(12) UK Patent Application (19) GB (11) 2 431 794 (13) A

(43) Date of A Publication 02.05.2007

(21) Application No: **0522159.3**

(22) Date of Filing: **31.10.2005**

(71) Applicant(s):
Sony United Kingdom Limited
(Incorporated in the United Kingdom)
The Heights, Brooklands, WEYBRIDGE,
Surrey, KT13 0XW, United Kingdom

(72) Inventor(s):
Karl James Sharman

(74) Agent and/or Address for Service:
D Young & Co
120 Holborn, LONDON, EC1N 2DY,
United Kingdom

(51) INT CL:
H04N 5/44 (2006.01) ***H04N 7/01*** (2006.01)

(52) UK CL (Edition X):
H4F FER

(56) Documents Cited:
None

(58) Field of Search:
UK CL (Edition X) **H4F**
INT CL^7 **H04N**
Other:

(54) Abstract Title: **Filtering pixels**

(57) Image processing apparatus in which output pixels of an output image are generated with respect to input pixels of an input image, a subset of the output pixels being identical to respective input pixels and another subset of the output pixels being derived from respective groups of one or more input pixels comprises a vertical filter for filtering the subset of pixels derived from respective groups of input pixels so as to attenuate high vertical frequencies in that subset of pixels, the filter not being applied to filter the subset of output pixels which are identical to respective input pixels.

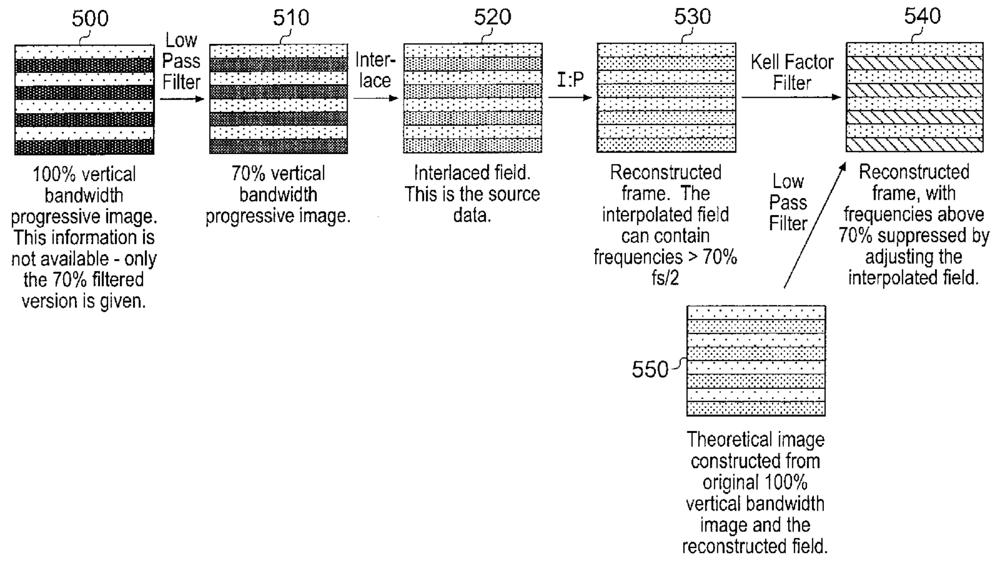
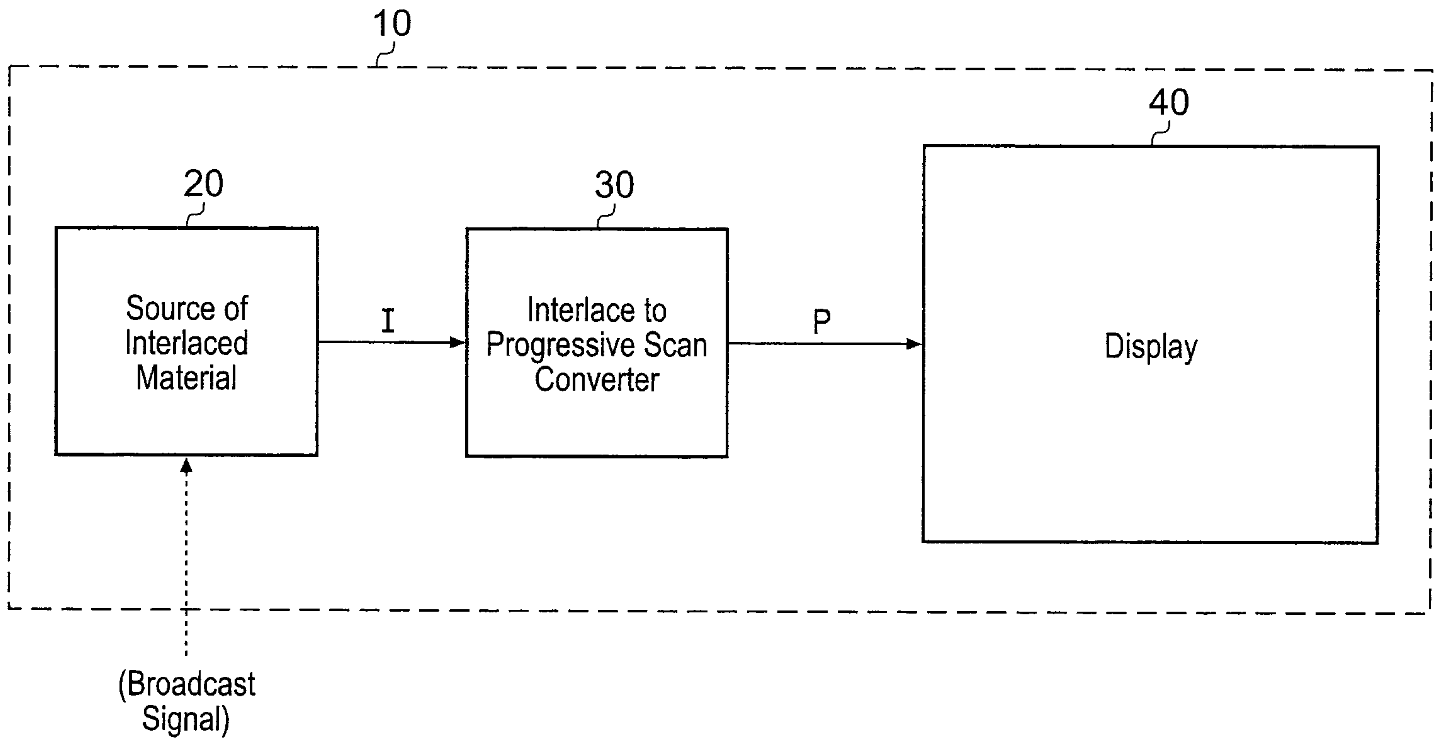

500
Low Pass Filter
510
Inter-lace
520
I:P
530
Kell Factor Filter
540
100% vertical bandwidth progressive image. This information is not available - only the 70% filtered version is given.
70% vertical bandwidth progressive image.
Interlaced field. This is the source data.
Reconstructed frame. The interpolated field can contain frequencies > 70% fs/2
Low Pass Filter
Reconstructed frame, with frequencies above 70% suppressed by adjusting the interpolated field.
550
Theoretical image constructed from original 100% vertical bandwidth image and the reconstructed field.

Fig. 11

GB 2 431 794 A

29 11 05



10
20
Source of Interlaced Material
I
30
Interlace to Progressive Scan Converter
P
40
Display
(Broadcast Signal)

Fig. 1

29 11 05

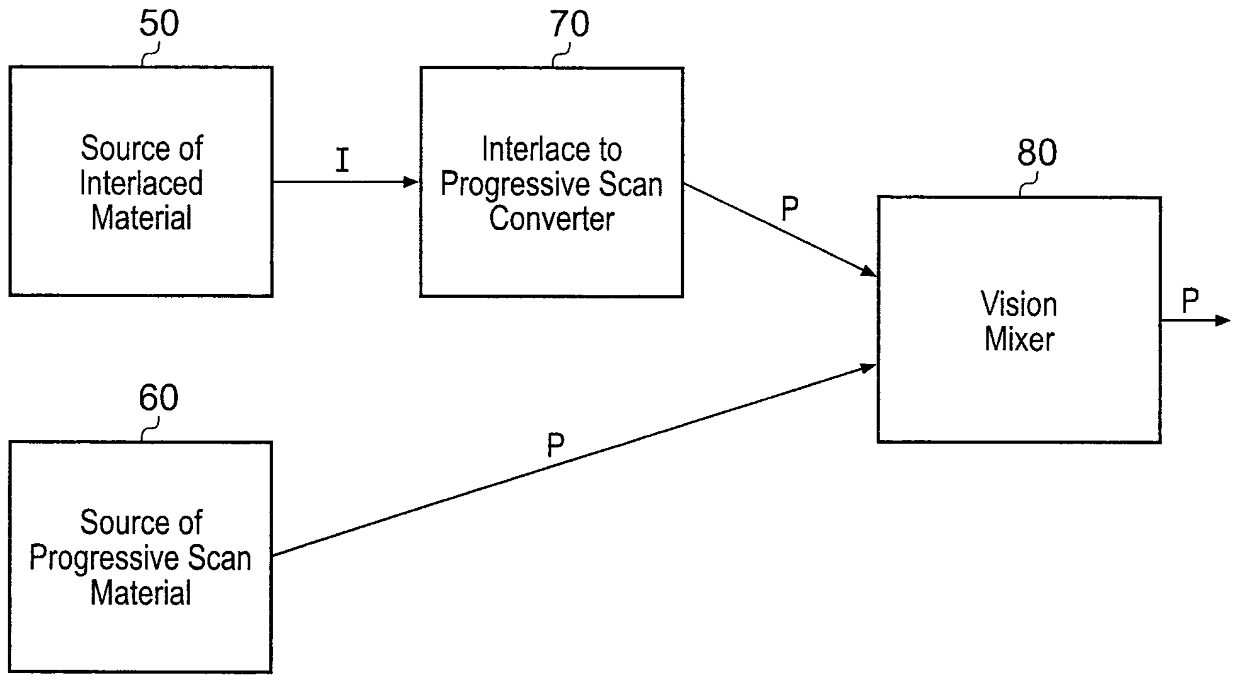

50
Source of Interlaced Material
I
70
Interlace to Progressive Scan Converter
P
60
Source of Progressive Scan Material
P
80
Vision Mixer
P

