A Novel Algorithm to Generate the Cover Map of a Remotely Sensed Image for GIS Applications

Arnab Chakrabarti, Abha Jain and A.K.Ray
Department of Electronics & Electrical Communication Engineering
Indian Institute of Technology, Kharagpur-721302, INDIA

Abstract: *In this paper, we present a procedure to approximate the cover map generated from a remotely sensed image. The procedure can detect the nodes and branches from a segmented description of an image resembling a network. In the first of a two part algorithm, the nodes and branches of the mesh are detected. Once this has been completed, all those branches encapsulating a segment are used to generate the cover map. Dominant-point detection algorithms have been used to detect the breakpoints on the individual branches. This allows the entire scene to be represented by a set of polygons.*

1: Introduction

The cover map of a remotely sensed image may be generated once the image is preprocessed and segmented and an edge linked description of each segment in the scene is available. The mesh-like pattern, that appears after edge detection and edge linking operations are performed on such an image, is subsequently processed to extract the nodes and the individual branches. Polygonal approximation of those branches enclosing a region are next achieved.

The following few paragraphs briefly introduce the organisation of this paper. In section 2, some key concepts have been formally defined in the context of digital meshes..

The criteria for detecting nodes have been laid down in section 3. This is followed by a traversal algorithm for representing the mesh as a set of nodes and branches. This procedure finds the Freeman chain codes for the branches in course of the traversal itself. The algorithm for traversal detects clusters of nodes from which the true nodes are found using a procedure of sequential elimination, while the false nodes are accommodated in branches connected to the node. After this, any suitable dominant point detection algorithm may be used to find breakpoints on the branches.

Section 4 provides the steps for generating the cover map of an image. Firstly, the nodes and branches are determined and then an iterative procedure is used to find the enclosures in the image.

In section 5, the procedure presented in section 3 has been used on an arbitrarily drawn mesh. The nodes and branches have been detected. This is followed by the detection of dominant points on these branches and generation of the cover map. Finally, the mesh is shown in the form of a set of polygons sharing their sides.

2: Preliminaries

2.1: Definitions of certain terms

Boundary point: A boundary point, as the name suggests, is a point that lies on the boundary of the image considered.

Branch: A branch consists of a series of points starting from a node (or boundary point) and ending at another node (or boundary point) with no node in between such that it is possible to represent all the points in the branch by means of a single Freeman chain code.

Node: If three or more branches meet at a point, it is not possible to represent all of them by a single Freeman chain. In such a case, the point at which the branches meet is called a node. Pnode is a preliminary node i.e., a point encountered during traversal that may be a node.

2.2: Condition for least deviation

This condition is being defined in order to facilitate the specification of certain conditions in our scheme for mesh traversal as well as true node identification.

In any continuous digital curve, there is a sequence of the points based on which the Freeman chain code is made.

Say,

$\vec{c}_i = \overline{p_{i-1}\, p_i}$

$\vec{c}_{i+1} = \overline{p_i\, p_{i+1}}$

We define deviation to be

$D_i = (\vec{c}_{i+1} - \vec{c}_i)$ if $(0 < (\vec{c}_{i+1} - \vec{c}_i) \leq 4)$

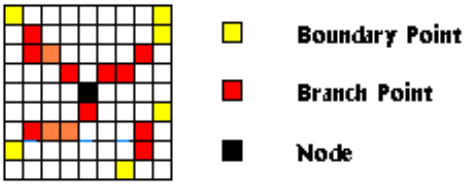
$D_i = 8 - (\vec{c}_{i+1} - \vec{c}_i)$ if $(4 < (\vec{c}_{i+1} - \vec{c}_i) \leq 7)$

$D_i = (\vec{c}_i - \vec{c}_{i+1})$ if $(-4 < (\vec{c}_{i+1} - \vec{c}_i) \leq 0)$

$D_i = 8 - (\vec{c}_i - \vec{c}_{i+1})$ if $(-7 < (\vec{c}_{i+1} - \vec{c}_i) \leq -4)$

The condition for least deviation, therefore implies that p_{i+1} must be chosen (in a mesh, there may be such a choice between two or more neighboring points) so that the value of D_i is the minimum.

