[JM92a] David Jones and Jitendra Malik. A computational framework for determining stereo correspondence from a set of linear spatial filters. In *Procceeding of the Second European Conference on Computer Vision*, 1992.

[JM92b] David Jones and Jitendra Malik. Determining three-dimensional shape from orientation and spatial frequency disparities. In *Procceeding of the Second European Conference on Computer Vision*, 1992.

[Koe88] J.J. Koenderink. Operational significance of receptive field assemblies. *Biological Cybernetics*, 58:163–171, 1988.

[LB90] Vincent Di Lollo and Walter Bischof. Yes, there is a half-cycle displacement limit for directional motion detection. *Vision Research*, 31(4):763–765, 1990.

[LK81] B. Lucas and T. Kanade. An iterative image restoration technique with an application to stereo vision. In *Procceeding of the DARPA IU Workshop*, pages 121–130, 1981.

[Mad59] A. Madansky. The fitting of straight lines when both variables are subject to error. *J. Amer. Statist. Assoc.*, 54:173–205, 1959.

[MP90] Jitendra Malik and Pietro Perona. Preattentive texture discrimination with early vision mechanisms. *Journal of the Optical Society of America A*, 7(5):923–932, 1990.

[Nag87] Hans-Hellmut Nagel. On the estimation of optical flow: relations between different approaches and some new results. *Artificial Intelligence*, (33), 1987.

[Pea01] K. Pearson. On lines and planes of closest fit to points in space. *Philos. Mag.*, 2:559–572, 1901.

[SAH91] Ero Simoncelli, Edward Adelson, and David Heeger. Probability distributions of optical flow. *International Journal of Computer Vision*, 6(2):133–145, 1991.

[Spr69] P. Sprent. *Models in Regression and Related Topics.* Methuen, London, 1969.

[Sri90] M.V. Srinivasan. Generalized gradient schemes for the measurement of two-dimensional image motion. *Biological Cybernetics*, 63:421–431, 1990.

[TP84] O. Tretiak and L. Pastor. Velocity estimation from image sequences with second order differential operators. In *Procceeding of the International Conference on Pattern Recognition*, Montreal, 1984.

[UGVT88] S. Uras, F. Girosi, A. Verri, and V. Torre. A computational approach to motion perception. *Biological Cybernetics*, 60:79–87, 1988.

[VGT90] A. Verri, F. Girosi, and V. Torre. Differential techniques for optical flow. *Journal of the Optical Society of America A*, 5:912–922, 1990.

[VV91] Sabine VanHuffel and Joos Vandewalle. *The Total Least Squares Problem: Computational Aspects and Analysis.* Frontiers in Applied Mathematics. SIAM, Philadelphia, 1991.

[You85] R.A. Young. The gaussian derivative theory of spatial vision: Analysis of cortical cell receptive field line-weighting profiles. Technical Report GMR-4920, General Motors Research, 1985.

Acknowledgements

This research was partially supported by Texas Instruments, NSF Presidential Young Investigator Grant IRI-8957274 to J.M., Xerox and the PATH project MOU 83. NSF Infrastructure Grant number CDA-8722788 supported the use of the CM-5. We wish to thank D. Fleet and J. Barron for providing the sequences and true flow, and A. Verri for helpful discussions.

References

[BA92] Jezekiel Ben-Arie. Muti-dimensional linear lattice for fourier and gabor transforms, multiple-scale gaussian filtering, and edge detection. In *Neural Networks for Perception*, volume 1, chapter 11.2. Academic Press, 1992.

[BAK91] Roberto Battiti, Edoardo Amaldi, and Christof Koch. Computing optical flow across multiple scales: An adaptive coarse-to-fine strategy. *International Journal of Computer Vision*, 6(2):133–145, 1991.

[BC90] Oliver Braddick and Robert Cleary. Is there a half-cycle displacement limit for directional motion detection? *Vision Research*, 31(4):761–762, 1990.

[BFB92] John Barron, David Fleet, and S.S. Beauchemin. Performance of optical flow techniques. Technical Report 299, University of Western Ontario, 1992.

