Graph theory for image analysis: an approach based on the shortest spanning tree

Article *in* IEE Proceedings · May 1986

DOI: 10.1049/ip-f-1:19860025 · Source: IEEE Xplore

A. G. Constantinides
Imperial College London

Graph theory for image analysis: an approach based on the shortest spanning tree

O.J. Morris, B.A., M. de J. Lee, B.Sc.(Eng.), A.C.G.I., and A.G. Constantinides, B.Sc.(Eng.), Ph.D., C.Eng., M.I.E.E.

Indexing terms: *Image processing, Mathematical techniques, Graph theory and topology*

Abstract: The paper describes methods of image segmentation and edge detection based on graph-theoretic representations of images. The image is mapped onto a weighted graph and a spanning tree of this graph is used to describe regions or edges in the image. Edge detection is shown to be a dual problem to segmentation. A number of methods are developed, each providing a different segmentation or edge detection technique. The simplest of these uses the shortest spanning tree (SST), a notion that forms the basis of the other improved methods. These further methods make use of global pictorial information, removing many of the problems of the SST segmentation in its simple form and of other pixel linking algorithms. An important feature in all of the proposed methods is that regions may be described in a hierarchical way.

1 Introduction

A common and fundamental problem in image analysis and understanding is the partitioning of a picture into regions that are in some sense homogeneous, and different from neighbouring regions. A number of properties of the image, such as shade, hue, texture or contrast, could be used to define the measure of homogeneity. In this paper pixel intensity will be used for the sake of simplicity, although other more complex pictorial properties are just as valid.

There are essentially two ways of describing a partition of an image. Regions can either be defined by the union of their component pixels, forming the basis of segmentation methods, or, alternatively, they can be described by the boundaries around them, resulting in edge detection techniques. To date many different methods of image segmentation and edge detection have been proposed [1, 2]. This paper unifies segmentation and edge detection by using graph theory to show that they are dual problems. Segmentation is considered initially.

A problem with any method of segmentation is to define the level of detail which is of interest. Sometimes fine detail is important, while in other cases larger-scale features are more significant. This variation in the 'scale of interest' means that most methods developed and currently used to date that are suitable for one situation may not be so for another. In order that segmentation techniques be effective in all cases, they must be able to describe regions in a hierarchical order of importance. For example, if two regions are required then the image must be partitioned in such a manner that the two most significant regions are found. If a third region is desired, a further partition must be made so that the next most significant region is obtained. Consequently, if only a few regions are generated, global features of the image are represented and, as more regions are generated, more detail in the image signal is uncovered.

In solving this problem only two aspects of the image are available for monochrome images: the intensity of each pixel (or some other pixel-related pictorial property) and the spatial relationship between one pixel, or group of pixels, and another. Many algorithms that use global information in the image for segmentation, such as histogram clustering methods, ignore the all-important spatial relationship. In this paper graph theory will be used to analyse images using both of these aspects.

Graph theory has been used previously by some authors for image analysis. Zahn [3] has shown that his minimum spanning tree, a concept equivalent to the shortest spanning tree (SST), can be used for cluster analysis in graphs to produce connected groups of vertices. The spatial Euclidean distance between these vertices was used as the weight function to be minimised by the tree. However, the problem of mapping images onto graphs was not addressed. Suk and Song [4] have used SSTs, based on Zahn's approach, to design boundary-following algorithms for joining edge points found by normal edge detectors. Indeed, the SST was not used for edge detection itself, but as a method of connecting the isolated pixels previously found by a Sobel operator. Narendra and Goldberg [5] have used directed graphs to define regions in edge-detected images. Again, the SST is not used, but the mapping of images onto graphs (albeit a different mapping from those used in this paper) was investigated. In an interesting paper, Cross and Jain [6] have used SSTs of eight-way connected graphs of images in order to find a measure of image homogeneity. However, no attempt was made at segmentation or edge detection. In a contribution often quoted in the literature, Horowitz and Pavlidis [7] used quadtrees in a graph-theoretic framework to obtain segmentations. However, the mapping they have adopted and the quadtrees that result from their approach are fundamentally different from work contained in this paper.

In this paper spanning trees are used to partition images, new hierarchical segmentation techniques are introduced that describe the image for all 'scales of interest', and it is shown that segmentation and edge detection are dual problems, thereby introducing a new class of edge detection methods.

Section 2 introduces some basic graph-theoretic methods. In Section 3 it is shown that the SST exhibits useful properties for describing images, but also has some shortcomings in its most simple form. In Section 4 methods are developed for the removal of these shortcomings, and which produce hierarchical descriptions of images by incorporating global pictorial information about the image. Section 5 adapts the segmentation methods for edge detection. Finally, in Section 6 some applications of these techniques are discussed.

Paper 4390F (E4), first received 5th June and in revised form 18th November 1985

The authors are with the Department of Electrical Engineering, Imperial College of Science & Technology, Exhibition Road, London SW7 2BT, United Kingdom

2 Graph theory

This Section introduces the terminology of graph theory and all the relevant methods needed to understand the segmentation and edge detection techniques presented in this paper.

