

Computing on a Systolic Screen: Hulls, Contours and Applications

by

Frank Dehne, Jörg-Rüdiger Sack and Nicola Santoro

School of Computer Science

Carleton University

Ottawa, Ontario

Canada K1S 5B6

1. INTRODUCTION

A *digitized plane* Π of size M is a rectangular array of lattice points (or pixels) with integer coordinates (i,j) , where $i,j \in \{1, \dots, \sqrt{M}\}$. A subset $S \subseteq \Pi$ is called an *image* (or digitized picture) on Π ; its complement $\Pi - S$ is denoted by S^c .

A *systolic screen* of size M is a $\sqrt{M} \times \sqrt{M}$ mesh-of-processors where each processing element P_{ij} represents the lattice point (i,j) . A systolic screen is a natural tool for the representation of images on the digitized plane. An image S can be represented by a binary "color-register" $C\text{-Reg}(i,j)$ at each P_{ij} , where

$$C\text{-Reg}(i,j) := \begin{cases} 1, & \text{if } (i,j) \in S \\ 0 & \text{otherwise} \end{cases}$$

A set of (disjoint) images can be represented simultaneously on a Systolic screen by assigning to the C-Registers a different integer value (color) for each image. The fundamental difference between the Systolic Screen and the Mesh-Connected Processor Array (MCPA) lies in the fact that the Screen is a mapping of the entire (digitized) plane while the MCPA (e.g., see [NS80,AK84,MS84,De85]) is a compact representation of the image only.

Mesh-of-processors (or Systolic Screen) have been already used to store images: The maximum size of an image typically ranges from 256x256 pixels in computer vision in an industrial environment to 4000x4000 pixels and larger for aerial photographs. A well known existing system is the MPP designed by NASA for analysing LANDSAT satellite data [Re84]. The MPP consists of 16,384 processing units organized in a 128x128 matrix with a local memory between 1K and 16K bits for each processing unit (to represent a subsquare of pixels).

Computing on a Systolic Screen has been the subject of recent investigations by Miller and Stout [SM84, MS85] they propose $O(\sqrt{M})$ algorithm for the computation of the distance between two images and the computation of the extreme points (with respect to the convex hull), diameter, and smallest enclosing circle of an image as well as for convexity and separability testing and related problems on a Mesh-of-Processors of size $\sqrt{M} \times \sqrt{M}$.

In this paper, we continue the study of computing in a Systolic Screen and present efficient solutions for the following problems:

- 1) computing *all* k^{th} m -contours of an image
- 2) computing *all* k^{th} retilinear convex hulls.

It is shown that both algorithms require $O(\sqrt{M})$ time on a Systolic Screen of size M , i.e. they are optimal. Furthermore, the solution to the first problem yields a new parallel solution to the *longest-common subsequence problem* (e.g. [Hi77, RT85]).

Before presenting the results, we will introduce some basic notation which will be employed throughout the paper (cf. [Ro79, Ki82]). The *4-neighborhood* of a pixel (x,y) is the set of its four horizontal and vertical 4-neighbors $(x\pm 1, y)$ and $(x, y\pm 1)$. The *8-neighborhood* (or neighbors) of (x,y) consists of its 4-neighbors together with its four diagonal neighbors $(x+1,y\pm 1)$ and $(x-1,y\pm 1)$. The *border* S^0 of S is the set of all points of S which have neighbors in S^c . The *interior* of S , $S-S^0$, is denoted by S° .

Let p,q be two points in Π . A *[4-] path* from p to q is a sequence of points $p=p_0, \dots, p_r=q$ such that p_i is a [4-] neighbor of p_{i-1} , $1 \leq i \leq r$. p and q are *[4-] connected* in S if there exists a [4-] path from p to q consisting entirely of points of S . With each pixel $p=(i,j) \in \Pi$ we associate its *cell* $\langle p \rangle := [i-0.5, i+0.5] \times [j-0.5, j+0.5] \subseteq \mathbb{R}^2$ and with each image $S \subseteq \Pi$ its *region* $\langle S \rangle := \bigcup_{p \in S} \langle p \rangle$.

2. DOMINANCE PROBLEMS

2.1 Determination of All K^{th} m -Contours

Given a digitized plane Π of size M and an image $S \subseteq \Pi$, a pixel $s=(i,j) \in \Pi$ *dominates* a pixel $s'=(i',j') \in \Pi$ (abbr. $s \geq s'$) if $i \geq i'$ and $j \geq j'$; and it is called *maximal* in S if there is no other $s' \in S$ which dominates it. The set $\text{MAX}(S)$ of all maximal pixels of S (sorted by x -coordinate) is called the *1st m -contour* of S .

