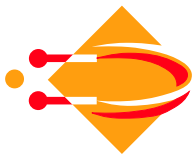

Variational Deinterlacing

Sune Keller
François Lauze
Mads Nielsen

IT University Technical Report Series **TR-2006-90**

ISSN 1600–6100 **June 2006**

ISSN 1600–6100

ISBN 87-7949-134-0

Copies may be obtained by contacting:

IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S
Denmark

Telephone: **+45 72 18 50 00**
Telefax: **+45 72 18 50 01**
Web `www.itu.dk`

Abstract

This is a technical report on the research results obtained in the field of inpainting based deinterlacing stating with Sune Keller's master thesis project and so far the first 15 months of his PhD work. Two methods of variational deinterlacing are presented, the first is motion adaptive and has in its earlier stages of development been presented in [19] and [17], the second one is motion compensated presented in a shorter version in [18].

1 Introduction

Interlaced scan has been use since the birth of television in the 1930's and is the scanning format used in the television standards PAL and NTSC. Interlacing is separates a full frame image into two parts called fields, one containing all horizontally odd numbered line and the other containing all the even lines. When recorded in interlaced scan the fields are separated in time and two neighboring fields cannot be merged to one full frame without problems.

Interlacing saves bandwidth and lowers the cost of cameras and CRTs as it is possible to combine a high rate of fields per second (to avoid large area flicker in the image) with a relatively high vertical resolution. This looked fine to the human visual system (HVS) in the early days of television but as screen size grew and television sets produced brighter images, interlacing artifacts started to show.

Interlacing artifacts have many names and are often mixed up when described, as they can be described both from a frequency analysis point of view and by their visual appearance. They are by visual appearance

- Line crawl due to vertical motion in the image and the time difference between the two fields composing one frame.

- Serration of edges due to horizontal motion in the image and the time difference between the two fields composing a frame. It happens to edges at all orientations except those close to and at horizontal orientation.

- Interline flicker due to fine stationary details appearing only in either odd or even fields of the image as they are to small (that is of to high a vertical frequency) to be sampled in both even and odd fields.

A further discussion of frequency analysis and aliasing in interlaced image sequences can be seen in [2] and [29].

One way of reducing the effect of these artifacts in terms of visibility to the human eye is to interpolate new fields and raise the field rate as done in 100 Hz TV sets [14] and [2]. Another way is to convert the interlaced sequence to a progressive sequence by interpolation of image information in the missing lines of the fields to make a full frame of each field. This conversion is called deinterlacing. Progressive scan is used in all PC monitors, in projectors, and in flat panel displays (LCDs and Plasmas). So as most new displays for television are progressive and as PC and television are merged (set top boxes for digital television, DVDs, television tuners for PCs, and video editing on PCs) there is obviously a big need for deinterlacing. Deinterlacing is difficult, as turning e.g. 50 fields per second into 50 frames per second requires a doubling of the amount of image data without introducing new artifacts to annoy the human visual perception.

In chapter 2 We propose a new scheme for deinterlacing developed from techniques used in image and image sequence inpainting, and we have implemented ten known and widely used deinterlacing schemes to compare it with. (The ten schemes are presented in section 1.2. Our scheme is motion adaptive (MA) and uses Total Variation (TV) based in a Bayesian framework and do MAP by minimizing an energy functional. This is accomplished by deriving and solving corresponding Partial Differential Equations (PDEs) obtained through the calculus of variations. This is in contrast to many known deinterlacers that have been developed ad hoc (and in a heuristic way) with online hardware implementation directly in eye. Therefore they are often simplified to keep hardware costs down. We start with a theoretically well-based off-line design that by further development could end up as online hardware.

The Motion Adaptive Total Variation Deinterlacing scheme of chapter 2 is quite advanced but even better results can be obtained by using motion compensation (MC) and that is the topic of chapter 3.

1.1 The Future of Deinterlacing

As mentioned above more and more TV-screens are progressive and in Scandinavia and other regions have decided to go for progressive broadcasting when switching to HDTV. All film recordings are by nature progressive and many TV productions are shoot on film and progressive (professional) CCD-based video cameras become more and more used. In general there seems to be a turn towards progressive so in a limited number of years deinterlacing will become obsolete except for use on archive TV material and why then spend time on further developing and improving deinterlacing algorithms? The answer is that deinterlacing is not becoming obsolete. Standard digital TV broadcast is still done interlaced and even though analog television seems dead, standard definition (SD) TV will still be broadcast digitally in 576 lines interlaced (576i) for PAL and 480i for NTSC until HDTV has taken over or is at least broadcast in parallel.

Even with the rise of HDTV deinterlacing is not necessarily dead; Scandinavia might have chosen 720p as their format, but USA and Japan has chosen 1080i an until they decide otherwise and HDTV has matured, deinterlacing

will still be needed in all progressive TV displays.[1]

1.2 Standard Deinterlacing Techniques

To measure the performance of our Total Variation Motion Adaptive Deinterlacing scheme, we implemented ten other schemes known from literature and/or available software and hardware. They are all basic or motion adaptive deinterlacers.

Line Doubling (LDB) is very simple. Every interpolated horizontal line is a repetition of the previous existing line ([33] and [35]). Line Averaging (LAV) is a vertical average of the above and below pixels, since they are both known ([2], [33] and [35]). Field Insertion (FI), a.k.a. merging or weaving, fills in the blanks with neighboring lines in time and is essentially a temporal version of LDB. The result is very similar to the image seen on an interlaced display ([2] and [35]). Field averaging (FAV) is a temporal version of LAV ([35]), while Vertical Temporal interpolation (VT) is a simple 50/50 combination of LAV and FAV ([35]). Many more advanced but not significantly better VT filters have been suggested, e.g. by BBC Research ([34]). All schemes mentioned so far are fixed, linear filters, whereas the next five are nonlinear and adapt to certain conditions in their local neighborhood and chose one of several possible interpolations depending on the local image content to yield better results.

Median filtering (Med) is a real classic in image processing and is used for deinterlacing in many variations ([2], [9], [15], [32], [31] and [33]). We have chosen a 3-tap vertical temporal version from [2] although we use the forward temporal neighbor instead of the backwards. Motion adaptive deinterlacing (MA) can be done in a countless number of ways and we have chosen a version suggested in [31] and [32]. It does simple Motion detection and takes advantage of the qualities of simpler schemes under different conditions: FAV in presence of no motion, Median filtering when motion is slow and LAV when fast motion is detected. Thresholds classify the motion. Weighted Vertical Temporal deinterlacing (wVT) is a simpler way of doing motion adaptation than the previous mentioned scheme, MA, and gives, instead of a hard switching between schemes, a smooth weighted transition between temporal and vertical interpolation. The scheme is described in detail in [21]. Edge Adaptive deinterlacing (EA) has been suggested in several forms, e.g. in [13], [21] and [33]. We have chosen a scheme that based on Summed Absolute Differences (SAD) selects a direction of interpolation as described in [33], although we have modified it to detect the best of five directions, 0^o, $\pm 26^o$ and $\pm 45^o$ from vertical. Successive Approximation (SA) is the second level of approximation in [21] although the its edge adaptive scheme working on the first deinterlaced approximation has been swapped with the EA scheme that works directly on the interpolated original and thereby taking the successiveness out of the scheme but in the same instance also removing the possibility of error propagation.

Med is a simple adaptive scheme, EA adopts to the orientation of edges while MA, wVT and SA are Motion adaptive.

For testing the color version of our color version of our Total Variation Motion Adaptive Deinterlacing Scheme, we made a test setup with what is considered as the state of the art deinterlacer in the market today, the Faroudja DVP-1010 Video Processor. The deinterlacing of the Faroudja processor known as DCDi® (Direction Correlated Deinterlacing) is basically a motion adaptive algorithm with a motion detector (MD) given the amount of motion on a 5-10 step scale. When there is no motion, a median spatio-temporal filter is applied, and this filter is turned stepwise off following the output from the MD, at the same time stepwise being mixed with the output of a purely spatial filter, which takes completely over when the highest level of motion is reached. The spatial filter is edge adaptive or direction correlated, as it detects edges at $\pm 45^\circ$ from horizontal (crossing direction)[2] and interpolated (LAV) along the edge. If no edge is detected LAV is used, which is also the correct solution for horizontal edges as well as smooth regions, but not good for vertical edges.

Standard motion compensated deinterlacing is the subject of section 3.1.

[1] In the world of television a generation change can take ages because of the popularity of the media as so many TV sets have to changed. As an example, when going from black and white to color, RGB would have been an obvious choice but a major reason for choosing YUV, was to allow the continued use of old b/w TV sets without having to double broadcast in b/w and color.

[2] The direction of an edges is here given as the direction to move in to cross the edge perpendicular and not the direction to walk along the edge, which could easily be argued to be the direction/orientation of the edge.

2 Total Variation Motion Adaptive Deinterlacing

This chapter is essentially work presented earlier in [17], but sections 2.5 and 2.6 present later work on an improved color version of our Total Variation Motion Adaptive Deinterlacing Scheme (TV MA DI).

In this section we introduce a novel deinterlacing scheme based on Total Variation minimization. We first proceed in a Bayesian fashion and deduce a variational formulation through MAP estimation in continuous settings following [28]. We then compute the associated Euler-Lagrange equations and their associated gradient descent formulations. The discretization of the latter will provide our numerical schemes. We first introduce the notations used in the sequel. Ω will denote the spatio-temporal domain of the progressive sequence, $F \subset \Omega$ the domain of the known fields, u_0 will denote the interlaced sequence, and by abuse of notations, it will also denote the known data on F.

2.1 Bayesian Framework

Let u denote a progressive sequence and u_0 the known sequence of interlaced fields. According to Bayes' Theorem

$$p(u|u_0) \propto p(u_0|u)p(u) \; . \tag{1}$$

The term on the left hand side is the a posteriori to be maximized (MAP) and the first term on the right hand the side is a model term and the second is a prior on image sequences. For the model term we choose a simple Dirac distribution $p(u_0|u) = \delta((u - u_0)|_F)$ because we wish to keep the existing pixels unchanged.

We have investigated two distributions for the prior term p(u). First, by viewing the image sequence u as a 3D volume, we set

$$p(u) \propto e^{-\lambda \sum_x |\nabla_3 u(x)|} \tag{2}$$

with x running over all the pixels in the sequence, $\nabla_3 u$ a discrete spatio-temporal gradient and λ a positive constant.

Nevertheless, it is somewhat unnatural to treat an image sequence as a 3D volume. We introduce therefore a simple model that separates spatial and temporal dimensions and we assume independence of the spatial and temporal distributions. Our image prior thus becomes

$$p(u) = p(u_s, u_t) = p(u_s)p(u_t) \tag{3}$$

where $p(u_s)$ refers to the spatial distribution of images and $p(u_t)$ to the temporal correlation between frames. For the spatial prior we use

$$p(u_s) \propto e^{-\lambda \sum_x |\nabla u(x)|} \tag{4}$$

with x again running over all the pixels in the sequence, ∇u a discrete *spatial* gradient and λ a positive constant. This has proven a robust model, well studied in the computer vision community; see for instance [1], [10] or [30]. The temporal prior

$$p(u_t) \propto e^{-\mu \sum_x |\partial_t u(x)|} \tag{5}$$

where $\partial_t u$ denotes the time-derivative of u and introduces the motion adaptive aspect of our algorithm, μ being a positive constant.

2.2 Variational Formulation - Euler-Lagrange Equations

Following [28] in order to compute the Maximum A Posteriori (MAP) solution, u, for our problems, we take the $-\log$ of each term to reformulate it as a minimization problem. Instead of using the $|\cdot|$ function which is non differentiable at the origin, we replace it by the approximation $\psi(s^2) = \sqrt{s^2 + \varepsilon^2}$, with $\varepsilon = 0.1$ or 0.01 in our experiments. From (2), with this modification we obtain u as the solution of

$$\operatorname*{Arg\,min}_u \int_\Omega \psi(|\nabla_3 u|)dx, \quad u = u_0|_F \; . \tag{6}$$

From standard calculus of variations and the fact that $\psi'(s)/s = 1/\psi(s)$, its Euler-Lagrange equation is

$$-\text{div}\left(\frac{\nabla_3 u}{\psi(|\nabla_3 u|)}\right) = 0, \quad u = u_0|_F \tag{7}$$

where div is the divergence operator. The associated gradient descent equation is

$$\partial_\tau u = \text{div}\left(\frac{\nabla_3 u}{\psi(|\nabla_3 u|)}\right) = 0, \quad u = u_0|_F \tag{8}$$

where τ denotes the evolution parameter (in order to not confuse it with the time parameter t of the sequence), which is a 3D *total variation* filter.

