Fixed-size geometric covering to minimize the number of disconnected components
by Andrew Joseph Tomascak

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science
Montana State University

Abstract:
Decomposing an image into fixed-sized sub-images has many applications in image storage and analysis, especially in the area of three dimensional imaging. The Connected Component Problem deals with the task of covering a binary image with tiles such that the sum of the number of connected components of each tile is minimized. This problem is relevant for storing and processing three dimensional images, in particular those using voxels. Interest in this study was originally spawned through the study of biological voxel images, and in particular, neural morphology. In this thesis, the Connected Component Problem is studied in both of its basic forms, allowing overlapping and the more restricted version where overlap is not allowed. The names of the two variations of the problem are the Connected Component Covering and the Connected Component Tiling. In the non-overlapping version, the Connected Component Tiling, a variation of this problem is proven to be NP-complete. An approximation factor is proven for both instances. Approximation algorithms are given for both cases along with upper bounds for each. Finally, real world results are given from actual implementations to show the effectiveness of the approximation algorithms.

FIXED-SIZE GEOMETRIC COVERING TO MINIMIZE THE NUMBER OF DISCONNECTED COMPONENTS

by

Andrew Joseph Tomascak

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

June 2003

APPROVAL

of a thesis submitted by

Andrew Joseph Tomascak

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographical style, and consistency and is ready for submission to the College of Graduate Studies.

Binhai Zhu, Ph.D. _____ June 24, 03
(Signature) Date

Approved for the Department of Computer Science

Michael Oudshoorn, Ph. D. Michael Oudshoorn _____ June 25, 2003
(Signature) Date

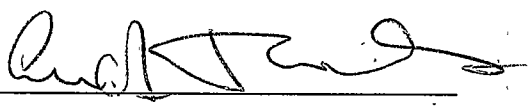
Approved for the College of Graduate Studies

Bruce McLeod, Ph. D. Bruce L. McLeod _____ 6-30-03
(Signature) Date

STATEMENT OF PERMISSION OF USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

Signature 

Date 25 June 2003

TABLE OF CONTENTS:

LIST OF FIGURES:

LIST OF TABLES:

Table Page

ABSTRACT

Decomposing an image into fixed-sized sub-images has many applications in image storage and analysis, especially in the area of three dimensional imaging. The Connected Component Problem deals with the task of covering a binary image with tiles such that the sum of the number of connected components of each tile is minimized. This problem is relevant for storing and processing three dimensional images, in particular those using voxels. Interest in this study was originally spawned through the study of biological voxel images, and in particular, neural morphology. In this thesis, the Connected Component Problem is studied in both of its basic forms, allowing overlapping and the more restricted version where overlap is not allowed. The names of the two variations of the problem are the Connected Component Covering and the Connected Component Tiling. In the non-overlapping version, the Connected Component Tiling, a variation of this problem is proven to be NP-complete. An approximation factor is proven for both instances. Approximation algorithms are given for both cases along with upper bounds for each. Finally, real world results are given from actual implementations to show the effectiveness of the approximation algorithms.

INTRODUCTION

Background

In the study of three dimensional images, it is common to store such images in the form of voxels. Voxels, or volumetric pixels, are a convenient way to store data in three dimensions. Voxel images are primarily used in the field of medicine and are applied to X-Rays, CAT (Computed Axial Tomography) Scans, and MRIs (Magnetic Resonance Imaging) so professionals can obtain accurate 3D models of the human body. They are also used for various other applications, from geological surveying to gaming algorithms that make 3D acceleration unnecessary [MMG99, Na00].

Storing three dimensional images in the form of voxels is extremely memory intensive. Voxelization is the process of adding depth to an image using a set of cross-sectional images known as a volumetric dataset [ACY93]. These cross-sectional images (or slices) are made up of pixels. The space between any two pixels in one slice is referred to as interpixel distance, which represents a real-world distance. The distance between any two slices is referred to as interslice distance, which represents a real-world depth. The dataset is processed when slices are stacked in computer memory based on interpixel and interslice distances to accurately reflect the real-world sampled volume.

The motivation behind this study is rooted in computational biology. In the study of neural biology, neurons are photographed in microscopes. This is done by staining a neuron so that it can be seen with the visible eye, and viewing the stained neuron under a

confocal microscope. The microscope is then given a precise focus and a picture is taken. The focus is then increased by a precise amount to slightly different depth of focus, and another picture is taken. This process repeats until the neuron has been fully studied. The end result is a "stack" of images covering all depths, allowing the images to be combined into a single three dimensional image, a voxel image.

