Shortest Paths in Triangular Grids with Neighbourhood Sequences

Benedek Nagy*

Institute of Mathematics and Informatics, University of Debrecen, Hungary

In this paper we analyse some properties of the triangular and hexagonal grids in the 2D digital space. We define distances based on neighbourhood relations that can be introduced in these grids. We present an algorithm, which calculates the distance from an arbitrary point to another one for a given neighbourhood sequence in the triangular grid. Moreover, this algorithm produces the shortest path between these points.

Keywords: discrete geometry, digital geometry, digital plane, triangular grid, hexagonal grid, neighbourhood relation, neighbourhood sequence, shortest path, algorithm.

1. Introduction

Digital geometry is an important part of theoretical image processing. In digital geometry the spaces we work in consist of discrete points with integer coordinates. Consequently, we define distance functions which take integer values.

In the first period, mainly the square grid was investigated, since this is the most usual space in image processing. One of the first results was the introduction of neighbourhood relations among the points defined by Rosenfeld and Pfaltz [17]. They gave two types of motions in the 2D square grid. The cityblock motion allows horizontal and vertical movements only, while at the chessboard motion the diagonal directions are also permitted. So, based on these motions, in this grid we have two kinds of distances. Figure 1 shows a point together with those points, which have distance 1 from it. Both cityblock and chessboard distances are shown. In [11,16] there is a short summary of examination of the square grid. In the case of square grid, each coordinate value of a point is independent of the others. In n dimension we use n coordinates. In the n dimensional cubic grid the structure of the nodes is isomorphic to the structure of the n dimensional cubes. The grid of nodes, and the grid obtained by representing each cube with its central point, are identical within a shift. So, the dual (a well-known concept of graph theory) of the square grid is a square grid as well. In Figure 1 both options are shown.

In [18], Yamasita and Ibaraki and in [2], Das, Chakrabarti and Chatterji introduced the concept of (periodic) neighbourhood sequences, which gives us the possibility to mix the city-

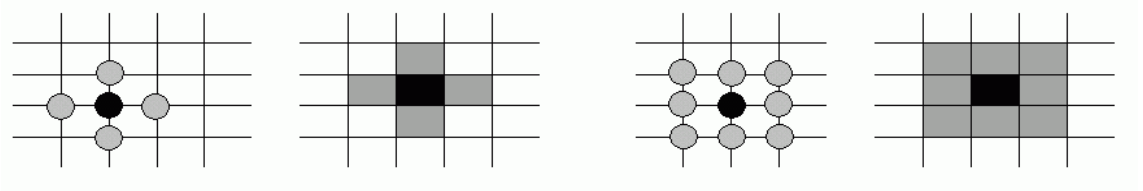

Fig. 1. 'Cityblock' and 'chessboard' neighbourhood relations in the square grid of nodes and of regions.

* Research supported in part by grant F043090 of the Hungarian National Foundation for Scientific Research.

block and chessboard motions. In [7], the authors extend this theory to infinite sequences, which need not be periodic. In this paper we use similar generalised neighbourhood sequences.

There is wide literature about the neighbourhood sequences in the square grids ([1-5,7,15]). We can extend the definition of types of neighbours to other grids too. We can assume that two geometrical objects (of the same dimension as the space where we are) are neighbours of each other if there is at least one point which is on the border of both. Two geometrical objects are neighbours of type *m* (or shortly, *m*-neighbours) if we can make *m* steps from one to the other in such a way that in each step we step through a border line of the two objects.

In the following sections we restrict our analysis only to 2D spaces, especially to the hexagonal and triangular grids. The hexagonal grid is also widely investigated. One of the first papers about the hexagonal plane is [10], in which the name 'rhombic array' was introduced. There is a natural neighbourhood relation in this grid and it is used almost every time. In [9] the distance based on this neighbourhood relation was calculated with two independent coordinate values. Because of the symmetric properties of the grid, in [8] the description uses three coordinates which have zero sum. In many applications the hexagonal grid seems to be more useful than the square grid (for example see [6]). The dual of the hexagonal grid is not hexagonal, but triangular. This is the reason why we examine these grids parallelly. The grid of triangular nodes is isomorphic to the grid of hexagonal areas. We investigate these – so-called hexagonal – grids in Section 2.

The triangular grid is the third "basic grid". It also has triangular symmetry, therefore three coordinates are recommended to examine this system, as we show here. There are three kinds of neighbours of each point in this grid (see [6]). We present a suitable method to formulate the concept of neighbourhood sequences in the triangular grid. We give an algorithm which finds the shortest path between arbitrary two points. The grid of the triangular areas (the so-called triangular grid) is isomorphic to the grid of hexagonal nodes. We analyse their properties in Section 3.

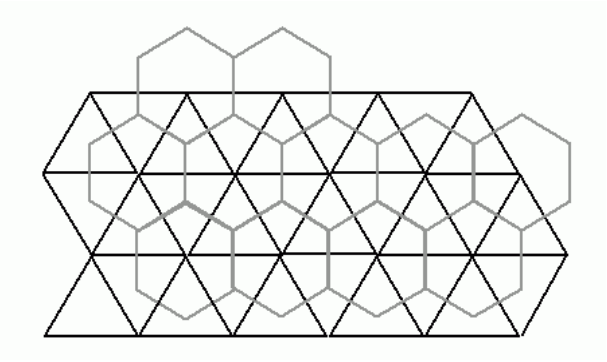

Fig. 2. The connection between the triangular and the hexagonal grids.

Summarizing, in this paper we study the triangular and the hexagonal grids which are the duals of each other (see Figure 2).

2. The Hexagonal Grid

We need to overview some results about the hexagonal grid, mainly the techniques we will use for the triangular grid. In this section we recall some previous results, for instance from [9]. However, we use a different approach from [8], which also provides a suitable way to represent the triangular grid.

2.1. Basic Definitions

It is well known that the grids of hexagonal areas and triangular nodes are duals of each other. We prefer to use the term hexagonal grid instead of triangular grid of nodes. In digital image processing the neighbourhood criterion, illustrated in Figure 3, is used almost uniquely, since this is the most natural for a human observer. This neighbourhood criterion was investigated, for example, in [8,9] and in [6].

