An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation

Zhenyu Wu and Richard Leahy

***Abstract*—A novel graph theoretic approach for data clustering is presented and its application to the image segmentation problem is demonstrated. The data to be clustered are represented by an undirected adjacency graph $\mathcal{G}$ with arc capacities assigned to reflect the similarity between the linked vertices. Clustering is achieved by removing arcs of $\mathcal{G}$ to form mutually exclusive subgraphs such that the largest inter-subgraph maximum flow is minimized. For graphs of moderate size ($\sim$ 2000 vertices), the optimal solution is obtained through partitioning a flow and cut equivalent tree of $\mathcal{G}$, which can be efficiently constructed using the Gomory-Hu algorithm. However for larger graphs this approach is impractical. New theorems for subgraph condensation are derived and are then used to develop a fast algorithm which hierarchically constructs and partitions a partially equivalent tree of much reduced size. This algorithm results in an optimal solution equivalent to that obtained by partitioning the complete equivalent tree and is able to handle very large graphs with several hundred thousand vertices. The new clustering algorithm is applied to the image segmentation problem. The segmentation is achieved by effectively searching for closed contours of edge elements (equivalent to minimum cuts in $\mathcal{G}$), which consist mostly of strong edges, while rejecting contours containing isolated strong edges. This method is able to accurately locate region boundaries and at the same time guarantees the formation of closed edge contours.**

***Index Terms*—Clustering, edge contours, flow and cut equivalent tree, graph theory, image segmentation, subgraph condensation.**

I. Introduction

DATA CLUSTERING is an important methodology in exploratory data analysis. Numerous clustering methods have been reported in the literature (e.g., see [1], [2]). Given M data points, $X = \{x_1, x_2, \cdots, x_M\}$, the objective of clustering is to partition the data set into K non-empty subsets such that *alike* data are grouped together and data in different subsets or clusters are not *alike*. In this paper we propose a new graph theoretic technique for data clustering. We then demonstrate its application to the problem of image segmentation.

Manuscript received July 25, 1991; revised June 1, 1992. This work was supported by the Kaprielian Technology Innovation Fund, a privately supported fund administered by the School of Engineering at the University of Southern California, and the Whittaker Foundation. Recommended for acceptance by Associate Editor R. Kasturi.

Z. Wu is with the Medical Image Processing Group, Department of Radiology, University of Pennsylvania, Philadelphia, PA 19104-6021.

R. Leahy is with Signal and Image Processing Institute, Department of Electrical Engineering-Systems, University of Southern California, MC 2564, Los Angeles, CA 90089-2564.

IEEE Log Number 9212247.

Many graph theoretic techniques have been proposed for cluster analysis. Commonly known techniques include 1) single-link and complete-link hierarchical algorithms formulated and implemented using a threshold graph [3], [4]; 2) forming clusters by breaking inconsistent arcs in the minimum spanning tree of the proximity graph [5] or graphs constructed based on limited neighborhood sets [6]; and 3) detecting clusters using directed trees [7]. The clustering technique presented in this paper is based on network flow theory. Here minimum cuts in an undirected adjacency graph are used for partitioning the data into clusters. This idea was first proposed in our previous paper [8].

The data to be clustered are represented by an undirected adjacency graph $\mathcal{G}$: each vertex of $\mathcal{G}$ corresponds to a data point, and an arc links two vertices in $\mathcal{G}$ if the corresponding data points are neighbors according to a given neighborhood system. A flow capacity is then assigned to each arc in $\mathcal{G}$. This flow capacity is chosen to reflect the feature similarity between the pair of linked vertices. The clustering is achieved by removing arcs of $\mathcal{G}$ to form mutually exclusive subgraphs. For the case of an unconstrained optimal K-subgraph partition of $\mathcal{G}$, the arcs selected for removal are those in a set of $K-1$ minimum cuts with the smallest $K-1$ values among all possible minimum cuts separating all pairs of vertices.

