Acta Mathematica Academiae Paedagogicae Nyíregyháziensis
16 (2000), 79–87
www.emis.de/journals

SKELETONIZATION ON QUADTREE REPRESENTED IMAGES

Attila Fazekas and István Sánta
fattila@math.klte.hu, santai@dragon.klte.hu
University of Debrecen
4010, Debrecen PO Box 12, Hungary

Abstract. The most frequent method to represent a digital image is the pixel-pixel type representation. Because of this, most image processing algorithms operate on pixel-pixel type images. Accordingly few years, claim arised to develop image processing algorithms, which operate on some special representation known from the literature. In this paper, we describe the theoretical background, which is required to apply a special image processing procedure – the skeletonization – to the quadtree representation.

Keywords. Digital geometry, octagonal distance

AMS Subject Classification. 68U10 Image Processing

INTRODUCTION

In digital image processing, skeletonization is a frequently used technique, when line drawings are processed. Without the knowledge of the exact notion of skeletonization, these procedures substitute the original object with its one pixel width "axis", whose connectedness corresponds to the connectedness of the original object. The theoretical results characterizing the exact, mathematical notion and most significant features of the skeleton are in [6], while the general analysis of basic skeletonizing techniques can be found in [4].

In the literature a number of methods are known to represent a digital image. The most frequent one is the pixel-pixel type representation, when the geometrical position and the intensity of the pixels of the image are represented in a matrix. The coordinates of the elements of the matrix denote the geometrical position and their values denote the intensity of the pixels. Furthermore, run-length coding [5], quadtree [1] and contour-representation [2] are used. In medical image processing only the perpendicular projections of binary images are available. When certain conditions are satisfied, we can consider these projections as a representation of an image [3].

In digital image processing most procedures operate on pixel-pixel type images. The reason is that almost all the available hardwares support this representation. This is also true for the skeletonizing algorithms, since they are special image processing procedures. However, we can find other algorithms in the literature which perform the skeletonizing operation on run-length coded images [5] or on projections of images [3].

In the last few years it became more and more interesting to develop image processing algorithms, which operate on some special representations known from the literature. In this paper we develop the theoretical background of skeletonization, which operates on such a special image representation – the quadtree representation.

The quadtree representation represents the digital image as the union of single-coloured squares of different sizes. The geometrical positions of these squares are described in a tree data structure. Since these squares contain more pixels of the same colour, applying this representation the memory size storing the image can be considerably reduced.

In the next chapter we describe the basic notions related to quadtree representation. We consider the possibility of the realization of two elementary operations applied in most skeletonization algorithms – to test the intensity of a pixel and to alter the intensity of a pixel – in quadtree representation.

1. BASIC NOTIONS AND DEFINITIONS

Definition 1.1. $\mathbf{Z}^n$ *is called the* n*-dimensional digital plane and its elements are points. A non-empty subset of the* n*-dimensional digital plane is called an* n*-dimensional digital set. If* X *is an* n*-dimensional digital set, then the function* $f : X \longrightarrow \{0, 1, \dots, m-1\} (m \in \mathbf{N})$ *is an* m*-level digital image on* X*. The set* X *is the coordinate set of* f*,* $\{0, 1, \dots, m-1\}$ *is the range of* f*. The ordered pair* $(p, f(p))$ $(p \in X)$ *is called a pixel, where* p *is the coordinate and* $f(p)$ *is the intensity of the pixel.*

Definition 1.2. *Let* X *be a 2-dimensional digital set. Then the digital image* $f : X \longrightarrow \{0, 1\}$ *is called a binary image. If* $X = [k, N-1+k] \times [h, N-1+h]$ $(k, h \in \mathbf{Z})$*, then* X *is a digital set of size* $N \times N$*, and* f *is a binary image of size* $N \times N$*.*

In the rest of this paper we apply the notion of the binary image as a binary image of size $2^n \times 2^n$. In fact this restriction for the size of the image can be done without loss of generality, because any image can be enlarged by adding some background pixels, to obtain such size.

