Collision-Free Path Planning for a Diamond-Shaped Robot Using Two-Dimensional Cellular Automata

Panagiotis G. Tzionas, Adonios Thanailakis, and Philippos G. Tsalides

***Abstract*— This paper presents a new parallel algorithm for collision-free path planning of a diamond-shaped robot among arbitrarily shaped obstacles, which are represented as a discrete image, and its implementation in VLSI. The proposed algorithm is based on a retraction of free space onto the Voronoi diagram, which is constructed through the time evolution of cellular automata, after an initial phase during which the boundaries of obstacles are identified and coded with respect to their orientation. The proposed algorithm is both space and time efficient, since it does not require the modeling of objects or distance and intersection calculations. Additionally, the proposed two-dimensional multistate cellular automaton architecture achieves high frequency of operation and it is particularly suited for VLSI implementation due to its inherent parallelism, structural locality, regularity, and modularity.**

***Index Terms*— Cellular automata, path planning, VLSI, Voronoi diagram.**

I. Introduction

PATH PLANNING typically refers to the design of geometric (kinematic) specifications of the positions and orientations of robots in the presence of obstacles. Path planning can be static or dynamic, depending on the mode in which the obstacle information is available. In a static problem, all the information about obstacles is known *a priori* and the motion of the robot is designed from the given information. In dynamic planning, only partial information is available about the obstacles.

This paper presents a new algorithm for two-dimensional (2-D) static path planning for a diamond-shaped robot among arbitrarily shaped obstacles, which are represented as a discrete image, and the design and VLSI implementation of a cellular automaton (CA) architecture for the efficient execution of the proposed algorithm. The geometry of the workspace is restricted in the sense that a generally shaped robot is enclosed within a diamond-shaped figure. However, this restriction is not critical and it provides generalization for the results of the proposed algorithm since it can be applied to any arbitrarily shaped robot that is enclosed within the diamond-shaped figure. Thus, arbitrary geometries can be "boxed" in this way and by choosing the size of the diamond-shaped figure, the algorithm will take into account errors in the geometric models of the problem. The criterion used for path planning is the "maximum clearance" criterion, and the resulting paths have the advantage of lying as far away from obstacles as possible. Thus, a collision-free path is determined. The proposed algorithm is based on a retraction of the free space onto the computational geometry concept known as the "Voronoi diagram." The Voronoi diagram is constructed through time evolution of cellular automata, after an initial phase during which the boundaries of obstacles are identified and coded with respect to their orientation.

More specifically, the proposed algorithm is divided in two phases. During the first phase, the boundaries of obstacles are determined and coded with respect to their orientation, using a set of template operators. During the second phase, the Voronoi diagram is constructed through time evolution of cellular automata. Each point that belongs on the Voronoi diagram is labeled with a value which represents its distance from the closest boundaries. This information is used for the path planning of a diamond-shaped robot among the obstacles. Additionally, the proposed algorithm imposes a safety margin on the planned motion. The proposed Cellular Automaton architecture was implemented in VLSI using a 1.2 μm double-layer metal CMOS technology and the dimensions for a Von-Neumann neighborhood processor are 2.35 mm $\times$ 2.29 mm $=$ 5.38 mm^2, whereas the maximum frequency of operation is 60 MHz.

The rest of the paper is organized as follows. Section II briefly describes related work in the area of path planning. Section III introduces the basic properties of the Voronoi diagram. Section IV provides a description of the CA architecture and the analysis of expansions which establish the Voronoi diagram. Section V describes the path planning algorithm which consists of two phases: 1) the object boundary detection phase and 2) the Voronoi diagram construction phase. Section VI provides simulation examples. Section VII presents the VLSI implementation of the Cellular Automaton architecture. Finally, Section VIII provides conclusions on the proposed approach.

II. Previous Work

The use of the Voronoi diagram for collision-free path planning was also suggested in [1]–[3]. The main advantage of the algorithm proposed in this paper, compared to already

Manuscript received October 17, 1995; revised March 27, 1996. This paper was recommended for publication by Associate Editor J. C. Trinkle and Editor A. Goldenberg upon evaluation of the reviewers' comments.

P. G. Tzionas and A. Thanailakis are with the Laboratory of Electrical and Electronic Materials Technology, Section of Electronics and Information Systems Technology, Department of Electrical and Computer Engineering, School of Engineering, Democritus University of Thrace, 67100 Xanthi, Greece.

P. G. Tsalides is with the Laboratory of Electronics, Section of Electronics and Information Systems Technology, Department of Electrical and Computer Engineering, School of Engineering, Democritus University of Thrace, 67100 Xanthi, Greece.

Publisher Item Identifier S 1042-296X(97)01040-9.

existing algorithms, is that the Voronoi diagram is constructed through simple, 2-D CA evolution on a 2-D grid, without requiring any distance or intersection computations, or the ordering of distances, and without using an explicit coordinate system or explicit modeling of objects. Moreover, since CA operation is based only on local rules and the metric used is also local in each time step of CA evolution, the proposed algorithm can be applied to path planning problems that include objects of arbitrary shape and size (shape is approximated with consecutive edges at different directions on the grid). Additionally, the proposed CA architecture is especially suitable for parallel implementation in VLSI due to its inherent locality, regularity, modularity, and homogeneous structure. An application-specific integrated circuit (ASIC) was designed and implemented for the proposed algorithm and, thus, execution speed for the construction of the Voronoi diagram is very high. This is considered to be a significant advantage, since an efficient and fast static motion planner is a vital ingredient of a dynamic motion planner in partially known environments [4], in path planning of multiple robots with conflicting/common objectives, where robots of higher precedence are regarded as moving obstacles for robots of lower precedence [5], [6], and in time-varying environments, where robot motions have to be planned from the predicted motions of obstacles and replanning is needed if obstacle motion deviates from prediction [7], [8]. A Voronoi planner of the type proposed in this paper is particularly required in the case where many motion planning problems, with different start and goal configurations within the same environment, are to be solved. In this case, we need to compute once a complete spatial representation and search in this representation for paths connecting many pairs of start and goal configurations. For a marginally changing environment, an updating scheme can be used to avoid complete recomputation of the representation.

An alternative approach, based also on the principle of retraction, is the visibility graph. In this method, a collection of lines is defined in free space such that it connects features of an object to those of another [9]. The visibility graph produces $O(n^2)$ edges for (n) features and, thus, it is a more complex approach than the algorithm proposed in this paper, which produces only $O(n)$ edges for the same number of features.

The potential field approach to path planning uses potential functions that are developed for obstacle avoidance. This approach suffers from local minima that trap the motion of the robot [10]. It has been shown that the simple Newtonian potential is inadequate for deriving accurate representations of the object space. High-order potentials must be used in order to obtain more accurate representations of space. In this case, the resulting path approaches the path computed using the Voronoi diagram, but it is associated with a much higher computational complexity [11].

Methods based on mathematical programming have also been proposed for path planning. These methods use a set of inequalities for the representation of the problem space. However, as the number of inequalities increases, in order to obtain a more accurate description of space, these methods lead to tedious nonlinear optimization problems [11].

Heuristic algorithms have also been proposed for path planning [11]. The algorithm proposed in [12], for example, uses the minimum width of the robot to eliminate narrow paths by computing closest distances to obstacles. This algorithm requires computations of distances and intersections and, thus, large execution times are also required, whereas the algorithm proposed in this paper establishes distance relations through simple CA evolution.

Sweep volume methods have been used to approximate obstacles for motion planning of polygonal robots among polygonal obstacles [13]. The main disadvantage of these methods is that they require huge memory space capacity when compared with the algorithm proposed in this paper, which requires minimal storage capacity, since each cell stores information only about its nearest neighbors.

Search methods such as depth-first, breadth-first, best-first, A*, bidirectional, and random searches have been proposed mainly for path-planning under the criterion of the shortest path [11]. Algorithms for 2-D and 3-D path planning under the shortest path criterion and without any maximum clearance considerations, based also on CA architectures, have been proposed by the authors in [14] and [15].

Finally, the cell decomposition approach to path planning [16] partitions the free space into regions and identifies possible contacts (called critical contacts) between the robot and obstacles in each region. The algorithm proposed in this paper can identify critical contacts of the diamond-shaped robot with the arbitrarily shaped objects with a much lower time complexity than the $O(n^5)$ time complexity reported for the cell decomposition approach [16].

The definition of various motion planning problems and approaches together with detailed explanations of landmark papers and a comprehensive list of references can be found in [17].

III. Basic Properties of the Voronoi Diagram

A usual definition of the Voronoi diagram is given as follows. Let S denote a set of n points (called *sites*) in the plane. For two distinct points $p, q \in S$, the *dominance* of p over q is defined as the subset of the plane being at least as close to p as to q. Formally,

$$\mathrm{dom}(p,q) = \{x \in R^2 | \delta(x,p) \leq \delta(x,q)\} \tag{1}$$

where δ denotes the Euclidean distance function. Clearly, $\mathrm{dom}(p,q)$ is a closed half plane bounded by the perpendicular bisector of the line segment between points p and q. This bisector separates all points of the plane closer to p from those closer to q and will be termed the *separator* of p and q. The *region* of a site $p \in S$ is the portion of the plane lying in all of the dominances of p over the remaining sites in S. Formally

$$\mathrm{reg}(p) = \cap_{q \in S - \{p\}} \mathrm{dom}(p,q). \tag{2}$$

Since the regions are formed by the intersection of $n-1$ half planes, they are convex polygons. However, some of the regions may be unbounded. These are defined by sites lying on the boundary of the convex hull of S because just for those sites there exist points arbitrarily far away but still closest.