Fig. 1: Nodes, branches and boundary points



3: Scheme for mesh traversal to detect nodes and branches

In this recursive scheme, there are two fields associated with each pixel. The first field specifies whether the pixel has a value 0 or 1. The other field specifies whether the pixel has been visited i.e., in the course of graph traversal, the pixel has been encountered and either classified as a pnode or made part of a branch. There are also two variables that are used to store the last visited pixel and the last visited pnode. In the scheme presented below, it should be assumed that these variables are suitably updated at each step. There is another point that needs to be mentioned. In the algorithm, there is no mention of boundary points. Actually, boundary points are treated in the same way as hanging branches by adding two layers of empty pixels on all sides of the image. In this way, the boundary points also become end points of hanging branches thereby permitting a more generalized treatment of the mesh. But before the scheme for traversal can be suggested, we must have a set of criteria for spotting points that may be nodes i.e., the pnodes.

3.1: Criteria for pnode identification

Two necessary conditions can be set down for a point to be a node. They are as below. The points that satisfy the necessary condition for being a node shall be referred to as pnodes (preliminary nodes) for convenience. In order to be classified as a pnode, a point needs to satisfy only one of the conditions.

Condition 1. In its first neighborhood, the point has one of the pixel configurations shown in Fig. 2. In each of these 3-pixel configurations, it is impossible to construct a continuous Freeman chain passing through all the points that does not pass through any point twice. Thus, it is not possible for any of these formations to be a part of a branch. These pnodes shall be called Type 1 pnodes.

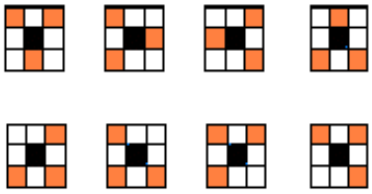


Fig. 2:Type 1 pnodes

or

Condition 2. The point has at least 3 pixels in its first neighborhood; and its second boundary when traversed as shown in Fig. 3 provides at least 3 such pairs of consecutive pixels along the path of traversal which have a pixel of value 1 immediately followed by a pixel of value 0. Such pnodes shall be called Type 2 pnodes.

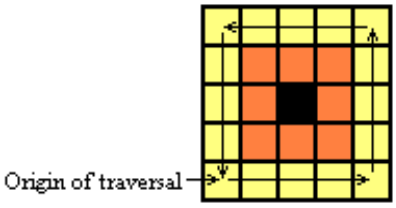


Fig. 3:Traversal of the second boundary in detecting nodes of type 2

This condition is imposed to imply that the branches must spread out sufficiently so that two branches are separated by at least a single pixel in the second boundary. Thus, not only must the second boundary have 3 or more pixels belonging to the mesh, there must be 3 or more continuous blocks (the term 'block' has been explained with the help of Fig.4) of such pixels separated by empty pixels. If this is not true, then at the stage of graph traversal, there is ambiguity in recognizing separate branches. So, this condition actually implies that the input mesh must have an adequate degree of resolution. This is true in Fig. 4.

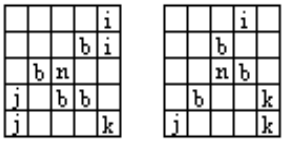


Fig. 4a **Fig. 4b**

Fig. 4 : Blocks of pixels in 2nd neighborhood of nodes

In both the figures there are 3 branches. In Fig.4a, the second boundary has three blocks; two of them having two pixels each (blocks with pixels marked i and j) and one in the bottom right having a single pixel (block with pixel k).

Similarly, Fig. 4b also has three blocks in its second boundary. Two of these have exactly one pixel (blocks with pixel marked i and j) and the other has two pixels (block with pixels marked k).

As already mentioned, all those points that satisfy either of the above conditions shall be called 'preliminary nodes' or 'pnodes'. In practice, this set of conditions (particularly the second one) may detect a small cluster of nodes instead of a single one at the meeting point of several branches. This defect is particularly pronounced when the resolution in digitizing is poor. Some of the points detected as preliminary nodes are actually parts of branches. They must therefore be incorporated into the proper branch.

It may appear that by using a higher order boundary (third, fourth etc.), the pnode may be identified more accurately. But this also increases the chance of pixels from an external branch being mistaken for a branch connected to the node (Fig.5).