Definition 1.3. *Let* X *be a digital set of size* $2^n \times 2^n$ *and* $g : X \longrightarrow [0, 2^n-1] \times [0, 2^n-1]$ *be a bijective function. For any* $p \in X$ *the ordered pair* $g(p) = (x, y)$ *is called the image coordinate of* p*, and* g *is the coordinate function.*

Let $X = [k, N-1+k] \times [h, N-1+h]$ $(k, h \in \mathbf{Z})$ be a digital set. From now on, unless the contrary is stated, we will assume that the coordinate function g is defined in the following way: for $p \in X, p = (p_1, p_2)$, $g(p) = (p_1 - k, N - p_2 + h - 1)$.

Definition 1.4. *Let* X *be a digital set of size* $2^n \times 2^n$*. The binary number* $x_1y_1x_2y_2 \dots x_ny_n$ *is called the quadtree coordinate of* $p \in X$*, where* $x_1x_2 \dots x_n$ *and* $y_1y_2 \dots y_n$ *are representations of* x *and* y *in binary form and* (x, y) *is the image coordinate of* p*.*

Definition 1.4 is a version of the definition of Gray-code (see [1]) modified according to our purposes.

We construct an n-level, fourth-degree ordered tree, whose each node (of course except its leaves) has four successors denoted by the labels: $00, 10, 11, 01$, in this order. Furthermore, we assign an intensity to all leaves. In what follows, we assume that all edges have a label, which is the same as the label of the node, to which that edge leads.

Definition 1.5. *Let X be a digital set of size $2^n \times 2^n$ and f a binary image on X. The complete quadtree representation of f is an n-level, complete, fourth-degree ordered tree. The intensity assigned to a leaf is equal to $f(p)$, where the quadtree coordinate of $p \in X$ is $x_1y_1x_2y_2\dots x_ny_n$ and in the above tree the ith edge of the path from the root to that leaf (i.e. the edge starting at the ith level) has label $x_{i+1}y_{i+1}$.*

If in the complete quadtree the root represents the complete image, then its four successors represent disjoint subimages of size $2^{n-1} \times 2^{n-1}$ of the original image, as it is shown by Figure 1.

00	10
01	11

Figure 1. Relation between the labels of edges of complete quadtree and subimages.

Observe that an element of a complete quadtree at the level i $(0 \le i \le n)$ represents a subimage of size $2^{n-i} \times 2^{n-i}$ of the original image. Especially, when $i = n$, these subimages are the pixels.

As a result of the construction, the quadtree coordinate of a pixel can be specified by concatenating the labels in the path from the root to the required leaf. In the next step we assign coordinates to all nodes of the complete quadtree, in the following way:

Definition 1.6. *Let T be an n-level, fourth-degree ordered tree, which is a complete quadtree representation of a binary image on a digital set of size $2^n \times 2^n$. We assign sequences of labels in the path from the root to all the nodes of the tree. These sequences are called quadtree coordinates of that nodes.*

As mentioned above, the complete quadtree is a representation of the binary image, but our purpose is to consider the reduction of the complete quadtree.

Definition 1.7. *Let S be a set of complete quadtree representations of binary images on a fixed digital set. Furthermore, let R be an operator, which assigns a fourth-degree ordered tree to each elements of S in the following way: if each successor of a node has the same intensity, then R assigns that intensity to the given node and deletes its successors. The tree obtained in this way is the elementary reduced tree of the original tree, and R is an elementary reduction.*

Clearly, the reduction operator in Definition 1.7 can be applied also on any reduced fourth-degree tree, whose edges and nodes have labels as mentioned above.

Definition 1.8. *Let f be a digital image of size $2^n \times 2^n$ and T its complete quadtree. Put $T_1 = R(T)$, $T_2 = R(T_1)$, $\dots$, $T_i = R(T_{i-1})$, where R is the reduction operator given in Definition 1.7. The element T_j of this sequence with $j = \min_i\{i|T_i = R(T_i)\}$, is the quadtree representation of f.*

We can see that in the quadtree those nodes will be leaves, which represent the single-coloured subimages of maximal size of the given image. As a result of the derivation method, the quadtree coordinates in the quadtree are equivalent to the quadtree coordinates in the complete quadtree. Nevertheless, in practice we do not need to construct the complete quadtree and apply the reduction operator, but the quadtree representation can be constructed directly from the image with a simple algorithm.

