MULTIDIMENSIONAL BINARY INDEXING FOR NEIGHBOURHOOD CALCULATIONS IN SPATIAL PARTITION TREES

José POVEDA, Michael GOULD
Departamento de Lenguajes y Sistemas Informáticos
Universitat Jaume I
E-12071, Castellón, Spain
{albalade, gould}@uji.es

ABSTRACT

We present a binary array encoding (location arrays) of the nodes in a spatial partition tree representing spaces of dimension k. This framework facilitates tree traversal for optimising access speed, and also supports simplified calculation of the neighbourhood of a subinterval of a particular partition. After defining the encoding we present a neighbour determination algorithm which extends work carried out by Samet [1] [2] and others on quadtrees. The primary extension is that the encoding and the neighbour determination algorithm extend to arbitrary dimensions beyond the 2-d quadtree case.

KEY WORDS

Spatial partition tree, binary location array, multidimensional indexing, multiresolution, neighbour calculation, terrain visualization

1. INTRODUCTION[1]

Quadtrees represent the primordial hierarchical spatial data structure, whose common salient feature is that of recursive decomposition of 2-d space. Quadtree types can be differentiated by the following characteristics:

1. The type of data used for its representation
2. The principle guiding the process of partitioning
3. The resolution (variable or not).

This data structure may be used for representing and handling point, curve, region and volume data. Decomposition of space can be regular at all the levels of the quadtree or non-regular, guided by the entry of the data to represent. The resolution of the decomposition (the number of times that subdivision is applied) can be set beforehand or can be guided for the properties of the input data. Depending on the specific application also a distinction can be made whether or not the structure is to define the border of a region, in the case of curves or surfaces, or if it is used to define its interior in the case of areas and volumes.

One of the principal variants of the quadtree data structure is the region quadtree, so much so that Samet [1] uses the terms synonymously. As an example we present the region in figure 1 that initially we have represented like a structure of a binary array of 8 x 8 where the value 1 represents a pixel inside the region to represent and 0 if the pixel is found outside the region.

In the corresponding tree structure in figure 1, the root node corresponds to the complete array. Each son node represents a quadrant (labelled NW, NE, SW, SE) of the region represented by the node. The terminal nodes correspond to the blocks of the array in which it is no longer necessary to continue subdividing. A terminal node is represented in white or in black depending on whether its corresponding block is completely located inside the area to represent (all the elements of the array in this block contain the label "1") or that block is found completely outside the region of interest (all the elements of the array belonging to that block are labelled with "0").

These same spatial partitioning concepts extend to the third dimension in the representation of solids with octrees [3], although the encoding of blocks instead of quadrants becomes more complicated. In this paper we refer to spatial partition trees, in general, even though our initial examples treat the common 2-dimensional case using quadtrees. This is because the method and algorithm proposed here does not restrict spatial partitioning to 2-d but rather is extensible to spatial partitioning in arbitrary dimensions

2. MULTIRESOLUTION VISUALIZATION

Let us look at an application case where neighborhood calculation using a quadtree structure is a crucial element in interactive terrain visualization. For this application we propose to optimize the visualizatin of an extensive Digital Terrain Model (DTM) utilizing management of levels of detail (LOD), that is, multirresolution.

[1] The majority of this introductory section was adapted from Samet [1] [2].

