반도체및디스플레이장비학회지 제8권 제1호(2009년 3월)
Journal of the Semiconductor & Display Equipment Technology, Vol. 8, No. 1. March 2009.

A study on N-dimensional quad-tree decomposition

Cheon-Hee Yi† and Jae-Young Yi

†Dept of Electronic Engineering, Chong-ju University

ABSTRACT

We have examined the problem of the number of quad-tree blocks that an n-dimensional rectangle will be decomposed into on the average. the contribution of this paper are both practical and theoretical. In this paper, we develops the overlapping multi-scale models and the region quad-tree models which is useful in computer graphics animation, image processing, pattern recognition and also for modeling three dimensional objects. These models, which represent something of a conceptual departure from other models developed for multi-scale framework were developed with the specific interest of producing smooth estimates.

Key Words : Multi-scale models, quad-tree, overlapping-tree, framework

1. Introduction

We develops the overlapping multi-scale models and the region quad-tree models which is useful in computer graphics animation, image processing, pattern recognition and also for modeling three dimensional objects.

These models, which represent something of a conceptual departure from other models developed for Multi-scale framework were developed with the specific interest of producing smooth estimates.

For the estimation Problems of interest in this paper specifically large estimation problems in which estimation error statistics are required, the straight forward application of the estimation methods[1] fails to be practical for computational reasons.

Most statistical estimation problems are characterized by an explicit prior statistical model, parameterized in terms of a number of random variables with specified probability distributions.

It should be emphasized that a great variety of tree structures are possible; the structure shown in the figure is just on possibility, although a rather common and convenient one for representing two dimensional processes.

This framework possesses further advantages : the production of estimates and estimation error variances on a hierarchy of scales facilitating resolution accuracy trade-offs leading to the direct extraction of estimation of coarser scale features, and the fusion of data of differing resolution with no change in algorithmic structure.

2. Analysis of the n-dimensional quad-tree decomposition.

The quad-tree[2] is a hierarchical data structure which has proven useful in many domains. The quad-tree decomposes a data set that does not meet some criterion into four quadrants; each quadrant is then examined in turn to see if it meets the criterion, with further subdivisions occurring as necessary.

Fig. 1 illustrates the region quad-tree, which is used represent region data. Here the criterion is simply to split a work into four equal quadrants wherever it is not homogeneous.

For quad-tree representations of typical images, a shift may change the positions and sizes of the blocks, however, the total number of nodes normally remains about the same[3]. This is due to the fact that the fragments of a block decomposed will merge with the

†E-mail : yicheon@cju.ac.kr

fragments created by a neighboring block of the same color. Nonetheless, it is worthwhile to examine the number of fragments produced by shifting an individual block to understand the amount of work that goes on behind the scenes during shifting operations.

Each node in a record containing seven fields. The first five fields contain pointers to the node is father and its four sons, labeled NW, NE, SE, and SW. They are also termed its boundaries and at times we speak of them as if they are directions.