By the formal definition, two objects are neighbours

— in the grid of triangular nodes: if there is a direct connection between these nodes,

— in the grid of hexagonal areas: if these hexagons have a common side.

In Figure 3 we can easily prove that every object has six neighbours.

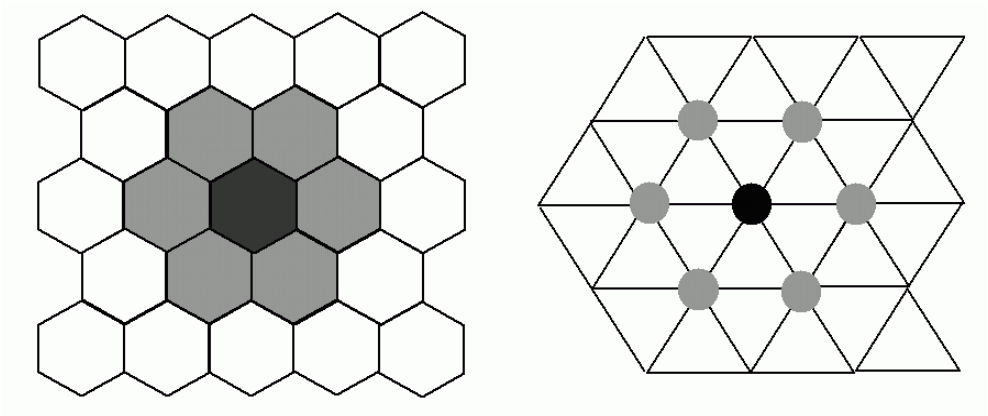

Fig. 3. Neighbours in the hexagonal grid of areas and in the triangular grid of nodes.

We introduce the coordinate system in the same way, as it was given in [8] for nodes of the triangular grid. We consider the hexagonal areas as points and assign three coordinate values to them, to reflect the geometrical symmetry of the grid. The necessity of this procedure is to be able to mathematically handle this hexagonal structure, for example to calculate numerically the distance between two objects. Figure 4 shows the result of this procedure.

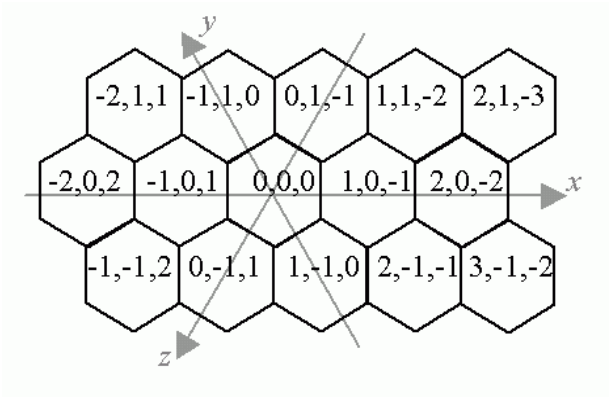

Fig. 4. Coordinate values on the hexagonal grid.

The following procedure helps us to make this assigning.

Procedure 2.1.1. *Choose a point for the origin, whose coordinate values are* $(0, 0, 0)$. *Then fix three coordinate axes as lines crossing the centre of the origin which are orthogonal to its two sides. The direction of the axes x, y, and z are taken as* $0°$, $120°$ *and* $240°$ *, respectively. We assign the coordinate values to the points inductively. If the coordinates of a point are* (a_1, a_2, a_3), *then let the coordinates of its neighbour in the direction x be* (a_1+1, a_2, a_3-1), *and in the opposite direction* $(a_1 - 1, a_2, a_3 + 1)$. *Similarly in the direction of y* $(a_1 - 1, a_2 + 1, a_3)$, *and in the opposite direction of y, it is* $(a_1 + 1, a_2 - 1, a_3)$. *According to the way the coordinates are introduced, moving in the direction z, the coordinate values are* $(a_1, a_2 - 1, a_3 + 1)$, *and to the opposite direction* $(a_1, a_2 + 1, a_3 - 1)$.

The sum of the coordinate values of every point is 0, so there are only two independent values, any of them is removable. In [9] the authors use only two independent coordinates. The coordinate value *z* is used only to preserve symmetry. Two points are neighbours if any of their coordinate value is the same, and the differences of the other two corresponding coordinate values are ± 1.

Definition 2.1.2. *By a step we mean moving from one point to a neighbouring point. A path is a sequence of points, such that each point (except the first one) is a neighbour of the previous one. The first element is the starting point, and the last element is the end point. The length of the path is the number of steps in the path (it is less by* 1 *than the number of points in the path). The distance between two points is the length of a shortest path between them.*

Since we consider only one type of neighbourhood relation, the concept of neighbourhood sequences coincides with that of the paths. We introduce only one hexagonal distance, based on the neighbourhood criterion at each step.

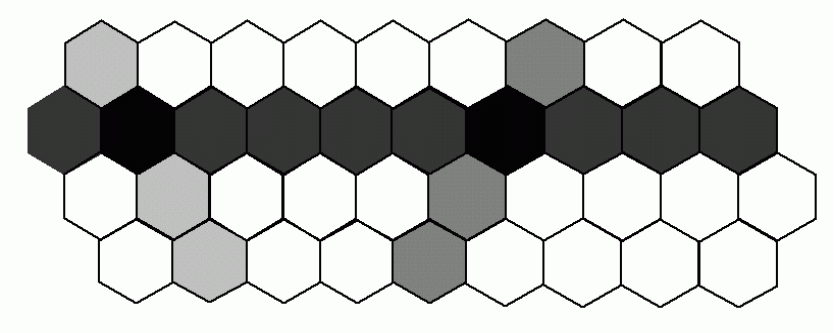

Fig. 5. Examples for lanes on the hexagonal grid.

Definition 2.1.3. *The sequence of objects, for which a coordinate value remains constant, forms a lane.*

A line including hexagons of a lane only is parallel to one of the coordinate axes.