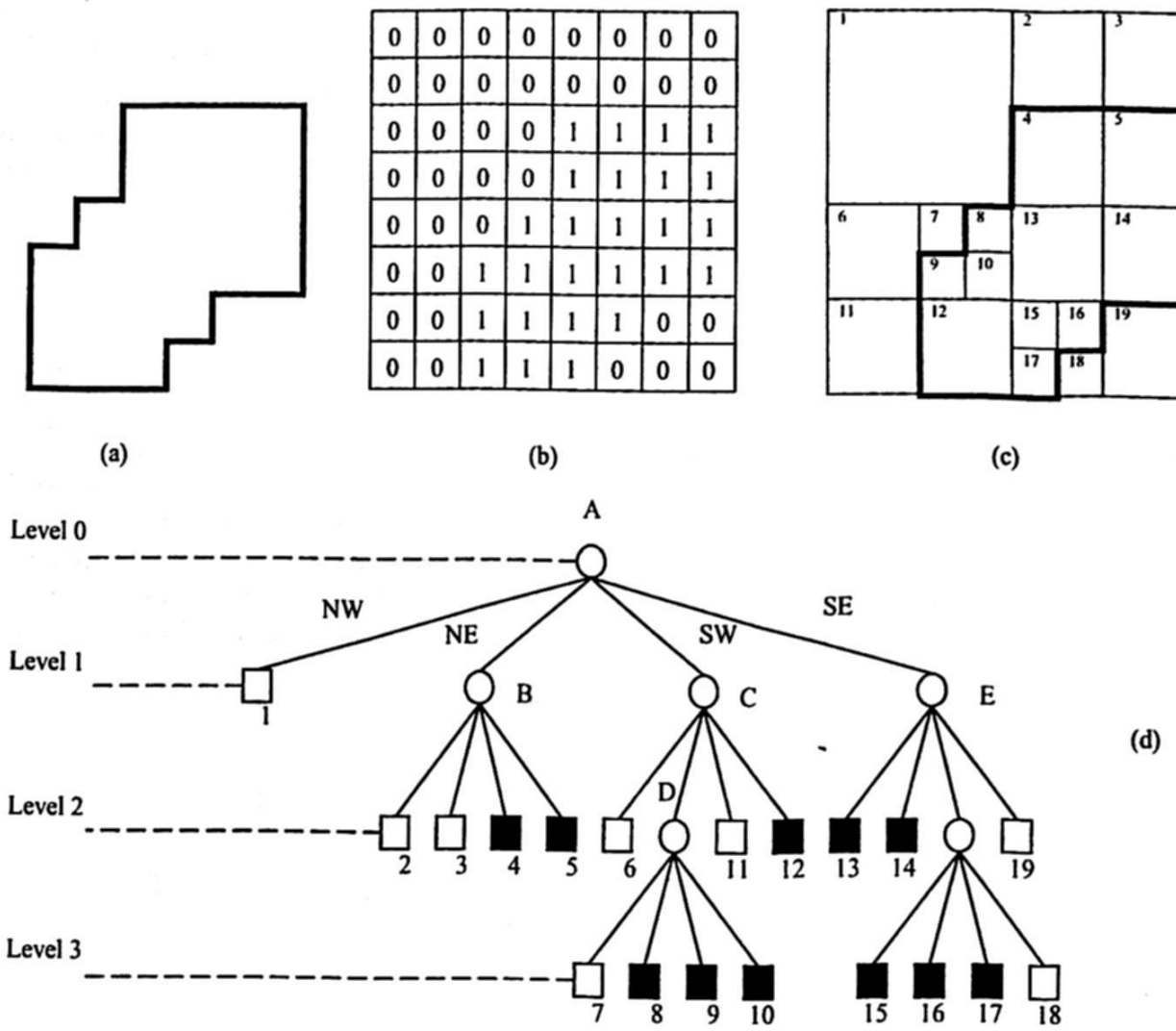

Figure 1. Example of quadtree (a) region, (b) structure of binary array, (c) maximum area blocks with constant value, (d) quadtree repres entation.

With this goal we define a process which renders to the screen relevant information, that is, only the information which might be appreciated from the viewpoint of the user. The data structure considered to manage the MDT with multiple levels of detail is the quadtree. In the quadtree nodes we store the elevation values which constitute the MDT as a regular grid with different levels of resolution, where each level of the tree corresponds to a level of resolution of the terrain.

In order to visualize the terrain, given a particular viewpoint, we traverse the quadtree and select the different levels of resolution represented by different levels in the tree. For the selection of nodes in the tree we use as main criterion the distance to the viewpoint, along with terrain roughness criteria. To insure against union of intervals with levels of resolution greater than 1, we run a balancing procedure over the pruned tree. Obtaining smaller regions after the pruning of larger adjacent regions, causes discontinuities (gaps) in the graphic representation of the DTM which cannot easily be corrected unless the difference in levels of resolution is no greater than one, see figure 2, in which case we make the correction interpolating the elevation of point b, taken from the region with higher resolution, which provoked the discontinuity, to the segment ac of the neighbouring region with lower resolution. For this process, therefore, it is necessary to calculate the intervals which are neighbouring a given interval. Analogous to the multi-resolution management, mentioned for the simplification of terrain geometry, it is also necessary to manage the simplification of the texture which overlays that geometry.

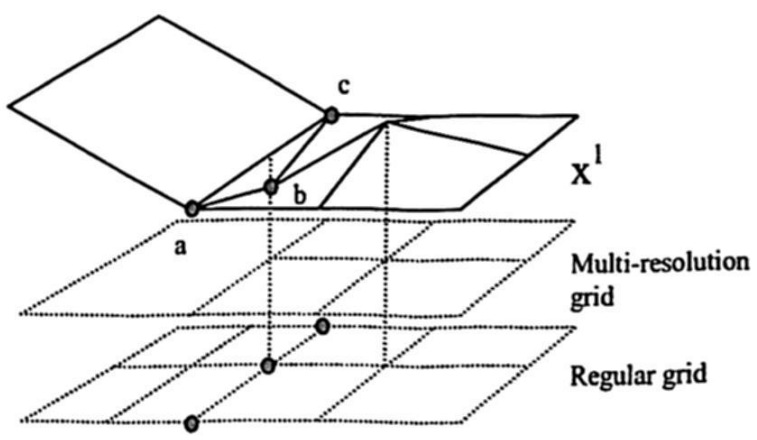

Figure 2. The union of differing levels of detail (LOD) here provokes a discontinuity in b: perspective view of the different layers utilized for generating the terrain.

3. BINARY ENCODING OF PARTITIONS

Frequently GIS related operations will require rapid access to a specific node in the tree. This normally requires traversal from root node downward through the tree, passing from node to node depending on the area representing the quadrant in question; this in turn supposes that position information is carried for each node. Normally quadtree quadrants are indexed according to a hierarchical base-4 ordering such as that proposed by Morton (the Z order in figure 1d). While this ordering simplifies operations in 2-d space and also facilitates human description in terms of four cardinal directions, a simpler referencing method is desirable as dimension increases. Here we describe such a method.

To increase access simplicity it is possible to apply binary encoding [4] to nodes so that these codes determine the traversal path down the tree. Rather than suppose four cardinal directions we encode each dimension with a 1-bit coordinate, describing the direction as positive or negative with respect to a given reference system. In particular, in two dimensions the positive directions --up and to the right (or East)-- are represented by a 1, while the negative directions --down and left-- are represented by 0. Consider the example in figure 3, where the "northwest"16 quadrant at level 1 is assigned a 0 (negative direction) in the x1 dimension and a 1 (positive direction) in the x2 dimension. In this 2-d case the assumed reference system has its origin in the central point of the area where the quadrants meet. The example in figure 4 expands on this binary encoding, showing the notation to three levels.