2.1 Definitions

Graph theory is the study of graphs and their applications. A graph $G = (V, E)$ is composed of a set of 'vertices' V_i connected to each other by 'links' $E_{i,j}$, where V_i and V_j are the terminal vertices that the link connects. In a weighted graph the vertices and links have weights associated with them, v_i and $e_{i,j}$, respectively. Each vertex need not necessarily be linked to every other, but if they are then the graph is complete. A partial graph has the same number of vertices but only a subset of the links of the original graph. A subgraph has only a subset of the vertices of the original graph, but contains all the links whose terminal vertices are within this subset.

A 'chain' is a list of successive vertices in which each vertex is connected to the next by a link in the graph. A 'cycle' is a chain whose end links meet at the same vertex. A 'tree' is a connected set of chains such that there are no cycles. A 'spanning tree' is a tree which is also a partial graph. A 'shortest spanning tree' of a weighted graph is a spanning tree such that the sum of its link weights, or some other monotonic function of its link weights, is a minimum for any possible spanning tree. A 'forest' is a set of trees, and a 'spanning forest' is a forest which is also a partial graph.

A graph is planar if it can be drawn on a plane with no links crossing. The dual of a planar graph is obtained by placing a vertex in each space between the links (including the space outside the graph), and joining them with new links such that each old link is crossed by a new one.

Fig. 1 shows an example of a graph and one of its two possible shortest spanning trees. A fuller explanation of these terms can be found in Reference 8.

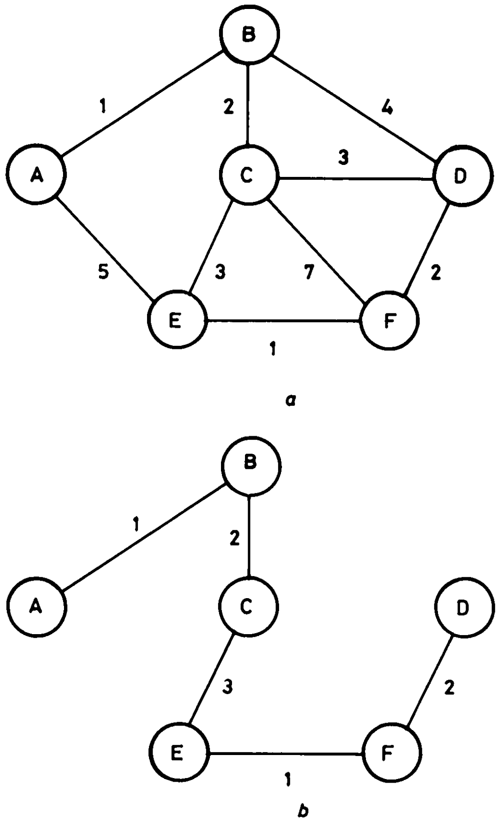


Fig. 1 *Example of a graph and its SST*
a Example graph
b SST of graph

2.2 Mapping images onto graphs

In order to analyse images using graph theory, the original image must be mapped onto a graph ready for processing. There are a number of possible mappings, but the most obvious one is to map each pixel in the image onto a vertex in the graph. The vertex weights can be made equal to the corresponding pixel intensities, or a more complex weight function can be used. Thus, if the image intensity at a point (x, y) is $I_{x,y}$, then the corresponding vertex weight is

$$v_i = I_{x,y} \tag{1}$$

where x, y is mapped onto i in a one-to-one mapping.

If the link weights are defined as the absolute value of the difference between the vertex weights that they join, i.e.

$$e_{i,j} = |v_i - v_j| \tag{2}$$

then the link weight is a measure of similarity between the two vertices, and hence between the corresponding two pixels.

A vertex could be linked to any other, but a useful simplification is only to link vertices to their nearest neighbours. Either four or eight neighbours may be considered. Fig. 2 shows the mapping of an image onto an eight-way-connected graph.

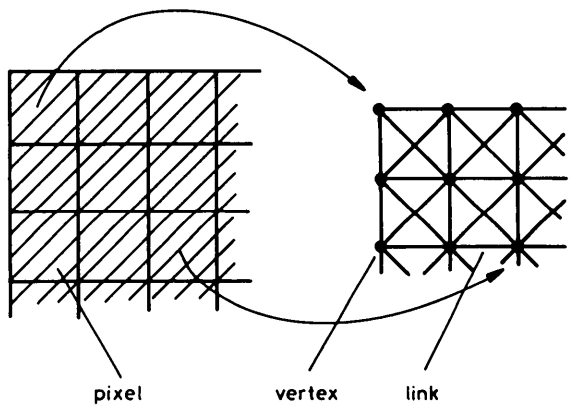


Fig. 2 *Mapping an image onto an eight-way-connected graph*

2.3 Obtaining shortest spanning trees

A formal description of the SST is given in Reference 9. A more informal description is given here. For any tree in a forest, if an link with the lowest weight which connects that tree to another is added, then the link will be in the SST. This is true even if there are links in the forest not in the SST [8, 9]. If the forest forms part of an SST then the complete SST can be found by successively adding these new links to it. The SST, therefore, tends to include links with low weights. More costly links are only included when there is no alternative.