From (4) and (5) we obtain the following minimization problem:

$$\underset{u}{\text{Arg min}} \int_\Omega \left(\psi(|\nabla u|) + \alpha\psi(|\partial_t u|)\right) dx, \quad u = u_0|_F \tag{9}$$

the corresponding Euler-Lagrange equation being

$$-\text{div}\left(\frac{\nabla u}{\psi(|\nabla u|)}\right) - \alpha\partial_t\left(\frac{\partial_t u}{\psi(|\partial_t u|)}\right) = 0, \quad u = u_0|_F \tag{10}$$

and its associated gradient descent equation is

$$\partial_\tau u = \text{div}\left(\frac{\nabla u}{\psi(|\nabla u|)}\right) + \alpha\partial_t\left(\frac{\partial_t u}{\psi(|\partial_t u|)}\right), \quad u = u_0|_F \tag{11}$$

which combines a 2D total variation filter for the spatial part and a simple 1D total variation filter for the temporal part. The constant $\alpha = \mu/\lambda$ is a weight between the spatial and the temporal part of the filter. This approach to energy minimization gives convex but not strictly convex, solutions so several global minimums might exist. Therefore the solution can be sensible to initialization.

2.3 Discretizations

The gradient descent equations are solved explicitly, using forward difference for the evolution derivative ∂_τ and central difference for the divergence terms.

For the 3D divergence, we have used a standard discretization on the 6 points spatio-temporal neighborhood (see for instance [11], appendices, for details). For the 2D divergence we have used three different schemes, one using a 4-point neighborhood of the current pixel and two using a full 8-point neighborhood, as described in [1]. The sensibility of the above PDEs to initial values has not given us problems: At $\tau = 0$ to initialize we take the LAV deinterlaced sequence as a rough estimate with good results.

2.4 Results

We present now the results obtained with four image sequences. The first one, `Person`, is a medium shot of a sitting person, turning the head and talking, the motion can be said to be small. The second sequence, `C&T`, is a shot of a driving car and truck followed by a tracking camera, the motion, which is primarily horizontal, is up to ten pixels between two consecutive frames. The last two sequences are both artificial with high contrast details. The `BSNM` sequence is stationary while the `BS` has vertical, horizontal and diagonal motion, both accelerated and constant. Figure 1 shows stills of the two sequences `BS` and `BSNM`, whereas stills of the sequences `C&T` and `Person` cannot be published due to copyright issues.

The four sequences are all progressive, so we have chosen to give the Mean Square Error (MSE) as an objective measure of the performance of the schemes,

$$MSE = \frac{1}{N}\sum_{\Omega\setminus F}(u - u_{org})^2 \tag{12}$$

which measures the square difference between the *N* interpolated pixels in the output, *u*, and their removed counterparts in the original progressive sequence, u_{org}.

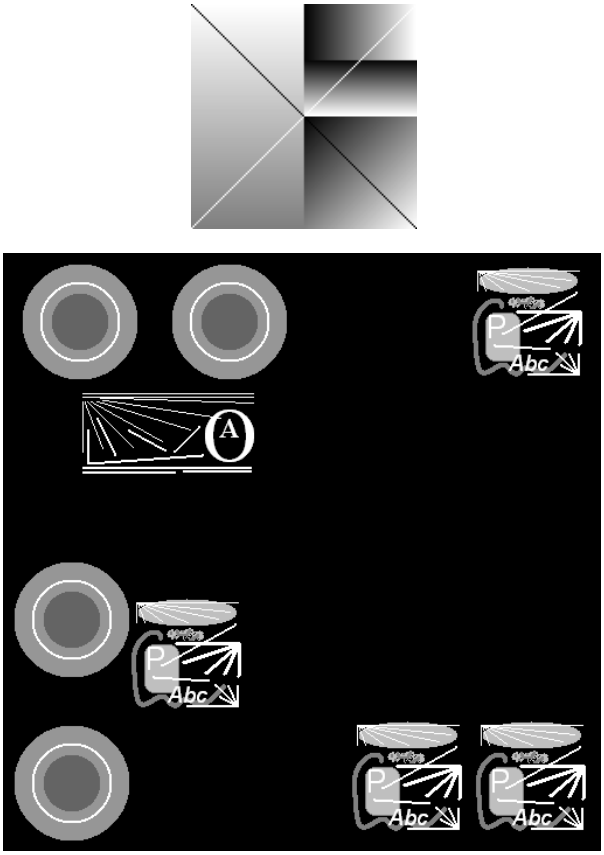

Figure 1: From the top: A frame from the 128x128 sequence BSNM and frame one of the 512x512 sequence BS.

Table 1: MSE from deinterlacing the four sequences Person, C&T, BS and BSNM. 3D and 2D+1D are the two versions of our scheme with the number of iterations given after the name. Clearly our schemes give the best results of all on the natural image sequences Person and C&T

Scheme		Person	C&T	BS	BSNM
LDB		17.90	79.72	1623.6	755.8
LAV		5.53	26.31	924.6	678.4
FI		22.26	472.25	3935.1	0
FAV		9.03	284.22	2514.4	0
VT		5.62	94.34	1146.8	169.6
Med		9.27	65.72	1154.1	363.4
MA		5.53	28.18	*840.6*	363.4
wVT		5.07	67.97	1056.0	0
EA		8.77	32.87	957.1	210.1
SA		5.36	48.79	862.4	82.0
3D TV	2	5.02	27.00	1066.2	666.8
3D TV	50	*4.85*	56.29	1078.3	461.8
2D+1D TV	2	4.97	*26.06*	919.2	666.2
2D+1D TV	20	*4.86*	*26.23*	890.9	567.9
2D+1D TV	200	5.11	32.89	*804.4*	242.5

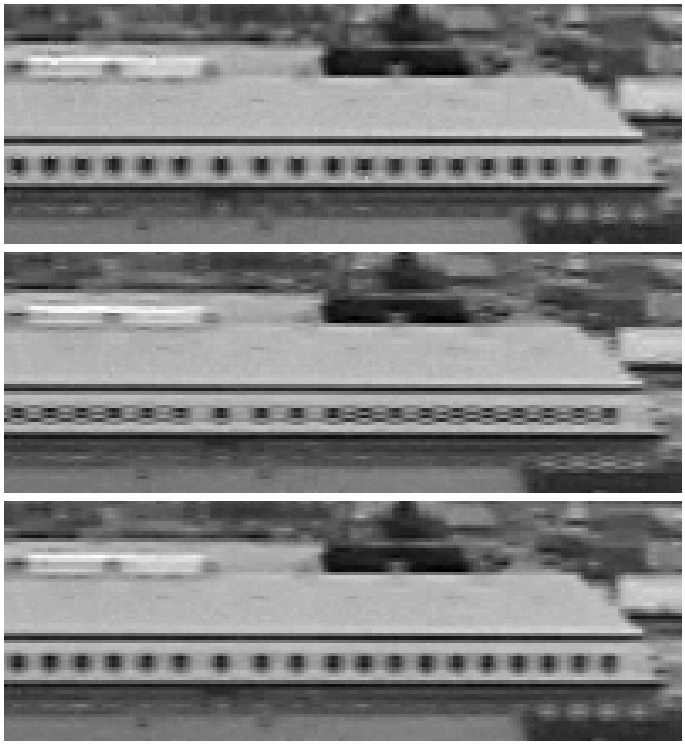

Figure 2: From top to bottom: deinterlacing with MA, wVT and 2D+1D TV. Only zoom-ins of the full frame is shown here

We also give a subjective evaluation as the final judge of the result is the human visual system. A discussion of how to determine the quality of deinterlacing is given in [2]. Table 1 gives the objective results.

On C&T and BS it is seen from the MSE's that in presence of large motion, our scheme offers only little improvement, and only for the motion adaptive 2D+1D version, where the spatial and temporal gradients are separated. The 3D version suffers from having a spatio-temporal gradient. Over time (in terms of number of iterations) the 2D+1D improves a lot on the BS but not on C&T, which contains the larger motions of the two. Although the 3D does not perform to good overall, it actually improves the per-frame MSE in 23 of the 98 frames in C&T. In presence of none or only small motion, our scheme wins as it can be seen from the MSE's on the BSNM and Person. After only 2 iterations a 10 % improvement is seen on the MSE of Person and it increases with the number of iterations. Taking SA as initial guess instead of LAV on BSNM gave a 9 % improvement in MSE after 20 iterations of 2D+1D.

The results for 2D+1D after 200 iterations show that convergence in MSE stops for the two natural sequences. This is due to the smoothing of the TV prior and noise in the original sequence. Further studies showed that the lowest values in MSE was reached after 40-60 iterations.

Subjectively our scheme produces the best visible results on all four sequences but BSNM. BSNM is fully stationary, so the temporal schemes give 100 % perfect results on it. On BS and C&T the improvement is moderate, but on Person the results from our scheme are clearly the best. After two iterations we already se a good subjective result for the 2D+1D scheme, but after 20 iterations (and 50 for the 3D) the results are really good, even making us doubt which is the deinterlaced when comparing to the progressive original.

Figure 2 top, middle and bottom also illustrate the potential of our method. The sequence used for the illustration shows a pan of Christiansborg Castle in Copenhagen and as it only exists as interlaced, no MSE's can be calculated. The top picture shows the result of the wVT scheme while the middle one shows the result of the MA scheme. Serious artifacts are visible for both schemes, serration for wVT and erroneous detection and interpolation for MA. The bottom picture shows the result of our 2D+1D scheme after 20 iterations, and clearly asserts the quality improvement obtained with our scheme.

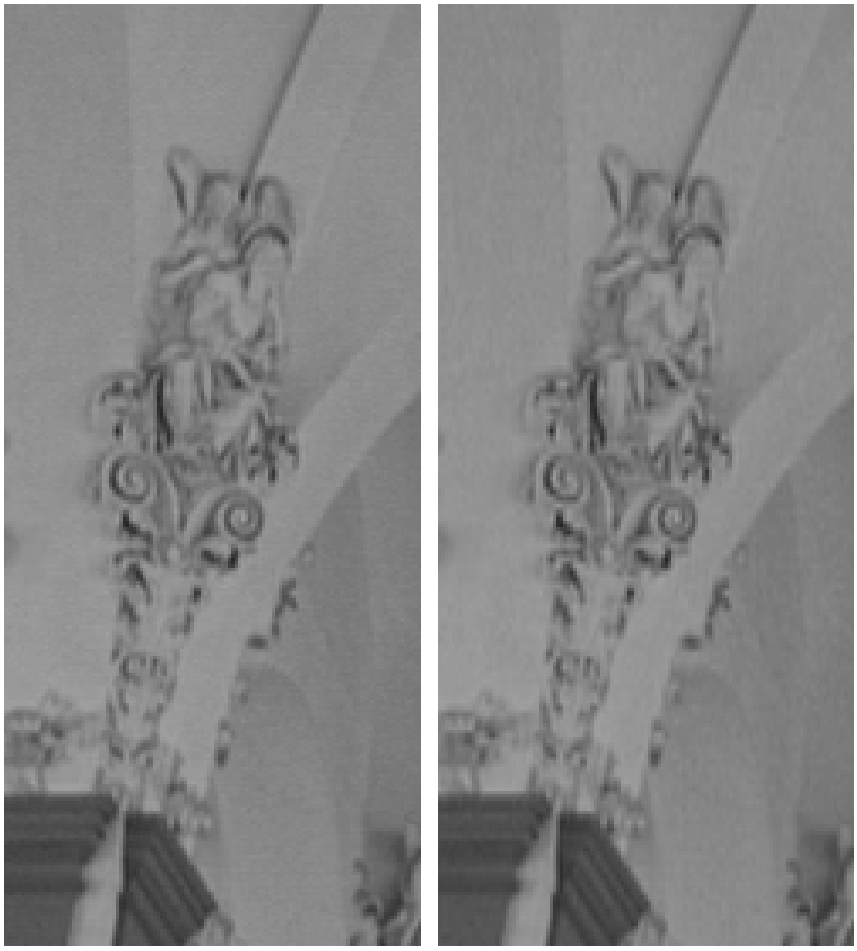

Figure 3: Deinterlacing of the sequence `Church`. Left: 2D+1D TV after 20 iterations. Right: LAV used as initialization for 2D+1D TV. Clearly 2D+1D TV improves on the LAV initialization. Only zoom-ins of the full frame is shown here. Notice the in particular the arm of the upper angel and the helixes (spirals) in the middle

LAV in itself is, given its simplicity, a remarkably well performing deinterlacer as the results in table 1 indicates, but as figure 3 shows, the 2D+1D scheme is able to improve the quality of deinterlacing significantly after 20 iterations. Note in the ornaments how the details have been sharpened and the jagged edges have been removed. The sequence used, `Church`, is stationary, shot with the camera on a camera mounting but rather noisy do to low lighting.