Fig. 2

22 11 05

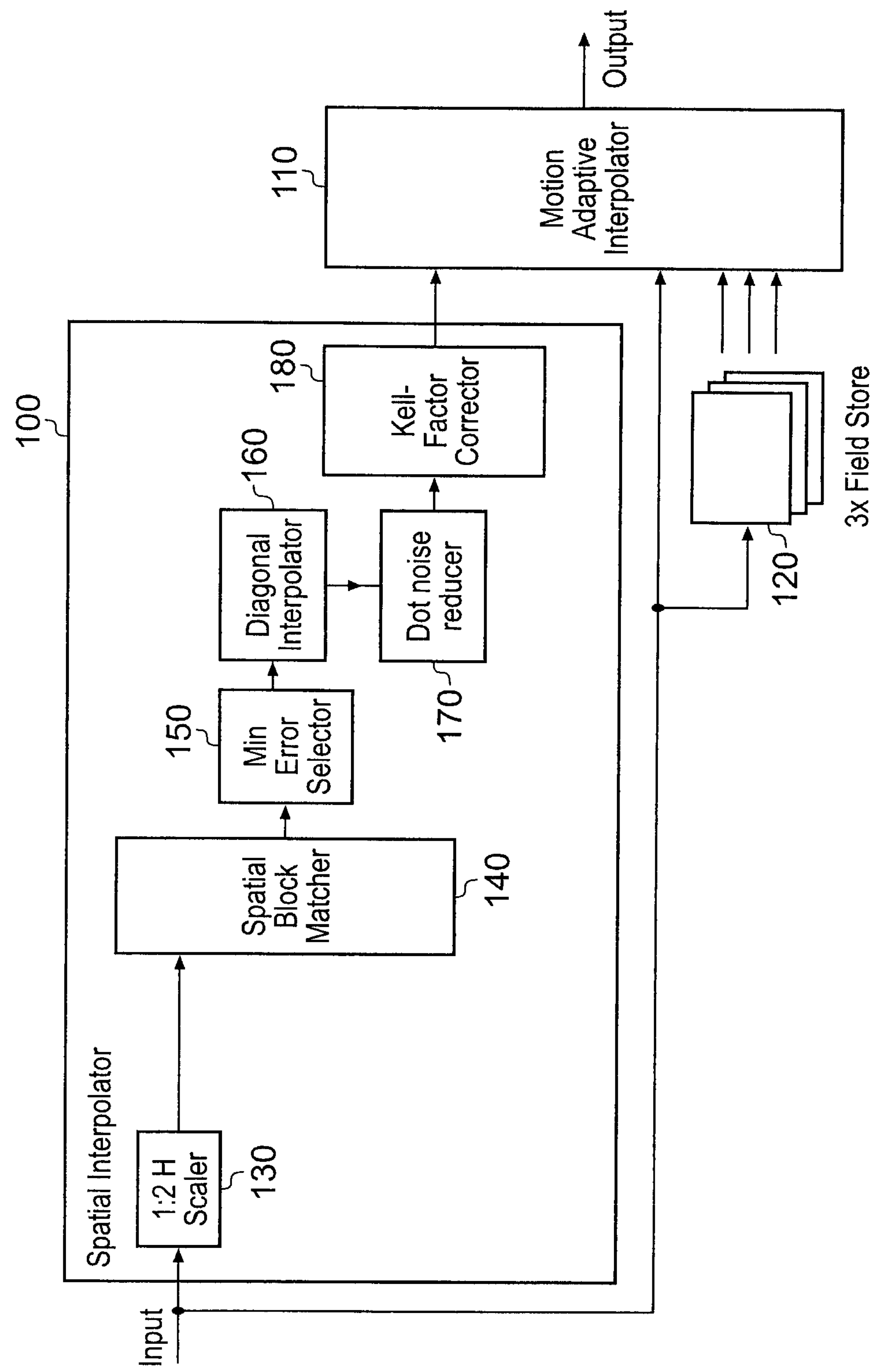

Input
Spatial Interpolator
100
1:2 H Scaler
130
Spatial Block Matcher
140
Min Error Selector
150
Diagonal Interpolator
160
Dot noise reducer
170
Kell-Factor Corrector
180
Motion Adaptive Interpolator
110
Output
3x Field Store
120

Fig. 3

29 11 05

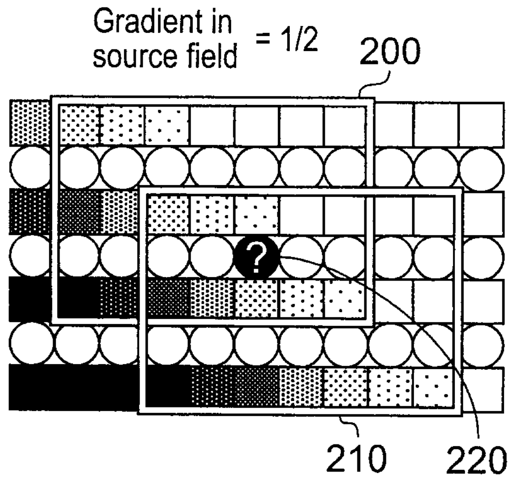


Fig. 4A

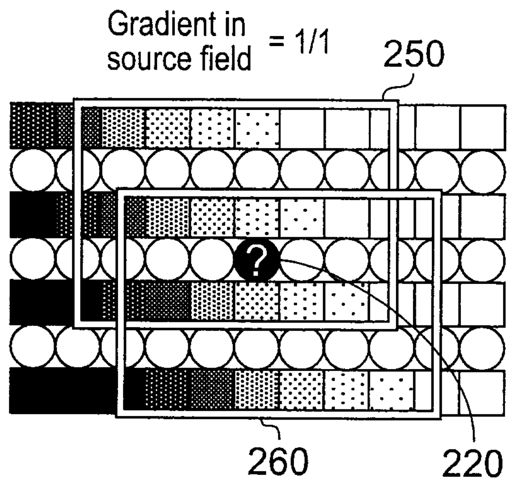


Fig. 4B

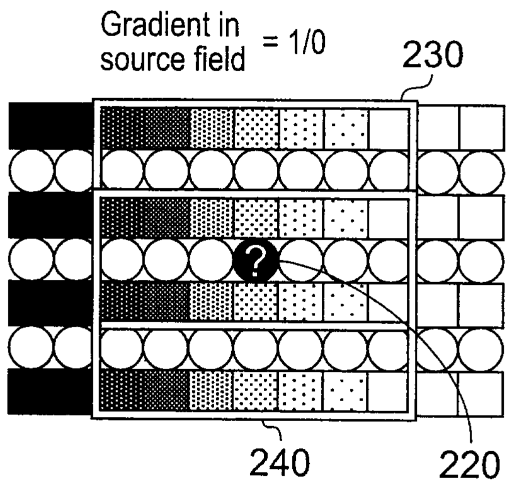
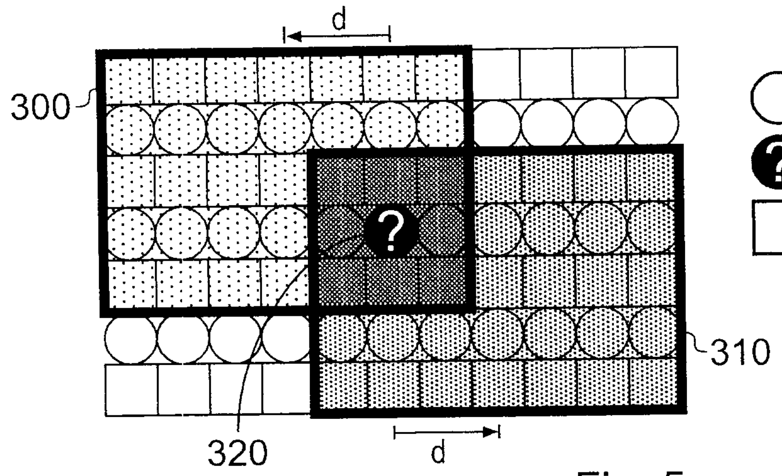


Fig. 4C

24 11 05



○ = Missing input field pixel (to be interpolated)

