THE CONVERGING SQUARES ALGORITHM: AN EFFICIENT MULTIDIMENSIONAL PEAK PICKING METHOD

Lawrence O'Gorman and Arthur C. Sanderson
Department of Electrical Engineering and The Robotics Institute
Carnegie-Mellon University,Pittsburgh, PA, 15213

ABSTRACT

The converging squares algorithm is a method for locating peaks in sampled data of 2 dimensions or higher. There are two primary advantages of this algorithm over other conventional methods. First, it is robust with respect to noise and data type. There are no empirical parameters to allow adjustment of the process, so results are completely objective. Secondly, the method is computationally efficient. The inherent structure of the algorithm is that of a resolution pyramid. This enhances computational efficiency as well as contributing to the quality of noise immunity of the method. The algorithm is detailed for 2-dimensional data. Quantitative comparisons of computation are made with two conventional peak picking methods.

INTRODUCTION

The converging squares algorithm is a method which locates peaks in peak regions, in sampled data on orthogonal planes of 2 dimensions or higher. The method is robust and unambiguous in its location of the peak. Effects of noise are minimized without altering the peak location, as may result due to filtering. It does not require specification of empirical constants, nor does it facilitate other adjustments which might yield different results. It performs best in locating peaks within convex and non-elongated shapes.

The method is equivalent to tracking the centroid of the image through successive increases in spatial resolution. In 2 dimensions, a square image of size $n \times n$, is segmented into 4 $(n-1)\times(n-1)$ size squares. The summation within each of these sub-squares is found, and the maximum of these is chosen as the square for the next cycle. This is continued until the comparison is among 4 samples, and the peak is found. Thus the sub-square chosen on each cycle represents the (square) region of maximum density. In fact, the process may be stopped at any size square to obtain a maximum density region rather than the peak point. The method is easily extended to higher dimensions.

The structure of the algorithm is that of a resolution pyramid [1, 2, 3], where processing is performed sequentially and monotonically on cycles from lowest to highest resolutions. This pyramidal structure, and the redundancy of overlap are both capitalized upon to make the algorithm computationally efficient. In addition, there are fewer computationally expensive conditional operations for the converging squares algorithm than for conventional methods. In 2 dimensions, the converging squares algorithm approaches 2 to 3 times the speed of peak detection by sample-by-sample maximum value location, and 5 to 6 times the speed of peak detection by 4-point smoothing followed by maximum value location.

CONVERGING SQUARES ALGORITHM

Purpose

1. To find the maximum density region and its peak.

Peak is defined here not as the point of maximum value, but as the point of maximum value within the region of maximum density. The density is defined here as the summation of values over all points in a region, divided by the number of points in that region. Thus, where a point of impulse noise may cause the peak to be located there using a maximum point algorithm, the converging squares algorithm will be primarily influenced by that *region's* density rather than by a single value. Furthermore, the term "region" has no fixed size, but rather due to the nature of the *converging* algorithm, covers the (discrete) continuum of square sizes from the entire image size down to the single sample peak. (Re: Fig. 1)

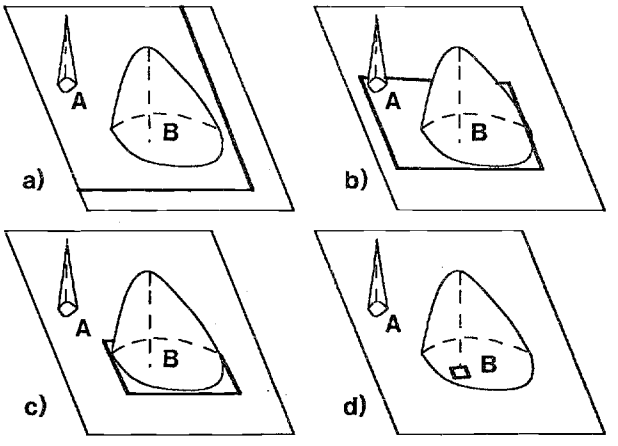

Figure 1: Progression of converging squares from large square in (a), to peak point in (d). Progression from (b) to (c) shows rejection of impulse noise at A.