This new clustering strategy possesses some desirable properties. Unlike many other clustering techniques proposed in the literature, the resulting K-subgraph partition is a globally optimal K-partition of the adjacency graph $\mathcal{G}$. It minimizes the largest inter-subgraph maximum flow among all possible K-partitions of $\mathcal{G}$, hence minimizing the similarity between the subgraphs (clusters). The difficulty in reaching a globally optimal solution for a particular partitional clustering technique arises from the requirement that a huge number of possible K-partitions must be considered. Locally optimal solutions are often found instead using iterative, hill-climbing techniques. Attempts have been made to identify and reject large numbers of obviously non-optimal partitions, using techniques such as dynamic programming [9], branch-and-bound [10], and conditional clustering [11]. Despite a substantial reduction in the number of partitions that need to be evaluated, these techniques are computationally infeasible even for problems of moderate size. In contrast, our clustering strategy can be efficiently implemented to handle very large graphs with several hundred thousand vertices. In addition, this clustering technique not only produces an optimal K-subgraph partition but also a nested sequence of partitions which are optimal for cluster numbers ranging from 2 to K. This is especially attractive when the cluster number has to be determined from the data.

In order to make this new strategy work as a data clustering method, two important issues need to be considered: 1) finding an efficient implementation scheme to make the clustering technique practical, and 2) constructing an adjacency graph $\mathcal{G}$ which can produce meaningful clusters. The minimum cuts of the undirected graph $\mathcal{G}$ can be computed from a flow and cut equivalent tree $\mathcal{T}^*$ of $\mathcal{G}$, which is constructed using the Gomory-Hu algorithm which was originally developed for solving the multi-terminal maximum flow problem for undirected graphs [12]. The Gomory-Hu algorithm involves the successive solution of exactly $M - 1$ maximum flow problems with M being the number of vertices in $\mathcal{G}$. Once $\mathcal{T}^*$ has been computed, the optimal K-partition of $\mathcal{G}$ can be equivalently obtained by simply disconnecting the $K - 1$ arcs in $\mathcal{T}^*$ with the $K - 1$ smallest arc capacities. This direct implementation is acceptable for graphs of moderate size ($\sim$ 2000). However, it quickly becomes impractical as the the size of $\mathcal{G}$ increases, due to the polynomial complexity of the algorithm. In order to overcome this problem, a fast hierarchical algorithm is developed that requires construction and partition of a partially equivalent tree $\mathcal{T}_c^*$ of greatly reduced size. This still results in an optimal solution equivalent to that obtained by partitioning the complete equivalent tree of $\mathcal{G}$. The algorithm is based on the observation that most of the minimum cuts found in $\mathcal{G}$ are never used since their associated values (the value of a cut is defined as the capacity sum of its arcs) are sufficiently large that the arcs in those cuts will not be removed to form subgraphs. New theorems providing sufficient conditions for subgraph condensation are derived. We show that many of the minimum cuts with large value can be identified using small local subgraphs, so the vertices linked by them can be condensed *before* constructing the equivalent tree. Consequently the Gomory-Hu algorithm is applied only to graphs of much smaller size, but without compromising the overall optimality of the clustering algorithm. As a result the new clustering technique can be applied to partition very large graphs.

The problem of properly constructing an adjacency graph is highly application dependent and involves the appropriate selection of a neighborhood system and an arc capacity function. We will address the problem in the context of image segmentation. It has been a common practice to segment images based on clustering techniques [13]–[15]. Many of the thresholding based segmentation techniques [16] are also implicitly related to clustering. Their approach is basically to compute a feature vector for each pixel of the image and then to segment the image by clustering these computed feature vectors together. Often little or no spatial information about the image is used. Here a new method for image segmentation is developed based on our clustering algorithm. This method effectively searches for closed edge contours which consist mostly of strong edge elements, while rejecting contours containing isolated strong edges.

There exists a class of edge operators, such as the Marr–Hildreth method [17], which are based on zero crossings that also guarantee the formation of closed edge contours. They usually suffer from inaccurately located region boundaries because they are tuned to a prespecified spatial resolution, and boundaries between objects with other spatial resolution may either be missed or inaccurately located. Furthermore there is an intrinsic drawback in finding connected regions using this type of edge operator. Region finding through zero crossings is essentially image coloring with only two colors. This forces all closed contours to be shared by exactly two connected regions. Consequently, complex region formations in an image may be poorly encoded. Edge detectors based on local intensity differences are able to locate edges more accurately at region boundaries, but do not in general provide closed edge contours for region formation. Another problem associated with this latter class is that thick edges often occur. Edge linking and thinning are usually needed to alleviate these problems. The edge-based, graph theoretic method described here attempts to combine the advantages of both of these types of edge detectors. An adjacency graph $\mathcal{G}$ is constructed with each vertex in $\mathcal{G}$ corresponding to a pixel of the image, and an arc is placed between two vertices if their associated pixels are longitudinal or vertical neighbors of each other. An edge element is defined as a separating element between the pixel pair, and its strength is computed using a local derivative operator. Strong edges are mapped to small capacities and weak edges are mapped to large capacities. Our clustering algorithm is applied to find minimum cuts in $\mathcal{G}$ with small value, or equivalently closed contours containing strong edge elements of the image. This new segmentation approach is able to accurately locate region boundaries, and at the same time guarantees the formation of closed edge contours.