[BFBB92] John Barron, David Fleet, S.S. Beauchemin, and T. Burkitt. Performance of optical flow techniques. In *Proceedings of the IEEE CVPR*, pages 236–242, Champaign, IL., 1992.

[Can86] John Canny. A computational approach to edge detection. *IEEE trans. PAMI*, 8:679–698, 1986.

[CB90] Robert Cleary and Oliver Braddick. Directional discrimination for band-pass filtered random dot kinematograms. *Vision Research*, 30:303–316, 1990.

[CJ85] J.J. Chang and B. Julesz. Cooperative and non-cooperative processes of apparent movement of rnadom-dot cinematograms. *Spatial Vision*, 1:39–45, 1985.

[CV90] M. Campani and A. Verri. Computing optical flow from an overconstrained system of linear algebraic equations. In *Proceedings of the 3rd International Conference on Computer Vision*, pages 22–26, 1990.

[DH73] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, Chichester, Brisbane, Toronto, Singapore, 1973.

[FJ90] David Fleet and Allan Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5:77–104, 1990.

[FT79] Claude Fennema and William Thompson. Velocity determination in scenes containing several moving objects. *International Journal of Computer Vision*, 9:301–315, 1979.

[GV80] G.H. Golub and C.F. VanLoan. An analysis of the total least squares problem. *SIAM Journal Numer. Anal.*, 17:883–893, 1980.

[Hee87] David Heeger. Optical flow from spatiotemporal filters. In *Proceedings of the First International Conference on Computer Vision*, pages 181–190, 1987.

[HS81] B.K.P. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, (17):185–203, 1981.

[Hub81] Peter J. Huber. *Robust Statistics.* Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, 1981.

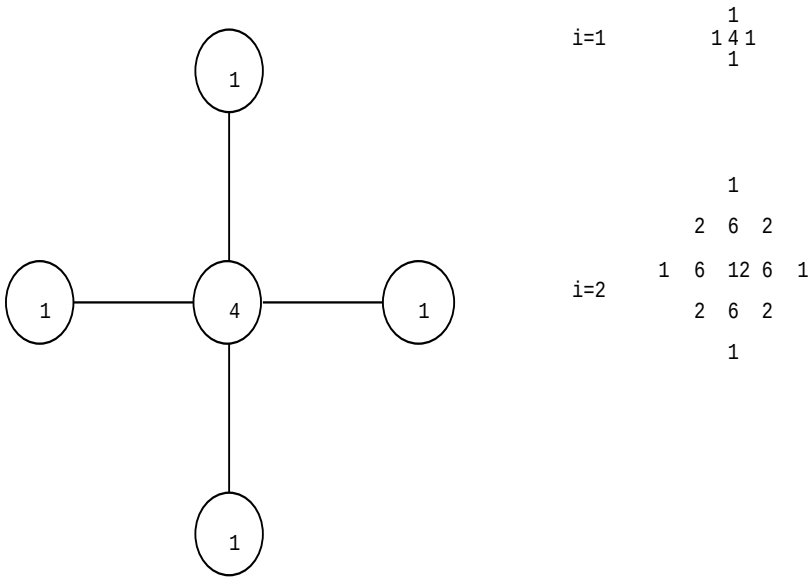

Figure 10: Simple two-dimensional connections between processing elements for computing a Gaussian convolution of the local image. The weights of the weighted sum are inside each processor circle. The result of applying this weighted sum are shown on the right for two iterations. After normalization this sequence converges to a Gaussian distribution.

The total number of step required by this processes is $4\sigma_0^{2(s-1)}$ where σ_0 is the base scale and s is the number of different scale groups. The amount of data values which must be passed between processors on each iteration is $4n/\sqrt{N}$ where the image is of size n^2 using N processors.

6.2 Implementation

The above algorithm was implemented on a CM-5, 128 processor machine. The images are divided up into 128 regions and sent to the processors. The two dimensional Gaussian operation outlined above was performed. At each step the processors performed the operation on the interior of their own patch. The borders of each patch needed to be shared with the neighboring processors in order to complete the step. For an image of dimension $n \times n$, this required $4 \times n/8$ elements to be passed per step per processor.

