Video Processing Applications
Image and Video Processing

Dr. Anil Kokaram anil.kokaram@tcd.ie

This section covers applications of video processing as follows

- Motion Adaptive video processing for noise reduction
- Motion compensated video processing for noise reduction
- An introduction to frame rate conversion
- An introduction to MPEG2

1 Motion and Video Processing

In typical video sequences, the scene content remains principally the same from frame to frame. This implies that for tasks like noise reduction and missing data interpolation. There is much more data that can be usefully employed to reveal the underlying 'original, clean' data than with still images. For instance, consider a sequence showing a newscaster reading the news, and say that one of the frames goes missing. Because we know that the scene did not change much between frames, we can simply replace the missing frames with one of the known ones nearby in time. We could not do this if we had a photograph and 90instance. Figure 1 shows a simple example illustrating the two basic approaches to processing video data. Processing may be achieved without acknowledging motion or using motion compensation.

In the non-motion compensated processing, data is extracted from the video stream along a trajectory that is always at right angles to the plane of the image. Pixels corresponding to the same location in space are simply collected and processed as if they came from the same underlying statistical process. This is shown in the top part of the diagram.

Using motion compensated processing, pixels are extracted along motion trajectories. These trajectories must be estimated using one of the motion estimators as discussed previously in this course. This type of processing is shown in the bottom half of figure 1.

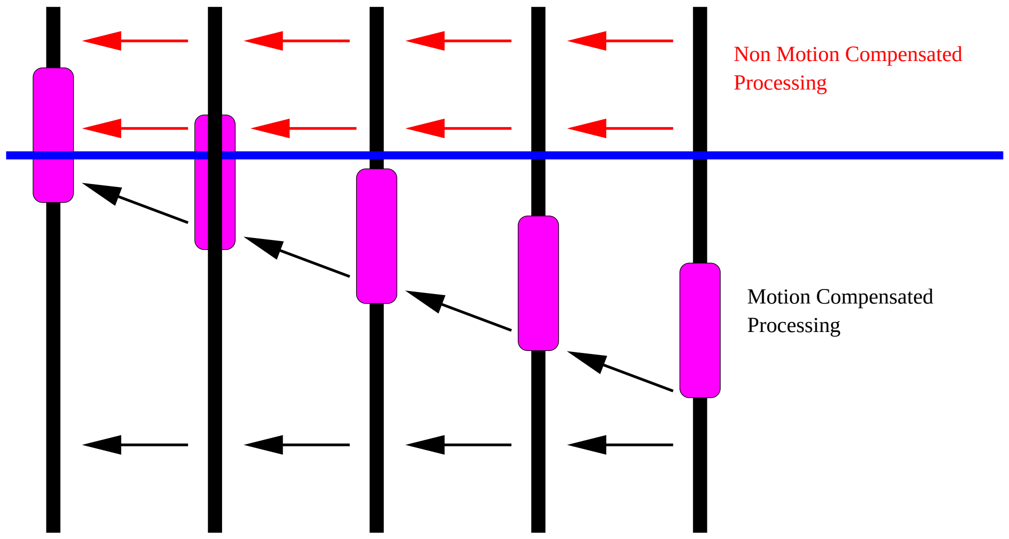

Figure 1: *Motion compensated and Non-motion compensated video processing. An object is shown moving along a 1-D image sequence with a blue line dividing the frames into a region which is processed without allowing for motion (top) and using motion compensated processing (below). The arrows show the direction of processing. Motion blurring and ghosting artifacts are heavily reduced when motion compensated processing is used.*

The figure shows movement of a single object against a background. Motion is typical of interesting video. The underlying idea of all video processing algorithms is to exploit the temporal redundancy between frames. However the motion causes complications. In the top half of figure 1, the pixels extracted do indeed have a relationship to each other initially, but as we go further back in time, eventually the extracted data crosses the path of a moving object which then destroys the statistical homogeneity. This is because the moving object is normally unrelated to the background (otherwise we would probably not be able to see it).

In the bottom half of figure 1 the extracted data is always statistically homogeneous since the extraction is following any motion that is present. However, the figure also shows a problem when occluded or uncovered regions are encountered. In those cases the trajectory can no longer be followed, or some allowance must be made for skipping frames to extract a complete data vector.

Therefore, we can process data without compensating for motion (top half of figure 1), but we ought to turn off processing when motion is detected. We can use a motion detector as discussed previously to do this. When using motion information to direct the processing, as in motion compensated processing, we do not normally have to detect motion since we are following the direction of maximum temporal redundancy. However, since motion estimators are not perfect and occlusion and uncovering do occur, more robust algorithms would double check the DFD before allowing processing to proceed along a trajectory.

The next sections show these tradeoffs at the same time as discussing various video processing applications.

2 Noise Reduction

Noise exists in any recorded video signal due to the transmission channel, or the nature of the recording medium. Old archive film and video is badly affected by noise due to the physical nature of film and magnetic media degradation. The VHS consumer tape standard also shows much noise degradation after just a few playbacks. Reducing noise in digital media also helps achieve a better compression ratio since the inherent non-homogeneity (random texture) caused by noise reduces the temporal and spatial redundancy of images.

