Motion Estimation
Image and Video Processing

Dr. Anil Kokaram anil.kokaram@tcd.ie

This section covers motion estimation and motion compensation. The following ideas are introduced.

- Why motion estimation?
- Motion Detection with the DFD
- Motion Estimation: solving the motion equation
- Block Matching Motion Estimation
- Gradient Based Motion Estimation
- Optic Flow

1 Why Motion Estimation?

Image data in an image sequence remains mostly the same between frames *along motion trajectories*. This is the same as saying that the scene content does not change much from frame to frame. To exploit the image data redundancy in image sequences there is a need to estimate motion in the image sequence so one can process along motion trajectories After motion estimation, modules such as noise reduction and compression can be executed. Note that this is a practical approach to what is a JOINT problem of motion estimation AND noise reduction or motion estimation AND compression.

2 Motion Detection

Motion estimation is typically the most compute intensive part of video processing. It is therefore sensible to perform motion estimation *only* where motion is present. Recall the simple translational

motion model (will drop arguments for displacement function $\mathbf{d}$ to make things easier to read.)

$$I_n(\mathbf{x}) = I_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}) + e(\mathbf{x})$$
$$I_n(i,j) = I_{n-1}(i + dx, j + dy) + e(i,j) \quad (1)$$

The lower equation reminds you explicitly that $\mathbf{x} = [i, j]$ and $\mathbf{d} = [dx, dy]$. Without motion we have

$$I_n(\mathbf{x}) = I_{n-1}(\mathbf{x}) + e(\mathbf{x}) \quad (2)$$

It is expected that $|e(\cdot)|$ will be small where $\mathbf{d} = 0$ but BIG where $\mathbf{d} \neq 0$. Therefore we can detect motion by measuring the Pixel Difference (PD) $= |I_n(\mathbf{x}) - I_{n-1}(\mathbf{x})|$. The PD is then thresholded to detect motion.

Assuming that motion is detected at a site when $m(\mathbf{x}) = 1$, the motion detector can be written as

$$m(\mathbf{x}) = \begin{cases} 1 & \text{if } |I_n(\mathbf{x}) - I_{n-1}(\mathbf{x})| > E_t \\ 0 & \text{Otherwise} \end{cases} \quad (3)$$

where E_t is the threshold intensity difference used to flag motion. 5 is typical.

2.1 Problems with Pixel Difference

Unfortunately, the PD can be 'noisy' because of noise in the sequence. This can cause 'ragged' edges in detected motion regions, and holes in otherwise completely moving objects. This can be overcome by smoothing the PD (using spatial Gaussian filter for example) before thresholding. This causes 'leakage' of detected motion into non-moving areas. Alternatively, one can use a block based strategy instead and average the PD over blocks to make a block-based motion detection decision. Note that we only want to do this to limit the image area over which we will be trying to estimate motion. Therefore it is sufficient for false alarms to be kept reasonably low.

This mechanism for motion detection will generally fail in regions that show no textural variation or have low image gradient.

3 Motion Estimation

Given the image sequence model below, the motion estimation problem is to estimate $\mathbf{d}$ at all pixel sites that are at moving object locations. The image sequence model is restated below for convenience.

$$I_n(\mathbf{x}) = I_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}) + e(\mathbf{x}) \quad (4)$$

Solution of this expression for motion is a massive optimisation problem, requiring the choice of values for $\mathbf{d}$ at all sites to minimise some function of the error $e(\mathbf{x})$. It is typical to choose the value of $\mathbf{d}$ such that it results in the minimum DISPLACED FRAME DIFFERENCE (DFD) defined as

$$DFD(\mathbf{x}) = I_n(\mathbf{x}) - I_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}) \quad (5)$$

The problem is complicated by the fact that the motion variable **d** is *an argument to* the image function $I_{n-1}(\cdot)$. The image function generally cannot be modelled explicitly as a function of position and this makes access to the motion information difficult.

Many motion estimators can be derived from the idea of minimising some function of the DFD. However the Bayesian framework is the only one from which *all* motion estimators can be derived. You have not examined Bayesian analysis yet in this course and this will not be discussed here.

Note that the motion equation above cannot be solved at a single pixel site. There are 2 variables: x and y components of motion, so at least 2 equations are required i.e. at least 2 pixel sites.

3.1 Motion Estimation: Types

Different types of motion estimator can be identified by the way in which they solve the image sequence model 4 for the motion variable **d**. These types are summarised as follows.

1. Simplest is Exhaustive search. That is: try every single possible value of **d** until DFD is minimised. Computationally intensive, but suitable for VLSI. Many other problems as well, but will consider that later. Block Matching is an example of this type of motion estimator.
2. Make the equation 4 an explicit function of **d** by using Taylor series to linearise the motion equation. Gradient based motion estimators: Optic Flow, Wiener based, Pel-recursive all fall into this category.
3. Pel-recursive techniques operate by making small adjustments to the motion vector estimates over a series of iterations.
4. Optic flow motion estimators include additional constraints (on the motion field itself) to estimate motion.

3.2 A note on Tracking

Beware that this course considers motion estimation for video processing, and not for tracking. Object tracking is the act of extracting a motion trajectory through an image sequence that is identified with the same object throughout the sequence. For instance, tracking is the process of extracting the position of the same robot fingertip through a sequence of images, or tracking the motion of human hand gestures for human computer interaction. Efficient tracking requires that the motion estimator incorporates knowledge about the expected smoothness of motion along a motion trajectory between frames. The motion estimators discussed in this course can be used for tracking, but some post-processing of the motion field would be necessary to ensure that the required object point was being tracked from one frame to the next.

4 Block Based Motion Estimation

4.1 Block Matching Motion Estimation

The most popular and to some extent the most robust technique to date for motion estimation is Block Matching (BM) [5, 9, 3].

Two basic assumptions are made in this technique.

1. Constant translational motion over small blocks (say 8×8 or 16×16) in the image. This is the same as saying that there is a minimum object size that is larger than the chosen block size.
2. There is a maximum (pre-determined) range for the horizontal and vertical components of the motion vector at each pixel site. This is the same as assuming a maximum velocity for the objects in the sequence. This restricts the range of vectors to be considered and thus reduces the cost of the algorithm.