It should also be noted that no vertices occur if all sites in S lie on a straight line. Thus, the boundary of a region consists of at most $n-1$ edges and vertices. Each point on an edge is equidistant from exactly two sites, and each vertex is equidistant from at least three. As a consequence, the regions are edge to edge and vertex to vertex, forming a polygonal partition of the plane. This partition is called the *Voronoi diagram* $V(S)$, of the finite point set S. A more formal definition of $V(S)$ and a mathematical analysis of its properties can be found in [18]. Although the definition of $V(S)$ is usually given with respect to the Euclidean metric function, the basic properties of the Voronoi diagram are also valid with respect to more general convex distance metrics [19]. Some fundamental properties of the Voronoi diagram that are implied in this paper are: 1) finiteness, 2) connectivity, and 3) local constructibility. An analysis of these properties and formal proofs can be found in [2] and [18].

IV. Description of the CA Architecture

A. Definition of Cellular Automata

The finite automaton is a mathematical model of a system, with discrete inputs and outputs. A finite automaton is completely defined by the quintuple

$$\{I, Z, Q, \delta_1, \omega\} \tag{3}$$

where the symbols represent sets which are defined as follows [20]:

- I set of inputs meaningful to the automaton
- Z set of outputs generated by the automaton
- Q set of discrete internal states of the automaton
- δ_1 function that relates every pair of elements taken from sets I and Q, i.e., i_t and q_t, respectively, to the "next" element of Q, i.e. q_t+1.
- ω function that relates every pair of elements i_t, q_t to an element of Z, i.e., z_t.

Cellular automata are a special class of automata that can be described by the quintuple of (3), and they contain large numbers of simple identical components with only local interconnections. They consist of an n-dimensional lattice of cells, each with a finite set of possible values. A CA evolves in discrete time steps and the value taken by a particular cell (*local state*) is affected by the cell values in its nearest neighborhood on the previous time step, according to a function known as the CA rule [21]. The global state for a CA is defined as the n-dimensional vector of the local states of its cells. Cellular automata range from fine-grained CA to microprocessor arrays, according to the complexity of their cells. Some theoretical aspects and applications of CA have been presented in [21]–[24].

The CA architecture proposed in this paper consists of a 2-D Cartesian lattice of cells. Interconnections between the cells are constrained within the local neighborhood known as the 4-nn (4-nearest neighbors) or the Von-Neumann neighborhood, as shown in Fig. 1. The Von-Neumann neighborhood of a cell is defined as the set that contains the cell itself and all neighbors of that cell that lie a unit distance away from it on the 2-D grid. The rule of CA operation over the local neighborhood is given as follows:

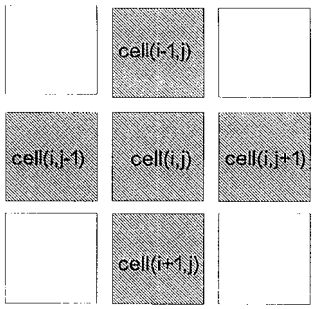


Fig. 1. Von-Neumann neighborhood.

$$z(i,j)_{t+1} = z(i-1,j)_t + z(i+1,j)_t + z(i,j-1)_t + z(i,j+1)_t + z(i,j)_t \tag{4}$$

where indexes i, j define the coordinates of a cell on the Cartesian grid, z_t is the output of a cell on time step t and operation $(+)$ is the logical OR operation. The main difference of the CA architecture proposed in this paper, relative to the binary CA proposed in [21] and [23], is that the cells can assume a finite range of values that are not constrained to the binary set. Thus, the proposed CA architecture is called a multistate CA architecture. The properties of the CA rule proposed in this paper are analyzed in the following subsections.

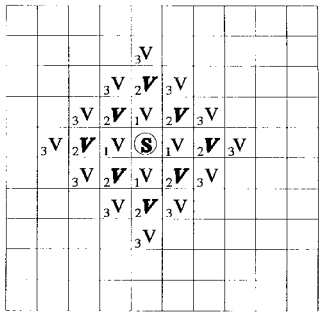
B. Expansion from a Single Cell

The effect of the application of the rule on the global state of the proposed CA architecture is straightforward, when considering time evolution of the CA, starting from an initial seed, as shown in Fig. 2(a). If a cell contains a nonzero neighbor in its nearest neighborhood on time step τ, then this cell takes the value of that neighbor, on time step $\tau+1$. Thus, a cell with value α on time step τ passes its value along to its nearest neighbors on time step $\tau+1$. Each of the nearest neighbors that received the value α on time step $\tau+1$ passes this value along to its own nearest neighborhood on time step $\tau+2$, and so on. Thus, an incremental diamond-shaped expansion effect (a square tilted by 45° clockwise with respect to the x axis) is obtained by the evolution of CA, as shown in Fig. 2(b). Fig. 2(c) displays the pseudocode for the expansion form a single cell. Although the pseudocode is shown in sequential form, the operations are performed in parallel across the 2-D CA grid.

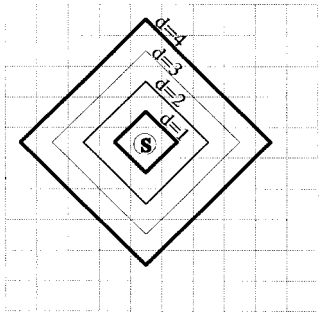
Since only unit steps are taken on each time step, the incremental expansion from a single cell (source) on the 2-D grid identifies the cells that are equidistant from the source cell, and the distance metric is a function of the number of time steps τ of CA evolution. Assuming that the source cell X lies at coordinates (x_i, x_j), then the distance between the source cell and a point Y, at coordinates (y_i, y_j) on the grid that is being reached on time step τ, is given by the following expression:

$$d(X,Y) = \tau = |x_i - y_i| + |x_j - y_j|. \tag{5}$$

The cells that satisfy this condition lie on an isometric diamond-shaped curve, which is defined by the Minkowski



(a)



(b)

Pseudo-code for the expansion from a single cell

```
# 1. Load Source value V on the cell at position k,l on the CA grid.
    Set its flag

C(k,l)=V;
Flag(k,l)=1;

#2. Repeat for a number of time steps of CA evolution, up to Testperiod

For t=1 to Testperiod do
begin

      #3. Repeat across the CA grid of dimensions mxm

      for i=1 to m do
      begin
            for j=1 to m do
            begin

            #4. Check the flags of the cells that belong in the
            Von-Neumann Neighbourhood

                  If Flag(i,j+1) OR Flag(i,j-1) OR Flag(i+1,j) OR
                  Flag(i-1,j)=1 then
                  begin

                  #5. If the flag of a neighbour is set, then pass source
                  value onto the current cell and set its own source flag
                  so as to act as a new source cell.

                  C(i,j)=V;
                  Flag(i,j)=1;

                  end;
            end;
      end;
end;
```

(c)

Fig. 2. (a) Cellular automaton evolution from a single source. (b) Diamond-shaped isometric curves. (c) Pseudocode for the expansion from a single cell.

p-metric [25]

$$\rho_p(X,Y) = \left[\sum_{i=1}^{d} |x_i - y_i|^p\right]^{1/p} \tag{6}$$

for d (dimensionality of space) $= 2$ and for $p = 1$. Fig. 2(b) provides an example of distance measurement on the 2-D grid according to the $\rho 1$ metric (also known as the Hamming metric).

Thus, CA evolution on time step τ identifies the locus of points $V(X)$ defined [26] as

$$V(X) = \{Y_k | \rho_1(X,Y)k) \leq \tau \text{ for } k = 1 \text{ to } n\} \tag{7}$$

with X being the source cell and n the number of points on the grid (denoted as Y_k, for $k = 1$ to n), and the value α of the source cell is passed along to these points on successive time steps, until time step τ is reached.

C. Expansion from Multiple Source Cells

If more than one cell, with either the same or a different value, are set as initial source cells, then CA evolution will have the effect of expanding independent diamond-shaped isometric curves relative to each source cell. These curves, initiated at different source cells, will eventually meet as τ increases.

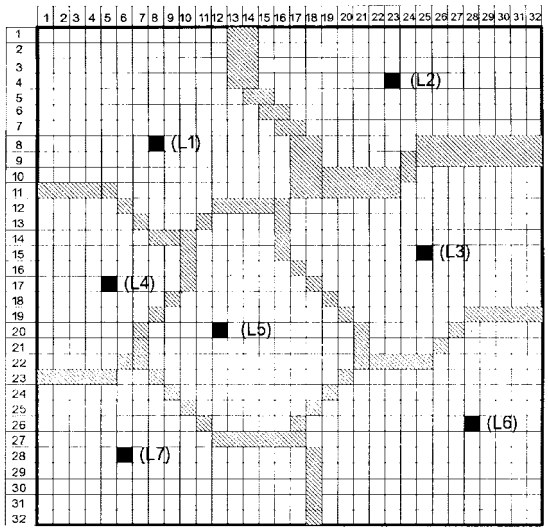
In this case, there exist cells, on the 2-D grid, that lie on more than one isometric curves at the same time. These cells receive simultaneously the values of more than one source points from different directions. Since the 4-nn rule is used for local interconnection between cells, a cell can receive a value from four possible directions on the 2-D grid: from its north neighbor, its south neighbor, its east neighbor, and its west neighbor. For cells of the CA that simultaneously receive a value from more than one of their nearest neighbors, a special flag is set to denote that these cells lie on the intersection of more than one isometric curves, and further expansion is disabled. Since the isometric curves expand on successive clock cycles according to the metric defined by (5), the cells that lie on the intersection of these curves on time step τ will be equidistant according to the $\rho 1$-metric, from the corresponding source cells, since their distance from each of the source cells will be equal to τ. For increasing values of τ, these points lie on a planar skeleton formed by the intersecting boundaries of the isometric curves, as shown in Fig. 3(a). For an odd $\rho 1$ distance between source cells, the isometric curves intersect on a single cell (since a "middle" exists). For an even $\rho 1$ distance between source cells, the isometric curves intersect on the boundary of two cells, as shown in Fig. 3(a).