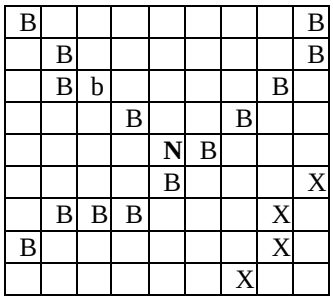


Fig. 5 : Ambiguity in node-identification that may be caused by pixels belonging to higher order boundaries

As seen in Fig. 5, the third and fourth boundaries have pixels from a branch (pixels marked as X) which is not connected to the node at the center.

The issue for finding the actual nodes of the mesh is addressed, after the branches of the mesh are detected. This is because the points that satisfy the necessary condition for being nodes, but are not true nodes, have to be incorporated in a branch of the mesh. Thus, identification of branches is a prerequisite.

After having identified those points that have a high likelihood of being nodes, the way the branches are joined is to be determined. In simple closed curves it is assumed that the Freeman chain code is available initially. But for real images, particularly involving meshes, such an assumption cannot be made. Therefore some means has to be found to detect as well as to represent the connections of various nodes and branches. For this, the graph has to be traversed. The following recursive scheme has been suggested for this.

3.2: Mesh Traversal Algorithm

Step1. Start at an arbitrarily selected pnode.
Go to step 2.
(The function shall return after the mesh has been completely traversed)

Step2. {
if (the point in question has no more unvisited first neighbors)
{
store the point;
if (it is a pnode)
{
/* indicate that a pnode has been found */
/* indicate branch termination */
return;
}
}
else
if (the point is a pnode)
{
store the point;
/* indicate that a pnode has been found */
for(all its unvisited first neighbours)
{
store the pnode;
traverse the neighbor i.e., go to step 2;
return;
}
}
else
{
if (the point has exactly one unvisited neighbor)
{
store the point;
traverse the unvisited neighbor i.e., go to step 2;
return;
}
}

else /* i.e. the point is not a pnode and has more than one neighbor */
{
store the point;
traverse the neighbor that has fewest neighbors i.e., go to step 2;
/*if (there is a conflict between two or more neighboring points i.e., all have the same number of neighboring points, which is also the least)
traverse the neighbor that satisfies the condition for least deviation*/
/* see section 2.2 for condition for least deviation */
return;
}
}

After the above procedure is complete, each point in the mesh can be specified as a pnode or as a part of a specified branch.

3.3: Procedure for finding true nodes of a mesh

Here it is assumed that the mesh has already been traversed as per the procedure given above. The procedure is presented as follows:

Step1. Find pnode clusters. A cluster, as the name suggests, is a set of pnodes, such that

a) Each pnode must belong to exactly one cluster.

b) Each pnode in a particular set is a first neighbor of some other pnode in the same set, except if the cluster has unit cardinality.

c) No pnode in a particular set is a first neighbor of a pnode that belongs to a different set.

Step2. Several branches are joined to each pnode cluster. To each such branch, a pair of terminating nodes must be assigned unless it terminates in a boundary point or is hanging at one end. In the process of mesh traversal, the branches are detected in such a way that each branch has a pair of terminating nodes, one at each end. These are the initial, or default terminating nodes. Now, consider those two points in each branch that are not pnodes but in the sequence in which points are detected in the branch, they occur just before or after the pnodes. In case, only one pnode in the cluster is its first neighbor, then this pnode is assigned as a terminating pnode of this branch. If there are several such pnodes as first neighbors, the pnode that satisfies the condition for least deviation (see Section 2) is chosen as the terminating pnode.

Step3. The true nodes are obtained through a process of sequential elimination that progresses in several stages.

A pnode may NOT be eliminated at a particular stage if

a) It is of type 1, or
b) All its first neighbors are pnodes, or
c) Two or more branches terminate on it.

In any of the above three cases, it is not possible to incorporate the pnode into an existing branch uniquely.
For pnode elimination, the following scheme is suggested.
For each cluster,

Step I. If no more pnodes can be eliminated, stop.
Else go to step II.

Step II. For each surviving pnode, compute
n1 = number of pnodes in its first neighborhood.
n2 = number of points in its first neighborhood which are not pnodes.
n3 = number of branches terminating at the pnode.