The image in frame n, is divided into blocks usually of the same size , $N \times N$. Each block is considered in turn and a motion vector is assigned to each. The motion vector is chosen by matching the block in frame n with a set of blocks of the same size at locations defined by some search pattern in the previous frame.

Given a possible vector $\mathbf{v} = [dx,\ dy]$, we can define the DFD between a pixel in the current frame and its *motion compensated* pixel in the previous frame as

$$DFD(\mathbf{x}, \mathbf{v}) = I_n(\mathbf{x}) - I_{n-1}(\mathbf{x} + \mathbf{v}) \tag{6}$$

Define the Mean Absolute Error of the DFD between the block in the current frame and that in the previous frame as

$$MAE(\mathbf{x}, \mathbf{v}) = \frac{1}{N^2} \sum_{\mathbf{x} \in \text{Block}} |DFD(\mathbf{x}, \mathbf{v})| \tag{7}$$

We can use Mean Squared Error (MSE) as well, but MAE is more robust to noise.

The block matching algorithm then proceeds as follows at each image block.

1. Pre-determine a set of candidate vectors $\mathbf{v}$ to be tested as the motion vector for the current block
2. For each $\mathbf{v}$ calculate the MAE
3. Choose the motion vector for the block as that $\mathbf{v}$ which yields the minimum MAE.

The set of vectors $\mathbf{v}$ in effect yield a set of candidate *motion compensated* blocks in the previous frame $n-1$ for evaluation. The separation of the candidate blocks in the search space determines the

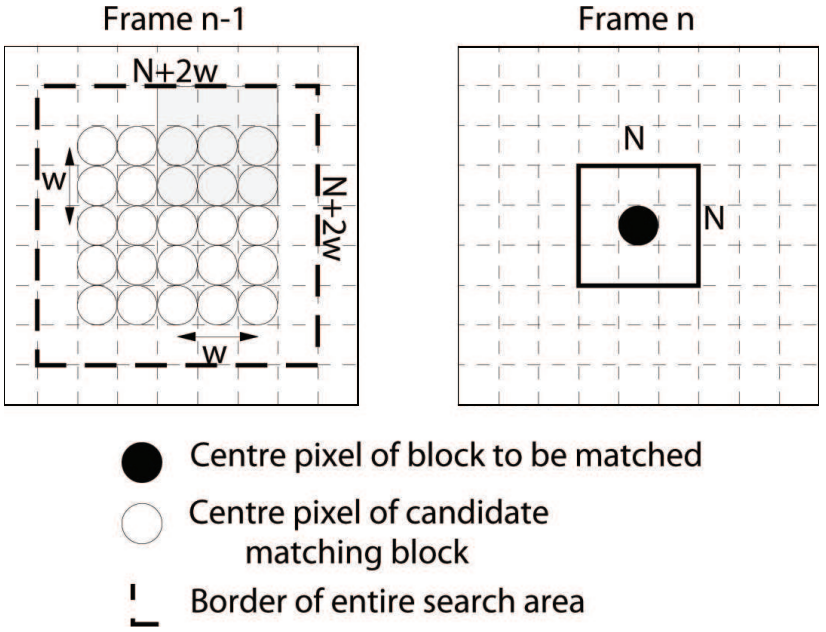

Figure 1: Motion estimation via Block Matching. The positions indicated by a ◯ in frame $n-1$ are searched for a match with the $N \times N$ block in frame n. One block to be examined is located at displacement $[1 \ -2]$, and is shaded.

smallest vector that can be estimated. For integer accurate motion estimation the position of each block coincides with the image grid. For fractional accuracy [6], blocks need to be extracted between locations on the image grid. This requires some interpolation. In most cases bilinear interpolation is sufficient.

Figure 1 shows the search space used in a **full motion search** technique. The current block is compared to every block of the same size in an area of size $(2w+N) \times (2w+N)$. The search[1] space is chosen by deciding on the maximum displacement allowed: in Figure 1 the maximum displacement estimated is $\pm w$ for both horizontal and vertical components.

The technique arises from a direct solution of equation 4. The BM solution can be seen to minimize the Mean Absolute DFD (or Mean Square DFD) with respect to $\mathbf{v}$, over the $N \times N$ block. The chosen displacement, $\mathbf{d}$ satisfies the model equation 4 in some 'average' sense.

4.1.1 Computation

The Full Motion Search is computationally demanding. Given a maximum expected displacement of $\pm w$ pels, there are $(2w+1)^2$ searched blocks (assuming integer displacements only). Each block considered requires on the order of N^2 operations to calculate the MAE. This implies $N^2(2w+1)^2$ operations per block for an integer accurate motion estimate. Several reduced search techniques have been introduced which lessen this burden. They attempt to reduce the operations required

[1] There are $(2w+1)^2$ searched locations.

