Co-design by Parallel Prototyping: Optical-Flow Detection Case Study

M. Fleury, A. C. Downton and A. F. Clark*

1 Introduction

Traditional development of embedded systems separates hardware and software into independent design streams. Alternatively, co-design involves the interaction between the two streams, possibly once selection of the main hardware engine is complete[1]. Unfortunately, if the solution involves high-performance multi-processor architectures co-design requires considerable prior experience of architectural alternatives, topologies and programming environments. In fact, the large number of degrees of freedom in an open-ended co-design problem is equivalent to an NP-complete problem[2].

Recently, system development based on a restriction of choice to pipelined processor farming (PPF) has proved successful for the domain of continuous-flow real-time vision systems[3]. In the systems of interest, data rather than control exigencies frequently dominate. Real-time constraints are soft, conditioned by various throughput, latency, and output-ordering specifications. Using our software tools for rapid prototyping on parallel pipelines, such as high-level software templates[4], deferred design becomes possible whereby considerable algorithmic development occurs before commitment to hardware. A semi-manual partition of the unfamiliar code is achieved based on a static analysis, while subsequent event tracing is helpful in showing how dynamic effects alter an initial performance estimate[5]. Fig. 1 shows our development model. Parallel prototyping can reduce design time by enabling development of multi-algorithm systems in a familiar constrained sequential environment but testing in a faster parallel environment without disturbing the original algorithms. Ideally, a direct mapping from the prototyping pipeline to the target system is available thereby bypassing the constraint problem. Conversely, parts of the algorithm which are intractable to parallelization on a homogeneous machine, for reasons of granularity, can be extracted from the prototyping pipeline and mapped to alternative hardware.

Parallelization of optical-flow (OF) by PPF prototyping has illustrated the need to delay design decisions. Initial implementation of an optical-flow algorithm selected by criteria of accuracy and suitability for eventual microchip implementation has revealed problems with a simple-minded approach. In fact, the question of accuracy is not clear cut and there are problems with high communication bandwidths if a video-rate solution were to be needed.

A 2-D OF dense vector field can be extracted from a sequence of video frames by processing of the image irradiance over time, without the difficulty of matching corresponding features in the 3-D scene structure. Finding the per-pixel OF is akin to forming a pre-attentive image

*The authors are with the Dept. of Electronic Systems Engineering, University of Essex

in early human vision[6]. Spatiotemporal filtering inevitably leads to lengthy processing times especially if direct mimicry of the human vision system is attempted. However, some potential applications of OF, such as video-compression, surveillance, and moving-target indication require video-rate responses. Video compression is the most likely high-volume area for VLSI applications, does not require a 3-D motion field, and has a variety of scene types. Increased accuracy will reduce the residual error from predictive coding of intensity levels, but high accuracy is not essential. Subsequent motion segmentation into irregular regions[7] may be a feature of the proposed very-low bit-rate generic codec, MPEG-4, which is targeted at mobile, low-power, devices.

Four OF routines have been parallelized based on public-domain code from a recent survey. The parallelizations illustrate the considerable time savings that may be made over using a workstation as the test environment.

