Review on Deinterlacing Algorithms

Karthik[1] , Vinay Varma B[2] ,
Akshay Narayan Pai[3]

Electronics and Communication
Dept.[1,2,3,4] R.V College of
Engineering , Bangalore, India

karthik.ec17@rvce.edu.in[1], vinayvarmab.ec17@rvce.edu.in[2], akshaynarayanp.ec17@rvce.edu.in[3]

Abstract: *Interlacing is a commonly used technique for doubling the perceived frame rate without adding bandwidth in television broadcasting and video recording. During playback, however, it exhibits disturbing visual artefacts such as flickering and combing. As a result in modern display devices, video deinterlacing is used where the interlaced video format is converted to progressive scan format to overcome the limitations of interlaced video. This conversion is achieved through interpolating interlaced video. Current deinterlacing approaches either neglect temporal information for real-time performance but poor visual quality, or estimate motion for better deinterlacing but higher computational cost.*
This paper focuses on surveying the deinterlacing algorithms which apply both spatial and temporal based methods and focus on different aspects of both motion-adaptive, non-motion adaptive and the time complexity through these implementations.

Keywords: Motion Adaptive Deinterlacer, Interlacing, IPP, Edge based line averaging, DCS, LCID,

1. INTRODUCTION

The ever-evolving technology manifests its growth in almost every walk of life. The television industry has witnessed transformation with advancement in technology and time. In the early nineteenth century, analogue televisions used interlaced video format displays. Now, the current technology televisions use uses progressive displays.

Analogue televisions, each picture frame is made up of two fields that do not occur at the same time, saving bandwidth. Thus, to save bandwidth interlacing was introduced into video signals. Recordings were made, played back, and transmitted using progressive or interlaced techniques. Because of its versatility and durability, interlaced has its origins in the broadcasting industry and is still commonly used today. Thus, deinterlacing techniques are used to transform interlaced content to progressive video to enable today's progressive display technologies. Without appropriate deinterlacing, some defects such as edge-flickering , crawling lines will result in unappealing visual artefacts.

As video productions transition from analogue to digital, more and more video production equipment will be required. Interlaced formats are used in today's analogue television standards, such as NTSC, PAL, and others. The market for progressive content will rise as these requirements move to digital, triggering a direct proportional rise in the demand for high-quality deinterlacing solutions that convert interlaced frames to progressive.

The various deinterlacing techniques, which are typically classified to two key branches: motion compensated and non-motion compensated. The best reconstruction efficiency is achieved using the motion compensated (MC) process, but it is computationally more costly due to the pixel shifting computations and measurement of a two-dimensional motion vector. Non-motion compensated (non-MC) processes, on the other hand, are less expensive and can strike a reasonable balance between efficiency and sophistication.

Intel Integrated Performance Primitives (Intel IPP) is a large collection of ready to use functions which are domain specific and are well tuned in for variety of Intel architectures. Its royalty-free APIs assist developers in the following ways:

1. Use SIMD (Single Instruction, Multiple Data) instructions to his advantage.
2. Signal processing, data compression, video processing, and cryptography all benefit

from improved performance.

3. Reduce software development and maintenance expenses and time to market.

IPP mainly supports four different applications. These applications are image processing, signal processing, data compression and cryptography.

2. Review

Different criterias and starting points lead to different classification of deinterlacing algorithms. A widely known way to classify deinterlacing algorithms as Inter-field spatial and intra-field temporal deinterlacing based on the property of the use of the neighbouring fields to consider for interpolation to get the final deinterlaced frame. Because of their convenience, intra-field deinterlacing algorithms are commonly implemented in real-time applications. Motion adaptive and motion compensation (MC) algorithms are used in inter-filed temporal deinterlacing.