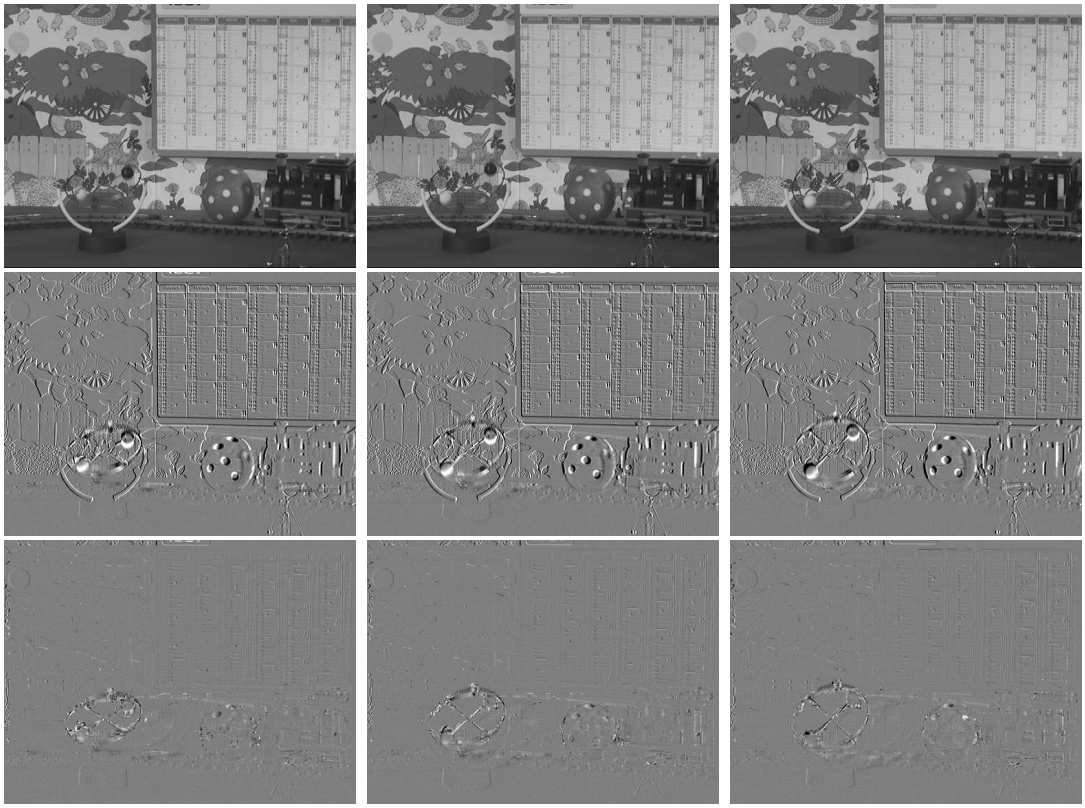

Figure 2: Top Row: Frames 2,3,4 of the Mobile and Calendar Sequence, Middle Row : Frame difference without motion compensation, Last Row : DFD after Integer BM ±4 pixel search.

either by reducing the locations searched [5] or by reducing the number of pixels sampled in each block [9]. However, reduced searches may find local minima in the DFD function and yield spurious matches.

4.1.2 Motion Compensation

Figure 2 shows the measurement of Displaced Frame Difference before and after motion estimation is applied. The Error shown is the error at each pixel site between the value at the site in the current frame and the value at the site in the previous frame to which the current pixel is mapped. That is it shows $e(h,k) = I_n(h,k) - I_{n-1}(h + v^x_{n,n-1}(h,k), k + v^y_{n,n-1}(h,k))$ at each site (h,k). Here the motion vector mapping position (h,k) in frame n into the previous frame $n-1$ is $(v^x_{n,n-1}(h,k), v^y_{n,n-1}(h,k))$, where the horizontal component of motion is $v^x_{n,n-1}(h,k)$ and $v^y_{n,n-1}(h,k)$ is the vertical component. The MAE over the whole frame between frames n and $n-1$ is then

$$E_{n,n-1} = \frac{1}{HK} \sum_{h=0}^{h=H-1} \sum_{k=0}^{k=K-1} |e(h,k)| \tag{8}$$

Figure 3 shows the MAE for the first 20 frames in the sequence. It shows how much improvement

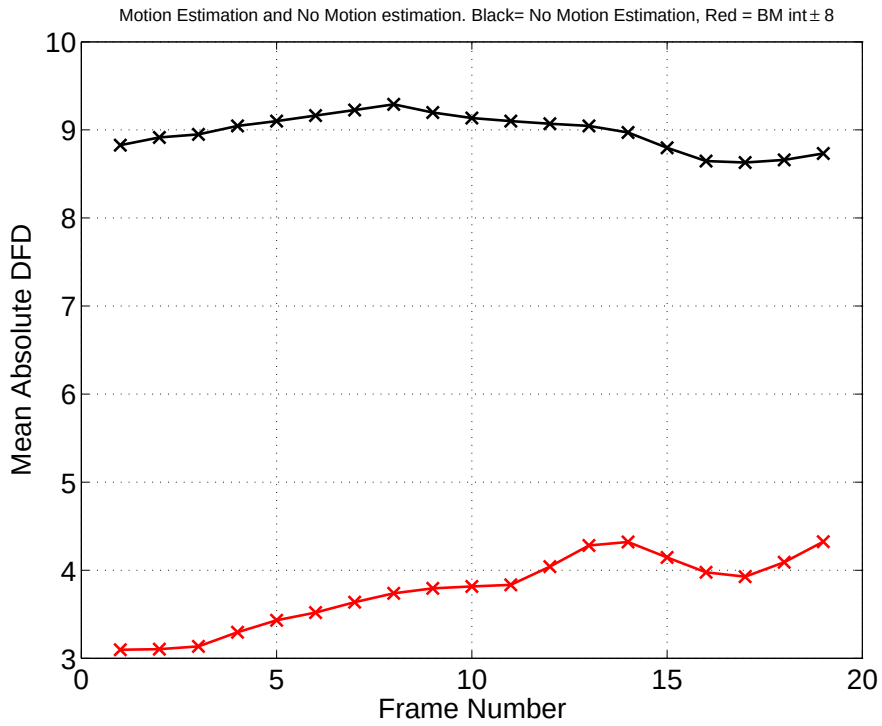

Figure 3: Showing the performance of Full Search Block Matching w.r.t. the mean absolute motion compensated error or MAE of DFD.

in prediction (or motion compensated) error $e(h,k)$ is caused by motion estimation.

4.1.3 Three step search

The simplest mechanism for reducing the computational burden of Full Search BM is to reduce the number of motion vectors that are evaluated. The Three-step search is a hierarchical search strategy that evaluates first 9 then 8 and finally again 8 motion vectors to refine the motion estimate in three successive steps. At each step the distance between the evaluated blocks is reduced. The next search is centred on the position of the best matching block in the previous search. It can be generalised to more steps to refine the motion estimate further. Figure 4 shows the searched blocks in frame $n-1$ for this process.

4.1.4 Cross Search

The cross search is another variant on the subsampled motion vector visiting strategy. It changes the geometry of the search pattern to a + or × pattern. Figure 4 shows the searched blocks in frame $n-1$ for this process. If the best match is found at the centre of the search pattern or the boundary of the search window, then the search step is reduced.