2.2. Finding the Shortest Path

In [9] a formula was given to calculate the distance between two points of the hexagonal grid represented by two independent coordinates. Now we give an algorithm, which produces the shortest path between two arbitrary points of the hexagonal grid. Our procedure is natural, and it uses our definition of lanes.

Procedure 2.2.1. *Let us consider two points and take the absolute values of the differences between their corresponding coordinates. If they have equal corresponding coordinates, then we go along the lane, for which this coordinate value remains constant. If all corresponding coordinate values are different, then there are two, whose absolute differences are smaller than the third. (If the third, the greatest, is equal to another, then since the differences have zero sum, there must be coinciding corresponding coordinate values.) So in this case, we go along those two lanes, where the coordinates, which have smaller absolute differences, remain constants. First, we go along the lane where one of the two smaller values is constant, until the other coordinate reaches its destination value. Then we take the lane where the already fixed coordinate remains constant until we reach the destination point.*

The procedure 2.2.1 shows an algorithm which can solve the minimal path problem in case of hexagonal grid. Using this algorithm we can easily calculate the length of the shortest path. The path-length is the largest of the absolute values of the difference of the corresponding coordinates. Hence, the length of this shortest path is the known distance [9]. (Our statement about the calculation of distance is equivalent to the main proposition in [9].)

The distance introduced above is close to the Euclidean distance. If the Euclidean length of a side of the hexagon is 1, and the hexagonal distance between two points is k, the Euclidean distance to the centre of these hexagons is between $1.5k$ and $\sqrt{3}k$.

3. The Triangular Grid

3.1. Introduction

The grid of triangular areas is equivalent to the grid of hexagonal nodes. In this section, we use triangular areas. It is the so-called triangular grid. We define three types of neighbours on the triangular grid (as in [6]), shown in Figure 6.

Each triangle (not considering the original one) has three 1-neighbours, nine 2-neighbours (the 1-neighbours, and six more 2-neighbours), and twelve 3-neighbours (nine 2-neighbours, and three more 3-neighbours). In Figure 6, for hexagonal grid of nodes we use the dark grey points to represent the 1-neighbours. With these points the light grey ones are the 2-neigbours, and with them the white points are the 3-neighbours. (Only the 1-neighbours are directly connected by a side, the 2- and 3-neighbours are at the positions of diagonals, respectively.)

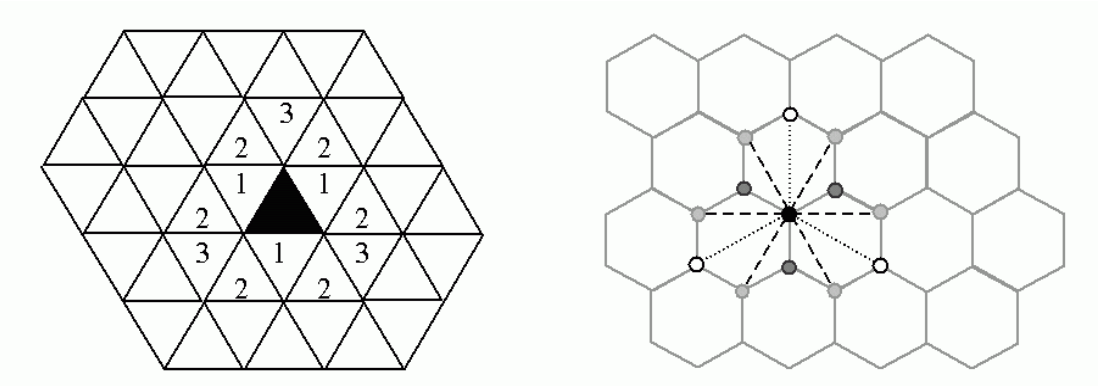

Fig. 6. Types of neighbours in the triangular grid of areas and in the hexagonal grid of nodes.

For similar reasons as in the case of the hexagonal grid, we use three coordinate values to represent the triangles, see Figure 7. The triangles are called points. This coordinatization (procedure 3.2.1) plays a very important role, because it is the main tool to get concrete results.

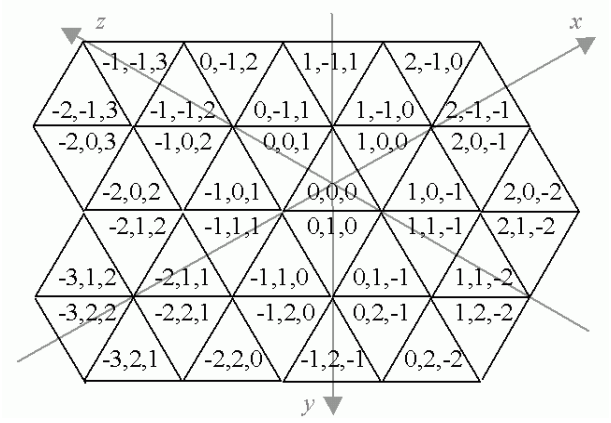

Fig. 7. Coordinate values on the triangular grid.

3.2. Basic Definitions

In order to reach the aims formulated in the introduction, we give the basic definitions and notation. First, we show how the coordinate values are assigned to the points of the grid.

Procedure 3.2.1. *Choose a point for the origin, whose coordinate values are* $(0, 0, 0)$. *Take the three lines through the centre of the origin triangle, which are orthogonal to its sides. Fix these lines as the coordinate axes x, y and z, as shown in Figure 7. We assign the coordinate values to the points inductively. Let the coordinate values of a triangle A be known. Consider a triangle B, which has not coordinate values yet and has a common side with A. This common side is orthogonal to one of the coordinate axes. According to the direction of this axis, we increase or decrease the corresponding coordinate value of A by* 1 *to get the corresponding coordinate of B. The other two values of A and B are equal.*

Figure 7 shows the result of the previous procedure, which uniquely determines the coordinate values of the triangles, as we will show later.