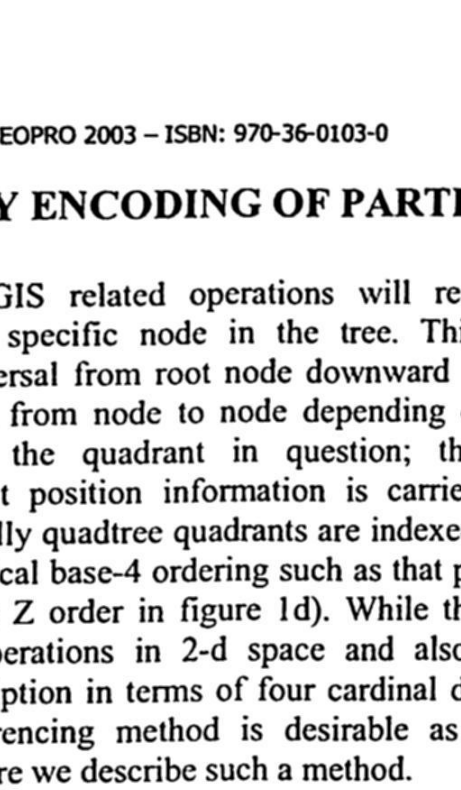

Figure 3. **Binary array encoding in the 2-d case.**

[16] While it is all too tempting to use cardinal references, we again underscore that the positive/negative direction notation applies beyond the 2-d case equally to all dimensions.

4. BINARY LOCATION ARRAYS

Taking into consideration the binary encoding of each dimension we define, as shown in figure 4, an array representation of each subset region which communicates both the location in the tree structure and, implicitly, the path to reach the node associated with that region.

The location array associated with a given node (here quadrant) is defined as the array of the father node, then adding on the right a new column vector with its own encoding according to the system in figure 3. The 3rd level node highlighted in figure 4 is thus represented by the 1,1 of the grandfather, the 1,0 of the father and then adding the 1,1 for its own level. Figure 5 shows additional examples to help clarify the encoding proposed. The location array representation of the shaded regions in figure 5 (four levels in 2-d space) would be the following:

$$A=\begin{bmatrix}0\\1\end{bmatrix};\ B=\begin{bmatrix}1\\1\end{bmatrix}$$

$$C=\begin{bmatrix}0&0\\0&1\end{bmatrix};\ H=\begin{bmatrix}0&0\\0&0\end{bmatrix}$$

$$I=\begin{bmatrix}0&1\\0&0\end{bmatrix};\ D=\begin{bmatrix}1&0&0\\0&1&1\end{bmatrix}$$

$$E=\begin{bmatrix}0&1&1&1\\0&1&0&1\end{bmatrix};\ F=\begin{bmatrix}1&0&0&0\\0&1&0&1\end{bmatrix}$$

$$G=\begin{bmatrix}0&1&1&1\\0&1&0&0\end{bmatrix}$$

Encoding higher dimensions follows the same principle described above. For example the following array P shows 5 levels of nodes (columns) in 4-d space (rows).

$$P=\begin{bmatrix}0&1&0&1&0\\0&1&1&1&1\\1&1&1&1&0\\0&0&1&1&1\end{bmatrix}$$

Because the location of the quadrant[17] (node) is defined by this simple binary location array, in section 6 we describe a sort of bit shifting on these arrays for simple neighbour calculation in any arbitrary dimension.

[17] Here the use of the term quadrant re fers to any interval of the spatial partition, at any dimension, and associated in the tree as a node.

5. MULTIDIMENSIONAL NEIGHBOUR CALCULATION

To achieve maximum data access efficiency and simplicity, necessary for applications such as interactive terrain visualization, it is useful to be able to reason about neighbour locations in the tree, identifying all possible neighbours of each node. These neighbourhood references are useful for tasks such as database paging or for adjusting resolution in a geometric model during an interactive flight over terrain [5] [6] [7]. If the terrain model includes multiple levels of resolution as a function of attributes such as distance to the viewer, then for each frame it would be necessary to recalculate the spatial partition tree and the neighbour relations of each node because quadrants of distinct levels of decomposition can cause discontinuities (gaps) which must be corrected by adjustment to the lower resolution. Because these applications will need to continuously recalculate these neighbour relations, it is essential that the algorithm for doing so deal efficiently with neighbours at multiple resolutions.

As a solution to clarify and optimise neighbour identification, we utilize the location array described above, in a simple algorithm of linear computational cost with respect to the depth of the tree.

To calculate the neighbour of E (in figure 5) in the positive direction at dimension 1 and at the same level of resolution (that is to say, quadrant of equal size or node at the same level), we begin with the location array of E and we visit each element, indicated with a dot, in the top row from right to left. Alternating the bits as we go along the first 1 becomes a 0, the second 1 a 0, the third also a 0, then the 0 a 1. When we negate a 0 and it switches to 1, at any point in the row, the process stops and the resulting location array describes the neighbour found, in this case F.

$$E=\begin{bmatrix}0 & 1 & 1 & \dot{1}\\ 0 & 1 & 0 & 1\end{bmatrix}\Rightarrow\begin{bmatrix}0 & 1 & \dot{1} & 0\\ 0 & 1 & 0 & 1\end{bmatrix}\Rightarrow\begin{bmatrix}0 & \dot{1} & 0 & 0\\ 0 & 1 & 0 & 1\end{bmatrix}$$

$$\Rightarrow\begin{bmatrix}\dot{0} & 0 & 0 & 0\\ 0 & 1 & 0 & 1\end{bmatrix}\Rightarrow\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 1\end{bmatrix}=F$$

Now, if we continue applying the same algorithm we may calculate at the same level of resolution the neighbour of E in the negative direction in the second dimension ("down"), producing the location array which represents region G (see figure 5). This time we have alternated the bits in the second row (representing the second dimension), until a 0 is obtained, at the first move in this case.

$$E=\begin{bmatrix}0 & 1 & 1 & 1\\ 0 & 1 & 0 & \dot{1}\end{bmatrix}\Rightarrow\begin{bmatrix}0 & 1 & 1 & 1\\ 0 & 1 & 0 & 0\end{bmatrix}=G$$

In general terms, then, we can state that in order to search for a neighbour in any positive direction, the stop condition in the bit shifting is the negation of any 0 to a 1. Equally then, to search for a neighbour in any negative direction the stop condition is the negation of any 1 to a 0.