Here we model noise degradation as follows.

$$G_n(i,j) = I_n(i,j) + \eta_n(i,j) \tag{1}$$

Where $G_n(i,j)$ is the observed signal grey scale value at position (i,j) in the nth frame, $I_n(i,j)$ is the actual non–degraded signal and $\eta_n(i,j)$ the added Gaussian noise of variance $\sigma_{\eta\eta}$.

2.1 Frame Averaging

The simplest technique for noise reduction in image sequences is frame averaging without motion compensation. It works because averaging noisy data gives an estimate of the mean of the data. If the data belongs to the same statistical process, then the mean is a good estimate of the clean data.

The technique has been used extensively and to good effect for video from electron microscope imagery. The implementation is usually a recursive estimator of the form

$$\hat{I}_n(i,j) = \frac{1}{n}[(n-1)\hat{I}_{n-1}(i,j) + G_n(i,j)] \tag{2}$$

Here, $\hat{I}_n$ represents the current output image (an estimate of the true original image, $I_n(i,j)$), $\hat{I}_{n-1}$ the previous output image and G_n the current noisy (observed) image sample that are input to the system. $\hat{I}_n$ can be recognized as the running average of all the past n frames. This is successful in microscopy applications because the image sequences observed represent stationary scenes. Exactly how it produces a running average of all past frames is shown below.

$$\begin{aligned}
\hat{I}_1(i,j) &= G_1(i,j) \text{ Initialization} \\
\hat{I}_2(i,j) &= \frac{1}{2}[\hat{I}_1(i,j) + G_2(i,j)] \\
&= \frac{1}{2}[G_1(i,j) = G_2(i,j)] \text{ Mean of first 2 frames} \\
\hat{I}_3(i,j) &= \frac{1}{3}[2 \times \hat{I}_2(i,j) + G_3(i,j)] \\
&= \frac{1}{3}[\hat{I}_1(i,j) + G_2(i,j) + G_3(i,j)] \\
&= \frac{1}{3}[G_1(i,j) + G_2(i,j) + G_3(i,j)] \text{ Mean of first 3 frames} \\
\ldots &= \ldots
\end{aligned} \tag{3}$$

In video with moving components this is not a useful noise reducer. It causes too much motion blurring eventually. Instead a motion-adaptive version can be designed.

2.2 Motion Adaptive temporal noise reduction

The recursive frame averaging idea can be generalised to a one-tap IIR (recursive) filter as follows

$$\begin{aligned}\hat{I}_n(i,j) &= \hat{I}_{n-1}(i,j) + \alpha(G_n(i,j) - \hat{I}_{n-1}(i,j)) \\ &= (1-\alpha)\hat{I}_{n-1}(i,j) + \alpha G_n(i,j)\end{aligned} \tag{4}$$

When $\alpha = \frac{1}{n}$, the filter shown in equation 2 is the result. The scalar constant α is chosen to respond to the size of the (non-motion compensated) frame difference $|\mathrm{DFD}_0| = (G_n(i,j) - \hat{I}_{n-1}(i,j))$. When this error was large, it implies motion, so α is set close to 1.0 to turn off the filtering. The is called *Motion Adaptive* filtering. Explicit motion estimation is not performed here. Instead the processing is reduced of turned off when motion is *detected*. The motion detector is built into the adaptation of α.

A small frame difference implies no motion and so α could be set to some small value to allow filtering. A typical example of a non- linear control mechanism for α is as below

$$\alpha = 1.0 - k_1 \mathrm{e}^{-(\frac{|\mathrm{DFD}_0| - k_2}{k_3})^2} \tag{5}$$

$$\textit{where}\ \mathrm{DFD}_0 = G_n(i,j) - \hat{I}_{n-1}(i,j) \tag{6}$$

k_1, k_2 and k_3 are constants chosen to select the best form of the characteristic with respect to the level of noise in the image sequence.

Although the technique performs well in stationary regions of the image, the final result is not satisfactory because of the following artifacts.

- Moving regions in the output frames would generally have a higher noise level than stationary regions.
- Stationary regions can suddenly become noisy if they began to move.
- There is a smearing effect due to filtering in a direction not close to motion trajectories.

2.3 Motion Compensated Temporal Recursive Filtering

The same sort of processing as described above can be implemented along motion trajectories to create a temporal recursive noise reducer. The filter should adapt to occlusion effects and errors in

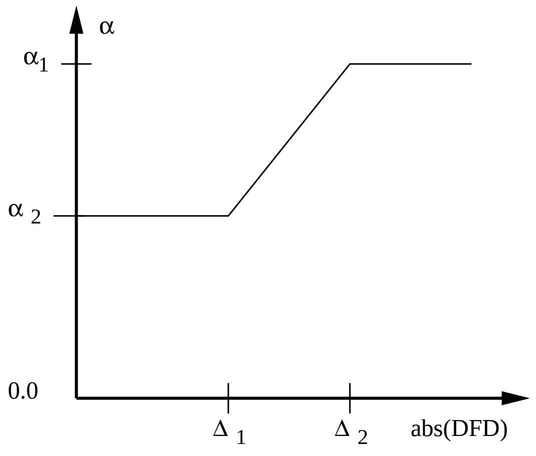

Figure 2: *The characteristic used to alter the level of noise reduction depending on the accuracy of motion estimation.*

motion estimation by varying α. A simple version uses a piecewise linear characteristic as follows

$$\alpha = \begin{cases} \alpha_1 & \text{for } |\text{DFDn}| \leq \Delta_1 \\ (\frac{\alpha_2-\alpha_1}{\Delta_2-\Delta_1})(|\text{DFDn}| - e_1) + \alpha_2 & \text{for } \Delta_1 < |\text{ DFD }| \leq \Delta_2 \\ \alpha_2 & \text{for } |\text{DFDn}| > \Delta_2 \end{cases} \quad (7)$$