As an example, consider Figure 2, where the background pixels are denoted by $\circ$ and the foreground pixels by $\bullet$. Lines denote places, where we need to divide the image. We can see, that we need to divide the image until all the subimages become single-coloured.

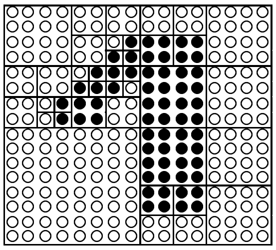

Figure 2. A binary image and its division to construct the quadtree representation.

In Figure 3. the quadtree of Figure 2. is shown. Here $\square$ denotes those nodes of the tree, which are not leaves, $\circ$ and $\bullet$ applied accordingly to Figure 2.

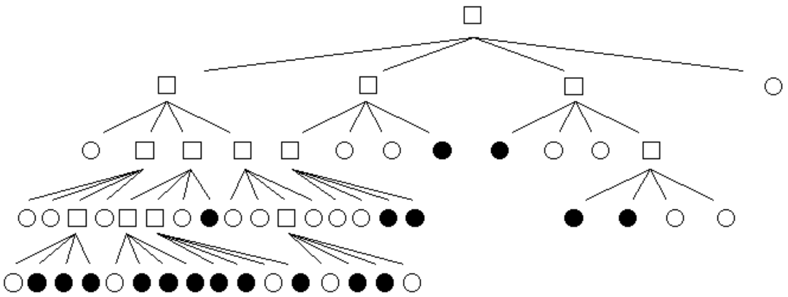

Figure 3. Quadtree representation of the binary image shown in Figure 2.

Consider the quadtree of the same image, showing – in the nodes – the labels that were used to obtain the quadtree coordinates.

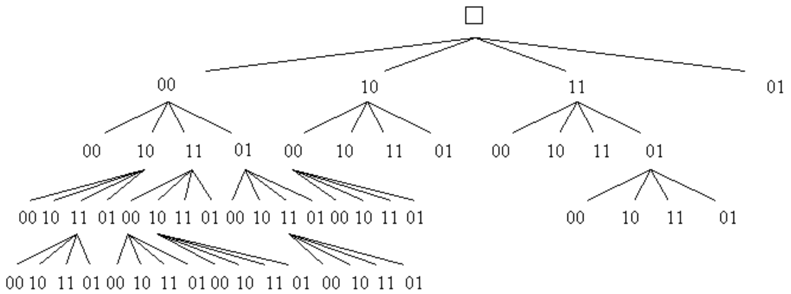

Figure 4. Quadtree representation of the binary image shown in Figure 2. with labels of edges.

2. OPERATIONS FOR SKELETONIZATION

Skeletonizing algorithms usually use two kinds of operations to extract the skeleton. One method is testing, the other is altering the intensity of a given pixel. However, there exist algorithms, in which a third operation is also used to restore a given pixel. It is the reverse operation of altering, so it can be carried out similarly.

2.1. Testing the intensity of a given pixel

In most skeletonizing algorithms we need to delete some pixels and leave other pixels unaltered to extract the skeleton. In many cases the condition of deletion is determined by the local neighbourhood of the pixel considered. Thus we need to know the intensity of pixels in the local neighbourhood of the given pixel to decide whether the pixel satisfies the condition, or not.

It is an easy task in case of pixel-pixel representation, since knowing the coordinate of the pixel, the intensity can be queried. It is not so easy in case of quadtrees. We need to reach the node, which represent the given pixel to test the intensity of that pixel.

In case of complete quadtrees, the coordinate of any pixel can be constructed in the way provided by the next theorem.

Theorem 2.1. *Let the coordinate of the upmost left pixel of an image of size $2^n \times 2^n$ be $(0,0)$. Let the quadtree coordinate of the subimage represented at level l in the complete quadtree be $x_1y_1 \ldots x_ly_l$. If the image coordinate of the upmost left pixel of the given subimage is (x,y), then we have*

$$(x,y) = \sum_{i=1}^{l} \frac{2^n}{2^i}(x_i, y_i). \qquad (1)$$

Proof. The statement can be proved by a simple calculation. □

Applying the above result, the intensity of any pixel can be calculeted using the inverse formula of (1). Indeed, considering (1), it turns out that the quadtree coordinate of the pixel (x,y) is $x_1y_1x_2y_2 \ldots x_ny_n$, where $x_1x_2 \ldots x_n$ and $y_1y_2 \ldots y_n$ are representations of x and y in binary form. Based on this fact, testing the intensity of the pixel can be performed by testing the intensity of the leaf identified by the path, which is determined by the quadtree coordinate constructed in the above way. At level i $(0 \leq i < n)$ of the tree the ordered pair x_{i+1}, y_{i+1} determines an edge, which we need to go along. At level n we need to test the intensity of the node representing the pixel. Namely, the complete quadtree can be used to search in.