Further investigations on the 2D+1D schemes has also shown that the number of iterations to obtain a certain quality of the result can be reduced by a factor of three to six by increasing the time step in the gradient descent without the loss of stability. The number of operations and complexity per iteration of the 2D+1D schemes are the same as for the most complex of the ten known deinterlacing schemes, SA. This together with the increase in time step and good results after a few iterations gives rise to our believes that an online hardware implementation of 2D+1D TV MA Deinterlacing is possible.

The test results presented here are the essence of a much bigger (broader but also deeper and more detailed) test presented in [19]. In [19] an extensive test of the weighing of the spatial and temporal part showed that this could improve performance in certain regions (spatial and temporal) but that it was impossible to find optimal values that could generally improve the output quality of the full test set. Thus it was concluded that an improvement of the TV MA Deinterlacing scheme is possible by improved local weighing based on motion detection beyond what is already build into the scheme.

Before venturing into the area of improving motion detection, some results extending the algorithm from working only on grey-scale sequences to working on color (YC_rC_b) sequences. This also tests the algorithm on 5-10 times as much video as used in [19] and show its general useability.

2.5 Color TV MA

Generally the major focus in deinterlacing is the luminance/brightness channel of a color video signal, the Y-values of a analog YUV (or YP_rP_b) signal or a digital YC_rC_b signal. The human visual system is much less sensitive to changes in color than to changes in luminance and thus in both analog broadcasting and digital video (DVDs) and braodcast the two color channels of the signal are subsampled with at least a factor of 2 compared to the luminance channel. In analog media by halving the bandwidth of each of the two chrominance channels compared to the bandwidth of the luminance channel [26] and in digital media (using standards like MPEG-2 and H.264/MPEG-4) by simply subsampling the two color channels as thoroughly described in [35] where a good overview of different color formats can be found.

This property is used to simplify deinterlacing of color video. All though it is often left unsaid ([2] and [31] being among the very few exceptions), advanced deinterlacing seems to be used only on the luminance channel while the color channels are deinterlaced using a simple algorithm like Line Averaging (LAV).

We used this approach to extend TV MA deinterlacing to run on full color signals. Compared to the gray scale deinterlacing results there were no visible differences in presence of artifacts and overall image quality. Therefore it is safe to conclude, that the color channels should just be deinterlaced using LAV. This might change if data with full sampling in the color channels become available, but even the HD test material of the European Broadcast Union (EBU) is in sampled in 4:2:2 and digital HD broadcast is also subsampled in the color channels. The properties of the human visual system (HVS) also contradicts that anyone will ever sample the full color channel, but if they do so, and actually use the interlaced video format, still it would be silly to do advanced deinterlacing of the color channels.

2.6 Improved Motion Detection

The discussion of the gray scale results given in section 2.4 only scratches the surface of the much more extensive discussion and test of different choices for the weights on the spatial and temporal parts respectively given in [19]. One of the major conclusions in [19] is that tuning the weights globally is not sufficient to avoid incorrect deinterlacing due to faulty motion detection (MD) and that one of the major potential improvements of TV MA Deinterlacing is an improved local adaptive MD.

As a part of the further development of TV MA, we thus wanted to improve to avoid artifacts to survive or be created in the deinterlacing process and get the optimal output quality. What we did in connection with the experiments with color TV MA Deinterlacing was to add a local adaption for the weight of the temporal interpolation part. The weight for the spatial part was fixed at $w_s = 1.0$ while the weight for the temporal part was found by

$$w_t = \frac{1}{1 + (D/D_0)^{2n}} \tag{13}$$

or

$$w_t = 0.2 + \frac{0.8}{1 + (D/D_0)^{2n}} \tag{14}$$

where the first is basically a Butterworth filter and the second a Butterworth with the minimum value lifted from 0 to 0.2. We tested a wide range of values on a bigger test data set than the one used in section 2.4. The optimal setting was found to be $D_0 = 6$, $2n = 4$ using (14). D is the measure of how much motion is present at a location and we used $D = |u(x, y, t-1) - u(x, y, t+1)|$ but also tried $D = |u(x, y, t-1) - u(x, y, t)|$, which proved unstable. $u(x, y, t)$ is the pixel currently being deinterlaced and it either took on the value obtained in the previous iteration or for the first iteration the value from initialization by MA deinterlacing. As a little sidestep we tried to initialize all new pixel by setting their value to 128 and got convergence – all though slower – to the same end result (except for a minimal and only objectively measurable difference most likely due to numerical errors). So the only thing obtained by better initialization is faster convergence.

Generally a slight improvement of the output quality was obtained but in some situations the quality was worse, e.g. imagine a thin, light object moving on a dark background. If the object has a motion large enough to pass the location *(x,y)* in frame *t* then the dark background is what is present at the location *(x,y)* in the neighboring frames *t-1* and *t+1*. In this and similar situations, little or no motion will be detected in pixel *(x,y,t)* and temporal interpolation will not be switched of, resulting in serration like artifacts in the output.

Making the MD less local will mostly prevent this problem, but about the same artifact will be seen along moving edges due to the loss of locality. MD in interlaced video is in general quite tedious an in [2] MD is discarded because of this, saying that the fact is that the needed complexity to get it working properly given its limitations in terms of output quality, is not worthwhile compared to motion compensated deinterlacing (MCDI), which is not necessarily that much more complex, but yields (much) better outputs. In the light of our experiments with MCDI, we agree, mostly due to the potentially much better results one can get knowing the motion and not just whether there is motion or not. MCDI is the topic of section 3.

Although our results are far from perfect and we might be better of doing MCDI, the performance of our scheme was always marginally better than or equal to that of the Faroudja DVP-1010 even on sequences with regular structures in motion, and thus showed that if we got a real time version of our scheme, we could compete with a 4,200/euro (price in Denmark without VAT) high end home cinema video processor said to be state of the art and best in the world.

2.6.1 The Interlacing Problem

Most often – not to say always – the problem of artifacts due to bad motion detection appeared in regions in motion and was only detectable to the HVS when the motion of the region in question was not to fast and the sequence played back in real time. The fact that motion detection on interlaced video is difficult to improve is only a minor problem compared to another problem, which cannot be solved, not even with perfect motion detection. On sequences with highly regular structure like the front grill off many cars and trucks, fences, grates, grills, bars or brick walls etc. (any periodic structures, mainly in vertical or near vertical direction) the deinterlacing output looks horrible. This problem of course also appears at vertical and almost vertical edges and lines (ridges) in motion but is easiest observed by – and therefore most annoying to – the human visual system (HVS) in larger groups of edges or lines grouped as periodic vertical structures.[3]

Both our TV MA DI and Faroudja's DCDi® are unable to solve what we like to call *The Interlacing Problem*. It is basically the problem of correctly handling spatially high frequency image content in motion as the example given in figure 4. Since a motion detection does not help to (re)create the details in the missing lines, we could solve The Interlacing Problem by smoothing the troubled regions to avoid any artifacts, but this procedure will also blur out details, an undesired side effect especially noticeable for regions moving only slowly. Thus we need to find the information necessary to (re)create the details (the high frequency content). A way to do so to properly solve The Interlacing Problem is to propagate information over time from neighboring frames similar in content to the new pixel positions the details, but this can only be done by motion compensated deinterlacing. The optical flow of a sequence gives us an representation of the true temporal coherence in a sequence for all regions. With motion compensated deinterlacing (MCDI) we thus get a obtain results like temporal and motion adaptive deinterlacers produce in stationary regions – if the details we are looking for has actually been sampled sufficiently in nearby frames.

Figure 4 clearly illustrate how difficult it would be to recreate the correct structure using only spatial information. Viewing the example in the figure 4 as video also shows that due to perforated structure of an interlaced image volume and the high frequency content, extracting a correct flow is not a simple task. So even though MCDI has the potential of solving The Interlacing Problem, it is not straight forward to do so, in particular the flow calculations is difficult as will be discussed in section 3.2.

2.7 Conclusion

We have shown that a technique so far used for inpainting can be redeveloped to do deinterlacing and further on our Total Variation Motion Adaptive Deinterlacing outperforms ten known fixed or adaptive deinterlacers. In a full

[3]The term periodic structure could also be replaced by the general texture, but then we move from talking of a specific (de)interlacing problem to dealing with the general problem of subsampling. The point is we only care about vertical fine structures and not texture in general. This might be nitpicking and might just be a consequence of the fact that we have many names for the things we love.

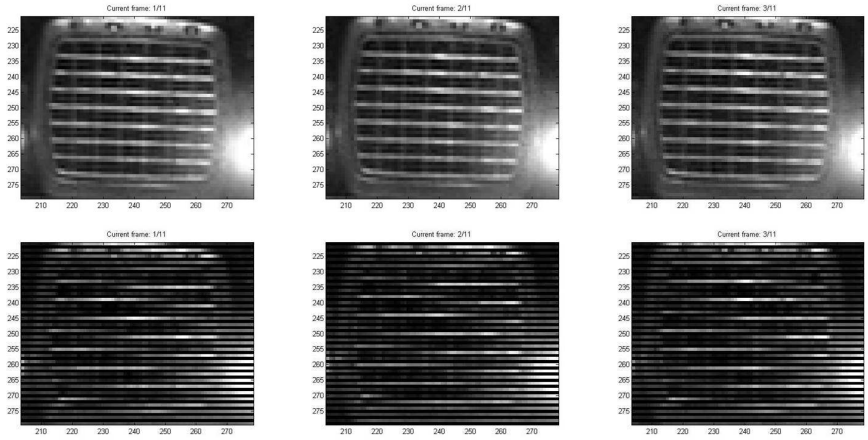

Figure 4: The Interlacing Problem. Top row: three consecutive progressive frames of a video sequence. Bottom row: the same three frames interlaced.

color version with improved motion detection we also (just) outperform the state of the art Faroudja DVP-1010 Video Processor with DCDi®. The quality of results and the computational complexity of TV MA DI indicate that a hardware implementation can produce high quality results in realtime.

Faroudja's DCDi® as well as our TV MA DI suffers from the inability to solve The Interlacing Problem as they are both motion adaptive and only motion compensated deinterlacing has the potential to truly solve The Interlacing Problem.

Variational based deinterlacing is a novel approach and introduces a whole new theoretical framework for video processing and the results for TV MA DI advocate the further exploration of the ideas presented here, and to significantly improve result an extension to the domain of motion compensated variational based deinterlacing is needed.

3 Motion Compensated Deinterlacing

3.1 Introduction

Motion adaptive deinterlacing takes advantage of temporal coherence in an image sequence whenever there is no motion in the sequence and relies on spatial coherence in regions with motion. As figure 4 clearly illustrate, it is not always possible to (re)create the original or true image content. Recreation of data is done in inpainting, whereas we create new lines in deinterlacing, but in test setups we sometimes first remove lines from a progressive sequence and then deinterlace it and thus actually recreate data with the sole purpose of measuring the objective quality of the deinterlacing. No matter if we create or recreate, deinterlace or inpaint, we want the data in the empty lines or inpainting region to be perceived as correct and true by the HVS. Modelling how the HVS perceives and senses visual input is thus just as good as modelling the actual image content. To stop a possible very long discussion on image sequence modelling, we just say that spatial information alone will not be able to create the structures seen in the top row of figure 4 from the corresponding interlaced fields in the bottom rows.