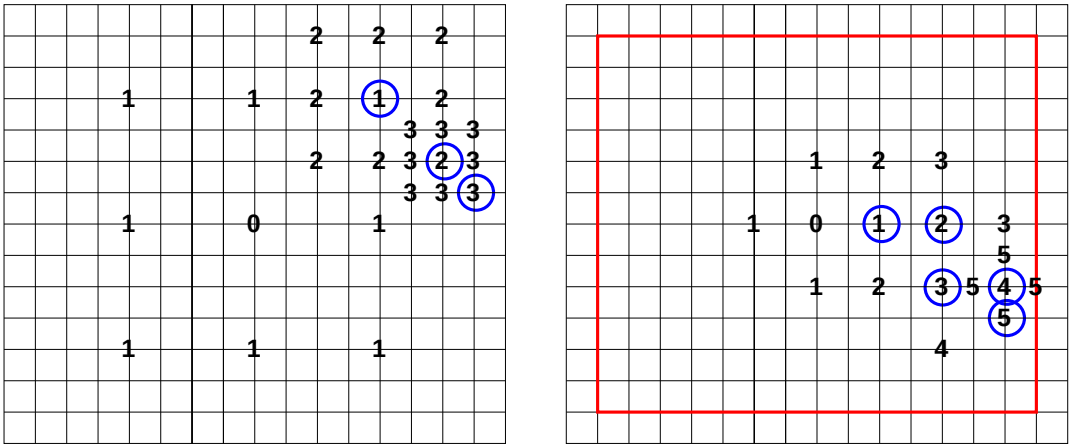

Figure 4: Illustration of searched locations (central pixel of the searched block is shown) in Three-step BM (left) and Cross-search BM (right). The search window extent is shown in red for Cross-search. The best matches at each search level are circled in blue.

4.1.5 Problems

The BM algorithm is noted for being a robust estimator of motion since noise effects tend to be averaged out over the block operations. However, if there is no textural information in the the two blocks compared, then noise dominates the matching process and causes spurious motion estimates.

This problem can be isolated [3] by comparing the best match found (E_m), to the 'no motion' match (E_0). If these matches are sufficiently different then the motion estimate is accepted otherwise no motion is assumed. A threshold acts on the ratio $r_b = \frac{E_0}{E_m}$. The error measure used is the MAE. If $r_b < t$, where t is some threshold chosen according to the noise level suspected, then no motion is assumed. This algorithm verifies the validity of the motion estimate once motion is detected.

The main disadvantages of Block Matching are the heavy computation involved (although these are byte wise manipulations) and the motion averaging effect of the blocks. If the blocks chosen are too large then many differently moving objects may be enclosed by one block and the chosen motion vector is unlikely to match the motion of any of the objects. The advantages are that it is very simple to implement[2] and it is robust to noise due to the averaging over the blocks.

4.2 Gradient based approaches

The second school of thought regarding motion estimation evolved out of a closer look at the image model in equation 4. The techniques were initially developed independently by workers in Video Coding and those in Computer Vision. The development of this class of techniques can be traced back to video coding work in 1976. The work was motivated by the need to find a motion estimator that required less computation than BM and also to avoid the problem of the motion averaging of

[2]It has been implemented on Silicon for video coding applications.

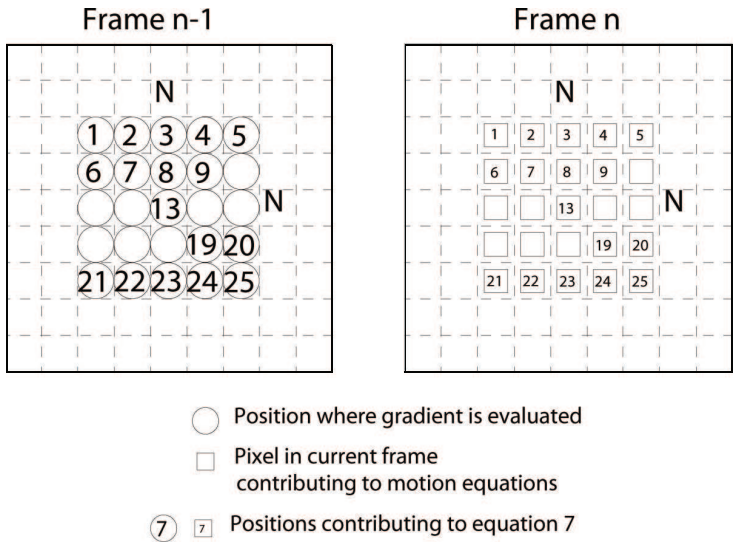

Figure 5: Motion estimation via spatial gradients. A block of pixels of size $N \times N$ in frame n is chosen to set up N^2 motion equations; $N = 5$. Each equation requires the evaluation of a gradient in the previous frame. The initial motion estimate is [0 0].

the blocks.

The basic connection between all the gradient based techniques is that they rearrange the model equation 4 to have direct access to the motion parameter $\mathbf{d}$. The equation 4 may be linearized about $\mathbf{d}$ using a Taylor series expansion as follows:

$$\begin{aligned} I_n(\mathbf{x}) &= I_{n-1}(\mathbf{x}+\mathbf{d}) \\ \text{Alternatively: } I_n(i,j) &= I_{n-1}(i+d_h, j+d_v) \\ \Rightarrow I_n(i,j) &= I_{n-1}(i,j) + d_h \frac{\partial I_{n-1}(i,j)}{dx} + d_v \frac{\partial I_{n-1}(i,j)}{dy} + e_n(i,j) \\ \Rightarrow I_n(\mathbf{x}) &= I_{n-1}(\mathbf{x}) + \mathbf{d}^T \nabla I_{n-1}(\mathbf{x}) + e_{n-1}(\mathbf{x}) \end{aligned} \tag{9}$$

Here $e_{n-1}(\mathbf{x})$ represents the higher order terms of the expansion, $\nabla = [\frac{\partial}{dx} \quad \frac{\partial}{dy}]$ which is the usual multidimensional gradient operator, and the motion vector has two components $\mathbf{d} = [d_h,\ d_v]$. The equation can be rearranged to yield an expression involving the Displaced Frame Difference with zero displacement.

