IMAGE UNDERSTANDING

Proceedings of a Workshop held at Pittsburgh, Pennsylvania November 14-15, 1978

Sponsored by the
Defense Advanced Research Projects Agency

Science Applications, Inc.
Report Number SAI-79-814-WA
Lee S. Baumann
Workshop Organizer and
Proceedings Editor

This report was supported by
the Defense Advanced Research
Projects Agency under DARPA
Order No. 3456, contract No. MDA903-78-C-0095
monitored by the
Defense Supply Service, Washington, D.C.

ROBUST PICTURE PROCESSING OPERATORS AND THEIR IMPLEMENTATION AS CIRCUITS

Michael Ian Shamos

Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213

ABSTRACT

The increasing use of satellites in image acquisition has made real-time data compression and summary essential. To reduce bandwidth and alleviate the load on land-based computers it is desirable to perform as much picture processing as possible via LSI circuitry aboard the satellite. Such circuits must be able to deal with a wide variety of images and must exhibit a high degree of reliability. In this paper we use some results from the theory of selection networks to produce a family of robust image smoothing operators suitable for LSI implementation. The circuits are (1) decomposable into small functional units, (2) easily testable, and (3) statistically insensitive to spikes or noise in the data.

1. Introduction

(The entire problem treated in this paper was suggested by Prof. Raj Reddy.) One technical difficulty in current image processing is that resolutions are so high that we literally are unable to see the forest for the trees. It is important to be able to "defocus" minute details to become aware of the larger object of which they are a part. A separate problem is to compress or summarize the image to reduce the telecommunications burden. The encoded picture will then be reconstructed on the ground and it is crucial to extract statistics that suffice to perform this task. Our purpose here is to suggest a new method by which this defocusing and compression may be accomplished.

2. Median Smoothing

In what follows we will assume that an "image" consists of a rectangular array of grey-scale intensities. Our operators will operate on n-by-n square submatrices of the image, where n is odd and small (typically n = 3 or 5). The function of the operator is to compute a descriptive statistic of the n^2 pixels on which it acts. Let F(i,j) denote the value of this statistic over the n-by-n submatrix centered at position (i,j) in the original image array I. One method of smoothing the image that is useful for detecting gross objects is to replace each element I(i,j) by F(i,j). (If F were the averaging operator, for example, then this would correspond to taking moving averages.) One may also effect data compression by a factor of n^2 by replacing the entire submatrix centered at I(i,j) by the single value F(i,j) whenever i and j are congruent to (n+1)/2 modulo n. This procedure can be applied recursively to produce a sequence of progressively defocused (blurred) images. For example, if I is a 625-by-625 matrix, then applying this operation once will yield a 25-by-25 matrix and applying it a second time will give a 5-by-5 result.

Which choices for the smoothing operator F are suitable for picture processing applications? It should possess at least the following properties:

a) F should be robust, that is, it should be relatively insensitive to outlying values, or spikes. (These may correspond to bright spots, reflections, or damaged areas on the retina.)

b) F(i,j) should equal at least one of the actual values in the submatrix on which it operates. This condition is imposed because if the submatrix contains parts of two or more objects, we would like F to serve as a descriptor for the object that occupies "most" of the submatrix. For example,

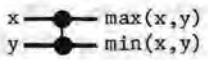
```
1 1 1 3 3
1 1 3 3 3
1 3 3 3 3
3 3 3 3 3
3 3 3 3 3
```

in the subimage at the left we have pieces of two objects, with intensities 1 and 3. We wish F to reflect the fact that the subimage is composed primarily of part of object 3.

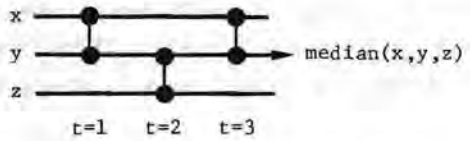
The averaging operator (mean) possesses neither of these two properties, but the median possesses both. In the next section we will attempt to design a circuit for computing the median but will compromise instead on an approximation to the median that is more suitable on several grounds for LSI implementation.

3. Circuits for the median and approximations

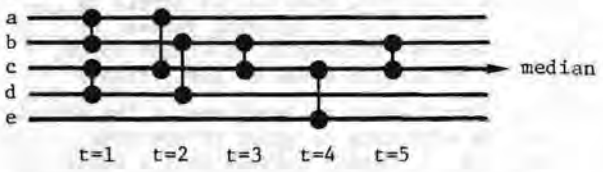
The chief difficulty in computing the median of $M = n^2$ quantities is that it is not an algebraic function of the M inputs and cannot be calculated using arithmetic operations alone -- comparisons are required. It is possible, in fact to compute the median using only comparisons. To show how to implement such algorithms as circuits, we will make use of comparator modules and selection networks, as described in [1]. A comparator module is a device with two input lines, x and y, and two output lines, as shown below. The input signals are



compared and the larger is routed to the upper output line, while the smaller appears at the lower output line. In our diagrams of comparator networks, signals will be assumed to enter from the left and exit at the right. The following network finds the median of three inputs using three comparators and a time delay of three:



(The above circuit actually sorts its inputs.) It may not be readily apparent, but the next network finds the median of five inputs:



The median-of-5 network exploits parallelism during the first two time steps to achieve an overall delay of five with seven comparators. It is shown in [1] that the number of comparators cannot be reduced. We have shown by exhaustion that a delay of five is optimal for comparator networks with fanout one.

