MOTION ESTIMATION FOR VIDEO BANDWIDTH COMPRESSION USING A HETEROGENEOUS PYRAMID IMAGE PROCESSING ARCHITECTURE

G.R.Nudd, S.C.Clippingdale, R.M.Howarth, T.J.Atherton, N.D.Francis, G.J.Vaudin, D.W.Walton

Dept. of Computer Science, University of Warwick, Coventry CV4 7AL, U.K.

ABSTRACT

In many image processing contexts there is a requirement for some form of motion estimation. A number of strategies exist, differing in the accuracy and granularity of the motion estimate and in their computational requirements. The simplest and most coarse-grained scheme is 'block' motion estimation, where the current frame is divided into blocks of pixels and for each block an estimate of its most likely motion is computed by reference to a neighbourhood in the previous frame. Such coarse motion estimates are of utility in situations where a fairly approximate but fast trajectory estimate is required. Examples are motion-compensated interframe predictive coders and various aerospace applications. The block motion estimator is simple and amenable to parallel computation, making it feasible for implementation at real-time frame rates. This paper describes the mapping of the block motion estimator on to the lower levels of the Warwick Pyramid Machine (WPM), a heterogeneous pyramid architecture for parallel image processing / understanding which is currently under development at the University of Warwick. The paper describes how the required computations are performed and presents simulated timings. The advantages of the multi-SIMD architecture of the WPM are illustrated.

INTRODUCTION

Motion is a fundamental property of time-varying imagery and indeed is usually responsible for those variations with time which are of interest to the end-user. It is natural, therefore, that in systems which process such imagery there is frequently a requirement for some kind of motion detection and estimation. Examples include systems for object tracking and identification, robot navigation, and motion compensation for interframe predictive coding. The motion estimation strategy chosen depends on the requirements of the particular application. For example, a system which aims to track a number of moving targets with complex motions may require an accurate motion estimate at every point in the scene, whereas interframe predictive coders are not as demanding, since the predictor is usually situated in the feedback loop at the coder and decoder, and a coarse-grained approximation to the image motion over blocks of pixels is usually adequate.

The most general, accurate and computationally expensive methods involve the calculation, at each pixel in a sequence of frames, of the *optical flow* vector field [1]. Such schemes may be designed in the spatiotemporal or spatiotemporal-frequency domains [2], and tend to use large amounts of three-dimensional filtering. There are no constraints for example on the number of moving objects or textures in the scene or on the permissible types of motion, nor is the background constrained to be stationary. Applications include aerial target tracking and sophisticated vision systems, and energy detection in the spatiotemporal-frequency domain has been postulated [3] as a mechanism for motion estimation in biological (e.g. human) vision.

Somewhat more limited in terms of permissible types of object and motion are the *feature correspondence* methods. These work by extracting particular features such as line segments or corners from each of two frames in the image sequence. Corresponding features are paired and motions sought which are consistent with the translations and rotations measured between the members of each feature pair (e.g. [4]). Data which is not representative of the predetermined feature types will be missed or may even produce erroneous results. Such constrained methods are useful in situations where there is *a priori* knowledge of the type of data involved, for example in systems for automatic registration of components in manufacturing where it might be known that there is only one object, with geometric characteristics such as corners, moving against a stationary background. In more general contexts these methods may be confused by 'unexpected' data; the computational burden is heavily data-dependent and may increase rapidly with the number of detected features.

The simplest motion estimation strategy, known as *block motion estimation,* divides the current frame from the image sequence into blocks of pixels and for each block attempts to locate the most likely translational motion within a *search window* from the previous frame. The method is relatively crude by comparison with optical flow and feature correspondence methods and produces a somewhat coarse-grained estimate as its output. However, it offers a simplicity of computation which renders feasible its implementation at real-time frame rates on a suitable processor. The accuracy of the estimate is sufficient to give a reasonable first approximation to object motions. This method is suitable for motion estimation in motion-compensated interframe predictive coders which, as noted above, do not require very fine-grained or accurate motion information and are often compelled to operate at real-time data rates.

After presenting an outline of the block motion estimation scheme, this paper will describe its implementation on the Warwick Pyramid Machine (WPM), a Multiple Single-Instruction-Multiple-Data (M-SIMD) heterogeneous pyramidal image processing architecture (figure 1).