$$\begin{aligned} \text{DFD}(\mathbf{x}, 0) &= I_n(\mathbf{x}) - I_{n-1}(\mathbf{x}) \\ &= \mathbf{d}^T \nabla I_{n-1}(\mathbf{x}) + e_{n-1}(\mathbf{x}) \end{aligned} \tag{10}$$

This equation represents the relation between a number of observables and $\mathbf{d}$. However, there is one equation and 2 variables. To solve the equation therefore, at least one more equation is necessary. If the assumption is made that within a small region all the pixels behave the same with respect to motion, then several more equations can be set up. This assumption will be referred to as the Smooth Local Flow assumption (already used above in BM). Therefore, neglecting the higher order terms allows an equation to be set up at each pixel in some defined region to yield a vector equation

as below. The situation is illustrated in Figure 5.

$$\begin{aligned} \mathbf{z}_0 &= \mathbf{G}\mathbf{d} \qquad (11) \\ \text{where } \mathbf{z}_0 &= \begin{bmatrix} DFD(\mathbf{x}_1, 0) \\ DFD(\mathbf{x}_2, 0) \\ \vdots \\ DFD(\mathbf{x}_{N^2}, 0) \end{bmatrix} \\ \text{and } \mathbf{G} &= \begin{bmatrix} \frac{\partial}{\partial x} I_{n-1}(\mathbf{x}_1) & \frac{\partial}{\partial y} I_{n-1}(\mathbf{x}_1) \\ \frac{\partial}{\partial x} I_{n-1}(\mathbf{x}_2) & \frac{\partial}{\partial y} I_{n-1}(\mathbf{x}_2) \\ \vdots & \vdots \\ \frac{\partial}{\partial x} I_{n-1}(\mathbf{x}_{N^2}) & \frac{\partial}{\partial y} I_{n-1}(\mathbf{x}_{N^2}) \end{bmatrix} \end{aligned}$$

In Figure 5, the set of N^2 equations is set up using the block indicated. The vectors are therefore generated by assembling the row–wise observations into columns. The gradient measurements are made using a simple difference technique, for example the horizontal gradient at pixel 2 in Figure 5 is $\frac{I(3)-I(1)}{2}$. This is admittedly a noisy process and a more robust class of gradient estimation techniques, using curve fitting can be used. These are not discussed here.

A solution for $\mathbf{d}$ can then be generated using a pseudoinverse approach as follows.

$$\begin{aligned} \mathbf{z}_0 &= \mathbf{G}\mathbf{d} \\ \Rightarrow \mathbf{G}^T\mathbf{z}_0 &= \mathbf{G}^T\mathbf{G}\mathbf{d} \\ \Rightarrow \mathbf{d} &= [\mathbf{G}^T\mathbf{G}]^{-1}\mathbf{G}^T\mathbf{z}_0 \qquad (12) \end{aligned}$$

This solution yields $\mathbf{d}$ in one step by using gradient measurements in some region as in Figure 5. It is one of the early solutions proposed by Cafforio et al, Netravali et al, Martinez [7] and others. The important drawback with this approach, recognized in [8], is that the Taylor series expansion is only valid over very small distances. To overcome this problem, a recursive solution was introduced called the pel–recursive approach. This new solution refines an initial estimate for $\mathbf{d}$ until no further reduction in DFD is observed.

4.2.1 Pel-recursive Motion Estimation

Rather than expand the model equation about $\mathbf{x}$, the image function is linearized about a current guess for $\mathbf{d}$, say $\mathbf{d}_i$. The idea is to generate an update, $\mathbf{u}$ for the current estimate that will eventually force the estimation process to converge on the correct displacement. The update is defined such that $\mathbf{d} = \mathbf{d}_i + \mathbf{u}_i$. The update is intended to be a small one to maintain the validity of the Taylor series expansion. Therefore, equation 9 becomes

$$\begin{aligned} \mathbf{u}_i &= \mathbf{d} - \mathbf{d}_i \\ \Rightarrow I_n(\mathbf{x}) &= I_{n-1}(\mathbf{x} + \mathbf{d}_i + \mathbf{u}_i) \end{aligned}$$

Expanding about $\mathbf{u}_i$ then yields

$$I_n(\mathbf{x}) = I_{n-1}(\mathbf{x} + \mathbf{d}_i) + \mathbf{u}_i^T \nabla I_{n-1}(\mathbf{x} + \mathbf{d}_i) + e_{n-1}(\mathbf{x} + \mathbf{d}_i) \tag{13}$$

The solution may then continue as before. In this case, the *only* difference between the solution in equation 12 and this one is that the displacement estimate in 12 is used to recursively update the current estimate. The process then continues again.

The algorithm can be summarised in effect as

1. Establish an initial estimate for motion $\mathbf{d}^i$
2. Using this estimate, fetch the motion compensated block in the previous frame that corresponds to the one being processed in the current frame.
3. Evaluate $\mathbf{G}$, $\mathbf{z}$ for these blocks
4. Solve for the *update* motion estimate, $\mathbf{u} = [\mathbf{G}^T\mathbf{G}]^{-1}\mathbf{G}^T\mathbf{z}$
5. Update the current estimate to yield a new estimate $\mathbf{d}^{i+1} = \mathbf{d}^i + \mathbf{u}$

4.2.2 A Wiener estimate

A Wiener solution for the displacement (first introduced in [1]) is more robust to noise. It handles the higher order terms more effectively by considering their effect on the solution to be the same as Gaussian white noise. The new vector equation to be solved becomes

$$\mathbf{z}_i = \mathbf{G}\mathbf{u}_i + \mathbf{e} \tag{14}$$

$e(\mathbf{x} + \mathbf{d}_i)$ represents the higher order terms which are considered to be Gaussian white noise of variance σ_{ee}^2. Note that the matrices $\mathbf{G}$, $\mathbf{z}_i$, now consist of terms similar to those defined earlier in equation 11, but the arguments are offset by the current estimated displacement, $\mathbf{d}_i$. Compare this to $\mathbf{z}_0 = \mathbf{G}\mathbf{d}$ which is used in standard gradient based motion estimation.