The definition of contour of S can be generalized to introduce the notion of the *K^{th} m -contour* of S , denoted by $\text{MAX}(S,k)$, $k \in \mathbb{N}$, as follows:

$$\text{MAX}(S, 1) := \text{MAX}(S)$$

$$\text{MAX}(S, k+1) := \text{MAX}(S - (\text{MAX}(S,1) \cup \dots \cup \text{MAX}(S,k)))$$

Since in a digitized plane pixels with the same x - or y -coordinate may occur very often, the following restricted definition of dominance on a digitized plane is also useful: a pixel $s=(i,j) \in \Pi$ *strictly dominates* a pixel $s'=(i',j') \in \Pi$ (abbr. $s > s'$) if $i > i'$ and $j > j'$. The *k^{th} m -contour* with respect to the strict dominance relation will be denoted by $\text{MAX}^*(S,k)$.

Assume that an image $S=\{s_1, \dots, s_n\}$ on a digitized plane Π of size M is stored in a Systolic Screen as described above. In addition to $C\text{-Reg}(i,j)$, each P_{ij} contains a second register $K\text{-Reg}(i,j)$. The K -Registers are used for storing the final result, i.e. all k^{th} m -contours, as follows:

$$\text{for all } P_{ij} \text{ for which } (i,j) \in S \quad (K\text{-Reg}(i,j) \leftarrow k) \iff (i,j) \in \text{MAX}(S,k).$$

In Figure 1 we present an algorithm to compute all k^{th} m-contours.

Theorem 1: Algorithm ALL-MAX computes all $\text{MAX}(S,k)$ in time $O(\sqrt{M})$.

Proof: Each processor element (PE) representing a pixel $s \in S$ sends messages which proceed towards the lower left corner of the Systolic Screen to all PEs which are dominated. Thus, in the worst case these messages have to proceed from the upper right to the lower left corner of the mesh taking time $O(\sqrt{M})$. The correctness of the algorithm can be proved by an induction on $|S|$: For $|S| = 1$ the algorithm obviously provides the correct result. Thus, assume $|S| > 1$ and let $s' \in \text{MAX}(S,1)$ be a maximal element of S . We observe that during execution of algorithm ALL-MAX the final status of the registers of each PE is independent on the order in which the PEs are reached by messages originated at other PEs representing pixels $s \in S$: Thus, we obtain the same result by applying algorithm ALL-MAX to $S - \{s'\}$ and then superimposing the messages originated at s' . With this we can easily prove that algorithm ALL-MAX applied to S provides the correct result. Figure 2 shows the possible cases which might occur, when the additional messages are superimposed. •

Algorithm ALL-MAX:

- (1) All PEs P_{ij} initialize their K-Register

$$\text{K-Reg}(i,j) \leftarrow \text{C-Reg}(i,j) .$$

- (2) All PEs with $\text{K-Reg} = 1$ send the contents of their K-Register to their lower and left neighbors, if they exist.
- (3) For ease of description we set $v_u, v_r = 0$ if no value is received. All PEs, P_{ij} , which receive at least one value v_u and/or v_r from their upper and/or right neighbor, respectively, update their K-Register

$$\text{K-Reg}(i,j) \leftarrow \max \{ \text{K-Reg}(i,j), \max \{ v_u, v_r \} + \text{C-Reg}(i,j) \}$$

and send the new contents of their K-Register to their directly connected lower and left neighbors, if they exist.

- (4) Step (3) is iterated until there is no more PE which has received at least one message.

Figure 1
Computation of all $\text{MAX}(S,k)$

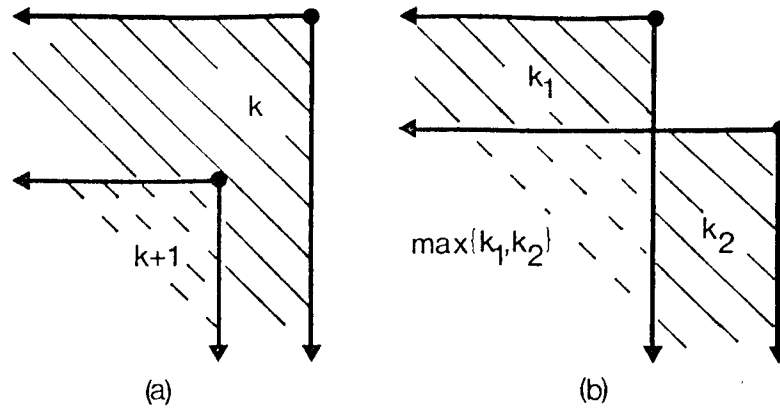


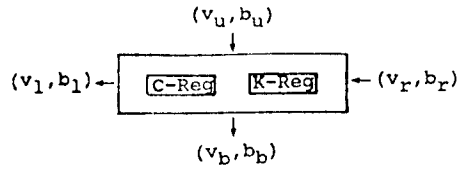
Figure 2

The algorithm for computing all $\text{MAX}^*(S,k)$, given in figure 3a, is essentially the same as the one for computing all $\text{MAX}(S,k)$. However, we have to take into account that a pixel cannot dominate another pixel which has the same x- or y-coordinate. Thus when a PE receives a message it has to know whether this message has been passed on a direct horizontal or vertical line, yet. To provide the necessary information an additional bit b_1 , b_b , respectively, is added to each message sent to a left, lower neighbor, respectively. A bit value 0 indicates that this message has been passed directly leftwards or downwards.