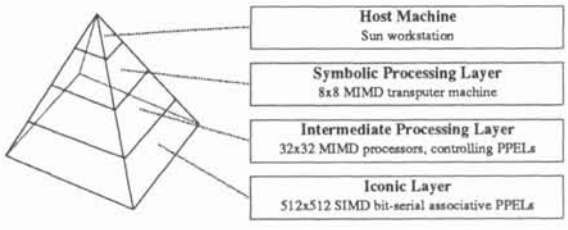

Figure 1
The Warwick Pyramid Machine

The machine is designed for the extraction of high-level, symbolic representations from iconic data and operates in a highly parallel manner at each of its levels. The block motion estimator, which is essentially a low-level or iconic process, maps on to the bottom two levels of the machine. Simulated timings for the computation of the motion estimate will be presented and shown to be compatible with real-time speed requirements. The advantages of the M-SIMD architecture will be discussed.

BLOCK MOTION ESTIMATION

The block motion estimator divides the current frame into blocks of M × M pixels and seeks the 'best match' for each *current block* over all displacements within a *search window* in the previous frame of (M + 2D) × (M + 2D) pixels, centered on the current block. The current block and search window are depicted in figure 2. The maximum tested displacement is D pixels in the horizontal and vertical directions. A typical and convenient value for the block size M is M = 8 or M = 16.

For each displacement $(i,j), -D \le i,j \le D$ within the search window, an index of the correspondence between the displaced current block and the M × M portion of the search window under the displaced current block is calculated. The index used is *mean absolute error* (MAE), which is simply the sum of the absolute values of the differences between corresponding pixels (with the appropriate displacement) in the two frames,

$$MAE\ {}^{I,J}_{i,j} = \sum_{k=0}^{M-1}\sum_{l=0}^{M-1} \left| u^{N}_{MI+k,MJ+l} - u^{N-1}_{MI+k+i,MJ+l+j} \right| , \tag{1}$$

where the displacement is (i,j), the current block is indexed (I,J), and the superscripts N and N–1 refer to the frame number. The MAE is calculated for all displacements (i,j), $-D \le i,j \le D$, and that displacement (u,v) which yields the minimum MAE,

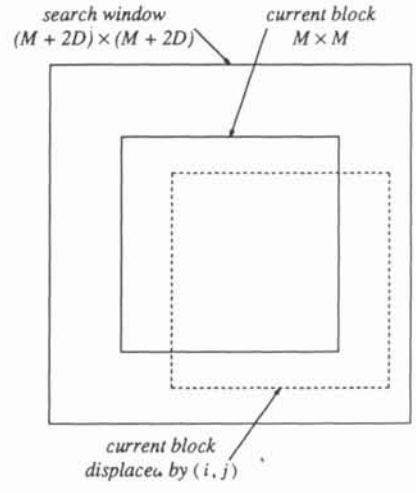

Figure 2
Search Window and Current Block

$$MAE\ {}^{I,J}_{u,v} \le MAE\ {}^{I,J}_{i,j} , \quad -D \le i,j \le D , \tag{2}$$

is retained as the best estimate for the motion of the current block (I,J). Should more than one displacement (i,j) yield the minimum value of MAE, the motions of the surrounding blocks are examined and that value of displacement which is most consistent with these motions is retained.

THE WARWICK PYRAMID MACHINE

The Warwick Pyramid Machine (WPM) architecture (see figure 1) consists of three processing layers, known as *iconic, intermediate* and *symbolic*. Each *intermediate layer processor* (ILP) controls a *cluster* of 16 × 16 single-pixel *pixel processing elements* (PPELs) on the iconic layer. The PPELs in each cluster operate in broadcast SIMD mode, but the ILPs are independent, providing individual local control for each cluster, and thus the combination of the lower two levels is a multiple-SIMD (M-SIMD) structure. The ILPs communicate with the symbolic (transputer) layer above via a dual-port memory and with the PPELs on the iconic layer through various associative response mechanisms. The relationship between the iconic and intermediate levels is depicted in figure 3. Note that for reasons of clarity, only those features used in the block motion estimation are shown.

The PPELs use bit-serial arithmetic and output serial data to the intermediate layer via a flag labelled 'X'. Also connected to the output of the X flag is a fast adder tree, which gives the ILP a count of the number of PPELs in the cluster which have the X flag set. The PPEL array possesses eight-way connectivity. A much-simplified block diagram of a PPEL element is shown in figure 4.