Note that DFDn is the *displaced* frame difference, i.e. the frame difference along an estimated motion trajectory between the current frame and the previous. This is defined as follows

$$\text{DFDn} = G_n(i,j) - \hat{I}_{n-1}(i + dx(i,j), j + dy(i,j)) \quad (8)$$

where the motion vector mapping the pixel at (i,j) from frame n into frame $n-1$ is given by $[dx(i,j),\ dy(i,j)]$.

The characteristic allows for three regions of operation as shown in figure 2. When motion is tracked effectively, filtering is performed with $\alpha = \alpha_1 < 1.0$. When motion cannot be tracked the filtering is turned off by setting $\alpha = \alpha_2 = 1.0$. The mid–range linear ramp represents a smooth transition in filtering between the tracked and untracked regions.

This process of adapting the filter in addition to motion compensation enables better image detail preservation. The following observations can be made.

1. Motion compensated filtering of image sequences enables better noise suppression and a better output image quality than non–motion compensated techniques.
2. It is important to adapt the extent of filtering to the accuracy of motion estimation. This would enable a better quality output image by reducing smearing artifacts when occlusion and uncovering occurs.

Therefore, although motion compensated filtering is more effective for image sequences, a good algorithm must also be robust to erroneous motion estimation.

2.4 The Dirty Window Effect

All purely temporal noise reducers treat each pixel in the image independently. This means that pixel $G_n(i,j)$ is processed completely independently of $G_n(i+1,j+1)$ for instance. As time progresses, the noise variation in *time* is reduced but in *space* it is not. That means that eventually, it appears as if the objects are moving behind a 'fixed' noise field or a 'dirty window'. This is because the correlation between local image data is generally high, and the noise field variation is easily perceived against this. This effect can be reduced by processing the data in space as well as time.

Such 3D or spatio-temporal processes can be designed by extracting volumes of motion compensated data on a block basis for instance. 3D Wiener and Wavelet filters can be created in this way. This is left for the next year in this course.

2.5 Perception of motion artifacts

When there is rapid motion, this motion cannot typically be estimated accurately. However this may not affect the subjective quality of the output despite the reduced effectiveness of the filtering operation. This is because the human perception of a fast moving object is much less than a slow moving one. Therefore it is possible to strike a useful compromise between output image quality and the speed of objects in a scene.

Nevertheless there are a mid-band of velocities for which the sensitivity to artifacts is greater than for stationary objects or very fast moving ones. Furthermore, when the eye tracks an object, it is transformed into a stationary one. Thus it is very difficult indeed to estimate quantitatively the human perception of defects in sequences and robust algorithms must rely on a combination of good motion estimation techniques and graceful degradation when motion estimation fails.

3 Video up/down-conversion

There are many digital video standards in use at the moment. PAL and NTSC we have covered and they operate at different line and field rates. Film operates at 24 fps (frames per second). The conversion of data between these formats is an important topic both commercially and from the point of view of design. Conversion from PAL to NTSC requires conversion from 25 fps to 30 fps, which involves the interpolation of new frames of data. NTSC to PAL conversion involves dropping the frame rate from 30 fps to 25 fps but interpolating more lines into each frame since PAL is 625 line and NTSC is 525. Conversion of film to NTSC again involves insertion of new frames of data to allow the creation of 30 fps material from 24 fps material. Frame rate conversion refers to progressive scan data sources, field rate conversion refers to interlaced data sources.