3A.3

CH1841-6/83/0000-0112 $1.00 © 1983 IEEE

2. To be computationally efficient.

   Important considerations here are the number of operations and the type of operations. If conditionals can be replaced by less time-consuming operations such as additions and subtractions then processing will be faster.

3. To utilize no empirical parameters, nor enable any adjustment of the algorithm to yield different results.

   The goal here is to make the algorithm as robust as possible. The converging squares algorithm has no parameters by which the results of the method may be altered.

4. To be relatively impervious to white noise in the data, and to minimize the effect of impulse noise without the use of filtering, which causes data distortion.

   The use of linear low-pass filters with non-linear phase to reduce the effects of noise, will distort the data, the degree which is dependent upon the amount of filtering and the filter parameters. The converging squares algorithm leaves the data — including noise — intact. Effects of noise are minimized by the nature of the algorithm which makes comparisons not sample-by-sample, but on regions of the image from low to high resolutions. On the initial cycles of the algorithm, regions are large, and resolution is low, so the effect of white noise in the sum of values in the region is essentially averaged out (much more so in fact than by a practical spatial domain smoothing filter of small mask size) and the effect of impulse noise is reduced. Toward the final cycles, regions are smaller and the averaging effect is less, so the algorithm is able to converge to a true peak which has not been reduced or shifted by any operation.

Method — 2 Dimensions

The method by which the converging squares algorithm finds the peak is quite simple. An image of square size $k \times k$ is subdivided into 4 overlapping squares of size $(k-1) \times (k-1)$ (Fig. 2). A comparison is made of the densities or summations within the 4 squares, and the maximum is chosen. The next cycle repeats with that smaller square, etc., until comparison is among 4 pixels, and the peak is found. Since for each cycle, the region of overlap (Φ in Fig. 3) is common to all 4 squares, there is no need to compute this region's summation, and computation can be reduced accordingly.

Figure 3 shows the image array $\{a_{ij}\}$. It is of size $k \times k$ with top left pixel a_{11} and bottom right pixel, a_{kk}. The algorithm converges for squares within this array of sidelengths k=n to k=3 in increments of k=1. Thus on each cycle, one row and one column are eliminated. These cycles are followed by a single step where 4 pixel values of the 2×2 square are compared and the maximum is chosen as the peak.

In the following steps, a superscript refers to the cycle number (from k=n to k=3), which is the same as the square sidelength on that cycle. Where the meaning is evident, superscripts will sometimes be omitted to aid in readability. The double subscripts are coordinates, with respect to the square of that cycle. For example, a_{1k} refers to the 1st row and k^{th} column of the $k \times k$ square of that cycle (and not to the original $n \times n$ square). The steps are as follows:

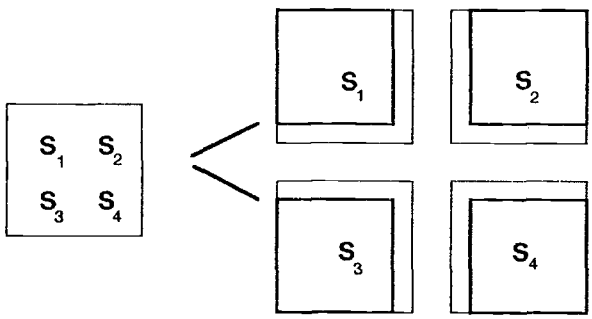

Figure 2:
Each square of sidelength k is separated into 4 squares of sidelengths k-1.

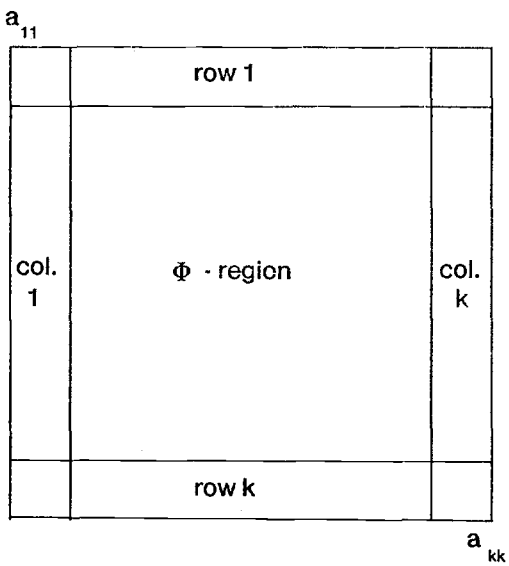

Figure 3:
This figure shows a square image a(i,j) of size $k \times k$ with rows 1 and k of size $(k-2) \times 1$ and columns 1 and k of size $1 \times (k-2)$. The Φ region is the region of overlap of squares S_1 to S_4 in Fig. 2.