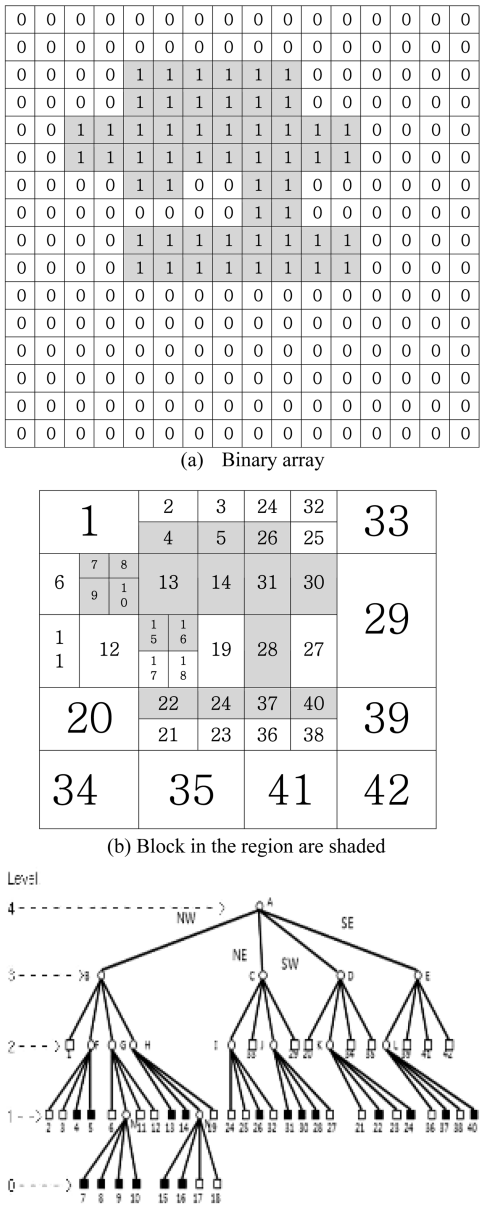


Fig. 1. A region, its binary array, Its maximal blocks, and the corresponding quad-tree.

In this section, we define a closed-form expression for the average number of n-dimensional quad-tree nodes required by an n-dimensional hyper - rectangle aligned with the axes. Our formula include is special cases the formula of the efforts for 2-dimensional spaces. It also agrees with theoretical and empirical results that the number of blocks depends on the hyper-surface of the hyper-rectangle and not on its hyper-volume. The practical use of the derived formula is that it allows the estimation of the space requirements of the n-dimensional quad-tree decomposition.

Quad-trees are used extensively in 2-dimensional spaces in graphics, robotics and 3-dimensional medical images.[4] Our formula permits the estimation of the space requirements for the data hyper-rectangles when stored in an index structure like a quad-tree, as well as the estimation of the search time for query hyper-rectangles.

Hierarchical decomposition of space plays an important role in every application that involves geometric data. The idea is that the space is decomposed recursively into smaller and smaller pieces, until the content of each such piece is homogeneous. The problem solved in this paper is the analytical estimation of the number of pieces that an n-dimensional rectangle(hyper-rectangular region) is decomposed into.

Consider a 2-dimensional image represented as a array of squares. Each such square is called a pixel. The length of the side of the image is called the granularity of the image. The quad-tree is an approach to image representation based on a successive sub-division of the array into quadrants.

This process is represented by tree of out-degree 4 in which the root node represents the entire array the four sons of the root node represent the quadrants, and the terminal nodes correspond to those blocks of the array for which no further subdivision in necessary.

We focus on representing one object only. An object with in an image is decomposed into blocks. This hierarchical decomposition approach has been

used in several areas as follows.

1) In graphics and robotics (3-dimensional space)[2].

2) In geographic information systems and spatial databases.

3) In traditional databases, where records with n attributes correspond to points in an n-dimensional space. Many methods have been suggested to store such a collection of data, utilizing the hierarchical decomposition approach.

4) In spatio-temporal and scientific databases, where time introduces one more axis[5].

5) In images databases, e.g[6], where 3-dimensional brain scans have to be stored. Regions in these brain scans can be encoded using oct-trees, to save space and to achieve faster response on range queries.

It generalizes the observation that the number of quad-tree blocks in proportional to the perimeter of the polygon. Our formula shows that, for 2-dimensional rectangles, the number of quad-tree blocks in approximately the perimeter of the rectangle, while for higher dimensionalities n>>, it is roughly half of the hyper-surface.

From the practical point of view, the number of quad-tree blocks of a decomposition is important, because it determines the number of nodes that a main-memory-based quad-tree will require; the number of entries in a linear quad-tree that will be required; also, the number of pieces that a range query will be decomposed into.

From the theoretical point of view, it proposes a methodology which we believe will be useful in the analysis of other quad-tree-related methods. The methodology consists of two steps:

Step 1 : solve the problem for the 'magic' rectangle(which is easy)

Step 2 : show that the formula for an arbitrary rectangle can be derived by linear interpolation from suitable 'magic' rectangle.

Problem dealing with rectangles in the plane are very important in the field of computational geometry[7], in particular with applications to VLSI layout[8].