Using our coordinate values we can describe the neighbourhood relation. Let p and q be two points of the triangular grid. The i-th coordinate of the point p is indicated by $p(i)$ $(i = 1, 2, 3)$, and similarly for q. Then the points p and q are m-neighbours $(m = 1, 2, 3)$, if the following conditions hold:

$$|p(i) - q(i)| \leq 1, \text{ for } 1 \leq i \leq 3,$$

$$|p(1)-q(1)|+|p(2)-q(2)|+|p(3)-q(3)| \leq m.$$

Definition 3.2.2. *The points, which have the same value for a corresponding coordinate, form a lane.*

Observe that each lane is orthogonal to one of the coordinate axes. For the points of a lane a coordinate value is fixed. The other two values change by ± 1, respectively.

The points and their coordinate values are assigned by a one-to-one mapping. Using the concept of lanes we can see it in the following way. Let us fix two coordinate values. They define two non-parallel lanes, whose intersection contains two points. The third coordinate

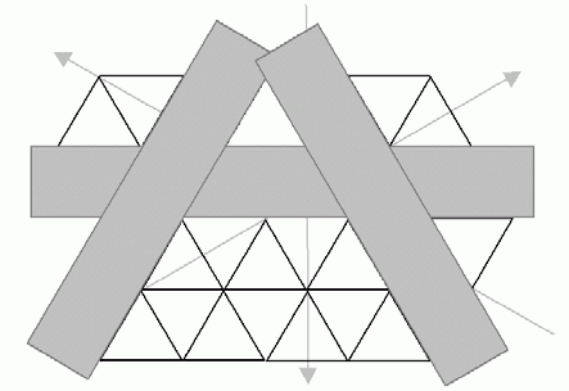

Fig. 8. Examples for lanes in the triangular grid.

values of these points should fulfill the requirement that the sum of the coordinates is 0 and 1, respectively.

Therefore we have two types of points according to the values of the sum of its coordinates. If the sum is 0, then we call the point even, if the sum is 1, then the point has odd parity. In our figures the even and odd triangles are of the shape $\triangle$ and ∇, respectively.

We can see that if two points are 1-neighbours, then their parities are different. If two points are 1-neighbours, then there are two lanes containing both of them. If two points are 2-neighbours, but not 1-neighbours, then only one lane contains both of them. If two points are 3-neighbours, but not 2-neighbours, then no lane contains both of them.

Definition 3.2.3. *The infinite sequence $B = (b(i))_{i=1}^{\infty}$, with $b(i) \in \{1, 2, 3\}$ for all $i \in N$, is called a neighbourhood sequence. If for some $l \in N$, $b(i) = b(i+l)$ holds for every $i \in N$, then B is called periodic, with period l. In this case we use the short form $B = (b(\dot{1}), \ldots, b(\dot{l}))$.*

For example, let $B_1 = (\dot{1}, \dot{1}, \dot{2})$ and $B_2 = (1, 3, 1, 2, \dot{2})$. Then B_1 is periodic with period 3, while B_2 is non-periodic. (We use the notation $\dot{x}$ to represent the repetition of items.) Now we show how one can use such sequences to define distances in the triangular grid.

Definition 3.2.4. *Let p and q be two points and B a neighbourhood sequence. The point sequence $\Pi(p, q; B)$ of the form $p = p_0, p_1, \ldots, p_h = q$, where p_{i-1} and p_i are $b(i)$-neighbours for $1 \leq i \leq h$, is called a B-path from p to q. The length of this path is h. The B-distance from p to q is defined as the length of the shortest path, and is denoted by $d(p, q; B)$.*

In Figure 9 there are some paths given between the points $p = (-3, 2, 1)$ and $q = (2, -1, 0)$ by the help of the previous B_1 and B_2. As we can see, there are paths with different lengths between the points. On the left-hand side of Figure 9 we show two paths with the neighbourhood sequence B_1, one of them has length 10 and the other has length 7. The points of the paths are represented as dashed and dotted figures. The numbers in the triangles refer to the steps of the given path. Similarly, in the right-hand side of the figure, we show other paths, using B_2, of lengths 5 (it is the shortest path) and 10 between the same points.

Remark 3.2.5. The distance of two points is independent of the chosen neighbourhood sequence if and only if they are at most 1-neighbours.

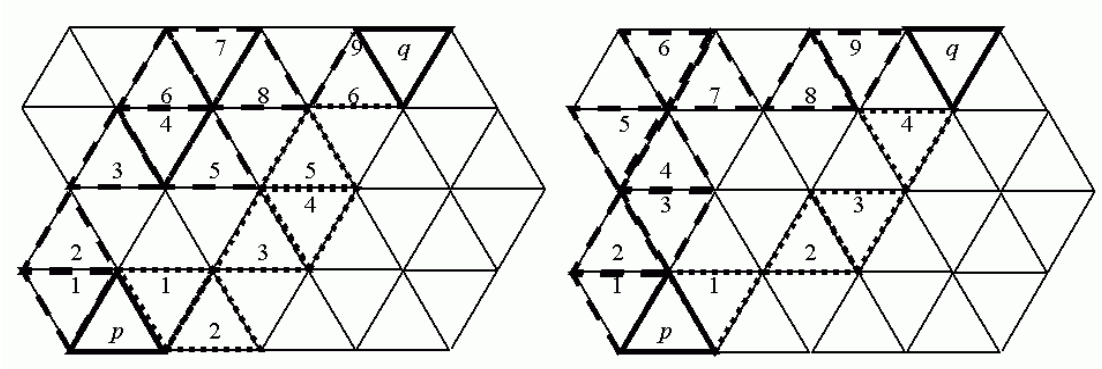

Fig. 9. Paths of different lengths from p to q using B_1 and B_2.

If the points are the same, then the distance is 0, and we do not need any step. If there is only one difference in the coordinate values, and it is 1, then we can step from one point to another with any kind of neighbouring criterion, therefore their distance is 1. It is easy to show that in any other case the neighbourhood sequences $B_1 = (\dot{1})$ and $B_2 = (\dot{2})$ define different distances.

