level. This is because a valid edge flow makes a valid edge as shown in the previous section. Our confidence-updating procedures for a horizontal or vertical reference edge are as follows:

Step 1: Make a horizontal (vertical) edge flow graph of 2 level boundary.

Step 2: Select the maximum cost path passing through the reference edge in the edge flow graph.

Step 3: Update the confidence of the reference edge if the cost of the selected path is above the threshold.

Experimental results and conclusion: We implemented our algorithm and the Prager non-directional edge relaxation algorithm [1] and compared their performance, including noise robustness, processing speed and output image quality, with test images of 256 grey-levels. Concerning the noise robustness, our algorithm takes the best cost path in a level 2 boundary with flow constraints.

Table 1: Comparisons of edge detection errors

Error type	Algorithm	Standard deviations of added noise					
		20	40	60	80	100	120
α-error	Yoon-Park	0.01	0.03	0.07	0.16	0.26	0.35
	Prager	0.06	0.32	0.48	0.55	0.63	0.66
β-error	Yoon-Park	0.00	0.01	0.04	0.09	0.13	0.16
	Prager	0.00	0.01	0.03	0.07	0.09	0.10

Table 1 shows the edge detection errors for each algorithm with increasing Gaussian input noise levels. We measured the conditional probability of α-error, *P(no_edge|true_edge)*, and β-error, *P(edge|true_no_edge)*, where *true_edge* and *true_no_edge* are reference decisions in the case of no-noise image. Our algorithm is better than the Prager algorithm in α-error, and similar in β-error. This means that our algorithm is more robust to noisy images than the Prager algorithm. Concerning the processing speed, we used the edge flow graph for finding the optimum path. This

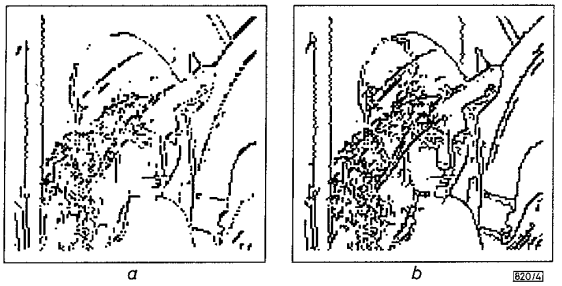

Fig. 4 *Test results for real image*

Lena, 128×128 pixels, 256 grey levels
a Result of the Prager algorithm
b Result of our algorithm

method speeds up the confidence measuring process. Finally, for the output image quality, we avoid unrealisable edge patterns which may arise in non-directional edge relaxation. Fig. 4 shows the test results for a real image. From the above results, we can conclude that our algorithm works better than a conventional non-directional edge relaxation algorithm. This is due to the elimination of unrealisable edges by considering the directional edge flow and finding the best edge path in the edge flow graph.

11 September 1995
Electronics Letters Online No: 19960015

Ki Cheon Yoon and Kyu Ho Park (*Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, 373-1, Kusong-dong, Yusong-gu, Taejon 305-701, Korea*)

References

1 PRAGER, J.M.: 'Extracting and labeling boundary segments in natural scenes', *IEEE Trans. Pattern Anal. Mach. Intell.*, 1980, **2**, (1), pp. 16–27

2 ROSENFELD, A., HUMMEL, R.A., and ZUKER, S.W.: 'Scene labeling by relaxation operations', *IEEE Trans. Syst. Man Cybern.*, 1976, **6**, (6), pp. 420–433

3 HANCOCK, E.R., and KITTLER, J.: 'Edge-labeling using dictionary-based relaxation', *IEEE Trans. Pattern Anal. Mach. Intell.*, 1990, **12**, (2), pp. 165–181

4 CHANG, S.K.: 'Principles of pictorial information system design' (Prentice-Hall Inc., Englewood Cliffs, 1989), pp. 43–47

Packing scheme for mean-filtering of an 8 bit image

Yeong Rak Seong, Tag Gon Kim and Kyu Ho Park

Indexing terms: Filtering and prediction theory, Image processing

Filtering is one of the most well known low-level image processing procedures. In most filtering procedures, the potential capability of an ALU in a processor is not fully used. The authors propose a packed mean filtering scheme. The scheme packs several pixels into a unit and processes them simultaneously. Experiments are held under three distinct machines to evaluate the performance of the scheme. The result shows that the scheme enhances processing speed in all three environments.