This paper is organized as follows. In Section II, we review several fundamental theorems of network flow theory. These are used to formulate and develop the basic idea behind the proposed clustering technique. In Section III, we address the problem of finding an efficient implementation of the technique. New theoretical results are presented for subgraph condensation. From these an efficient hierarchical clustering algorithm is developed. This hierarchical approach results in significant reductions in computation cost relative to the direct implementation. In Section IV, the image segmentation problem is formulated as a clustering problem which seeks closed contours of strong edges. Experimental results, in which the algorithm is applied to tissue segmentation of magnetic resonance (MR) images of the human brain, are presented in Section V.

II. Formulation

In this section, we show how the problem of clustering is formulated in terms of the optimal partitioning of an undirected graph into a number of subgraphs. The new clustering technique is described here in the context of image segmentation. The algorithm is however applicable to other clustering problems where a neighborhood relationship between data points can be established and the clustering objective can be properly defined in terms of an arc capacity function.

A. Review of Network Flow Theory

The problems in network flow theory which particularly interest us are 1) finding the maximum flow from one vertex to another and 2) finding the maximum flow between every pair of vertices. The methods for solving these problems provide the foundation of our graph theoretic algorithm for data clustering.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a graph with vertex set $\mathcal{V} = \{v_1, v_2, \cdots, v_M\}$ and arc set $\mathcal{A} = \{a_{ij}\}$ with a_{ij} denoting the arc between vertices v_i and v_j. Associated with every arc a_{ij}, there is a positive number c_{ij} called the flow capacity of a_{ij}. A flow from an arbitrary vertex s to a vertex t with value F_{st} is defined by a set of numbers f_{ij} assigned to all $a_{ij} \in \mathcal{A}$ which satisfy the following conditions:

$$\sum_{v_j \in \Gamma(v_i)} f_{ij} - \sum_{v_k \in \Gamma^{-1}(v_i)} f_{ki} = \begin{cases} F_{st} & \text{if } v_i = s \\ -F_{st} & \text{if } v_i = t \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

and $0 \leq f_{ij} \leq c_{ij}$ for all $a_{ij} \in \mathcal{A}$. In (1), F_{st} denotes the value of the net flow from s to t, $\Gamma(v_i) = \{v_j | a_{ij} \in \mathcal{A}\}$ and $\Gamma^{-1}(v_i) = \{v_k | a_{ki} \in \mathcal{A}\}$, respectively. In the clustering procedure described in Section II-C, we are interested in the maximum net flow, $\overline{F}_{st}$, between all pairs of vertices, s and t. This maximum flow, for a specific pair s and t, is the maximum value of F_{st} for which there exists a set $\{0 \leq f_{ij} \leq c_{ij}, \forall a_{ij} \in \mathcal{A}\}$ which satisfies (1).

Theorem 1 (Ford–Fulkerson Theorem [18]): The maximum flow from a vertex s to vertex t, $\overline{F}_{st}$, is equal to the value of the minimum cut $(\mathcal{V}_m \rightarrow \tilde{\mathcal{V}}_m)$ separating s and t.

A cut $(\mathcal{V}_o \rightarrow \tilde{\mathcal{V}}_o)$ is a set of arcs in $\mathcal{A}$ which start from $\mathcal{V}_o$ and end in $\tilde{\mathcal{V}}_o$, where $\mathcal{V}_o$ and $\tilde{\mathcal{V}}_o$ are two vertex sets such that $s \in \mathcal{V}_o$, $t \in \tilde{\mathcal{V}}_o$ and $\mathcal{V}_o \cup \tilde{\mathcal{V}}_o = \mathcal{V}$. The value of the cut $(\mathcal{V}_o \rightarrow \tilde{\mathcal{V}}_o)$ is defined as

$$\overline{F}(\mathcal{V}_o \rightarrow \tilde{\mathcal{V}}_o) = \sum_{a_{ij} \in (\mathcal{V}_o \rightarrow \tilde{\mathcal{V}}_o)} c_{ij}.$$

The minimum cut $(\mathcal{V}_m \rightarrow \tilde{\mathcal{V}}_m)$ for the vertex pair s and t is then the cut with the smallest such value. The proof of the theorem can be found in [18, ch.1]. Ford and Fulkerson [18] have also developed an algorithm for solving the maximum flow problem through finding the minimum cut separating s from t. Observe that the minimum cut optimally partitions $\mathcal{V}$ into two subsets given the "seed" vertices s and t.