Algorithm **ALL-MAX***:

- (1) All P_{ij} initialize their K-Registers
 $\text{K-Reg}(i,j) \leftarrow \text{C-Reg}(i,j)$
- (2) All PEs with $\text{K-Reg} = 1$ send a message $(\text{K-Reg}, 0)$ to their directly connected lower and left neighbors, if they exist.
- (3) All PEs which receive at least one message (v_u, b_u) and/or (v_r, b_r) from their upper and right neighbor update their K-Reg and send a message (v_b, b_b) and (v_l, b_l) to their lower and left neighbor, respectively, as described in figure 3b.
- (4) Step (3) is iterated until there is no more PE which has received at least one message.

Figure 3a
 Computation of all $\text{MAX}^*(S,k)$



$b_u b_r$	$v_u > v_r$	$v_u = v_r$	$v_u < v_r$
* 0		$[K-Reg] \leftarrow \max\{[K-Reg], v_r\}$ $b_l = 0, b_b = 1 - [C-Reg]$	
0 *		$[K-Reg] \leftarrow \max\{[K-Reg], v_u\}$ $b_l = 1 - [C-Reg], b_b = 0$	
* 1		$[K-Reg] \leftarrow \max\{[K-Reg], v_r + [C-Reg]\}$ $b_l = b_b = 1 - [C-Reg]$	
1 *		$[K-Reg] \leftarrow \max\{[K-Reg], v_u + [C-Reg]\}$ $b_l = b_b = 1 - [C-Reg]$	
0 1	$[K-Reg] \leftarrow \max\{[K-Reg], v_u\}$ $b_l = 1 - [C-Reg], b_b = 0$	$[K-Reg] \leftarrow \max\{[K-Reg], v_r + [C-Reg]\}$ $b_l = b_b = 1 - [C-Reg]$	
1 0	$[K-Reg] \leftarrow \max\{[K-Reg], v_u + [C-Reg]\}$ $b_l = b_b = 1 - [C-Reg]$		$[K-Reg] \leftarrow \max\{[K-Reg], v_r\}$ $b_l = 0, b_b = 1 - [C-Reg]$
0 0	$[K-Reg] \leftarrow \max\{[K-Reg], v_r, v_u\}$ $b_l = 1 - [C-Reg], b_b = 0$	$b_l = b_b = 0$	$b_l = 0, b_b = 1 - [C-Reg]$
1 1	$[K-Reg] \leftarrow \max\{[K-Reg], \max\{v_u, v_r\} + [C-Reg]\}$ $b_l = b_b = 1 - [C-Reg]$		
$v_l = v_b = [K-Reg]$			

(* = no message)

Figure 3b

I/O Operations Performed by Each PE

Theorem 2: Algorithm ALL-MAX* computes all MAX*(S,k) in time $O(\sqrt{M})$.

Proof: see [DSS86] •

The points of each k^{th} m-contour computed by algorithm ALL-MAX* define a 4-path of pixels which we will refer to as the k^{th} m-chain, denoted M-CHAIN(S,k), of S. We observe that upon termination of algorithm ALL-MAX*, the K-Registers of PEs of all pixels which lie on the k^{th} m-chain or below the k^{th} m-chain and above the $(k+1)^{\text{th}}$ m-chain have value k. We refer to this set of pixels of S, with K-Register value equal to k, as the k^{th} m-belt of S, and denote it by M-BELT(S,k).

Corollary 3: On a Systolic Screen of size M, all m-chains and m-belts of an image can be computed in time $O(\sqrt{M})$.

2.2 The Longest Common Subsequence Problem

The proposed algorithm for parallel computation of all k^{th} m-contours yields a new parallel solution of the longest common subsequence problem which is defined as follows:

Given two strings $A=A(1) \dots A(n)$ and $B=B(1) \dots B(m)$, $n \geq m$, over some finite alphabet Σ , a *substring* C of A

is defined to be any string $C=C(1) \dots C(r)$ for which there exists a monotone strictly increasing function $f: \{1, \dots, r\} \rightarrow \{1, \dots, n\}$ with $C(i)=A(f(i))$, for all $1 \leq i \leq r$. The *longest common subsequence problem* is to find a string of maximum length which is a substring of both A and B.

A table of currently known sequential solutions of the longest common subsequence problem is given in figure 4 (p denotes the length of a longest common subsequence and r is the total number of ordered pairs (i,j) with $a_i=b_j$).

	Running Time	Worst-case behaviour
[HS77]	$O((r+n) \log n)$	$O(n^2 \log n)$
[Hi77]	$O(p \cdot n)$	$O(n^2)$
[Hi77]	$O(p(m+1-p) \log n)$	$O(n^2 \log n)$
[NKY82]	$O(n(m-p))$	$O(n^2)$

Figure 4
Sequential Solutions of the Longest Common
Subsequence Problem (from [RT85])

Hirschberg [Hi78] proved an $\Omega(n \log n)$ information theoretic lower bound for sequential solutions of the longest common subsequence problem. Recently, [RT85] introduced a parallel algorithm which computes a longest common subsequence in time $O(n)$ using a one-dimensional systolic array of size m with a systolic stack of size n associated with each PE.