Algorithm 1a (Neighbour identification in direction $+x^i$)

1.- First element processed: last element of the row i, A(i,j) in the location array.

2.- **IF** the processed element is a 0, we negate this element and the process stops having encountered a 1. The resultant array is the location associated with the neighbouring node in direction x^i.

3.- **ELSE** the processed element negates to a 0 and repeat step 2 with the next element to the left of that row.

For the case of a neighbouring node in the opposite (negative) direction, the algorithm would be modified as follows.

Algorithm 1b (Neighbour identification in direction $-x^i$)

1.- First element processed: last element of the row i, A(i,j) in the location array.

2.- **IF** the processed element is a 1, we negate this element and the process stops having encountered a 0. The resultant array is the location associated with the neighbouring node in direction x^i.

3.- **ELSE** the processed element negates to a 1 and repeat step 2 with the next element to the left of that row.

The principal advantages of this neighbour calculation algorithm are its generality with respect to spatial dimensions and its simplicity. An alternate representation of the algorithm (for either direction) is the given Calculate Neighbour algorithm.

6. THE MULTIRESOLUTION CASE

As stated earlier, applications such as real-time terrain generation and visualization depend on the optimal determination of multiple levels of resolution for any particular view [6], [7], [8], [9], [10]. Let us now demonstrate a method for exploiting the location array notation described above, to determine the neighbours of X (in figure 6) at multiple levels of resolution.

Algorithm CalculateNeighbour (A[i][j], $\pm e_k$)

Input. Location array A[i][j] of the region quadtree of interest, in direction e_k, where r is the level of resolution.
Output. The array of the neighbour encountered in direction e_k

```
1.- if (ek >0) then
2.-   while (A[k][r]==1)||(r==0)
3.-       A[k][r] <-not(A[k][r]);
4.-       r<-r-1;

5.-   A[k][r] <-not(A[k][r]);
6.- else
7.-   while (A[k][r]==0)||(r==0)
8.-
9.-       A[k][r] <-not(A[k][r];
          r<-r-1;

10.-  A[k][r] <-not(A[k][r]);
11.-return A
```

The array representation of the shaded regions in figure 6 would be the following:

$$X=\begin{bmatrix}0&1&1&1\\0&1&0&0\end{bmatrix}$$

$$A=\begin{bmatrix}0&1&1&1\\0&1&0&1\end{bmatrix};\ B=\begin{bmatrix}0&1&1&1\\0&0&1&1\end{bmatrix}$$

$$C=\begin{bmatrix}0&1&1&0\\0&1&0&0\end{bmatrix};\ D=\begin{bmatrix}1&0&0&0\\0&1&0&0\end{bmatrix}$$

$$G=\begin{bmatrix}1&0&0\\0&1&0\end{bmatrix};\ F=\begin{bmatrix}0&1&1\\0&0&1\end{bmatrix}$$

$$H=\begin{bmatrix}1&0\\0&1\end{bmatrix};\ E=\begin{bmatrix}0&1\\0&0\end{bmatrix};\ I=\begin{bmatrix}1\\0\end{bmatrix}$$

We first calculate for the first dimension the same-resolution neighbour of region X, in the positive direction, yielding D as this "eastern" neighbour.

$$X=\begin{bmatrix}0&1&1&\dot{1}\\0&1&0&0\end{bmatrix}\Rightarrow\begin{bmatrix}0&1&\dot{1}&0\\0&1&0&0\end{bmatrix}$$

$$\Rightarrow\begin{bmatrix}0&\dot{1}&0&0\\0&1&0&0\end{bmatrix}\Rightarrow\begin{bmatrix}\dot{0}&0&0&0\\0&1&0&0\end{bmatrix}$$

$$\Rightarrow\begin{bmatrix}1&0&0&0\\0&1&0&0\end{bmatrix}=D$$

In the next step we identify all sub-arrays which represent neighbouring regions at lower dimensions. This is accomplished merely by eliminating the final column on the right side. The number of columns remaining informs of the level of resolution of each neighbouring region. In this case we obtain the four arrays associated with the neighbouring regions of X in the positive direction at the first dimension: D at the same resolution (4th level), G at the 3rd level, H at the 2nd level and finally I at the 1st level (or entire quadrant).

$$\begin{bmatrix}1&0&0&0\\0&1&0&0\end{bmatrix}=D;\begin{bmatrix}1&0&0\\0&1&0\end{bmatrix}=G$$

$$\begin{bmatrix}1&0\\0&1\end{bmatrix}=H;\begin{bmatrix}1\\0\end{bmatrix}=I$$

Applying again the algorithm in the first dimension we calculate the neighbours of region X in the negative direction:

$$X=\begin{bmatrix}0&1&1&\dot{1}\\0&1&0&0\end{bmatrix}\Rightarrow\begin{bmatrix}0&1&1&0\\0&1&0&0\end{bmatrix}=C$$

We obtain in the first step, the array characterizing the only neighbour of G in the negative direction. Repeating the process for the second dimension (the bottom row), in the positive direction, we obtain:

$$X=\begin{bmatrix}0&1&1&1\\0&1&0&\dot{0}\end{bmatrix}\Rightarrow\begin{bmatrix}0&1&1&1\\0&1&0&1\end{bmatrix}=A$$

and for the neighbours in the negative direction:

$$X=\begin{bmatrix}0&1&1&1\\0&1&0&\dot{0}\end{bmatrix}\Rightarrow\begin{bmatrix}0&1&1&1\\0&1&\dot{0}&1\end{bmatrix}$$

$$\Rightarrow\begin{bmatrix}0&1&1&1\\0&\dot{1}&1&1\end{bmatrix}\Rightarrow\begin{bmatrix}0&1&1&1\\0&0&1&1\end{bmatrix}=B$$