● = Missing input field pixel under test

□ = Current input field pixel

Fig. 5

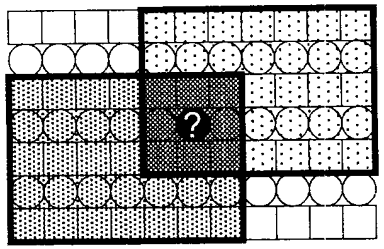


Fig. 6A

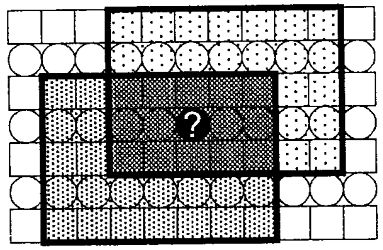


Fig. 6B

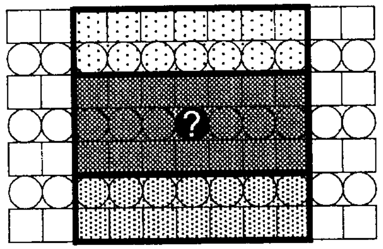


Fig. 6C

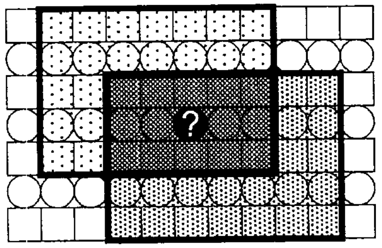


Fig. 6D

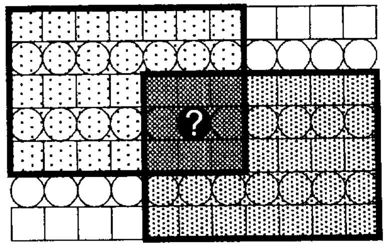


Fig. 6E

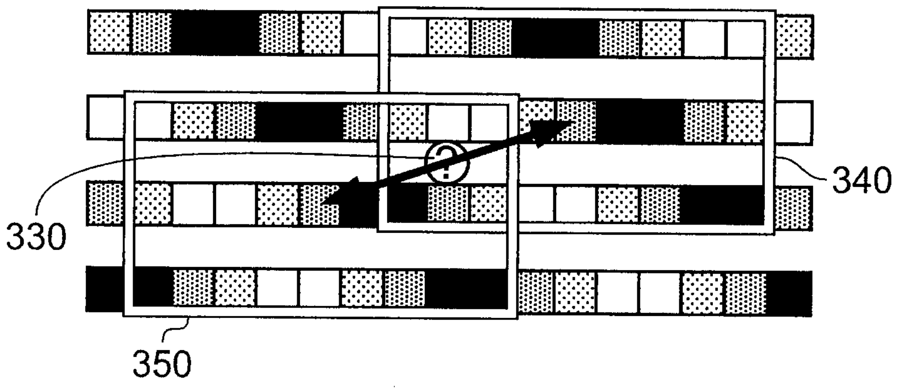


Fig. 7A

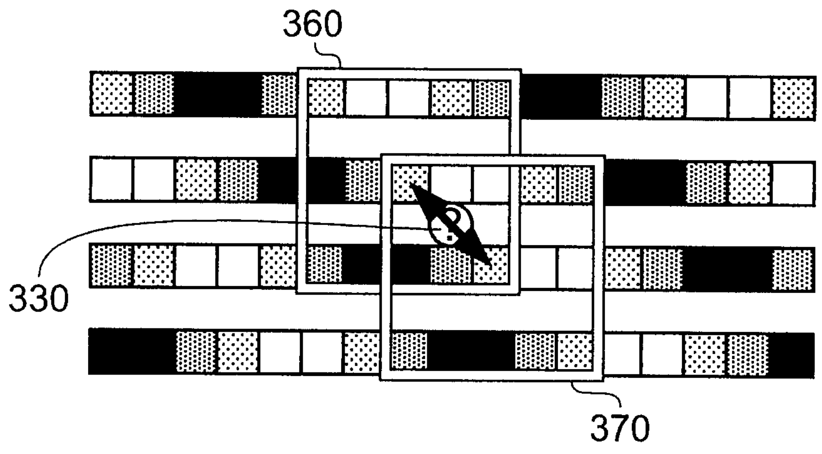


Fig. 7B

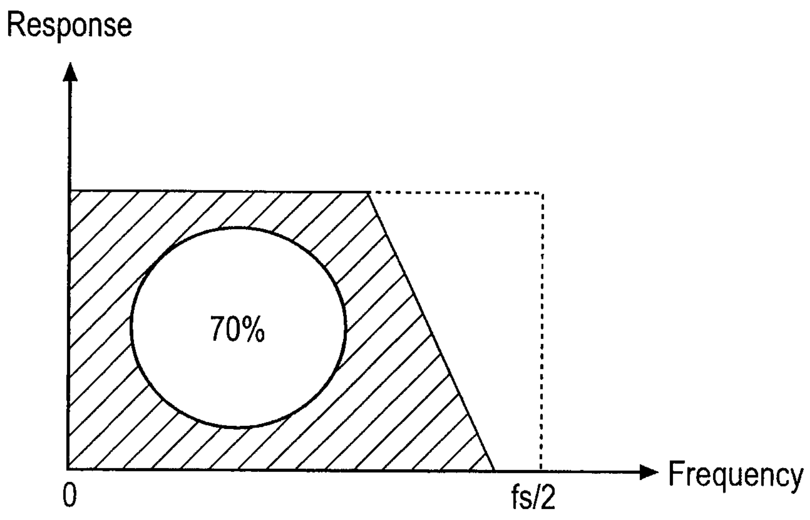


Fig. 9

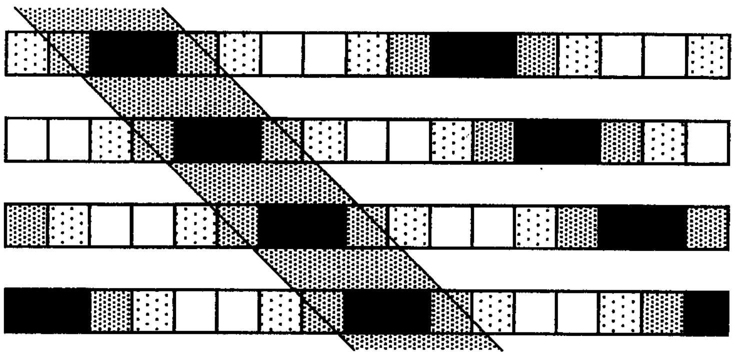


Fig. 8A

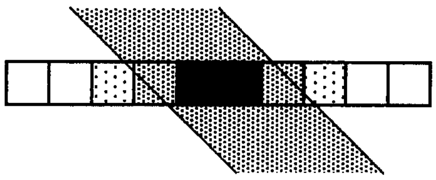


Fig. 8B

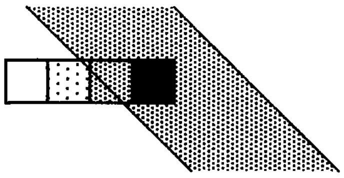


Fig. 8C

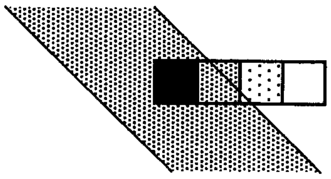


Fig. 8D

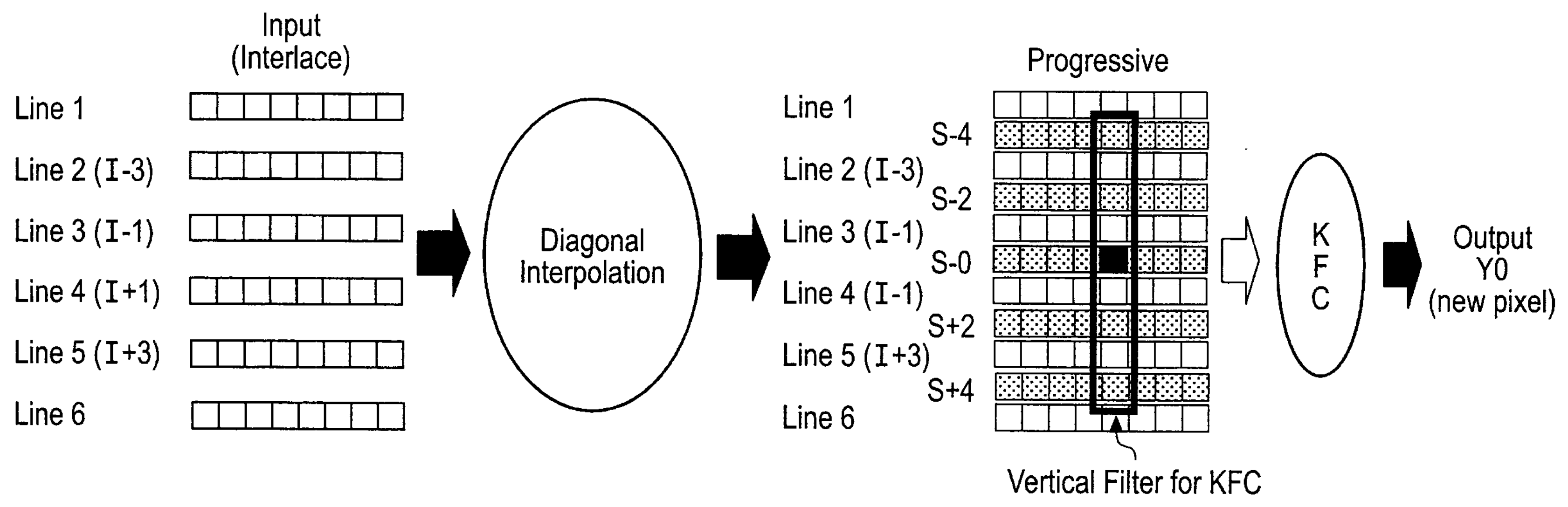

Input
(Interlace)
Line 1
Line 2 (I-3)
Line 3 (I-1)
Line 4 (I+1)
Line 5 (I+3)
Line 6
Diagonal Interpolation
Line 1
Line 2 (I-3)
Line 3 (I-1)
Line 4 (I-1)
Line 5 (I+3)
Line 6
S-4
S-2
S-0
S+2
S+4
Progressive
Vertical Filter for KFC
K
F
C
Output
Y0
(new pixel)

Fig. 10

29 11 05

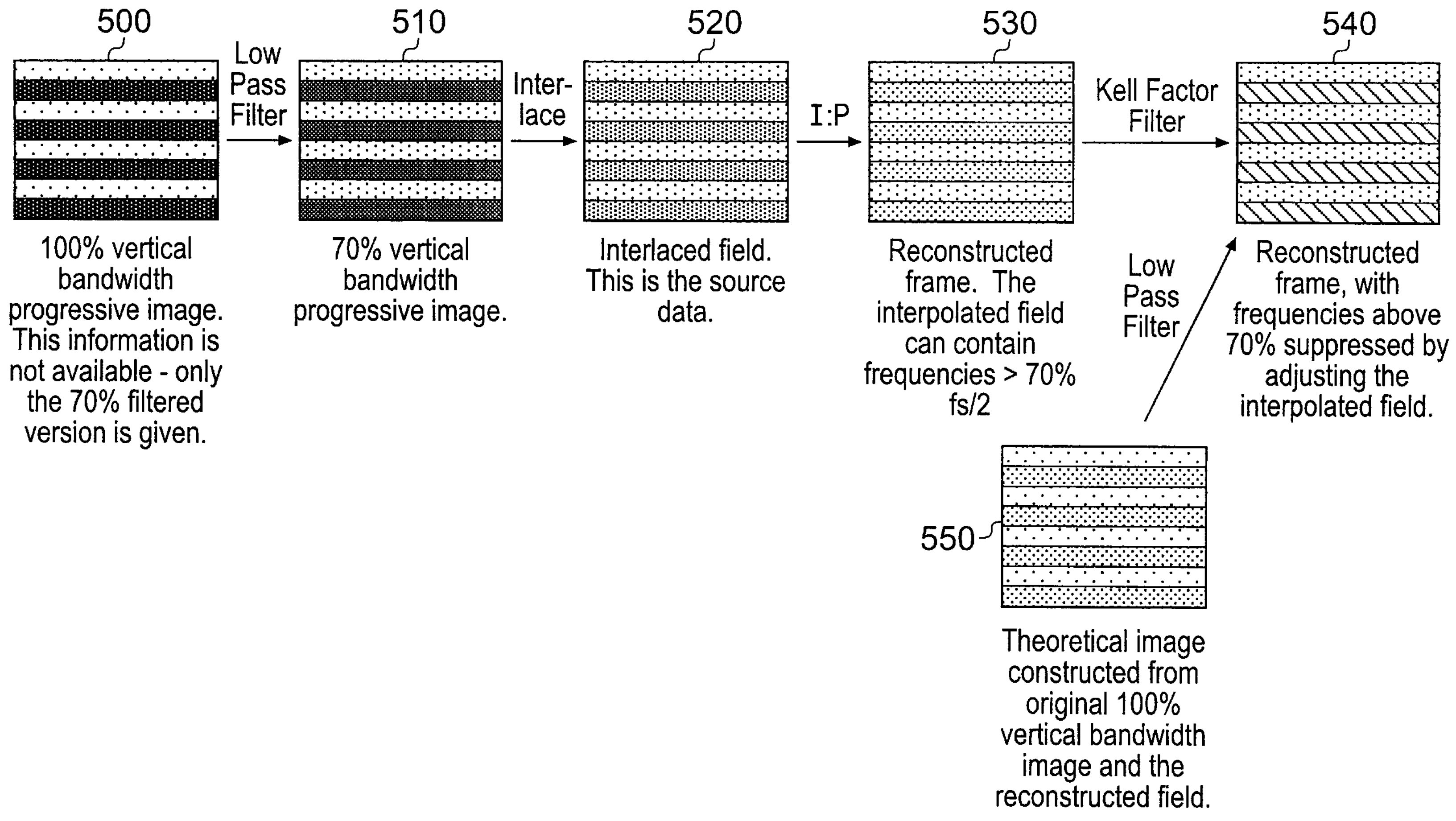

500
Low Pass Filter
510
Inter- lace
520
I:P
530
Kell Factor Filter
540
Low Pass Filter
550
100% vertical bandwidth progressive image. This information is not available - only the 70% filtered version is given.
70% vertical bandwidth progressive image.
Interlaced field. This is the source data.
Reconstructed frame. The interpolated field can contain frequencies > 70% fs/2
Reconstructed frame, with frequencies above 70% suppressed by adjusting the interpolated field.
Theoretical image constructed from original 100% vertical bandwidth image and the reconstructed field.

