures for the proposed algorithm. Section 5 concludes the paper.

## 2 PRIOR ART

The simplest methods of spatial interpolation (”Bob”) just duplicate lines of the field to fill in the missing lines or perform linear (or cubic) averaging of 2 (or 4) nearest pixels. This often results in an artifact known as *jagged edges* (Fig. 1a) and it is especially noticeable where vertical direction of interpolation doesn’t match with local direction of edges. A well-known solution to the problem of jagged edges is *edge-directional interpolation* methods. Such methods estimate the local edge direction and apply interpolation along this direction.

A simple edge-directional interpolation for deinterlacing is known as ELA (edge-based line averaging) [2]. It calculates 3 or 5 directional derivatives of the image around the central pixel (Fig. 2). For color images, all 3 color channels can be differenced and their differences being added for coherent interpolation across color channels.

$$D_d = |I_{x+d,y-1} + I_{x-d,y+1}| \quad (1)$$

Here $d \in [-2, 2]$ is the direction of differencing. The direction of the least absolute derivative is chosen as the local edge direction, and interpolation is performed by averaging 2 pixel values along this direction.

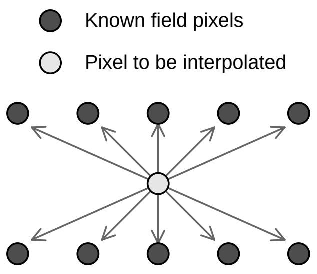


**Figure 2:** Aperture of 5-directional ELA method.

One of the problems with this approach is the lack of robustness in presence of thin lines. When a directional derivative is evaluated across a thin line, it can take a low value if image to the both sides of the line has the same color (Fig. 3b). This may lead to erroneous decision on the direction of interpolation and produce artifacts (Fig. 6a, 7b).

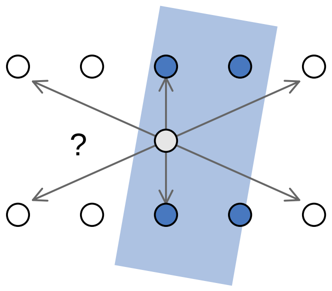


**Figure 3:** Uncertainty of interpolation direction in ELA method.

Another method known as EDDI (edge-dependent deinterlacing) [3] processes a video field with a combination of a high-pass and low-pass filters to produce the edge mask. The analysis of edge mask zero crossings in adjacent lines leads to the estimate of the local edge orientation. This orientation is used for interpolation.

Several adaptations of conventional image interpolation methods to the problem of deinterlacing are discussed in [4].

## 3 THE PROPOSED ALGORITHM

A set of modifications to the ELA algorithm is proposed in this paper, which significantly improves its quality and robustness to noisy or complex video data.

### 3.1 Interpolation aperture

The first modification extends the number of interpolation directions from 3 or 5 to 17 (this number can be an algorithm quality parameter). Consideration of high number of directions allows for smooth interpolation of edges that are very close to horizontal (Fig. 4). However, consideration of directional derivatives between points that are up to 16 pixels apart can lead to errors similar to errors of ELA in presence of thin lines.

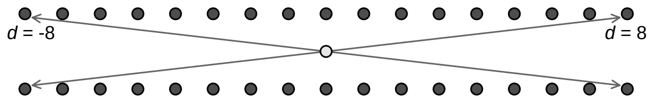


**Figure 4:** Aperture of the proposed method
(only 2 of 17 possible interpolation directions are shown).

### 3.2 Use of full-resolution image

Two modifications are made to improve the robustness of the interpolation of near-horizontal edges. The first one suggests calculating directional derivatives from an interpolated frame instead of the original field. This requires having some initial estimate of the interpolated frame, which can be obtained by a simple line averaging (linear or cubic). When the pre-interpolated frame is available, directional derivatives can be calculated between its adjacent lines; this reduces distances between differenced pixels by a factor of 2.