For a set S of n source cells $c_i (i = 1, \cdots, n)$ on the 2-D grid, the polygonal planar skeleton formed by CA evolution around c_i is defined as

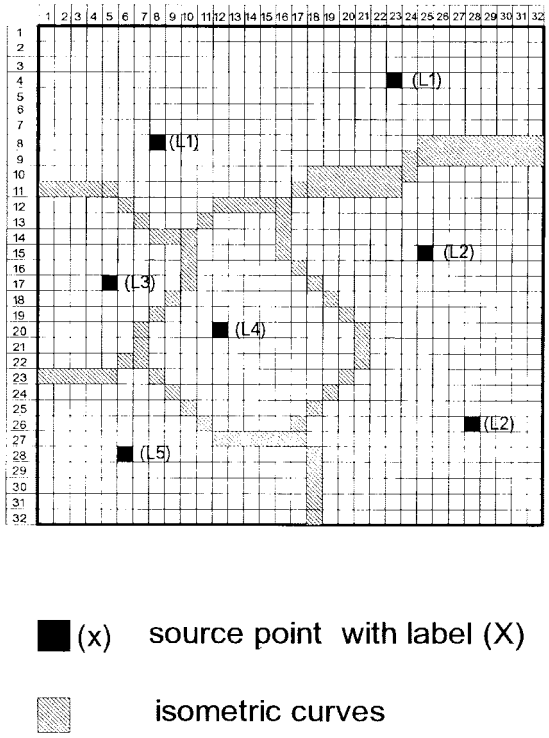
$$L(c_i) = \{c | \rho 1(c, c_i) \leq \rho 1(c, c_j) \text{ for all } j \neq i\} \tag{8}$$

and $L(c_i)$ is the locus of cells which are closer to c_i than to any other cell of S, according to the $\rho 1$ distance metric. This definition is identical to the definition of the Voronoi diagram under the $\rho 1$ metric [19], [26].

Assuming that m^2 cells are available for an mxm grid and since the expansion of the isometric curves is performed in



(a)



(b)

Fig. 3. (a) Polygonal planar skeleton formed by the intersection of isometric curves initiated at different source points. (b) Merging of loci initiated at source points of the same value.

parallel on the 2-D grid, the construction of the $\rho 1$ Voronoi diagram is performed in $O(m)$ and, thus, it is a much simpler technique than the tree-bucketing method [27] or the divide-and-conquer technique [28], both used for the establishment of Voronoi diagrams.

Furthermore, it is required for the path planning process to merge the loci of source points of the same value, as it will be explained in the following section. For two points c_i, c_j of the same label value, a new locus of points is defined as

$$L'(c_i, c_j) = L(c_i) \cup L(c_j) \tag{9}$$

and $L'(c_i, c_j)$ is the locus of cells which are closer either to c_i or to c_j than to any other cell of S, according to the $rho1$ distance metric. Fig. 3(b) displays the result of merging some of the loci shown in Fig. 3(a), initiated at source points of the same label value.

Thus, a special class of $\rho 1$ Voronoi diagrams is constructed, including discriminant curves between the loci of different source points. For this purpose, comparison logic is added on each cell of the CA. This logic is able to identify, for a cell that receives more than one value from its nearest neighborhood, whether these values are identical to each other or they are different. If these values are identical, then the cell assumes the specific value, and this leads to merging of loci initiated at source cells of the same value. If, however, these values are different, a flag is set to denote that the cell belongs on the planar skeleton of the $\rho 1$ Voronoi polygon.

V. Description of the Path Planning Algorithm

The principle aim of the proposed algorithm is the efficient construction of the Voronoi diagram for a static model of the environment through the time evolution of 2-D CA. In order to map the proposed algorithm onto the 2-D CA architecture, the following assumptions were made.

1) The proposed algorithm is focused solely on the most computationally intensive part of the path planning process, i.e., the construction of the Voronoi diagram. Higher level path planning tasks such as image acquisition, path representation, etc., are not considered in the implementation of the algorithm since they would lead to a far more complex architecture.
2) For the purposes of the proposed algorithm, it is assumed that a model of the static environment, in the form of a binary image, has been acquired by means of an appropriate imaging device (camera, CCD, etc.), and it has been stored in external memory. The contents of this memory are used to initialize the 2-D CA architecture.
3) The results of the algorithm, i.e., the output of the 2-D CA architecture, are also stored in external memory to be further processed by graph searching techniques, topological algorithms, etc., as required for higher level robot path planning tasks.

The proposed algorithm for path planning is divided in two phases of operation. Points that lie in the periphery of the binary representation of each of the objects (boundary points) are identified during the first phase. These points constitute the initial set of source points and CA evolution in time establishes the $\rho 1$ Voronoi diagram for this set, during the second phase. The phases of operation of the proposed algorithm are presented in the following subsections.

A. Object Boundary Detection

In static motion planning in a well-known environment, a robot usually has a model of objects in its environment before

it can plan a collision-free motion. For on-line and reactive motion planning in an unknown or a dynamic environment where no model is available, objects usually have to be sensed in an appropriate time interval. Typically, visual sensors, or range sensors based on sonar, infrared, or laser light, are used. In a well-known environment, object data are usually available in the form of CAD data. Once the information about shapes and configurations of objects is acquired, it can be represented in a number of ways. Commonly used representations [11] are grid, cell-tree, polyhedral, constructive solid geometry (CSG), and boundary representation (B-rep).

A boundary representation approach for the modeling of objects is developed in this paper. This approach has the advantage that it does not require computations of intersection and distance, as required by polyhedral representations [29], nor the computation of unions, intersections, and set differences of primitive shapes, as required by solid geometry modelers [30]. Additionally, since a rectilinear grid is used for cellular automata operations, the proposed algorithm avoids the overhead of computing adjacency between cells as it is required in cell-tree representations [12].

Since objects and free space are represented with different binary values, a simple edge-detection process is sufficient for the detection of boundary points. The parallel edge-detection algorithm proposed in this paper consists of a 2-D CA operation on a Von-Neumann type of neighborhood. Here, it is assumed that the image has been mapped onto a 2-D CA of the same dimensions. The concepts used in this paper are directly related to the "template matching" and "compass gradient" operators, frequently used for edge detection [31], [32], applied here to binary images.

Consider the configurations of binary values (called templates) of the cells of the 4-nn neighborhood shown in Fig. 4(e)–(l). Each of these configurations is coded with respect to the edge direction implied by the change of binary values, and the resulting coding scheme identifies and codes the boundary points of each of the objects. Edge directions are quantized into one of eight directions on the 2-D CA grid in accordance with the "compass" operators for edge detection. Whenever there exists a match between the 4-nn neighborhood of a cell of the 2-D CA and one of the templates, the cell assumes the edge-direction code specified by that template.

The implementation of the boundary detection and coding process on the 2-D CA can be described as follows.

Each of the templates is implemented as a CA local rule on a 4-nn neighborhood, requiring only logical "and" and "not" operations. The template shown in Fig. 4(e) (denoted as t_e for simplicity), for example, is implemented as

$$t_e(i,j) = z(i-1,j) \cdot \overline{z(i+1,j)} \cdot z(i,j-1) \cdot (i,j+1) \cdot z(i,j) \tag{10}$$

where indexes i, j define the coordinates of a cell on the CA grid, z is the cell output, operation $(\cdot)$ is the logical AND operation and the bar denotes the logical NOT operation. All rules that correspond to the different templates are stored in each of the cells of the 2-D CA. Due to symmetries of the configurations, optimization of the logic operations reduces considerably the number of logical operators required for the implementation of the set of templates in each cell. Whenever there exists a match between the values of a 4-nn neighborhood and a specific template t_n, then $t_n = 1$, otherwise $t_n = 0$. The template values in each cell form a vector of the form $T = \{t_a, t_b, \cdots, t_l\}$ and coding of this vector produces the edge code for the cell.

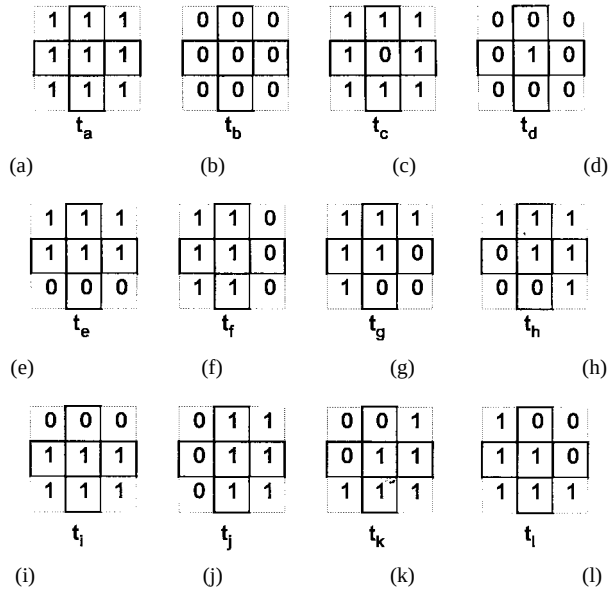


Fig. 4. Proposed set of templates for the coding of edge directions.