There are analogous networks for larger sets of inputs but they become progressively more complex and difficult to design. We do not know how to construct networks that are optimal either with respect to time delay or number of comparators for any but the smallest values of M. Furthermore, the structure of near-optimal circuits is highly irregular and not readily decomposable into simple functional units. A problem that looms larger, however, is that of testability. Once a network is constructed, either theoretically or in practice, how can we verify that it works? If each of the M inputs can assume any of C possible distinct values, it would seem that C^M separate tests are required. For a circuit consisting solely of comparators, through, it suffices to verify its correctness when each input is restricted to be either zero or one. This result is known as the 0-1 Principle [1] and it reduces the number of tests required to just 2^M. While this is a significant improvement, even if we are able to design a median network for 5-by-5 submatrices, verifying all 2^{25} possible binary inputs would be out of the question. To circumvent this difficulty, we will explore an alternative to the exact median which has excellent statistical properties, is decomposable, and is easily tested.

To obtain an approximation to the median we will generalize an idea due to Tukey [2]. For M = 9, let us compute p = median(a,b,c), q = median(d,e,f), and r = median(g,h,i). Now let s = median(p,q,r), that is, the median of the medians. If we implement this computation via a comparator network, it is easy to see that p,q, and r can all be found in parallel in three time steps using nine comparators by replicating the median-of-3 circuit at the left three times. The quantity s can then be found with three more comparators and three additional time steps by using a fourth copy of this circuit in an elegant cascade arrangement. The total number of comparators is 12 and the time delay is six. (The number of comparators can be reduced to nine by re-using one of the first three median circuits.) For M = 25 a similar partitioning into medians of five gives a circuit with 35 comparators and a delay of 10 that can be tested by trying only $5 \cdot 2^5 = 160$ different inputs as opposed to 2^{25}.

The cascade median circuit can be generalized directly for arbitrary odd values of n, the number of tests required being $n2^n$ for n^2 inputs. However, it must be emphasized that these circuits do not compute the median, but only some approximation to the median. We will now investigate how good this approximation is. Let A_n denote the cascade median as found above. If n = 3, then we are trying to find the median of nine elements, that is, the element that has rank five. It is shown in [2] that if all 9! permutations of the inputs are equally likely then A_3 is the exact median (rank 5) with probability 4/7, or approximately 0.571. A_3 will have rank four or rank six with equal probabilities 3/14. Determining the distribution of A_n, even under the assumption of equal probability for each permutation (an assumption that can be relaxed somewhat), is a difficult combinatorial problem. For n = 5 (25 elements) it was easier to obtain the distribution by simulating 100,000 cases than by attempting an exact calculation. The results of the simulation are given below. The exact median has rank 13 out of 25.

P(rank = 9)	= P(rank = 17)	≈	0.0052
P(rank = 10)	= P(rank = 16)	≈	0.0313
P(rank = 11)	= P(rank = 15)	≈	0.1023
P(rank = 12)	= P(rank = 14)	≈	0.2162
P(rank = 13)		≈	0.2900

Distribution of A_5 (obtained by simulation)

Thus A_5 is the exact median with probability 0.29 and has rank that is within one of the correct median with probability > 0.72. We see that A_5 is strongly peaked about the true median. It is clear also from the symmetry of the algorithm that the expected rank of A_n is $(n^2+1)/2$, that is, the true median. We now show that A_n is guaranteed to filter out the upper and lower quartiles of the data completely.

Theorem. $\text{rank}(A_n) \geq (n^2 + 2n + 1)/4$ and
$\text{rank}(A_n) \leq (3n^2 - 2n + 3)/4$.

Proof: To obtain the first inequality we need only observe that A_n surely exceeds $(n + 1)/2$ of the values in $(n - 1)/2$ of the n-sets and $(n - 1)/2$ of the values in its own n-set. The proof of the second inequality is analogous. □

In summary, A_n has the following desirable properties:

a) It is unbiased for the median.

b) It is strongly concentrated about the median.

c) It is outlier-resistant because the upper quarter and lower quarter of the data are eliminated completely.

We contend that A_n is an easily-computable and admirable substitute for the median in picture processing applications.

4. Extensions and Unsolved Problems

The mean and variance are sufficient statistics for normally-distributed data. Their robust analogs are the median and interquartile range, respectively. (The interquartile range is the difference between the first and third quartile values.) It would be useful to generalize the cascade circuits so that they produce an estimate of the interquartile range. One would also like to obtain the exact distribution of A_n and the interquartile range estimate.

The comparator modules discussed in this paper are not ideal for LSI implementation and the median computation can be performed using more suitable primitives. The methods presented here, however, at least illustrate the theoretical tools that one might use to design and test an actual implementation. Other probabilistic approaches are suggested in [3].

5. Acknowledgements

The author wishes to thank Professors Jon Bentley and Bruce Weide for engaging discussions. This research was supported in part by the Office of Naval Research under Contract N000014-76-C-0373.

REFERENCES

[1]. Knuth, D. E., The Art of Computer Programming, Volume III: Sorting and Searching, Addison-Wesley (1973). Section 5.3.4.

[2]. Tukey, J. W., The ninther, a technique for low effort robust(resistant) location in large samples, in David, H. A., ed., Contributions to Survey Sampling and Applied Statistics, Academic Press (1978), pp. 251-257.

[3]. Weide, B. W., Statistical Methods in Algorithm Design and Analysis, Ph.D. Thesis, Department of Computer Science, Carnegie-Mellon University (1978), unpublished.