Therefore, we have determined in three steps that B is the neighbour of X at the same resolution and then, independent of the resolution, that X has three neighbouring regions which we deduced from the original array, as follows: B at the 4th level, F at the 3rd level and E at the 2nd level. The objective region X has no neighbour at the 1st level of resolution in this negative direction.

$$\begin{bmatrix}0&1&1&1\\0&0&1&1\end{bmatrix}=B$$

$$\begin{bmatrix}0&1&1\\0&0&1\end{bmatrix}=F;\begin{bmatrix}0&1\\0&0\end{bmatrix}=E$$

Although beyond the third dimension we cannot easily visualize geometrically the indexing concepts described here, from a logical point of view we can demonstrate how the method applies equally well in k-dimensions. This in contrast to traditional quadrant notation following cardinal directions which are restricted to the 2-d plane.

To illustrate this point we will calculate, in a 4-d space (using the array P from section 4), the neighbours in the positive direction and in the 4th dimension, and additionally at all levels of resolution.

$$P=\begin{bmatrix}0&1&0&1&0\\0&1&1&1&1\\1&1&1&1&0\\0&0&1&1&1\end{bmatrix}$$

We apply the neighbour algorithm described earlier:

$$P=\begin{bmatrix}0&1&0&1&0\\0&1&1&1&1\\1&1&1&1&0\\0&1&0&1&\dot{1}\end{bmatrix}\Rightarrow\begin{bmatrix}0&1&0&1&0\\0&1&1&1&1\\1&1&1&1&0\\0&1&0&\dot{1}&0\end{bmatrix}$$

$$\Rightarrow\begin{bmatrix}0&1&0&1&0\\0&1&1&1&1\\1&1&1&1&0\\0&1&\dot{0}&0&0\end{bmatrix}\Rightarrow\begin{bmatrix}0&1&0&1&0\\0&1&1&1&1\\1&1&1&1&0\\0&1&1&0&0\end{bmatrix}$$

As is illustrated above, the process stops when a 0 to 1 bit shift occurs. We then determine using the method described earlier, that there exist three regions neighbouring P at 3 levels of resolution, described by the following arrays:

$$P_1=\begin{bmatrix}0&1&0&1&0\\0&1&1&1&1\\1&1&1&1&0\\0&1&1&0&0\end{bmatrix}$$

$$P_2=\begin{bmatrix}0&1&0&1\\0&1&1&1\\1&1&1&1\\0&1&1&0\end{bmatrix}\text{ and }P_3=\begin{bmatrix}0&1&0\\0&1&1\\1&1&1\\0&1&1\end{bmatrix}$$

7. MULTIPLE NEIGHBORHOODS

Based on the neighbourhood algorithm and the location matrix notation of an interval I, we have seen how to calculate the location matrix of the neighbours of I at the same level of resolution. Based on that matrix we have also seen how to determine all other possible neighbours at lower resolution than I for a given quadtree partition. We now complete this section with the determination of the neighbours of I at higher resolution, which implies necessarily multiple vecinity in the same dimension. To illustrate this point we begin with the following quadtree partition, in figure 7.

We propose that the calculus of the neighbours of interval X in the positive direction and in the second spatial dimension, as seen in figure 7, results in the intervals A, B and C, whose associated location matrices are:

$$X=\begin{bmatrix}1&0\\0&0\end{bmatrix};\ A=\begin{bmatrix}1&0&0&0\\0&1&0&0\end{bmatrix}$$

$$B=\begin{bmatrix}1&0&0&1\\0&1&0&0\end{bmatrix};\ C=\begin{bmatrix}1&0&1\\0&1&0\end{bmatrix}$$

On the basis of the location matrix defining the interval X, we calculate the neighbouring interval in the positive

direction of the second dimension and at the same level of resolution as I.

Once the location matrix of the neighbour interval at same resolution is obtained, we verify its existence in the quadtree to determine if it is a terminal node or, on the contrary, possess son nodes. For the terminal node case, the process concludes obtaining the location matrix of the neighbour of X in the direction and dimension desired. For the case of son nodes, we repeat the process for only those sons in the opposite direction of the dimension of interest. Below we show how to calculate neighbour location matrices, using the tree structure navigation described.