Step III. Of the pnodes that may be eliminated, eliminate the one that has the minimum value of n1. In case there are two or more pnodes with the same value of n1, which is also the minimum, eliminate the one that has the minimum value of n2. In case there is still a conflict, eliminate the pnode with the minimum value of n3 (n3 may be zero). It may so happen that the pnode chosen for elimination has no branches terminating on it. By the criteria given in Step III of Step3 of the procedure for finding true nodes, the pnode to be eliminated must have a neighboring point that is not a pnode. However, it is not necessary that this neighboring point must belong to a branch that terminates on this pnode. In such a case, the terminating pnode of the branch must be changed so that the branch now terminates on the pnode in question. Following this, the pnode may be eliminated. If the conflict is still not resolved, remove that pnode for which the new terminating point has the maximum value of (n1+n3). Elimination consists of making the pnode a part of a branch that shall now terminate on a new pnode within the same cluster, subject to the condition for least deviation.

Sometimes, there may be ambiguity because two points appear to satisfy the condition for least deviation equally well. In this case that new terminating node should be chosen as the one that has the maximum value of (n1+n3). In case there is still no clear choice, either of the two pnodes may be chosen for termination. After elimination go back to S*tep I*.

Step IV. After the above three steps are complete, from each cluster, one or more true nodes will be obtained. In some cases, two nodes may be detected that touch. In such cases, there may be an exceptional condition where these two nodes may be coalesced into one. Let the nodes be named *a* and *b*. If *n* branches meet at node *a* (actually, on either node), of which *n-1* touch node *b* before terminating on *a*, then these *n-1* branches may be made to terminate on *b* directly while the node *a* itself may be made part of the remaining branch which may then be made to terminate on *b*.

4: Generation of the cover map

Meshes or networks occur in several different types of images. In particular, the scheme presented above may be used for generating the cover maps for remotely sensed images. Here the problem is one of finding enclosures bounded by the branches of a mesh. The branches may represent roads, borders etc.

The basic principle of the scheme is that: from any point that is neither a node nor a part of a branch (i.e. not a point ON the mesh), the points are scanned in all eight directions until a boundary or a branch is encountered. If even a single boundary point is encountered, it obviously means that the enclosure extends beyond the limits of the image. Otherwise, the branches encountered are checked to see if they can form a complete enclosure. This can be simply ensured by putting the nodes that form the end points of the branches encountered into a set (in which elements may be repeated). When every node occurs exactly twice in this set, the enclosure is complete. Here, it is assumed that there are no hanging branches.
Formally, the following procedure is proposed:
Initially, all branches and the ending nodes are available.

Step 1. The various branches are put into two sets,

a) set A with branches with nodes at both ends,
b) set B with all other branches.

Step 2. Let *N* be the set of all the nodes in the mesh.

$$N = \{n_1, n_2, \ldots n_K\}$$

where *K* is the total number of nodes

For each of these nodes, put all the second neighbors that are not a part of the mesh, into a set *S*. These points serve as the starting points from which we start our search for enclosures.

$$S = \{s_1 , s_2 , \ldots s_P \}$$

where P is the total number of starting points

Such a set of starting points is chosen because it nearly always ensures that there will be at least one starting point within each enclosure. An alternative would be to start with those points that are not points ON the mesh, but are first neighbors of the mid-points of the branches.

Step 3. To each of these points s_i, assign two sets S_{ib} and S_{in}.

Starting from each point s_i in the set S, points are scanned in all eight neighboring directions. Whenever a branch is encountered, the branch is made an element of the set S_{ib} and the two terminating nodes are put into the set S_{in}. S_{in} is not truly a set in the sense that a node may be put into it twice. An enclosure is said to be complete when each of the elements in S_{in} occurs twice, i.e. the branches enclose the region entirely. If it be found that traversing in eight directions from a starting point fails to detect all the branches, then search for remaining branches may be resumed from other points which are known to be within the same enclosure; such as points which have been encountered during the search for branches and which are not points on the mesh. This process is repeated until all enclosing branches have been found. In case, a boundary point, or a branch that terminates on the boundary, is encountered, it is known that the enclosure extends beyond the image boundary and the search should be aborted.