Motion compensated (MC) methods will use the input information from a motion estimation or optical flow computation process about the temporal coherence of all parts each frame/field with its neighboring frames/fields.[4] Ideally one would have correct flow information in all parts of an image sequence, but due to problems like the aperture problem, occlusions, numerical imprecisions and to simple modelling, one seldom has flawless flows (see e.g. [19], [22] or [35] for discussions and details on flow problems). The quality of motion compensated deinterlacing heavily relies on the quality of its input flow fields and if the flow is of high quality, one will only have to use spatial information sparsely and thus almost fully solve The Interlacing Problem getting a highly detailed image also in regions of motion. In [4] Black and Anandan emphasize the need for a good quality control of the flow to make sure one do not use incorrect flow values and since somebody is still to come with the perfectly flawless, generally applicable motion estimator, this holds today.

Motion compensation is somewhat more complex than motion detection, but with the need of stepwise knowledge of the 'amount' of motion, the difficulties of working in interlaced image volumes and the general complexity of doing high quality, reliable motion detection, Bellers and de Haan argue in [2], that MC is not that much more complex than MD and is then clearly the better choice as the resulting output quality is (potentially) much higher. This is of course true when you limit yourself (to keep costs down etc.) to using a simple MC method as block matching as Bellers and de Haan uses in the Integrated Circuits (ICs) they develop for use in Philips TV sets [2]. When you use heavier variational based methods for flow computations, the complexity is somewhat higher to say the least, but the gained output quality is also significantly better.

3.1.1 Standard Motion Compensated Deinterlacing

The performance of standard motion compensated deinterlacing is not frightening from a competitive point of view. In [3] Biswas and Nguyen presents a motion compensated deinterlacing algorithm with a demo given at `http://videoprocessing.ucsd.edu/deint.htm`. The Apollo example given clearly shows a high level of serration and thus the actual output quality of the scheme is poor.

Bellers and de Haan has done extensive work on motion compensated deinterlacing and a selection of 11 algorithms are presented and tested in [2]. All algorithms use flow fields generated using block matching motion estimation also described in [2]. The 11 algorithms are tested against five standard deinterlacer on a given set of test sequences. On moving sequences, using MSE as a quality measure, line averaging (LAV) scores 43 and four of the motion compensated algorithms actually scores worse (45-72)! The seven algorithms performing better scores 27-36 thus performing up to 37% better.

[4] In one interpretation, flow computation or estimation is a very broad term covering almost any measurement of changes between (in some way) consecutive data volumes and the term optical is prefixed when data is recorded using an optical device, e.g. a photographic or film camera and not scans like MR, PET etc. Motion estimation is a more narrow term and describes basically the same as optical flow: the apparent changes over time (or displaced frame difference to add another term) in a 2D recording of the 3D world using a movie/video camera, but some times also indicates an ambition to recover the true 2D projection of the physical 3D motion or actually recover the true 3D motion. This is only one interpretation and we will use the terms (optical) flow and motion estimation interchangeably about trying to find a set of vectors tracing the changes in time of an images sequence, such that we get the best possible up-scaling (deinterlacing in this case).

Two other measures, MSE_i and MTI, are introduced and both measures the square difference along the motion trajectories, which of course favors the motion compensated methods. MSE_i will punish bad flows as original pixels are always used as one of the inputs and for MTI the authors claim the same, but that is not true. To punish bad flow the true flow must be known and it is not for the sequences used in test as they are all real world optical recordings with block matching flows. Thus MTI will not punish bad flows when the same flow is used in both the algorithm and the error measure unless the algorithm has a perfect switch-off mechanism for bad flow input and the presented algorithms do not have that. As a consequence MSE_i on the same test set scores 74-95 for the 11 MC algorithms, which is lower than LAV's 132, but suddenly field insertion that scored 322 with MSE now scores 95 thus is suddenly a good algorithm! MTI makes the MC schemes even better with scores of 44-90 with LAV scoring 127 and FI 324. The authors go on with elaborate graphical displays of the errors on the test sequences to show how good the MC schemes presented are, but on sequences with complex motion LAV and/or simple spatio-temporal deinterlacing perform just as well as the MC methods and on the most complex test sequence used (named `bicycle`) the MC methods break down and LAV outperforms them all measured in MSE and is only beaten marginally by nine of the MC methods using MTI.

At least one of these motion compensated schemes have been used in circuits for Philips TV sets and progressive DVD-players.[5] Funnily enough Philips has also used Faroudja/Genisis Microchips DCDi® ICs in some TV sets and at least one of their top of the line DVD-players.[6] The authors also argument heavily against motion adaptive deinterlacing and has completely left those out in the test, which of course seriously devaluates the credibility of any conclusion drawn from these test results. Objective testing is however not the optimal test measure and from the subjective testing in [2] no substantial proof of superior performance of the presented motion compensated schemes are given. Although we criticize the testing in [2], we do believe it is the comprehensive single piece of literature on deinterlacing.

To stop this bashing of other methods, we do not think the authors of [2] have used the best ways of doing motion compensated deinterlacing, and from the results available and the results from our motion compensated deinterlacer presented in this chapter, we claim to do much better. And then we even think we are far from having shown the full potential of our framework.

3.2 Motion Interpolation on Interlaced Sequences

This is like the problem of putting a triangular bolt in a square hole: standard ME is developed for full frame progressive image sequences and thus not well-suited for use on interlaced image sequences. The typical approach is to use a simpler non-motion compensated deinterlacer to create an initial progressive sequence to be used for the motion estimation step ([2] and [21]). This of course influences the resulting obtained flow due to the interlacing/deinterlacing artifacts in the sequence, this being most obvious in the regions with motion, where spatial interpolation will always be the choice of anything simpler than MC DI.

Due to this problem, we wanted to calculate the optical flow in another way, but just like in inpainting it is a hen-egg problem: in the new pixel positions (the missing data locus), should you compute the flow or the intensities first. Before 'solving' the hen-egg problem, we do in all cases need a flow to initialize the motion compensated deinterlacer, preferable calculated with as few assumptions as possible,preferably without simple deinterlaced intensity values in the image sequence used for motion estimation. We have mainly considered five possible solutions:[7]

1. *The standard approach*: Do motion estimation on a simple deinterlaced version of the full sequence, which is what is often done in standard motion compensated deinterlacing.

2. *The cost efficient approach*: Use the original pixels of the odd fields and then use The LAV (or other deinterlacing) of the even fields to get odd line data for the full sequence. This can of course also be done the other way around. This cuts the computational cost by 50 % but then requires a vertical up-scaling or 'scaling-smoothing' of the flow field.

[5] Both authors are or have been employed by Philips Research.

[6] According to specifications in Philips home entertainment catalogues 2003-2005.

[7] The term field is used for both interlaced frames and for fields of flow vectors. To avoid confusion we will try to use flow field whenever we talk of flow vectors of an image sequence frame and say fields (odd and even) whenever we talk of a frame of intensities missing every second line and thus being interlaced.

3. *The 'half a pixel up half a pixel down' approach*: use original data only and correct for the misalignment of half a pixel when upscaling like in 1). This seems a simple solution but in practice it will not be easy to handle top and bottom boundary conditions and as all motions will be imposed a up-down zigzagging motion, which should also be reflected in the prior, thus requiring an all new prior. Alternatively one could use information at half-grid positions but this is nontrivial to do in actual implementations and would bring us close to solution 2 suggested above.

4. *The original data only approach*: Use original data at original positions only but process the even fields and the odd fields as two independent progressive sequences. This is what we actually chose to do for the test of which the results are given in this chapter. To obtain the final flow field from the two separately computed even and odd 'time to time+2' flow fields a 'fusion-scaling-smoothing' is needed. To fuse the two flow fields an assumption is made: flows are assumed to be linear over two consecutive frames. A 'fusion-scaling-smoothing' scheme without this assumption, allowing flows to vary from t to $t+1$ instead of from t to $t+2$ is on the drawing board.

5. *The ideal approach*: Develop an interlaced motion estimation, which is specifically designed to run on the perforated interlaced data volume. This would require a long list of assumptions of how flow develops over time, mainly the relation between 't+1' and 't+2' flows.

All these solutions would in a traditional setting be seen as a preprocessing step to produce input for the actual motion compensated deinterlacing. A computationally more costly but in terms of potential output quality far superior solution is to do simultaneous calculations of flow and intensities as done for inpainting in [23] and [22] and described for video up-scaling in the patent applications [24] and [25], thus in some sense solving the hen-egg problem as you iteratively update the flow and intensities and improve both based on better versions of each other. A basic advantage of using iterative schemes in combination with a temporal aperture spanning both backwards and forwards in time and simultaneous calculations of flow and intensities – as suggested in the above references and text – is an increased spatial and temporal coherence in both intensities and flow field. In deinterlacing this approach have so far only been implemented with respect to temporal backward and forward aperture, the simultaneousness in the form of iterative updates of flow and intensities has so far only been done for inpainting, but is expected to take motion compensated deinterlacing to the next, higher level. Still, 'just' using flow calculated with variational methods as input to a variational MC deinterlacing has proven highly successful as section 3.7 will show.

3.3 A Common Framework For Simultaneous Computation of Flow Field and Intensities

Deinterlacing can be seen as a subset of super resolution (SR), that is increasing the spatial resolution of each frame in a sequence. Both super resolution and deinterlacing are again very similar to temporal super resolution (TSR a.k.a. super resolution in time or super slow) that is increasing the number of frames in a sequence. Deinterlacing can also be considered an hybrid of SR and TSR as it is both an increase of resolution in time and space (horizontally). Covering the full set of the three transformations, we call it image sequence *up-scaling*. Given the familiarity of these different up-scalings, it is only natural to define a common framework for them.

In our work on motion adaptive deinterlacing we have all the time kept the original intensity values, but it is not a given fact, that we should not at all change them. Optimally we do not change them, as they are the 'true' data, but in real life this data might be noisy and as one adds the knowledge of the flow to what one does – and anyway create 50% of the output data from scratch – it might improve the output to (slightly) alter the intensity values at the original pixel positions. We therefore present two possible frameworks the only difference being that one leaves the original pixels untouched while the other allows for a controllable diffusion of them: In stead of updating the intensity values in each iteration from the result of the previous iteration as done for the new pixels, the original pixel value is always used and can be seen as an anchor tying the new values to the original input values so the don't drift to far from the original.

A model of an image sequence is formulated and expressed as one or more equations giving the energy of the image sequence in terms of the pixel information (grey level intensities or color values) and/or flow field. The concept of energy of an image and image sequence is known per se. In the present context, it has its roots in the use of Bayes' Theorem to formulate the probability of a desired (final) output image sequence given an input (original) image sequence and a chosen model or set of models of image sequences and flow fields. The process of maximizing

this probability, which is the desired goal to produce the final image sequence, can be reformulated as the process of minimizing the energy [28]. Details on this is given in [22] for inpainting and it is shown in [18] that on this high level of abstraction it is practically the same for deinterlacing.

In the framework described herein, the energy of the image sequence is given in the form of a functional of the pixel values and the flow field. In this functional there are given mathematical and statistical models of what an image sequence and its flow fields should look like. These models are used to find the best or good (sub-optimal) estimates of pixel information in the new pixel positions and displacement values for all pixels positions in an image sequence. Inevitably, there is a trade-off between obtaining the best possible image quality to the human visual system (i.e. detailed but without artifacts stemming from the up-scaling process) and mathematical tractability (which in practice is necessary, especially as the theoretical formulation cannot be implemented in practice as it has no numerical counterpart). Moreover, of course, the method has to be embodied in apparatus, such as an up-scaler or general purpose computer, which inevitably has practical constraints on its processing power and the like.

The following notation will be used:

- u_0 the known pixel values and K their locations in the new resolution settings,
- u the sequence to be created,
- u_s a local spatial distribution (the neighborhood of each pixel within the same frame) for u,
- u_t a local temporal distribution (the neighborhood of each pixel in the previous following frames) for u and
- $\vec{v}$ the flow field representing the displacement between two consecutive frames.
- Ω is the domain of the full progressive sequence and $K \subset \Omega$ is the domain of the known original input data.

The (up-scaled) final output image sequence u and its corresponding flow field are computed as (suboptimal) minimizers of the constrained energy:

$$E(u, \vec{v}) = E_1(u_s) + E_2(u_s, u_t, \vec{v}) + E_3(\vec{v}), \quad u = u_0|_K \tag{15}$$

where $E_1(u_s)$, $E_2(u_s, u_t, \vec{v})$ and $E_3(\vec{v})$ each help model image sequences and flows. Just as in TV MA Deinterlacing, modelling of pixel information is only to be imposed in new pixel positions while the values in original pixel positions (i.e. of pixels in the original image sequence) are retained. The displacement field is modelled in all pixel positions, new and original.