In this thesis, we will discuss both the Connected Component Covering and the Connected Component Tiling problems and their different restrictions. The goals remain the same for both of these problems, and we shall refer to the general version of these problems as the Connected Component problem. A fast Connected Component algorithm could speed up storage and access times of voxel objects by allowing the compression of the cross-sectional images and optimized data access. This is the primary motivation behind studying the Connected Component problem. Such an image may be too large to work with in its entirety but smaller sub-images may be perfectly acceptable to work with. Subdividing the cross-sectional images into just relevant image data (storing only the parts of the image containing information) will decrease the storage requirements and therefore speed up access and operations on this data. We can further increase this speed up by making sure that the data is grouped together in an intelligent manner. For instance, it is more desirable to have an image object contained in a single subdivision than split across two or more subdivisions. The more subdivisions that are used to store an object, the more memory must be accessed to do operations on this object.

This study of the Connected Component problem makes all of these optimizations possible. Using a good Connected Component algorithm, one can subdivide a three dimensional voxel image into smaller images that have a minimum number of connected components. This, in turn, allows faster processing and analysis to be done. The Connected Component problem is the task of making these subdivisions (equal in size and shape) such that each object is stored in as few subdivisions as possible. In the case of the neuron imaging, it is much easier to handle images with small numbers of connected components. Generally, a researcher is trying to study a single feature of the neuron, and additional components in the image only complicate the automation processes used for study.

Definition of the Connected Component Problem

To properly define this problem in more formal terms, some definitions must be given first. The input to this problem consists of two components: an image and a tile size. The input ***image*** component is a two dimensional $n \times m$ array, where each entry corresponds to a pixel value. For the purposes of this paper, all images are considered to be binary in nature, meaning each pixel is either "on" or "off" (a pixel is ***on*** if the array location is drawn black and is ***off*** if the array location is drawn white). In actual use on a computer, 1's and 0's are used in lieu of black and white in the array. A ***tile*** is a $j \times k$ subset of the array or image, and all pixels within this tile are said to be "covered". We will denote a single covering tile as $T = \{p_1, \ldots, p_u\}$ where p_i is each *on* pixel within the

tile. Let $|T| = u$ where u is the number of *on* pixels in T and $ccn(T)$ be the number of connected components within T. Also, for simplicity, all images and tiles will be considered square $n \times n$ and $k \times k$ arrays respectively, but all results from this paper can be extended to $n \times m$ arrays or images and $j \times k$ sized tiles. A ***covering*** of an image is a set of tiles such that all on pixels are a member of at least one tile in the set, denoted here as $C = \{T_1, T_2, \ldots, T_v\}$. Let $|C| = v$, where v is the number of tiles in covering C. A ***tiling*** of an image is the same as covering, with the further restriction that no two (or more) tiles cover the same pixel. The number of connected components in each individual tile is then calculated and added to the sum of all the other tiles, producing a ***score*** for that particular tiling, given in Equation 1.1.

$$score(C) = \sum_{i=1}^{m} ccn(T_i) \text{ where } ccn(T_i) \text{ is the number of connected components in covering tile } T_i. \tag{1.1}$$

Given these definitions, one possible goal could be to find the covering that uses the minimum number of covering tiles, studied in [FPT81]. We will refer to this problem as the Minimum Tile Covering Problem and its optimal solution value as $opt_{\#}$. This is a packing problem directly related to the Set Cover problem and is discussed more in Section 3. The goal of the Connected Component problem is to minimize both the number of covering tiles and the total number of connected components (or score) within each covering or tiling. We shall call the optimum solution value to the second part

$opt_{\#CC}$. Therefore, an optimal answer to the Connected Component problem is found by first determining $opt_{\#CC}$ and then finding $opt_{\#}$ for that score,

$$opt_{\#CC} = \min\left(score(C)\right). \tag{1.2}$$

Connectivity, for this paper, is 4-neighbor connectivity, or *on* pixels that are 4-adjacent [GW02]. In Figure 1.1, there are three examples of connectivity that explain this more clearly. In all three examples, the grey pixel is the current pixel we are looking at. Neighboring *on* pixels are shown in black. The example on the left has four neighbors, all of them sharing an edge with the current pixel. These are the four spots that make up 4-adjacency. These pixels are all considered connected by 4-neighbor connectivity, which is simply the collection of all pixels that are 4-adjacent to each other. In the center example, all eight neighboring pixels are black, and these positions make up all the 8-adjacent positions. This is also known as 8-neighbor connectivity, and all pixels that share and edge or a corner are connected by 8-neighbor connectivity. In the right example, a pixel is shown that is 8-neighbor connected, but not 4-neighbor connected. We also consider 8-adjacency (8-neighbor connectivity) briefly, but it does little to change the basic problem.