Algorithms for finding an SST of a graph are well known [8–12]. As the graphs to be considered in this paper are very sparse, there being only a few links per vertex, Kruskal's algorithm is used [12]. This algorithm has an execution time proportional to $O(E \log (V))$, where E is the number of links and V is the number of vertices in the graph. Faster algorithms have been proposed [9] which would be useful if speed were a limiting factor.

Kruskal's algorithm works as follows:

(1) Initially the forest contains no links.
(2) Repeat:
  (3) Find the next least-weighted link.
  (4) If the link would not form a cycle with the forest:

(5) Add the link to the forest.
(6) Else:
(7) Discard the link.
(8) Until the forest becomes a tree spanning the graph.

2.4 Forming segmentations from spanning trees

A spanning tree of a graph can be used to form a partition by cutting its links. This is because the forest so produced consists of a set of disjoint trees which are used to define each partition. Vertices in each tree of the forest describe the pixels in that partition. The forest still spans the graph, so that no vertices are left out of the segmentation. If T is a tree in a forest, then it defines a partition $P(T)_i$:

$$P(T)_i = \begin{cases} 1, \text{ if } V_i \in T \\ 0, \text{ otherwise} \end{cases} \tag{3}$$

Once a partition of the graph has been obtained, the reverse problem must be considered; that is the mapping of the graph back onto an image.

A method of doing this to generate a segmentation image $S_{x,y}$, such that each pixel in any one region is assigned a constant intensity value which is the mean of all the pixels within that region. Thus, let $p(T)$ be the partition weight, such that

$$p(T) = \sum_i P(T)_i \cdot v_i \Big/ \sum_i P(T)_i \qquad \forall i \tag{4}$$

If $M_{x,y}$ is the mapping from (x, y) to T, then

$$S_{x,y} = p(M_{x,y}) \tag{5}$$

3 SST segmentation

Now that all the necessary principles have been introduced, we proceed to the explanation of the SST segmentation, give an example and discuss its properties.

3.1 Principles of SST segmentation

The object of the segmentation method is to group together pixels that are similar in some sense, and to separate pixels that are dissimilar according to the same measure. A simple measure of similarity between pixel values is to take the absolute value of the difference used in eqn. 2. In the graph the most similar vertices are linked by locating the lowest link weights. This is done for the whole graph by finding an SST. As the link weight function is to be minimised, any monotonically increasing distance function can be used, but the function of eqn. 2 is a simple one to implement. Once the SST has been found it can be used to form a segmentation by cutting it at its highest link weights, so forming partitions that differ from their neighbours by the maximum amount. This guarantees that each pixel has neighbours in its own region which have values closer to its own than any neighbouring pixel in a different region. If a further region is required by making a further cut in the SST, then the next most contrasting region is found. This means that the segmentation is hierarchical, providing regions in order of maximum contrast.

A simple segmentation algorithm can now be envisaged to form N regions:
(1) Map the image onto a weighted graph.
(2) Find an SST of the graph.
(3) Cut the SST at the $N - 1$ most costly links.
(4) Map the forest onto a segmentation image.

3.2 Example of SST segmentation

Fig. 3 shows the result of the SST segmentation on a 128^2 pixel image with 256 grey levels. The example was chosen to illustrate many properties of this method. The process works well for large numbers of regions. Fig. 3*b* shows an

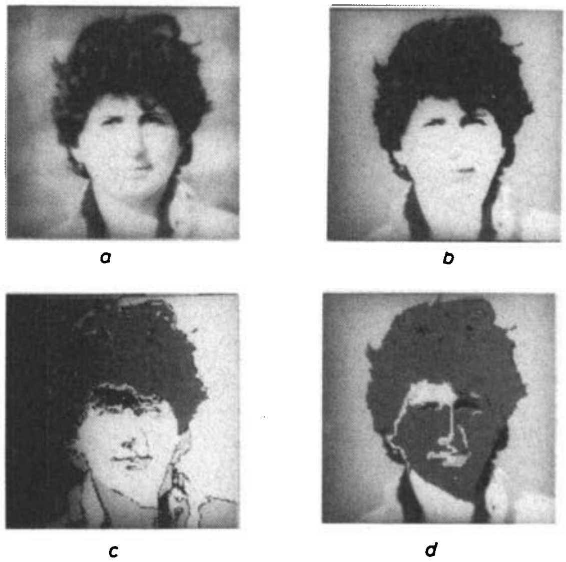


Fig. 3 *SST segmentation example*
a Original image
b 826-region SST segmentation
c Region map of Fig. 3*b*
d 825-region SST segmentation

826-region segmentation image. Many of the regions are not immediately apparent, because the intensities of neighbouring regions are very similar. Fig. 3*c* is a region map of the segmentation image, such that each region is contrasted from its neighbours, revealing many very small regions. When the number of regions is small the SST segmentation does not fare as well, because no global information about the image is used. Indeed, neighbouring regions which are apparently different may have adjacent pixels which are very similar, and so will be merged. This is illustrated by Figs. 3*b* and *d*, which differ by only one region. It can be seen that the two regions of the face and hair have merged into one.