The Wiener solution chooses the update $\hat{\mathbf{u}}_i = \mathbf{L}\mathbf{z}_i$ which minimizes the expected value of the squared error between the true update $\mathbf{u}_i$ and the estimate $\hat{\mathbf{u}}_i$. The objective function is therefore,

$$E[(\mathbf{u}_i - \hat{\mathbf{u}}_i)^T(\mathbf{u}_i - \hat{\mathbf{u}}_i)] \tag{15}$$

Minimizing this function with respect to $\mathbf{L}$ yields the following solution for $\hat{\mathbf{u}}_i$.

$$\begin{aligned} \hat{\mathbf{u}}_i &= [\mathbf{G}^T\mathbf{G} + \mu\mathbf{I}]^{-1}\mathbf{G}^T\mathbf{z} \\ \mu &= \frac{\sigma_{ee}^2}{\sigma_{uu}^2} \end{aligned} \tag{16}$$

Here, σ_{uu}^2 is the variance of the estimate for $\hat{\mathbf{u}}_i$, provided that both components of the motion have the same variance. Also, note the use of μ as a regularizing term in the required matrix inverse. The

success of the method lies in treating the observables $\mathbf{z}, \mathbf{G}, \mathbf{e}$ as random variables. The estimator will be referred to as the Wiener based motion estimator (WBME) in future references.

After the update is estimated, $\mathbf{d}_i$ is refined to yield the displacement that is used in the next iteration, $\mathbf{d}_{i+1} = \mathbf{d}_i + \mathbf{u}_i$, and the process continues again.

4.2.3 Interpolation and convergence

During the iterative process of gradient based motion estimation, the observables may have to be evaluated at fractional displacements. This is because the solution of the equations above do not typically result in integer accurate motion estimates. Usually bilinear interpolation is employed to keep computation at a minimum.

Iteration is terminated when the algorithm converges. This is usually detected by thresholding the size of the update vector and the size of the current Mean Square Error. When either of these two quantities are below their respective thresholds, convergence is assumed.

4.2.4 Advantages and Disadvantages

The main advantage of the Wiener based pel–recursive estimator is that it requires much less computation than the Block Matching algorithm. It is also capable of resolving both fractional and integer displacements without changing the number of operations involved. Because the computational complexity is so low, many in the Video Coding field have used the technique to estimate a motion vector at each pixel. This is done so that there is less chance of the 'motion averaging' effect of blockwise manipulations. The approach is of limited use because some finite support region must always be employed to set up the necessary equations. Depending on the size of that region, errors are likely.

The most important disadvantage of the estimator is that even as an iterative process, it cannot estimate large displacements because of the limited effectiveness of the first order Taylor series expansion. This is in contrast to BM which is only limited by the defined search space. This is a very *large* problem with gradient based motion estimators.

Finally, all gradient based motion estimators fail when the conditioning of the matrix product $\mathbf{G}^T\mathbf{G}$ is poor. This happens when the searched block has no texture, or when there is a single edge though the block. It is possible to use the eigenvalues and eigenvectors of this matrix to identify when these problems occur. This is a more advanced topic.

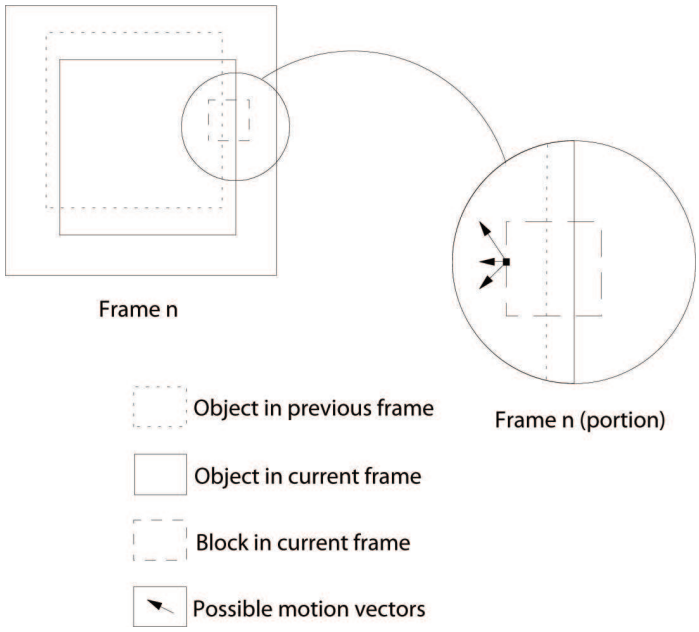

Figure 6: Motion ambiguity at an edge. *The motion estimate is most accurate along the direction of maximum image gradient.*

5 Ambiguity in motion estimation

A fundamental fact about the translational model of motion is that it cannot be solved using one observation at one pixel alone. To overcome this difficulty the Smooth Local Flow assumption must be made to allow observations at several pixels to be used.

This assumption causes an ambiguity in block based motion estimation. The situation is illustrated in Figure 6. The rectangular object of uniform intensity is moving diagonally and so has two non zero components of motion. If the block used for estimating motion in either a BM or WBME algorithm is located at an edge as shown in the Figure then several motion estimates can be found which will cause a zero DFD. Only one of these is the true motion vector. The situation illustrates two points,

- The motion estimate is most accurate in the direction of maximum image gradient (or contrast). At edges this direction is perpendicular to the edge, at corners there are two directions along which motion estimation is accurate. Hence, for corners both components of motion (horizontal and vertical), can be estimated accurately.
- The size of the block used for estimating motion affects the locality of the motion estimate. Because the block in Figure 6 is small compared to the size of the object, a local motion estimate results which may not reflect the true motion of the object. It is this effect which causes the motion estimated in that block to be different from the global motion of the object. This is called the *Aperture Effect*

If the block used for motion estimation is located in the centre of the moving object in Figure 6, then it is likely that neither estimator will estimate any motion at all. There will be zero error for zero displacement and so the motion detector will not flag motion. Even if motion estimation were to be engaged then the zero or near zero DFD coupled with the zero gradient in the previous frame will bias the result toward a zero motion vector.