It is clear that templates t_a, t_b, corresponding to the configurations shown in Fig. 4(a) and (b), respectively, are set when applied to homogeneous areas of the objects or the free space and, thus, they are of no interest for the boundary detection process. Additionally, templates t_c, t_d, corresponding to the configurations shown in Fig. 4(c) and (d), respectively, are set in cases which are considered as manifestations of a type of noise induced on binary images, known as "salt and pepper noise," and they are also discarded.

The confidence level associated with the assignment of a cell to an edge direction using a 4-nn neighborhood is lower than the corresponding confidence level produced when using an 8-nn neighborhood, due to the lower neighbor vote count. However, it was found that this has only marginal effect on the performance of the path planning algorithm, as indicated by the simulation results presented in Section VI, whereas, on the other hand, the adoption of the same type of neighborhood for CA operations both for the edge-detection algorithm and for the expansion algorithm reduces considerably the overall design complexity. Fig. 5 displays the pseudocode for the object boundary detection. Although the pseudocode is presented in sequential form, the operations are performed in parallel across the 2-D CA grid.

In the proposed algorithm, every edge of each of the objects is considered to be a distinct feature that contributes to the Voronoi diagram and, thus, the Voronoi diagram constructed during the second phase of the proposed algorithm is more accurate than that obtained when the whole of the boundary of each of the objects is considered as a single feature [11] (since intra-object features are also taken into consideration).

Pseudo-code for object boundary detection

```
#1. Repeat across the CA grid

For i=1 to m do
begin
      for j=1 to m do
      begin

      #2. Apply the vector of template values T

      t_a(i,j)=z(i-1,j) AND z(i+1,j) AND z(i,j-1) AND z(i,j+1) AND z(i,j);
      t_b(i,j)= ...
      ...
      ...
      t_e(i,j)=z(i-1,j) AND (NOT(z(i+1,j))) AND z(i,j-1) AND
            z(i,j+1) AND z(i,j);
      ...
      ...
      ...
      t_l(i,j)= ...

      T={t_a,t_b,t_c,... t_l};

      #3 Code edges with respect to vector T

      if T={1,0,0,...0} then edge_code(i,j)=1;
      if T={0,1,0,...0} then edge_code(i,j)=2;
      ...
      ...
      ...
      if T={0,0,0,...1} then edge_code(i,j)=12;

      end;
end;
```

Fig. 5. Pseudocode for the object boundary detection.

B. Construction of the Voronoi Diagram

Once the initial sites are determined by the CA operation that implements the edge-detection process, they are loaded as the initial source points onto the CA that establishes the Voronoi polygon, and their coded value is loaded as their initial source value. The CA starts its evolution in time and the expanding wave fronts initiated at the source points intersect with each other for increasing values of τ, following (5) and (8). Thus, the Voronoi diagram, with respect to the initial sites, is established as presented in Section IV-C. The loci of neighboring points with the same coded value are merged according to (9). This is due to the fact that consecutive neighboring points of the same coded value correspond to edge points lying on the same direction and, thus, they are considered to belong on the same "side" of the object, whereas the path planning process requires the examination of at least two different boundaries of objects in order to establish an acceptable point.

In order to allow the determination of the maximum dimensions of the diamond-shaped robot that can be placed at every point of the Voronoi diagram, a special label is associated with every cell of the proposed CA architecture. This label stores the time step of CA evolution, τ_c, during which the cell was found to belong to the Voronoi diagram. This label is produced by a counter variable that is triggered by the system clock and it is set to zero for cells that do not belong to the Voronoi diagram. The labeling process is part of the Voronoi diagram construction process, since a label is assigned to a cell at the same time step of CA evolution during which it is determined that the specific cell belongs to the Voronoi polygon. Thus, a special type of Voronoi diagram, the "labeled Voronoi diagram," is proposed in this paper. Fig. 6 displays the pseudocode for the construction of the labeled Voronoi diagram. Although the pseudocode is presented in sequential form, the operations are performed in parallel across the 2-D CA grid.

Pseudo-code for the construction of the Voronoi diagram

```
#1. Procedure for loading the edge_codes from the object boundary detection
phase (edge_code values <5 are disregarded since they correspond either to
homogeneous areas or noise).

# Repeat across the CA grid

for i=1 to m do
begin
      for j=1 to m do
      begin

            #Load edge_code as the cell value and set its flag

            if edge_code(i,j)>4 then
            begin
                  C(i,j)=edge_code(i,j);
                  Flag(i,j)=1;
            end;
      end;
end;


#2. Procedure for the implementation of expansion around
    multiple source points.

# Repeat for a number of time steps of CA evolution equal to Testperiod

For t=1 to Testperiod do
begin

      #Repeat across the grid

      for i=1 to m do
      begin
            for j=1 to m do
            begin

            #Code the flags in the neighbourhood

            ni=Flag(i-1,j); si=Flag(i+1,j);
            wi=Flag(i,j-1); ei=Flag(i,j+1);

            # If only one flag is set in the neighbourhood of the cell then we
            have expansion in free space. Current cell receives the value of
            the neighbour whose flag is set.

            If ni=1 AND si=0 AND wi=0 AND ei=0 then
            C(i,j)=C(i-1,j);
            if si=1 AND ni=0 AND wi=0 AND ei=0 then
            C(i,j)=C(i+1,j);
```

(a)

```
            if wi=1 AND ni=0 AND si=0 AND ei=0 then
            C(i,j)=C(i,j-1);
            if ei=1 AND ni=0 AND si=0 AND wi=0 then
            C(i,j)=C(i,j+1);

            #If more than one flags are set in the neighbourhood, then the
            edge_code values of the cells that contain these flags are
            compared to each other.

            If ni=1 AND ei=1 AND (C(i-1,j) ≠ C(i,j+1)) then

            #Cell belongs on the Voronoi diagram. The time step value is
            stored as the Voronoi label Vor(i,j).

            Vor(i,j)=t;

            If ni=1 AND wi=1 AND (C(i-1,j) ≠ C(i,j-1)) then
            Vor(i,j)=t;
            ...
            ...
            ...
            #similar comparisons for the rest of the neighbour combinations

            end;
      end;
end;
```

(b)

Fig. 6. Pseudo-code for the construction of the labeled Voronoi diagram.

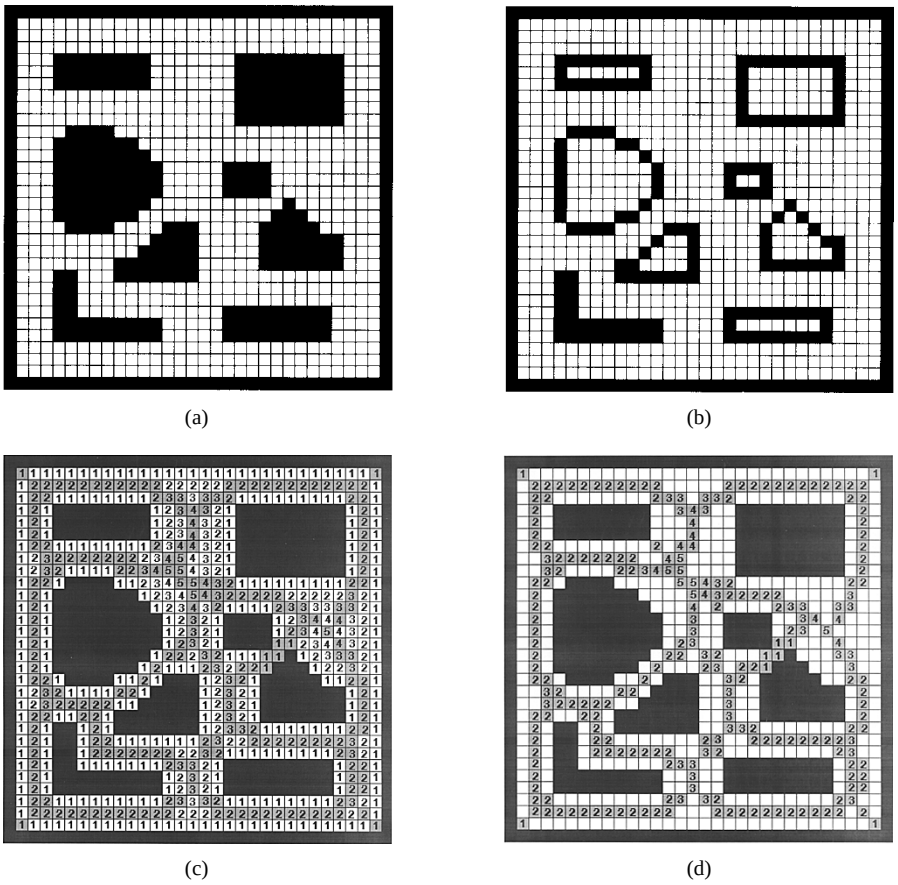


Fig. 7. Characteristic example, demonstrating the operation of the algorithm under the set of criteria described in Section VI. (a) Configuration of obstacles. (b) Boundary detection phase. (c) Timing information for CA expansion. (d) Collision-free path.

In order to demonstrate the effect of keeping as far away from obstacles as possible, according to the $\rho 1$ metric, consider a hypothetical inversion of the labeling process. Each cell that belongs to the labeled Voronoi diagram will be considered as a new source cell and its value will be set to "1." The CA will start its evolution in time, as described in Section IV-B, but the number of time steps for the incremental expansion around each of the source cells, according to (5), will be controlled by the label τ_c that is stored in it. Thus, each of the cells that lie onto the labeled Voronoi polygon will act as new source cell and expansion will be allowed around them for a number of time steps equal to the label value that is associated with each one of them, respectively.