### 3.3 Averaging of derivatives

The next modification improving the robustness of interpolation is spatial averaging (smoothing) of directional derivatives. It is proposed to increase the radius of such averaging as the distance between differenced pixels increases. The rationale for this decision is that differencing of distant pixels is more prone to the "problem of thin lines", and thus requires more averaging to ensure that no lines of the opposite direction are crossing the interpolated area. Such a spatial averaging can be effectively performed by a box or triangular filter. A higher-quality alternative is the use of smoother Hann filter [5]. The suggested filter radius depends on the differencing direction $d$, $-8 \le d \le 8$ as follows:

$$R(d) = round\left(a + b|d|^{\gamma}\right) \quad (2)$$

The constants used in the preferred implementation are $a = 0.6$, $b = 0.8$, $\gamma = 3/2$.

### 3.4 Interpolation

When the derivatives are spatially averaged, interpolation weights for each of 17 interpolation directions are calculated as

$$W_d = k\left(\frac{M_d}{\max\{0.01, D_d\}}\right)^8 \quad (3)$$

Here $k$ is the normalizing constant to ensure $\sum_{d=-8}^{8} W_d = 1$, $M_d$ is the set of constant weights for biasing the algorithm toward more cautious interpolation in directions close to horizontal, $D_d$ is the smoothed directional derivative along direction $d$. In the preferred implementation, $M_d$ are calculated as follows:

$$M_d = \exp\left(-c\,|d|\right) \quad (4)$$

Here $c$ is the parameter adjusting the bias toward vertical or horizontal directions, in the preferred implementation $c = 0.12$.

After interpolation weights are calculated, the interpolation is performed as follows:

$$I_{x,y} = \sum_{d=-8}^{8} 0.5 W_d \left( I_{x+d,y-1} + I_{x-d,y+1} \right) \tag{5}$$

In this way, ”hard” switching of interpolation directions is replaced with a ”soft” decision. This helps interpolating edges with intermediate slopes between fixed interpolation directions. For example, for the edge in Fig. 3, the resulting interpolation direction will be a mixture of $d = 0$ (larger weight) and $d = 1$ (smaller weight), with other weights being close to zero.

## 3.5 EM algorithm

The next modification is improving the interpolation quality by iterating the algorithm after the first interpolation is done. Since the first iteration estimates directional derivatives from the roughly interpolated image, these derivatives are not accurate, which leads to a sub-optimal interpolation result. Re-estimating directional derivatives after the first iteration of interpolation provides a better estimate of directional derivatives of a full-resolution image, and this leads to improved interpolation quality after next iterations, according to Expectation Maximization (EM) strategy. In the proposed implementation, 2 iterations have provided good quality, with only little change after additional iterations.

The overall conceptual work flow of the algorithm is presented in Fig. 5.

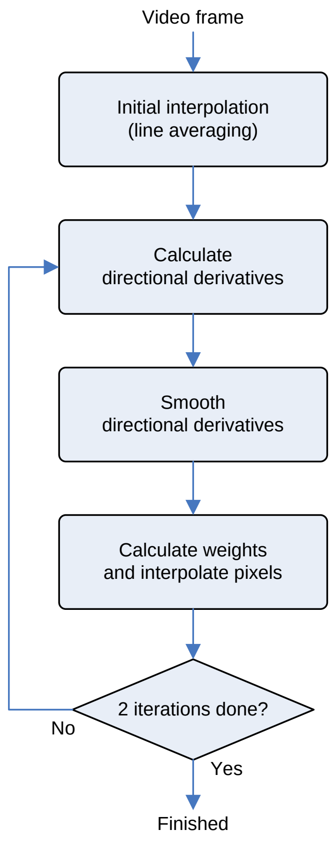

Video frame
Initial interpolation (line averaging)
Calculate directional derivatives
Smooth directional derivatives
Calculate weights and interpolate pixels
2 iterations done?
No
Yes
Finished

**Figure 5:** Flowchart of the proposed algorithm.

## 3.6 Optimization of complexity

The computational complexity of the proposed algorithm is significantly higher than of most prior art methods: on a 2 GHz Pentium machine it achieves 3 fps processing speed for CIF video. However several optimizations are possible.

Firstly, interpolation stage can be simplified to selecting only one most probable interpolation direction instead of mixing interpolation along all directions with different weights.

Another optimization possibility comes from the fact that calculation of directional derivatives followed by a box-filter averaging is equivalent to a well-known block matching algorithm, which can be effectively implemented in hardware.