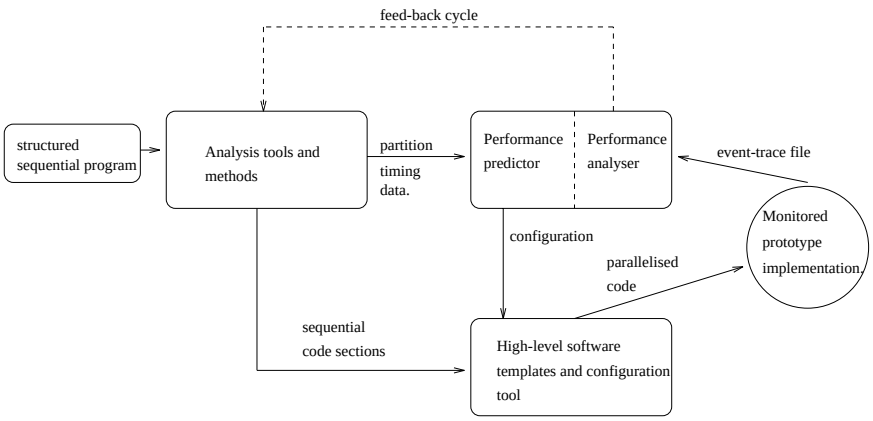

Figure 1: The PPF Design Cycle

2 Optical Flow Methods

In a recent survey[8], nine OF methods from four categories (differential, energy, phase, and correlation) were quantitatively surveyed with regard to accuracy. Authenticated code and a set of standard frame sequences can be found by anonymous `ftp` to `csd.uwo.ca` in `/pub/vision/`. Though two methods, the Lucas-Kanade (LK) gradient-based method[9, 10], and the Fleet-Jepson (FJ) phase-based method[11] emerged as the most accurate, the main contribution of the survey may have been to draw attention to the question of implementation. Techniques can be improved by:

- pre-processing by temporal smoothing over a large number of frames, as well as the usual Gaussian smoothing and possibly demodulation over larger spatial neighbourhoods;
- choice of window size in search areas and parameters in relaxation routines; and
- in post-processing through choice of threshold or confidence measure, which should be correlated with known errors found from synthetic image sequences.

With suitable implementational additions OF methods are more closely characterised as multi-algorithm systems rather than single algorithm routines, as would be more apparent if subsequent processing stages such as motion segmentation were to be included. Analysis of the JF method revealed at least six processing stages (Fig. 2). Clustering of estimates in order to detect multiple motions at the same locality is not performed.

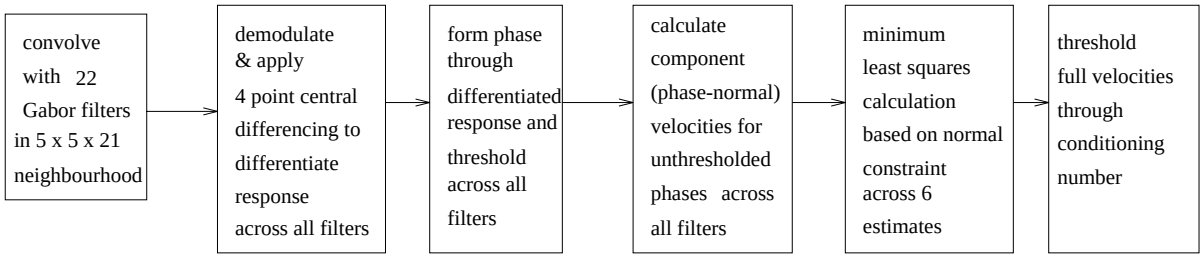

Figure 2: The JF Processing Pipeline

Judgement should be exercised in assessing the results of black-box testing of OF methods as potential applications of the methods may vary. In the survey[8], motions were always less than five pixels and corruption of the recorded images was limited which favoured the differential methods tested. Only one method, that of Anandan (AN)[12], was tested in the survey in a multiresolution mode, advisable for large motions to avoid temporal aliasing. The SI method, also using correlation, was modified, possibly detrimentally, to increase the search area without using image pyramids, which require a linked-list data-structure. The correlation methods were implemented in the survey with a two or three frame search extent. This makes such methods more appropriate for real-time work as there is reduced delay, and reduced bandwidth. Some applications, (say) using time-lapsed imaging, also have reduced data availability. However, correlation accuracy results might easily improve if temporal smoothing were to be applied (even though temporal differentiation is not applied), rather than compensating with location uncertainty distributions[13] as in the SI method.

In work by Scott[14], there is an example of a street scene with a 'bus in which detection is shown to have failed at sites of specularity on the chassis, occlusion at the edge of the 'bus, and through false signals caused by 'video shatter'. Multiple windows and robust statistics[15] are of value in localised resolution of occlusion boundaries, and global detection[16] is possible by building in discrimination against flow estimates made at edges. Specific methods are applied to deal with particular image types. Energy methods[17] will cope with randomly-patterned surfaces, i.e. too much image structure, in a way that methods relying on conservation of image irradiance over the interlude of a frame will not. Phase-based methods are immune to contrast variance in a way that energy-based methods are not. Both energy and phase-based methods will fail if the complex Gabor detection filters are not tuned to the spatiotemporal frequency of interest.