To obtain the image convolved with a geometric series of Gaussians, (i.e. with Gaussians of standard deviations $\sigma = 1, 2, 4, 8, 16$ pixels), the simple procedure was performed $4, 16, 64, 256$, and 1024 times. After the smoothing iterations, finite differences were performed in order to obtain approximations to the result of convolving the original image with the derivatives of Gaussians of various sizes. The final result is a series of numbers for each pixels. These numbers represent a series of new images which are the outcome of convolving the original image with a series of linear filters.

The CM-5 is not an ideal machine to perform the convolution method outlined above because it requires much communication between processors. The parallel C we used on the CM-5 had a high overhead for communication between some processors, taking many cycles per byte of information passed. Our purpose however was to demonstrate the parallizabibilty of the algorithm and the speed up experienced when operated in parallel. Convolution of a 128 pixel squared image took about 10 seconds, including I/O. We believe that with specialized hardware real-time implementations are possible.

Sequence	Full Convolutions		Subsampled Filters	
	Avg. Error	Std. Dev.	Avg. Error	Std. Dev.
Translating Tree	0.49	0.35	0.55	0.36
Diverging Tree	3.18	2.50	3.30	2.72
Yosemite	3.43	5.35	3.77	4.83

Figure 9: Comparison of the full implementation where each filter output is computed at each pixel and a subsampling scheme where a filter output is calculated at every 2^n pixels.

6 Multiprocessor Implementation

The algorithm described is massively parallel. Each estimate is formed from a small spatio-temporal window of the motion sequence. The previous results were obtained from a SUN workstation. Processing time was dominated by the convolutions since velocity estimates required only a few simple operations on the convolution results per pixel. A series of 36, 3-d separable convolutions on a 128 pixel square image took about 4 minutes per frame. To examine the speedup possible with a parallel implementation, a parallel version of the algorithm was created for the 128 processor CM5 from Thinking Machines.

6.1 Parallel Convolution

Instead of the linear convolutions with filter kernels that was used on the serial machine, we used the interconnectivity of the processors to perform convolution in a different manner. One way of computing a series of convolutions with Gaussian functions of various size is to make use of the Central Limit Theorem. Specifically, if the processor element at each pixel were to perform a simple manipulation of its own data and the data of its neighboring pixel elements, the resulting distribution after a large number of iterations will be a Gaussian.

For example, suppose we have a one dimensional string of processing nodes, each initially containing a single number. Let each processing node them perform the following operation:

> *Take two times the value contained at this node and add it to the value held at the node to the immediate right and left. Make this sum the new stored value.*

This operation creates a weighted sum of the form $(1, 2, 1)$ centered at each node. At the next iteration, we have the following weighted sum $(1, 4, 6, 4, 1)$. Placing them in an array we have:

$$\begin{array}{ccccccc} & & 1 & 2 & 1 & & \\ & 1 & 4 & 6 & 4 & 1 & \\ 1 & 6 & 15 & 20 & 15 & 6 & 1 \end{array} \tag{16}$$

The result is the even lines of Pascal's triangle which consists of the even binomial coefficients. It is known that these coefficients approximate the Gaussian distribution for large numbers. Specifically, the binomial coefficients of degree n can be approximated by

$$\left(\begin{array}{c} k \\ n \end{array} \right) \simeq C \exp(-2(k - n/2)^2/n) \tag{17}$$

Thus these coefficients are proportional to a Gaussian with standard deviation $\sigma^2 = n/4$. In our algorithm we wish to obtain convolutions with Gaussians of increasing size. We also want derivatives of Gaussian functions. These can be obtained via finite differences. If a processor element takes the difference of the values of its neighbors, it obtains an estimate of the derivative at its location. This can be repeated for higher order derivatives. Other filters can be approximated by similar elementary operations [BA92].

If we normalize the example process such that after the summation the processor divides by four, we can preserve the sum total of the data stored at the nodes. Figure 10 shows a two-dimensional extension of the Gaussian kernel. The right side of the figure shows the impulse response of the resulting filter after one and two iterations. After normalization, this series converges to a Gaussian distribution.