Finally, it can be noted that in most cases interpolation weights are spatially smooth. This allows for sparse recalculation of interpolation weights, depending on the smoothing filter radius.

## 4 RESULTS

Evaluation of visual quality of the proposed algorithm in comparison to other existing methods has been performed on various video test sequences. In addition to published and referenced interpolation algorithms, several commercially available programs have been evaluated. In Fig. 6, 7, a visual comparison of resulting images is given for a few methods. More comparisons and sample images are available in [6].

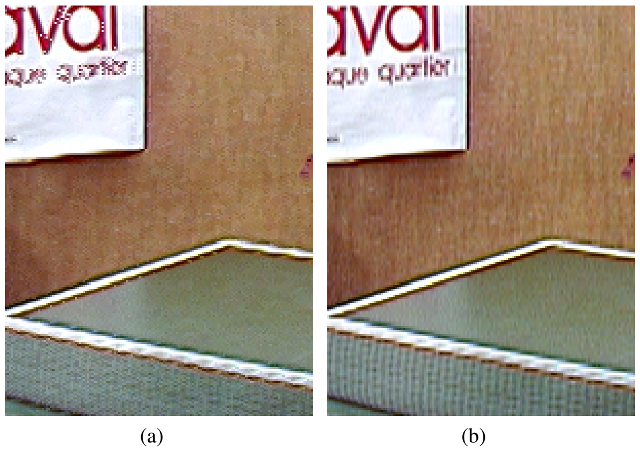
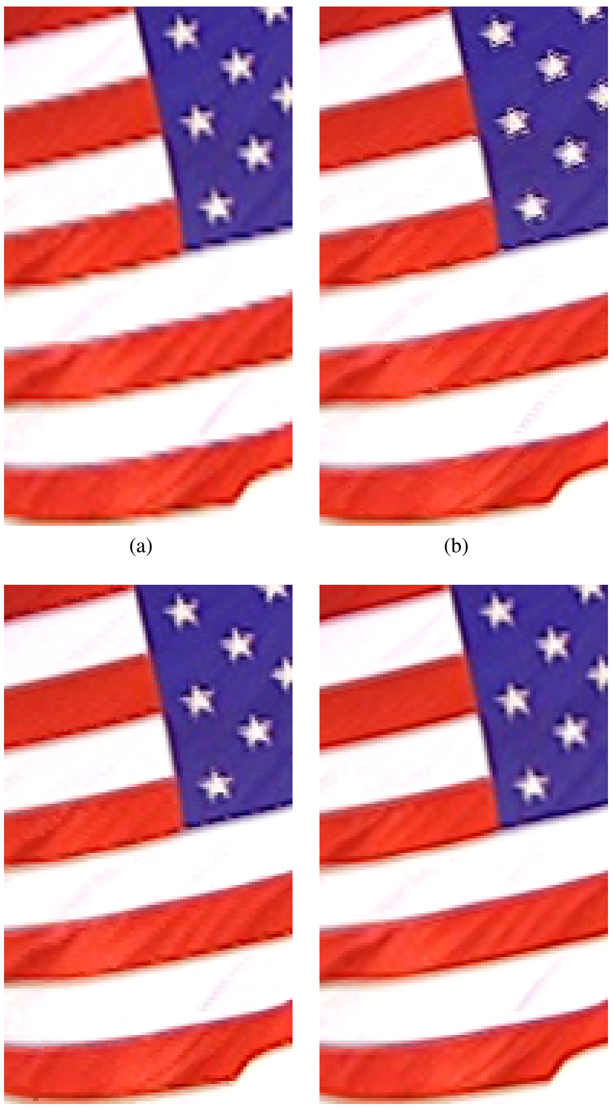


**Figure 6:** a: ELA 5 method, b: Proposed method.

It can be seen that a simple non-adaptive line averaging produces noticeable jagged edges on flag stripes, but works well on small objects, like stars (Fig. 7). Simple edge-directional method ELA 5 fails to correctly resolve true edge direction in many cases, which leads to spurious interpolated pixels near fine details and thin lines (poster text, tennis table and net in Fig. 6). The result of EDDI looks much better with only few erroneous pixels at thin lines. The result of the proposed method shows no visible artifacts.