Introduction: Recently, microprocessor technology has been growing rapidly, and the size of ALUs has increased. However, in many applications, the increased size is not used fully. A simple example would be using a 32 or 64 bit ALU for processing an eight bit image. In such a case, the number of bits needed to process a pixel is much smaller than the number of bits of an ALU. To increase profitability, several pixels can be packed into a unit, normally a long integer, and processed simultaneously. On the subject of mean filtering [1], this Letter proposes a pixel packing scheme for exploiting parallelism in a single processor. This is very similar to vectorisation in an array computer.

Algorithm: To store intermediate values in the mean filtering of an eight bit pixel, 12 bits are required. For simplicity, this Letter considers the intermediates to be 16 bits. In packed mean filtering, therefore, an eight bit image input is stored in a 16 bit image buffer, then pixels are packed. At this time, the number of pixels packed into a unit is defined by degree of vectorisation (DOV).

$$DOV = \frac{\text{the number of bits of the ALU processing the filtering}}{\text{the number of bits required to process a pixel}}$$

Indeed, DOV represents the ideal speedup by using the scheme. The pixels packed into a unit are processed simultaneously. Finally, the resulting image is reconstructed into an 8 bit image.

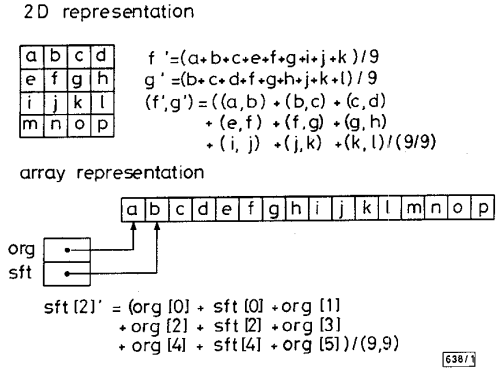

Fig. 1 *Packed mean filtering*

To process a pixel in mean filtering, its neighbouring pixel should be accessed. Therefore, when a set of pixels which are packed into a unit is processed, the neighbouring pixels should also be packed and accessed simultaneously. Fig. 1 gives an exam-

ple. Assume that the underlying machine is a 32 bit computer, then DOV is 2(=32/16). By using the proposed packed mean filtering, two pixels would be processed simultaneously. Consider that pixels 'f' and 'g' are to be packed. Then, the algorithm would access the neighbours of ('f', 'g'): ('a', 'b'), ('b', 'c'), ('c', 'd'), ('e', 'f'), ('g', 'h'), ('i', 'j'), ('j', 'k'), and ('k', 'l'). Such accessing is implemented simply by using two pointers, 'org' and 'sft', pointing to a 32 bit data array. Since the pointers are 32 bit pointers, 2(=DOV) pixels can be accessed simultaneously. Also, since we have already inserted eight bits between two consecutive pixels while transforming an eight bit image to a 16 bit image, simultaneous processing of the pixels does not affect the pixels.

Unfortunately, many RISC processors have a limitation in accessing misaligned data. For example, in the case of a Sparc processor, the value of a pointer addressing a 32 bit data array should be exactly divided by four. Otherwise, if the processor accesses 32 bit data with the pointer, a bus alignment error occurs. Now, consider the proposed packed mean filtering scheme again. In the scheme, pointer 'sft' is shifted by 16 bits from pointer 'org'. Therefore, one of the pointers is not divided by exactly four. Consequently, the algorithm cannot be employed in such processors without any modification. To solve this problem, the Letter uses multiple image buffers rather than two image pointers. Fig. 2 shows the scheme. Now, an 8 bit image is converted into a non-shifted 16 bit image and several shifted 16 bit images. The number of shifted images depends on the number of bits of the underlying processor: one shifted image for a 32 bit machine and two shifted images for a 64 bit or more machine. Now, no pointer is misaligned; thus no bus alignment error will occur.

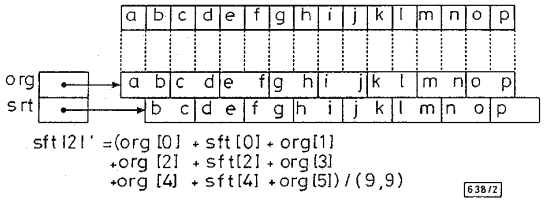

Fig. 2 *Packed mean filtering with multiple image buffers*

Experiment: The performance of the proposed scheme is measured under three distinct environments: (i) a DEC-3000 M600 workstation with a 64 bit Alpha processor [2] and the OSF B1 operating system, (ii) a SUN-Sparc II workstation with a 32 bit Sparc processor [2] and the Solaris operating system, and (iii) a IBMPC-486 compatible with a 32 bit i486 processor [2] and the Linux operating system. Alpha and Sparc processors cannot access misaligned data, while an i486 processor can.