Fig. 11

20 11 05

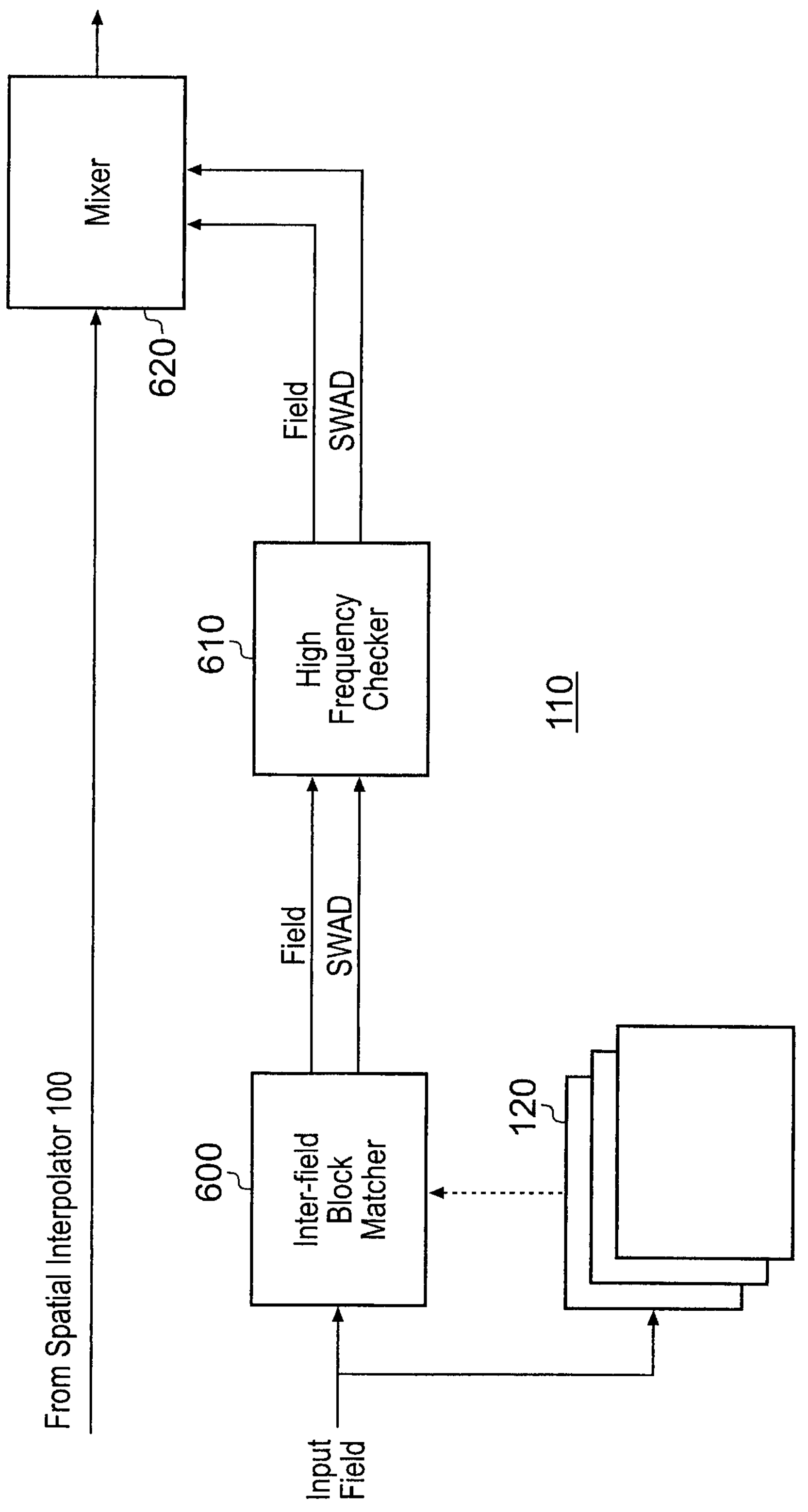


Fig. 12a

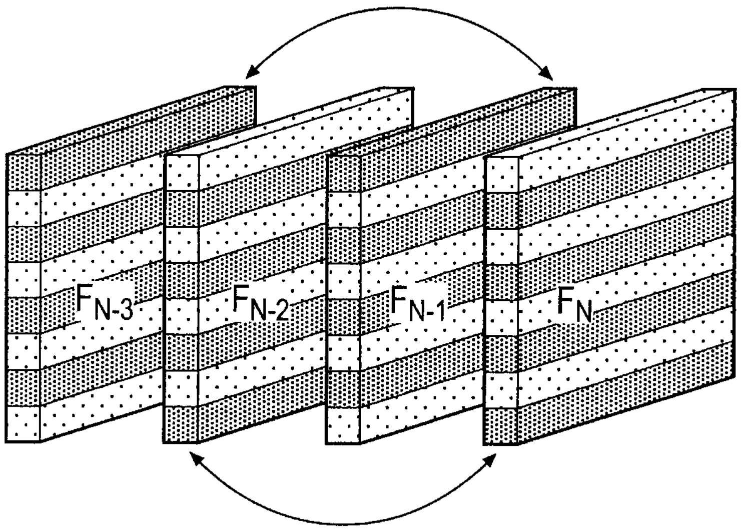


Fig. 12B

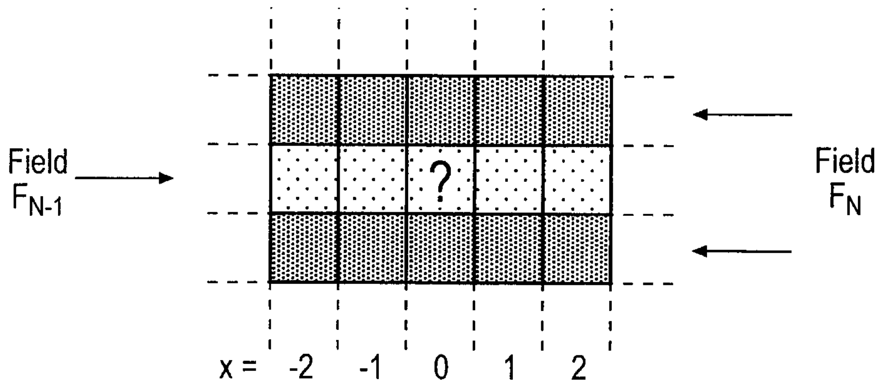


Fig. 13

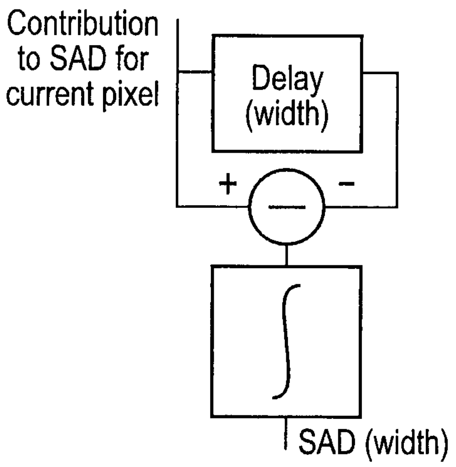


Fig. 18

23.11.05

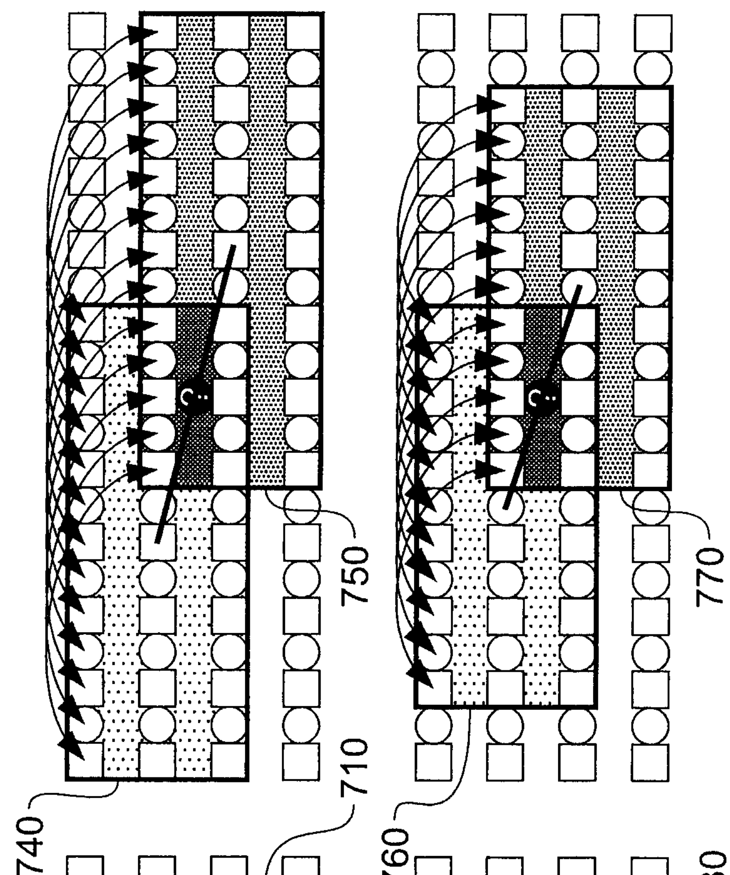


Fig. 14b

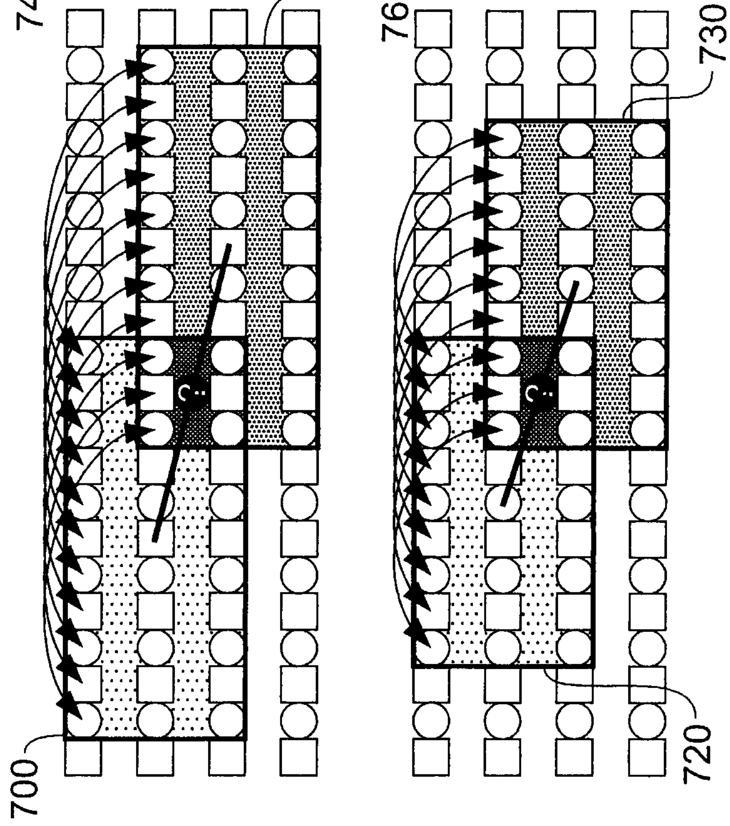


Fig. 14a

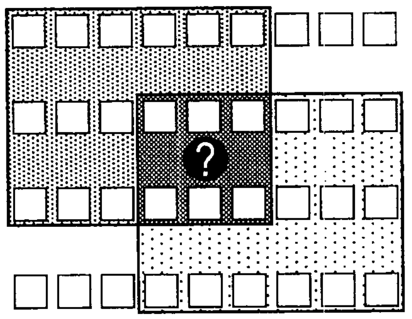
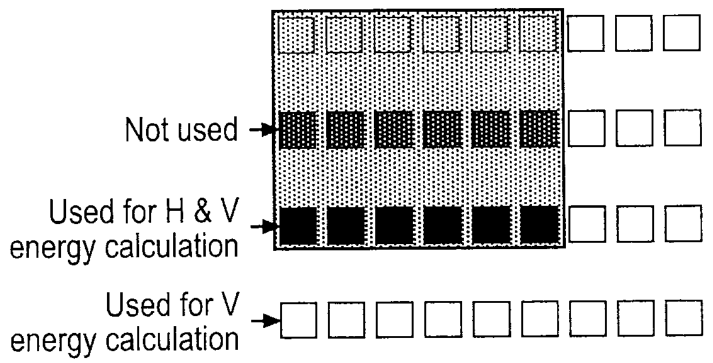


Fig. 15


Not used
Used for H & V energy calculation
Used for V energy calculation
Fig. 16A

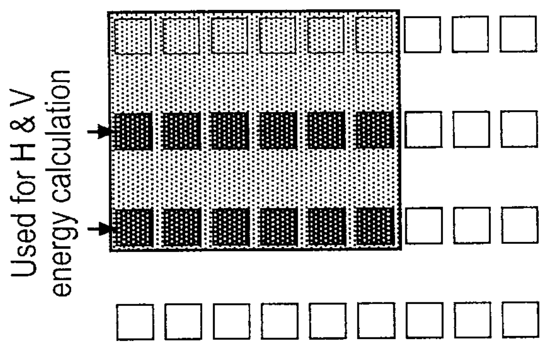
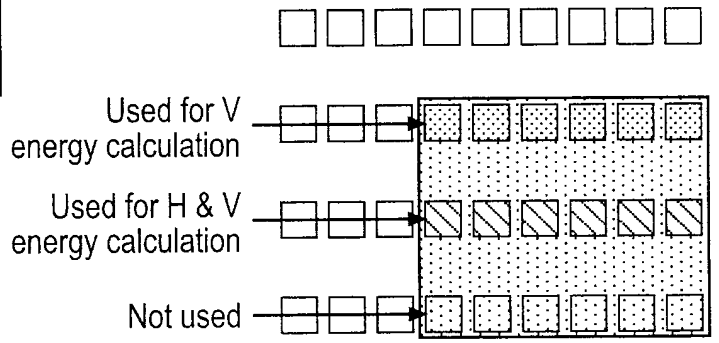


Fig. 17A


Used for V energy calculation
Used for H & V energy calculation
Not used
Fig. 16B

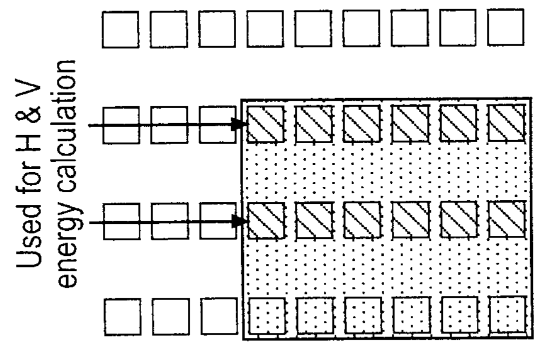


Fig. 17B

2431794

IMAGE PROCESSING

This invention relates to image processing.