Assuming that a label τ_1 is associated with a specific source cell, then CA evolution will identify the locus of points $V(X_1)$, defined as

$$V(X_1) = \{Y_k | \rho_1(X_1, Y_k) \leq \tau_1 \text{ for } k = 1 \text{ to } n\} \qquad (11)$$

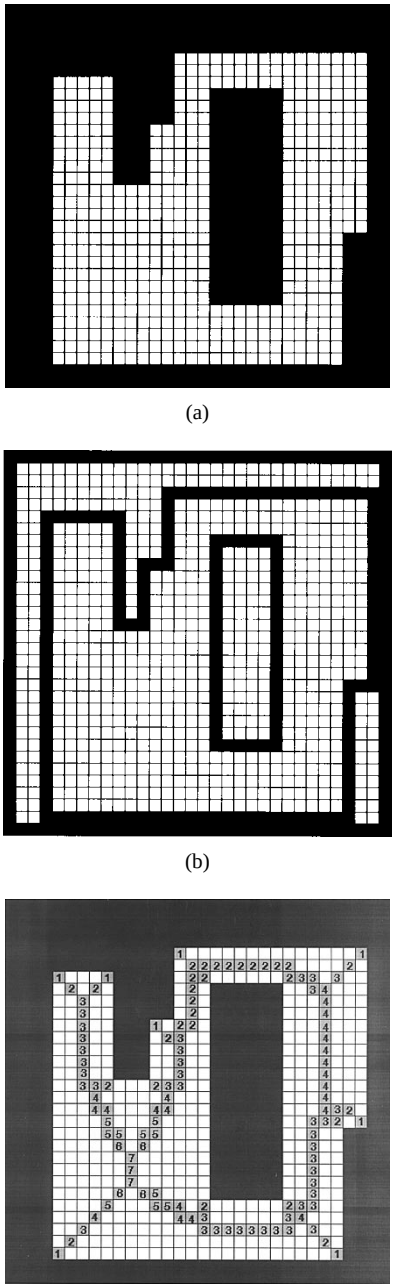
with X_1 being the source cell and for n points on the grid (denoted as Y_k, for $k = 1$ to n), and the value "1" will be passed along to these points on successive time steps, until time step τ_1 is reached. This locus will have a diamond-shaped form (a square tilted by 45° clockwise with respect to the x axis), as described in Section IV-B and it is shown in Fig. 2(b). For the rest of this paper, it is assumed that a robot can be totally enclosed within such a shape, and thus, the path planning procedure is referred to as path planning of a diamond-shaped robot.

For s points belonging to the labeled Voronoi diagram, the free space F can be covered by the union of loci of the form presented in (11) as

$$F = \bigcup_{m=1}^{s} V(X_m). \qquad (12)$$

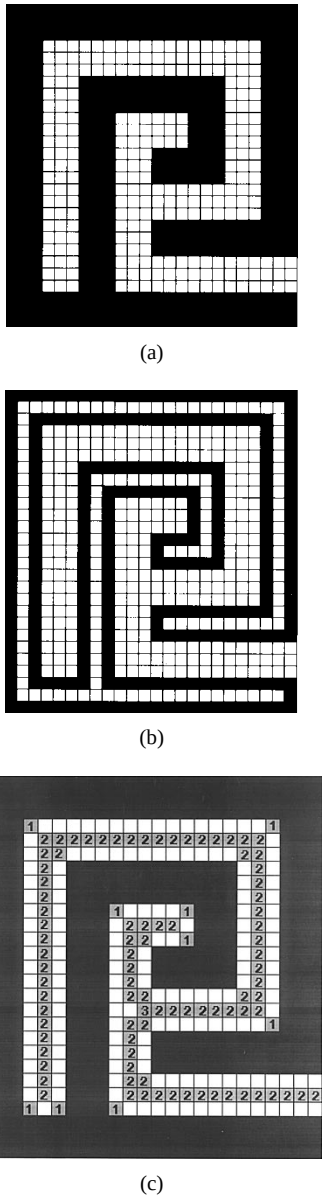
Thus, the Voronoi diagram provides a way to compute "chunks" of free space. Computations of "chunks" of free or occupied configuration space have been presented in [33], but the main difference with the proposed algorithm is that they consist of a partitioning of free space in spheres. This, due to the fact that the Euclidean metric was used in [33], results in circle calculations, whereas the $\rho 1$ (or $L1$) metric proposed in this paper result in diamond-shaped calculations. An advantage of the proposed algorithm is that diamond shapes can fill free space without cracks, which Euclidean spheres cannot do [11].

The geometry of the workspace is restricted in the sense that a generally shaped robot is circumscribed by a diamond-shaped figure and paths feasible for this figure are also feasible for the robot. However, this restriction is not critical since the proposed path is suitable for any arbitrarily shaped robot that can be enclosed within the diamond-shaped figure and it also provides a good generalization on the results of the proposed algorithm, since one cannot take into consideration the specific geometrical structure for every particular robot. Moreover, similar restrictions on the shape of robots are



(a) (b) (c)

Fig. 8. Characteristic example, demonstrating the operation of the algorithm under the set of criteria described in Section VI. (a) Configuration of obstacles. (b) Boundary detection phase. (c) Collision-free path.



(a) (b) (c)

Fig. 9. Characteristic example, demonstrating the operation of the algorithm under the set of criteria described in Section VI. (a) Configuration of obstacles. (b) Boundary detection phase. (c) Collision-free path.

commonly used in order to generalize the results of robot path planning algorithms, as, for example, in [34], where the robot is enclosed within a disk, or in [35], where the translational motion of a robot enclosed in a convex planar figure is presented.

Once the Voronoi diagram is established, the path planning process consists of the movement of the center of the diamond-shaped robot along the edges of the diagram. For a diamond shape of diagonal size d, the path planning process selects those Voronoi edges that consist of points with labels of value $l \geq d + 1/2$, whereas edges with label values $l < d + 1/2$ are rejected, since on these edges the diamond-shaped robot will collide with objects ($\rho 1$ distance between point of Voronoi diagram and an object boundary is less that the half diagonal of the diamond shape). It must be noted that a canonical procedure, which will move the diamond-shaped robot from any given start and goal points of unoccupied space onto edges of the Voronoi diagram, is required. However, it has been shown [11] that a straight-line movement in one of the coordinate axes usually suffices to solve this problem. Additionally, the fundamental properties of the Voronoi diagram [2] guarantee that the proposed path is finite (consists of a finite number of cells), connected (cells that belong to the path form a connected diagram), and locally constructable.

Assuming that the maximum $\rho 1$ distance between different sides of objects is equal to D, and taking into account the fact that CA expansion is implemented by parallel incremental operations on the 2-D CA grid, as described by (5), for successive time steps, the maximum number of time steps required for the establishment of the Voronoi diagram is equal

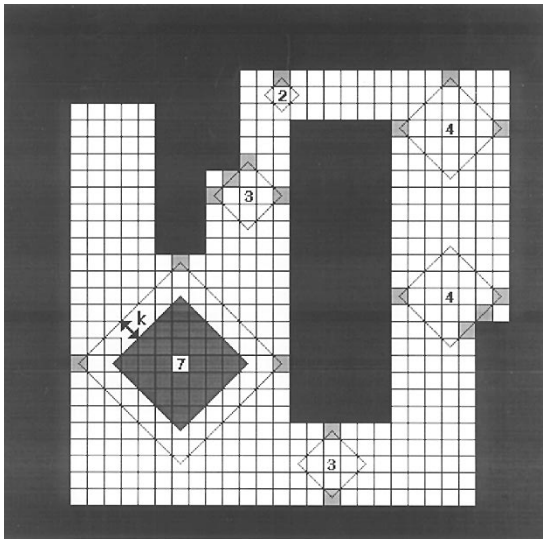


Fig. 10. Critical contacts and safety margin corresponding to Fig. 8(c).

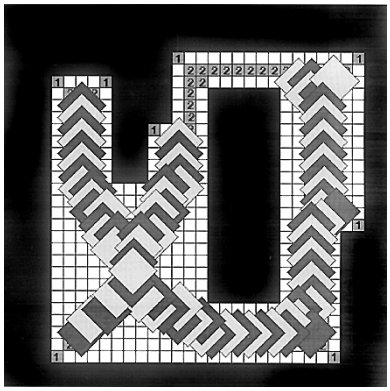


Fig. 11. Path planning for a diamond-shaped robot of diagonal length $d = 5$, along the labeled Voronoi diagram of Fig. 8(c).

to $D/2$ (wavefronts initiated simultaneously on both different sides). Additionally, the boundary detection phase requires a constant amount of time for its completion (a single time step). Thus, the worst-case running time for the algorithm on a $m \times m$ CA grid is $O(m)$ whereas, for an implementation of the algorithm on a sequential machine the complexity becomes $O(m^3)$, since there exist m^2 grid points and the algorithm takes $O(m)$ time for a signal to propagate across the grid.

However, the dimensions of the CA grid would have to increase with increasing complexity of the workspace. For discrete imaging applications, it is often required to process grids with $m = 256$ or 512. Since the proposed architecture is implemented in VLSI, its dimensions are fixed, and due to the limitations of the current VLSI technology, a chip array, consisting of identical chips (possibly implemented in wafer scale integration), must be used in order to process arbitrarily large workspace dimensions in parallel, as it will be discussed in Section VII of the paper. Alternatively, a CA architecture of fixed dimensions can be used to operate sequentially on blocks of the workspace with the same dimensions. However, this would result to a considerable increase in the time required to process the overall workspace.

It is obvious that more than one path may exist for a diamond-shaped robot and the optimality criterion for the proposed algorithm is the maximum safety clearance from obstacles rather than the path length (CA-based algorithms for path planning under the minimum length criterion have been proposed by the authors in [14] and [15]).

