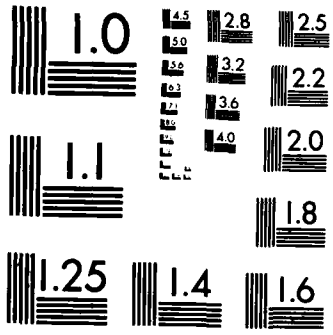
AD-A125 589 DIAMETER-TIME CELLULAR WARPING(U) MARYLAND UNIV COLLEGE PARK COMPUTER VISION LAB A ROSENFELD NOV 82 TR-1230 AFOSR-TR-83-0068 AFOSR-77-3271 1/1

UNCLASSIFIED F/G 12/1 NL



TR-1230
AFOSR-77-3271

November 1982

DIAMETER-TIME CELLULAR WARPING

Azriel Rosenfeld

Computer Vision Laboratory
Computer Science Center
University of Maryland
College Park, MD 20742

COMPUTER SCIENCE TECHNICAL REPORT SERIES

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

TR-1230
AFOSR-77-3271

November 1982

DIAMETER-TIME CELLULAR WARPING

Azriel Rosenfeld

Computer Vision Laboratory
Computer Science Center
University of Maryland
College Park, MD 20742

ABSTRACT

This note describes a cellular array algorithm for performing a general class of geometrical operations on a nxn digital image in O(n) time.

The support of the U.S. Air Force Office of Scientific Research under Grant AFOSR-77-3271 is gratefully acknowledged, as is the help of Janet Salzman in preparing this paper.

1. Introduction

Cellular arrays are arrays of processing elements arranged in a Cartesian grid, with each processor ("cell") connected to its neighbors in the grid. Such an array, with its cells operating synchronously, is a very appropriate multiprocessor architecture for parallel image processing [1]. Several large cellular arrays have actually been built; the largest to date is NASA's Massively Parallel Processor (MPP), which is of size 128×128.

Efficient cellular array algorithms for most of the basic classes of image processing operations are presented in [2,3]. These operations include local operations, where the value of an output pixel depends only on the values of the corresponding input pixel (in the same position) and a set of its neighbors; and statistics computations, where the value of an output pixel is (e.g.) a weighted sum of all the pixel values in the input image. (The values of a transform of the image, such as its Fourier transform, are all weighted sums of this type.)

The purpose of this note is to outline a cellular array algorithm for performing a general class of geometric operations ("warpings") on an image. A geometric operation is defined by a coordinate transformation of the form $x=f(x',y')$, $y=g(x',y')$, where x,y and x',y' are coordinates in the input and output images, respectively. The value of the output pixel at (x',y') depends only on the values of the input pixels in a neighborhood of (x,y), where the relationship between (x,y) and (x',y') is defined by the pair of functions f and g.

An MPP algorithm for rotating an image through an arbitrary angle is described in [2]. This note generalizes that algorithm and shows that it applies to a large class of geometric operations, including all the standard operations composed of translations, rotations, reflections, magnifications, perspective transformations, etc. (For a discussion of image resampling (and rescaling) by a cellular array see [4].)

2. Cellular geometric operations

In an arbitrary geometric operation, the value of each output pixel depends only on the values of a bounded set of input pixels, but these input pixels are at an arbitrary location relative to the output pixel, and the relative location may be very different for different pixels. In order to compute the value of an output pixel, we must bring the needed input values into the output position; but it is not obvious how to do this simultaneously for all the output pixels, since the needed information must travel along a different route in each case, and this may lead to communication conflicts. Note that for simple geometric operations such as translation, it is easy to shift all the needed values synchronously, without conflict; but it is not clear how to do this for an arbitrary operation. A brute-force solution would be to shift the entire input image in such a way that every value passes through every position, thus allowing each output pixel to collect the values that it needs; but this shifting would require $O(n^2)$ time for an $n \times n$ image. In this section we show how to do the necessary shifting in $O(n)$ time for a large class of geometric operations.

To illustrate the approach, suppose first that the desired geometric operation has the following property: Let S_{ij} denote the set of input pixels needed to compute the value of output pixel (i,j); let R_i denote the i^{th} row of the image, and C_j the j^{th} column, where $1 \le i,j \le n$. Let us assume that for all i,j,

the set

$$(\bigcup_i S_{ij}) \cap R_i$$

is of bounded size, i.e., its size does not grow with n. Note that $\bigcup_i S_{ij}$ is the set of input pixels needed to compute the output values for the entire j^{th} column; this set itself is not of bounded size, but we assume that its intersection with any given row is of bounded size. If this assumption is true, we can use the following procedure (compare [2]) to compute the values of all the output pixels in time O(n). The procedure begins with the input image stored in the cellular array, one pixel per cell.

a) Cyclically shift the input image horizontally. During this process, each cell (i,j) "sees" every input value in R_i, and can collect all those values belonging to the set $(\bigcup_i S_{ij}) \cap R_i$. Since this set is of bounded size, the cell has room to store it.

b) Now cyclically shift the stored values vertically. During this process, each cell in the (arbitrary) j^{th} column "sees" all the values in $\bigcup_i [(\bigcup_i S_{ij}) \cap R_i] = \bigcup_i S_{ij}$ - i.e., it sees all the input values needed to compute all the output values in its column. It can thus collect all the data it needs to compute its own output value.