3.3 Properties of SST segmentation

From the example, various important features of the SST segmentation are apparent:

(*a*) Spatial information about neighbouring pixels is used, unlike most histogram segmentation methods [2, 13], which group pixels together if they share a common feature, without requiring these pixels to be connected.

(*b*) Region boundaries are defined very accurately. Jagged edges are not produced as they are, for example, by split-merge algorithms [2, 7].

(*c*) The regions produced are closed. There is no need to use edge-following procedures to convert edge points into boundaries.

(*d*) The SST contains all the information needed for splitting the image into any number of regions defined in a hierarchical order of contrast. Consequently the addition or removal of a region is a simple operation.

(*e*) When a new region is added, only that specific region which is split in two is changed. This means that established boundaries do not move as more detail is included in the segmentation when regions are added.

The SST segmentation method in its present form, however, has some problems that limit its usefulness:

(*f*) No global information is used in finding the SST. Spikes in the image, which may be isolated pixels, are identified as separate regions. This means that the segmentation does not fare well when the image is very noisy.

(*g*) If a single link is left connecting two regions that ought to have been separated, they will be merged together. Fig. 2 shows that this can cause unexpected results. Similarly, regions which are very different may be merged if they are connected by a series of pixels which differ only slightly from each other, but form a large overall difference. For example, a black region and a white region may be connected by a series of pixels ranging evenly from black to white, so that the weights of the links connecting them are all very small. These two regions are likely to be merged together.

All of the problems in the SST segmentation algorithm are caused by the weight functions making no use of global information. Similar results have been found with single linkage region growing schemes [2]. Any improvement on the basic method must take account of features that cover large areas of the image, and this is done in Section 4.

4 Improved segmentation methods based on the SST

Three distinct methods for improving the SST segmentation are now described which make more complex use of the SST in order to use the global information in the image. These are:

(i) Recursive SST segmentation.
(ii) Minimax SST segmentation.
(iii) Local averaging SST segmentation.

4.1 Recursive SST segmentation

The recursive SST segmentation presents a powerful solution to the problem of incorporating global information into the segmentation method. If an image has M pixels, then a simple algorithm to find N regions is:

(1) Copy the image into an image buffer.
(2) For $I = M - 1$ down to N, do:
  (3) Generate an I-region segmentation using the SST segmentation from the image buffer.
  (4) Copy the segmentation image into the image buffer.

The algorithm expressed in this form is inefficient, but it nevertheless illustrates an important principle. At each stage of the recursion a segmentation image is generated with one fewer region than the stage before. Pixel values for each region in the segmentation image are constant, and equal to the mean of the pixels that the region covers in the original image. As the pixel values are modified according to all the pixels in the region, it is possible for one pixel to affect another which is not necessarily its nearest neighbour. The link weights between regions will also be altered to take account of all the pixels in both the regions that the link connects, so that global region information is incorporated into the link weights. Initially, only local information is used as each region consists of only one pixel, but as the recursion progresses and the regions enlarge in size the graph contains more global information.

As every pixel in each region is the same, the graph can be simplified so that a vertex represents a whole region. When the SST is cut into a forest to describe the partition, each tree can now be thought of as a single entity, which becomes a single vertex. The graph-to-image mapping is now a one-to-many mapping, and the partition $P(i)_{x,y}$ defined by a single vertex V_i is redefined as

$$P(i)_{x,y} = \begin{cases} 1, \text{ if } V_i \text{ maps onto } (x, y) \\ 0, \text{ otherwise} \end{cases} \tag{6}$$

The vertex weight function is now

$$v_i = \sum_{x,y} P(i)_{x,y} I_{x,y} \Big/ \sum_{x,y} P(i)_{x,y} \qquad \forall x, y \tag{7}$$

The initial mapping from the image to the graph is the same as before, but the structure of the graph is altered during the recursion.

An improved recursive SST segmentation can now be envisaged using the new graph. Initially, each pixel is considered to be a separate region, and each iteration of the recursion merges two regions together. At each stage the two most similar vertices, V_i and V_j, are merged across the shortest link. The new vertex V_k that is formed is assigned the mean value of the new partition:

$$v_k = \frac{\sum_{x,y} (P(i)_{x,y} I_{x,y} + P(j)_{x,y} I_{x,y})}{\sum_{x,y} (P(i)_{x,y} + P(j)_{x,y})} \qquad \forall x, y, \tag{8}$$

and the new partition $P(k)_{x,y}$ is the union of two most similar partitions:

$$P(k)_{x,y} = P(i)_{x,y} + P(j)_{x,y} \tag{9}$$

The link that joined vertices V_i and V_j is removed and saved for use when mapping the graph back to the image domain. All the other links that had vertices V_i and V_j at their ends have their weights re-evaluated using the new vertex weight. If there are now any links which are identical then the redundant ones are removed. This stage is repeated until there are no links left.