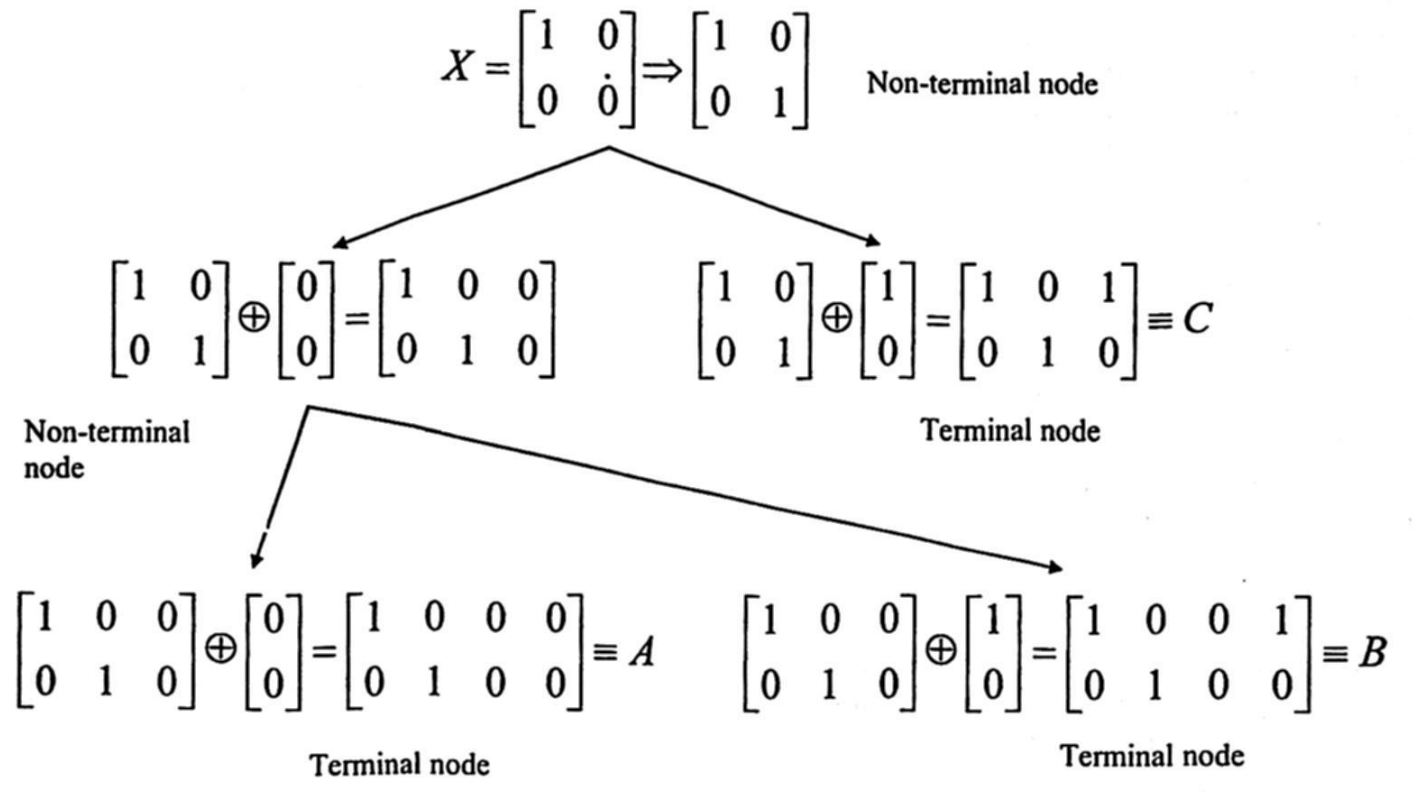

Figure 8. Calculus of the neighbours of X at higher resoluti on in the second dimension and in the positive direction

For the case of neighbours of X in the negative direction and the first dimension, we would derive an analogue process:

Therefore the terminal nodes represent the neighbours of interval X at equal or higher resolution. For the case of the other two directions which for the moment we have not considered, we show how we would arrive at singular cases. In the first place let us calculate the neighbours of X in the negative direction of the second dimension.

$$X=\begin{bmatrix}1 & 0\\ 0 & \dot{0}\end{bmatrix}\Rightarrow\begin{bmatrix}1 & 0\\ \dot{0} & 1\end{bmatrix}\Rightarrow\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}$$

As we did not encounter any unitary (one) value, we deduce that the interval X is on the border for the negative direction at the second dimension, and therefore has no neighbours in this partition. For the case of neighbours in the positive direction of the first dimension:

$$X=\begin{bmatrix}1 & \dot{0}\\ 0 & 0\end{bmatrix}\Rightarrow\begin{bmatrix}1 & 1\\ 0 & 0\end{bmatrix}$$

in which being a terminal node, concludes the defined process and thus we deduce that it is the only neighbour of X in the positive direction of the first dimension of equal or greater resolution. We conclude the calculation of neighbours of X at level of detail equal or less than X, identifying as neighbours all those terminal nodes obtained in the trees created. We are left with the location matrices obtained as neighbours of X in the positive direction of the second dimension, those corresponding to the matrices:

$$\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}\equiv C\ ;\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{bmatrix}\equiv A$$

$$\begin{bmatrix}1 & 0 & 0 & 1\\ 0 & 1 & 0 & 0\end{bmatrix}\equiv B$$

and for the negative direction of the first dimension

$$\begin{bmatrix}0 & 1 & 1\\ 0 & 0 & 0\end{bmatrix}\equiv D\ ;\begin{bmatrix}0 & 1 & 1 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}\equiv E$$

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \equiv F ,$$

as may be verified in the figure 7, where we have labelled the intervals which are neighbours of X.

8. COMPLEXITY EVALUATION

Finally, we provide a proposition in order to define the complexity of the neighbour calculation algorithm described in this paper.

Proposition

Given that T is a quadtree of depth h, the neighbour of a given node n in a particular direction can be encountered at a maximum cost of O(h).

Demonstration

In the best case, in which a neighbour in a given direction has the same father in the tree, the computational cost will be of one comparison, such as in the case of E, G in figure 5.

In the worst case, that of E, F in figure 5, according to the algorithm presented here the number of operations (comparison/assignment) will be at maximum that of the level of resolution of the region considered, and therefore corresponding at maximum to the depth of the tree.

9. CONCLUSIONS AND FUTURE WORK

We have presented a binary encoding notation for representing spatial partitions, and therefore have introduced the concept binary location arrays. Secondly, we presented an algorithm for exploiting this notation for the simplified calculation of neighbours of any region, in arbitrary dimensions. This notation provides substantial generality over current quadtree notations which do not extend well into higher dimensions. Thirdly, we demonstrated how the neighbour calculation algorithm extends to determine neighbours at any level of resolution.