At the end of this exercise, corresponding to each point s_i, we shall have either all the branches surrounding an enclosure or the information that the search was aborted. Two or more starting points within the same enclosure may cause it to be detected repeatedly. These repetitions should be omitted and this representation of the collection of enclosures is equivalent to the cover map of the image.

5: Polygonal approximation

The following are the steps to be followed:

1. Firstly, all nodes must be treated as dominant points since they contain vital information about the mesh.
2. Regarding the rest of the points on the branches, it is proposed that each branch be treated individually.
3. Any dominant point detection scheme that is applicable to closed curves may also be used for breakpoint detection on the branches. In some such schemes, modifications may be needed for points near the nodes. In particular, algorithms that require regions of support larger than one shall have to be modified to take into account the endpoints of branches.

In this paper, for finding dominant points, we shall use the following scheme that requires a region of support no larger than unity so that no modifications for the ends of the branch are required.

In this scheme,

Let the points on the closed curve be denoted as

p_i, $i = 0, \ldots, n-1$

P = starting point

a_i = a quantity related to the angle of the ith point, which is defined as follows

If $(f_{i+1} = f_i)$ $a_i \leftarrow 0$

else if $(f_{i+1} - f_i = 1 \text{ or } -7)$ $a_i \leftarrow 1$

else if $(f_{i+1} - f_i = -1 \text{ or } 7)$ $a_i \leftarrow -1$

otherwise $a_i \leftarrow 2$.

t = maximum allowable error limit

n = total number of points on the closed curve

$p_l.x$ = x co-ordinate of the lth point p_l

$p_l.y$ = y co-ordinate of p_l

p_f = the point on the portion of the curve between the dominant points p_l and p_m that is farthest from the straight line segment joining p_l and p_m.

The steps in detecting the dominant points are as below.

Step 1. Determine the parameter a_i at each point

Step 2. Start at an arbitrarily chosen point P.

Step 3. If $(a_i = 2)$

set p_i as dominant

If $(|a_i| = 1)$

Let p_k be the previous point encountered in traversing the curve that had non- - zero curvature (not necessarily p_{i-1}).

If $(a_i = a_k = \pm 1)$

set p_i , p_k as dominant points. (here i represents i modulo n.)

If the first point detected as dominant is re-encountered,

Go to step 4.

Else repeat step 3.

Step 4. Consider pairs of dominant points such that no other dominant point occurs between the pair. Start from the initial dominant point pair detected.

Step 5. Let the points be p_l and p_m.

Let

$p_l.x$ = x co-ordinate of p_l

$p_l.y$ = y co-ordinate of p_l

Similarly for p_m.

Now,

If $(|p_l.x - p_m.x| > t)$ and $(|p_l.y - p_m.y| > t)$

and $(||p_l.x - p_m.x| - |p_l.y - p_m.y|| > \sqrt{2}\, t)$

then find which point lying between p_l and p_m is farthest from the straight line joining p_l and p_m.

Let the point be p_f and its distance be d_f.

If $(d_f > t)$

store p_f as dominant

$m \leftarrow f$

Else

$l \leftarrow m$

$m \leftarrow$ the next dominant point.

Step 6. Terminate if all pairs have been considered. Else repeat step 5.