Since the terminal vertices of each of the saved links are merged at each stage of the recursion, the next stored link can never form a circuit with links previously saved. These links thus form a spanning tree of the original graph, although not necessarily an SST. They can be used to obtain a recursive SST segmentation of the image rapidly for any number of regions N by cutting the tree at the last $N - 1$ links found by the recursion.

The recursive SST segmentation algorithm for obtaining N regions is therefore:

(1) Map the image onto a weighted graph.
(2) While there is more than one vertex in the graph:
  (3) Find the next least weighted link.
  (4) Save the link.
  (5) Merge the two vertices joined by this link.
  (6) Recalculate the new vertex weight and link weights.
  (7) Remove duplicated links.
(8) Form a spanning tree with the saved links.
(9) Cut the spanning tree at the $N - 1$ most costly links to form a spanning forest.
(10) Map the forest onto a segmentation image.

Fig. 4 illustrates the result of applying the recursive SST segmentation to the image that was used in Fig. 3. The most important improvement over the SST segmentation is that global information about the image is used more and more as the regions are merged. The result is that if a segmentation with many regions is required, then regions are found which are locally significant. Segmentations with fewer regions will provide more globally significant regions. Segmentations are produced that find equally contrasting regions for a given number of regions N. The 'scale of interest' is automatically adjusted to the required number of regions. The recursive SST segmentation is therefore hierarchical with respect to the 'scale of interest' of the image, as well as to the contrast between regions as in the SST segmentation. Globally contrasting regions take precedence over locally contrasting regions as the number of regions is reduced.

Fig. 4d illustrates how region boundaries do not move as regions are merged. At each stage of the recursion the boundary common to the two regions to be merged is

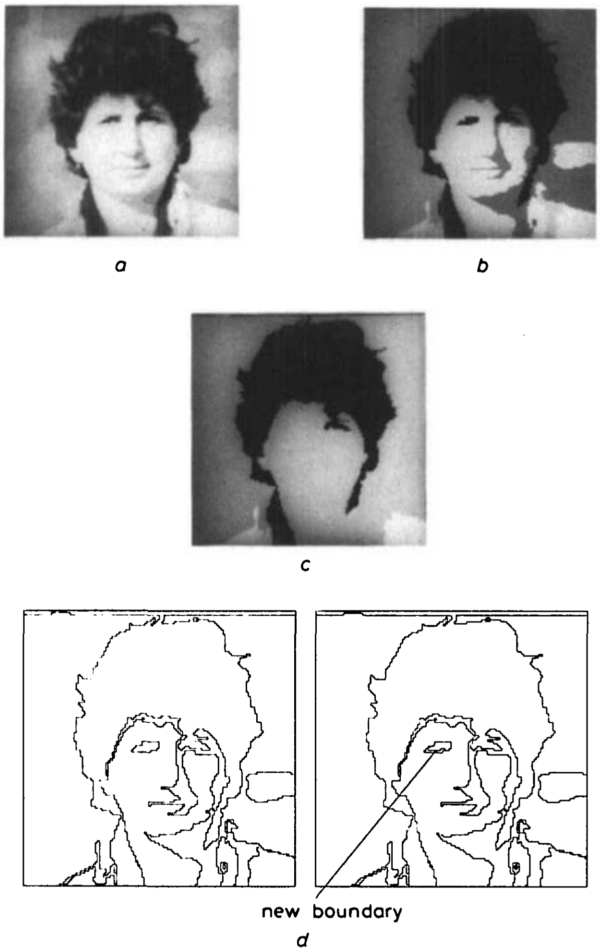


Fig. 4 *Recursive SST segmentation example*
a 826-region recursive SST segmentation
b 20-region recursive SST segmentation
c 5-region recursive SST segmentation
d Boundary map of 20- and 21-region recursive segmentation, showing how previous boundaries remain stationary

removed, but the rest of the segmentation boundaries remain unaltered.

As the vertex weight is an average of all the pixel intensities in the partition, noise becomes less of a problem as the segmentation progresses.

4.2 Minimax SST segmentation

The recursive SST segmentation is one way that global information may be incorporated into the segmentation. Another is to partition the graph into subgraphs, such that a suitable objective function evaluated on each of the resulting subgraphs is minimised. Unfortunately, algorithms for the solutions of this problem require evaluation of all possible partitions of the graph [14]. This is impracticable for a graph representing an image of any reasonable size. The minimax SST segmentation finds a suboptimal solution to this problem in a computationally efficient way.