Table 1: Experiment

Environment	Method	T_{mean}	T_{total}
ENV1: DEC3000 M600	scalar	149913	170019
64 bit Alpha processor	vector	37868	98478
OSF1 operating system	speedup	3.96	1.73
ENV2: SUN workstation	scalar	7788000	953000
32 bit Sparc processor	vector	496000	835000
Solaris operating system	speedup	1.57	1.14
ENV3: IBMPC compatible	scalar	336000	477000
32 bit i486 processor	vector	172000	328000
Linux operating system	speedup	1.95	1.45

Time unit: μsec

A 256×256 8 bit image is used in this experiment. Table 1 shows the result. In the table, T_{mean} means the true mean filtering time for which all pixels are filtered, and T_{total} means the total elapsed time, including image buffer allocation and initialisation and output generation. The result shows that both T_{mean} and T_{total} are reduced by the packing scheme. In ENV1 especially, the speedup of T_{mean} nearly reaches DOV, the theoretical upper bound. However, since Alpha is a 64 bit processor, three image buffers are used for the filtering. Hence, the speedup of T_{total} is not very large owing to the buffer allocation and initialisation overhead. In the case of ENV2, the speedup of T_{mean} is not very large compared to ENV1. This may be a result of the differences between processor architectures and programming environments. In the case of ENV3, only one image buffer is used. The difference between T_{mean} and T_{total} is smaller than the two other cases. Also, more speedup can be measured compared to ENV2.

Conclusion: This Letter exploits parallelism in a processor by packing several pixels into a unit and processing them simultaneously in the mean filtering of an eight bit image. In particular, the multiple image buffer scheme is used for the processors which cannot access misaligned data. To show the performance of the proposed scheme, experiments are performed under three distinct environments. From the result, we can conclude that the proposed scheme exploits the parallelism very well.

24 August 1995
Electronics Letters Online No: 19960027

Yeong Rak Seong, Tag Gon Kim and Kyu Ho Park (*Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, 373-1, Kusong-dong, Yusong-gu, Taejon 305-701, Korea*)

References

1 PRATT, W.K.: 'Digital image processing' (John Wiley & Sons, Inc., 1978)
2 TABAK, D.: 'Advanced microprocessors' (McGraw-Hill, Inc., 1995), 2nd Edn.

Acoustic material signature extension

R.D. Weglein

Indexing term: Acoustic microscopy

Measurements and an empirical equation are presented for Δz_M, the maxima separation along the lens axis between focal plane and first AMS peaks in the metrology mode of the acoustic microscope. The new addition Δz_M extends the upper range of the AMS family to higher Rayleigh velocity materials at microwave frequencies.

Introduction: The discovery in 1979 that a wide-angle acoustic microscope lens could provide accurate Rayleigh velocity information on material [1] and layered structure surfaces [2] led to an operational mode that would complement the instrument's already well established high resolution imaging capability. This microacoustic metrology mode [3] ushered in the area of nondestructive material characterisation on a microscopic scale as well as other applications in the then expanding field of small-scale nondestructive testing.

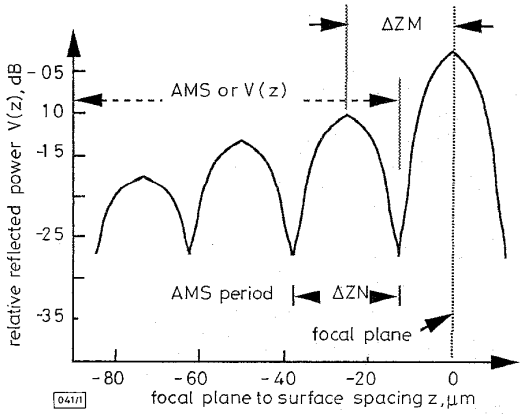

Fig. 1 *Typical acoustic material signature (AMS) or V(z) at 370 MHz*

The two portions of the AMS are the multi-period Δz_N, measured along the axial lens travel between nulls or minima and Δz_M the separation of the first AMS maximum from the focal plane peak