For an undirected graph $\mathcal{G}$, the problem of finding the maximum flow between every pair of vertices or the "multi-terminal maximum flow problem" can be efficiently solved using an algorithm due to Gomory and Hu [12]. A complete description of their algorithm can be found in [19, Ch.11] and will not be given here. However, the following properties of undirected graphs are fundamental to the use of the Gomory-Hu algorithm in our work.

Theorem 2 ([12]): Let $(\mathcal{V}_m \rightarrow \tilde{\mathcal{V}}_m)$ be a minimum cut separating s from t. Let r and r' be two arbitrary vertices in $\mathcal{V}_m$ (or in $\tilde{\mathcal{V}}_m$). Then the maximum flow and the minimum cut between r and r' may be computed from a smaller graph with all vertices in $\tilde{\mathcal{V}}_m$ (or in $\mathcal{V}_m$) condensed into a single vertex $\tilde{v}_c$ (or v_c).

The condensation of $\tilde{\mathcal{V}}_m$ is such that for each vertex $v_i \in \mathcal{V}_m$ all arcs a_{ij} in $(\mathcal{V}_m \rightarrow \tilde{\mathcal{V}}_m)$ are replaced by a single arc $a'_{i,\tilde{v}_c}$ of capacity

$$c'_{i,\tilde{v}_c} = \sum_{v_j \in \tilde{\mathcal{V}}_m \cap \Gamma(v_i)} c_{ij}, \quad \forall v_i \in \mathcal{V}_m.$$

The implication of the theorem is that if $(\mathcal{V}_m \rightarrow \tilde{\mathcal{V}}_m)$ is a minimum cut separating s from t, then there exists a minimum cut for each pair of vertices in $\mathcal{V}_m$ (or in $\tilde{\mathcal{V}}_m$) which will not further partition $\tilde{\mathcal{V}}_m$ (or $\mathcal{V}_m$). This is crucial to the optimality of our clustering algorithm, and to the efficiency of the Gomory-Hu algorithm.

Theorem 3: Let $\mathcal{G}^* = (\mathcal{V}, \mathcal{A}^*)$ be a hypothetical complete graph with the capacity of arc $a^*_{ij} \in \mathcal{A}^*$ equal to the maximum flow $\overline{F}_{ij}$ between the corresponding vertices v_i and v_j of the original graph $\mathcal{G}$ and let $\mathcal{T}^*$ be a maximal spanning tree of $\mathcal{G}^*$. Then $\mathcal{T}^*$ is flow-equivalent to the original graph $\mathcal{G}$. Furthermore the flow-equivalent tree $\mathcal{T}^*$ constructed using the Gomory-Hu algorithm is also cut-equivalent to $\mathcal{G}$.

Cut-equivalency (flow-equivalency) means that $\mathcal{G}$ and $\mathcal{T}^*$ are indistinguishable as far as the minimum cut (maximum flow) between vertices is concerned. A proof of this theorem is given by Gomory and Hu [12]. A more comprehensive proof of Theorem 3 can be found in [19, ch.11] for flow-equivalency and in [20, ch.2] for cut-equivalency. From now on we will refer to this cut and flow equivalent tree $\mathcal{T}^*$ simply as an equivalent tree.

The construction of the equivalent tree $\mathcal{T}^*$ using the Gomory–Hu algorithm involves the successive solution of precisely $M - 1$ maximum flow problems with M being the number of vertices in $\mathcal{G}$. In addition, most of these maximum flow problems involve much smaller graphs due to vertex condensation. Once $\mathcal{T}^*$ has been constructed, the maximum flow between any pair of vertices s and t is equal to the minimum of the capacities among arcs on the unique path in $\mathcal{T}^*$ which leads from s to t. Any arc in that path with capacity equal to the maximum flow between s and t can be selected to form a minimum cut separating s and t.