If the SST is cut into a spanning forest F of trees T, then a cost function for each tree, $c(T)$, is defined:

$$c(T) = \max\ [\,|v_i - v_j|\,] \qquad \forall i, j \in T \tag{10}$$

The minimax SST segmentation algorithm for obtaining an N-region segmentation is:

(1) Map the image onto a weighted graph.
(2) Find an SST of the graph.
(3) Repeat N times:
(4) Find the tree T_{max} with the largest cost $c(T_{max})$, such that:

$$c(T_{max}) = \max\ [c(T)] \qquad \forall T \in F \tag{11}$$

(5) Cut T_{max} at the link $E_{i,j}$ which minimises the maximum value of the costs of T_i and T_j, the two subtrees so created:

$$\min\ [\max\ [c(T_i), c(T_j)]] \qquad \forall E_{i,j} \in T_{max} \tag{12}$$

(6) Map the forest onto an image.

Fig. 5 shows the result of applying this algorithm. It provides a hierarchical segmentation, as does the recursive

Fig. 5 *Minimax SST segmentation example*
a 826-region minimax SST segmentation *b* 20-region minimax SST segmentation

SST segmentation, finding large-scale features first and adding more detail as more regions are found. The cost function used enables images with a gradual change in intensity between two regions to be segmented. The minimax cost function is one of the simplest that can be defined for a region of the picture. Cost functions that model other features of the image, such as texture, for example, could be used.

4.3 Local averaging SST segmentation

In previous Sections global information was introduced into the algorithms, either by modifying the graph as the segmentation progressed in the recursive SST segmentation, or by cutting the original SST so as to optimise a cost function defined over the resultant trees in the minimax SST segmentation. Regional information can be built into the original graph directly by making the vertex weights depend on an area around the corresponding pixel in the image. A consequence of this is that the algorithm is more stable when the image is noisy. The only difference between this method and the SST segmentation or the recursive SST segmentation is in the assignment of vertex weights in the graph. The new vertex weight function is

$$v_i = \sum_{x=x-N}^{x+N} \sum_{y=y-N}^{y+N} I_{x,y}/(2N+1)^2 \tag{13}$$

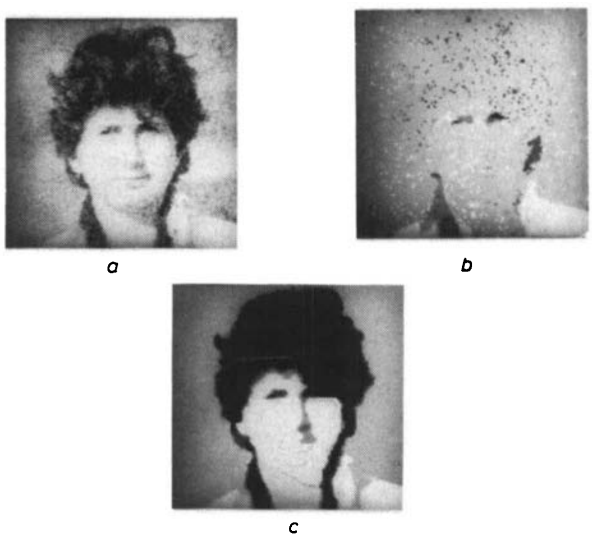


Fig. 6 *Local averaging SST segmentation example*
a Original with 10 dB white noise *b* 1300-region SST segmentation
c 1300-region local average SST segmentation

so that v_i is the local mean of the region around pixel (x, y) for an area of $2N + 1$ pixel square.

Fig. 6 uses the original image with white noise added (signal/noise ratio of 10 dB) to compare the local averaging SST segmentation (with $N = 1$) with the SST segmentation. The local averaging method has the disadvantage that it may produce spurious regions along real boundaries, because the edge is spread out by the averaging. This means that for noiseless images it is not as good as taking the difference between pixel values. However, when the image is noisy, it performs considerably better than the original method, as the segmentation image is much less speckled.

4.4 Comparison of segmentation methods

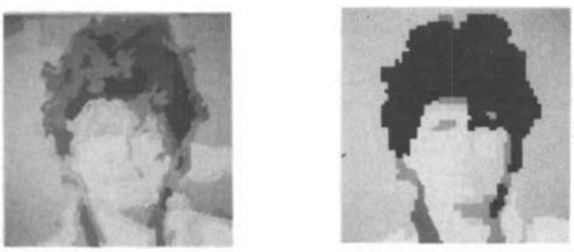
The three improvements to the SST segmentation method proposed in Sections 4.1 to 4.3 are not unrelated. The local averaging SST segmentation method differs only in the mapping from an image to a graph, and so could be incorporated into the other two methods. The minimax SST segmentation varies in its method of cutting the spanning tree, and so could also be applied to the recursive SST segmentation method, which is the only method to find a different spanning tree from the basic SST segmentation.