$E_1(u_s)$ punishes too large local spatial variations in each frame as large local variations in pixel information values is the prevalent feature of noise and thus not desirable. The term also tries to pull pixel information in new pixel positions in the direction of smooth images. Choosing a good functional for the E_1-term allows smooth regions and edges as long as they are not too large in variation.

The last term, $E_3(\vec{v})$, enforces a reasonable local smoothness for the displacement field. It is assumed that the displacement in the image sequence is caused by objects moving so that each pixel does not necessarily have an individual displacement but will have a displacement similar to that of (some of) its neighbors as they represent different parts of the same object – the flow is grouped in regions representing different objects. The functional imposed on the flow field in $E_3(\vec{v})$ is often the same as the one imposed on the pixel information in $E_1(u_s)$ as there are edges in the displacement field between regions or objects moving differently just as there are edges in the pixel information.

In general, $E_2(u_s, u_t, \vec{v})$ is the most complex of the three terms. It links local spatial and temporal sequence content with the image sequence flow field $\vec{v}$ and conversely links $\vec{v}$ to the image sequence content, penalizing large discrepancies. This term works both on the pixel information and the displacement. It is a representation of the well-known Optical Flow Constraint (OFC) stating that the pixel information should stay the same along the displacement field. To allow for changes in lighting and disocclusions (i.e. where one object moves behind another and disappears), edges in time in the displacement fields should be allowed.

It is desired to model an image sequence that is a recording of the real world, where coherence, order and causality exist. Minimizing the energy means that the image sequence should obey the constraints imposed on it by equation

(15). (In practice, owing for example to technical constraints in the apparatus carrying out the process, it is not always possible to obtain the optimal solution (with minimal energy) so the one closest to is used, i.e. the suboptimal solution.)

Alternatively, it may be desirable to add de-noising to the resolution enhancement by computing minimizers of the unconstrained energy:

$$E(u, \vec{v}) = E_0(u, u_0) + E_1(u_s) + E_2(u_s, u_t, \vec{v}) + E_3(\vec{v}) \tag{16}$$

where the term $E_0(u, u_0)$ allows deviation from the original values u_0, but penalizes too large deviations. De-noising by adding the term $E_0(u, u_0)$ simply helps clean up the image sequence and equation (16) may be used instead of equation (15) if the input sequence is very noisy or generally smooth images are desired. Adding the term $E_0(u, u_0)$ will in most cases not only de-noise but also to some extend smooth out details.

This technique using energy minimization as in equation (15) and (16) has so far been used for the image sequence restoration technique known as inpainting to repair blotches, scratches and other damage on digitized films (such as movies and medical scans), in which damaged pixels are repaired. See for example [23]. However, it has to the best of our knowledge not been used for up-scaling as proposed herein, in which new pixels are created and added to image sequences.

3.3.1 Choice of Functionals for the Terms E_i

To minimize the energy, an iterative technique is preferred, but first functions or functionals for the terms of equations (15) and (16) (E_i with $i = 0, 1, 2, 3$) are selected in accordance with the preferred embodiment. The functions or functionals need to be good estimates of the image sequences and displacements concerned (typically being projected recordings of the real world, including physical motion and changing lighting conditions), but at the same time be mathematically tractable. To suit these purposes, various alternatives for both u and $\vec{v}$ may be used. Examples include:

Gaussians (harmonics):

$$E_1 = \lambda_1 \int |\nabla u|^2 dx \tag{17}$$

$$E_2 = \lambda_2 \int |\vec{v}\nabla u + u_t|^2 dx \tag{18}$$

$$E_3 = \lambda_3 \int (|\nabla v_1| + |\nabla v_2|)^2 dx \tag{19}$$

where $\vec{v} = (v_1, v_2)^T$ and ∇ denotes the spatial gradient and u_t is the first derivative of u with respect to time, t.

Gaussians impose smoothing on both the intensities (u) and the displacement ($\vec{v}$). Smoothness is a generally desirable and aesthetically pleasing property in an image except at edges (the edges being edges in intensity and in the flow fields boundaries between objects moving differently).

More generally, the exponent, 2, in all three examples above (equations (17) - (19)) can be replaced by α, e.g.:

$$E_1 = \lambda_1 \int |\nabla u|^\alpha dx \tag{20}$$

The case where $\alpha = 1$ is called total variation, and is very interesting as it possesses the same smoothing properties as the Gaussians, but by allowing larger variations it also preserves edges by lowering the smoothing across edges, at the same time still allowing for smoothing along the edges. This property ensures a good continuation of edges through new pixel positions when doing upscaling.

A possibility, and sometimes a must, is to use a regularization of the gradient magnitude, e.g. $|\nabla u|$, because the abs-function ($|\cdot|$) is not differentiable at the origin, and the differentiability is a desired property:

$$E_i = \lambda_i \int \psi(|\nabla u|^2) dx, \quad \underbrace{\psi(s^2) = \sqrt{s^2 + \varepsilon^2}}_{strictly convex} \quad or \quad \underbrace{\psi(s^2) = \log(s^2 + \varepsilon^2)}_{nonconvex} \tag{21}$$

where ε has a small value, such as 0.1 or 0.001 for example.

All these examples are rather simple functions. A more complex example is:

$$E_2 = \int \psi\Big([u(x+\vec{v}) - u(x)]^2 + \gamma|\nabla u(x+\vec{v}) - \nabla u(x)|^2\Big)dx \tag{22}$$

where γ is a constant and x denotes spatio-temporal coordinates (x, y, t).

In some cases (especially concerning the calculations of the flow) u may be replaced by u_σ, the frames pre-smoothed by convolution with a Gaussian kernel. This is done to get a more accurate and smooth flow field and can give significant improvements.

3.3.2 Typical Way of Minimizing the Energy

Typically the energy functional (which is an integral equation) is turned into a set of partial differential equations (PDE) named Euler-Lagrange equations (EL) by the use of calculus of variations:

$$\begin{cases} \dfrac{\partial E}{\partial u} & = 0 \\ \\ \dfrac{\partial E}{\partial \vec{v}} & = 0 \end{cases} \tag{23}$$

The derivation of Euler-Lagrange equations is a well-known technique but becomes increasingly difficult when the energy functionals become more complex, like the one given in equation (22). The choice of optimization strategy (solving the Euler-Lagrange equations) partially depends on the different terms in $\frac{\partial E_i}{\partial u}$ and $\frac{\partial E_i}{\partial \vec{v}}$ but also the shape of the loci of the new pixels. (Spatial super resolution, deinterlacing, and temporal super resolution/super slow have different loci, the loci being the local organization of new and original pixel positions with respect to each other.) Typically, however, it can be done by minimization of the Euler-Lagrange equations' corresponding Gradient Descent equations, which then becomes the iterative step mentioned above. Other methods of solving the Euler-Lagrange equations are Jacobi solvers, Gauss-Seidel solvers or SOR solvers. All three are direct solvers, solving a linear system using fixed point iterations to get values for the possible non-linear elements of an Euler-Lagrange equation. It is also possible to use the less known multigrid solvers.

In general terms, one example of the iterative process can be described as:

$$\begin{aligned} u(\tau+1) &= f_u\big(u(\tau), \vec{v}(\tau)\big) \\ \vec{v}(\tau+1) &= f_{\vec{v}}\big(u(\tau+1), \vec{v}(\tau)\big) \end{aligned} \tag{24}$$

where τ is the evolution time of the algorithm (to distinguish it from the temporal dimension t of the sequence, as these two are completely different parameters). f_u and $f_{\vec{v}}$ are solvers, e.g. PDE solvers as the gradient descent, that compute improved updates of the intensities and displacement field of the sequence. So for each increment of τ in the sequence, the energy of the sequence in equation (15) or (16) is updated ultimately to minimize the energy and thus the quality of the image is enhanced.

It should be noted that the example of the iterative process mentioned above uses $u(\tau)$ and $\vec{v}(\tau)$ (i.e. "previous" values for pixel information and flow from the previous iteration) to find $u(\tau+1)$, and $u(\tau+1)$ and $\vec{v}(\tau)$ (i.e. the current iterated value for pixel information and the previous value for flow) to find $\vec{v}(\tau+1)$. However, in general, in the iterative process to find $u(\tau+1)$ and $\vec{v}(\tau+1)$, any of $u(\tau)$, $u(\tau+1)$, $\vec{v}(\tau)$ and $\vec{v}(\tau+1)$ may be used, and these may be used interchangeably and variously through the iteration process. Say one iterates through an image row by row from top to bottom, in each row going through each pixel from left to right. Then for any neighboring pixel above or before in the current row you can use either $u(\tau)$, $u(\tau+1)$, $\vec{v}(\tau)$ or $\vec{v}(\tau+1)$ values, but for the later and lower neighbors you can only use $u(\tau)$, or $\vec{v}(\tau)$. To save memory one might overwrite the old values and thus only have $u(\tau+1)$ and $\vec{v}(\tau+1)$ for the upper and left neighbors. One might iterate differently and one might use parallel processing, but no matter what, in general one can choose any values from the current $(\tau+1)$ or previous (τ) set of values that optimizes the end result.

At $\tau = 0$, it is necessary to initialize the new pixel positions, and the better it is done the faster an optimal result is obtained.

3.3.3 An Example

An example of an energy formulation and its corresponding set of Euler-Lagrange equations are using total variation:

Energy formulation:

$$E(u,\vec{v}) = \lambda_1 \int \psi_1(|\nabla u|^2)dx + \lambda_2 \int \psi_2(|\nabla u \cdot \vec{v} + u_t|^2)dx + \lambda_3 \, int\big(\psi_3(|\nabla v_1|^2) + \psi_3(|\nabla v_2|^2)\big)dx, \quad u = u_0|_K \tag{25}$$

where: $\psi(s^2) = \sqrt{s^2 + \varepsilon^2}$, ∇ is the spatial gradient operator, λ_i are some constants and v_1 and v_2 are the x- and y-components of the flow field, i.e. $\vec{v} = (v_1, v_2)^T$.

The resulting EL equations are (one for u and a pair for each of the two components of the displacement field)[8]:

$$\frac{\partial E}{\partial u} = -\lambda_1 \text{div}_2 \left(\frac{\nabla u}{\psi_1(|\nabla u|^2)} \right) - \lambda_2 \text{div}_3 \left(\frac{\psi_2'(|\nabla u \cdot \vec{v} + u_t|^2)}{|\nabla u \cdot \vec{v} + u_t|} (\nabla u \cdot \vec{v} + u_t)\vec{V} \right) = 0 \tag{26}$$

$$\frac{\partial E}{\partial v_i} = -\lambda_2 \frac{\psi_2'(|\nabla u \cdot \vec{v} + u_t|^2)}{|\nabla u \cdot \vec{v} + u_t|} (\nabla u \cdot \vec{v} + u_t)u_i - \lambda_3 \text{div}_2 \left(\frac{\nabla v_i}{\psi_3(|\nabla v_i|^2)} \right) = 0, \quad i = 1, 2 \tag{27}$$

div_2 is the divergence in 2D (x, y) known from physics and div_3 is the divergence in 3D (x, y, t) and $\vec{V} = (v_1, v_2, 1)^T$.

For the first term in the middle part of equation (26), the spatial part will stop smoothing across edges in each frame because of the denominator, as it is in essence the gradient magnitude, the most commonly used edge detector in image processing. The same goes for the second term in the middle part of equation (27) just for spatial edges in the displacement field that separates object or regions with different displacements. For the remaining somewhat more complex terms representing the OFC mentioned earlier, something similar happens for edges in time (e.g. occlusions), generally allowing propagation of information along the trajectory of the flow field.

Minimization of these Euler-Lagrange equations can be used for deinterlacing, spatial super resolution and temporal super resolution/super slow.