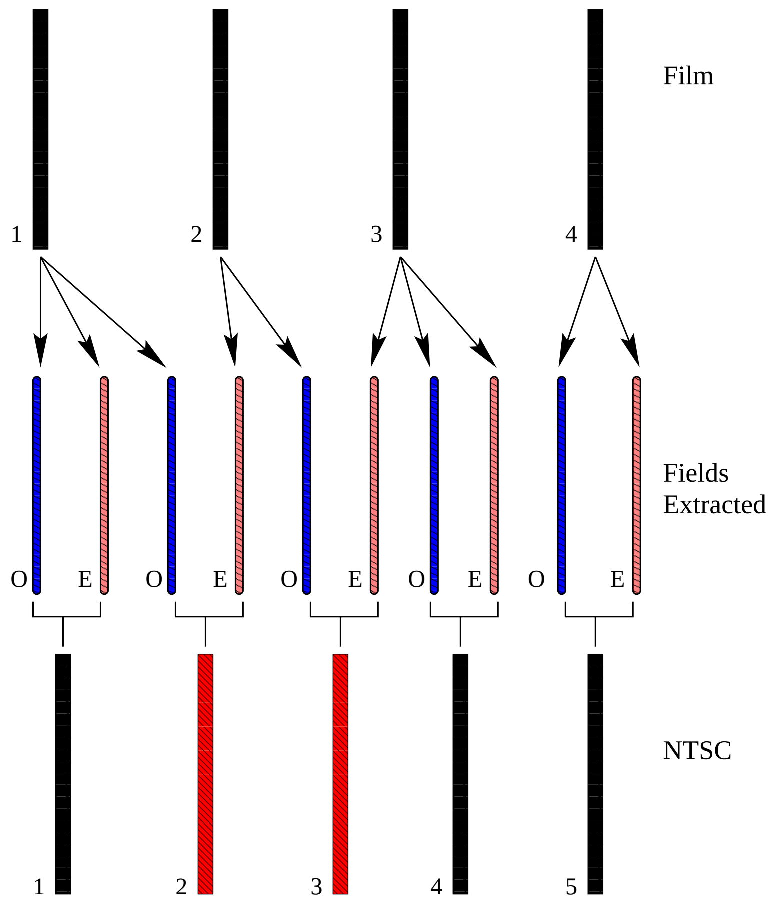

Figure 3: *3:2 Pulldown for converting from Film rate to NTSC. TV frames built from fields taken from different Film Frames are indicated in red.*

3.1 Non-Motion Compensated Frame/Field Rate conversion

The simplest way to upsample frames is to repeat them. The well established '3 to 2 pulldown' method of converting film material to NTSC is an example. This is shown in figure 3. Each odd frame of film material is repeated three times and each even frame repeated twice to yield an upsampled frame based sequence at 60 frames per second. The odd and even fields of NTSC are then taken from consecutive frames of this sequence giving a 60 field per second NTSC sequence. In every 5 frames of NTSC therefore, three of the five are constructed using fields from the same film frame, but the remaining two are constructed using fields from different frames (shown in red in figure 3). The 3:2 pulldown method can also be described by upsampling the original film sequence by a factor of 5 using a zero-order hold, and then downsampling the resulting sequence by two.

This method for conversion is good enough for home TV viewing, but very poor for high resolution HDTV standards. There is substantial motion jerkiness caused by the zero-order hold effect of the process.

For converting from Film rate to PAL, typically no frame rate conversion is done since 24 fps

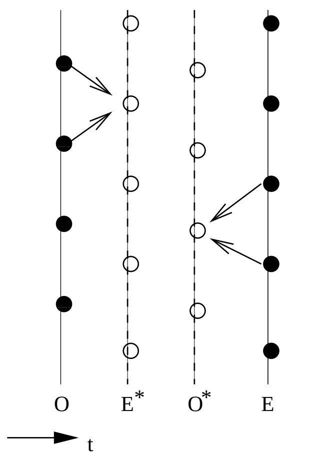

Figure 4: *Two point line averaging for field rate upconversion.*

and 25 fps is considered to be sufficiently close. Thus the 24 fps film rate material is simply played back at 25 fps. There are indeed sound and video synchronisation problems, but these are slight.

3.1.1 Scan Rate Doubling and deinterlacing

Given an interlaced TV sequence, it is often necessary to deinterlace the sequence or increase the frame rate by interpolating lines in the separate even and odd fields. A good example of this is conversion between NTSC and PAL. The conversion from NTSC (30 fps at 262.5 lines per field) to PAL (25 fps at 312.5 lines/field) can be achieved by dropping a complete frame every five frames and spatially interpolating for the missing lines. Similarly, PAL⇒NTSC requires dropping some lines per frame and creating a complete frame every five frames.

The simplest form of line interpolation is created by repeating lines, using zero order hold. Thus a new frame can be created from one field by repeating the lines of that field. This zero-order hold de-interlacing causes jagged edges in stationary regions, but to aliasing.

Another simple mechanism for line interpolation (and for field rate up conversion) is shown in figure 4. In this process the lines of the new even field are created by averaging two pixels from the lines of the current odd field as below

$$\hat{I}_{n/Even}(i,j) = \frac{1}{2}[I_{n/Odd}(i,j-1) + I_{n/Odd}(i,j+1)] \tag{9}$$

where $\hat{I}_{n/Even}(i,j)$ is the estimated intensity of the Even field in frame n at site (i,j), and $I_{n/Odd}(i,j-1)$ is the observed intensity of the odd field in frame n in line $(j-1)$.

Similarly, the new Odd field can be created by averaging pixels in the observed even field.

$$\hat{I}_{n+1/Odd}(i,j) = \frac{1}{2}[I_{n/Even}(i,j-1) + I_{n/Even}(i,j+1)] \tag{10}$$

From these new fields two new frames can be created with the frame pairs $I_{n/Odd}$, $\hat{I}_{n/Even}$ and $I_{n/Even}$, $\hat{I}_{n/Odd}$.