Future work may include application of the algorithm to interactive terrain visualization and other potentially interesting applications, and extensions for other data structures.

ACKNOWLEDGEMENTS

This work was partially supported by European Union projects Esprit 25029 (GeoIndex) and IST-2001-37724 (ACE-GIS).

REFERENCES

[1] Samet, Hanan, "The Quadtree and Related Hierarchical Data Structures", ACM Computing Surveys 16, 1984, pp. 187-200.

[2] Samet, H. "Applications of Spatial Data Structures", Addison-Wesley, 1995.

[3] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, "Computational Geometry, Algorithms and Applications", Springer, Second Edition, 2000, pp. 291-303.

[4] Goodchild, M.F. (Ed.), 1990. "Quadtree algorithms and spatial indexes". Technical issues in GIS. NCGIA Core Curriculum. Unit 37.

[5] De Berg, M., Dobrindt, K. T. G., "On levels of Detail in Terrains", 11th ACM Symposium on Computational Geometry", 1995, pp. 1-15.

[6] Falby, J. S., Zyda, M. J., Pratt, D. R., and Mackey, R. L., "Hierarchical Data Structures for Real-Time Three-Dimensional Visual Simulation", Computer and Graphics 17, 1993, pp. 144-151.

[7] De Floriani, L. and Puppo E., "Hierarchical Triangulation for Multiresolution Surface Description", ACM Transactions on Graphics 14, 1995, pp. 363-411.

[8] Poveda, J., Gould M., Sevilla, J. "GeoIndex Project: A virtual window to Geographic Informations Systems". In Rault, J.-C. (Ed.): La Lettre de l'Intelligence Artificielle, Proceedings of the International Conference on Complex Systems, Intelligent Systems & Interfaces (NIMES'98), Nimes, France. Nimes 1998.

[9] Schee, L. H., Jense, G. L., "Interacting with Geographical Information in a Virtual Environment", Proc. Joint European Conference on Geographical Information. The Hague, NL, 1995, pp. 151-156.

[10] Schroder, F. and Rossbach, P., "Managing the Complexity of Digital Terrain Models", Computer and Graphics 18, 1994, pp. 775-783.