Definition 3.2.6. *Let p, q be two points. The difference $w = (w(1), w(2), w(3))$ is defined for the points p and q, in the following way: $w(i) = p(i) - q(i)$. If $w(1) + w(2) + w(3) = 0$, then the parity of w is even, otherwise it is odd.*

We can define the distance between points and lanes, which depends only on these objects and is independent of the neighbourhood sequences. The distance of a point and a lane is the minimal number of lanes in which we have to go through to reach the point from the lane. We can calculate this distance with iteration.

Procedure 3.2.7. *The distance between the lane A and the point p is zero, if p is on the lane A. The next parallel lanes of the lane, which contains the point p, have distance 1 from p. Let the distance of the lane B and p be d. Then the next parallel lane with B, – which has not distance $d-1$ from the point p – has distance $d+1$ from p.*

If the distance of the lane A and the point p is d, then starting from p we can reach some points of A in the d-th step by using the constant neighbourhood sequence $(\dot{2})$.

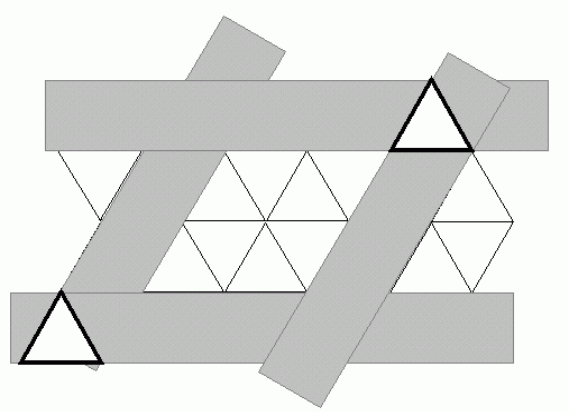

Fig. 10. An Example for a parallelogram between two points.

As in Procedure 2.2.1 in the hexagonal case, using one or two lanes we can get from one triangle to any other one. In the latter case, the angle of these lanes can be chosen to be 120°. (In one of these lanes, the coordinate value which has the smallest absolute value in the difference w of the points remains constant and in the other the coordinate value which has the second largest absolute value in w is fixed.) Generally, we obtain a parallelogram, see Figure 10.

3.3. The Shortest Paths

In the triangular grid of areas, we have some difficulties in changing the coordinate values. Such difficulties do not occur in cases of the hexagonal and square grid. In the triangular grid, when moving from a point to one of its neighbours, we have to take care about the parity of these points. Namely, we have to change the coordinates of a point in such a way that the sum of the coordinate values is 0 or 1.

Now we give an algorithm, which solves the problem of constructing the shortest path between two given points. We prove that the algorithm is correct, i.e. that it finds the shortest path from the first point to the other one.

Algorithm 3.3.1.

Input: two points p, q; a neighbourhood sequence B.

Output: one of the shortest paths Π from p to q, using B.

STEP 1 Let w be the difference of q and p $(w(i) = q(i) - p(i),\ i = 1, 2, 3)$. Let $x_0 = p$, $\Pi = x_0$ and $j = 0$.

STEP 2 If $w(i) = 0$ $(i = 1, 2, 3)$, then go to step 11.

STEP 3 Let $j = j + 1$. Let h_i $(i = 1, 2, 3)$ be a permutation of $(1, 2, 3)$, such that $|w(h_1)| \geq |w(h_2)| \geq |w(h_3)|$, and $sgn(w(h_1)) \neq sgn(w(h_2))$.

STEP 4 If $b(j) = 1$, then if x_{j-1} is even/odd, change by 1 the positive/negative one from $w(h_1)$ and $w(h_2)$, respectively:
$w(h_i) = sgn(w(h_i))|w(h_i) - 1|$, where $i = 1$ or 2; go to step 8.

STEP 5 If $b(j)=2$, then let $w(h_1)=sgn(w(h_1))|w(h_1)-1|$ and $w(h_2)=sgn(w(h_2))|w(h_2)-1|$; go to step 8.

STEP 6 If the parity of x_{j-1} is even, and w has two coordinates with positive values, then let $w(i)=sgn(w(i))|w(i)-1|$ $(i=1,2,3)$, else let $w(h_1)=sgn(w(h_1))|w(h_1)-1|$ and $w(h_2)=sgn(w(h_2))|w(h_2)-1|$.

STEP 7 If the parity of x_{j-1} is odd, and w has two negative coordinate values, then let $w(i)=sgn(w(i))|w(i)-1|$ $(i=1,2,3)$, else let $w(h_1)=sgn(w(h_1))|w(h_1)-1|$ and $w(h_2)=sgn(w(h_2))|w(h_2)-1|$.

STEP 8 Let $x_j(i)=q(i)-w(i)$ $(i=1,2,3)$.

STEP 9 Concatenate x_j to the path Π.

STEP 10 Go to step 2.

STEP 11 The output is Π. The length of the path is j. End.

Now we give a detailed description of the algorithm.

In Step 1, we initialise the algorithm: p is the starting point, $p=x_0$ is the first element of the path Π which contains x_0 only, w is the difference between the last point of Π and q. The length of the path j starts from 0.

In Step 2, we check whether we have finished or not. If yes, we go to Step 11, where the output values are given, and the algorithm terminates.

In Step 3, we increase the length of the path j, and order the elements of the difference w. First, suppose that among these elements, there is one, with the largest absolute value. In this case, this element has opposite sign from the others, or some of the others are equal to zero. If two elements have the same absolute value, and the third one has smaller absolute value, then the two elements with larger absolute values have opposite signs. Then the third element must be 0 or ± 1, because of the restriction of the sum of the coordinates of w. Hence the permutation satisfies our conditions. If all the three elements have the same absolute value, then this value must be 1, and their sum is ± 1. Hence the permutation can be made in every case.

As we mentioned at the end of subsection 3.2. and showed in figure 10, we can connect points x_j and q by two lanes and we have a parallelogram. If we can move to a 1- or a 2-neighbour of x_j then we make this step on the lane which goes through x_j and is closer to q.