In short, there are considerable degrees of freedom in algorithmic choice which no one survey can provide guidance on. The variables appear to be:

- the method of implementation in support of the basic algorithm;

- the number of frames available;
- the accuracy of algorithm;
- the density of measurements required by the application;
- the camera viewing arrangements;
- the type of motion of objects in imaged scene; and
- the image structure present, including noise.

Bespoke 'ideal' parallelizations of OF algorithms may take time to produce and in the end lack the necessary flexibility. Alternatively, a generic scheme to quickly parallelise a number of algorithms is more cost effective.

3 Computational Structures for Optical Flow

A number of previous generic algorithmic structures have been devised. Anandan[12] found five components in any hierarchical framework consisting of a spatial-frequency decomposition method; a match criterion at each scale; a control strategy to combine results across scales; a confidence measure to select reliable results; and a method of propagating reliable results. Any one component has a variety of algorithmic possibilities. Singh[18] develops a two-step structure involving: the location of a conserved quantity such as intensity; and the assessment of neighbourhood information (through a confidence test).

Fig. 3 is our generic implementation structure based on the extended stages of pre- and post-processing in recent algorithms. The structure is a pipeline, which provides multiple opportunities for parallelization, involving the farming out of work in various ways as anticipated by PPF. The value of a predetermined generic structure is that omissions from algorithms perhaps developed in earlier epochs can be remedied. Notice that some extant algorithms concatenate stages of the generic structure, for instance Sobey and Srinivasan[19] merge spatial filtering into their basic algorithm which is a differential technique.

Some algorithmic components within the pipeline are not amenable to parallelization on medium-grain hardware. Relaxation methods have been explored extensively[20] as a way of propagating results stemming from work by Horn and Schunk[21]. Repeated synchronization and transfer of data within each pixel neighbourhood is necessary making the algorithms particularly appropriate for SIMD processors[22] or the more recent embodiment of SIMD principles in VLSI. Interpolation of detected flow vectors is a viable alternative, though requiring global data access. Genetic algorithms adapted to asynchronous processing may be an alternative method of optimisation that is amenable to medium-grained processing[23].

If an algorithm component is not immediately parallelizable it can be masked within the pipeline structure by a stage consisting of a single processor. Another phenomenon that occurs in OF methods is imbalance in processing requirements between the stages within the method. Again the pipeline allows a short-lived stage to be treated as a centralized stage for the purpose of prototyping. No mapping will be ideal as the enhanced power of recent processors forming the components of general-purpose machines makes them inefficient for short-lived activity.

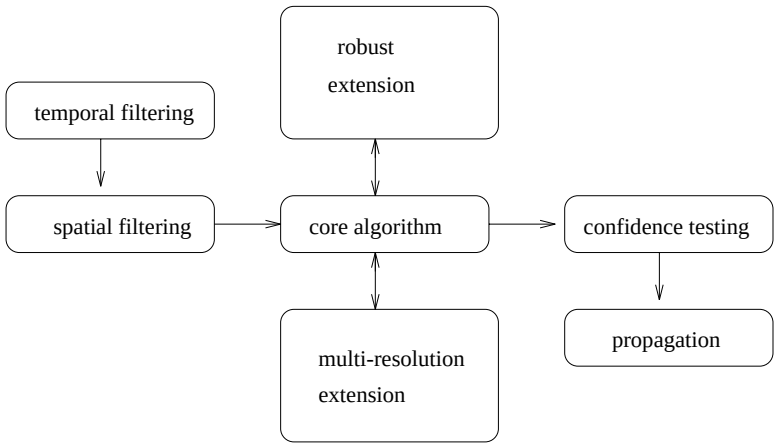

Figure 3: Pipelined Optical-Flow System

4 Transferring to a Parallel Pipeline

We used a medium-grained message-passing parallel processor operating in single-user mode, the Transtech Paramid, equipped with eight superscalar 4-way vector-processing i860s[24], running at 50 MHz. Matching T805 transputer communication coprocessors, link speed 20 Mb/s, connect the i860s in an electronically reconfigurable network. The Paramid is envisaged as an intermediate system from which alternative versions of basic algorithms are transferred to a target system. Therefore, the Paramid plays the same rôle as other dedicated co-design systems such as the COSYMA environment or Edwards' and Forest's system[25]. An example co-design process for two target systems is shown in Fig. 4, where the FPGA is introduced to accelerate part of the code by hardware means. If the prototyping parallel processor is also the target then mapping one would be the identity mapping.