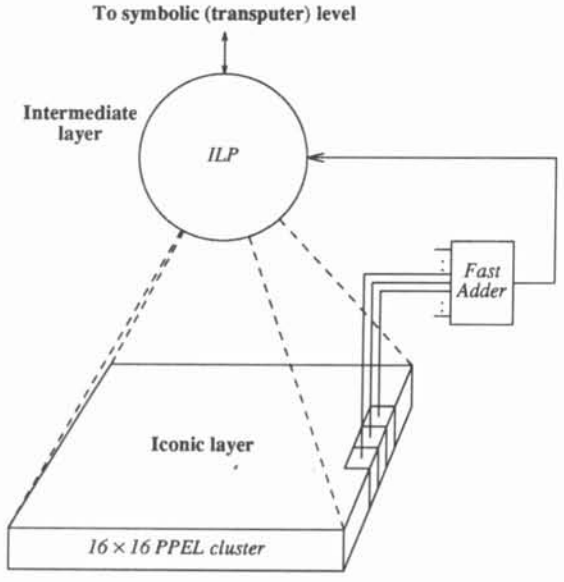

Figure 3
ILP/PPEL cluster (simplified)

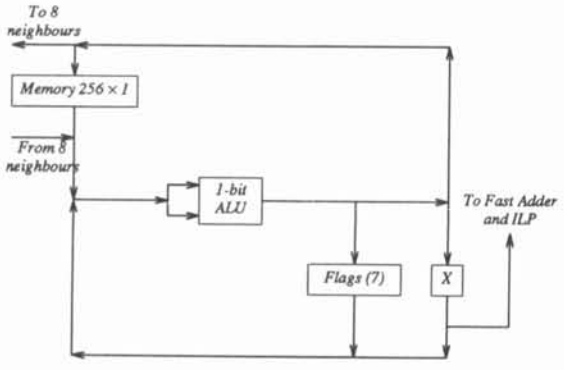

Figure 4
Simplified PPEL Block Diagram

BLOCK MOTION ESTIMATION ON THE WARWICK PYRAMID MACHINE

The design of the iconic and intermediate layers of the WPM facilitates the parallel computation of equation (1). Suppose that frames N and N–1 reside in PPEL memory, with one pixel from each frame stored in the memory of each PPEL. Suppose also that the two frames are aligned such that the displacement (i, j) in (1) is $(i, j) = (0, 0)$. Then the absolute differences

$$|u^{N}_{MI+k,MJ+l} - u^{N-1}_{MI+k,MJ+l}|$$

may be formed in parallel over the entire array by a subtraction at each PPEL followed by an absolute value operation. These absolute differences are then available, for every current block, for $0 \le k \le M-1$ and $0 \le l \le M-1$. The summation in (1) may be performed in parallel over each cluster of 16 × 16 PPELs using the fast adder tree, and the resulting MAE value is stored at the ILP level. This illustrates the advantage of using a blocksize of M = 8 or M = 16, since each ILP handles exactly four current blocks for M = 8 or one for M = 16. For the case of M = 8, the four sums (MAE values) must be formed in sequence using the ILP address mask facility [5],[6] to select the appropriate 8 × 8 block of PPELs.

In order to evaluate (1) at the next value of displacement (say $(i, j) = (0, 1)$), the previous frame (frame N–1) must be shifted by $(-i, -j) = (0, -1)$ relative to the current frame. This is achieved by using the eight-way neighbour connectivity across the PPEL array — in this example, each PPEL reads from the neighbour to its east. The value of $MAE^{I,J}_{0,1}$ is calculated in the same fashion as was $MAE^{I,J}_{0,0}$ and is compared with $MAE^{I,J}_{0,0}$ for each current block (I, J). The lower value is retained along with the corresponding displacement (i, j).

The process of shifting the previous frame, forming the mean absolute error values and retaining only the lowest of these together with its corresponding displacement continues until all displacements (i, j), $-D \le (i, j) \le D$ have been covered. (Non-minimum MAE values may, if desired, be retained in the ILP memory for further processing.) At this stage, each ILP contains in memory the optimum displacement with its associated MAE value for one current block if the blocksize is M = 16 or for four current blocks if M = 8. These motion estimates are now available to subsequent processing such as the predictor stage of an inter-frame coder.

SIMULATION AND TIMINGS

A simulator for the lower levels of the WPM has been developed, in the form of an implementation of the *cluster programming language* (CPL) [6]. CPL is a high-level language which is used to program the intermediate and iconic layers of the WPM. The simulator presently runs on a Sun-3, but will soon be ported to the 8 × 8 transputer array (the symbolic layer) which is now operational. The use of bit-serial arithmetic in the PPEL ALU means that the processing time is linear with respect to the desired intensity (gray-level) resolution. Assuming 8-bit data, the subtraction in (1) requires 8 machine cycles, as does the absolute value operation. Formation of the sum requires that, for each bit, the PPELs in a cluster write the bit into the X flag, and a count of the number of active X flags in the cluster is then available to the ILP at the output of the adder tree. The process is repeated for the remaining bits in the modulus term, with the sum being accumulated at the ILP. Each one-bit summation over the cluster requires two machine cycles and so for 8-bit data, 16 machine cycles are required for the accumulation. For a current block size of M = 8, four accumulations must be performed at each ILP and so 64 cycles are needed. The frame shift requires eight cycles, bringing the total count to 40 or 88 machine cycles for each displacement (i, j) for block sizes of M = 16 or M = 8 respectively. Since there are $(2D + 1)^2$ values of displacement within the