We use Step 4 if we move to a 1-neighbour. In this case we can move from x_{j-1} to a point of different parity. We decrease by 1 one of the absolute values of the first two elements of the permutation of w. If x_{j-1} is even, then the sum of the elements of w is 1, if q is odd, and the sum of the elements of w is 0 if q is even. Since w has a non-zero element, $w(h_1)$ or $w(h_2)$ must be positive, as well. So we can change this positive value. If x_{j-1} is odd, then the sum of the elements of w is -1 if q is even, and the sum of the elements of w is 0 if q is odd. Since w has a non-zero element, $w(h_1)$ or $w(h_2)$ must be negative as well. So we can change this negative value. Thus, at this step we get closer to q in a coordinate by 1.

Let us consider step 5. If we can move to a 2-neighbour of x_{j-1}, we have two options. First, if there is only one non-zero element of w, then it must be ± 1, so we change this element to 0. In the other case, the first two elements of the permutation have opposite signs, hence we move from x_{j-1} to a point of the same parity, by changing these two values.

In Steps 4 and 5 we step in the lane less distant from the destination point. We can decrease the higher absolute values in w, but if we can move to a 3-neighbour, then we step in the lane in which we can decrease the higher absolute values in w, and if possible we step to the next parallel lane which is closer to the end-point. So if we move to a 3-neighbour of x_j, then if possible, we step to the point which is in the intersection of two lanes which are closer to q than the previous ones. (This point is on the next lanes as x_j.) If it is impossible, we move to a 2-neighbour of x_j. We describe these cases below.

We use Step 6 or Step 7 if $b(j)=3$, so we move to a 3-neighbour of x_{j-1}. If the parity of x_{j-1} is even (Step 6), then we may step to an odd point and change all its coordinates by 1, if w contains two positive and one negative values. If x_{j-1} has odd parity and w contains two negative and one positive values, then we step to an even point by changing every coordinate value by 1. If w does not allow us to do so, we can step only to a 2-neighbour, like in Step 5.

w	j	$w(h_1)$	$w(h_2)$	$w(h_3)$	$b(j)$	x_j
$(-3,4,-2)$	0	–	–	–	–	$(3,-4,2)$
$(-3,4,-2)$	1	y:4	$x:-3$	$z:-2$	2	$(2,-3,2)$
$(-2,3,-2)$	2	y:3	$x:-2$	$z:-2$	3	$(1,-2,1)$
$(-1,2,-1)$	3	y:2	$x:-1$	$z:-1$	1	$(1,-1,1)$
$(-1,1,-1)$	4	$x:-1$	y:1	$z:-1$	3	$(0,0,0)$

Table 1. Construction of a shortest path to example 3.3.2.

In Steps 8 and 9 the algorithm calculates the coordinates of x_j, by using the values of w, and adds x_j to the path Π (w is the difference of q and x_j).

Step 10 guarantees the repetition of the procedure, starting from Step 2.

The next example shows, how the algorithm works in practice.

Example 3.3.2. *Let $p = (3,-4,2)$ and $q = (0,0,0)$ be two points, and $B = (\dot{2},\dot{3},\dot{1},\dot{3})$ a neighbourhood sequence. Table 1 shows how the values change during the algorithm. The notations used in Table 1, are introduced in Algorithm 3.3.1. The first row of the table contains the initial values of the algorithm. Every row contains values obtained after moving to the next neighbour. The shortest path is presented in Figure 11.*

Theorem 3.3.3. *Algorithm* 3.3.1 *provides the shortest path.*

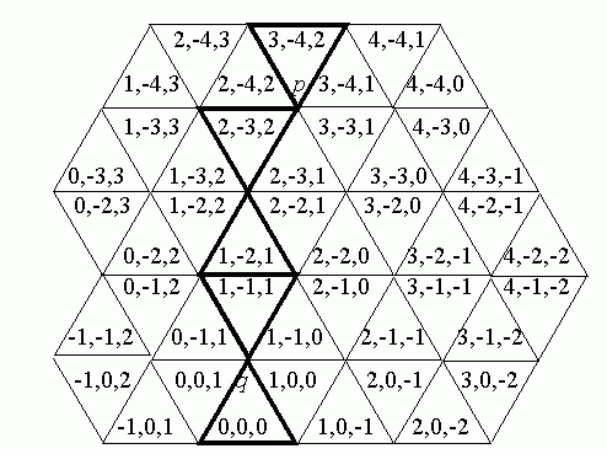

Fig. 11. The shortest path in Example 3.3.2.

Proof. Let $p = y_0, y_1, \ldots y_m = q$ be a B-path, and for $i = 0, \ldots, m$ put $v_i = (q(1) - y_i(1), q(2) - y_i(2), q(3) - y_i(3))$ and $h_i = |y_i(1) - q(1)| + |y_i(2) - q(2)| + |y_i(3) - q(3)|$. Similarly, for the path provided by the algorithm ($p = x_0, x_1, \ldots x_n = q$), let $w_i = (q(1) - x_i(1), q(2) - x_i(2), q(3) - x_i(3))$ and $g_i = |x_i(1) - q(1)| + |x_i(2) - q(2)| + |x_i(3) - q(3)|$ $(i = 1, \ldots, n)$.

We show that $g_i \le h_i$ for all $i \le \min(m, n)$. We use induction. For $i = 0$ we have $g_0 = h_0$, so our assumption holds. Suppose that $g_i \le h_i$. We prove that $g_{i+1} \le h_{i+1}$. We distinguish three cases according to the values of $b(i)$.

If $b(i) = 1$, then $g_{i+1} = g_i - 1$, and $h_{i+1} \ge h_i - 1$. Hence, by $g_i \le h_i$, we have $g_{i+1} \le h_{i+1}$.

If $b(i) = 2$, then $h_{i+1} \ge h_i - 2$, and if $g_i \ge 2$ then $g_{i+1} = g_i - 2$. Hence, using $g_i \le h_i$, we get $g_{i+1} \le h_{i+1}$. If $g_i = 1$, then $g_{i+1} = 0$, and $g_{i+1} \le h_{i+1}$ holds.