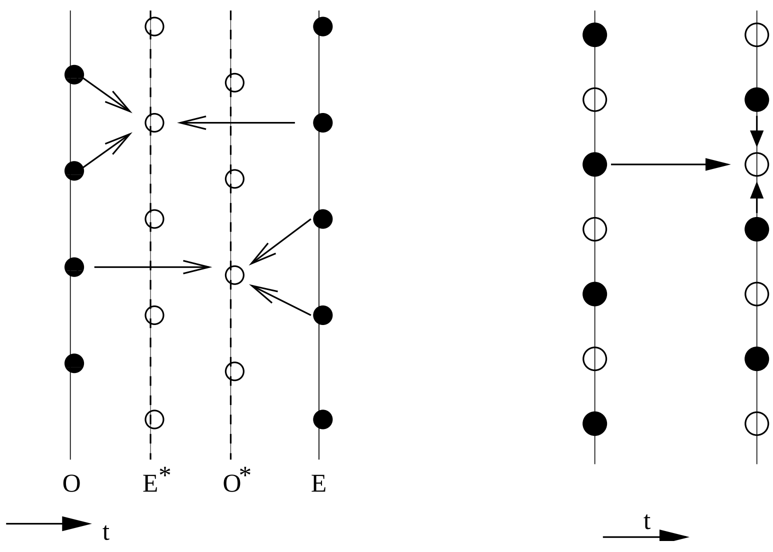

Figure 5: *Left: Three point filtering for field rate doubling. Right: Two field filtering for de-interlacing.*

This process does not account for motion. It turns out that in moving areas, the result is acceptable, but in stationary regions the blurring caused by the line averaging is apparent.

3.2 Motion Adaptive Conversion

At this point it should be clear that the two main processes required in standards conversion are de-interlacing (creating frames from fields) and frame interpolation (inserting new frames between existing ones). Note that the process of field rate conversion is not the same as de-interlacing. De-interlacing requires the generation of a new even field (for instance) that *coincides* with the time instant of the odd field. In the case of Field rate up-conversion by a factor of 2, this requires the creation of a new pair of even and odd fields that occur at 1/3 and 2/3 the time interval between the original odd and even fields.

Figure 5 (left) shows the structure of a simple three point filter that can be used for field rate doubling. The filter output could be taken as the average of the pixels shown and this would work reasonably well in stationary areas. However to account for motion, a median filter can be used as follows

$$\hat{I}_{E^*}(i,j) = \text{Median}[I_O(i,j-1),\ I_O(i,j+1),\ I_E(i,j)]$$

Figure 5 (right) shows the structure of a three point filter for de-interlacing. The underlying idea is that in stationary areas, a de-interlaced Odd frame can be created simply by copying the Odd field in the next frame into the even lines of the current frame and vice-versa for the de-interlaced Even frame. In other words, when there is no motion, a the two fields come from the same imaged picture and can be used to create a complete frame. However, when there is motion, the data in

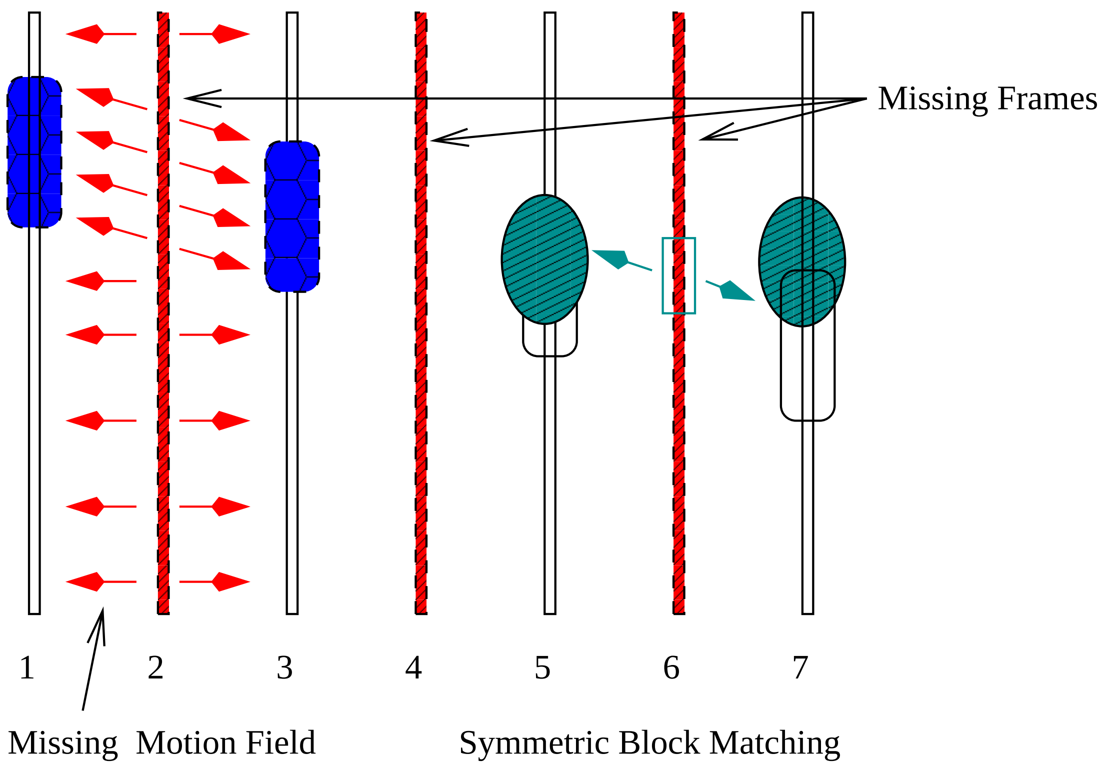

Figure 6: *Motion compensated frame rate conversion and Symmetric Block Matching.*

the next field would be uncorrelated with the missing lines in the current field. Hence some motion adaptation must be used. A simple form of linear adaptation is as follows

$$\hat{I}_O(i,j) = \alpha I_E(i,j-1) + (1-\alpha) I_E(i,j+1) + \beta I_E(i,j) \tag{11}$$

where α, β depend on the values of a motion detection function that would typically depend on DFD. The process of de-interlacing therefore switches between intraframe interpolation when motion is detected (causing $\beta = 0$) and merging when motion is not detected (causing $\alpha = 1$, $\beta = 1$ for instance).