1. For the initial image of size $n \times n$, sum the start and end rows and columns, not including the corner pixels:

$$r_1 = \sum_{j=2}^{n-1} a_{1j} \ , \ r_n = \sum_{j=2}^{n-1} a_{nj}$$
$$c_1 = \sum_{i=2}^{n-1} a_{i1} \ , \ c_n = \sum_{i=2}^{n-1} a_{in}. \qquad (1)$$

2. Find the sums of the pixel values of sub-squares S_1 to S_4 (Fig. 2), not including the summation within the common Φ area:

$$S_1 = c_1 + r_1 + a_{11} \ , \ S_2 = r_1 + c_n + a_{1n}$$
$$S_3 = c_1 + r_n + a_{n1} \ , \ S_4 = r_n + c_n + a_{nn}. \qquad (2)$$

3. Find the maximum density sub-square (of size n-1),
$$S_{max}^{n-1} = max\{S_1, S_2, S_3, S_4\}. \qquad (3)$$

4. For the square images of sidelength from $k = n-1$ to $k = 3$:

a. Determine the coordinates, and calculate the new row and column summations:

case 1: if $S_{max}^{k} = S_1$,

then, the coordinates of the new square are:
$$i_1^k = i_1^{k+1} \ , \ i_k^k = i_k^{k+1} - 1$$
$$j_1^k = j_1^{k+1} \ , \ j_k^k = j_k^{k+1} - 1 \qquad (4)$$

and, the new row and column summations are:
$$r_1^k = r_1^{k+1} - a_{1k} \ , \ r_k^k = \sum_{j=2}^{k-1} a_{kj}$$
$$c_1^k = c_1^{k+1} - a_{k1} \ , \ c_k^k = \sum_{i=2}^{k-1} a_{ik} \qquad (5)$$

case 2: if $S_{max}^{k} = S_2$,

then, the coordinates of the new square are:
$$i_1^k = i_1^{k+1} \ , \ i_k^k = i_k^{k+1} - 1$$
$$j_1^k = j_1^{k+1} + 1 \ , \ j_k^k = j_k^{k+1} \qquad (6)$$

and, the new row and column summations are:
$$r_1^k = r_1^{k+1} - a_{11} \ , \ r_k^k = \sum_{j=2}^{k-1} a_{kj}$$
$$c_1^k = \sum_{i=2}^{k-1} a_{i1} \ , \ c_k^k = c_k^{k+1} - a_{kk} \qquad (7)$$

case 3: if $S_{max}^{k} = S_3$,

then, the coordinates of the new square are:
$$i_1^k = i_1^{k+1} + 1 \ , \ i_k^k = i_k^{k+1}$$
$$j_1^k = j_1^{k+1} \ , \ j_k^k = j_k^{k+1} - 1 \qquad (8)$$

and, the new row and column summations are:
$$r_1^k = \sum_{j=2}^{k-1} a_{1j} \ , \ r_k^k = r_k^{k+1} - a_{kk}$$
$$c_1^k = c_1^{k+1} - a_{11} \ , \ c_k^k = \sum_{i=2}^{k-1} a_{ik} \qquad (9)$$

case 4: if $S_{max}^{k} = S_4$,

then, the coordinates of the new square are:
$$i_1^k = i_1^{k+1} + 1 \ , \ i_k^k = i_k^{k+1}$$
$$j_1^k = j_1^{k+1} + 1 \ , \ j_k^k = j_k^{k+1} \qquad (10)$$