Fig. 5 shows how the minimax SST segmentation gives results comparable to those of the recursive SST segmentation shown in Fig. 4. The results are not as good, but the method is much simpler and faster to run. It illustrates the importance of cutting the SST in an intelligent way. Table 1 lists the main properties of the four SST Segmentation methods.

Table 1: Comparison of SST segmentation method properties

	SST	Recursive SST	Minimax SST	Local average SST
Number of regions definable	Yes	Yes	Yes	Yes
Noise immunity	Poor	OK	OK	Good
Region accuracy	Good	Excellent	Good	Blurred
Speed	Fast	Slow	OK	Fast
Neighbourhood of information	Local	Global	Global	Mask

Fig. 7 shows two different segmentation methods for comparison. Fig. 7*a* is a segmentation of the original



a   *b*

Fig. 7 *Alternative segmentation methods*
a 584-region Ohlander segmentation
b 23-region split-merge segmentation

image using Ohlander's recursive region splitting method [2, 13]. Each region is split using histograms of a number of its properties such as intensity or variance. The partitions to be made in the histograms are found using a set of six heuristics. The regions so produced are themselves split recursively until an end condition is met. The main properties of this segmentation are:

(*a*) When splitting a region no use is made of the relationship between the region and its neighbours, and so no global information outside the region is used.

(*b*) Large features are found roughly, but smaller features are not located accurately. The histograms are a poorer estimate of the probability density functions as the regions become smaller.

(*c*) At each stage of the segmentation the method produces a variable number of regions which cannot be specified beforehand. The number of regions produced cannot be controlled directly.

(*d*) Noise blurs the region histograms, resulting in less accurate region splitting. The method therefore has a low noise immunity.

Fig. 7*b* is a split-merge segmentation [2, 7] of the same image. The image is described by quarters, which are themselves made up of quarters in a recursive fashion. These quarters are represented by a quadtree in which nodes are split or merged depending on which would best satisfy a minimax cost function. Properties of the split-merge method are given here for comparison:

(i) The segmentation is made up of block-like regions because of the rigidly defined structure of the quadtree.

(ii) The method has a higher noise immunity, because the weight function is a minimax function defined over the total area of each region.

(iii) The number of regions required can be defined.

5 Duality and edge detection using SST

If the image is mapped onto a graph using four-way connectivity, then the graph is planar and its dual can be found. The mapping is shown Fig. 8. The links of the dual

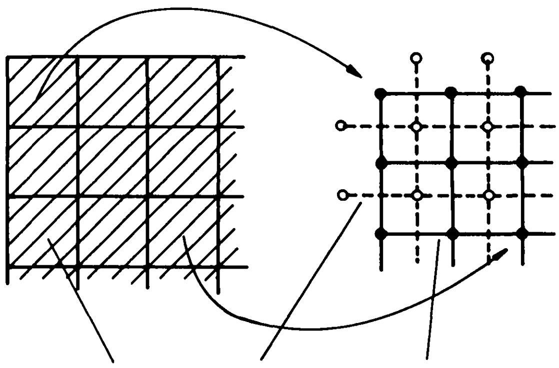


Fig. 8 *Mapping an image onto a four-way-connected dual graph*

graph lie along the boundaries between the pixels in the image, and so can be used to describe edges. Edge detection algorithms must locate edges between pixels that have very different intensities. The original graph link weights represent the similarity between pixels. Thus, if the dual graph links are made to be the negative of the links in the original graph that they cross, then an SST of the dual graph will locate the most different boundaries between pixels.

The SST edge detection algorithm is therefore the same as the SST segmentation, except that a dual graph is used. Links are taken from the SST in order of lowest weight to draw an edge image for as many edge elements N as are required. The algorithm is:

(1) Map the image onto a weighted graph.
(2) Find the dual weighted graph.
(3) Find the SST of the dual graph.
(4) Draw a line in the edge image at the co-ordinates that the N least costly links represent.

Fig. 9 illustrates the result of the SST edge detector compared to a Sobel operator edge detector. The Sobel oper-

Fig. 9 *SST edge detection example*
a 2624-point SST edge detection *b* 2624-point Sobel operator edge detection

ator edge detection was found using a suitable threshold, and the number of edge points counted. The SST edge detection was then found using the same number of edge points. The SST edge detector does not give the more bold features such thick lines as does the Sobel operator edge detector, and so is able to find more detail for the same number of edge points.

In the above algorithm the edge detection is defined by the number of edge points to be found. Alternatively, a threshold on the largest link weight to be examined could be specified in a manner comparable to the usual operation of the Sobel operator edge detector. More complex algorithms could be developed for edge detection in a similar way to the more advanced SST segmentation methods. For example, the minimax SST cutting algorithm could be applied to the dual graph to find edges between regions of similar dynamic range in order to obtain an edge detection that would be more robust in the presence of noise.

6 Discussion of SST segmentations and their applications

Segmentation and edge detection form the foundation of many problems in image analysis and understanding. The properties of the SST segmentation and edge detection methods allow solutions to a few problems that require special mention. These are:

(*a*) Composite source model image coding and restoration.