The gradient descent equations corresponding to equations (26) and (27), which are to be discretized and solved iteratively, are:

$$\frac{\partial u}{\partial \tau} = -\frac{\partial E}{\partial u}$$

and

$$\frac{\partial v_i}{\partial \tau} = -\frac{\partial E}{\partial v_i}$$

i.e.

$$\frac{\partial u}{\partial \tau} = \lambda_1 \text{div}_2 \left(\frac{\nabla u}{\psi_1(|\nabla u|^2)} \right) + \lambda_2 \text{div}_3 \left(\frac{\psi_2'(|\nabla u \cdot \vec{v} + u_t|^2)}{|\nabla u \cdot \vec{v} + u_t|} (\nabla u \cdot \vec{v} + u_t)\vec{V} \right) = 0 \tag{28}$$

$$\frac{\partial v_i}{\partial \tau} = \lambda_2 \frac{\psi_2'(|\nabla u \cdot \vec{v} + u_t|^2)}{|\nabla u \cdot \vec{v} + u_t|} (\nabla u \cdot \vec{v} + u_t)u_i + \lambda_3 \text{div}_2 \left(\frac{\nabla v_i}{\psi_3(|\nabla v_i|^2)} \right) = 0, \quad i = 1, 2 \tag{29}$$

where τ is again the iteration/evolution time. This equation is then discretized using well known numerical schemes.

As an example, The discretization of the term $\text{div}_2(f|\nabla u|)$ in equation (28) where $f = 1/\psi_i(|\nabla u|^2)$ and $\nabla u = (\partial u/\partial x, \partial u/\partial y)^T = (\partial_x u, \partial_y u)^T$ will now be considered. Now by definition of the divergence, $\text{div}(f|\nabla u|) = \partial_x(f\partial_x u) + \partial_y(f\partial_y u)$. This expression can be rewritten as a finite difference scheme using the well-known Taylor's approximation for (partial) differential equations. The good thing about finite difference schemes is that they can be used directly on (digital) discrete image sequences.

[8]It is assumed here that we are operating on grey scale intensity values. For deinterlacing and SR with lower factor resolution enhancement, color channel processing is simple. For TSR and SR with a big increase in resolution full color processing is needed and a discussion on the linking of the channels in an RGB-signal to give a vectorial equation or three coupled single channel equations (and how you couple them is given in [22].

The finite difference scheme looks like this:

$$\text{div}(f\nabla u) \approx \delta^o_{x,h/2}(f\delta^o_{x,h/2}u) + \delta^o_{y,h/2}(f\delta^o_{y,h/2}u) \tag{30}$$

where $\delta^o_{x,h/2}$ denotes the well known central difference scheme for the first derivative of u in the x-direction and $\delta^o_{y,h/2}$ is the same, but in the y-direction. h is the distance between two horizontally or vertically neighboring pixel positions (grid points) and most often may be set to 1. The central difference is given like:

$$\delta^o_{x,h/2} = \frac{u(x+h/2,y,t) - u(x-h/2,y,t)}{h} \tag{31}$$

which can be derived from Taylor approximations (also known as Taylor expansions). Now information in points between grid points, where no information exists, is required. This is however not major problem and to get to a solution, we continue by rewriting equation (30) using equation (31):

$$\begin{aligned}\text{div}(f\nabla u) \quad \approx \quad & \frac{1}{h^2}\big(f_{i+1/2j}u_{i+1j} + f_{i-1/2j}u_{i-1j} + f_{ij+1/2}u_{ij+1} + f_{i-1/2j}u_{ij-1}\big) \\ & -\frac{1}{h^2}\big(f_{i+1/2j} + f_{i-1/2j} + f_{ij+1/2} + f_{ij-1/2}\big)\end{aligned} \tag{32}$$

where ij is a short form for the (x,y)-coordinates. Now only values for $f = 1/\psi_i(|\nabla u|^2)$ are needed at half grid points (i.e. between pixel positions), and thus need to be interpolated. Now consider $f = 1/|\nabla u| = 1/\sqrt{(\partial_x u)^2 + (\partial_y u)^2}$. (Using $\psi(s^2) = \sqrt{s^2+\varepsilon^2}$ is equivalent to adding ε^2 in the square root.) Computing $\partial_x u$ at for example $(x,y) = (i-1/2, j)$ is easy using equation (31) but the $\partial_y u$-term needs to be interpolated, for example by using equation (31) for the y-direction on both sides of the half grid point:

$$\delta^o_y \approx \frac{\frac{u_{i-1j+1} - u_{i-1j-1}}{2h} + \frac{u_{ij+1} - u_{ij-1}}{2h}}{2} \tag{33}$$

Similar expressions can easily be found for the three remaining terms to get all four half grid point calculations of f in equation (32).

Using Taylor's approximation to arrive at numerical expressions can also be done for the v_i terms (second term in the middle part) of equation (29), $\text{div}_2(f|\nabla v_i|)$ where $f = 1/\psi_3(|\nabla v_i|^2)$ in a way very similar to the example given above. Details on this as well as descriptions on the somewhat more complex discretizations of the remaining two terms in equation (28) and (29) can be found in [22] and the references therein.

The technique can be generalized and used for many versions of equation (15) and (16) with different functions used for the E_i-terms, when it can be rewritten as partial differential equations. This is due to the fact that Taylor approximations allow for discrete (finite) re-writings of (partial) differential equations.

3.4 Motion Estimation

From the example given in the previous section, a separate Euler-Lagrange equation for the flow 'pops out' of the joint energy formulation for finding both intensities and flow. Leaving out the term(s) (E_0 and) E_1 yields the general energy formulation for computing flow alone and is the framework of state of the art flow computations, see e.g. [5].[9] It also illustrates how elegantly methods of finding intensities and determining the flow fits into one joint theoretical framework. Thus changing the flow calculations is to change the terms substituted for E_2 and E_3 in equations (15) and (16) and possibly adding additional constraints on the flow (that is to let several terms represent E_3).

Flows can also be calculated by other methods that do not fit into the framework of energy minimization given in this report, e.g. by block matching.[10] Should this other method produce better flow fields than variational based methods, it would be an improvement of the scheme, but variational methods produce the best and most accurate flow

[9] The term(s) (E_0 and) E_1 will disappear naturally when differentiating w.r.t. $\vec{v}$ in order to derive an Euler-Lagrange equation for the energy.

[10] Block matching is similar to the method given by Lucas and Kanade in [27] and thus close to the energy formulation in theory, but in practice it is a very different approach. All motion/flow estimators is based on the idea of looking for regions of similarity (in the form of blocks, single pixels, objects etc.) in neighboring frames of image sequences.

field of any methods available today (see e.g. the survey in [6]). The use of variational flow schemes is in coherence with the use of variational methods for intensity calculations and the two parts integrates nicely into one joint scheme to truly finding intensities and flows simultaneous in positions where both are unknowns. This approach is also in accordance with viewing this problem as a hen-egg problem.

In practice we ended up doing variational based flow field calculations using our own implementation of the total variation based scheme presented in [5]. The flow included in the example given in section 3.3.3 and equation (25) has two terms. The first is the optical flow constraint (OFC), the E_2-term, which has been the base of all variational based flow algorithms since it was introduced by Horn and Schunck in [16]. The second is the smoothness constraint on the gradient of the flow, the E_3-term, which accounts for the coherence between neighboring pixels as explained earlier. In the flow estimation algorithm from [5], which we ended up using, is similar: In the E_2 term the OFC is replaced by the brightness constancy assumption and contains an additional constraint.[11] The additional constraint that is added in the flow we use, is the gradient constancy assumption, which assumes that the spatial gradient of the intensities are constant along the flow. The brightness constancy assumption (and the OFC as well) is very sensitive to small changes in intensity and using the additional constraint on the gradient allows for small changes intensities while getting a more precise and robust scheme e.g. under changes in lighting.

The full three term total variation flow algorithm is

$$E(\vec{v}) = \lambda_1 \int \psi\big(|u(x+\vec{v}) - u(x)|^2 + \gamma|\nabla u(x+\vec{v}) - \nabla u(x)|^2\big)dx + \lambda_2 \int \big(\psi(|\nabla_3 v_1|^2) + \psi(|\nabla_3 v_2|^2)\big)dx \quad (34)$$

with ∇ being the spatial gradient and ∇_3 the spatio-temporal gradient. The derivation of the Euler-Lagrange equation and its discretization is done along the same lines as given in [22] and [5] using the semi discretized brightness constancy assumption instead of the OFC (and the same goes for the constant gradient assumption) making this work doable. The derivation yields this Euler-Lagrange equation (vectorial data term (E_2) and scalar regularization term):

$$\frac{\partial E}{\partial \vec{V}} = 2\psi' \left(\left| \frac{\partial u}{\partial \vec{V}} \right|^2 + \gamma \left| \frac{\partial \nabla u}{\partial \vec{V}} \right|^2 \right) \left[\frac{\partial u}{\partial \vec{V}} \nabla u(x+\vec{V}) + \gamma \mathcal{H}\big(u(x+\vec{V})\big) \frac{\partial \nabla u}{\partial \vec{V}} \right] = 0$$

$$\frac{\partial E}{\partial \vec{v_i}} = \lambda_3 \mathrm{div}_2 \left(\frac{\nabla v_i}{\psi_3(|\nabla v_i|^2)} \right) = 0, \qquad i = 1, 2 \quad (35)$$

where $\mathcal{H}$ is the spatial hessian and $\vec{V} = (v_1, v_2, 1)^T$.

We have made our numerical implementation using a fixed point solver to isolate the linear part of the system, which is then solved using a successive over-relaxation (SOR) run until either a convergence threshold or a maximum number of iterations is reached. The process is using a multiresolution setting to handle large motions and to speed up convergence, see [22], [23], [5], [4] and others, employing a down- and up-sampling scheme devised in [7].

As mentioned in section 3.2 we calculated the flow for the odd original fields and the even original fields separately, and for each of the two, we calculated both forward and backward flows and then had to 'fusion-scale-smooth' the pair of even and odd forward flow fields to end up with one full frame forward flow and ditto for the pair of even and odd backward flow fields to get one full frame backward flow. This merge is the subject of the next section.

3.5 Fusion-Scaling-Smoothing of Flow Fields

Let us just look at the forward flow as an example, knowing that the backward flow can be treated just the same, the mere difference being the direction of the flow.

We have two forward flow fields, one for the odd original fields and one for the even original fields, both containing flows from fields at time t to time $t+2$ w.r.t. to the original interlaced sequence. To get a full frame flow field as needed as input to the motion compensated deinterlacing, we need to *fuse* the even and odd flow fields into one full frame flow field, *scale* the flow to frames of twice the height of the fields and to be from time t to time $t+1$ and not $t+2$, and *smooth* the new flow field to get it consistent, remove outliers and other problems due to having calculated the

[11] The OFC can be seen as the linearized version of the discrete brightness constancy assumption and assumes differentiability as well. See [5] for further discussion.

flow separately on each half of the sequence and generally try to overcome the problem of working on Swiss cheese data (a.k.a. interlaced image sequences). This all ends up being a *fusion-scaling-smoothing* and can be expressed mathematically like this:

$$\vec{v} = \underset{\vec{v}}{\text{Arg min}} \int_K (\vec{v} - \vec{v}_{o/e})^2 d\vec{v} - \lambda \int_\Omega \psi(|\vec{v}|) dx \tag{36}$$

where composing $\vec{v}_{o/e}$ is the fusion-scaling part and the full minimization is the smoothing. As before $\psi(s^2) = \sqrt{s^2 + \varepsilon^2}$. The fusion is basically done by just interleaving the two flow fields and initializing the flow of the empty lines by line averaging. The scaling is initially done by halving the length of each flow vector to make them t to $t+1$ and doubling the size of the y-component to account for the doubled height going from vertically compressed fields to full frames. It is then up to the smoothing as given by equation (36) to complete the fusion and scaling to smooth out inconsistencies and give one optimally merged flow field. The smoothing uses total variation in its second term to avoid smoothing across boundaries between regions with different flows, and the first term ties the values to the input values to smooth out only noise (outliers) in the process.

To make this scheme even better one could device an additional forward-backward flow fusion to get optimal forward-backward consistency in the flow, but at a first glance at the problem, it does not come cheap in terms of complexity.

The removal of the assumption of flow being linear over two frame as mentioned in section 3.2 would also be of great interest and make the flow more accurate. This is of course only interesting in deinterlacing as long as we do calculations of time t to $t+2$ flows and need *fusion-scaling-smoothing* or similar algorithms depending of the choice of general strategy (choosing from the list in section 3.2. Working on removing this assumption will always be interesting in temporal super resolution (TSR), where you always do at least time t to $t+2$ flow calculations everywhere you insert new frames.[12]

3.6 The Motion Compensated Deinterlacing Scheme

With the motion estimation part of the scheme settled all we need now is the actual deinterlacing part. We chose to use the version given in section 3.3.3 extending it with a de-noising term, a E_0-term as suggested in equation (16) giving us

$$E(u) = \lambda_0 \int_K (u - u_0)^2 dx + \lambda_1 \int \psi_1(|\nabla u|^2) dx + \lambda_2 \int \psi_2(|\nabla u \cdot \vec{v} + u_t|^2) dx \tag{37}$$

with K being the domain of the original interlaced fields used as input to the algorithm. λ_0 looks superfluous, but serves an important purpose; set to zero it makes the scheme the exact same as in the example given in section 3.3.3. This enabled us to easily test both versions making the actual discrete implementation only slightly more complex.