3. Inter-Field Spatial Deinterlacing

Intra-field spatial interpolation is another basic method of interpolation deinterlacing where missing lines are estimated from lines present in the current field which is being displayed. Line Averaging is the simplest algorithm from this family. The advanced version of this is the Edge based line averaging which performs border search of image to perform spatial interpolation. The major advantage of this algorithm is that it produces very less artifacts for motion frames of the video. But it reduces the vertical resolution because it uses only the current field information for estimation

3.1 Line Averaging (LA)

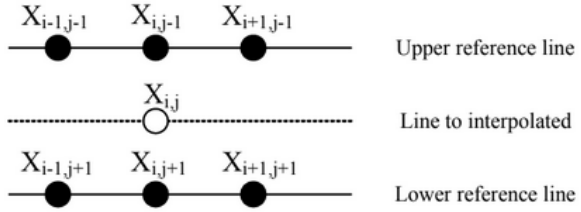
The line average technique is a basic interpolation technique where the pixel is interpolated by taking the average value of the pixels situated up and down of it. This line averaging technique is commonly employed because of its simplicity and clarity, and also there are no motion artefacts. However, before picture is interpolated, the vertical part of the resolution of the picture is cut in half on the input side, resulting in a loss in the quantity of features in the final image.[1] The below formula represents [10]

$$\square_{\square,\square} = \frac{\square_{\square,\square-1} + \square_{\square,\square+1}}{2} \quad (1)$$

3.2 Edge-Based Line Averaging (ELA)

The edge based line average (ELA) approach, which interpolates a line missed in-between neighbours in an interlaced pictures using edge-directional correlation to avoid this problem. Edge based line average is a well known intra field deinterlacing technique based on edges that divides edge directions into 45° - 90° - 135° degrees before interpolating linearly in desired direction. Edge based Line Averaging is a simple algorithm that may be incorporated to reduce the blurry effect of Line Averaging in locations where the edges are accurately approximated. Approaching in this manner, on the other hand, performs poorly in high spatial frequency boundries and may overestimate the detection of edges, resulting in picture deterioration.[2] The implementation of it is as shown below:

Figure 1. Edge based Line Avergaing



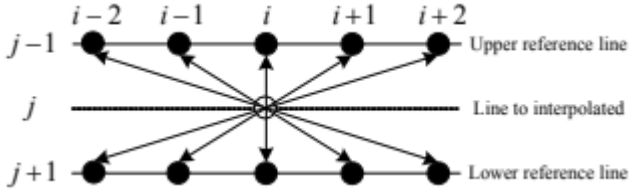
$$\begin{aligned} \square &= |\square_{\square-1,\square-1} - \square_{\square+1,\square+1}| \\ \square &= |\square_{\square,\square-1} - \square_{\square,\square+1}| \\ \square &= |\square_{\square-1,\square+1} - \square_{\square+1,\square-1}| \end{aligned} \quad (2)$$

$$\square_{\square,\square} = \begin{cases} \dfrac{\square_{\square-1,\square-1} + \square_{\square+1,\square+1}}{2} & \square\square\ \square\square\square(\square,\square,\square) = \square \\ \dfrac{\square_{\square,\square-1} + \square_{\square,\square+1}}{2} & \square\square\ \square\square\square(\square,\square,\square) = \square \\ \dfrac{\square_{\square+1,\square-1} + \square_{\square-1,\square+1}}{2} & \square\square\ \square\square\square(\square,\square,\square) = \square \end{cases} \quad (3)$$

3.3 Efficient ELA and Modified ELA

In order to overcome these limitations, a number of unique techniques have tried to solve based on ELA algorithm approach. Efficient edge based line average (EELA) [5] adds helpful metrics to approach more effectively determining the spatial correlation with direction. There have also been approaches for deinterlacing algorithms based on analysis of the content [6] or the horizontal edge pattern [7]. Modified ELA[6] was proposed with 3D spatial correlations that take in consideration vertical, anti-diagonal. MELA can be modified based on the number of diagonal elements considered as shown. It can range from 3 to 6 diagonals and the more the diagonals the more will be the accuracy but in turn the complexity also increases.