The proposed algorithm is able to identify critical contacts [16] between the diamond-shaped robot and any obstacle. In the context of this paper, a critical contact is defined as any point contact between the points on the periphery of the diamond-shaped robot and any point on the periphery of any of the obstacles. Critical contacts for a diamond-shaped robot of diagonal d occur at points of the Voronoi diagram with label value l such that

$$d = 2 \cdot l - 1 \tag{13}$$

(the point on the Voronoi diagram is equidistant, according to the $\rho 1$ metric, from object boundary points that are separated by a distance d). Thus, critical contacts are identified by the labeling process, without any additional cost. This is a considerable advantage of the proposed algorithm, since the complexity associated with the detection of critical contacts in 2-D space is usually high (complexities of $O(n^5)$ have been reported, for n obstacle edges [11]).

The notion of allowing a minimum safe distance between the robot and the obstacles, called a safety margin [36], can be applied to the proposed algorithm without any additional cost in complexity. Let k denote the minimum safe $\rho 1$ distance required between the robot and the obstacles. Then, for a diamond-shaped robot of diagonal d, the choice of a path along the Voronoi diagram will be altered as follows. Voronoi edges that consist of points with label values will be followed, whereas edges that include points with label values will be rejected.

A well-known problem in binary image applications is that of the discretization of the plane on a grid. Due to the finite number of cells of the 2-D CA architecture and also due to the presence of noise during the image acquisition phase, some of the obstacle points are expected to be misplaced from their true positions. This gives rise to model uncertainty. The concept of a safety margin can be used to overcome this problem. The proposed algorithm accounts for such uncertainties by imposing a safety margin on the path which will keep the robot clear from problem regions.

An application-specific integrated circuit (ASIC) was designed and implemented for the proposed algorithm and, thus, execution speed for the construction of the Voronoi diagram is very high. This is considered to be a significant advantage, since an efficient and fast static motion planner is a vital ingredient of a dynamic motion planner in partially known environments [4], in path planning of multiple robots with conflicting/common objectives, where robots of higher precedence are regarded as moving obstacles for robots of lower precedence [5], [6], and in time-varying environments [7], [8]. A Voronoi planner of the type proposed in this paper is particularly required in the case where many motion planning problems,

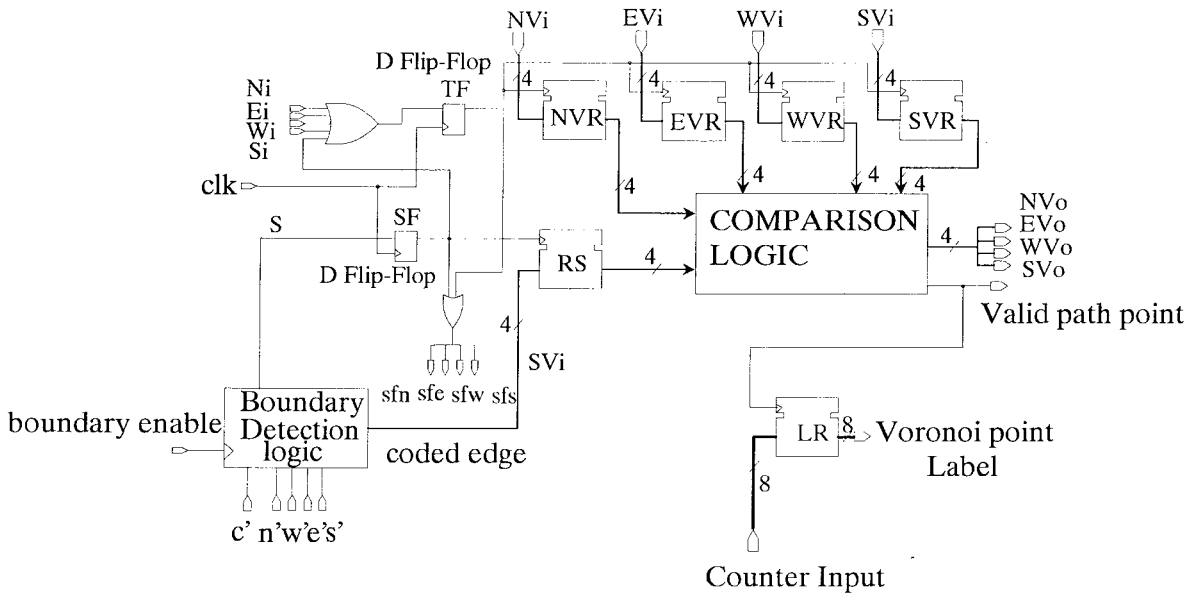


Fig. 12. Typical cell of the proposed 2-D CA architecture.

with different start and goal configurations within the same environment, are to be solved and also in the case of a marginally changing environment, as stated in Section II of this paper.

VI. Simulation Results

The simulation examples used to evaluate the performance of the proposed algorithm follow a set of criteria adopted by many researchers [11].

For the first example, the number of obstacles is eight (this number should be kept between five and ten, as suggested in [11]) and some of the obstacles are concave, as shown in Fig. 7(a). Fig. 7(b) displays the effect of the boundary detection phase of the proposed algorithm, as described in Section V-1, Fig. 7(c) displays all timing information for the CA evolution around all of the obstacles, and Fig. 7(d) shows the solution to the collision-free path planning problem, as a result of the application of the proposed algorithm. The algorithm is applied on a 32×32 grid, where cells occupied by obstacles are represented by black color, whereas cells that belong to free space are white. The algorithm establishes the labeled Voronoi diagram, as described in Section V-B, and cells that belong to the diagram are represented by gray color and their label values are written inside each of them, respectively, as shown in Fig. 7(c). Each point of the labeled Voronoi diagram is equidistant from the closest object boundary points of different coded edge directions, as described in the previous section. Label (l), stored in each cell of the labeled Voronoi diagram, determines the maximum diagonal length for a diamond-shaped robot whose center point can be placed on that cell. Following (13), $d = (2 \times l - 1)$ is the maximum diagonal length for a diamond-shaped robot that can be centered on the cell without colliding with any of the object boundary points (but at critical contact with some of them), as described in the previous section.

The second example, shown in Fig. 8(a), includes a narrow space at some point along the solution path, as suggested in [11]. The result of the boundary processing phase is shown in Fig. 8(b). Fig. 8(c) displays the solution path, as it was established by the proposed algorithm. It is clear that a diamond-shaped robot of diagonal length $d = 5$ cannot be allowed to move along the top part of the diagram that consists of a sequence of labels equal to two, because collisions will occur since $d > 2 \cdot l - 1$, as described in the previous section. Similarly, collisions will occur if a diamond-shaped robot of diagonal length $d = 7$ were to be moved along the upper left part of the diagram, consisting of a sequence of labels smaller or equal to three.

The third example, shown in Fig. 9(a), includes a trap in the problem space (free space close to the center is surrounded by boundaries), so that backtracking is required in order to find a solution path for some pairs of points, as suggested in [11]. Fig. 9(b) displays the result of the boundary processing phase, whereas Fig. 9(c) shows the labeled Voronoi diagram for the problem, i.e., the solution path that includes the necessary backtracking path from the trap.

Fig. 10 illustrates the notions of critical contacts and safety margin, as described in the previous section. Some of the points that belong to the labeled Voronoi diagram of Fig. 8(c) are chosen randomly, and they are shown together with their diamond shapes, whose diagonal length is determined by the respective label value. Points of critical contact between the diamond-shaped robot and an obstacle are filled in light gray. A safety margin with $k = 2$ is imposed to the point with label value $l = 7$. Points that lie within this safety margin are shown in darker gray.

Fig. 11 displays all possible positions of a diamond-shaped robot of diagonal length $d = 5$, along the labeled Voronoi diagram of Fig. 8(c) (allowing critical contacts). Clearly, such a diamond-shaped robot cannot move along the uppermost part of the diagram.

It is clear that the proposed algorithm operates successfully for all of the examples and, thus, produces a solution path under all the above-mentioned criteria of difficulty for a path planning process, as proposed in [11]. Several additional