Video image capture represents a spatial and temporal sampling process. An image is captured as a set of pixels arranged in rows or lines. Successive images are captured at spaced instants in time.

The smallest detail that can be reproduced in the image is a single pixel. Ideally, each of the smallest details of the original scene should be represented by one pixel in the captured image.

In practice, however, some of the details in the scene fall between pixel lines, so two pixel lines are required for such small details. Thus some vertical resolution can be lost. Measurements of this effect show that only about 70% of the vertical detail which ought to be captured is in fact provided by the pixel lines. This factor (70%) is known as the Kell-factor.

Some authorities assert that the effect quantified Kell-factor applies irrespective of the manner of image scanning, i.e. whether the lines follow each other sequentially (progressive scan) or alternately (interlaced scan). Other authorities assert that the Kell-factor relates only to the degradation caused by interlacing.

In the present description, it is those effects relating to the Kell-factor which are dependent upon the image being interlaced which are considered. In these circumstances, it is generally considered that only 70% (the Kell-factor) of the possible vertical bandwidth is (or should be) represented. But in fact any effect which gives rise to the source material not making full use of its available bandwidth is also contemplated.

It will of course be appreciated that the figure of 70% is a mere approximation for the purposes of the present description.

Hence if an interlaced to progressive scan conversion is being carried out, it is potentially hazardous to attempt to produce a *full* vertical bandwidth image. Instead, a compensation to account for a Kell-factor of less than unity may be used.

This invention provides image processing apparatus in which output pixels of an output image are generated with respect to input pixels of an input image, a subset of the output pixels being identical to respective input pixels and another subset of the output pixels being derived from respective groups of one or more input pixels, the apparatus comprising:

a vertical filter for filtering the subset of pixels derived from respective groups of input pixels so as to attenuate high vertical frequencies in that subset of pixels, the filter not being applied to filter the subset of output pixels which are identical to respective input pixels.

The invention recognises that a simple approach to Kell-factor filtering of an image generated as defined above (i.e. just to filter the entire output image with a 70% bandwidth vertical low-pass filter) would not be expected to produce a useful result. This is because some of the pixels in the output image correspond to pixels in an input image which has already been captured or generated by a technique which limits the vertical bandwidth by the Kell-factor.

In contrast, in the present invention, such pre-existing pixels are not altered by the filtering operation.

Further respective aspects and features of the invention are defined in the appended claims.

Embodiments of the invention will now be described, by way of example only, with reference to the accompanying drawings in which:

Figure 1 schematically illustrates a flat-screen display arrangement;

Figure 2 schematically illustrates video mixing operation in a studio environment;

Figure 3 schematically illustrates an interlace to progressive converter;

Figures 4a to 4c schematically illustrate gradient detection;

Figures 5 and 6a to 6e schematically illustrate a spatial block matching operation;

Figures 7a and 7b schematically illustrate an alias situation;

Figures 8a to 8d schematically illustrate alias detection techniques;

Figure 9 schematically illustrates a Kell-factor filtering operation;

Figures 10 and 11 schematically illustrate stages in a Kell-factor filtering operation;

Figure 12a schematically illustrates a motion adaptive interpolator;

Figure 12b schematically illustrates motion detection between successive video fields;

Figure 13 schematically illustrates a high frequency check operation;

Figures 14a and 14b schematically illustrates a variation which reduces the need for image scaling;

Figures 15, 16a, 16b, 17a and 17b schematically illustrate a primary and an alternative pixel testing strategy; and

Figure 18 schematically illustrates the generation of a difference value.

Figure 1 schematically illustrates a flat screen display arrangement 10 comprising a source of interlaced video material 20, an interlace to progressive scan converter 30 and a display panel 40 such as a liquid crystal (LCD) or plasma display. This illustrates a typical use of interlace to progressive scan conversion, in that many broadcast signals are in the interlaced format whereas many flat panel displays operate most successfully in a progressive scan format. Accordingly, in Figure 1, a broadcast signal received by the source of interlaced material 20 is used to generate an interlaced signal for display. This is passed to the interlace to progressive scan converter 30 to generate a progressive scan signal from the interlaced signal. It is the progressive scan signal which is passed to the display 40.

It will be appreciated that the source of interlaced material 20 need not be a broadcast receiver, but could be a video replay apparatus such as a DVD player, a network connection such as an internet connection and so on.

Figure 2 schematically illustrates a video mixing operation in a studio environment, in order to give another example of the use of interlace to progressive scan conversion. Here, a source of interlace material 50 and source of progressive scan material 60 are provided. These sources could be cameras, video replay apparatus such as video tape recorders or hard disk recorders, broadcast receivers or the like.

The interlaced output from the source of interlaced material 50 is supplied to an interlace to progress scan converter 70 to generate a progressive scan signal. This can be processed by the vision mixer 80 along with the progressive scan material from the source 60 to generate a processed progressive scan output. Of course, the progressive scan output of the vision mixer 80 can be converted back to an interlaced format if required, e.g. for subsequent broadcast or recording. It will also be appreciated that the vision mixer 80 is just one example of video processing apparatus; instead, a digital video effects unit, for example, could be used at this position in Figure 2.

Figure 3 schematically shows an interlace to progressive scan converter. In general terms, the converter comprises a spatial interpolator 100, a motion adaptive interpolator 110 and a set of three field stores 120.

The converter of Figure 3 operates to generate output progressive scan frames at the same repetition frequency as the input interlaced fields. Therefore, a main requirement of the converter is to generate the “missing” pixels in each interlaced field to turn that interlaced field into a progressive scan frame. This can be achieved in one of two ways. On one hand, the spatial interpolator 100 generates the “missing” pixels by spatial interpolation within the field concerned. In other words, this is an intra-field operation. On the other hand, the motion adaptive interpolator generates the missing pixels by inserting pixels from an adjacent field of the opposite polarity. This is valid only if there is no image motion between the fields, so the basic organisation of Figure 3 is that the output of the spatial interpolator 100 is used at image positions where image motion is detected, while the output of the motion adaptive interpolator 110 is used at pixel positions where image motion is not detected. For simplicity of operation, the spatial interpolator operates at each pixel position, and the motion adaptive interpolator either selects the output of the spatial interpolator, or selects a pixel from another field of the opposite polarity for output, or mixes the two.

The motion adaptive interpolator will be described in more detail below. First, the spatial interpolator will be described.

The spatial interpolator comprises a 1-2 horizontal pixel scaler 130, a spatial block matcher 140, a minimum error selector 150, a diagonal interpolator 160, a dot noise reducer 170 and a Kell-factor corrector 180. The operation of each of these will be described in more detail below.

The scaler 130 uses horizontal linear interpolation to generate one additional pixel value between each two pixels of the input interlaced field. So, the horizontal resolution (at least in terms of number of available pixel values) is doubled, but no difference is made at this stage to the vertical resolution.

The reason why horizontal scaling is appropriate will be described with reference to Figures 4A to 4C.

As will be described below, the overall operation of the spatial block matcher 140 and the diagonal interpolator 160 is to detect the orientation of an image feature relevant to a pixel position where a new pixel is to be interpolated, and then to apply an interpolation along that image feature direction. So, if a current pixel position to be interpolated lies within a diagonal image feature (a line, an edge etc.) at, say, 45° to the horizontal, interpolation of that new pixel would take place along that 45° direction. This can tend to give a better output result than restricting the interpolation to horizontal or vertical interpolation. A key part of this process, clearly, is therefore to detect the direction of an image feature at each pixel position.

This detection is carried out using a block matching process. The actual block matching process will be described in much more detail below. But at this stage, Figure 4A schematically illustrates a successful block match between two blocks 200, 210 of pixels around the position of an unknown pixel 220 (a circle with a question mark inside). Indeed, the notation used in the present drawings is that a square indicates a known pixel where as a circle indicates a pixel to be interpolated by the diagonal interpolator 160. The shading in Figures 4A to 4C is a schematic representation of an image feature.

So, referring to Figure 4A, a successful block match is obtained between the blocks 200, 210 around the unknown pixel position 220, indicating a gradient of an image feature of 1/2.

Turning now to Figure 4C, an image feature is vertical and there is again a successful block match between overlapping blocks 230, 240.

However, in Figure 4B, the image feature has a gradient of 1/1. It is not possible to obtain a successful block match with the blocks at integral pixel positions. A successful match between blocks 250, 260 occurs at a half integral pixel position. Accordingly, in order to detect gradients of this nature (indeed any gradients sharper than 1/2), it is necessary to operate at a sub-pixel accuracy. In the present case, a half-pixel accuracy was adopted. If a greater accuracy still was used, (e.g. quarter-pixel accuracy) then gradients yet closer to vertical could be detected. In this embodiment, half-pixel accuracy was selected as a balance between processing requirements and accuracy at very acute gradients.

In fact, in embodiments of the invention to be described below, not all of the image is subjected to 1:2 scaling by the scaler 130 at any one stage. This selective use of scaling will be described in more detail below.

Figures 5 and 6A to 6E schematically illustrate the spatial block matching operation.

As noted above, spatial block matching is carried out at sub-pixel accuracy; in this case half-pixel accuracy.

A range of block sizes is used, with corresponding search ranges (maximum displacements relative to the pixel position under test). Taking into account the 1:2 scaling operation, example block sizes and search ranges are given in the following table:

Block Size (in scaled pixels)	**Search Range (in scaled pixels)**
3v x 3h	0h
3v x 5h	±1h
3v x 7h	±2h
...	...
3v x 41h	±19h

Figure 5 schematically illustrates a block match operation between two blocks of 3v x 7h pixels 300, 310, around an unknown pixel position 320. The variable d signifies a horizontal displacement of the block's horizontal centre from the pixel position under test. A condition applied to the block matches is that the blocks must always overlap the pixel position under test. Also, the blocks are displaced in integral numbers of real pixel displacements (so a displacement of *m* corresponds to a displacement of 2*m* interpolated pixels). Accordingly, the particular block size shown in Figure 5 allows five possible tests at a displacement of -2 pixels (Figure 6A) -1 pixel (Figure 6B), 0 pixels (Figure 6C), +1 pixel (Figure 6D), and +2 pixels (Figure 6E).

Note that the displacement is indicated as a displacement from the centre, in scaled pixels. The two blocks are displaced by equal amounts, though in opposite directions. Symmetrical displacements are used because otherwise the block matching could detect lines or edges which are not relevant to the pixel under test.

A sum of absolute differences (SAD) is calculated for each block match. This is defined as:

$$SAD(x,y,d,n) = \sum_{dx=-n}^{n} \sum_{dy=-3,-1,1} \sum_{RGB/YCbCr} \left| p(x-d+dx, y+dy) - p(x+d+dx, y+dy+2) \right|$$

where x, y represent the current pixel co-ordinate (y being a frame line number), d is the displacement being tested, and n is the radius of the block (the block width is n'=2n+1).

In general terms, the SAD values for three colour components (red, green and blue) are combined, and a minimum normalised SAD value determines a gradient for interpolation. Various checks are made to avoid poor interpolation, as described below.