Figure 2. Modified ELA



3.4 Low-Complexity Interpolation technique for Deinterlacing (LCID)

The low complexity interpolation technique for de-interlacing (LCID) [6] employs four of the simple directional differences than MELA: diagonal, horizontal, anti-diagonal and vertical.
In this algorithm first the correlation is estimated between the spatial directions and then the interpolation is performed on the pixels which are missing further. Assuming x(i, j) denotes the pixel which will be interpolated, where j is the number of vertical lines present and i is the horizontal lines present. For the purpose of checking the existence of an edge the horizontal, vertical and diagonal directions accordingly, as Dh, Dv and Dd1, Dd2. The least value among them will have the strongest correlation which will be considered for the calculation of the missing interpolated pixel value as follows.[9]

$$\begin{aligned} \square_{\square 1} &= |\square(\square-1,\square-1) - \square(\square+1,\square)| + |\square(\square-1,\square) - \square(\square+1,\square+1)| \\ \square_{\square 2} &= |\square(\square-1,\square) - \square(\square+1,\square-1)| + |\square(\square-1,\square+1) - \square(\square+1,\square)| \\ \square_{\square} &= |\square(\square-1,\square) - \square(\square+1,\square)| \square 2 \\ \square_{\square 1} &= |\square(\square-1,\square-1) - \square(\square-1,\square)| + |\square(\square-1,\square-1) - \square(\square+1,\square)| \end{aligned} \quad (4)$$

$$\square = \square\square\square(\square\square 1, \square\square 2, \square\square)$$

$$\square(\square,\square) = \begin{cases} \square(\square-1,\square-1), & \square\square\ \square_{\square} = 0 \\ (\square(\square-1,\square-1) + \square(\square+1,\square) + \square(\square-1,\square) + \square(\square+1,\square+1))/4, & \square\square\ \square = \square_{\square 1} \\ (\square(\square-1,\square) + \square(\square+1,\square-1) + \square(\square-1,\square+1) + \square(\square+1,\square))/4, & \square\square\ \square = \square_{\square 2} \\ (\square(\square-1,\square) + \square(\square+1,\square))/2, & \square\square\ \square = \square_{\square} \end{cases} \quad (5)$$

4. Inter-Field

Inter-field temporal interpolation is one of the simplest families of deinterlacing algorithms. It uses the information of adjacent frames to estimate the value for missing lines. The direct method of doing this is to replicate the previous line. This method produces the highest resolution output when there is no movement. But produces severe artifacts when there are large movements.

4.1 Motion Adaptive Deinterlacing

Motion adaptive family raised because of the differences between Temporal and spatial interpolation deinterlacing methods. Spatial method produces good output only for the motion part of frames and the Temporal method for the static part of the frame. So motion adaptive deinterlacing emerged with hybridisation of both of these methods. This method uses a method to identify the pixel variation between the consecutive fields. And uses the variation value to choose between temporal and spatial interpolation. Even this algorithm produces loss in vertical resolution for moving objects in the video which is significantly less than when using spatial interpolation method only. Motion adaptive deinterlacing is a widely used algorithm.

4.2 Motion Compensated Deinterlacing

Motion compensated algorithms use motion vector calculations for estimating movement between two consecutive fields. This algorithm performs well when there are only a few pixel movements in the image and movement vectors are estimated well, which happens for texture images with high vertical resolution. If movement is for a large number of pixels then then movement vectors can produce artifacts which cause visible artifacts. For this reason the motion compensated method is combined with other methods to reduce errors in movement vector calculations and reduce visible artifacts in cases where large numbers of pixels produce motion. This method uses significantly more computation power for better output. These basic methods are discussed in [11] the book.

4.3 Vertical Temporal Filtering

Temporal filters take the information of neighbouring pixels of adjacent frames to obtain the pixel value of the interpolated region. This method achieves best output when there is little movement in the video. But when there are large movements, the field replication method can generate artifacts that are highly visible. In the case of a vertical temporal filter we consider the vertical neighbour fields and make the contribution from them to be limited by high frequency vertically.This makes it avoid artifacts in a smooth motion region. Thus this filter is used for the motion smooth region pixel interpolation. The vertical-temporal filter uses the below formula for calculating the value of the interpolated region.