We will now give an $O(n)$ time solution of the longest common subsequence problem on a Systolic Screen of size $n \times m$ such that all processing elements are of one type only (cf. [RT85]). Our solution has the additional advantage that it determines also *all* longest common subsequences in time $O(n)$. The central idea which leads to this method is a transformation of the longest common subsequence problem to the K^{th} m-contour determination. This reduction was also used by Hirschberg in [Hi77].

Lemma 4:

- (a) $A(i_1) \dots A(i_r) = B(j_1) \dots B(j_r)$ is a common subsequence of A and B if and only if $(i_1, j_1) < (i_2, j_2) < \dots < (i_r, j_r)$.
- (b) The length of a longest common subsequence is $k_{\max} := \max\{k \in \mathbb{N} / \text{MAX}^*(S_{A,B}, k) \neq \emptyset\}$ with $S_{AB} := \{(i,j) / A(i)=B(j)\}$.

Proof: see [Hi77] . •

An illustration of Lemma 4 is given in figure 5. The computation of a longest common subsequence of two strings A and B can be mapped into all m-contours problem with respect to the set S_{AB} .

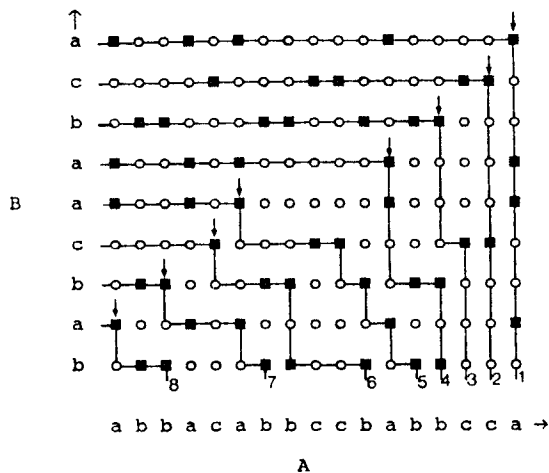


Figure 5

Before we consider the longest common subsequence problem we will first introduce the MAXBELOW searching procedure.

Given two sets P' , P'' of processors on a $\sqrt{n} \times \sqrt{n}$ subsquare of a Systolic Screen of size N each containing three registers X , Y , and Z (for positive numbers) then we define a procedure MAXBELOW (P' , P'' , X , Y , Z) which results in the following:

$$(\forall p' \in P'): X(p') \leftarrow \max \{ Y(p'') / p'' \in P'' \text{ and } Z(p'') \leq Z(p') \} \cup \{ 0 \}$$

with $X(p)$ [$Y(p)$, $Z(p)$] denoting the contents of the register X [Y , Z , respectively] contained in processor p .

Theorem 5: Procedure MAXBELOW (P' , P'' , X , Y , Z) as described in figure 6 performs MAXBELOW search on a $\sqrt{n} \times \sqrt{n}$ submesh in time $O(\sqrt{n})$.

Procedure **MAXBELOW** (P', P'', X, Y, Z):

- (1) Sort $P' \cup P''$ with respect to the contents of their register Z in snake-like ordering (cf. [TK77]).
- (2) Perform one shift procedure for each row of PEs and compute the following :
 - For each PE in the row compute the maximum contents of the Y -registers of all PEs $p'' \in P''$ in the row (0 if no such PE exists) and store this value using an additional register $ROWY$.
 - For each PE $p' \in P'$ in the row compute the maximum contents of the Y -registers of all PEs $p'' \in P''$ in the row which have lower rank with respect to the snake-like ordering computed in step (1) and store it into its X -register (0 if no such PE exists).
- (3) Perform a shift procedure for each column of PEs and assign to the X -register of each $p' \in P'$ the maximum of its current contents and the contents of the register $ROWY$ of all PEs in the column which have lower rank with respect to the snake-like ordering.

Figure 6

Theorem 6: Given two strings $A=A(1) \dots A(n)$ and $B=B(1) \dots B(m)$, $n \geq m$. Using a Mesh-of-Processors of size $n \times m$ the following problems can be solved in time $O(n)$:

- (a) Computation of a longest common subsequence of A and B .
- (b) Computation of all longest common subsequences of A and B .