B. Clustering Rationale

In this and the following section, we will assume that the data set can be adequately represented by an adjacency graph $\mathcal{G}$: each vertex corresponds to a data point (a component); an arc indicates a neighborhood relationship between the two linked components, and a similarity measure between these two is assigned to the arc as its flow capacity. In the case of image segmentation, one could construct such a graph in which each vertex represents a pixel in the image and arcs are placed between pairs of neighboring pixels.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be an adjacency graph, formed from the data set, with vertex set $\mathcal{V} = \{v_1, v_2, \cdots, v_M\}$ and arc set $\mathcal{A} = \{a_{ij}\}$ with a_{ij} denoting the arc between vertices v_i and v_j with capacity c_{ij}. Here c_{ij} is a similarity measure between v_i and v_j, *i.e.* the larger the arc capacity c_{ij}, the more similar are the vertices v_i and v_j. The purpose of a clustering algorithm is to group together the components into a reduced number of clusters. Each cluster contains components with similar features. This problem can be formulated equivalently in terms of the adjacency graph $\mathcal{G}$ formed from the components: divide $\mathcal{G}$ into a number of unconnected subgraphs by removal of the arcs joining the subgraphs. The set of vertices in each subgraph then represents a single cluster. In the case of image segmentation, each of the remaining subgraphs contains a set of connected vertices or components whose union represents a spatially connected region of the image. Recall that the arc capacities are defined as a measure of similarity between each component and its connected neighbors, i.e., the larger the arc capacity, the more similar are the two linked vertices. Suppose we wish to partition the graph $\mathcal{G}$ into two subgraphs with as dissimilar features as possible. A good criterion for choosing such a partition is to minimize the maximum flow between the two subgraphs. Following Theorem 1, such an optimal partition can be found by removing arcs in $\mathcal{G}$ corresponding to the cut set with the smallest cut value among all minimum cuts between every pair of vertices in $\mathcal{G}$. Note that any other cut would result in greater or equal maximum flow (and hence similarity) between the two subgraphs. Finding these minimum cuts in an undirected graph is a well known problem in graph theory and has been efficiently solved by Gomory and Hu [12] through the construction of an equivalent tree $\mathcal{T}^*$.

One can extend this principle to the subdivision of the graph $\mathcal{G}$ into K subgraphs. The objective here is to partition a graph $\mathcal{G}$ into K subgraphs where the largest inter-subgraph maximum flow is minimized among all possible K-partitions of $\mathcal{G}$. We will illustrate how to find this optimal K-partition for the case where $K = 3$. Let $(\mathcal{V}_m \rightarrow \tilde{\mathcal{V}}_m)$ be a minimum cut with the smallest cut value among all minimum cuts separating each pair of vertices of $\mathcal{G}$, and let r and r' be any other pair of vertices. Note that if r and r' lie on opposite sides of $(\mathcal{V}_m \rightarrow \tilde{\mathcal{V}}_m)$, then $(\mathcal{V}_m \rightarrow \tilde{\mathcal{V}}_m)$ has to be a minimum cut between them. Therefore in order to partition $\mathcal{G}$ further, we need only consider those pairs of vertices, r and r', such that both r and r' belong to either $\mathcal{V}_m$ or $\tilde{\mathcal{V}}_m$. If both r and r' are in $\mathcal{V}_m$ (or in $\tilde{\mathcal{V}}_m$), the minimum cut between r and r' can partition further only $\mathcal{V}_m$ (or $\tilde{\mathcal{V}}_m$) according to Theorem 2, which implies a partition of $\mathcal{G}$ into exactly three subgraphs. Hence, the optimal 3-partition is obtained by choosing the second cut with the smallest possible cut value.

A similar argument applies to cases with more clusters. Namely, the optimal K-partition of $\mathcal{G}$ is obtained by choosing the $K-1$ smallest minimum cuts from the set of all minimum cuts between each pair of vertices. From Theorem 3, the equivalent tree $\mathcal{T}^*$ has arcs with capacity equal to the maximum flow between the vertices. Thus one can easily remove arcs in $\mathcal{T}^*$ in order of increasing maximum flow to partition the original graph into any number of subgraphs. Equivalently, a K-partition of the graph using the method described above produces a segmentation of the image into the K regions which are most dissimilar (in the sense defined by the capacity function) among all K-region segmentations of the image. A proof of the optimality of this clustering procedure follows from Theorem 3, and will be given in the next subsection.