Measures are taken to avoid problems caused by alias situations. Figures 7A and 7B illustrate a possible alias situation.

Referring to Figure 7A, a block match between blocks 340 and 350 suggests that an unknown pixel 330 should be a dark grey colour. Here, the block match is 100% successful and so the SAD value would be zero (note that this is a schematic example!)

However, in Figure 7B, a block match between blocks 360 and 370 is also 100% successful, again giving a SAD value of zero. The block match of Figure 7B suggests that the unknown pixel 330 should be a light grey colour.

This conflict of block match results is a product of aliasing between the closely spaced diagonal image features in the image portions shown in Figures 7A and 7B. While it may at first appear that either diagonal line is equally valid (i.e. a steep diagonal line from upper left to lower right or a more gentle diagonal line from upper right to lower left) is valid, a processing rule has been set up to allow an appropriate selection to be made. This rule will now be described.

The basis of the rule is that the block match process is restricted to that only areas considered to be "line segments" are detected. That is to say, each block in a block match should contain a line segment.

A digitised line segment is considered to have two properties. Firstly, it is monotonic along the central scan line row of the block in question, and secondly there is a vertical transition between scan lines in the block in question. The way in which these tests are applied will be described with reference to Figures 8A to 8D.

In Figure 8A, a source field contains multiple diagonal lines. Figure 8B schematically illustrates one row of pixels within the image of Figure 8A. Figures 8C and 8D illustrate the two edges of the diagonal line shown in Figure 8B. It will be seen that each of these edges has a region of pixels which show a monotonic variation in luminance. Also, referring back to Figure 8A, it can be seen that such segments exhibit a vertical transition between adjacent rows of pixels.

So, turning back to Figures 7A and 7B, the block match of Figure 7A would be rejected in favour of the block match of Figure 7B according to the rule described above. This is because the central line of pixels of the two blocks of Figure 7B shows a monotonic variation in luminance, whereas the centre line of pixels of the blocks 340, 350 in Figure 7A does not.

In fact, the rule is not expressed in such stark terms. The actual rule involves a balance between the various properties of the line as follows:

$$\text{Monotonicity} = \sum_{y=-1}^{0}\left(\left\{\sum_{i=2}^{blockWidth}\left|x(i,y)-x(i-1,y)\right|\right\}-\left|x(blockWidth,y)-x(1,y)\right|\right)$$

$$\text{Average Vertical Energy} = \left(\sum_{i=1}^{blockWidth}\left|x(i,0)-x(i,-1)\right|\right)/\,blockWidth$$

The rule is that:

Monotonicity + C1 + (C2/blockwidth) < Average vertical energy

where C1, C2 are constants which allow for noise in the data. If this condition is met, the relevant block match is used as a valid interpolation direction.

Note that the tests are performed separately in respect of each of the colour components (e.g. R, G and B). All three tests must be passed separately. Alternatively, for example to save hardware, fewer than three tests could be performed. For example, only the luminance, or only one colour component, might be tested. Of course, a YCbCr or YPbPr representation could be tested instead.

A second rule, hinted at above, which is applied to block matching is that the minimum normalised SAD value determines the gradient for subsequent interpolation.

Normally, the direction is selected from the block match with the smallest error (smallest normalised SAD). In such cases, the Kell-factor correction of the corrector 180 will *not* later be used if the normalised SAD is less than a threshold KFC_{thresh}.

However, if the normalised SAD value is similar (within a threshold amount) for multiple block offsets or if the smallest normalised SAD value obtained across all of the block matches is greater than a predetermined maximum value, then vertical interpolation will be used, Kell-factor correction will always be used in these cases and the normalised SAD value is set to the predetermined maximum value mentioned above.

A third rule applicable to block matching is as follows.

The basis of the rule is that neighbouring 'unknown' (to be interpolated) pixels should indicate a similar gradient to a current pixel under consideration.

For the current 'unknown' pixel, let G be the predicted gradient. Let the search range be $|G|/4+1$ (the division is necessary since 'G' is in terms of scaled pixels). Let C3 ...be programmable constants

To apply the rule, examine all pixels in the current missing line at positions from –search range to +search range (inclusive) relative to the current pixel. Each predicted gradient, g, in that range that satisfies the following inequality increments a value 'counter':

$(|G| - d) <= C3(g-G) <= (|G| + d)$

Then, if counter*C4 < C5*(2 x range+1), let v be the original value and v' be the vertically interpolated value, and set the result to C6 v' + (1- C6)v (preferred version) or C6 v' + C7v (more general version). Increment the normalised SAD score for the pixel by C8, which constant can be set to encourage the use of the KFC.

The diagonal interpolator 160 is a simple pixel averager: given a direction it picks the pixel in that direction on the line below and the pixel in that direction on the line above and averages them.

The dot noise reducer 170 involves a process which is applied to the output of the diagonal interpolator 160. A test is applied to detect whether an interpolated pixel lies within the maximum and minimum values of the four vertically and horizontally pixels, i.e. the pixels immediately above, below, left and right of the interpolated pixel. Note that the pixels above and below the interpolated pixel will be real pixels, whereas those to the left and right will be interpolated themselves.

If the interpolated pixel does not lie within this range, then;

Let v be the original value of the pixel under consideration, and let v' be v, clipped to lie within the range of the four locally neighbouring pixels.

Let the new pixel value be kDNR v' + (1- kDNR)v, where kDNR is programmable.

The operation of the Kell-factor corrector 180 will now be described with reference to Figures 9 to 11.

Kell-Factor Correction

In the present discussion, references to the Kell-factor are simply to help explain the operation of this part of the system. What the filter is actually exploiting is simply the knowledge that the source image did not use the full bandwidth available to it, whether that is because of scanning artefacts or because of a low pass filtering process.

The Kell-factor is a quantity which represents a property of progressive scan and interlaced images. In order to represent the information being scanned, it is generally considered that only 70% (the Kell-factor) of the possible vertical bandwidth is (or should be) represented. Hence when performing an interlace to progressive scan conversion, it is potentially hazardous to attempt to produce a *full* vertical bandwidth image. Instead, a compensation to account for a Kell-factor of less than unity may be used.

One method to compensate for the Kell-factor would be to use a 70% bandwidth filter on the frame output of any interlace to progressive scan algorithm. However, one of the fields in the frame is 'real' data – i.e. it was sampled correctly, so the content arising from that field must by definition be perfect. Thus a method to filter just the interpolated lines is used.

A generic Kell-factor filtering operation is schematically illustrated in Figure 9.

The following mathematics and algorithm describe a filter suitable for performing a 70% bandwidth filter on interpolated lines. In empirical tests the filter has been found to significantly reduce artefacts, and it is important to note that the results are also significantly better than simply performing a 1:2 vertical up-scaling operation on the input interlaced video.

Figures 10 and 11 schematically demonstrate the concept. The general idea is shown in Figure 10, whereby a vertical filter is (selectively) applied to real pixels I and interpolated pixels S in a progressive scan frame.

To explain the technique it will be useful to consider some data which does not exist – in particular, the missing lines of an interlaced field. By treating these lines (for part of the derivation) as though they do exist, filter techniques can be derived to apply to the interpolated versions of those lines so as to generate an approximation to a filtered version of the missing lines. This will become clear from the following explanation.

Referring to Figure 11, a hypothetical 100% vertical bandwidth progressive image 500 is considered. This does not of course exist; at best it would be 70% filtered, due to the inherent filtering in the sampling process. This forms a realisable progressive frame 510. However, even this frame is not available to us; instead, it is provided in interlaced form as a field 520. The field 520 is the source data to the present interlace to progressive scan conversion process.

The interlaced field 520 is supplied to an interlace-to-progressive converter, which then generates the missing field lines to produce a reconstructed frame 530. However, the frame 530 may now contain frequencies that are above 70% of the available bandwidth. Generating such frequencies is not always desired, since the information was not present in the source data to the interlace to progressive scan converter. Hence a version 540 of this frame that does not exceed this bandwidth limit should be created, and this should be achieved by adjusting just the reconstructed field lines and not the source field lines – those source field lines are from an image that has already been passed through such a filter, and are therefore 'perfect'.

Producing the required output can therefore be thought of as the low-pass filter output of a theoretical image 550 constructed from the lines from the original 100% vertical bandwidth image interlaced with the reconstructed field. However, the original 100% vertical bandwidth image is not available, and so a technique to circumvent this problem is now described.

<u>A 3-tap source filter method</u>

This is the first of the three methods to create a suitable filter; only the latter two methods are recommended, and this method is included in the present description to show progression.

Consider a 3 tap low-pass symmetrical filter of the form

$$filter = \begin{bmatrix} a \\ b \\ a \end{bmatrix}$$

and also a column of pixels, P, from five lines of video that has been converted from interlace to progressive scan:

$$input = \begin{bmatrix} P_{-2} \\ P_{-1} \\ P_0 \\ P_1 \\ P_2 \end{bmatrix} = \begin{bmatrix} I_{-2} \\ F_{-1} \\ I_0 \\ F_1 \\ I_2 \end{bmatrix}$$

where F are pixels from a real field of data, and I are interpolated pixels. Note that these values are known.

The real field pixels, F, must be output from the system. These have already been filtered according to the Kell-factor during the recording of the video. Hence consider the raw video prior to the filtering – this is a conceptual video source, since the actual Kell-factor filter would be intrinsic in the sampling of the optical information. Let this raw, or pseudo input video be described as:

$$pseudoInput = \begin{bmatrix} I_{-2} \\ x_{-1} \\ I_0 \\ x_1 \\ I_2 \end{bmatrix}$$

where x are unknown values from the conceptual 100% vertical bandwidth source, and the I are pixels that have been spatially interpolated, and

$output = pseudoInput * filter$

Let us now consider output pixel, y_0, which is vertically aligned with pseudo input pixel I_0.

$$\begin{bmatrix} \vdots \\ y_0 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ I_{-2} \\ x_{-1} \\ I_0 \\ x_1 \\ I_2 \\ \vdots \end{bmatrix} * \begin{bmatrix} a \\ b \\ a \end{bmatrix} = \begin{bmatrix} \vdots \\ ax_{-3} + bI_{-2} + ax_{-1} \\ aI_{-2} + bx_{-1} + aI_0 \\ ax_{-1} + bI_0 + ax_1 \\ aI_0 + bx_1 + aI_2 \\ ax_1 + bI_2 + ax_3 \\ \vdots \end{bmatrix}$$

We do not know the value of x_{-1} and x_1, and thus the value of the output pixel, y_0, is not immediately known. However, we know that $y_{-1}=F_{-1}$ and $y_1=F_1$, since the output field lines for the current field are known.

Thus x_{-1} and x_1 can be found, and subsequently so can y_0:

$$x_{-1} = \frac{1}{b}\left(y_{-1} - a(I_{-2} + I_0)\right) = \frac{1}{b}\left(F_{-1} - a(I_{-2} + I_0)\right)$$

$$x_1 = \frac{1}{b}\left(y_1 - a(I_0 + I_2)\right) = \frac{1}{b}\left(F_1 - a(I_0 + I_2)\right)$$

$$y_0 = a(x_{-1} + x_1) + bI_0$$

$$y_0 = \begin{bmatrix} -a^2/b \\ a/b \\ b-2a^2/b \\ a/b \\ -a^2/b \end{bmatrix} \cdot \begin{bmatrix} I_{-2} \\ F_{-1} \\ I_0 \\ F_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} -a^2/b \\ a/b \\ b-2a^2/b \\ a/b \\ -a^2/b \end{bmatrix} \cdot \begin{bmatrix} P_{-2} \\ P_{-1} \\ P_0 \\ P_1 \\ P_2 \end{bmatrix}$$

Now, a suitable 70% vertical bandwidth filter is given by the vector:

$$vFilter3Tap70pcnt = \frac{1}{32768}\begin{bmatrix} 5138 \\ 22492 \\ 5138 \end{bmatrix}$$

Hence a=5138 and b=22492. Thus:

$$y_0 = \begin{cases} \begin{bmatrix} P_{-2} \\ P_{-1} \\ P_0 \\ P_1 \\ P_2 \end{bmatrix} * \begin{bmatrix} -1174 \\ 7485 \\ 20145 \\ 7485 \\ -1174 \end{bmatrix} & \text{for } P_0 \text{ aligned to interpolated lines} \\ P_0 & \textit{otherwise} \end{cases}$$

Hence the problem is actually solved using a 5-tap filter.