A formal PSNR evaluation has also been performed on a set of 8 popular test images that were artificially interlaced. Fig. 8 shows that ELA 3 and ELA 5 methods have a worse PSNR performance than line averaging due to incorrectly resolved edge directions. However EDDI method and the proposed algorithm have better performance. The proposed algorithm provides up to 2 dB improvement over the line averaging method, depending on image complexity and presence of fine details.



(a) (b)

(c) (d)

**Figure 7:** a: Line averaging, b: ELA 5 method, c: EDDI method, d: Proposed method.

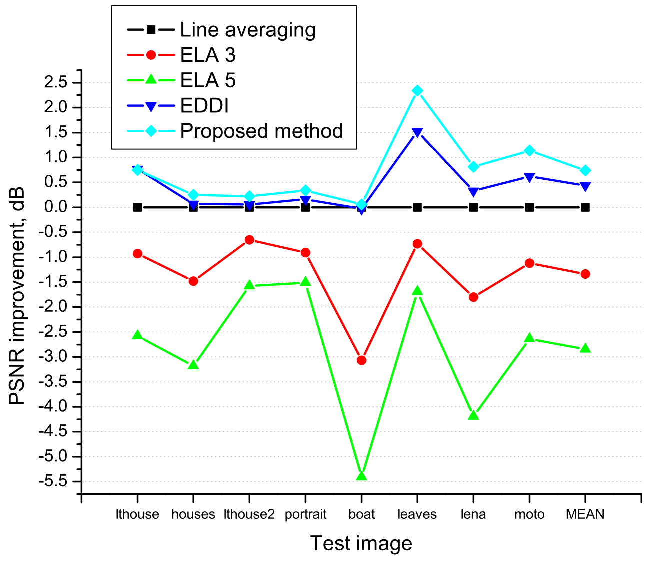


**Figure 8:** PSNR plot for a set of test images, referenced to the line averaging method.

## 5 CONCLUSION

The described algorithm for intra-field interpolation shows good results in video deinterlacing. Spatial averaging of directional derivatives and mixing of different interpolation directions allow achieving high-quality interpolation of near-horizontal edges, and at the same time preventing spurious pixel artifacts that are common to other methods. Demonstrated visual improvement on real-world interlaced video sequences is supported by PSNR improvement on artificially interlaced images. The proposed spatial interpolation algorithm is used as a part of a motion-compensated video deinterlacing method.

## Acknowledgements

This work has been supported by RFBR grant 06-01-39006-GFEN. Author thanks Denis Kubasov for introduction to LaTeX and many comments and suggestions on the text.

## References

[1] G. de Haan, E. Bellers ”De-interlacing — An Overview” *Proceedings of the IEEE*, vol. 86, Issue 9, pp. 1839–1857, Sep. 1998.

[2] T. Doyle, M. Looymans ”Progressive Scan Conversion using Edge Information” *Signal Processing of HDTV*, II, Elsevier Science Publishers, 1990, pp. 711–721.

[3] G. de Haan, R. Lodder ”De-interlacing of Video Data using Motion Vectors and Edge Information” *Digest of the ICCE'02*, pp. 70–71, June 2002.

[4] M. Zhao, G. de Haan ”Intra-field De-interlacing with Advanced Up-scaling Methods” *Proc. of the ISCE*, Sep. 1–3, 2004, Reading, UK, pp. 315–319.

[5] A. Lukin ”Tips&Tricks: Fast Image Filtering Algorithms” *Proceedings of GraphiCon'2007*, Moscow, Russia, June 2007, pp. 186–189.

[6] Demo web-page with sample images http://imaging.cs.msu.ru/ ˜lukin/deinterlacing.html

## About the author

Alexey Lukin, Ph.D., is a member of scientific staff at Moscow State University, Department of Computational Mathematics and Cybernetics, working in Graphics & Media Lab and Laboratory of Mathematical Methods of Image Processing. His scientific interests include image processing, audio processing, adaptive ways of spectral analysis and multiresolution filter banks.

Email: lukin@graphics.cs.msu.ru