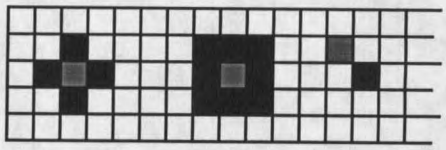


Figure 1.1 Examples of adjacency and connectivity.

The Connected Component problem is defined as the following: given an n by n array of pixels, with the pixels being on or off, find a tiling C of k by k tiles such that the total number of connected components of all tiles is minimized and the total number of tiles is also minimized.

In Figure 1.2, there are two instances of the same Connected Component problem, with two different solutions. In these two examples, the covering tiles are all 5 by 5 pixels and the array itself is 15 by 15 pixels. Both of these represent possible correct solutions, as all *on* pixels are indeed covered by at least one tile. The total size of the cover is $|C| = 2$ for the left tiling and $|C| = 3$ for the right tiling and the scores of both of these covers are $score(C) = 3$. There are only three components total in these instances, so this is the lowest score possible and therefore is $opt_{\#CC}$. The left tiling also computes $opt_{\#}$, and therefore is the optimal choice for this Connected Component problem.

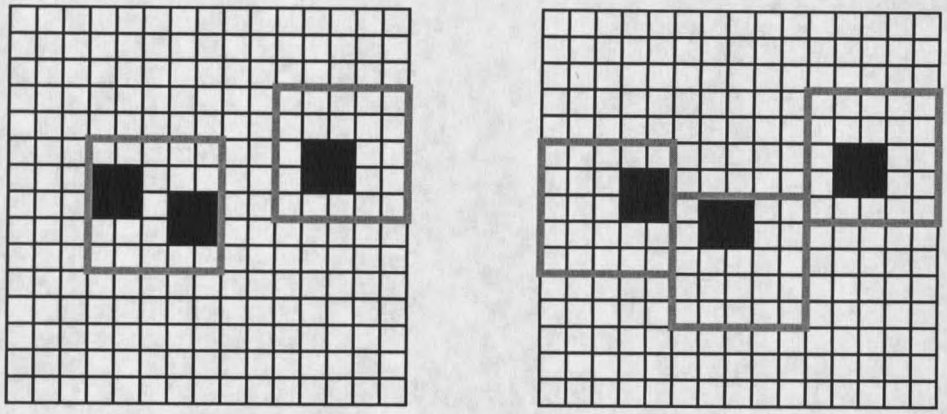


Figure 1.2 Two instances of the general Connected Component Problem.

Given a set of points in two dimensions (2D), covering them using the minimal number of fixed sized boxes (squares) was proven to be NP-complete more than two decades ago [FPT81]. Recently, Khanna, *et al.* studied a slightly different problem; namely, partition an $n \times n$ array of non-negative numbers into the minimum number of tiles (fixed-sized rectangles) such that the maximum weight of any tile is minimized. (Here the weight of a rectangle is the sum of the elements contained in it.) They proved that this problem is also NP-complete and that it can not be approximated closer than a factor of 1.25 in polynomial time. They also proposed a factor-2.5 approximation algorithm [KMP98]. We prove in this thesis that the Connected Component Tiling problem, in which overlapping is not allowed, is also NP-complete, reducing it from the 3SAT problem. Furthermore, we propose a factor-k^2 approximation algorithm.

We consider two variations of this problem. Variation one is the case where overlapping of covering tiles is allowed and will be referred to as the Connected Component Covering problem. The second case is where overlapping is not allowed, and we will refer to it as the Connected Component Tiling problem. This would correspond to real-world situations where redundancy of information is not desired. Examples would include image data that would be loaded twice in the course of processing. Because the second variation is simply a further constrained version of the first, any legal covering of the second type will also be a legal covering of the first given any particular image. A simple upper bound can easily be found for both types simply by finding an upper bound to the second case. The largest number of disconnected components that can be in a single tile, given 4-neighbor connectivity, is $\left\lceil \frac{k^2}{2} \right\rceil$ (in a checkerboard pattern) and given 8-neighbor connectivity is $\left\lceil \frac{k}{2} \right\rceil^2$. We will use $\frac{k^2}{2}$ from now on for 4-neighbor connectivity and $\frac{k^2}{4}$ for 8-neighbor connectivity as it has minimal effect on the results, and simplifies the reading. The largest number of tiles needed to cover an n by n array is $\left(\frac{n}{k}\right)^2$, again ignoring the ceiling function. Therefore the highest score a minimum tiling could achieve would be $\frac{n^2}{k^2} \cdot \frac{k^2}{2} = \frac{n^2}{2}$ for 4-neighbor and $\frac{n^2}{4}$ for 8-neighbor. This makes sense because that would also be the maximum number of disconnected components within an n by n array. This is our initial upper bound to begin with. The

obvious initial lower bound for either version of the problem is to simply use the actual number of connected components within the image or array, as this will always be the minimum that a covering can score.