The Euler-Lagrange equation corresponding to the energy formulation in (37) is

$$\frac{\partial E}{\partial u} = 2\lambda_0 \chi (u - u_0) - \lambda_1 \text{div}_2 \left(\frac{\nabla u}{\psi_1(|\nabla u|^2)} \right) - \lambda_2 \text{div}_3 \left(\frac{\psi_2'(|\nabla u \cdot \vec{v} + u_t|^2)}{|\nabla u \cdot \vec{v} + u_t|} (\nabla u \cdot \vec{v} + u_t) \vec{V} \right) = 0 \tag{38}$$

The scheme presented here for motion compensated deinterlacing has been used for inpainting, which is thoroughly described in [22] and [11] and as the motion estimation we use is also fairly well-known, see [22] and [5], the part predicted most likely to cause trouble is the scheme devised to use the motion estimator on interlaced sequences.

3.7 Experiments

Before we present the actual results from testing our algorithm, there is some details on the experimental setup that need to be mentioned.

[12] If you start discarding original frame in 'ultimate' TSR with an non-integer increase factor, e.g. 25 fps to 60 fps and keeping the frames distributed evenly in time.

3.7.1 Quality Measurements

The output quality of a deinterlacing scheme is often measured using objective measures as it has already been discussed shortly in section 3.1 and chapter 2. In [19] an extensive discussion (in Danish) is given on the advantages and drawbacks on objective as well as subjective measures and the topic is also touched upon in [2]. Objective measures have some major drawbacks and have not been used in this experiment and we will justify our choice of relying solely on subjective measures in this section.

First, the major pluses about objective measures is that they are easy to implement and makes it extremely easy to compare different algorithms. The most commonly used measure is the mean square error (MSE, also known as the L2-norm) presented in section 2.4, but also the rooted mean square error (RMSE) is often used whereas the highly questionable (see [19] Absolute mean error (AME, a.k.a. the L1-norm) is rarely used.[13]

The first of the two major problems in using MSE or any of the other measure given just above, is that one needs to do artificial deinterlacing in the sense that one removes half the lines from a progressive sequence to create an artificially interlaced sequence and still have a knowledge of what the correct pixels values in the lines to be deinterlaced are. This excludes any testing on interlaced recorded material. To overcome this problem Bellers and de Haan has in [12] and [2] suggested the MSE_i and MTI measures that can be used on interlaced recorded sequences. Both require knowledge of the flow and favors motion compensated deinterlacers as mentioned in section 3.1 but in their defense, they can be used to compare MC deinterlacers using the same flow as input.

The second and last major problem in using these measures is that they do not model the most important judge of the quality, the human visual system (HVS), very well. Thus, they are not a good measure of the quality really sought for in deinterlacing: perfect images free of artifacts as judged by the HVS. One could argue that you could measure the MSE at any pixel, but then it would not be mean anymore and the data just as extensive as the image sequence output and not any lighter to evaluate. In practice one would get one number per (short) test sequence or at the most a vector of values for each frame in the sequence. These numbers are then the perfect example of how bad objective measures model the HVS by smoothing out or hiding major local errors: one small region of maybe a few hundred pixels with bad artifacts highly noticeable to the HVS would not give a bad MSE (or other objective score) if the rest of the frame is fine. Actually one would be able to get a worse MSE score on sequence with a small error not noticeable to the HVS but spread out all over the frame.

Objective measures can, all though not very good, be used as an indication of what is bad and what is good, but an subjective evaluation is the only way to go about testing the quality of deinterlacers. In [2] a large scale subjective test of the difference in quality between progressive and interlaced HD formats is set up and the difference in judging between layman and experts is close to nothing. Thus we have chosen to do only subjective expert evaluation of our results and doing some A/B comparisons with Faroudja's DCDi® in the DVP-1010 Video Processor.

3.7.2 Test Setup

We have in our testing used four different displays (progressive CRT, LCD, Plasma and DLP projector) and found that results do differ significantly with a change of the display used.

The detailed non-realtime evaluation of the results of our own algorithm, looking at the sequences played slowly frame by frame or as stills was done on PC CRT monitor at different resolutions but always examining the sequences at their given resolution.

The Faroudja DVP-1010 processor is a 'closed box' system and due to Macrovision and HDCP copy protections we were unable to due any objective measures on it and all subjective evaluation had to be done realtime. The expert evaluation by the authors was done using an LCD PC monitor with a resolution of 1280 $\times$ 1024 and up-scaling from PAL (720 $\times$ 576) using the build in bilinear up-scaler of the DVP-1010 and additional testing was done on a Pioneer Plasma (1024 $\times$ 768, 43"). We feed the Faroudja interlaced sequences via the S-VHS 30 (using a $100cable) from a DVD - player(1000)$. The displays and the Faroudja processor was connected via DVI.

For the A/B panel comparison of the Faroudja and our algorithm (described later in section 3.7.5) we used the above setting for the Faroudja, but with a Samsung DLP-projector as display. The results of our algorithm had been

[13] MSE can also be rewritten as the picture (or peak) signal to noise ratio, $PSNR = 10\log(255^2/MSE)$, but is mostly used in this form in experiments testing the quality of coding.

recorded onto DVD video and was feed as a progressive PAL resolution (720 $\times$ 576 or 576p) via HDMI/DVI to the Faroudja and thus also up-scaled by the build-in bilinear algorithm in the Faroudja to the resolution of the projector.

Feeding the signal to the DVP-1010 via S-VHS ensured us that it did indeed receive an interlaced signal. DVI does not support interlaced at rates lower than 1080i, so we could not use that. Some would claim this would be favoring our algorithm as it was feed to the setup via HDMI/DVI, but actually we would claim, that the slight blurring of the image sequence done in the S- VHS connection would actually remove some of the details that might trouble the deinterlacer and thus actually help hide potential artifacts created by bad deinterlacer. Looking ahead to section 3.7.5 reveals that our algorithm clearly outperformed Faroudja and we thus never found it necessary to get the cables to test the analog YP_rP_b/RGB connection.

Extensive realtime evaluation of our progressive output results on DVD video was also done on the above mentioned Pioneer Plasma screen, which was feed the signal via a HDMI connection from the DVD-player.

3.7.3 Test Sequences

As test material we have used a mix of interlaced and progressive sequences, the later artificially interlaced by removing every other line alternating odd and even, so we have an original for comparison.

3.7.4 Parameter Testings and Intermediate Results (Flows)

In our algorithm there are 17 parameters to set. In each of the three steps it is: the maximal number of fixed point and SOR iterations and a convergence threshold for the inner linear solver. For the Motion estimation step it is the levels and scale in the multiresolution pyramid, and most importantly the weights γ and λ_3 in (35). For the fusion-scaling-smoothing it is the regularization weight λ in (36). In the Deinterlacing it is in (38) the spatial and temporal interpolations weights λ_1 and λ_2 and λ_0 controlling the de-noising. After initial testing we chose a set of standard settings and tested at least two alternative values to each parameter besides the values tested initially. Of course this far from covers all possibilities, but the amount of output data created in such a test and the time needed to evaluate it would require resources beyond our reach.

For motion estimation on PAL resolution material (720 $\times$ 576) we found that using 60-80 levels with a scale factor of 1.04 to be optimal. lowering the number of levels does affect the flow mildly, but any differences disappear almost completely when the flow is further processed with the fusion-scaling-smoothing. We suggest setting $\lambda_3 = 70$ as lower values will over-segment the flow and higher values will provide a unnecessary smoothing of the flow as some room for smoothing is needs to be left for the fusion-scaling-smoothing step. Setting γ = 100 is a good choice; higher values will possibly provide additional details to the flow, but these are smoothed out together with any over-segmentation in the fusion-scaling-smoothing step.

In the fusion-scaling-smoothing λ = 1 to 3 is fine; a higher values will produce a smoother flow, which can both help get rid of over-segmentation but also remove fine details in the flow. 10 outer fixed point and 100 inner iterations with a convergence threshold of 10^{-6} is fine. In figure 5 it can be seen how typical flow outputs look before and after fusion-scaling-smoothing.

In the deinterlacing step, setting $\lambda_0 = 0$ and thus shutting of any de-noising of original pixels results in more temporal flicker in moving detailed regions and we suggest using $\lambda_0 = 1$. Five fixed point outer and 50 inner iterations is recommended with a convergence threshold of 10^{-7}. The temporal weight λ_2 should be at least one, but values of five and 10 has also yielded good results, although they do introduce some smoothing in moving regions. To establish the best value, further research is needed to determine when the smoothing is just perceived as motion blur. For now we recommend $\lambda_2 = 5$ with spatial weighting $\lambda_1 = 0.1$ (increasing λ_1 increases noise in the form of temporal flicker).

Generally our three step algorithm has – within limits – proven to be quite robust to changes in parameter settings, but further parameter testing might still improve performance. Especially the flow calculations can give very detailed (or controlled poorly, over-segmented) flows, but they are smoothed out by the fusion-scaling-smoothing and it would be nice to make sure the most accurate and detailed flow possible reaches the deinterlacing, and thus working on improved motion estimation in deinterlacing will most likely prove worthwhile.

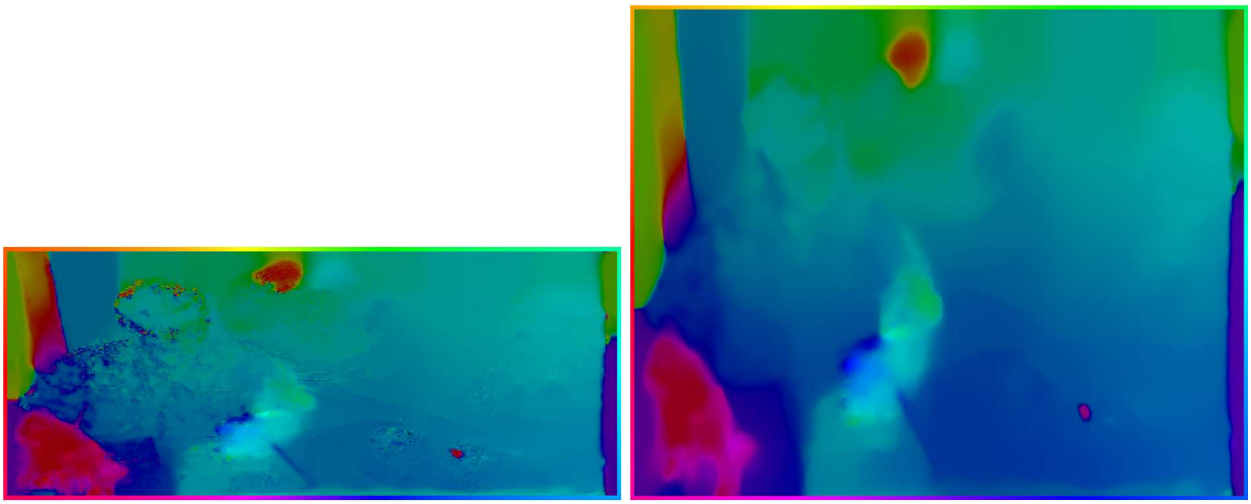

Figure 5: Typical flows, here from a sequence showing a person at a desk. The border shows the hue-coding of the direction of the flow and the saturation codes the magnitude of the flow. Left and right shows the flow before and after fusion-scaling-smoothing. The sub-sampling of interlaced image sequences often results in over-segmented flows before fusion-scaling smoothing

3.7.5 Results

On seven different test sequences with the interlacing problem – details in motion – our algorithm clearly outperformed the Faroudja DCDi® according to all five test persons. Each of the three test sequences having a progressive original was judged almost indistinguishable from their original. On an eighth sequence with very complex motion (close-up of fingers typing on a keyboard and hand held camera) the motion was not recovered correctly and some serration were observed by one person in the test panel. In figure 6 some examples of the high quality outputs are shown. Due to copy protection we are unable to produce stills or record video of the output from the Faroudja processor. On the `Copenhagen Pan` there is no flicker on the buildings in the output of our deinterlacer while Faroudja has some flicker and also produces quite a lot of serration on the roof windows of the castle, where our deinterlacer produces a perfect result. On `BBC3` details are recovered and motion is smooth, while Faroudja has staggering motion and the details are hard to see. On `Truck`, which was used to illustrate the interlacing problem in figure 4, we are able to (re)create the correct structure of the grill while Faroudja fails at this and creates some dark diagonal lines instead. The test sequence `Credits` (not shown) has rolling titles, a type of content on which most deinterlacers perform horribly, but the output of our algorithm is indistinguishable from the progressive original. The performance of the Faroudja is typical for motion adaptive deinterlacer: It is a bit better then the motion adaptive deinterlacer presented in [31] and is on the same level as the variational motion adaptive deinterlacer in [17], which we tested.