Proof: (a) see [DSS86]. (b) Given the set of all m -contours of S_{AB} . From Lemma 4 we know that each longest common subsequence is induced by a sequence s_1, \dots, s_r of points of S_{AB} such that $s_1 < \dots < s_r$. The set of all such sequences is obtained by computing for each $s = (i, j) \in \text{MAX}^*(S_{AB}, k)$ its set of next dominances $\text{ND}(s) := \{ s' \in \text{MAX}^*(S_{AB}, k-1) / s < s' \}$, $1 < k \leq k_{\max}$. We observe that $\text{ND}(s) = \{ (i', j') \in \text{MAX}^*(S_{AB}, k-1) / i' > i \text{ and } j' > j \}$. Thus, it suffices to store for each $s = (i, j) \in \text{MAX}^*(S_{AB}, k)$, $1 < k \leq k_{\max}$, the two values $i^* := \min \{ i' > i / (i', j') \in \text{MAX}^*(S_{AB}, k-1) \}$ and $j^* := \min \{ j' > j / (i', j') \in \text{MAX}^*(S_{AB}, k-1) \}$. This can be performed by a global MAXBELOW search procedure in time $O(n)$ as described above for each $[\text{MAX}^*(S_{AB}, k), \text{MAX}^*(S_{AB}, k-1)]$, $1 < k \leq k_{\max}$, in parallel. •

3. DETERMINATION OF ALL K^{TH} RECTILINEAR CONVEX HULLS

Considerable attention has been given to finding estimators which identify the center of a set S and the depth of points with respect to S (see [Sh78], [OL81], [LP84]). For sets $S \subseteq \mathbb{E}^2$ in the Euclidean plane Shamos (cf. [Sh78]) described a sequential $O(n^2)$ time algorithm for "peeling" S by iterating the following process: compute the convex hull (see [PS85]) of S and remove its vertices from S , which is the two-dimensional analogous to the concept of the α -trimmed mean used in robust statistics (see [Sh78] p. 83 ff, [Hu72]). He also proved an $\Omega(n \log n)$ (sequential) lower bound for this problem. Subsequently Overmars and van Leeuwen [OL81] and Chazelle [Ch83] gave $O(n \log^2 n)$ and $O(n \log n)$, respectively, (sequential) solutions for this problem. Obviously all convex layers are a suitable representation of a set of point comparable with the sorted order in the one-dimensional case. Chazelle, Guibas and Lee [CGL83] demonstrated how to apply this structure to improve upon previous solutions of the halfplane range query problem.

This section will deal with the concept of "peeling" an image $S = \{s_1, \dots, s_n\}$ of size M in a digitized plane, i.e. iterating on the following process: compute the *rectilinear* convex hull of S and remove its vertices from S .

On a Systolic Screen of size M we give an $O(\sqrt{M})$ parallel algorithm to peel an arbitrary image. We call S *rectilinear convex*, if the intersection of its region $\langle S \rangle$ and an arbitrary horizontal or vertical line in $\langle \Pi \rangle$ consists of at most one line segment.

The intersection of all rectilinear convex images $S' \subseteq \Pi$ which contain S is called the *rectilinear convex hull* of S and denoted by $\text{HULL}(S)$. The rectilinear hull determination has been discussed in Sack [Sa84], Wood [Wo84], Monturo [Mo82]. The k^{th} *rectilinear convex hull* $\text{HULL}(S, k)$ and the k^{th} *rectilinear convex belt* $\text{BELT}(S, k)$ of S ($k \in \mathbb{N}_0$) are defined as follows:

- (a) $\text{HULL}(S, 0) := \Pi$
- $\text{HULL}(S, 1) := \text{HULL}(S)$
- $\text{HULL}(S, k+1) := \text{HULL}((\text{HULL}(S, k) \cap S) - \text{HULL}(S, k)^{\circ})$
- (b) $\text{BELT}(S, k) := \text{HULL}(S, k) - \text{HULL}(S, k+1)$

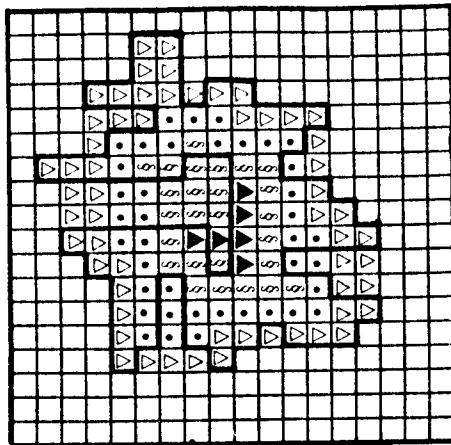
See figure 7 for an illustration.

The maximum of all $k \in \mathbb{N}_0$ such that $\text{HULL}(S, k) \neq \emptyset$ is called the *depth* of S and denoted by $\text{DEPTH}(S)$.

For each pixel $s \in \Pi$ we define its depth $\text{DEPTH}(s, S)$ in S :

$$\text{DEPTH}(s, S) := k \quad \Leftrightarrow \quad s \in \text{BELT}(S, k).$$

Obviously, $\text{DEPTH}(S) = \max \{ \text{DEPTH}(s, S) \mid s \in \Pi \}$.