$$\square'_{(\square,\square)} = \frac{1}{18}\left(\sum_{\square=-3}^{\square=3} \square_{(\square,\square+\square)} \ℎ(\square)\right) \quad (6)$$

where h(k) ={1,-5,8,10,8,-5,1} for k ={-3,-2,-1,0,1,2,3} image

4.4 3-D Edge based Line Averaging

To find the interpolated pixel in the motion texture region 3D ELA is used. ELA uses directional corelation to interpolate the pixel missing in the current field. Edge-based Line Averaging is extended to three dimensions for obtaining the motion information for use in deinterlacing methods. 3D ELA follows the same procedure as that of novel ELA but with slight difference in consideration of pixel pairs. First find the absolute difference between pairs of pixels as per the equation in the temporal domain. This makes 6 values for one interpolated

pixel. Now in the spatial domain calculate the absolute difference for 3 pairs of pixels using the equation. The interpolation direction is that of the pair of pixels having the lowest absolute difference value. The final interpolated pixel value is the average of those values of pixels along the interpolated direction.

5. Edge-Slope Tracing

Pre-existing edge-based interpolation techniques are generally either overly complicated or have problem towards gently sloped edges. To solve this a larger mask will be required to detect both interpolate and slope. Leading to burdened calculations and low efficiency due to many correlation calculations involved in interpolating. EST technique[3] is used to interpolate along these gently sloped edge regions of the image. A lower complexity algorithm based on information on slope of previous pixels is taken into consideration to estimate slope by predicting slope for the current pixel. Gradual slope domain changes are experienced in most of the edges. EST uses slope information from adjacent pixels to estimate the slopes in the edge domain. This reduces the computation complexity and increases the efficiency of edge slope calculation. Recursive calculation slope information of adjacent pixel is done instead of large correlation and masks.

Uses a previous slope constant kprev which holds 0 when in the beginning pixel of the image rows or after discontinuity. In other times it holds the slope value of adjacent pixel(previous pixel). kcur is calculated using the left(SL), mid(SM) and right(SR) absolute difference as below.

$$\begin{aligned} \square_{\square} &= |\square(\square-1,\square+\square_{\square\square\square\square}+1) - \square(\square+1,\square-\square_{\square\square\square\square}-1)| \\ \square_{\square} &= |\square(\square-1,\square+\square_{\square\square\square\square}-1) - \square(\square+1,\square-\square_{\square\square\square\square})| \\ \square_{\square} &= |\square(\square-1,\square+\square_{\square\square\square\square}) - \square(\square+1,\square-\square_{\square\square\square\square})| \end{aligned} \quad (7)$$

$$\square_{\square\square\square} = \begin{cases} \square_{\square\square\square\square\square}+1 & \square\square\ \square\square\square(\square_{\square},\square_{\square},\square_{\square}) = \square_{\square} \\ \square_{\square\square\square\square\square} & \square\square\ \square\square\square(\square_{\square},\square_{\square},\square_{\square}) = \square_{\square} \\ \square_{\square\square\square\square\square}-1 & \square\square\ \square\square\square(\square_{\square},\square_{\square},\square_{\square}) = \square_{\square} \end{cases} \quad (8)$$

A resetting criterion is used in case of existence of wrong value for kprev. Reset is triggered if minimum absolute difference of two adjacent pixels in the same line has difference more than a threshold(|Smin(i,j)- Smin(i,j-1)|>T). This suggests that the area is no tsmooth. The threshold is calculated experimentally(usually T=10).

With the kcur from EST, the interpolated pixel value is calculated by

$$\square'(\square,\square) = \frac{(\square(\square-1,\square+\square\square\square\square) + \square(\square+1,\square-\square\square\square\square))}{2}$$

6. Bilateral filtering : Deinterlacing using closeness and smoothness (DCS)

This is a version of intra field deinterlacing which uses both similarity and proximity to compute the edge direction successfully. A non linear spatial average is used to estimate the missing pixel value without edge degradation by combining the values of neighbouring pixels in a nonlinear fashion. Based on similarity and proximity geometrically, gray levels are combined with preference given to near value rather than far value[13]. This DCS algorithm is based on presumption of similarity and closeness between neighbour pixels which can be close to each other. An Adaptive monotonically decreasing function is used here to generate the weight m,n for further calculation of missing pixels.

The average of the 2 pixels in close proximity or two of them in the vertical-direction is used to

estimate the present interpolated hi-resolution pixels p'(i,j). This algorithm also incorporates a closeness weight coefficients with the pixel values given by $□_{□,□} = □□□(\frac{-(□^{2}+□^{2})}{2□_{□}^{2}})$ that decreases with distance.