3. Parallel construction of quad-trees

We give parallel algorithms for mesh generation. The finite element method is often performed on parallel computers, but parallel mesh generation is less common. In some applications, a single mesh is generated and used many times; in this case the time for mesh construction is not critical and a relatively slow, sequential algorithm would suffice. In other applications is not critical and a relatively slow, sequential algorithm would suffice. In other applications, especially when the physics or geometry of the problem changes with time, a mesh is used once and then discarded or modified. Then a parallel mesh generator would offer considerable speed-up over a sequential generator.

We parallelize the quad-tree-based methods[9]. The gird-based method, and a grid-based modification algorithm[10], both parallelize easily, but as mentioned above these may produce too many triangles. It is currently unknown whether Ruppert's method[11] has an efficient parallel version.

A quad-tree is a recursive partition of a region of the plane into axis-aligned squares. One square, the root, covers the entire region. A square can be divided into four child squares, by splitting it with horizontal and vertical line segments through its center. The collection of squares then forms a tree, with smaller squares at lower levels of the tree.

It may seem that quad-tree are easy to construct in parallel, a layer a time. In practice this idea may work well, but it does not provide an asymptotically efficient algorithm, since the quad-tree may have depth proportional to its total size. Instead our algorithm follows the following strategy. We first find a "framework", a tree of quad-tree squares such that every internal node has two nonempty children. This framework then guides the computation of the complete quad-tree. We then balance the quad-tree so that no square is adjacent to a square much larger or smaller than itself. Finally, we perform local "warping" as in[12], to construct a guaranteed-quality triangulation.

4. Illustration of an overlapping-tree

We discard the standard assumption that distinct nodes at a given level of our tree correspond to

disjoint portion of the image domain. Instead we construct models in which distinct tree nodes correspond to overlapping portions of the image domain. As a consequence of this idea, a given physical image pixel may now correspond to several node at the finest scale.

In this way, we remove the hard boundaries between image-domain pixels because now multiple tree nodes may contribute to each of these pixels, this reducing the effective tree distance between the two sets of nodes corresponding to these pixels and spreading the correlation that must be captured among a set of nodes. For obvious reasons, we refer to these Multi-scale models as overlapped-tree models.

Because our overlapped-tree approach represents something of a conceptual departure from other approaches, we make the ideas. more concrete through the use of a simple example.

However, for most other types of fields, the generation of sample paths is computationally quite complex. For example, sample paths may be simulated using the following three procedure :

1) compute the square root of the covariance matrix,

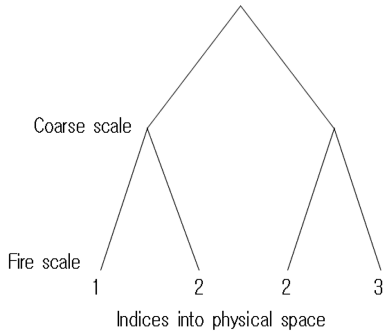
2) generate a vector of unit variance uncorrelated random variable,

3) compute the sample path as.

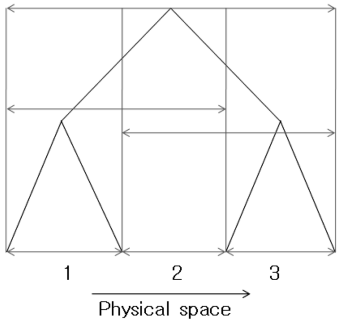
Although conceptually straight forward, there is a considerable challenge in computing the matrix square root, requiring in general 0()(overlapping tree parameterization vector) calculations for a random field of K points.

Let us index our Multi-scale model on a dyadic tree that has four finest-scale nodes, thereby providing only a minimal amount of redundancy; Fig. 2 displays an example of such a tree. On the right side of the figure, we depict the tree with on indication of the subsets of real, physical points (i.e, subsets of {1,2,3}) to which each node corresponds. Thus the top node correspond to all three points (i.e,{1,2,3}) and two nodes at the second level correspond to {1,2} and {2,3} respectively[13].

At the bottom level there is a single node



(a) Mapping of physical space onto the finest tree scale.



(b) Aggregate representations of tree node in Physical space.