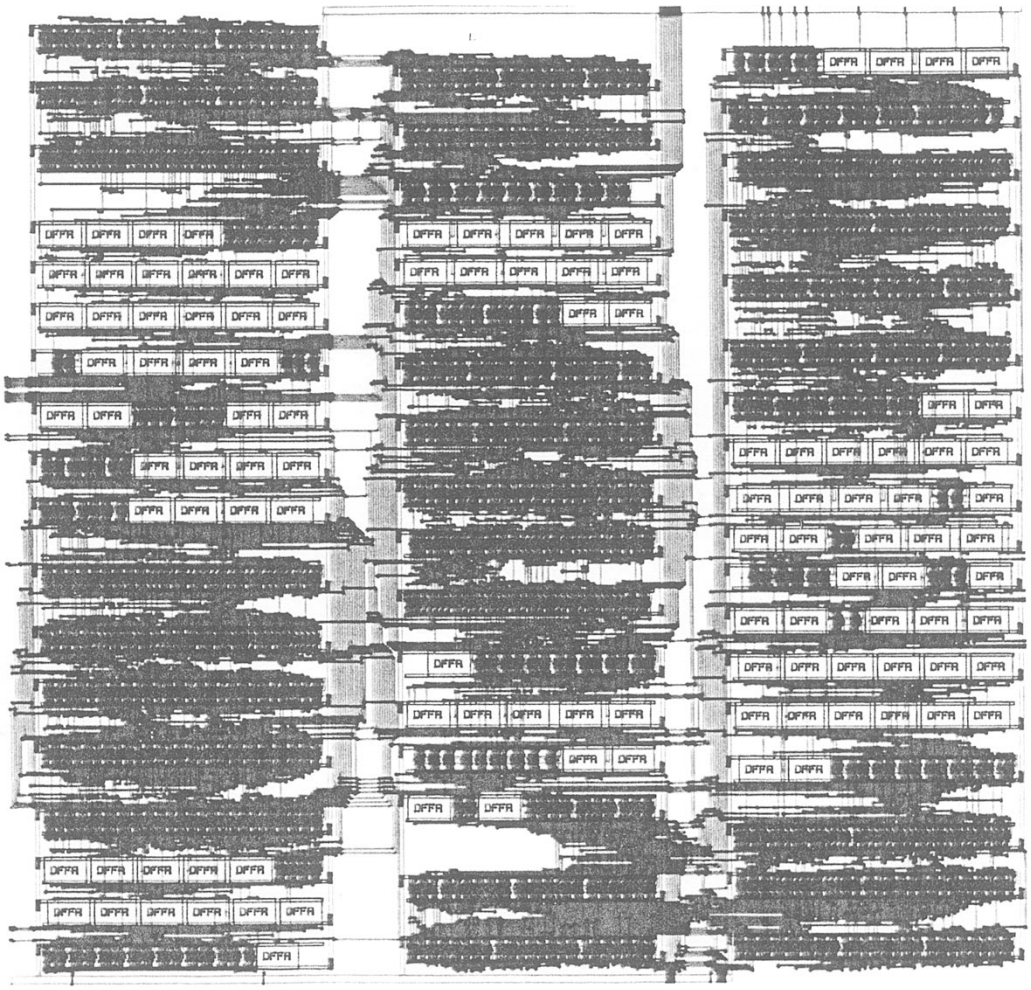


Fig. 13. Layout of the chip.

examples, of varied difficulty (the number and shapes of objects were varied) were used, and the proposed algorithm was successful in producing collision-free paths under all circumstances.

VII. VLSI Implementation

A typical cell of the proposed CA architecture is shown in Fig. 12. The boundary detection logic implements the set of templates described in Section V-A, based on the pixel values of the cell itself, c', and the pixel values of its 4-nn neighbors, n', w', e', s', respectively, and it produces the coded edge output signal SVi, as well as the valid source signal S. Signals S and SVi are produced during the first phase of the proposed algorithm which is controlled by the *boundary enable* signal. Signal S is set to 1 if the cell is a valid source cell, i.e., an edge cell. This value is loaded into latch SF and it is communicated to the cell's nearest neighborhood by means of the signals sfn, sfe, sfw, and sfs, which are connected to the cell's north, east, west, and south neighbors, respectively. Whenever S is set, the source cell coded edge value (SVi) is loaded into register RS. Labels Ni, Ei, Wi, and Si denote the inputs to the cell coming from its nearest neighborhood, i.e., from the north, east, west, and south neighbor's, respectively. Signal Ni is connected to the cell's north neighbor's signal sfs, signal Ei is connected to the cell's east neighbor's signal sfw, signal Wi is connected to the cell's west neighbor's signal sfe, and signal Si is connected to the cell's south neighbor's signal sfn. The communication of these signal values is achieved on successive clock cycles through latches SF and TF. Whenever one of the inputs Ni, Ei, Wi, and Si is set to one, latch TF is set and it enables the loading of registers NVR, EVR, WVR, and SVR with the coded edge values of the cells in the nearest neighborhood, i.e., NVi, EVi, WVi, and SVi. These values are compared with each other in the comparison logic of the cell, and if it is found that only one of them is nonzero, then this is the value stored in the current cell. If more than one of these values are different from zero, but they are identical to each other, then their value is stored in the current cell. The current cell value is communicated to the cell's nearest neighborhood through signals NVo, EVo, WVo, and SVo. Signal NVo is connected to the SVi of the cell's north neighbor, signal EVo is connected to the WVi of the cell's east neighbor, and similarly for the rest of the signals. If, however, more than one of these coded edge values are different from zero and they differ from each other, a special value is stored in the current cell to denote that the cell belongs to the $\rho 1$ Voronoi polygon, as described in Section V-B and the *valid path point* signal is set to one. The *valid path point* signal triggers latch LR which stores the point label deduced from the counter input, as described in Section V-B.

The incremental expansion of the isometric curves, described in Sections IV-B and IV-C, is achieved through the SF and TF latches and the OR gates, on successive clock cycles. Assuming that SF is set in a cell on clock cycle τ, then its value is transmitted through sfn, sfe, sfw, and sfs to the appropriate Ni, Ei, Wi, and Si inputs of its nearest neighborhood. Thus, the five-input OR gates of the cells in the nearest neighborhood will have output 1 (one input in each OR gate of the cells in the nearest neighborhood is set) on clock cycle τ. On clock cycle $\tau + 1$, the TF latch will be set for each of the nearest neighbors and it will also set their sfn, sfe, sfw, and sfs variables. These signals will activate the Ni, Ei, Wi, and Si signals of the nearest neighborhood of each of the source's nearest neighbors, and so on.

A Von-Neumann neighborhood processor was implemented on a single VLSI chip using a 1.2-μm double-layer metal CMOS technology and the Solo 1400 design tools provided by European Silicon Structures (ES2). The processor consists of five identical cells, of the type shown in Fig. 12, located at the respective positions of the Von-Neumann neighborhood shown in Fig. 1, with the appropriate local interconnections established between them and with the necessary control signals. Layout of the chip is shown in Fig. 13. The dimensions of the chip are 2.35 mm $\times$ 2.29 mm $=$ 5.38 mm^2 and the maximum frequency of operation, obtained after loaded simulation, is 60 MHz.

The standard cell design approach was used for the implementation of the chip. Cellular automata are particularly suited for such a VLSI implementation due to their inherent parallelism, the high regularity of their structure, and the locality of their interconnections which are only to nearest neighbors, thus avoiding the need for long feedback links. In this approach, a basic module is designed initially, and it is then repeated in order to implement the overall architecture. The modules are then placed together and the necessary routing of interconnection signals is established among them. In the case of the proposed architecture, a number of different placement and routing configurations were tried out in order to obtain the smallest possible dimensions for the chip. However, one significant limitation for the VLSI implementation of the CA architecture is the high pin count imposed by the interconnection network. This pin count increases in proportion to the number of perimeter elements (cells) of the chip, i.e., for the 2-D CA architecture proposed in this paper, in proportion to the square root of the number of cells on the chip. Additionally, since the number of cells that can be housed in a single chip is limited by the maximum area constraint of the current VLSI technology, a number of such chips (modules) has to be employed in order to cover large input dimensions.

The proposed architecture is synchronous, i.e., a global clock is used to control the operation of all cells. The ES2 library of standard cells was used for the implementation of the digital logic (library cells are optimized both for area and speed). The choice of a CMOS technology resulted in very low power consumption for the chip as well as high noise-immunity levels.

A set of input/output signal configurations, derived from the simulation examples presented in Section VI of this paper, were used to test the functional behavior of the chip, as follows: input signals were applied to the chip and its output was monitored and compared to the expected output. No discrepancies between the chip's output and expected output were detected.

Additionally, special care was taken in order to force as many of the internal nodes of the chip as possible to change state for the efficient detection of internal faults (the percentage of nodes that changes state after the application of a series of test vectors is called "node toggle count" [37]). The set of input test vectors applied to the chip resulted in a node toggle count of 100% and the output of the chip matched exactly the expected output, corresponding to the simulation examples of Section VI of this paper. Since the design is synchronous, additional care was taken to ensure that chip operation was not affected by various amounts of clock skew [37]. Skewing of the system clock did not produce any deviations from the expected behavior of the chip.

The proposed CA architecture is highly granular, regular, and modular, and, thus, it is particularly suitable for VLSI implementation. Only local (neighborhood) interconnections are required for the interprocessor communication and, thus, interprocessor wiring maps to highly regular structures and the communication overhead is reduced. Additionally, the proposed architecture is simple, since it does not require any arithmetic modules, and storage requirements are minimal, since the only knowledge stored in each cell is that of its nearest neighbors. Since the dimensions of the chip are small, CA architectures of larger dimensions can be implemented on a wafer. The inherent modularity, regularity, and homogeneity properties of the proposed CA architecture make it especially suitable for wafer scale integration [38].

VIII. Conclusion

A new parallel algorithm for collision-free path planning of a diamond-shaped robot among arbitrarily shaped objects and its implementation in VLSI were presented in this paper. The proposed algorithm is based on a retraction of free space onto the Voronoi diagram, which is constructed through the time evolution of cellular automata, after an initial phase during which the boundaries of obstacles are identified and coded with respect to their orientation. The proposed algorithm is both space and time efficient, since it does not require the modeling of objects or distance and intersection calculations. Additionally, the proposed 2-D multistate cellular automaton architecture achieves high frequency of operation and it is particularly suited for VLSI implementation due to its inherent parallelism, structural locality, regularity, and modularity.

References

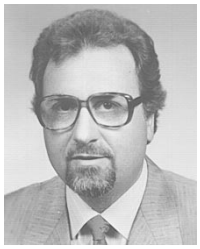
[1] N. S. V. Rao, "An algorithmic framework for navigation in unknown terrains," *IEEE Trans. Comput.*, pp. 37–43, June 1989.
[2] N. S. V. Rao, N. Stoltzfus, and S. S. Iyengar, "A 'retraction' method for learned navigation in unknown terrains for a circular robot," *IEEE Trans. Robot. Automat.*, vol. 7, pp. 699–707, Oct. 1991.
[3] K. Sugihara, "Approximation of generalized Voronoi diagrams by ordinary Voronoi diagrams," *Graphical Models and Image Processing*, vol. 55, no. 6, pp. 522–531, Nov. 1993.
[4] N. S. V. Rao, S. Iyengar, and G. DeSaussure, "The visit problem: Visibility graph-based solution," in *Proc. IEEE Int. Conf. Robot. Automat.*, Philadelphia, PA, Apr. 24–29, 1988.
[5] S. J. Buckley, "Fast motion planning for multiple moving robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, Scottsdale, AZ, May 14–19, 1989, pp. 322–326.