6 Adaptive Gradient Based Motion Estimation

Most of the problems with the block based gradient estimators like WBME occur when the gradient matrix is ill-conditioned. This can be dealt with by adapting μ in the WBME to the ill–conditioning of $\mathbf{M}_g = [\mathbf{G}^T\mathbf{G}]$ during the pel–recursive process. One possibility is shown below

$$\begin{aligned} \mathbf{d} &= [\mathbf{G}^T\mathbf{G} + \mu\mathbf{I}]^{-1}\mathbf{G}^T\mathbf{z} \\ \textit{where } \mu &= |\mathbf{z}|\frac{\lambda_{max}}{\lambda_{min}} \end{aligned} \tag{17}$$

where λ are the eigen values of the 2×2 matrix $\mathbf{M}_g$. As the required inverse solution becomes ill–conditioned (measured as a ratio of eigenvalues), μ increases, thus increasing the damping in the system. The additional multiplying factor of $|\mathbf{z}|$ helps to further stabilize the situation by relaxing the damping when the error is small and the motion estimate is close to convergence.

A full SVD decomposition of $\mathbf{M}_g$ leads to the best adaptive scheme. Since $\mathbf{M}_g$ is just a 2×2 matrix, the computational load of this step is very small.

7 Optic Flow

Given $I(x, y, t)$ is the intensity of a pixel at (x, y) at time t, assuming the intensity is constant along a motion trajectory we have

$$\frac{dI(x(t), y(t), t)}{dt} = 0 \tag{18}$$

Using the chain rule for differentiation this can be expressed as

$$\frac{\partial I(x, y, t)}{dx}\frac{\partial x}{dt} + \frac{\partial I(x, y, t)}{dy}\frac{\partial y}{dt} + \frac{\partial I(x, y, t)}{dt} = 0 \tag{19}$$

The partial derivatives $\frac{\partial x}{dt}$, $\frac{\partial y}{dt}$ are the components of horizontal and vertical velocities at site (x, y) respectively. This equation is known as the Optical Flow Equation and can be written as

$$u_h g_h(\mathbf{x}, t) + u_v g_v(\mathbf{x}, t) + \frac{\partial I(\mathbf{x}, t)}{dt} = 0 \tag{20}$$

where u_h, u_v are the horizontal and vertical components of motion, g_h, g_v are horizontal and vertical spatial gradients, and $\frac{\partial I(\mathbf{x},t)}{dt}$ is the instantaneous DFD at pixel site $\mathbf{x}$. This equation can also be derived from the Taylor Series expansion in equation 10 assuming the error in the expansion is zero.

The Optic Flow constraint is valid only for small u_h, v_h and has been used to estimate motion on a pixel basis. To do so, it is necessary to incorporate a constraint on the motion field as well. Assuming that objects tend to be rigid, then the motion field in a local region should be smooth. These two constraints were used by Horn and Schunck to estimate motion using an iterative algorithm.

Define the horizontal and vertical components of motion at a site $\mathbf{d}(\mathbf{x}) = [d_1(\mathbf{x}),\ d_2(\mathbf{x})]$. Then the task is to minimise the spatial variation in the motion field $D_s(\mathbf{x})$

$$D_s(\mathbf{x}) = \left(\frac{\partial d_1}{dx}\right)^2 + \left(\frac{\partial d_1}{dy}\right)^2 + \left(\frac{\partial d_2}{dx}\right)^2 + \left(\frac{\partial d_2}{dy}\right)^2 \tag{21}$$

with the constraint that the Optical Flow equation is satisfied.

It is possible to further adapt this method by altering the spatial motion smoothness measure D_s above. Since edges in an image usually coincide with object edges, and since different objects can move in different directions, motion should not be smooth across an object edge, hence should not be the same across an image edge. Thus the smoothness constraint above can be relaxed across image edges. This leads to a much improved motion estimation process.

8 Multiresolution Motion Estimation

Throughout the discussion so far it has been noted that gradient based techniques can only effectively estimate small displacements. A correspondence technique, such as BM, is only limited in this respect by the extent of the search space used. However, increasing the search space to deal with a large displacement, increases the computational requirement. Considering that motion pictures can involve displacements in excess of 10 pixels per frame, increasing the search space of BM to deal with this motion results in a huge increase in computation.

It has been widely accepted that the multiresolution schemes provide the most practical way of dealing with this problem effectively. The idea of a multiresolution representation was presented early on for use in Image Coding by Burt and Adelson [4]. The technique can reduce the computational requirement to estimate a particular displacement in the case of BM [2]. In the case of gradient based techniques, the scheme can make the problem better conditioned and increase convergence rates.

The concept is simple. Create a series of images at successively coarser resolution, then begin motion estimation at the coarsest scale. The motion estimates are refined at each successively higher resolution level.

8.1 Low Pass Pyramid

The multiresolution scheme begins by building successively smaller versions of the images in terms of spatial extent, while still maintaining useful image information in each version. The original images

are low pass filtered and subsampled until the displacements measured in the subsampled frames are small enough to allow reasonably accurate pel–recursive, or BM motion estimation with a small search space.

The motion vectors found at the lower resolutions are then projected onto the higher resolutions where the motion estimation process is allowed to continue. Successive projections of the estimated motion field eventually leads to an initial motion field at the highest resolution and then it is presumed that only a few iterations will be needed for a pel–recursive algorithm to converge. In the case of BM the search space at each level can be drastically reduced, because a small displacement at the higher levels represents a large displacement at the original level. It is typical to subsample by a factor of 2 so that if the original level is of resolution $H \times V$ then level l of N levels has a size $\frac{H}{2^l} \times \frac{V}{2^l}$. Motion of magnitude k pixels at level 0 (the original resolution level), is then reduced to $k \times 2^{-(N-1)}$ at level $N-1$. Figure 7 illustrates the pyramidal representation.

The underlying motivation for a multiresolution approach here is to create a representation of the original image in which the motion is small enough to allow a motion estimation algorithm to terminate successfully. To this end the basic character of the image at the varying scales must be maintained. This is necessary to ensure that the motion fields estimated at the lower resolutions bear some resemblance to the actual motion field at the original resolution.