Fig. 2. Illustration of an overlapping - tree representation of a process of length three showing both the dyadic tree.

corresponding to process elements . and another , but there are two nodes corresponding to : in the lifted domain on the tree, process element is lifted to have two finest-scale tree nodes. Thus if order the four fine-scale nodes from left to right (as shown in the left half of Fig. 2), we are led to the following choice of : which implies that

$$P_l \equiv G_x P G_x^T = \begin{bmatrix} 1 & 0.5 & 0.5 & 0 \\ 0.5 & 1 & 1 & 0.5 \\ 0.5 & 1 & 1 & 0.5 \\ 0 & 0.5 & 0.5 & 1 \end{bmatrix}$$

This example illustrates the basic constraints that we place on any lifting matrix :

a) It consists entirely of zeros and ones.

b) Each column has at least on nonzero entry.

c) Each row has exactly one nonzero entry.

These conditions ensure the following basic properties :

a) Every position in the original domain corresponds to at least one position in the overlapped domain.

b) Every position in the overlapped domain corresponds to exactly one position in the original domain.

Thus the lifting process in local and is in fact trivial to compute once has been specified. The specification of in typically carried out implicitly in terms of the overlapping structure of the tree : any given overlap structure uniquely specifies a corresponding . We can certainly imagine more general lifting schemes, allowing for example for fine-scale lifted nodes that we associated with more than on real data pointer ; however, we will find our restricted lighting scheme to be sufficient for our purposes[14].

5. Analysis of the overlapping framework.

In this section we describe an implicit, compact, and efficient method for specifying the operators, and directly from the overlapping structure that is chosen in order to achieve our desired objective of producing random fields and estimates with some desired level of smoothness[12]. The intent of this section is to provide notion of the means of specifying these operators ; parallels the subject matter of this section, but in greater generality and detail.

As we have already discussed and will see in our examples, as decreases (i.e., as the amount of overlap increases) the resulting simulated or estimated fields (i.e., those fields in the original domain after projection by) become smoother, but the computational complexity of the resulting simulation and estimation algorithms increases. For simplicity in exposition we focus on the most important case, namely the one in which the tree and overlap structure are regular. Specifically, we focus here on regular overlapping quad trees representing 1-D random fields, where each node above the finest scale possesses q descendants. The overlap structure in presumed to be spatially stationary, that is, for any two nodes and on the same scale of the tree, the manner in which their descendants overlap is same. Our focus on the one-dimensional case is strictly for reasons of clarity. We will make explicit two aspects of the overlapping framework which details how an overlapping tree structure O may be selected, and which details how the projection operators may be determined from O.

Throughout this discussion we will make the assumption that the overlapping tree is regular, the tree lies in d-dimensional space, has M scales, and each node above the finest scale has offspring. An example of a two-dimensional regular tree is shown is Fig. 3.

As it is suggested pictorially by Fig. 5, each of the nodes on the Multi-scale tree is associated with a bounded d-dimensional case in the original domain. Now consider the notation described in Fig. 4[15].

The regularity of the tree implies that the uses associated with the Multi-scale nodes on the same scale all have the same shape and size ; for those uses associated with nodes on scale m, we denote by the length of those cube edges parallel to dimension .

Next, the cubes associated with the children of a common parent may overlap. Consider two Multi-scale nodes on scale m which have a common parent, and which are neighbors along dimension , then the amount of overlap, measured along dimension, between the two cubes associated with these two nodes is denoted by[15].

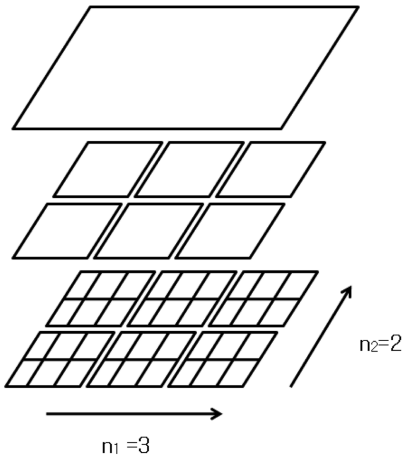


Fig. 3. A simple example of a two-dimensional regular tree.