BELT(S,0)  BELT(S,1) 
 BELT(S,2)  BELT(S,3) 
 BELT(S,4) 

Figure 7
All BELT(S,k) of an Image S

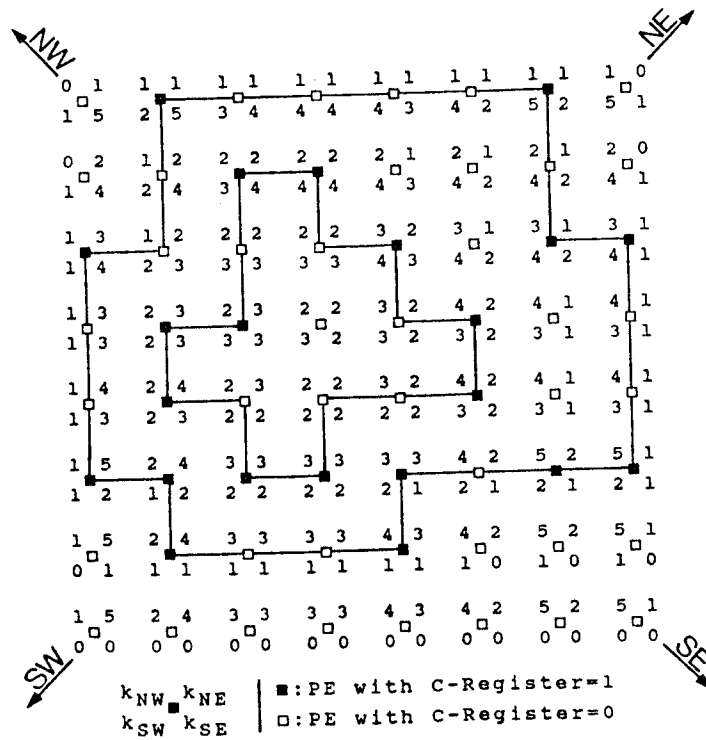


Figure 8

Given a pixel $s \in \Pi$. We define $k_{NE}(s,S) := k$ [$k_{SE}(s,S) := k$, $k_{SW}(s,S) := k$, $k_{NW}(s,S) := k$] if $s \in M\text{-BELT}(S,k)$ with respect to the NE-direction [SE-direction, SW-direction, NE-direction, respectively], see figure 8 for an illustration.

Lemma 7:

- (a) $(\forall s \in \Pi): s \in \text{BELT}(S,k) \iff \min\{k_{NW}(s,S), k_{SW}(s,S), k_{NE}(s,S), k_{SE}(s,S)\} = k$
- (b) $(\forall 0 \leq k \leq \text{DEPTH}(S)): \text{BELT}(S,k) = \{ s \in \Pi / \min\{k_{NW}(s,S), k_{SW}(s,S), k_{NE}(s,S), k_{SE}(s,S)\} = k \}$
- (c) $\text{DEPTH}(S) = \max \{ \min\{k_{NW}(s,S), k_{SW}(s,S), k_{NE}(s,S), k_{SE}(s,S)\} / s \in \Pi \}$

This yields the following

Theorem 8: On a Systolic Screen of size M all k^{th} rectilinear convex hulls $\text{HULL}(S,k)$, all rectilinear convex belts $\text{BELT}(S,k)$ and the depth $\text{DEPTH}(S)$ of an image S can be computed in time $O(\sqrt{M})$.

REFERENCES

- [AG85] M.J. Atallah, M.T. Goodrich, EFFICIENT PARALLEL SOLUTIONS TO GEOMETRIC PROBLEMS, Report CSD-TR-504, Purdue Univ., March 1985
- [AHU76] A.V. Aho, D.S. Hirschberg, J.D. Ullmann, BOUNDS ON THE COMPLEXITY OF THE LONGEST SUBSEQUENCE PROBLEM, J. ACM, Vol. 23, No. 1, 1976, pp 1-12
- [AH85] M.J. Atallah, S.E. Hambrusch, SOLVING TREE PROBLEMS ON A MESH-CONNECTED PROCESSOR ARRAY, Report CSD-TR-518, Purdue Univ., West Lafayette, April 1985.
- [AK84] M.J. Atallah, S.R. Kosaraju, GRAPH PROBLEMS ON A MESH-CONNECTED PROCESSOR ARRAY, J. ACM, Vol. 31, No. 3, July 1984, pp 649-667
- [Ch83] B.M. Chazelle, OPTIMAL ALGORITHMS FOR COMPUTING DEPTHS AND LAYERS, Proc. 21st Allerton Conference on Communication Control and Computing, Oct. 1983, pp 427-436
- [Ch84] B.M. Chazelle, COMPUTATIONAL GEOMETRY ON A SYSTOLIC CHIP, IEEE Trans. on Computers, Vol. C-33, No. 9, Sept. 1984, pp.774-785
- [CGL83] B.Chazelle, L.J.Guibas, D.T./Lee, THE POWER OF GEOMETRIC DUALITY, Proc. 24th IEEE Symp. on Found. of Computer Science, Tucson, Ariz., 1983
- [De85a] F. Dehne, SOLVING GEOMETRIC PROBLEMS ON MESH-CONNECTED AND ONE DIMENSIONAL PROCESSOR ARRAYS, in H. Noltemeier (ed.), Proceedings of the WG'85 International Workshop on Graphtheoretic Concepts in Computer Science, June 18-21, 1985, Linz: Trauner Verlag, 1985, pp 43-59
- [De85b] F. Dehne, A ONE DIMENSIONAL SYSTOLIC ARRAY FOR THE LARGEST EMPTY RECTANGLE PROBLEM, Proc. of the 23rd Annual Allerton Conference on Communication, Control and Computing, Monticello, Illinois, October 2-4, 1985, pp 518-528
- [De85c] F. Dehne, $O(N^{1/2})$ ALGORITHMS FOR THE MAXIMAL ELEMENTS AND ECDF SEARCHING PROBLEM ON A MESH-CONNECTED PARALLEL COMPUTER, Information Processing Letters, Vol 22, No 6, May 1986, pp 303-306
- [DL81] P.E. Danielson, S. Levialdi, COMPUTER ARCHITECTURE FOR PICTORIAL INFORMATION SYSTEMS, IEEE Computer, Nov. 1981
- [DSS86] F. Dehne, J.-R. Sack, N. Santoro, COMPUTING ON A SYSTOLIC SCREEN: HULLS, CONTOURS AND APPLICATIONS, Tech. Rept. , School of Computer Science, Carleton University, Ottawa, July 1986.
- [Ha83] S.E. Hambrusch, VLSI ALGORITHMS FOR THE CONNECTED COMPONENT PROBLEM, SIAM J. COMPUT., Vol. 12, No. 2, May 1983, pp.354-365