<u>A 5-tap source filter method</u>

The initial symmetrical 3-tap filter has relatively poor filtering characteristics, and thus a method to improve upon this by using an initial symmetrical 5-tap filter was developed.

However, this method also requires the use of the 3-tap filter, otherwise the problem does not have a solution.

Let the symmetrical 5 tap filter consist of three coefficients: a, b, c, and the full filter be described by [a, b, c, b, a] ^{T}.

Similarly, let the 3 tap filter consist of two coefficients: d, e, and the full filter be described by [d, e, d] ^{T}.

From before, but extending to use the 5-tap filter,

$$\begin{bmatrix} \vdots \\ y_0 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ I_{-2} \\ x_{-1} \\ I_0 \\ x_1 \\ I_2 \\ \vdots \end{bmatrix} * \begin{bmatrix} a \\ b \\ a \end{bmatrix} = \begin{bmatrix} \vdots \\ aI_{-4} + bx_{-3} + cI_{-2} + dx_{-1} + eI_0 \\ ax_{-3} + bI_{-2} + cx_{-1} + dI_0 + ex_1 \\ aI_{-2} + bx_{-1} + cI_0 + dx_1 + eI_2 \\ ax_{-1} + bI_0 + cx_1 + dI_2 + ex_3 \\ aI_0 + bx_1 + cI_2 + dx_3 + eI_4 \\ \vdots \end{bmatrix}$$

If we attempt to solve this using the same method as before, the problem requires solving a set of simultaneous equations with as many unknowns as half the height of the image. This would thus not be suitable for a sensible hardware design.

Instead, considering the predicted form of the filter, it is likely that the results from the more distant neighbours have a reduced effect on the output than the immediate neighbours of y_0: standard low-pass filters have this form of heavy weighting around the central tap. Therefore, when considering a pixel y_0, the result from the central tap and the immediate neighbours can be considered to be formed from a 5-tap filter, and the more distant results from a 3-tap filter. Hence the following equation:

$$\begin{bmatrix} \vdots \\ y_{-1} \\ y_0 \\ y_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ F_{-1} \\ y_0 \\ F_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ dI_{-2} + ex_{-1} + dI_0 \\ aI_{-2} + bx_{-1} + cI_0 + bx_1 + aI_2 \\ dI_0 + ex_1 + dI_2 \\ \vdots \end{bmatrix}$$

We can now solve for y$_0$, eliminating the unknown 'x's:

$$x_{-1} = \frac{F_{-1} - dI_{-2} - dI_0}{e}$$

$$x_1 = \frac{F_1 - dI_0 - dI_2}{e}$$

$$\begin{aligned} y_0 &= aI_{-2} + bx_{-1} + cI_0 + bx_1 + aI_2 \\ &= \left(a - \frac{bd}{e}\right)I_{-2} + \frac{b}{e}F_{-1} + \left(c - 2\frac{bd}{e}\right)I_0 + \frac{b}{e}F_1 + \left(a - \frac{bd}{e}\right)I_2 \end{aligned}$$

An example 70% 5 tap filter is:

$$vFilter5Tap70pcnt = \frac{1}{32768}\begin{bmatrix} -4006 \\ 8486 \\ 23808 \\ 8486 \\ -4006 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ b \\ a \end{bmatrix}$$

The example 3-tap filter defined above is:

$$vFilter3Tap70pcnt = \frac{1}{32768}\begin{bmatrix} 5138 \\ 22492 \\ 5138 \end{bmatrix} = \begin{bmatrix} d \\ e \\ d \end{bmatrix}$$

Hence the output pixel y_0 can be calculated using the following matrix:

$$y_0 = \frac{1}{32768}\begin{bmatrix} I_{-2} \\ F_{-1} \\ I_0 \\ F_1 \\ I_2 \end{bmatrix} \bullet \begin{bmatrix} -5945 \\ 12363 \\ 19930+2 \\ 12363 \\ -5945 \end{bmatrix}$$

where the +2 is used to ensure there is unity DC gain (this is required due to rounding errors).

<u>A 7-tap source filter method</u>

Similar to the 5-tap method, the 7-tap method requires the use of a 7-tap filter, a 5-tap filter and a 3-tap filter.

Let the symmetrical 7 tap filter consist of four coefficients: *a*, *b*, *c*, *d*, and the full filter be described by [*a*, *b*, *c*, *d*, *c*, *b*, *a*] ^{T}.

Similarly, let the 5 tap filter consist of three coefficients: *e*, *f*, *g*, and the full filter be described by [*e*, *f*, *g*, *f*, *e*] ^{T}.

Finally, let the 3 tap filter consist of two coefficients *j*, *k*, and the full filter by described by [*j*, *k*, *j*] ^{T}.

Let us now consider the output pixel y_0, and its neighbours:

$$\begin{bmatrix} \vdots \\ y_0 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ aI_{-6} + bx_{-5} + cI_{-4} + dx_{-3} + cI_{-2} + bx_{-1} + aI_0 \\ ax_{-5} + bI_{-4} + cx_{-3} + dI_{-2} + cx_{-1} + bI_0 + ax_1 \\ aI_{-4} + bx_{-3} + cI_{-2} + dx_{-1} + cI_0 + bx_1 + aI_2 \\ ax_{-3} + bI_{-2} + cx_{-1} + dI_0 + cx_1 + bI_2 + ax_3 \\ aI_{-2} + bx_{-1} + cI_0 + dx_1 + cI_2 + bx_3 + aI_4 \\ ax_{-1} + bI_0 + cx_1 + dI_2 + cx_3 + bI_4 + ax_5 \\ aI_0 + bx_1 + cI_2 + dx_3 + cI_4 + bx_5 + aI_6 \\ \vdots \end{bmatrix}$$

Again, this is not practical to solve, and thus we shall solve this using a combination of a 7-tap, 5-tap and 3-tap filter:

$$\begin{bmatrix} \vdots \\ y_{-3} \\ y_{-2} \\ y_{-1} \\ y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ F_{-3} \\ y_{-2} \\ F_{-1} \\ y_0 \\ F_1 \\ y_2 \\ F_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} 0 \\ jI_{-4} + kx_{-3} + jI_{-2} \\ eI_{-4} + fx_{-3} + gI_{-2} + fx_{-1} + eI_0 \\ aI_{-4} + bx_{-3} + cI_{-2} + dx_{-1} + cI_0 + bx_1 + aI_2 \\ ax_{-3} + bI_{-2} + cx_{-1} + dI_0 + cx_1 + bI_2 + ax_3 \\ aI_{-2} + bx_{-1} + cI_0 + dx_1 + cI_2 + bx_3 + aI_4 \\ eI_0 + fx_1 + gI_2 + fx_3 + eI_4 \\ jI_2 + kx_3 + jI_4 \\ 0 \end{bmatrix}$$

This equation can be solved:

$$x_{-3} = \frac{F_{-3} - j(I_{-4} + I_{-2})}{k}$$

$$x_3 = \frac{F_3 - j(I_2 + I_4)}{k}$$

$$\begin{aligned} F_{-1} + F_1 &= \left\{ \begin{matrix} & b(x_{-3} + x_3) + (b+d)(x_{-1} + x_1) \\ + & a(I_{-4} + I_4) + (a+c)(I_{-2} + I_2) + 2cI_0 \end{matrix} \right\} \\ &= \left\{ \begin{matrix} & \frac{b}{k}(F_{-3} + F_3 - j(I_{-4} + I_{-2} + I_2 + I_4)) + (b+d)(x_{-1} + x_1) \\ + & a(I_{-4} + I_4) + (a+c)(I_{-2} + I_2) + 2cI_0 \end{matrix} \right\} \end{aligned}$$

$$\Rightarrow (x_{-1} + x_1) = \frac{1}{b+d} \left\{ \begin{matrix} F_{-1} + F_1 - \frac{b}{k}\left(F_{-3} + F_3 - j(I_{-4} + I_{-2} + I_2 + I_4)\right) \\ -a(I_{-4} + I_4) - (c+a)(I_{-2} + I_2) - 2cI_0 \end{matrix} \right\}$$

$$\begin{aligned} y_0 &= a(x_{-3} + x_3) + b(I_{-2} + I_2) + c(x_{-1} + x_1) + dI_0 \\ &= \left\{ \begin{matrix} \frac{a}{k}\left(F_{-3} + F_3 - j(I_{-4} + I_{-2} + I_2 + I_4)\right) + b(I_{-2} + I_2) + dI_0 \\ \frac{c}{b+d} \left\{ \begin{matrix} F_{-1} + F_1 - \frac{b}{k}\left(F_{-3} + F_3 - j(I_{-4} + I_{-2} + I_2 + I_4)\right) \\ -a(I_{-4} + I_4) - (c+a)(I_{-2} + I_2) - 2cI_0 \end{matrix} \right\} \end{matrix} \right\} \\ &= \frac{1}{(b+d)k} \begin{bmatrix} I_{-4} \\ F_{-3} \\ I_{-2} \\ F_{-1} \\ I_0 \\ F_1 \\ I_2 \\ F_3 \\ I_4 \end{bmatrix} \bullet \begin{bmatrix} cbj - ack - aj(b+d) \\ a(b+d) - cb \\ (bk - aj)(b+d) + cbj - ack - c^2k \\ ck \\ d(b+d)k - 2kc^2 \\ ck \\ (bk - aj)(b+d) + cbj - ack - c^2k \\ a(b+d) - cb \\ cbj - ack - aj(b+d) \end{bmatrix} \end{aligned}$$

An example 70% 7-tap filter is:

$$vFilter7Tap70pcnt = \frac{1}{32768}\begin{bmatrix}-3766\\-8514\\12315\\32698\\12315\\-8514\\-3766\end{bmatrix} = \begin{bmatrix}a\\b\\c\\d\\c\\b\\a\end{bmatrix}$$

The example 5-tap filter defined above is:

$$vFilter5Tap70pcnt = \frac{1}{32768}\begin{bmatrix}-4006\\8486\\23808\\8486\\-4006\end{bmatrix} = \begin{bmatrix}e\\f\\g\\f\\e\end{bmatrix}$$

The example 3-tap filter defined above is:

$$vFilter3Tap70pcnt = \frac{1}{32768}\begin{bmatrix}5138\\22492\\5138\end{bmatrix} = \begin{bmatrix}j\\k\\j\end{bmatrix}$$

Hence the output pixel y_0 can be calculated using the following matrix:

$$y_0 = \frac{1}{32768}\begin{bmatrix}I_{-4}\\F_{-3}\\I_{-2}\\F_{-1}\\I_0\\F_1\\I_2\\F_3\\I_4\end{bmatrix} \bullet \begin{bmatrix}1787\\830\\-12997\\16686\\20156\\16686\\-12997\\830\\1787\end{bmatrix}$$

This equation is used on each spatially interpolated pixel. The response is a bandstop filter, with the stop band between 0.66 and 0.8 $f_s/2$; however, interpreting the response is complicated as it is only used for alternate output pixels.

Figure 12a schematically illustrates the operation of the motion adaptive interpolator 110. The interpolator 110 comprises and inter-field block matcher 600, a high frequency checker 610 and a mixer 620.