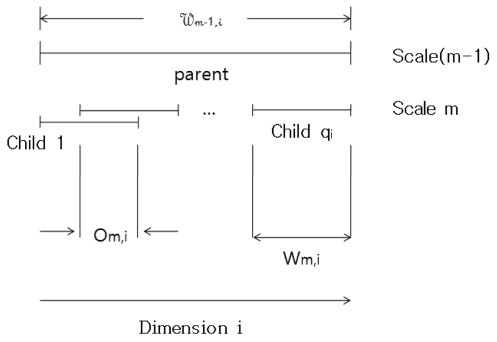


Fig. 4. The basic overlap notation required for the purpose of this section.

6. Conclusion

We have examined the problem of the number of quad-tree blocks that an n-dimensional rectangle will be decomposed into on the average. the contribution of this paper are both practical and theoretical . From the practical point of view, the number of quad-tree blocks of a decomposition is important, because it determines the number of nodes that a main-memory-based quad-tree will require; the number of entries in a linear quad-tree that will be required; also, the number of pieces that a range query will be decomposed into which will be proportional to the response time for this query.

A significant contribution of this paper is the development of a novel Multi-scale structure known as an overlapping tree, which is able produce smooth estimates, even with sparse measurements. This approaches not a specific Multi-scale model, rather it is a general technique which may be applied to a broad variety of Multi-scale models. We derive theoretical conditions for applicability of the overlapping tree technique, and demonstrate its use in texture estimation. In this paper, we develops the overlapping Multi-scale models and the region quad-tree models which is useful in computer graphics animation, image processing, pattern recognition and also for modeling three dimensional objects. These models, which represent something of a conceptual departure from other models developed for Multi-scale framework were developed with the specific interest of producing smooth estimates.

References

1. R. Daley, Atmosphic data analysis, Cambridge university press, Newyork, 1991.
2. H. Samet, The Design and analysis of spatial data structures. Addison-Wesley, 1990.
3. H. Samet, The Quad-tree and related hierachical data structures, ACM Computing Surveys 16, 2, 1984.
4. C.A. Shaffer and H.Samet, optimal quad-tree construction algorithm and image processing 37, p402-419, March 1987.
5. Curtis P. Kulovson and Michael Stonebraker. Segment indexes : Dynamic indexing techniques for multi-dimensional interval data. Proc. ACM SIGMOD, p.138-147, May 1991.
6. Manish Arya, William Cody, christos Faloutsos, Joel Rechardson, and Arthur Toga. Qbism : Extending a dbms to support 3d medical images, Tenth Int. Conf. On Data Engineering, p.314-325, Feb. 1994.
7. F. P. Preparata and M. I. Shamos, Computational Geometry. New york : Springer-verlag, 1985.
8. S. K. Nandy, I. V. Ramakrishna, “Dual quad-tree representation for VLSI design,” 23rd design Automation Conference, p. 633-666, 1986.
9. M. Bern and D. Eppstein. Mesh generation and optimal triangulation. In Euclidean Geometry and the computer, eds. D. Z. Du and F. Hwang, World Scientic, 1992.
10. L. P. chew. Guaranted-quality triangular meshes. Tech. Rep. TR-89-983, Cornell university, 1989.
11. J. Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation 4th Symp. Discrete Algorithms p.83-92, 1993.
12. E. A. Melissaratos and D.L Souvaine : Coping with inconsistencies : A new approach to produce quality triangulations of polygonal domains with holes. 8th Symp. Comput. Geom. p.202-211, 1992.
13. F.T, Leigthon, M. Lepley, G.L. Miller, “Layouts for the shuffle-exchange graph based on the complex plane diagram.” SIAM J. A.C. G. DISC. Math. Vol.5, No.2, 1984.
14. William R. Crum, Oscar Camara, and Derek L. G. Hill, “Generalized overlap measures for evaluation and validation in medical image analysis,” IEEE Trans, on Medical Imaging, Vol. 25, No. 11, 2006.
15. M. Luettgen , A. S. willsky, “Likelihood calculation for a class of Multi-scale stochastic models, with application to texture discrimination.”, IEEE Trans. Image processing(4)#2, p.194-207, 1995.

접수일: 2009년 3월 3일, 심사일: 2009년 3월 11일
게재확정일: 2009년 3월 13일