3.3 Motion Compensated Conversion

Figure 6 illustrates that the fundamental problem in video frame rate conversion is the reconstruction of missing data at regular intervals in time. The figure shows a sequence of 4 frames numbered 1,3,5,7 and the process of doubling this frame rate to create frames between these numbered at 2,4,6. The underlying problem is to estimate the missing motion field as is shown for the first frame to be interpolated, frame 2. Once that missing motion field can be estimated for the missing data then one can average motion compensated pixels in frames 1 and 3 to create the upsampled frame 2.

If we assume that there is no acceleration in the sequence, then $\mathbf{d}_{n,n-1}(\mathbf{x}) = -\mathbf{d}_{n,n+1}$ and we can write our image sequence model with reference to the missing frame n as

$$\begin{aligned} I_n(\mathbf{x}) &= I_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}(\mathbf{x})) \\ &= I_{n+1}(\mathbf{x} - \mathbf{d}_{n,n-1}(\mathbf{x})) \end{aligned} \tag{12}$$

According to these assumptions therefore

$$I_n(\mathbf{x}) - I_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}(\mathbf{x})) = I_n(\mathbf{x}) - I_{n+1}(\mathbf{x} - \mathbf{d}_{n,n-1}(\mathbf{x})) \tag{13}$$

Therefore, to estimate the missing motion field, one can use a method called "Symmetric Motion Estimation". In the case of symmetric Block Matching, the frames $n-1$ and $n+1$ are searched with respect to n. Using a DFD defined as DFD_s

$$\text{DFD}_s = \sum_{\mathbf{x} \in \mathcal{B}} |I_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}(\mathbf{x})) - I_{n+1}(\mathbf{x} - \mathbf{d}_{n,n-1}(\mathbf{x}))| \tag{14}$$

This DFD is therefore minimised over the search area as indicated in figure 6, creating a motion field which is symmetric and follows the assumption of no acceleration. The motion field is also generated at the site of missing pixel data as required.

The interpolated frame can then be created by averaging motion compensated pixels, for instance as follows.

$$\hat{I}_n(\mathbf{x}) = \frac{1}{2}[I_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}(\mathbf{x})) + I_{n+1}(\mathbf{x} - \mathbf{d}_{n,n-1}(\mathbf{x}))] \tag{15}$$

This process can work well, except for the problems of occlusion and uncovering. Also there is the additional problem of acceleration and 'difficult' motion like moving cloth and other heavily deformable shapes. Taking those problems into account is a matter of current research.

4 Video Compression

Video compression is concerned with coding image sequences at low bit rates. In an image sequence, there are typically high correlations between consecutive frames of the sequence, in addition to the spatial correlations which exist naturally within each frame.

Video coders aim to take maximum advantage of *interframe* temporal correlations (between frames) as well as *intraframe* spatial correlations (within frames).

4.1 Motion-Compensated Predictive Coding

Motion-compensated predictive coding (MCPC) is the technique that has been found to be most successful for exploiting interframe correlations.

Fig 7 shows the basic block diagram of a MCPC video encoder.

The transform, quantise, and entropy encode functions are basically the same as those employed for still image coding. The first frame in a sequence is coded in the normal way for a still image by switching the prediction frame to zero.

For subsequent frames, the input to the transform stage is the difference between the *input* frame and the *prediction* frame, based on the previous decoded frame. This difference frame is usually known as the *prediction error* frame.

The purpose of employing prediction is to reduce the energy of the prediction error frames so that they have lower entropy after transformation and can therefore be coded with a lower bit rate.

If there is motion in the sequence, the prediction error energy may be significantly reduced by *motion compensation*. This allows regions in the prediction frame to be generated from *shifted* regions from the previous decoded frame. Each shift is defined by a *motion vector* which is transmitted to the decoder in addition to the coded transform coefficients. The motion vectors are usually entropy coded to minimise the extra bit rate needed to do this.

The multiplexer combines the various types of coded information into a single serial bit stream, and the buffer smooths out the fluctuations in bit rate caused by varying motion within the sequence and by scene changes. The controller adjusts coding parameters (such as the quantiser step size) in order to maintain the buffer at approximately half-full, and hence it keeps the mean bit rate of the encoder equal to that of the channel.

Decoded frames are produced in the encoder, which are identical to those generated in the decoder. The decoder comprises a buffer, de-multiplexer, and entropy decoder to invert the operations of the equivalent encoder blocks, and then the decoded frames are produced by the part of the encoder loop comprising the inverse quantiser, inverse transform, adder, frame store, motion compensator and switch.

H.261 is a CCITT standard for video encoding for video-phone and video conferencing applications. Video is much more important in a multi-speaker conferencing environment than in simple one-to-one conversations. H.261 employs coders of the form shown in figure 7 to achieve reasonable quality head-and-shoulder images at rates down to 64 kb/s (one ISDN channel). A demonstration of H.261 coding at 64 and 32 kb/s will be shown.