C. The Clustering Algorithm and Its Properties

Basic Clustering Algorithm:

Step 1) Form the adjacency graph $\mathcal{G}$ from the data set.
Step 2) Find an equivalent tree $\mathcal{T}^*$ for $\mathcal{G}$.
Step 3) Successively remove arcs in $\mathcal{T}^*$ in order of increasing arc capacity until a given stopping condition is met.

This clustering algorithm produces an optimal partition of the graph $\mathcal{G}$ into a given number of subgraphs, in the sense that the maximum of the inter-subgraph flows is minimized among all possible partitions of $\mathcal{G}$ into the same number of subgraphs.

Corollary 1 (Corollary to Theorem 3): The K-partition of an undirected graph $\mathcal{G}$, obtained by removing the K-1 arcs with the smallest capacity in the equivalent tree $\mathcal{T}^*$ of $\mathcal{G}$, minimizes the largest inter-subgraph maximum flow among all possible K-partitions of $\mathcal{G}$.

Proof of Corollary 1: Following Theorem 3, the maximum flow between any two subgraphs is equal to the minimum of the capacities among arcs on the unique path in $\mathcal{T}^*$ which connects the subgraphs. Along that path, at least one of the arcs must have been marked for removal in order to disconnect the two subgraphs. Hence the maximum flow between any subgraph pair cannot exceed the largest capacity of those $K-1$ marked arcs. But note also that between those subgraph pairs linked only by a single marked arc, the maximum flow must be equal to the capacity of that marked arc. Therefore we conclude that the largest inter-subgraph maximum flow is equal to the largest of the $K-1$ marked arc capacities, which is minimized when the $K-1$ arcs in $\mathcal{T}^*$ with the smallest capacity are marked for removal. □

Note also that in this optimal subgraph partition, the maximum flow between any pair of vertices in the same subgraph (intra-subgraph maximum flow) is always greater than or equal to the maximum flow between vertices in two different subgraphs (inter-subgraph maximum flow). A given undirected graph $\mathcal{G}$ may have more than one equivalent tree $\mathcal{T}^*$, but $\mathcal{T}^*$ will be unique if all of its arcs have distinct capacities [18, Ch. 4]. In the case of a unique $\mathcal{T}^*$ a unique optimal K-partition is indeed guaranteed. However the optimal K-partition may still be unique even if $\mathcal{T}^*$ is not.

Corollary 2 (Corollary to Theorem 3): The K-partition of an undirected graph $\mathcal{G}$, obtained by removing the $K-1$ arcs with the smallest capacity in the equivalent tree $\mathcal{T}^*$ of $\mathcal{G}$, is unique if all $K-1$ arcs in $\mathcal{T}^*$ marked for removal have capacity strictly less than the remaining unmarked arcs.

Proof: Let $\mathcal{F}_T$ be the smallest arc capacity of all unmarked arcs in $\mathcal{T}^*$. In other words, the maximum flow between any pair of vertices in the same subgraph is at least $\mathcal{F}_T$. However the maximum flow between any pair of vertices belonging to two different subgraphs is strictly less than $\mathcal{F}_T$ by assumption. Hence any other K-subgraph partition would have to result in at least one intra-subgraph maximum flow less than $\mathcal{F}_T$ and hence the optimal partition is unique. □

As the number of subgraphs increases, so does the number of corresponding regions in the segmented image. Since the goal of clustering is to produce a small number of regions from

TABLE I-A
GRAY SCALE INTENSITIES FOR THE REGIONS OR PATCHES IN FIG. 1

Patch	Intensity
1	255
2	176
3	112
4	120
5	112
6	120
7	176
8	255
9	184
10	144
11	136
12	80

TABLE I-B
VALUES OF THE CAPACITY FOR THE ARCS OF THE EQUIVALENT TREE SHOWN IN FIG. 1(d)

Order	Arc	Capacity
I	$a^*_{1,7}$	7.40×10^{-11}
II	$a^*_{8,10}$	1.56×10^{-10}
III	$a^*_{1,12}$	3.78×10^{-10}
IV	$a^*_{1,10}$	4.32×10^{-10}
V	$a^*_{5,12}$	7.56×10^{-5}
VI	$a^*_{2,10}$	1.12×10^{-4}
VII	$a^*_{3,4}$	3.1638
VIII	$a^*_{5,6}$	3.1638
IX	$a^*_{4,5}$	6.8555
X	$a^*_{2,9}$	11.073
XI	$a^*_{10,11}$	12.128

the image, a stopping condition should be used to prevent the formation of too many regions. This could be achieved either interactively by terminating the arc cutting procedure when sufficient regions have been formed, or by using a threshold so that arcs with a maximum flow above a given threshold are not cut. The problem of systematically choosing a suitable threshold is clearly important. Currently, we choose the threshold empirically.