- [Hi77] D.S. Hirschbeg, ALGORITHMS FOR THE LONGEST COMMON SUBSEQUENCE PROBLEM, J.ACM, Vol. 24, No. 4, 1977, pp 664-675
- [Hi78] D.S. Hirschberg, AN INFORMATION THEORETIC LOWER BOUND FOR THE LARGEST COMMON SUBSEQUENCE PROBLEM, Inform. Proc. Lett., Vol. 7, No. 1, 1978, pp 40-41
- [Hu72] P.J. Huber, ROBUST STATISTICS: A REVIEW, Ann. Math. Stat., Vol. 43, No. 4, 1972, pp 1041-1067
- [HF83] K. Hwang, K.S. Fu, INTEGRATED COMPUTER ARCHITECTURES FOR IMAGE PROCESSING AND DATABASE MANAGEMENT, IEEE Computer, Vol. 16, pp. 51-61, Jan. 1983
- [HS77] J.W. Hunt, T.G. Szymanski, A FAST ALGORITHM FOR COMPUTING LONGEST COMMON SUBSEQUENCES, C. ACM, Vol. 20, 1977, pp 350-353
- [Ki82] C.E. Kim, DIGITAL DISKS, Report CS-82-104, Computer Science Dept., Washington State University, Dec. 1982
- [KI79] R. Klette, A PARALLEL COMPUTER FOR IMAGE PROCESSING, Elektronische Informationsverarbeitung und Kybernetik, EIK 15 (1979) 56/6, pp. 237-263
- [Le79] C.E. Leiserson, SYSTOLIC PRIORITY QUEUES, Proc CALTECH Conference on VLSI, (ed. C. E. Leitz), California Institute of Technologies, Pasadena, CA, 1979
- [[LP84] D.T. Lee, F.P. Preparata, COMPUTATIONAL GEOMETRY - A SURVEY, IEEE Trans. on Computers, Vol. C-33, No. 12, Dec. 1984, pp 1072-1101
- [LP85] E. Lodi, L. Pagli, A VLSI ALGORITHM FOR A VISIBILITY PROBLEM, in Bertolazzi, Luccio (Ed.), 'VLSI: Algorithms and Architectures', North Holland, 1985
- [Mi84] P.L. Mills, THE SYSTOLIC PIXEL: A VISIBLE SURFACE ALGORITHM FOR VLSI, Computer Graphics Forum 3, North Holland 1984
- [Mo70] G.U. Montanari, ON LIMIT PROPERTIES IN DIGITIZATION SCHEMES, J. ACM 17, 1970, pp 348-360
- [Mo82] M.F. Montuno, A. Fournier FINDING THE X-Y CONVEX HULL OF A SET OF X-Y POLYGONS", Report, CSRG-148, University of Toronto, Toronto, Nov. 1982.
- [MS84] R. Miller, Q.F. Stout, COMPUTATIONAL GEOMETRY ON A MESH-CONNECTED COMPUTER, Proc. Int. Conf. on Parallel Processing, 1984
- [MS85] R. Miller, Q.F. Stout, GEOMETRIC ALGORITHMS FOR DIGITIZED PICTURES ON A MESH-CONNECTED COMPUTER, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-7, No. 2, March 1985, pp 216-228
- [NKY82] N. Nakatsu, Y. Kambayashi, S. Yajima, A LONGEST COMMON SUBSEQUENCE ALGORITHM SUITABLE FOR SIMILAR TEXT STRINGS, Acta Informatica 18 (1982), pp 171-179
- [NS79] D. Nassimi, S. Sahni, BITONIC SORT ON A MESH-CONNECTED PARALLEL COMPUTER, IEEE Trans. on Computers, Vol. C-28, No. 1, Jan. 1979, pp. 2-7
- [NS80] D. Nassimi, S. Sahni, FINDING CONNECTED COMPONENTS AND CONNECTED ONES ON A MESH-CONNECTED PARALLEL COMPUTER, SIAM J. COMPUT., Vol. 9, No. 4, Nov. 1980, pp. 744-767
- [NS81] D. Nassimi, S. Sahni, DATA BROADCASTING IN SIMD COMPUTERS, IEEE Trans. on Computers, Vol. C-30, No. 2, Feb. 1981, pp 101-106
- [OL81] M.H. Overmars, J.V. Leeuwen, MAINTENANCE OF CONFIGURATIONS IN THE PLANE, Report RUU-CS-81-3, Dept. of Computer Science, Univ. of Utrecht, Feb. 1981
- [Pr84] F.P. Preparata, VLSI ALGORITHMS AND ARCHITECTURES, Proc. Mathematical Foundations of Computers Science, Praha 1984, Lecture Notes in Computer Science 176, Springer 1984, pp 149-161
- [PS85] F.P. Preparata, M.I. Shamos, COMPUTATIONAL GEOMETRY, Springer 1985
- [Re84] A.P. Reeves, SURVEY, PARALLEL COMPUTER ARCHITECTURES FOR IMAGE PROCESSING, Computer Vision, Graphics, and Image Processing 25, 1984, pp 68-88
- [Ro79] A. Rosenfeld, DIGITAL TOPOLOGY, Amer. Math. Monthly 86, 1979, pp 621-630
- [RT85] Y. Robert, M. Tchuente, A SYSTOLIC ARRAY FOR THE LONGEST COMMON SUBSEQUENCE PROBLEM, Inform. Proc. Lett., Vol. 21, Oct. 1985, pp 191-198
- [Sa84] J.-R. Sack, RECTILINEAR COMPUTATIONAL GEOMETRY, Technical Report SCS-TR-54, School of Computer Science, Carleton Univ., Ottawa, June 1984
- [Sh76] M.I. Shamos, GEOMETRY AND STATISTICS: PROBLEMS AT THE INTERFACE, in J.F. Traub (ed.), Algorithms and Complexity, Academic Press, New York 1976, pp. 251-280
- [Sh78] M.I. Shamos, COMPUTATIONAL GEOMETRY, Ph.D. Thesis, Yale Univ., 1978
- [SM84] Q.F. Stout, R. Miller, MESH-CONNECTED COMPUTER ALGORITHMS FOR DETERMINATING GEOMETRIC PROPERTIES OF FIGURES, 7th Int. Conf. on Pattern Recognition, Montreal, Canada, July 30 - August 2, 1984