[6] Y. H. Liu, S. Kuroda, T. Naniwa, H. Noborio, and S. Arimoto, "A practical algorithm for planning collision-free coordinated motion of multiple mobile robots," in *Proc. IEEE Int. Conf. Robotics and Automation*, Scottsdale, AZ, May 14–19, 1989, pp. 1427–1432.
[7] N. Kehtarnavaz and S. Li, "A collision-free navigation scheme in the presence of moving obstacles," in *Int. Conf. Computer Vision*, IEEE Computer Society, Los Angeles, CA, 1988, pp. 808–813.
[8] J. H. Reif and M. Sharir, "Motion planning in the presence of moving obstacles," in *Proc. 26th Annu. IEEE Symp. Foundations Comput. Sci.*, Portland, OR, Oct. 21–23, 1985, pp. 144–154.
[9] T. Asano, L. Guibas, J. Hershberger, and H. Imai, "Visibility-Polygon search and Euclidean shortest path," in *26th Symp. Foundations Comput. Sci.*, Portland, OR, Oct. 21–23, 1985, pp. 155–164.
[10] D. E. Koditschek, "Robot planning and control via potential functions," in *Robotics Review*, O. Khatib, J. Graig, and T. Lozano-Perez, Eds. Cambridge, MA: MIT Press, 1989, vol. 1.
[11] Y. K. Hwang and N. Ahuja, "Gross motion planning—A survey," *ACM Computing Surveys*, vol. 24, no. 3, pp. 219–291, Sept. 1992.
[12] H. Noborio, T. Naniwa, and S. Arimoto, "A feasible motion planning algorithm for a mobile robot on a quadtree representation," in *Proc. IEEE Int. Conf. Robot. Automat.*, Scottsdale, AZ, May 14–19, 1989, pp. 327–332.
[13] D. Zhu and J. C. Latombe, "Constraint reformulation in a hierarchical path planner," in *Proc. IEEE Int. Conf. Robot. Automat.*,, Cincinnati, OH, May 13–18, 1990, pp. 1918–1923.
[14] P. Tzionas, P. Tsalides, and A. Thanailakis, "Cellular automata based minimum cost path estimation on binary maps," *IEE Electron. Lett.*, vol. 28, no. 17, pp. 1653–1654, 1992.
[15] ———, "A new algorithm for 3-dimensional minimum cost path planning and its VLSI implementation using a 3-dimensional cellular automata architecture," *Opt. Eng. J.*, vol. 32, no. 11, pp. 2974–2985, 1993.
[16] J. T. Schwartz and M. Sharir, "On the piano movers' problem: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers," *Commun. Pure Appl. Math.*, vol. 36, no. 3, pp. 345–398, May 1983.
[17] C. J. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer, 1991.
[18] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. New York: Springer–Verlag, 1985.
[19] R. Klein and D. Wood, "Voronoi diagrams based on general metrics in the plane," Springer LNCS 294, pp. 281–291, 1988.
[20] I. Aleksander and F. K. Hanna, *Automata Theory: An Engineering Approach*. New York: Crane Russak, 1975.
[21] S. Wolfram, "Statistical mechanics of cellular automata," *Rev. Mod. Phys.*, vol. 55, p.601, 1983.
[22] W. Pries, R. D. McLeod, A. Thanailakis, and H. C. Card, "Group properties of cellular automata and VLSI applications," *IEEE Trans. Comput.*, vol. C-35, pp. 1013–1024, 1986.
[23] N. Packard and S. Wolfram, "Two-dimensional cellular automata," *J. Stat. Phys.*, vol. 38, p. 901, 1985.
[24] K. Kaneko, "Attractors, basin structures and information processing in cellular automata," in *Theory and Applications of Cellular Automata*, Advanced Series on Complex Systems. World Scientific, 1986, vol. 1, pp. 367–398.
[25] T. P. Yunck, "A technique to identify nearest neighbors," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, pp. 678–683, Oct. 1976.
[26] P. Tzionas, P. Tsalides, and A. Thanailakis, "A new, cellular automaton-based, nearest neighbor pattern classifier and its VLSI implementation," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 343–353, Sept. 1994.
[27] T. Ohya, M. Iri, and K. Murota, "A fast Voronoi–diagram algorithm with quaternary tree bucketing," *Inf. Process. Lett.*, vol. 18, no. 4, pp. 227–231, 1984.
[28] T. Asano, M. Sato, and T. Ohtsuki, "Computational geometry algorithms," in *Advances in CAD for VLSI*, T. Ohtsuki, Ed. North Holland, Amsterdam: Elsevier, 1986, vol. 4, ch. 9.
[29] F. Avnaim, J. D. Boissonnat, and B. Faverjon, "A practical motion planning algorithm for polygonal objects amidst polygonal obstacles," in *Proc. IEEE Int. Conf. Robotics and Automation*, Philadelphia, PA, Apr. 24–29, 1988, pp. 1656–1661.
[30] J. Keil and J. R. Sack, "Minimum decomposition of polygonal objects," in *Computational Geometry*. North Holland, Amsterdam: Elsevier, 1985, pp. 197–216.
[31] G. S. Robinson, "Edge detection by compass gradient masks," *Comput. Graphics and Image Process.*, vol. 6, pp. 492–501, 1977.
[32] J. F. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, pp. 679–698, 1986.
[33] B. Paden, A. Mees, and M. Fisher, "Path-planning using a Jacobian based free space generation algorithm," in *Proc. IEEE Int. Conf. Robotics and Automation*, Scottsdale, AZ, May 14–19, 1989, pp. 1732–1737.
[34] C. O' Dunlaing and C. K. Yap, "A retraction method for planning the motion of a disc," *J. Algorithms*, vol. 6, pp. 104–11, 1985.
[35] D. Leven and M. Sharir, "Planning a purely translational motion of a convex robot in 2-dimensional space using Voronoi diagrams," *Discrete Comput. Geom.*, vol. 2, pp. 9–31, 1987.
[36] R. C. Arkin, W. C. Carter, and D. C. Mackenzie, "Active avoidance: Escape and dodging behaviors for reactive control," *Int. J. Pattern Recognition and Artificial Intell.*, vol. 17, no. 1, pp. 175–192, 1993.
[37] European Silicon Structures, "Solo 1400 Reference Manual: Shipment (Preparing for Fabrication)," European Silicon Structures Ltd., U.K., 1990.
[38] C. Jesshope and W. Moore, *Wafer Scale Integration*. Bristol, U.K.: Adam Hilger, 1986.

Panagiotis G. Tzionas was born in Kozani, Greece, on September 8, 1964. He received the B.Eng. degree in electrical and electronic engineering from Imperial College, University of London, U.K., in 1988, the M.Sc. degree in digital electronics from King's College, University of London, U.K., in 1990, and the Ph.D degree in electrical and computer engineering from Democritus University, Thrace, Greece, in 1994.

He is currently with the Laboratory of Electrical and Electronic Materials Technology, Democritus Univeristy of Thrace, Greece. His research interests are mainly in the area of VLSI implementationof ASICS for computer vision and pictorial data manipulation.

Dr. Tzionas is a member of the Technical Chamber of Greece and the Association of City and Guilds Institute of London, U.K.



Adonios Thanailakis was born in Greece on August 5, 1940. He received B.Sc. degrees in physics and electrical engineering from the University of Thessaloniki, Greece, in 1964 and 1968, respectively, and the M.Sc. and Ph.D. degrees in electrical engineering and electronics from UMIST, Manchester, England, U.K., in 1968 and 1971, respectively.

He has been a Professor of Electrical and Electronic Materials Technology in the Department of Electrical and Computer Engineering, School of Engineering, Democritus University of Thrace, Xanthi, Greece, since 1977, and he was a visiting Professor in the Department of Electrical Engineering, University of Manitoba, Canada, during the Academic year 1984–1985. He has been active in solid-state electronic devices research since 1968. His current research activity includes amorphous materials and devices, photovoltaic conversion of solar energy, applications of group theory in physical and computational systems, and VLSI systems design. He served as Rector (President) of the Democritus University of Thrace. He has published a great number of scientific and technical papers, as well as four textbooks on materials, devices, electronic noise, and photovoltaic conversion of solar energy.

Prof. Thanailakis is a Member of the Institute of Electrical Engineering, the Institute of Physics, the European Physical Society, the Technical Chamber of Greece, and the Institute of Heliotechnics of Greece.

Philippos G. Tsalides was born in Mirina Limnou, Greece, on October 14, 1953. He received the Dipl. degree from the University of Padova, Italy, in 1979 and the Ph.D. degree from Democritus University of Thrace, Greece, in 1985.

He is a Professor of Applied Electronics in the Department of Electrical Engineering, Democritus University of Thrace. His current research interests include VLSI architectures, VLSI systems, BIST techniques, applications of cellular automata in image processing, as well as in computational systems. He has published a number of papers and a textbook on VLSI systems *(Basic Principles of Design and Fabrication)*).

Dr. Tsalides is a Fellow of the Institute of Electrical Engineering and a member of the Technical Chamber of Greece.