Finally, let $b(i) = 3$. Then $g_{i+1} = g_i - 3$ yields $g_{i+1} \le h_{i+1}$, as $h_{i+1} \ge h_i - 3$. If $g_{i+1} = 0$, then we also have $g_{i+1} \le h_{i+1}$. Otherwise, $g_{i+1} = g_i - 2$, and we have two possibilities: the parity of x_j is even, and w_j contains two negative values, or the parity of x_j is odd, and w_j contains two positive values. If $h_i > g_i$, then $g_{i+1} \le h_{i+1}$, since $h_{i+1} \ge h_i - 3$. If $h_i = g_i$, then the parity of w_i is the same, as the parity of v_i. We have the following cases. If v_i has the same number of positive and negative elements as w_i, then y_{i+1} can differ from y_i in at most two coordinates, and we still have the inequality $g_{i+1} \le h_{i+1}$. Otherwise y_{i+1} differs from y_i in all coordinates, and we have $g_{i+1} \le h_{i+1}$ again, since the difference of some coordinate of y_{i+1} and q must grow. If $h_i = g_i$ then v_i

cannot contain the same number of positive elements as the number of negative elements of w_i, and vice versa. It is because the number of the positive and negative elements in w_i is the same as in the difference between q and p (until one or more of them become zero). If v_i contains an element c, which has opposite sign in w_i, then h_i must be greater than the sum, where c is replaced by 0. Since we assumed that $g_i = h_i$, it is a contradiction.

We have $g_i \leq h_i$ for all $i \leq \min(m, n)$, and the sequence g_i strictly monotonously tends to zero. It implies that $n \leq m$. So the algorithm stops after finite number of steps, and it provides the shortest path from p to q.

The distance between any two points p and q, with respect to a neighbourhood sequence B, depends on the difference and the parity of the points, and on the neighbourhood sequence only.

Algorithm 3.3.1 is a greedy algorithm, since at every step it changes as many coordinate values as possible to get closer to the end point.

Let us examine the complexity of Algorithm 3.3.1. It is clear that we need only memory to know what is the point where we are (x_i), and what is the following element of the neighbourhood sequence. So if we can write the output path and we can read the sequence while the algorithm is running, then we need only constant memory.

What is the time-complexity of our algorithm?

It is easy to show that there is a constant upper bound for the time that an iteration takes. (In Steps 3-9 there is ordering of three elements, evaluation of conditions and changing values.) So an iteration takes maximum c time. In the worst case (with neighbourhood sequence $(\dot{1})$) we must make $|w(1)| + |w(2)| + |w(3)|$ steps, where w is the difference between the start and end points.

So our algorithm terminates in linear time. Therefore we can say that it is efficient.

3.4. Interesting Examples

In this section, we study distance functions with strange properties. We illustrate that our algorithm works also in the case when the distance based on a neighbourhood sequence is not symmetric and/or does not meet the triangular inequality. In the following examples, we use the same notation as in Example 3.3.2.

Example 3.4.1. *Let $r = (1, -2, 1)$ and $s = (0, 0, 0)$ be two points, and $B = (\dot{1}, \dot{3}, \dot{2})$ a neighbourhood sequence. First, we calculate $d(r, s; B)$ by using Algorithm 3.3.1. By Table 2 we get $d(r, s; B) = 2$. Now let us calculate $d(s, r; B)$. The result is in table 3: $d(s, r; B) = 3$. As $d(r, s; B) \neq d(s, r; B)$, this distance function is not symmetric.*

w	j	$w(h_1)$	$w(h_2)$	$w(h_3)$	$b(j)$	x_j
$(-1,2,-1)$	0	–	–	–	–	$(1,-2,1)$
$(-1,2,-1)$	1	y:2	x:-1	z:-1	1	$(1,-1,1)$
$(0,1,-1)$	2	x:-1	y:1	z:-1	3	$(0,0,0)$

Table 2. The shortest path from r to s in Example 3.4.1.

w	j	$w(h_1)$	$w(h_2)$	$w(h_3)$	$b(j)$	x_j
$(1,-2,1)$	0	–	–	–	–	$(0,0,0)$
$(1,-2,1)$	1	y:-2	x:1	z:1	1	$(1,0,0)$
$(0,-1,0)$	2	y:-2	z:1	x:0	3	$(1,-1,1)$
$(0,0,-1)$	3	y:-1	x:0	z:0	2	(1,-2,1)

Table 3. The shortest path from s to r in Example 3.4.1.

w	j	$w(h_1)$	$w(h_2)$	$w(h_3)$	$b(j)$	x_j
$(0,1,-1)$	0	–	–	–	–	$(0,0,0)$
$(0,1,-1)$	1	y:1	z:-1	x:0	2	$(0,1,-1)$

Table 4. Calculating the shortest path from $(0, 0, 0)$ to $(0, 1, -1)$ by $B = (\dot{2}, \dot{1}, \dot{1})$.

w	j	$w(h_1)$	$w(h_2)$	$w(h_3)$	$b(j)$	x_j
$(0,1,-1)$	0	–	–	–	–	$(0,1,-1)$
$(0,1,-1)$	1	y:1	z:-1	x:0	2	$(0,2,-2)$

Table 5. Calculating the shortest path from $(0, 1, -1)$ to $(0, 2, -2)$ by $B = (\dot{2}, \dot{1}, \dot{1})$.

w	j	$w(h_1)$	$w(h_2)$	$w(h_3)$	$b(j)$	x_j
$(0,2,-2)$	0	–	–	–	–	$(0,0,0)$
$(0,2,-2)$	1	y:2	z:-2	x:0	2	$(0,1,-1)$
$(0,1,-1)$	2	y:1	z:-1	x:0	1	$(0,2,-1)$
$(0,0,-1)$	3	z:-1	x:0	y:0	1	$(0,2,-2)$

Table 6. Calculating the shortest path from $(0, 0, 0)$ to $(0, 2, -2)$ by $B = (\dot{2}, \dot{1}, \dot{1})$.