A development of this, H.263, allows the bit rate to be reduced down to about 20 kb/s, without too much loss of quality, for modems and mobile channels. This uses some of the more advanced motion methods from MPEG (see later).

4.2 Comments on Motion Estimation

Block Matching (BM) is the most common method of motion estimation in compression. Typically each macroblock (16×16 pels) in the new frame is compared with shifted regions of the same size from the previous decoded frame, and the shift which results in the minimum error is selected as the

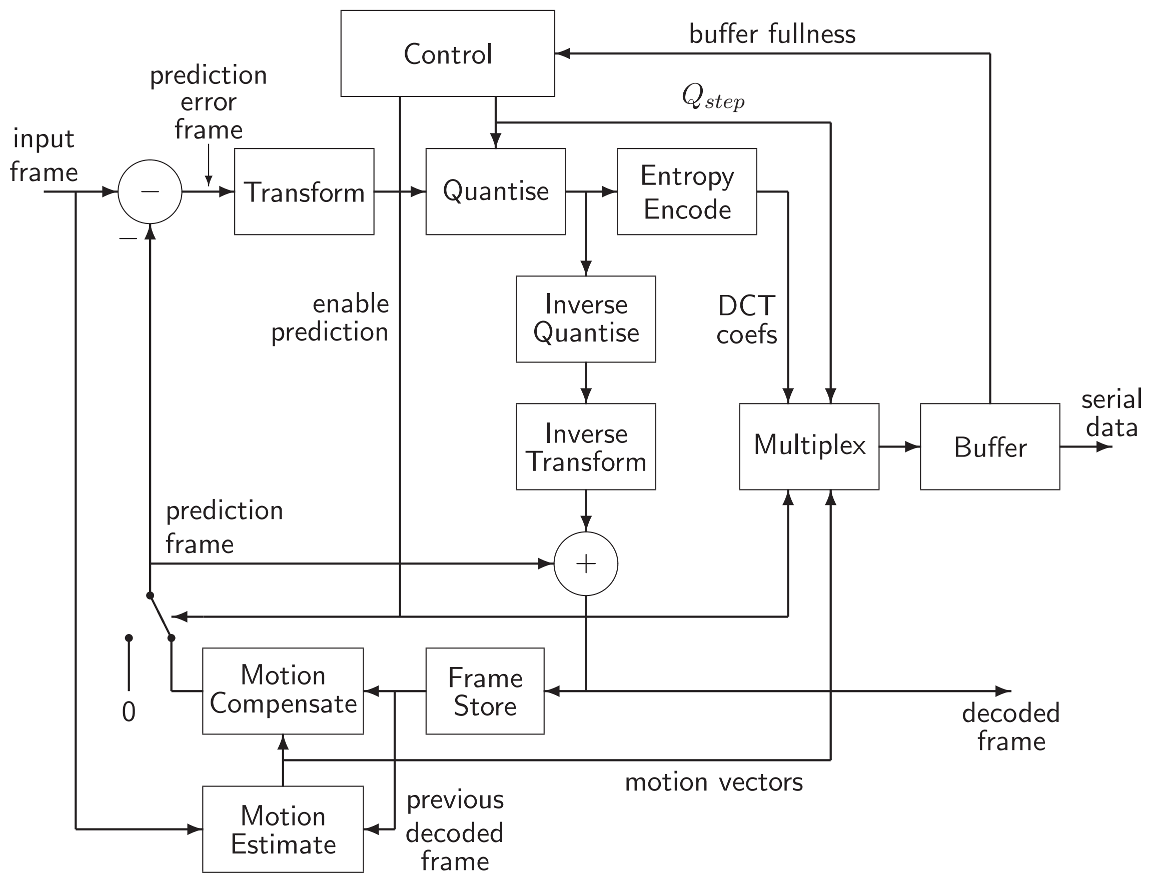

Figure 7: Motion compensated prediction coding (MCPC) video encoder.

best motion vector for that macroblock. The motion compensated prediction frame is then formed from all of the shifted regions from the previous decoded frame.

BM can be very computationally demanding if all shifts of each macroblock are analysed. For example, to analyse shifts of up to ± 15 pels in the horizontal *and* vertical directions requires $31 \times 31 = 961$ shifts, each of which involves $16 \times 16 = 256$ pel difference computations for a given macroblock. This is known as *exhaustive search BM*.

Significant savings can be made with *heirarchical BM*, in which an approximate motion estimate is obtained from exhaustive search using a lowpass subsampled pair of images, and then the estimate is refined by a small local search using the full resolution images. Subsampling 2:1 in each direction reduces the number of macroblock pels *and* the number of shifts both by 4:1, producing a computational saving of 16:1!

There are many other approaches to motion estimation, some using the frequency or wavelet domains, and designers have considerable scope to invent new methods since this process does not need to be specified in coding standards. The standards need only specify how the motion vectors should be interpreted by the decoder (a much simpler process). Unfortunately, we do not have time to discuss these other approaches here.

4.3 The MPEG Standard

As a sequel to the JPEG standards committee, the Moving Picture Experts Group (MPEG) was set up in the mid 1980s to agree standards for video sequence compression.