A program profiler, `Quantify`[26], was utilised to seek a static partition of the sequential code. By using counter-based object-code instrumentation `Quantify` provides timings independent of system load. `Quantify` had to be applied with caution to the OF code as the sequential code was the result of consolidation or augmentation of existing research code. The method works well on code which has been written in a homogeneous and structured manner[27].

5 Transferring the LK Method

The LK method at first sight would appear to be a suitable candidate for parallelisation with a view to meeting some real-time criterion. Benign features include good relative accuracy and use of localised processing. The processing at each stage is regular and deterministic as it should not be necessary to calculate eigenvalues in the confidence test. As the LK method relies on numerical differentiation it is essential to remove high-frequency components which otherwise produce erratic response.

Fig. 5 is part of a function call graph for a run of the LK method on the standard 'Yosemite' sequence, with effectively only the control loop function, `main`, missing. By the thickness of the lines the call time is apparent. Fig. 6 gives quantitative data on the call times

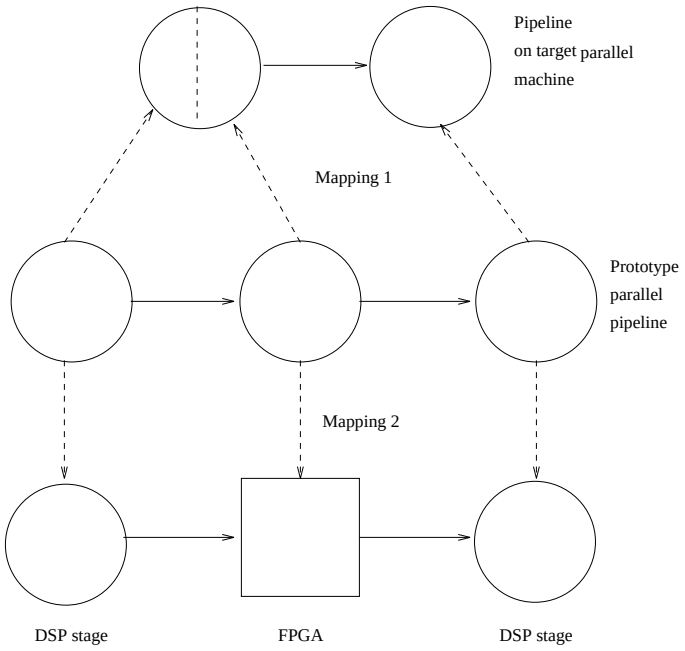

Figure 4: The Co-Design Process

for the major functions. The two main processing stages are approximately balanced in time, suggesting a two-stage pipeline. Having decided on a partition, the sequential code is slotted more or less without alteration into two instances of a data-farming template. The next step to rapid transfer to the processing pipeline is the design of message data structures. In Fig. 7, the processing cycle for the LK algorithm is summarized, though note that processing stages two to four are placed in stage two of the pipeline. To combine numerical differentiation and calculation of output motion vectors in one pipeline stage an intermediate data structure was needed (Fig. 8).

If the two pipelines stages can be decoupled the LK algorithm becomes an asynchronous pipeline in regard to image rows. Demand-based scheduling in stage one will result in out-of-order row arrivals to stage two. After spatiotemporal smoothing, seven rows from five frames per message are needed to perform 4 point central differencing before a linear least-squares estimate is made on a 5×5 neighbourhood, for a single row of output. In the algorithm of Fig. 9 for stage two, a linked-list data structure enables a message to be assembled in the correct order and storage nodes to be discarded when a message is transferred to a circular message buffer. Worker requests are 'fairly' demultiplexed, though in alternation with the servicing of stage one arrivals.