The inter-field block matcher 600 uses data from the current input field and the three field stores 120 to carry out inter-field motion comparisons. This involves comparing blocks of pixels the current field (F_N in Figure 12b) with correspondingly positioned blocks in the previous field of the same type (F_{N-2}) and likewise comparing the preceding field (F_{N-1}) and the previous field of the same type (F_{N-3}). The results of these comparisons are used to detect a degree of motion in the image.

In particular, weighted sums of absolute differences (SWADs) are generated as follows.

Four block matches are performed to produce two weighted SADs. These are:

- a 5hx4v weighted block match on fields F_N and F_{N-2}.
- a 5hx3v weighted block match on fields F_{N-1} and F_{N-3}.
- a 1hx1v weighted block match on fields F_{N-1} and F_{N-3}.
- a 1hx2v weighted block match on fields F_N and F_{N-2}.

Weighted block matches sum weighted absolute differences between coincident pixels, SWAD.

$$SWAD = \sum_{dx=-2}^{2} \sum_{dy=-2,0,2} \sum_{RGB/YCbCr} w(dx,dy)\left|F_{N-1}(dx,dy) - F_{N-3}(dx,dy)\right|$$

where $F_{N-1}(dx,dy)$ is the value at the frame-relative position dx, dy to the current pixel. Example values of w can be found in the Appendix.

Summing the first two SWADs gives an area-based block match, $SWAD_{AREA}$

Summing the latter two SWADs gives a localised block match, $SWAD_{LOCAL}$

All three colour components contribute to the SWADs in the same manner. The system need only maintain a SAD of the three components for each pixel, which is then weighted and combined with the values from the other pixels in the block. This means that this aspect of the process requires only 5 line stores of about 10 bpp (bits per pixel).

The high frequency checker 610 is arranged to detect high frequencies in the input fields. The algorithm is based on the following principle.

If interleaving the two source fields produces a lot of high frequency energy, then it is appropriate to try to make sure that the inputs are reasonably static. Only static video can produce reliable high frequencies; highly aliased motion can produce high frequencies, but this is not a desirable situation for inter-field interpolation. If motion is present, then high frequencies may be produced where the fields are incorrectly interleaved.

Referring to Figure 13, the high frequency checker uses the lines above and below the currently interpolated pixel from the current field F_N and the line from the preceding field F_{N-1} that corresponds to the missing line. The HFC may be considered as a 5x3 pixel neighbourhood check.

Let $HFC_{thresh1}$ and $HFC_{thresh2}$ be two programmable constants, with the former greater than the latter.

```
Set a flag: exceededHighEnergy=false
Over each component (or a subset of them) (RGB/YPbPr) – where YPbPr indicate the colour space in a high definition system, in a similar way to YCbCr in a standard definition system :
    Set energy=0
    For the pixels having a horizontal position x=-2,-1,0,1,2 (relative to the current pixel), let the interleaved field value be $v_0$, and the current field value of the line above and below be $v_{-1}$ and $v_1$, then:
        if $v_0<\min(v_1,v_{-1})$, set diff= $\min(v_1,v_{-1})-v0$
        else if $v0>\max(v1,v-1)$, set diff= $v0 - \max(v1,v-1)$
        else set diff=0
    If ($diff>HFC_{thresh1}$), set energy = energy + $(HFC_{thresh1} - HFC_{thresh2})$*weighting[x]
    else if ($diff> HFC_{thresh2}$), set energy = energy + $(diff- HFC_{thresh2})$*weighting[x]
    If energy > $HFC_{allowance}$, set flag exceededHighEnergy=true
This ends the processing carried out over each component.
```

Subsequently, if exceededHighEnergy=true, increase $SWAD_{AREA}$ by a programmable constant value, $HFC_{penalty}$.

The increase in $SWAD_{AREA}$ will tend to act against the use of the motion adaptive pixel at that output position.

The mixer 620 operates according to the criteria $SWAD_{AREA}$ and $SWAD_{LOCAL}$ and also various thresholds $thresh_1$ ....

```
If $SWAD_{LOCAL} > thresh_1$, use only spatially interpolated field, $F_{N'}$
Else if $SWAD_{AREA} > thresh_2$, use only spatially interpolated field, FN', only
Else if $SWAD_{AREA} < thresh_3$, use only field $F_{N-1}$
Else mix field FN-1 and FN':
```

let $a=(thresh_2-SWAD_{AREA})/(thresh_2-thresh_3)$

The resulting pixel value = a F_{N-1} + (1-a) $F_{N'}$

Figures 14a and 14b schematically illustrates a variation which reduces the need for image scaling. In particular, Figure 14a schematically illustrates the operation of embodiment described above, where each row of pixels (indicated in this diagram as □○□○ ... contains real pixels and 1:2 interpolated pixels from the scaler 130. An example block match between two blocks 700, 710 and another between two blocks 720, 730 is illustrated.

In common between the two block matches in Figure 14a is the fact that the extreme edge pixels of the blocks are 1:2 interpolated pixels ○. Therefore, this method relies on the whole image having been scaled by the 1:2 scaler.

In Figure 14b, the blocks are changed in size very slightly so that the extreme edge pixels are real rather than interpolated pixels. In the upper part of Figure 14b, the same image feature angle is tested as in the upper part of Figure 14a, but the blocks 740, 750 are made slightly wider so that their edge pixels are real pixels □. Similar considerations apply to the lower part of Figure 14b, which shows a match between blocks 760, 770 which tests the same image feature angle as the match shown in the lower part of Figure 14a.

It is then the case that the comparisons between pairs of interpolated pixels are not actually needed for large blocks such as those shown in Figures 14a and 14b. The same information can be obtained from a basic comparison of the real (non-interpolate) pixels. Recall, however, that the original reason why the pixel interpolation was introduced was to allow for steep (near vertical) image feature angles to be detected. Therefore, in one arrangement, the present embodiment retains the 1:2 interpolation operation for smaller block offsets, in order to allow such steep angles to be better detected. For example, a range of block offsets of ± 1 and 0 could use 1:2 interpolated pixels, whereas block offsets of greater than this need not use the 1:2 interpolated pixels.

In a further variation, each block size is used in a match operation for only its most positive and most negative offsets. This can dramatically reduce the number of block matching operations needed, while not actually losing any information, since the previous scheme repeated (for example) the block matching tests at zero displacement (vertical image feature) for each block size.

Taken together, these measures can allow the 1:2 interpolation to be carried out "as required", for example within the block matcher, rather than as a whole field of interpolated pixels by the scaler 130. This reduces the need for a field store to hold the field of interpolated pixels. Instead, just a few interpolated pixels over a few lines need to be buffered by the block matcher.

Figures 15, 16a, 16b, 17a and 17b schematically illustrate a primary and an alternative pixel testing strategy.

A further alternative to the arrangement described above is another alteration to alter the rules applicable to block matching in order to reduce the hardware requirements of this operation.

The original equation for horizontal energy, described above, was:

$$\left(\sum_{i=2}^{blockWidth} |x_i - x_{i-1}|\right) - |x_{blockWidth} - x_1|$$

where x referred to the centre row of the block. This can be replaced by the equation:

$$\sum_{y=-1}^{0}\left(\left\{\sum_{i=2}^{blockWidth} |x(i,y) - x(i-1,y)|\right\} - |x(blockWidth,y) - x(1,y)|\right)$$

The revised test is symmetrical about the interpolated row. Figure 15 schematically illustrates a block match. Figures 16a and 16b schematically illustrate the original test for horizontal and vertical energy which would have been performed in respect of that block match. Figures 17a and 17b schematically illustrate the revised tests which show that the same two rows are used for both tests.

This means that the same logic can carry out both tests, leading to a reduction in the hardware required to implement this rule.

Finally, Figure 18 schematically illustrates the generation of a difference value. With the arrangement of Figure 18, SAD values are calculated as a rolling average, with a new value being inserted and an old one discarded. This again can cut down the processing and hardware required to implement this feature.

Although the embodiments have been described in respect of interlace to progressive scan conversion, the same techniques could be applicable to altering the size (e.g. height) of an image such as a progressive scan image. For example, the techniques could perform a 1:3 or 1:4 scaling.

It will be appreciated that the invention can be implemented in programmable or semi-programmable hardware operating under the control of appropriate software. This could be a general purpose computer or arrangements such as an ASIC (application specific integrated circuit) or an FPGA (field programmable gate array). The software could be supplied on a storage medium such as a disk or solid state memory, or via a transmission medium such as a network or internet connection, or via combinations of these.

Appendix

Some example Parameters:

	YPbPr setting	RGB setting
C1	-128	-300
C2	0	3
KFC_{Thresh}	31	41
C3	8	8
C4	3	3
C5	2	2
C6	1	1
C8	64	512
HFC weighting	[2 4 5 4 2]	[2 4 5 4 2]
$HFC_{thresh1}$	40	40
$HFC_{thresh2}$	8	8
$HFC_{allowance}$	218	128
$HFC_{penalty}$	10	25
thresh1	60	120
thresh2	20	50
thresh3	25	60

CLAIMS

1. Image processing apparatus in which output pixels of an output image are generated with respect to input pixels of an input image, a first subset of the output pixels being identical to respective input pixels and a second subset of the output pixels being derived from respective groups of one or more input pixels, the apparatus comprising:

a vertical filter for filtering the second subset of pixels, the filter not being applied to filter the first subset of output pixels, the filter having a response such that high vertical frequencies are attenuated in the output image.

2. Apparatus according to claim 1, in which:
the input image is an interlaced image; and
the output image is a progressive scan image.

3. Apparatus according to claim 2, in which the filter has a frequency response relating to a Kell factor of the input image.

4. Apparatus according to any one of the preceding claims, in which the output pixel values are generated with respect to pixels of the input image selected in accordance with an image feature direction in the input image, the apparatus comprising means for comparing blocks of pixels of the input image, the blocks being disposed with respect to a pixel position under test so that a degree of correlation between the blocks is indicative of an image feature direction applicable to that pixel position;

and in which the vertical filter is selectively operable in respect of an output pixel in dependence upon the degree of correlation between blocks used to select the image feature direction for that output pixel.

5. Apparatus according to any one of the preceding claims, comprising means for restricting the output of the vertical filter in respect of a pixel position in dependence on a range of pixel values defined by values of a set of pixels at predetermined positions with respect to that pixel position.

6. Image processing apparatus substantially as hereinbefore described with reference to the accompanying drawings.

7. Vision mixing apparatus comprising image processing apparatus according to any one of the preceding claims.

8. Video display apparatus comprising image processing apparatus according to any one of claims 1 to 6.

9. An image processing method in which output pixels of an output image are generated with respect to input pixels of an input image, a first subset of the output pixels being identical to respective input pixels and a second subset of the output pixels being derived from respective groups of one or more input pixels, the method comprising the steps of:

vertically filtering the second subset of pixels and not vertically filtering the first subset of output pixels, the vertical filter having a response such that high vertical frequencies are attenuated in the output image.

10. An image processing method substantially as hereinbefore described with reference to the accompanying drawings.

11. Computer software having program code which, when executed by a computer, is arranged to cause the computer to carry out a method according to claim 9 or claim 10.

12. A medium by which software according to claim 11 is provided.

13. A medium according to claim 12, the medium being a storage medium.

14. A medium according to claim 12, the medium being a transmission medium.

Application No:	GB0522159.3	**Examiner:**	Iwan Thomas
Claims searched:	1 and 9	**Date of search:**	15 December 2005

Patents Act 1977: Search Report under Section 17

Documents considered to be relevant:

Category	Relevant to claims	Identity of document and passage or figure of particular relevance
		None

Categories:

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

Field of Search:

Search of GB, EP, WO & US patent documents classified in the following areas of the UKC^{X} :

H4F

Worldwide search of patent documents classified in the following areas of the IPC^{07}

H04N

The following online and other databases have been used in the preparation of this search report

Online: WPI, EPODOC, PAJ