12. ANNEX OF FIGURES

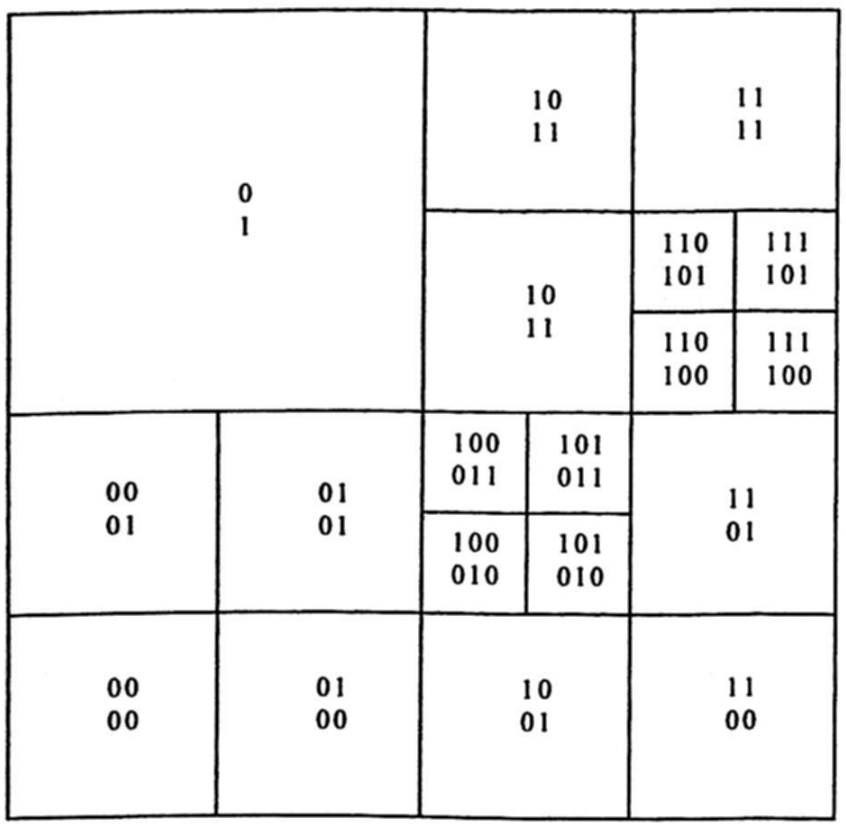

Figure 4. Binary location arrays of a quadtree to the 3rd level

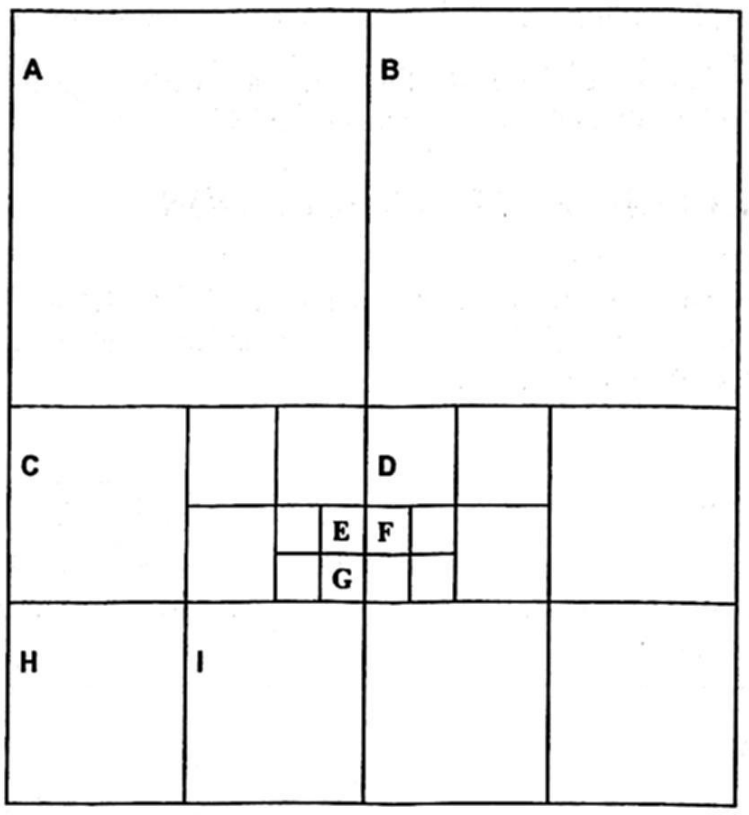

Figure 6. Study case of neighbours of X at several levels of resolution

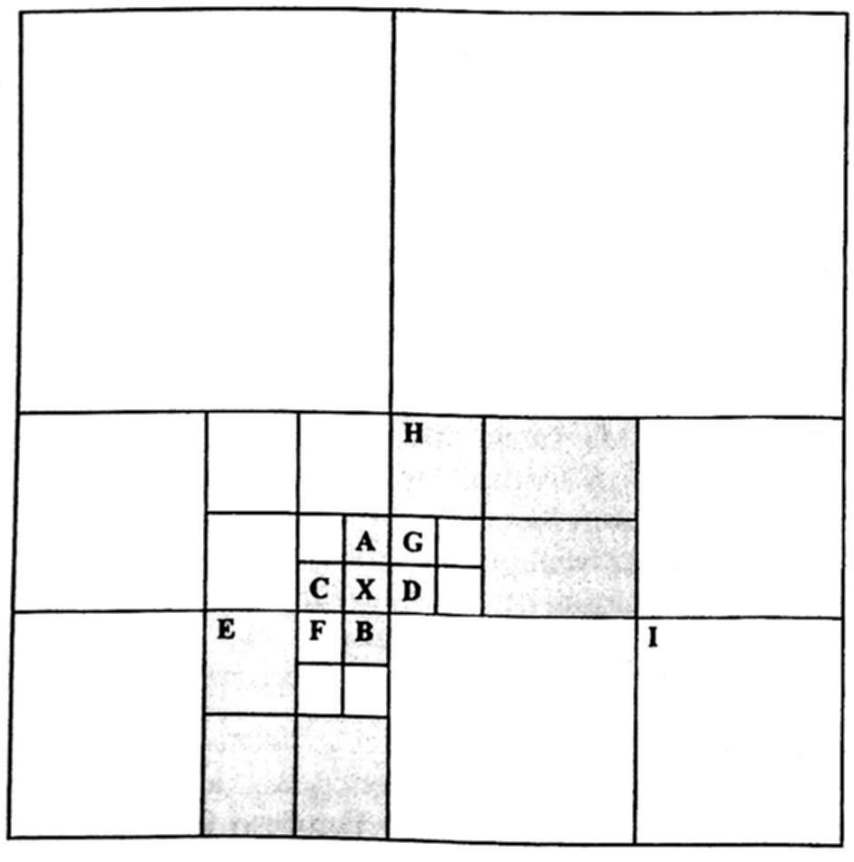

Figure 5. Region quadtree example

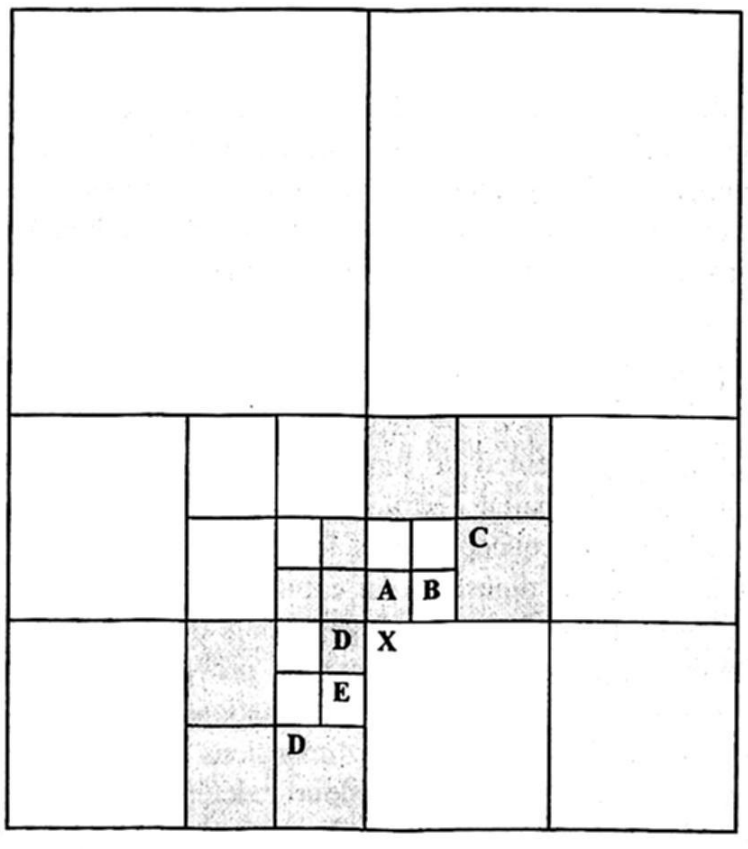

Figure 7. Partition used for calculus of ne ighbours of interval X in the positive direction of the second spatia l dimension. The result should be A, B and C, each a different (lower) levels of resolution.