Comparing our results with the ones from the survey in [2], there is to few subjective results given in [2] to judge by. Taking the relatively small gap in objective evaluated performance compared to line averaging and the fact that no motion adaptive schemes where tested, we believe to outperform all the given schemes given the subjective performance of our deinterlacer presented above.

3.8 Conclusion

As the previous section revealed we have a very good deinterlacer, but it is also very computationally heavy. We devised a number of ways to use standard motion estimators on interlaced video and choose to use the one predicted to yield the best possible flows. We have shown that a technique so far used for inpainting (see [23]) can be redeveloped to do deinterlacing and further on our Total Variation Motion Compensated Deinterlacing outperforms Faroudja's state of the art deinterlacer. We produce high quality outputs when the input flow is correct and precise, and generally solve The Interlacing Problem. Problems with the output quality of one of eight test sequences and the analysis of its flow

Figure 6: Three examples of the high quality output of our Total Variation Motion Compensated Deinterlacer. Top: `Copenhagen Pan` (the motion in the sequence is a right-to-left camera pan), notice the details on the house in the bottom middle and the roof window of the castle. Left: `Truck`, compare to original in figure 4. Right: In this still from `BBC3` (motion is a rotation around an axis in the lower left corner of the frame) a few artifacts are seen, but they are not seen when the sequence is played back.

revealed that there is still work to be done on motion estimation on interlaced video.

3.9 Future Work

As shown in our test, we get high quality outputs when the input flow is correct and precise and the deinterlacing parts seems to perform optimally, but analysis of the flow on some sequences revealed obvious errors in the produced flow fields. They are either caused by how we use the motion estimator on interlaced video, the 2-2 scheme, or the motion estimator itself. Therefore it would be a good idea to test the motion estimator on progressive versions of troublesome sequences to optimize its performance. Initial results shows that it might be difficult to get our motion estimator to give highly accurate flows on all sequences (`Keyboard` being an example). Also trying one or more of the other ways of doing ME on interlaced sequences suggested in section 3.2 might prove worthwhile. Besides this immediate problem, performance could most likely be improved by other modifications; we could improve both the output quality and the running time.

To improve quality of the output we could just tweak the parameters of the existing system or better the numerical implementation of it. More drastically we could device better priors and maybe even data terms, but this should be done paying attention to the balance between mathematical tractability and quality as perceived by the human visual system. Switching to a Simultaneous MCDI, iterating between updating the flow and the intensities as done

for inpainting in [22] has to us the greatest potential of improvement, but will be computationally costly and calls for improving the running time.

To lower the computational cost and complexity of our scheme and get our software running fast enough to be converted to real time hardware (with a competitive price), we can do several things. The most obvious is to find a faster solver, the most probable choice being multigrid solvers for variational methods as presented in [7] and [8], which will give a tremendous speedup. In [8] a flow algorithm very similar to the one given in 3.3.3 is implemented using a multigrid solver and is able to compute accurate flow of 12 frames per second of a 120 x 160 sequence on a standard PC. This indicates that with dedicated hardware even a very complex variational motion compensated deinterlacer can become realtime at a reasonable cost.

A very obvious way to speed things up is by parallelization. In [20] a very high level parallelization of variational methods (in this case optical flow) is suggested by splitting frames into subregions computed on each their node. As variational methods are global, a complex scheme for with extensive communication (which only takes up a rather small fraction of the total computational time) to share boundary region data is needed. On 2000 x 2000 frames a speedup of 3.7 compared to a multigrid solver is achieved using 144 single CPU computers in a cluster. At least 25 computers are needed to be faster than one PC running a multigrid solver on the full data set. This is not very cost efficient but will probably be improved. The authors focus on finding a way of parallelization variational schemes on a mathematical level and formulating and coming up with a first solutions is in focus. They do not in their paper consider using computer architectural parallelization at all. Since they also seem to aim at doing computations on mainly scientific and medical data and focus on using standard PCs, this might not be the best choice for us unless significant improvements to the scheme is found. In our case aiming at dedicated hardware and a architectural parallelization would be in place, e.g. several processors iteratively solving the system in each their region with a shared memory (as a minimum for the pixel located at boundary between regions being processed at different processors). When running iterative solvers as discussed in section 3.3.3 it is not that important whether the values are the new $t + 1$-values or not so new t-values and one could just let all the pixels be scheduled randomly (or in a given order) to a number of processors with no need to consider boundaries. This require an all shared memory for the intensity and flow values. Any of these suggestions would lower the complexity and most likely lower the computational time of a multigrid solver almost linearly with the number of processors applied.

4 Conclusion

We have defined the Interlacing Problem only solvable by motion compensated deinterlacing.

The development of Total Variation Motion Adaptive Deinterlacing paved the way of variational methods entry into image sequence up-scaling and even though performing at the same level or higher than other state of the art motion adaptive deinterlacers w.r.t. output quality, it has clearly been outperformed by it younger sibling, Total Variation Motion Compensated Deinterlacing (TV MC DI). We have seen no rival to TV MC DI when it comes to output quality, and even though a heavy algorithm in terms of running times, we do strongly believe it will be able to run in realtime in hardware in the near future. We do not intend to spend any time in the future on further development of our motion adaptive deinterlacer, but will focus our deinterlacing work on the motion compensated version.

Up next is also the development and implementation of variational super resolution in time as well as spatially – both methods seems directly at hand from the common up-scaling framework of simultaneous computation of flow field and intensities presented in chapter 3.

References

[1] G. Aubert and P. Kornprobst. *Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations*, volume 147 of *Applied Mathematical Sciences*. Springer-Verlag, January 2002.

[2] E.B. Bellers and G. de Haan. *De-interlacing. A Key Technology for Scan Rate Conversion*. Elsevier Sciences Publishers, Amsterdam, 2000.

[3] M. Biswas and T.Q Nguyen. A novel de-interlacing technique based on phase plane correlation motion estimation. *International Symposium on Circuits and Systems, ISCAS*, 2:604–607, 2003.

[4] M.J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *CVGIP: Image Understanding*, 63(1):75–104, 1996.

[5] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In T. Pajdla and J. Matas, editors, *Proceedings of the 8th European Conference on Computer Vision*, volume 4, pages 25–36, Prague, Czech Republic, 2004. Springer–Verlag.

[6] A. Bruhn, J.Weickert, and C.Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[7] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr. Variational optic flow computation in real-time. *IEEE Trans. Image Proc.*, 14(5):608–615, 2005.

[8] Andrés Bruhn, Joachim Weickert, Timo Kohlberger, and Christoph Schnörr. Discontinuity-preserving computation of variational optic flow in real-time. In Ron Kimmel, Nir A. Sochen, and Joachim Weickert, editors, *Scale Space and PDE Methods in Computer Vision, 5th International Conference, Scale-Space 2005, Proceedings*, volume 3459 of *LNCS*, pages 279–290, Berlin, 2005. Springer.

[9] L. Capodiffero. Interlaced to progressive conversion by median filtering. In L. Chiariglione, editor, *Signal Processing of HDTV, II*. Elsevier Sciences Publishers, Amsterdam, 1990.

[10] T. Chan and J. Shen. Mathematical models for local nontexture inpainting. *SIAM journal of appl. Math*, 62(3):1019–1043, 2002.

[11] Laurent Chanas. *Méthodes Variationnelles pour la Restauration de Séquences d'Images Fortemenent Dégradées. Application aux Images Infrarouges Eblouies par Laser*. PhD thesis, Université de Cergy-Pontoise, October 2001.

[12] G. de Haan and E.B. Bellers. Deinterlacing, an overview. In *Proc. IEEE*, volume 86, 1998.

[13] T. Doyle and M. Looymans. Progressive scan conversion using edge information. In L. Chiariglione, editor, *Signal Processing of HDTV, II*. Elsevier Sciences Publishers, Amsterdam, 1990.

[14] H.J. Dreier. Line Flicker Reduction by Adaptive Signal Processing. In L. Chiariglione, editor, *Signal Processing of HDTV, II*. Elsevier Sciences Publishers, Amsterdam, 1990.

[15] P. Haavisto, J. Juhola, and Y. Neuvo. Scan rate up-conversion using adaptive weighted median filtering. In L. Chiariglione, editor, *Signal Processing of HDTV, II*. Elsevier Sciences Publishers, Amsterdam, 1990.

[16] B.K. Horn and B.G. Schunck. Determining Optical Flow. *Artificial Intelligence*, 17:185–203, 1981.

[17] S. Keller, F. Lauze, and M. Nielsen. A total variation motion adaptive deinterlacing scheme. In R. Kimmel, N. Sochen, and J. Weickert, editors, *Procedings of the 5th Scale-Space Conf.*, volume 3459 of *LNCS*, pages 408–419, Berlin, 2005. Springer.

[18] S. Keller, F. Lauze, and M. Nielsen. Variational motion compensated deinterlacing. In *SMVP, ECCV 2006 Workshop Proceedings*, 2006.

[19] Sune Høgild Keller. PDE based deinterlacing. Master's thesis, IT University of Copenhagen, Denmark, 2004.

[20] T. Kohlberger, C. Schnörr, A. Bruhn, and J. Weickert. Domain decomposition for variational optical flow computation. *IEEE Trans. Image Proc.*, 14(8):1125–1137, 2005.

[21] J. Kovacevic, R.J. Safranek, and E.M. Yeh. Deinterlacing by successive approximations. *IEEE Transactions on Image Processing*, 6(2):339–344, 1997.

[22] F. Lauze. *Computational Methods For Motion Recovery, Motion Compensated Inpainting and Applications*. PhD thesis, IT University of Copenhagen, 2004.

[23] F. Lauze and M. Nielsen. A variational algorithm for motion compensated inpainting. In S. Barman A. Hoppe and T. Ellis, editors, *British Machine Vision Conference*, volume 2, pages 777–787. BMVA, 2004.

[24] François Lauze, Sune Keller, and Mads Nielsen. A method of adding information to a frame or a field in a video sequence.

[25] François Lauze, Sune Keller, and Mads Nielsen. A method of adding information to a frame or a field in a video sequence.

[26] G. Lu. *Communication and Computing for Distributed Multimedia Systems*. Artech House, Boston, MA, 1996.

[27] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[28] D. Mumford. Bayesian rationale for the variational formulation. In *Geometry-Driven Diffusion In Computer Vision*, pages 135–146. Kluwer Academic Publishers, 1994. Computational Imaging And Vision.

[29] S. Pigeon and P. Guillotel. Advantages and drawbacks of interlaced and progressive scanning formats. *CEC RACE/HAMLET*, 1995. Deliverable no R2110/WP2/DS/R/004/b1.

[30] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[31] A. Skarabot, G. Ramponi, and L. Buriola. FPGA architecture for a videowall image processor. In *Proceedings of the SPIE International Symposium on Electronic Imaging*, volume 4304, pages 75–84, 2001.

[32] A. Skarabot, G. Ramponi, and D. Toffoli. Image sequence processing for videowall visualization. In *Proceedings of the International Society for Optical Engineering*, volume 3961, pages 138–147, 2000.

[33] A.M. Tekalp. *Digital Video Processing*. Prentice-Hall, 1995.

[34] G.A. Thomas. A comparison of motion-compensated interlace-to-progressive conversion methods. *Signal Processing, Image Commun.*, 12(3), 1998.

[35] Y. Wang, J. Ostermann, and Y.Q. Zhang. *Video Processing and Communications*. Prentice-Hall, 2002.