6 Timing Results

Using 15 frames of the 252×316 sized 'Yosemite' sequence, a SPARCstation 1+ takes 154.9s to derive the OF field by the LK method using the `gcc` compiler without optimisation and 60.3 s with full level 3 optimisation (from `Quantify's` count). After applying an optimising compiler, an i860 took 39.5 s for the same task. With a two-stage pipeline utilising two active

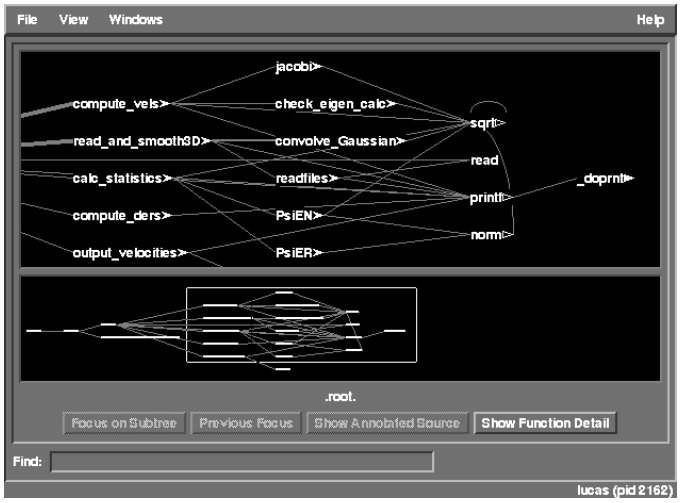

Figure 5: Call Graph for the LK Sequential Code

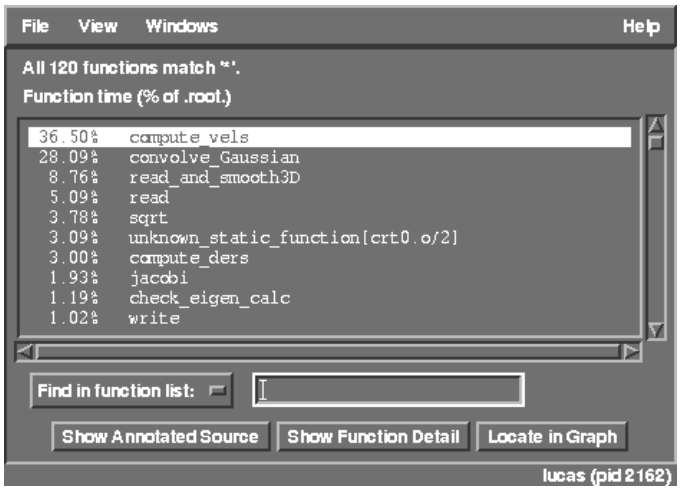

Figure 6: Timing Breakdown for LK Sequential Code Functions

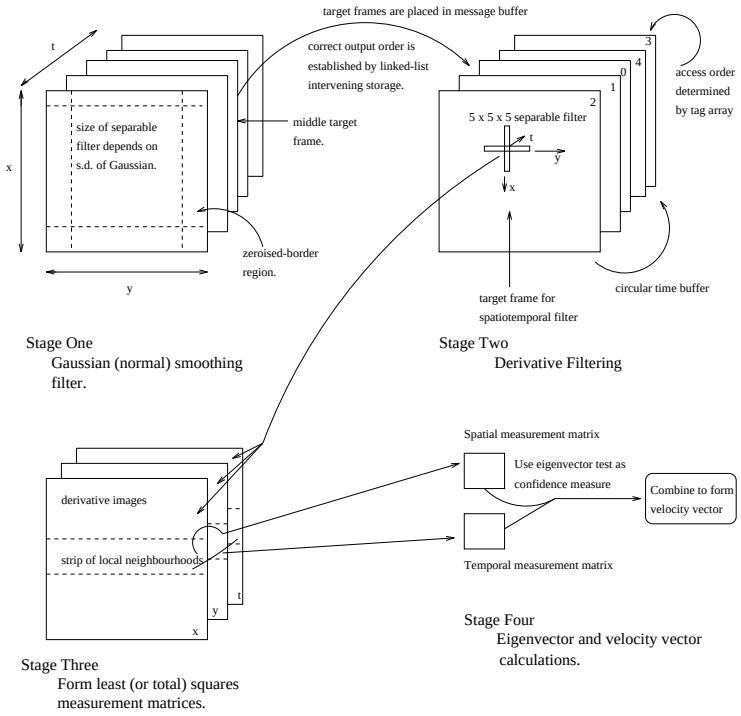

Figure 7: Data Processing in the LK Method

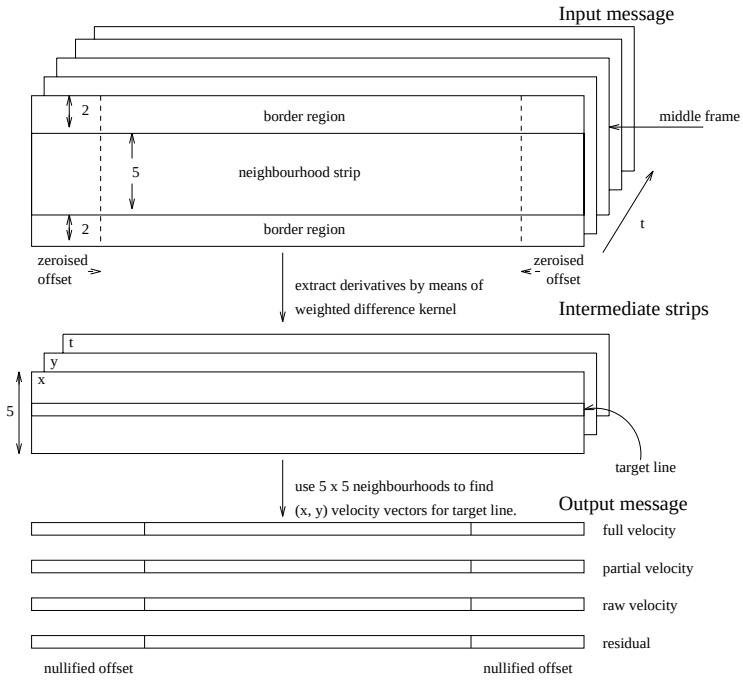

Figure 8: Pipeline Second-Stage Message Structure

processors in each stage took a minimum of 10.1 s, though there is considerable variation according to relative message sizes. This would suggest an efficient implementation, but in fact increasing the number of processors resulted in no improvement in timings. With a single i860 performance is slowed by disc access over a SCSI link to a swap file. The parallel solution did not generate swap files and with two processors on each stage did not saturate the bandwidth of the interconnect. The messaging requirements, evident from Section 5, even with packetising by the virtual channel system, did saturate the system when more processors were added.

The data requirements for the AN method are reduced to two frames though two levels of an image pyramid were required. An 11×11 search area is required for each pixel. Table 1 records timings as additional stages of processing were incrementally introduced into a single farm pipeline. Considerable savings over workstation timings are possible. The SI method was also parallelised as a single stage pipeline but only for the correlation stage and some preparatory calculations for confidence testing. Three frames are needed for any estimate with a minimum 21×21 search region for any one pixel's estimate. One iteration of the subsequent relaxation algorithm was equivalent in time to the complete correlation timings with limited improvement in accuracy. Again, Table 2, considerable advantage over workstation timings were found as well as some potential scalability.