As we mentioned above, the quadtree coordinates in the quadtree and the quadtree coordinates in the complete quadtree provide similar information. Thus we also have that the intensity of a subimage can be found in the quadtree based on the quadtree coordinate in the same way as the intensity of a pixel can be found in the complete quadtree; namely the quadtree can be used as a search tree, too. Image coordinates of the pixels obtained in some subimage, can be determined – by Theorem 2.1 – from the quadtree coordinate and size of that subimage. Vice versa, we need to find the smallest subimage, which

contains the given pixel. The intensity of the subimage found in this way is equal to the intensity of the given pixel. (Of course it can happen that we find the pixel itself.) In this case, searching is started with constructing the quadtree coordinate of the pixel, too. Accordingly, we need to search in the tree as deep as we can. The node found at last represents the subimage, which we were searching for.
The next pseudo code is a formal description of the above procedure.

```
Procedure Testing Intensity of The Pixel x1y1x2y2...xnyn
Let the current node be the root
While the code of the current node is not ○ and is not ● Do
Begin
    Consider the next two positions (in the first iteration the first two) of the quadtree
    coordinate of the pixel
    Go along the edge – from the current node – having the label obtained in the
    previous step, and let the current node be the node obtained
End
The intensity of the given pixel is equivalent to the intensity of the current node
```

We can see that the above algorithm performs a search in a search tree.

2.2. Altering the intensity of a given pixel

In fact, deletion of a pixel alters the intensity of the pixel to the intensity of the background, and the restoration of a pixel alters the intensity of the pixel to the intensity of the foreground. In this chapter we describe just the deletion, because the restoration can be performed in an analogous way.

Consider the complete quadtree. The deletion is very simple. Using the result of the previous chapter we search the node representing the pixel, then alter its intensity to the colour of the background.

However, deletion of a pixel is a bit more difficult in the quadtree. Of course, it can be performed by extending the quadtree to a complete quadtree, deleting the pixel and then using the reduction operator we used to obtain the quadtree. However, it is too difficult and – as we shall see – needless.

In the quadtree we can directly delete by applying the inverse of the reduction operator to some nodes (in fact, we extend the quadtree to a complete quadtree with it, too). Using the result of the previous chapter we find the node representing the subimage containing the pixel, which we want to delete. We extend this node – using the reduction operator – then we choose from its successors the node containing the pixel. We repeat this procedure until the given node represents the given pixel itself. Then as mentioned above we alter the intensity of the leaf. It can happen that the given pixel is directly stored in the quadtree (at level n), so altering it can cause that each successors of the ancestor of the given node have the same intensity (it will be the colour of the background). Thus we need to use the reduction operator (which is not a problem). It can happen that we need to repeat this step at a higher level.

The next theorem shows that the above two methods are equivalent:

Theorem 2.2. *The quadtree obtained by the reduction operator, after deleting any pixel from the complete quadtree representing a binary image, is equal to the quadtree obtained in the following way: extend the node representing the subimage containing the pixel, delete the required pixel, and reduce the quadtree.*

Proof. We divide the proof into two cases.