Final DCS formula used to generate missing pixel considering both smooth and edge region:

$$□'(□,□) = \frac{\sum_{(□,□)□□} □_{□,□□} □_{□,□}□(□+□,□+□)}{\sum_{(□,□)□□} □_{□,□□} □_{□,□}} \quad (9)$$

7. Deep Convolution Neural Network based Video Deinterlacing

7.1 Training Dataset

Here data collection of interlaced and progressive video is required to train the system. A collection of different varieties of videos in different genres,cartoons and scenic views captured in a progressive device are used. Then randomly sample pairs of consecutive frames from gathered videos. Resize the frame to required size ex: 512x512 and use it as original frames. Then synthesise interlaced frames from two original frames i.e, copy odd fields from first frame and copy even fields from second frame. Further divide the triplets of 512x512 into 64×64-resolution patch triplets. Use the interlaced frame and original frames as training input<Ip,xt,p,xt+1,p>. Use the even lines and odd lines used to construct the interlaced frame as outputs <xevent,p , xoddt+1,p>. Use 80 percent of data for training and the rest 20 percent for validation.

7.2 Neural Network Architecture

Neural network made of five convolution layers is used. First three layers are sequentially connected and are common for two pathways.

The paper suggest the following details :

First convolutional layer is built up of sixty four 3 × 3 × 1 sized kernels. Second convolutional layer is built up of sixty four 3 × 3 × 64 sized kernels connected to first layer output. Third convolutional layer is built up of sixty four 1 × 1 × 64 sized kernels follows the second layer. Fourth convolutional layer is built up of sixty four 3 × 3 × 64 sized kernels with each path having its 32 kernels. Fifth convolutional consists of two kernels with size 3 × 3 × 64 with each pathway having one kernel.

Fourth and fifth layer branches have no connection between them. The first two layers' activations are ReLU functions, whereas the remaining layers' activations are identify functions. Convolution strides for the top 4 layers are one pixel. To get half-height photos, the vertical and horizontal stride should be kept at one and 2 pixels respectively for the final layer.

Optimal weights W* are trained using objective function

$$□^{*} = □□□\,□□□\,\frac{1}{□_{□}}\sum_{□}(||□'_{□□□□,□} - □_{□□□□,□}||_{2}^{2} + ||□'_{□□□,□+1,□} - □_{□□□,□+1,□}||_{2}^{2}) + □_{□□}(□□(□'_{□,□}) + □□(□'_{□+1,□})) \quad (10)$$

where Np is the number of samples to train the dataset , x' event,p , x' odd t+1,p are the neural network's predicted outputs.

TV() is total variance regulariser and TV is regularisation constant.

"The learning rate is 0.001 and λ_{TV} is set to 2×10^{-8} in our experiments. The number of epochs is 200 and the batch size for each epoch is 64. It took about 4 hours to train the neural network." [14]

When motion is extreme between the frames and when horizontal thin structures are present in interlaced video this method produces artifacts.

8. Wavelet Content Adaptive Back Propagation

This is a intra-field deinterlacing neural network based on pixel classification for wavelet content adaptive backpropagation. Overcomplete wave-let transform classify image into regions of smooth,edge and texture. The classified pieces of image with similar property are assigned to oneNN. Back propagation NN is a supervised learning method. This algorithm uses the sample set to train the weight of the neural network to get the interpolated pixel. Training involves extracting odd or even fields of each image sample as a separate image with low resolution and the sample image as the original hi-res image. Then, using BP-NN training weights, create a new high-resolution image. Take the difference between the stock high-resolution picture and NN-generated new high-resolution picture.[12]

Derivatives of the error as usual, through the network are back-propagated to calculate error gradient for linking weights. This algorithm uses six neighbour low-resolution pixels as input to obtain the high resolution missing pixel as the output pixel of the BP-NN. When the learning process is done the weights are fixed to a specific value. The wavelet transform classifies the image into three types: low-pass, horizontal and vertical high-pass. Each image type needs a neural network trained separately producing better output.