Example 3.4.2. *Let $r = (0, 0, 0), s = (0, 1, -1)$ and $t = (0, 2, -2)$ be three points, and $B = (\dot{2}, \dot{1}, \dot{1})$ a neighbourhood sequence. The calculation of $d(r, s; B)$ is in Table 4. So $d(r, s; B) = 1$. In Table 5 we present the determination of $d(s, t; B) = 1$. The calculation of $d(r, t; B)$ is presented in Table 6. As we can see, $d(r, t; B) = 3$, and $d(r, t; B) > d(r, s; B) + d(s, t; B)$. Thus the triangular inequality does not hold.*

As we can see in the previous examples, we can find neighbourhood sequences which do not generate metric. In [14] we examine metric generated by neighbourhood sequences in detail.

4. Summary

The case of the hexagonal grid is simple. We used one natural distance function (which is used in the hexagonal grid almost every time in digital geometry), based on the natural neighbouring relation. We used three coordinate values to describe the grid, as in [8]. An algorithm was presented to solve the shortest path problem in this grid using our concept of lanes. With the help of the presented shortest path we calculated the distance between two objects, and we repeated the result of [9] in symmetric form using three coordinates. The advantage of this grid over the triangular and rectangular grids is that the methods used for approximating Euclidean distance are not that difficult. So the hexagonal grid is useful and offers an easy way to make digital geometry look similar to Euclidean.

We analysed three types of neighbourhood relations in the triangular grid. We introduced the concept of neighbourhood sequences, and with their help, we were able to define distance functions on the triangular grid. We presented an algorithm, which constructed the shortest path from one point to another, and calculated their distance value with respect to a given neighbourhood sequence. We also examined the complexity of our algorithm. Examining the distance functions generated by neighbourhood sequences, we showed that these distances did not prove to be a metric for every sequence. In [14] we presented a sufficient and necessary condition for neighbourhood sequences to determine metric spaces. Since we have two types of points on the triangular grid, we had to face some additional problems that are not present in the square and the hexagonal grids.

References

[1] P.P. DAS, P.P. CHAKRABARTI, B.N. CHATTERJI, Generalized distances in digital geometry, *Inform. Sci.* **42** (1987), pp. 51–67.

[2] P.P. DAS, P.P. CHAKRABARTI, B.N. CHATTERJI, Distance functions in digital geometry, *Inform. Sci.* **42** (1987), pp. 113–136.

[3] P.P. DAS, Lattice of octagonal distances in digital geometry *Pattern Recognition Lett.* **11** (1990), pp. 663–667.

[4] P.P. DAS, B.N. CHATTERJI, Octagonal Distances for Digital Pictures, *Inform. Sci.* **50** (1990), pp. 123–150.

[5] P.P. DAS, Best Simple Octagonal Distances in Digital Geometry, *J. Approx. Theory* **68** (1992), pp. 155–174.

[6] E.S. DEUTSCH, Thinning algorithms on rectangular, hexagonal and triangular arrays, *Comm. ACM*, **15** (1972), pp. 827–837.

[7] A. FAZEKAS, A. HAJDU, L. HAJDU, Lattice of generalized neighbourhood sequences in nD and ∞D, *Publ. Math. Debrecen* **60** (2002), pp. 405–427.

[8] I. HER, Geometric transformations on the hexagonal grid, *IEEE Trans. Image Process.* **4** (1995), pp. 1213–1221.

[9] E. LUCZAK, A. ROSENFELD, Distance on a Hexagonal Grid, *IEEE Trans. Computers* (1976), pp. 532–533.

[10] B.H. MCCORMICK, The Illinois pattern recognition computer–ILLIAC III, *IEEE Trans. Electronic Computers* **12** (1963), pp. 791–813.

[11] R.A. MELTER, A survey of digital metrics, *Contemporary Mathematics* **119** (1991), pp. 95–106.

[12] B. NAGY, Finding Shortest Path with Neighbourhood Sequences in Triangular Grids, *ITI–ISPA 2001, Pula, Croatia*, pp. 55–60.

[13] B. NAGY, Neighbourhood Sequences in Triangular Grids, *Technical Reports No. 2001/16, University of Debrecen.*

[14] B. NAGY, Metrics Based on Neighbourhood Sequences in Triangular Grids, *Pure Math. Appl.* **13** (2002), pp. 259–274.

[15] B. NAGY, Distance functions based on generalized neighbourhood sequences in finite and infinite dimensional space, *ICAI'01, 5th International Conference on Applied Informatics, Eger, 2001*, pp. 183–190.

[16] A. ROSENFELD, R.A. MELTER, Digital Geometry, *Math. intelligencer* **11** (1989), pp. 69–72.

[17] A. ROSENFELD, J.L. PFALTZ, Distance functions on digital pictures, *Pattern Recognition* **1** (1968), pp. 33–61.

[18] M. YAMASHITA, T. IBARAKI, Distances defined by neighborhood sequences, *Pattern Recognition* **19** (1986), pp. 237–246.

Received: October, 2002
Revised: March, 2003
Accepted: May, 2003

Contact address:
Benedek Nagy
Institute of Mathematics and Informatics
University of Debrecen, Hungary
e-mail: nbenedek@math.klte.hu

BENEDEK NAGY was born on July 28, 1973. He received BSc degree in programming mathematics and MSc degrees in physics, in philosophy (with spec. logic), in computer science and in general and applied linguistics in 1996–2000 from the University of Debrecen (Kossuth University before 2000), Hungary. He was a PhD scholar in mathematics and informatics from 1998 to 2001. Since 2001 he is a lecturer at the Department of Computer Science in University of Debrecen. Since 2003 he is an associate professor.

After graduating, he was a scholar for one semester at the Free University, Berlin, Germany in 1999; at the Indiana University, Bloomington, IN, USA in 2002 and at the University of Rovira i Virgili, Tarragona, Spain in 2003.

His research fields are artificial intelligence, logical systems and digital geometry.