Their first standard was MPEG-I, designed for CD-ROM applications at 1.5Mb/s, and their more recent standard, MPEG-II, is aimed at broadcast quality TV signals at 4 to 10 Mb/s and is also suitable for high-definition TV (HDTV) at 20 Mb/s. We shall not go into the detailed differences between these standards, but simply describe some of their important features.

MPEG coders all use the MCPC structure of figure 7, and employ the 8×8 DCT as the basic transform process. So in many respects they are similar to H.261 coders, except that they operate with higher resolution frames and higher bit rates.

The main difference from H.261 is the concept of a Group of Pictures (GOP) Layer in the coding heirarchy. However we describe the other layers first:

- The Sequence Layer contains a complete image sequence, possibly hundreds or thousands of frames.
- The Picture Layer contains the code for a single frame, which may either be coded in absolute form or coded as the difference from a predicted frame.
- The Slice Layer contains one row of macroblocks (16×16 pels) from a frame. (48 macroblocks give a row 768 pels wide.)

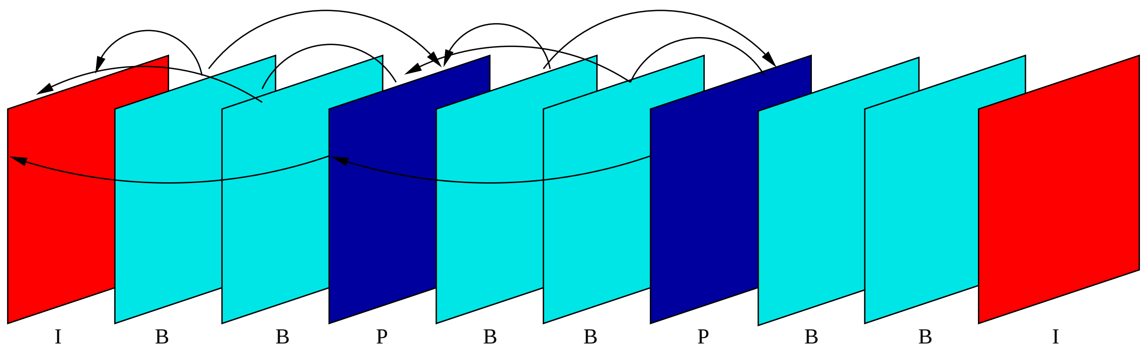

Figure 8: *A typical Group of Pictures (GOP). I-frames are Intra-coded. B frames are bi-directionally predicted from the previous I (or P) and next P (or I) frames.*

- The Macroblock Layer contains a single macroblock — usually 4 blocks of luminance, 2 blocks of chrominance and a motion vector.
- The Block Layer contains the DCT coefficients for a single 8×8 block of pels, coded almost as in JPEG using zig-zag scanning and run-amplitude Huffman codes.

The GOP Layer contains a small number of frames (typically 12) coded so that they can be decoded completely as a unit, without reference to frames outside of the group. There are three types of frame:

I — *Intra* coded frames, which are coded as single frames as in JPEG, without reference to any other frames.

P — *Predictive* coded frames, which are coded as the difference from a motion compensated prediction frame, generated from an earlier I or P frame in the GOP.

B — *Bi-directional* coded frames, which are coded as the difference from a bi-directionally interpolated frame, generated from earlier and later I or P frames in the sequence (with motion compensation).

The main purpose of the GOP is to allow editing and splicing of video material from different sources and to allow rapid forward or reverse searching through sequences. A GOP usually represents about half a second of the image sequence.

Figure 8 shows a typical GOP and how the coded frames depend on each other. The first frame of the GOP is always an I frame, which may be decoded without needing data from any other frame. At regular intervals through the GOP, there are P frames, which are coded relative to a prediction from the I frame or previous P frame in the GOP. Between each pair of I / P frames are one or more B frames.

The I frame in each GOP requires the most bits per frame and provides the initial reference for all other frames in the GOP. Each P frame typically requires about one third of the bits of an I frame, and there may be 3 of these per GOP. Each B frame requires about half the bits of a P frame and there may be 8 of these per GOP. Hence the coded bits are split about evenly between the three frame types.

B frames require fewer bits than P frames mainly because bi-directional prediction allows uncovered background areas to be predicted from a subsequent frame. The motion-compensated prediction in a B frame may be forward, backward, or a combination of the two (selected in the macroblock layer). Since no other frames are predicted from them, B frames may be coarsely quantised in areas of high motion and comprise mainly motion prediction information elsewhere.

In order to keep all frames in the coded bit stream causal, B frames are always transmitted *after* the I / P frames to which they refer.

One of the main ways that the H.263 (enhanced H.261) standard is able to code at very low bit rates is the incorporation of the B frame concept.

Considerable research work at present is being directed towards more sophisticated motion models, which are based more on the outlines of objects rather than on simple blocks. These form the basis of the new low bit-rate standard, MPEG-4 (there is no MPEG-III).

5 Summary

This section has covered three useful applications of motion estimation. The most industrially relevant application is video compression.

The book *Digital Video Processing*, Murat Tekalp, Prentice Hall, ISBN 0-13-190075-7 has good coverage of video filtering, standards conversion techniques and reasonable overview of video compression. Lim covers a useful technique for deinterlacing in pages 507–509.

References