(*b*) Depth estimation using stereo image analysis.

(*c*) Image understanding.

Composite image models using segmentation images form the basis of advanced image coding and restoration techniques [15, 16]. These methods rely on accurate segmentation of the image so that regions can be modelled closely. The accuracy of the SST segmentation methods, coupled with the fact that they produce bounded regions, means that they are well suited to these applications. Significant improvements in coding rates and restoration quality would be possible using these new techniques.

As the recursive SST segmentation provides a hierarchical description of regions in an image, it could be used as a method of solving the correspondence problem in stereo image analysis. The most globally significant regions in an image could be matched by starting with an image partition of only two regions. These would be split successively by finding the next most globally contrasting regions, and then matched with the equivalent regions in the other image. This could be done recursively to match all regions in the image on an increasingly detailed level, until the whole image is matched. Special care would need to be taken with poorly matched regions, as these may result from occlusions in the scene.

The hierarchical description of images has an important implication for image understanding techniques. Pattern matching algorithms could be used recursively on the regions, starting at the most global level and working down to the smallest region, until a match is found (if one exists). This eliminates the need to correlate a pattern over the whole image for all possible pattern sizes. Images are often transformed into the frequency domain so that position- and scale-invariant recognition algorithms can be used. These complex processes, which are incapable of separating different objects in the same scene, appear to be unnecessary if the recursive SST segmentation is used. The structure of the graph defined by the interaction of links and vertices could also be used, as the spatial relationship between regions is an important aspect in image understanding.

7 Conclusions

Methods of segmentation and edge detection based on graph-theoretic descriptions of images have been developed in this paper. In this context, edge detection and segmentation are shown to be dual problems. The main features of these methods are that region boundaries are found very accurately, and that the description of images is hierarchical. Comparisons of these methods are made with others, and some implications for other areas of research have been outlined.

8 Acknowledgments

We would like to thank the UK Science & Engineering Research Council for financing this research. We would also like to thank Prof. N. Christofides for his valuable comments on the graph-theoretic content of this paper.

9 References

1 ROSENFELD, A., and KAK, A.C.: 'Digital picture processing, vol. 2' (Academic Press, 1982, 2nd edn.)
2 HARALICK, R.M., and SHAPIRO, L.G.: 'Image segmentation techniques', *Comput. Vision, Graphics & Image Process.*, 1985, **29**, pp. 100–132
3 ZAHN, C.T.: 'Graph-theoretic methods for detecting and describing gestalt clusters', *IEEE Trans.*, 1971, **C-20**, pp. 68–86
4 SUK, M., and SONG, O.: 'Curvilinear feature extraction using minimum spanning trees', *Comput. Vision, Graphics & Image Process.*, 1984, **26**, pp. 400–411
5 NARENDRA, P.M., and GOLDBERG, M.: 'A graph-theoretic approach to image segmentation'. Proceedings of IEEE Computer Society conference on pattern recognition and image processing, 1977, pp. 248–256
6 CROSS, G.R., and JAIN, A.K.: 'Measures of homogeneity in textures'. Proceedings of IEEE Computer Society conference on computer vision and pattern recognition, June 1983, Washington, DC, pp. 211–216
7 HOROWITZ, S.L., and PAVLIDIS, T.: 'Picture segmentation by a tree traversal algorithm', *J. Assoc. Comput. Mach.*, 1976, **23**, pp. 368–388
8 CHRISTOFIDES, N.: 'Graph theory: an algorithmic approach' (Academic Press, 1975)
9 CHERITON, D., and TARJAN, R.E.: 'Finding minimum spanning trees', *SIAM J. Comput.*, 1976, **5**, pp. 724–42
10 PAPADIMITRIOU, C.H., and STEIGLITZ, K.: 'Combinatorial optimization: algorithms and complexity' (Prentice-Hall, 1982)
11 SEDGEWICK, R.: 'Algorithms' (Addison-Wesley, 1983)
12 KRUSKAL, J.B. Jun.: 'On the shortest spanning subtree of a graph and the traveling salesman problem', *Proc. Am. Math. Soc.*, 1956, **7**, pp. 48–50
13 OHLANDER, R., PRICE, K., and REDDY, D.R.: 'Picture segmentation using a recursive region splitting method', *Comput. Graphics & Image Process.*, 1978, **8**, pp. 313–333
14 BELLMORE, M., and JENSEN, P.A.: 'An implicit enumeration scheme for proper cut generation', *Technometrics*, 1970, **12**, pp. 775–788
15 MITRAKOS, D.K., and CONSTANTINIDES, A.G.: 'Nonlinear image processing for optimum composite source coding', *IEE Proc. F., Commun., Radar & Signal Process.*, 1983, **130**, (5), pp. 441–451
16 MORRIS, O.J., and CONSTANTINIDES, A.G.; 'Segmented filtering: a new technique for image restoration'. IEE Colloquium digest no. 1985/51, 1985