8.2 Creating the pyramid

A Gaussian shaped low pass filter is typically used to create the pyramid and also slightly blur the image at each scale. This latter point is useful in that it artificially increases the proportion of the first order term in the Taylor series expansion of the image function, thus stabilizing the iterative process. Further, the filter is non directional and so does not favour motion estimation in any direction.

8.3 Algorithm

The multigrid motion estimation algorithm is enumerated below

1. Generate L levels of the multiresolution pyramid, such that level $l = 0$ is the original resolution image, and level $l = L - 1$ is the image with the smallest resolution.
2. Set the initial level to be the level with the most coarse resolution, level $l = L - 1$. Set the initial field at this level to be a set of zero vectors.
3. Generate an estimate of the motion field at the resolution level l using the current starting field.
4. If $l = 0$ then goto 8.

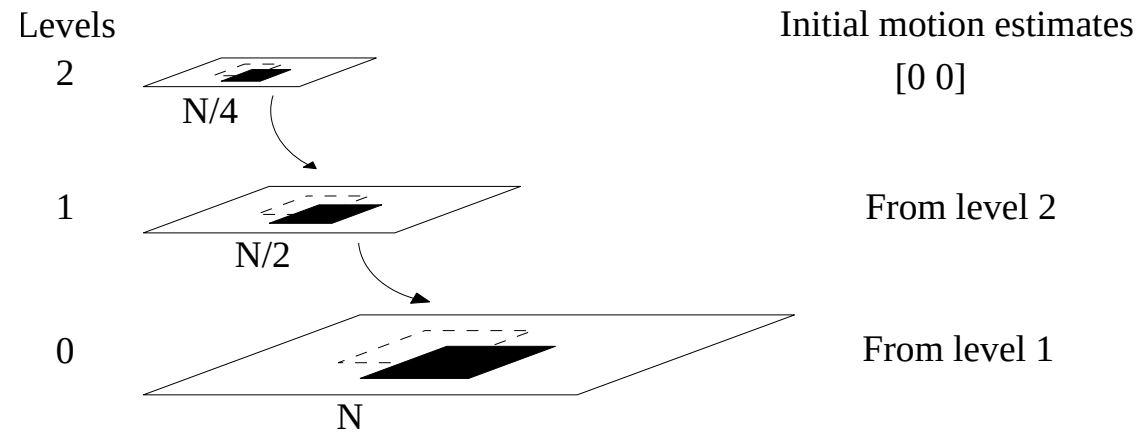

Figure 7: Multiresolution pyramid showing 3 level decomposition of an $N \times N$ frame. The dotted lines indicate the position of the object in the previous frame. Note this displacement is reduced at the same rate as the level dimensions.

5. Propagate the motion field from the coarse resolution level l down to the next higher resolution level $l - 1$.
6. Set the current resolution level to be $l = l - 1$.
7. Goto 3.
8. Stop.

Some interpolation must be used to propagate the motion field one level down, i.e. onto the higher resolution level. This is necessary to assign all pixel sites at the new resolution with a start vector. Bilinear interpolation is risky since it will 'blur' the start motion field unnecessarily. Zero-order hold interpolation (repeat vectors over 2×2 pixels) is less prone to this problem.

8.4 Motion Haloes

There is a drawback with the multiresolution motion estimation scheme. It brings similar regions, not related by motion, closer together. A typical effect is the assignment of displacement vectors to stationary regions just outside moving regions. For gradient based algorithms this could cause a lack of convergence at the lower levels. The effect is called the vector halo effect. A simple solution to this problem [2] is to use a motion detector at the highest resolution level level, regardless of the final vector estimate. Therefore, before assigning a motion vector to a region, this is a double check to verify that it is in fact moving.

9 Summary

Motion estimation is key for the processing of digital video. There are two main classes of motion estimator, those based on Block Matching and those based on gradient analysis of some kind. All the motion estimators can estimate motion pixel by pixel or on a block basis, however it is understood that the motion equation cannot be solved at a single pixel site alone.

The two main problems with block based motion are the aperture problem and the motion ambiguity at a strong image edge. Motion estimation is therefore most accurate at *corners* in the image. The drawbacks of gradient based motion are typically that motion cannot be estimated where there is no image gradient, and the motion must be small. However gradient based estimators are generally of lower computational load than BM estimators, and are more accurate.

Many of the problems with BM and computational load can be overcome with a multiresolution scheme for motion estimation. Similarly the difficulties with the small motion assumption in gradient based motion estimation can be overcome with multiresolution schemes.

The book *Digital Video Processing*, Murat Tekalp, Prentice Hall, ISBN 0-13-190075-7 has excellent coverage of motion estimation techniques. Lim covers a practical range of techniques in pages 497-506.

References

[1] J. Biemond, L. Looijenga, D. E. Boekee, and R.H.J.M. Plompen. A pel–recursive Wiener based displacement estimation algorithm. *Signal Processing*, 13:399–412, 1987.

[2] M. Bierling. Displacement estimation by heirarchical block matching. In *SPIE VCIP*, pages 942–951, 1988.

[3] J. Boyce. Noise reduction of image sequences using adaptive motion compensated frame averaging. In *IEEE ICASSP*, volume 3, pages 461–464, 1992.

[4] P. Burt and E. Adelson. The Laplacian pyramid as a compact image code. *IEEE transactions on Communications*, 31:532–540, April 1983.

[5] M. Ghanbari. The cross–search algorithm for motion estimation. *IEEE Transactions on Communications*, 38:950–953, July 1990.

[6] B. Girod. Motion–compensating prediction with fractional–pel accuracy. *IEEE Transactions on Communications*, Accepted March 1991.

[7] D. M. Martinez. *Model–based motion estimation and its application to restoration and interpolation of motion pictures.* PhD thesis, Massachusetts Institute of Technology, 1986.

[8] A. Netravali and J. Robbins. Motion–compensated television coding: Part 1. *The Bell System Technical Journal*, 58:631–670, March 1978.

[9] A. Zaccarin and B. Liu. Fast algorithms for block motion estimation. In *IEEE ICASSP*, volume 3, pages 449–452, 1992.