- [Sn81] L. Snyder, OVERVIEW OF THE CHIP COMPUTER, in VLSI 81: Very Large Scale Integration (ed. J.P. Gray), Academic Press, London, 1981, pp.237-246
- [Sn82] L. Snyder, INTRODUCTION TO THE CONFIGURABLE HIGHLY PARALLEL COMPUTER, IEEE Computer 15 (1), Jan. 1982, pp.47-65
- [Sp85a] Th. Spindler, BILDVERARBEITUNG ALS WERKZUEG FÜR COMPUTATIONAL GEOMETRY PROBLEM?, Lecture presented at 3rd Workshop on Computational Geometry, Karlsruhe, March 1985
- [Sp85b] Th. Spindler, private communication
- [St83] Q.F. Stout, MESH-CONNECTED COMPUTERS WITH BROADCASTING, IEEE Trans. on Computers, Vol. C-32, No. 9, Sept. 1983, pp826-830
- [TK77] C.D. Thompson and H.T. Kung, SORTING ON A MESH-CONNECTED PARALLEL COMPUTER, Comm. of the ACM, Vol. 20, No. 4, April 1977, pp. 263-270
- [U184] J.D. Ullman, COMPUTATIONAL ASPECTS OF VLSI, Principles of Computer Science Series, Computer Science Press, 1984
- [Un58] S.H.Unger, A COMPUTER ORIENTED TOWARDS SPACIAL INTERACTION, Proc. IRE, Vol.46, 1958, pp. 1744-1750
- [Wo84] D. Wood, AN ISOTHETIC VIEW OF COMPUTATIONAL GEOMETRY, in Computational Geometry, (ed. G.T. Toussaint), North Holland, 1984, pp. 429-459

Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

258

PARLE Parallel Architectures and Languages Europe

Volume I: Parallel Architectures
Eindhoven, The Netherlands, June 15–19, 1987
Proceedings

Edited by
J.W. de Bakker, A.J. Nijman and P.C. Treleaven



Springer-Verlag

Berlin Heidelberg New York London Paris Tokyo