9. Conclusion

In most cases, temporal techniques outperform spatial ones in static regions in relation to quality of the image, however it does need high computations and more space in the memory, but produces artifacts in motion regions. Spatial deinterlacing performs better than temporal interpolation in these motion regions. Different temporal deinterlacing methods work differently on the motion regions based on the texture and edge properties of the image. For this purpose upgraded ELA methods like MELA are used in texture regions. EST and DCS are also good algorithms for texture and edge regions. It is important to understand the nature of video which is to be deinterlaced before selecting the deinterlacing method. Based on the requirement any one of these algorithms or set of algorithms should be used. Motion adaptive deinterlacing is the best approach for most video deinterlacing. This can be combined with edge and texture detection to separate the regions according to the image properties and separate algorithms can be used on each region to produce better output. Neural networks based deinterlacing requires understanding of machine learning concepts and requires a large set of data for training and validation.

For software tools any tool with vector processing or block coding helps in faster computation. Hence IPP is a better choice since it provides a lot of functions to ease the processing of video. The only disadvantage is the installation and setup part which is not simple as other tools.

10. Acknowledgement

This work was supported by Amagi Media Labs, India and carried out under the guidance of Mr. Swapnil Dabhade, Amagi Media Labs and Mrs Sahana B, Assistant Professor at R.V. College of Engineering.

REFERENCES

10.1 Journal Articles

[1] E. B. Bellars and G. De Haan, Deinterlacing: A Key Technology forScan Rate Conversion, Elsevier, Amsterdam (2000).

[2] T. Doyle, “Interlaced to sequential conversion for EDTV applications,” in Proc. 2nd Int. Workshop Signal Processing of HDTV, pp. 412–430(1998)
[3] Khan, Sajid & Lee, Dongho. (2015). Efficient deinterlacing method using simple edge slope tracing. Optical Engineering. 54. 103108. 10.1117/1.OE.54.10.103108.
[4] Huang Q., Gao W., Zhao D., Sun H. (2005) Adaptive Deinterlacing for Real-Time Applications. https://doi.org/10.1007/11582267_48
[5] T. Chen, H. R. Wu, and Z. H. Yu, “Efficient deinterlacing algorithm using edge-based line average interpolation,” Opt. Eng. 39(8), 2101–2105 (2000).
[6] W. Kim, S. Jin, and J. Jeong, “Novel intra deinterlacing algorithm using content adaptive interpolation,” IEEE Trans. Cons. Elect 53(3),1036–1043 (2007).
[7] P.-Y. Chen and Y.-H. Lai, “A low-complexity interpolation method for deinterlacing,” IEICE Trans. Inf. Syst. E90-D(2), 606–608 (2007).
[8] H. Yoo and J. Jeong, “Direction-oriented interpolation and its application to deinterlacing,” IEEE Trans. Consum. Electron 48(4), 954–962 (2002).
[9] Chen, P. and Yao-Hsien Lai. “A Low-Complexity Interpolation Method for Deinterlacing.” IEICE Trans. Inf. Syst. 90-D (2007): 606-608.
[10] J. Wang, G. Jeon and J. Jeong, "Efficient adaptive intra-field deinterlacing algorithm using bilateral filter," 2012 3rd IEEE International Conference on Network Infrastructure and Digital Content, 2012, pp. 468-472, doi: 10.1109/ICNIDC.2012.6418797.
[11] Serrano, Rebeca Sanz, “Deinterlacing Algorithms” ,Albalá Ingenieros S.A ,2016
[12] Wang, J., Jeong, J. Wavelet-content-adaptive BP neural network-based deinterlacing algorithm. Soft Comput 22, 1595–1601 (2018).DOI: 10.1007/s00500-017-2968-x
[13] Wang, Jin & Wu, Zhensen & Wu, Jiaji. (2016). Efficient Adaptive Deinterlacing Algorithm Using Bilateral Filter. MATEC Web of Conferences. 61. 02021. 0.1051/matecconf/20166102021.
[14] Haichao, Zhu & Liu, Xueting & Mao, Xiangyu & Wong, Tien-Tsin. (2017). Real-time Deep Video Deinterlacing.