CONNECTED COMPONENT COVERING

In the case where overlapping of boxes is allowed, we allow covering tiles to overlap. Two or more tiles are allowed to cover the same pixel or set of pixels, but these pixels still count toward the score of each tile covering them. This would correspond to a real world problem where we are concerned with reducing the total number of components to a minimum. This would be the case if the per object cost of an operation is of more concern than the storage and redundancy of extra data. This instance of the Connected Component problem can be reduced to the Set Cover Problem [CLRS01] and gives us an approximation bound. We shall refer to this instance as the Connected Component Covering, CCC.

Set Cover Problem

In the case where overlapping is allowed, the Connected Component Covering highly resembles the Set Cover problem. The Set Cover problem is defined as follows: Given a set S of n elements, $S = \{x_1, x_2, \ldots, x_n\}$, and a collection C of subsets of S, find the minimum number of subsets such that every element in S is in at least one subset. In Figure 2.1, we can see a visual example of the Set Cover problem. In this example we can see a master set of elements, represented by dots, and the subsets which are smaller collections of the elements. To solve this problem, we would look for a collection C of

subsets, such that all elements are included in at least one set of the collection. It can be seen here that some subsets *must* be included, because some elements are only included in one subset.

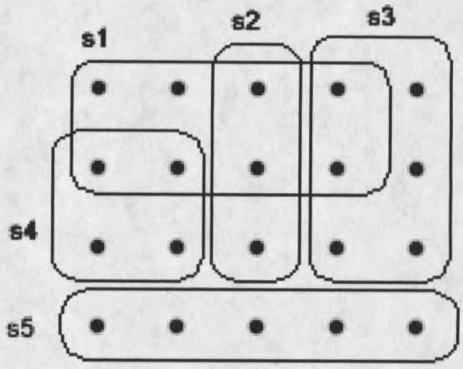


Figure 2.1 An example of the Set Cover problem.

Johnson proved that the Set Cover problem is approximable within a factor of $1+\ln|S|$, using a simple greedy algorithm, in which the subset with the largest number of non-included elements is picked repeatedly until all elements are included [Jo74]. The idea behind the proof of the GREEDY-SET-COVER algorithm is to give all elements a cost, then to total these costs to give a maximum cost to any one subset. Basically, the proof shows that the optimal answer may not be larger than the harmonic of the largest set. The harmonic of the set is $H_n = 1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{n}=\sum_{k=1}^{n}\frac{1}{k}\le \ln n+1$, where n is size of the set. In Figure 2.1, the greedy algorithm would pick s1, s5, s3, s4, and s2, in that order. At this point the algorithm would end, as all elements are now covered. In this particular example all subsets were used, however this is not always the case.

In the study of the CCC problem, each possible covering tile can be seen as a set of pixels with a score given by the number of connected components within the tile. Reducing the problem as such, we can now use known algorithms for the Set Covering problem to solve the Connected Component Covering problem. Both [Ch79] and [Jo74] have proven an upper bound to the Set Covering problem using a simple greedy heuristic running in $O(n \log n)$ time. The algorithm simply picks a covering box that covers up the most uncovered pixels that will still be connected within a single box. When no more covering boxes can be found that meet these criteria, and there are still uncovered pixels remaining, the connective limit is raised by one. Now the algorithm picks the box covering the most pixels that will still be only be covering two unconnected components. And so forth, until all remaining pixels are covered. First established in [Jo74] and [Lo75], an ever better upper bound was proven (using the same algorithm) later in [Ch79]. When this bound is applied to the Connected Component Covering, this basic greedy algorithm is guaranteed to return an answer within a factor of $1 + 2 \cdot \ln(k)$. In some special cases, an approximation algorithm is not needed because the optimal solution can be found in polynomial time itself. We will look at such a case next.

Special Case

If the height of the bounding cover is the same as or greater than the height of the overall array $(j \geq m)$, then it is basically a one dimensional problem, in the sense that the