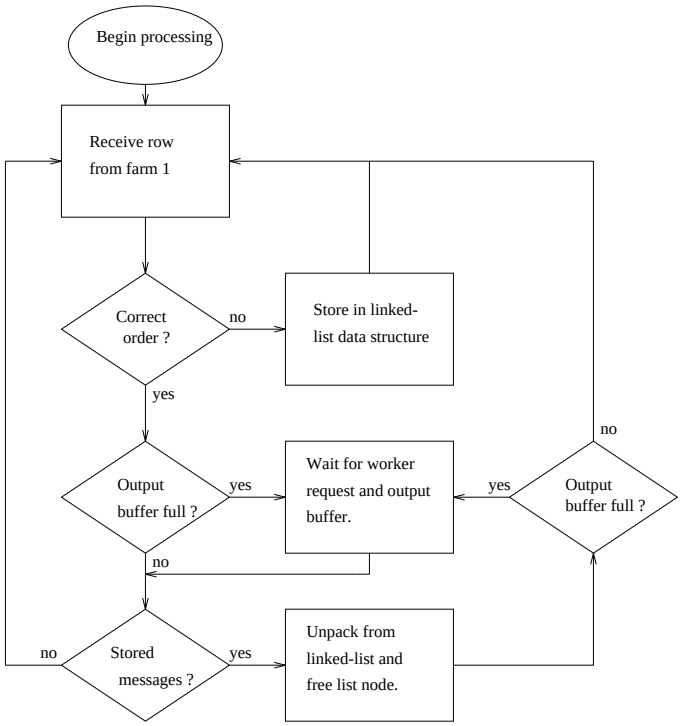

Figure 9: Buffering Out-of-Order Arrivals

Preliminary FJ timings, Table 3, were taken for a full run on the workstation and for the parallelized Gabor filtering stage on the Paramid machine. A timing analysis by means of `Quantify` showed that almost 80% of processing time is spent at the filtering stage. The FJ data demands are large consisting of at least fifteen frames input and twenty-two intermediate

sets of five frames each. In fact, the Paramid system proved inadequate to the task of processing larger images. Were larger speeds expected in the test image sequence then a further set of filters would be needed to cover possible flow directions. It is self-evident that using the optimisation features of any compiler is essential for a processing task similar to phase-based OF processing. It is also evident that processing demands from current OF methods are beyond routine application of general-purpose hardware. The FJ algorithm is not alone, a Sparc-10 workstation is reported to have taken about $3\frac{1}{2}$ hours using the algorithm of Ong and Spann[15] and 30 minutes for the AN algorithm on a 256×256 sized image sequence.

	SPARC station 1+		Worker i860s								
Size	gcc	opt. 3	1	2(a)	2(b)	2(c)	4(a)	4(b)	4(c)	4(d)	4(e)
100×100	171.3	103.6	23.5	15.2	13.2	12.3	12.2	9.4	8.2	3.8	0.3
150×150	458.8	282.5	60.0	34.3	29.7	27.8	27.5	20.9	18.2	8.4	0.8
252×316	1445.5	895.3	192.2	122.7	107.2	100.1	98.1	75.6	65.2	29.8	2.9

Table 1: AN Method Timings (s) Parallelizing: (a) flow vectors (b) flow vectors and confidence measures (c) with coarse-level calculations, and (d) relaxation (e) pyramid construction overheads.

	SPARCstation 1+		Worker i860s		
Size	gcc	opt. 3	1	2	4
100×100	398.4	113.4	31.6	16.1	8.2
150×150	1051.8	301.7	83.8	42.5	21.4
252×316	4269.3	1224.3	339.7	171.3	86.4

Table 2: SI Method Timings (s) for Correlation Processing

	SPARCstation 1+		Worker i860s		
Size	gcc	opt. 3	1	2	4
100×100	834.1	257.9	152.7	105.8	51.4
150×150	2476.7	775.3	353.2	165.5	114.9
252×316	10500.1	3392.6	-	-	-

Table 3: Preliminary FJ Method Timings (s)

7 Conclusion

Current OF routines have achieved greater accuracy over larger speeds by improvements in implementation. Unfortunately, the improvements require the processing of more image frames or larger spatial regions. General-purpose (gp) multicomputers can be deployed to reduce the timings recorded on workstations. Even the most favourable method in terms of accuracy and speed, the LK method, is not scalable on a gp machine and because of

its data requirements would probably not transfer to VLSI. To maintain adequate levels of accuracy at least twelve frames are probably needed for small motions for the LK method. Individual applications may tolerate reduced temporal smoothing but note that temporal aliasing commonly occurs in cinematography and 'stop-and-shoot' test sequences. In the work of Fleet and Langley[28], by compromising somewhat on accuracy, a recursive differentiation filter can reduce processing and data requirements. Recursive filtering may be more suited to parallel machines with a global address space. Gp machines are suitable for reducing the time taken either in algorithmic development work or in repeatedly finding the OF field over many sets of image sequences. The four methods parallelised herein are not unusual amongst OF methods in being amenable to data-farming. Parallelising a number of routines in a systematic manner is possible if a generalised framework is available such as is provided by the PPF methodology and knowledge of the common structure to OF methods. The advantage of parallel pipelines may be that independent stages of the pipeline can at a later time be transferred to more suitable hardware in the manner of co-design if the balance within the pipeline is maintained.