Thus at the end of the process, each cell can compute its output value. The cyclic horizontal and vertical shifting processes each take time O(n), so that the entire procedure takes time O(n). [Cyclic shifting is not essential to the algorithm; the same

effect can be accomplished, in at most twice the time, by shifting a copy of the input data first in one horizontal direction, then in the other, and similarly for the vertical.]

The procedure just described assume that $(\bigcup_i S_{ij}) \cap R_i$ is of bounded size; but this is not true for all geometric operations. For example, consider the operation that simply transposes the image; then to compute each column of output values we need a row of input values, so that $\bigcup_i S_{ij} = R_j$. In fact, consider the operation of rotation by 90° about the center (i_0, j_0); then to compute the values in C_{j_0}, we need all the values in R_{i_0}, i.e., $\bigcup_i S_{ij_0} = R_{i_0}$.

To see how to handle these cases, note that we can transpose an image by simply shifting the input data (cyclically) in the diagonal direction. [The cellular array need not have diagonal neighbor connections; to achieve "diagonal" shifting we simply alternate horizontal and vertical shifts.] This actually brings each input value past the output cell that needs it, which can thus store it, so that when the shifting is complete, all the needed output values are in their proper places. (Diagonal shifting brings the input value (i,j) into the successive positions, e.g., (i+1,j-1), (i+2,j-2),..., modulo n; these are just the positions (h,k) for which h+k=i+j, so that they include the position (j,i).)

The use of diagonal shifting to transpose an image suggests a generalization of our geometric operation procedure, based on

(cyclic) shifts in two directions, not necessarily horizontal and vertical. Let θ be any direction, e.g., horizontal, vertical, diagonal, or more generally, a direction defined by a given periodic sequence of moves - e.g., in the first octant ($0 \leq \theta \leq \frac{\pi}{4}$), sequences of k horizontal moves separated by single diagonal moves. Let R_θ denote a "θ-row", i.e., a succession of pixels lying along direction θ. Our generalized procedure depends on the following general assumption: There exists a pair of distinct directions α, β such that the set $\left(\bigcup_{(i,j) \in R_\alpha} S_{ij} \right) \cap R_\beta$ is of bounded size, for all α-rows R_α and all β-rows R_β. If this is true, we proceed by cyclically shifting the input image along the β-rows, and allowing each cell to store all the values that will be needed to compute the outputs in its α-row. We then cyclically shift the stored values along the α-rows, so that each cell can collect the data needed to compute its own output value.

The generalized assumption is evidently valid for all of the standard types of geometric operations. For translation and rescaling we can use the horizontal and vertical directions; for reflection and rotation, we use a pair of directions that do not reflect or rotate into one another (e.g., for transposition we use a diagonal direction; for 90° rotation, we use a pair of non-perpendicular directions). Perspective transformations are also easily handled, since such a transformation can be thought of as a scale change (by a different amount) along each line parallel to a given direction.

3. Concluding remarks

It is known [1,3] that a cellular array can perform local operations on an $n \times n$ image in O(const) time, and can perform statistics computations, and compute discrete transforms, in O(n) time. In this note, generalizing a result in [2], we have shown that a cellular array can also perform a wide class of geometric operations on an image, including all the standard types of operations, in O(n) time.

References

1. A. Rosenfeld, Cellular architectures: from automata to handware, in K. Preston, Jr. and L. Uhr, eds., Multi-computers and Image Processing - Algorithms and Programs, Academic Press, NY, 1982, 253-260.

2. J. P. Strong, Basic image processing algorithms on the massively parallel processor, ibid., 47-85.

3. T. Kushner, A. Y. Wu, and A. Rosenfeld, Image processing on MPP: 1, Pattern Recognition 15, 1982, 121-130.

4. M. R. Warpenburg and L. J. Siegel, Image resampling in an SIMD environment, Proc. 1981 IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management, Nov. 1981, 67-75.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER **AFOSR-TR- 83-0068**	2. GOVT ACCESSION NO. AD-A125589	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DIAMETER-TIME CELLULAR WARPING		5. TYPE OF REPORT & PERIOD COVERED Technical
		6. PERFORMING ORG. REPORT NUMBER TR-1230
7. AUTHOR(s) Azriel Rosenfeld		8. CONTRACT OR GRANT NUMBER(s) AFOSR-77-3271
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Vision Laboratory Computer Science Center University of Maryland College Park, MD 20742		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Math. & Info. Sciences, AFOSR/NM Bolling AFB Washington, DC 20332		12. REPORT DATE November 1982 13. NUMBER OF PAGES 9
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Image processing
Geometric transformations
Warping
Parallel processing
Cellular arrays

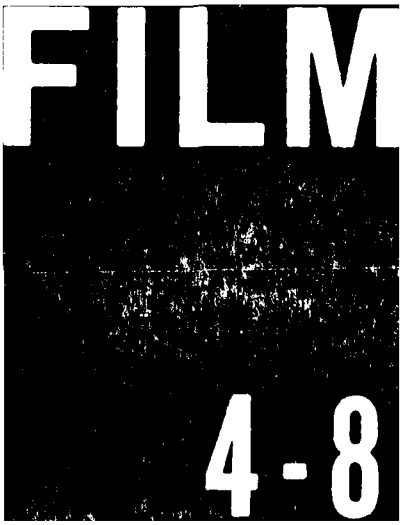
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This note describes a cellular array algorithm for performing a general class of geometrical operations on a nxn digital image in O(n) time.

DD FORM 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)


FILM
4-8