and, the new row and column summations are:
$$r_1^k = \sum_{j=2}^{k-1} a_{1j} \ , \ r_k^k = r_k^{k+1} - a_{k1}$$
$$c_1^k = \sum_{i=2}^{k-1} a_{i1} \ , \ c_k^k = c_k^{k+1} - a_{1k} \qquad (11)$$

b. Find the sums of the pixel values of sub-squares S_1 to S_4 (Fig. 2), not including the summation within the common Φ area:

$$S_1 = c_1^k + r_1^k + a_{11} \ , \ S_2 = r_1^k + c_k^k + a_{1k}$$
$$S_3 = c_1^k + r_k^k + a_{k1} \ , \ S_4 = r_k^k + c_k^k + a_{kk} \qquad (12)$$

c. Find the maximum density sub-square (of size k-1),
$$S_{max}^{k-1} = max\{S_1, S_2, S_3, S_4\}. \qquad (13)$$

5. Determine the coordinates of the final square. Same as step 4a, under only the "then" clauses, (not the "and" clauses).

6. Find the maximum pixel value within the 2×2 image. This is the peak.

$$a_{max} = max\{a_{11}, a_{12}, a_{21}, a_{22}\}. \qquad (14)$$

If the *inverse* peak (or lowest region) is desired instead of the

peak, the only modification required is to change the term *max* to *min* in steps 3, 4c, and 6.

The methods for 3 and higher dimensions are similar to that for 2 dimensions.

Computation

In this section, computation required by the converging squares algorithm is compared to that required by two other simple 2-dimensional peak finding methods. The other methods are: maximum value detection by sample-by-sample comparisons, and 4-point average filtering followed by maximum value detection. It should be noted that each method gives different results, so a comparison strictly on the basis of computation is misleading.

For 2-dimensional data, the computations required are as follows:

- converging squares: $A(n^2+7n-22)+C(5n-7)$
- maximum value: $C(n^2-1)$
- 4-point smoothing and maximum value: $3An^2+C(n^2-1)$,

where A is the time for an addition operation, and C is that for a conditional operation. Although timing varies widely for different processors, we take an approximation of the relation for a Motorola 68000 microprocessor [4]. A conditional (including a compare, conditional jump and a move) is about twice as slow as an addition (including an addition and a move). Using this relationship, the converging squares algorithm approaches twice the speed of the maximum value method, and 5 times the speed of the smoothing method. For the Digital Equipment Corporation VAX 11/780 computer, the timing relationship is estimated at $C=2.75A$ for the same operations. Using this relationship, the converging squares algorithm is 2.75 times faster than the maximum value method, and 5.75 times faster than the smoothing method.

Applications

The converging squares algorithm is currently being used in the area of biomedical image processing for the location of nuclei within liver tissue images. In these images, the nuclei are dark convex objects on a cluttered gray-level background. The algorithm is used to find a nucleus, and the (inverse) peak point within the nucleus. The nucleus edges are then found, and the area within these edges is set to high intensity. This process is repeated until all nuclei in the region are found.

The converging squares algorithm has also been used to find a higher density area of nuclei (an infiltrate) within tissue. In this case, the progress of the algorithm is stopped before reaching a peak point. The boundaries of the square at stoppage enclose the region of (inverse) peak density for this size square.

This algorithm has also been applied to industrial inspection tasks to find peaks and peak regions within gray scale images containing objects such as tools.

Acknowledgement

The authors wish to thank Kendall Preston Jr. for his helpful comments, and Rafael Bracho for his review of this paper. This work was supported in part by NIH grant 1R01GM28221-02.

References

1. D.H. Ballard, C.M. Brown, *Computer Vision*, Prentice-Hall, 1982.
2. M.D. Kelly, "Edge detection in computers by computer using planning," in *Machine Intelligence*, B. Meltzer, D. Mitchie, ed., Ellis Horwood Ltd., England, 1971.
3. S.L. Tanimoto, "Regular Hierarchical Image and Processing Structures in Machine Vision," in *Computer Vision Systems*, Hanson and Riseman, ed., Academic Press, New York, 1978.
4. Motorola, *16-Bit Microprocessor User's Manual*, 3rd ed., 1982.