Acknowledgement

This work is being carried out under EPSRC research contract GR/K40277 'Parallel software tools for embedded signal-processing applications' as part of the EPSRC Portable Software Tools for Parallel Architectures directed programme.

References

[1] W. H. Wolf. Hardware-software co-design of embedded systems. *Proceedings of IEEE*, 82(7):967–989, July 1994.

[2] R. Ernst, J. Henkel, Th. Benner, W. Ye, D. Herrmann, and M. Trawny. The COSYMA environment for hardware/software cosynthesis of samll embedded systems. *Microprocessors and Microsystems*, 20:159–166, 1996.

[3] A. C. Downton, R. W. S. Tregidgo, and A. Çuhadar. Top-down structured parallelisation of embedded image processing applications. *IEE Proceedings, Part I (Vision, Image and Signal Processing)*, 141(6):431–437, December 1994.

[4] M. Fleury, H. P. Sava, A. C. Downton, and A. F. Clark. Designing and instrumenting a software template for embedded parallel systems. In *UK Parallel '96*, pages 163–180. Springer, London, 1996.

[5] M. Fleury, A. C. Downton, and A. F. Clark. Modelling pipelines for embedded parallel processor system design. *Electronic Letters*, 33(22):1852–1853, 1997.

[6] E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America A*, 2(7):284–298, February 1985.

[7] M. T. Orchard. Predictive motion-field segmentation for image sequence coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 3(1):54–70, February 1993.

[8] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[9] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In $7^{\underline{th}}$ *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[10] E. P. Simoncelli, E. H. Adelson, and D. J. Heeger. Probability distributions of optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 310–315, 1991.

[11] D. J. Fleet and A. D. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5(1):77–104, 1990.

[12] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–310, 1989.

[13] A. M. Waxman, J. Wu, and F. Bergholm. Convected activation profiles and the measurement of visual motion. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 717–723, 1988.

[14] G. L. Scott. 'Four-line' method of locally estimating optic flow. *Image and Vision Computing*, 5(2):67–72, May 1987.

[15] E. P. Ong and M. Spann. Robust multiresolution computation of optical flow. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 1938–1941, 1996.

[16] H-H Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):565–593, September 1986.

[17] D. J. Heeger. Model for the extraction of image flow. *Journal of the Optical Society of America A*, 4(8):1455–1470, August 1987.

[18] A. Singh. An estimation-theoretic framework for image-flow computation. In $3^{\underline{rd}}$ *International Conference on Computer Vision*, pages 168–177, 1990.

[19] P. Sobey and M. V. Srinivasan. Measurement of optical flow by a generalized gradient scheme. *Journal of the Optical Society of America A*, 8(9):1488–1498, September 1991.

[20] M. A. Snyder. On the mathematical foundations of smoothness constraints for the determination of optical flow and for surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1105–1114, November 1991.

[21] B. K. P. Horn and B. G. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[22] T. J. Fountain. A survey of bit-serial array processor circuits. In M. J. B. Duff, editor, *Computing Structures for Image Processing*, pages 1–14. Academic Press, London, 1983.

[23] I. Turton, S. Openshaw, and G. Diplock. Some geographical applications of genetic programming on the Cray T3D supercomputer. In *UK Parallel '96*, pages 135–150. Springer, London, 1996.

[24] M. Atkins. Performance and the i860 microprocessor. *IEEE Micro*, pages 24–27,72–78, October 1991.

[25] M. Edwards and J. Forrest. A practical hardware architecture to support software acceleration. *Microprocessors and Microsystems*, 20:167–174, 1997.

[26] Pure Software Inc., 1309 South Mary Ave., Sunnyval. CA. *Quantify User's Guide*, 1992.

[27] M. Fleury, A. C. Downton, and A. F. Clark. Pipelined parallelization of face recognition. *Machine Vision and Applications*, 1997. Submitted for publication.

[28] D. J. Fleet and K. Langley. Recursive filters for optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):61–67, January 1995.