search window, the total load is

$$L = 40 . (2D + 1)^2 \quad (3)$$

machine cycles for M = 16, or

$$L = 88 . (2D + 1)^2 \quad (4)$$

for M = 8. This figure excludes the comparison and storage of MAE values at the ILP level, since these functions may be performed while the PPEL layer is shifting the previous frame and computing the MAE for the next displacement. For a typical search window size of 15 × 15 pixels (D = 7) and M = 16, the total load is L = 9000 cycles, which with a 100ns cycle time requires 0.9ms, or approximately five percent of the total available interframe processing time for a 50Hz frame rate. The variation of processing time with search window size (in terms of the maximum displacement D) is illustrated in Table 1 below. The percentage of the frame interval required at a 50Hz frame rate is displayed as P_{FT} .

D	Time		P_{FT}	
	M = 16	M = 8	M = 16	M = 8
3	0.20ms	0.43ms	1.0%	2.2%
5	0.48ms	1.06ms	2.4%	5.3%
7	0.90ms	1.98ms	4.5%	9.9%
10	1.76ms	3.88ms	8.8%	19.4%
15	3.84ms	8.46ms	19.2%	42.3%

Table 1
Processing times

ADVANTAGES OF THE M-SIMD ARCHITECTURE

The block motion estimator as described above could not be implemented easily on a pure SIMD architecture, since the formation of the MAE values requires that processing is compartmentalised within the blocks defined in the image. A 'hybrid SIMD' architecture consisting of two SIMD layers of the appropriate granularities (one processor per current block on the upper layer and one per pixel on the lower) would suffice. However, the M-SIMD approach is superior where different types of processing may be required in different regions of the image (e.g. in different current blocks in the present application). If for some current block there are two 'candidate' displacements which yield the lowest MAE, for example, the ILP which maps on to that block may request information concerning the motion estimates for the surrounding blocks from the neighbouring ILPs in an attempt to select the most likely single motion from the candidates. On the other hand, if the MAE for any displacement (most likely (0 , 0)) were below some low threshold, the search on the particular current block could be aborted and the displacement taken immediately as the final value, possibly saving some time in subsequent computations. The ILP(s) in question could be assigned a useful task such as encoding while the rest of the array completes the motion estimation. Similarly, the M-SIMD architecture allows those ILPs which are situated at the edges of the array to handle edge effects by running code which differs from that used to program the interior ILPs.

In the case where a number of objects move independently in the scene, their motions *per se* may indicate that different processing will subsequently be required. For example, in aerospace applications a fast-moving object may call for immediate further processing with all of the requisite machine resources in order to ascertain whether it poses a threat or indicates a dangerous condition. Alternatively, in coding applications it may be advantageous for neighbouring current blocks with similar motion estimates to be merged and considered as one. The same is true for initial segmentations using motion as a basis. In this case, the nature of subsequent computation might depend on the class to which the current block is assigned. The spatial adaptivity offered by the M-SIMD approach thus has benefits both for the motion estimation itself and for subequent processing.

ACKNOWLEDGMENTS

This work is supported by the UK DTI/SERC Alvey program and by the US SDIO Innovative Science & Technology Office under contract N00014-87-G-0241 administered by Dr. K. Bromley, Office of Naval Research.

REFERENCES

[1] B.K.P.Horn, B.G.Schunk, *Determining Optical Flow*, Art. Intell. **17**, 185-203, 1981

[2] A.B.Watson, A.J.Ahumada, Jr., *A Look at Motion in the Frequency Domain*, NASA Tech. Memo. TM-84352, 1983

[3] E.H. Adelson, J.R. Bergen, *Spatiotemporal Energy Models for the Perception of Motion*, J. Opt. Soc. Am. **A 2**, 284-99, 1985

[4] H.H. Chen, T.S. Huang, *An Algorithm for Matching 2-D Line Segments With Application to Multiple-Object Motion Estimation*, Proc. IEEE Workshop on Comp. Vision, 1987

[5] G.R.Nudd, R.M.Howarth, T.J.Atherton, N.D.Francis, G.J.Vaudin, D.W.Walton, *A Heterogeneous Architecture for Parallel Image Processing*, Proc. IED Conf. Information Technology, Swansea, 1988

[6] R.M.Howarth, N.D.Francis, *Cluster Programming Language: Definition and User Manual*, Research Report RR125, Dept. of Computer Science, University of Warwick, July 1988