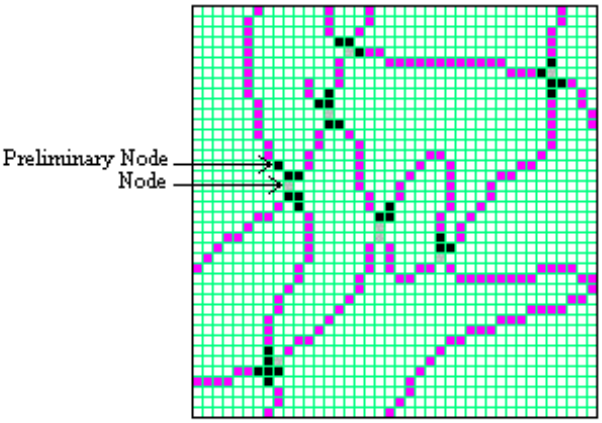


Fig 6a. Pnodes and Nodes on the Mesh

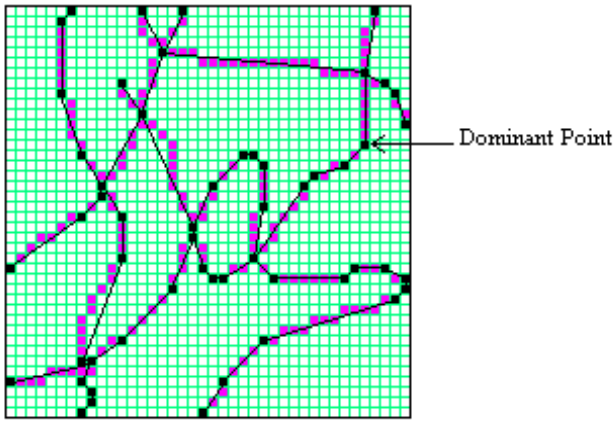


Fig 6b. Dominant Point Detection on Branches of the Mesh

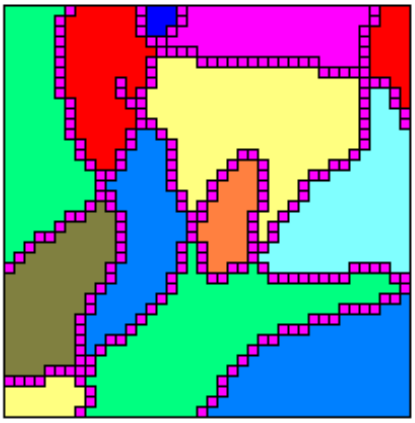


Fig 6c. Cover region on the scene

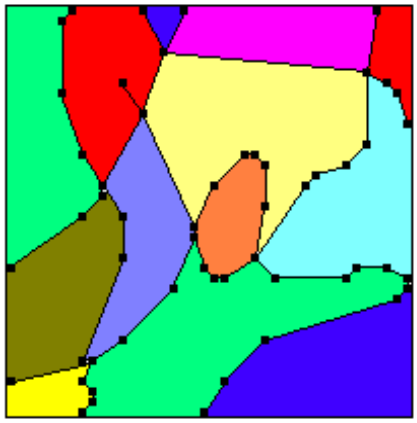


Fig 6d. Polygonal approximation of the cover map.

6 : Results and Conclusions

As seen above, in Fig. 6, the algorithm works quite well on the mesh that has been used as input. The satellite images resemble a network of curves after the operations of segmentation, edge-detection and edge linking have been performed. The results after the various stages of the procedure proposed here have been displayed in different figures. The criteria for pnode detection detect clusters of points around the true nodes as shown in Fig . 6a. The enclosures can be reliably detected (Fig. 6c). This is followed by dominant point detection, the results of which have been shown in Fig. 6b. Finally, the entire mesh is represented by a number of polygons sharing their sides (Fig. 6d). This algorithm works very well provided that the resolution is adequate and the edges are continuous. This is not a serious limitation because these requirements can be fulfilled without much difficulty. This algorithm should be useful for approximating a cover map and finds lot of applications in GIS.. As already mentioned, the general procedure for node and branch detection is expected to find applications in other fields also. Some of the aspects of this algorithm may be further improved. For instance, identification of true nodes may be achieved using the genetic algorithm keeping the basic approach unchanged.

References

1) F.Attneave, " Some informational aspects of visual perception, *Psychol. Rev.*, vol. 61, no.3, pp. 183-193, 1954.

2) A.Rosenfeld and E.Johnston, " Angle detection on digital curves,"*IEEE Trans. Comput.*, vol. C-22, pp. 875-878, Sept. 1993

3) C.Teh and R.T.Chin, "On the detection of dominant points on digital curves," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-11, pp. 859-872, Jan. 1989.

4) N.Ansari and K.Huang, " Non-parametric dominant point detection,"*Pattern Recognition*, vol. 24, No.9, pp. 849-862,1991.

5) K.Wall and P.E.Danielsson,"A fast sequential method for polygonal approximation of digitized Curves," *Comput.Vision,Graphics,Image Processing*,vol. 28,pp 220-227,1984.