- *The pixel is directly stored in the quadtree (at level n).* In this case searching for the pixel consists of the same steps as searching in the complete quadtree. However, it can happen that each successor of the ancestor of the node has the same intensity (it can only be the background colour), thus using the reduction operator on this node, and repeat this step to its ancestors (if it is necessary). The result is the same as constructing the quadtree after the deletion of the pixel from the complete quadtree. To prove this, it is sufficient to consider when we could use the reduction operator in the quadtree. This case – since the deletion alters just one node – we needed to use it on the same node in the complete quadtree, when we obtained the quadtree.
- *The pixel is contained in a subimage represented by a node at level i ($i < n$) in the quadtree.* In this case while we find the subimage, the search contains the same steps as in the complete quadtree. We need to use the inverse reduction operator in the remaining steps. It is sufficient to apply to those nodes, which contain the given pixel, because the deletion can alter only the path defined by them. In this case for any level it is true that after using the inverse reduction operator on a given node it is the only non-leaf node among the successors of its ancestor, the others are leaves with foreground colour. The next step in the quadtree is the same as in the complete quadtree. When we arrive to level n, we perform the deletion. In this case we cannot use the reduction operator and it is true to this path of the complete quadtree, since at level n there are three nodes of foreground colour and one of background colour. Since we did not extend other nodes along the given path, so reducing the complete quadtree we get this tree. □

Consider Figure 2 and delete pixel $(9,5)$ (the upmost left pixel is $(0,0)$). The next figure shows the quadtree of the image obtained after deleting.

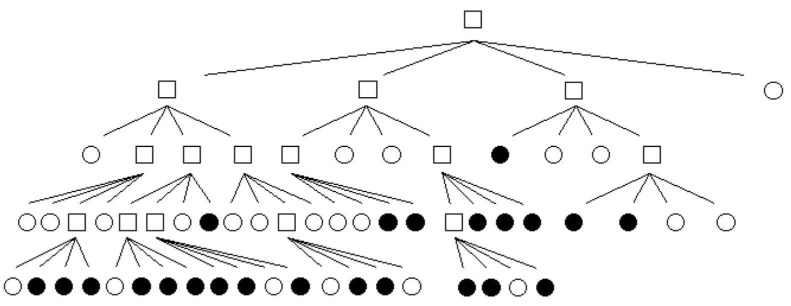

Figure 5. Quadtree of the binary image shown in Figure 2. after deletion of pixel (9,5).

The next pseudo-code performs the above described procedure.

```
Procedure Deletion of Pixel x1y1x2y2...xnyn
Find the node containing the given pixel (using the result of Chapter 2.1), let the
current node be this node
While the length of the quadtree coordinate of the current node is not equal to the
length of the quadtree coordinate of the given pixel do
Begin
    Replace the code of the current node with □
    Create the successors of the current node with code •
    Let the new current node be the successor of the previous one containing the
    given pixel
End
Replace the code of the current node with ◦
Reduce the tree using the reduction operator
```

In fact the third, fourth and fifth steps perform the work of the inverse reduction operator, and the seventh step is the reduction itself.

3. CONCLUSIONS

Most image processing methods are not implemented to most image representations. It has the disadvantage that performing these methods we need an additional operation, which converts the current representation to a better usable one. The size of storing memory can be increased with this step.

In this paper we investigated the direct applicability of the skeletonizing algorithm on the quadtree representation. Our results can be used in case of image processing methods, in which we need the next elementary operations: testing intensity and deletion of a given pixel.

We made a program, which implements the above theoretical results into practice by an algorithm from [7].

Our results described in this paper are applicable with some modification for the octtree representation of 3-dimensional images. These ideas can be generalized to multi-level images, too.

REFERENCES

1. L. A. BREENE Quadtrees and Hypercubes: Grid Embedding Strategies Based on Spatial Data Structure Addressing. *The Computer Journal*, **36** (1993), 562–569.

2. C. ARCELLI Pattern Thinning by Contour Tracing. *Computer Graphics and Image Processing*, **17** (1981), 130–144.

3. A. FAZEKAS, G.T. HERMAN AND S. MATEJ On Processing Binary Pictures Via Their Projections. *The Internation Journal of Imaging Systems & Technology*, **9** (1998), 99–100.

4. L. LAM, S.-W. LEE AND C. Y. SUEN Thinning Methodologies – A Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **14** (1992), 869–882.

5. J. PIPER Interval Skeletons. *11th IAPR*, **III** (1992), 468–471.

6. J. SERRA Introduction to Mathematical Morphology. *Computer Vision, Graphics, and Image Processing*, **35** (1986), 283–305.

7. E. S. DEUTSCH Thinning Algorithms on Rectangular, Hexagonal, and Triangular Arrays. *Communications of the ACM*, **15** (1972), 827–836.

Received November 1, 1999.