D. A Clustering Example

A simple example is described here to illustrate how the clustering algorithm works for image segmentation. Fig. 1(a) shows a computer generated image of 128×128 pixels. Different intensities are assigned to the background, the 4 objects and their overlapped areas. The image is thus partioned into 12 connected regions (patches) of constant intensity (Fig. 1.b). The respective grey scale intensities for these patches are listed in Table I-A. Note that the smallest intensity difference between neighboring overlapped and non-overlapped areas is 8. An adjacency graph, $\mathcal{G}$, is formed from these patches, as shown in Fig. 1(c). Each vertex of the graph corresponds to a patch. An arc is placed between a pair of vertices if their corresponding patches are neighbors of each other, i.e., if they are spatially connected. A flow capacity is then assigned to each arc. The capacities may be any nonnegative symmetric function. For the purposes of this example, we choose these functions as a measure of similarity between the patches (vertices). In particular the following capacity function, with the control parameter $\sigma = 10$, is used in our example:

$$c_{i,j} = k_{i,j} \cdot e^{-(\frac{\mu_i - \mu_j}{\sigma})^2}, \qquad (2)$$

where $k_{i,j}$ denotes the number of neighboring pixels between vertices v_i and v_j, and μ_i and μ_j are the sample means of patches i and j. An equivalent tree $\mathcal{T}^*$ is then generated using the Gomory-Hu algorithm; $\mathcal{T}^*$ is shown in Fig. 1(d). The capacity values for all arcs of $\mathcal{T}^*$ are listed in increasing order in Table I-B.

Arcs are now removed, in order of increasing capacity, from the equivalent tree $\mathcal{T}^*$. The first arc to be removed is $a^*_{1,7}$, which results in a subtree consisting of the region 7 and another containing the rest of the image. Fig. 2(a) shows the resulting image partition with all pixels displayed as their

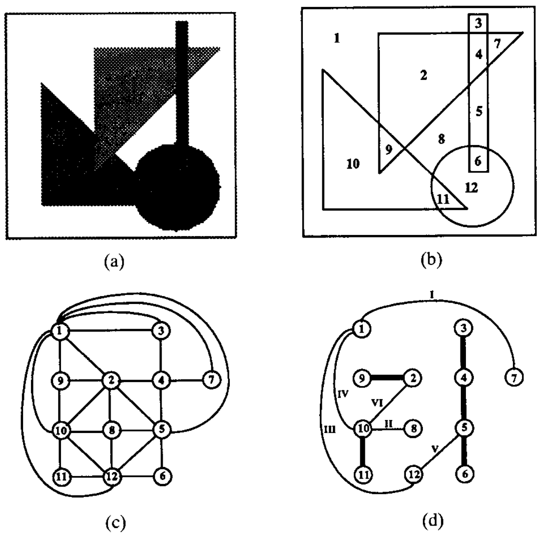


Fig. 1. The test image, its adjacency graph $\mathcal{G}$ and theequivalent tree $\mathcal{T}^*$.

respective subtree's mean intensity. Similarly the arcs $a^*_{8,10}$, $a^*_{1,12}$, $a^*_{1,10}$, $a^*_{5,12}$ and $a^*_{2,10}$ are removed to further partition the image (Fig. 2(b)–(f)). Since Fig. 2(f) correctly classifies neighboring regions with similar features (in our case the pixel intensity), no additional clusters are needed. The remaining arcs in $\mathcal{T}^*$ are shown in Fig. 1(d) with bold links. Note that regions 1 and 8 are not clustered together because they are not spatially connected. The same is true for regions 2 and 7. The final number of clusters is 7.

III. HIERARCHICAL IMPLEMENTATION

A direct implementation of the new graph theoretic clustering method is practical only for graphs of moderate size. The number of maximum flow problems involved in the Gomory–Hu algorithm increases linearly with the number of vertices in the graph. The time needed for solving those problems increases at a much faster rate. This is due to the fact that finding the maximum flow between two vertices has polynomial complexity with respect to the graph size (see [21] for complexities of various maximum flow algorithms). In our experiments on a Sun SPARCstation, our direct implementa-

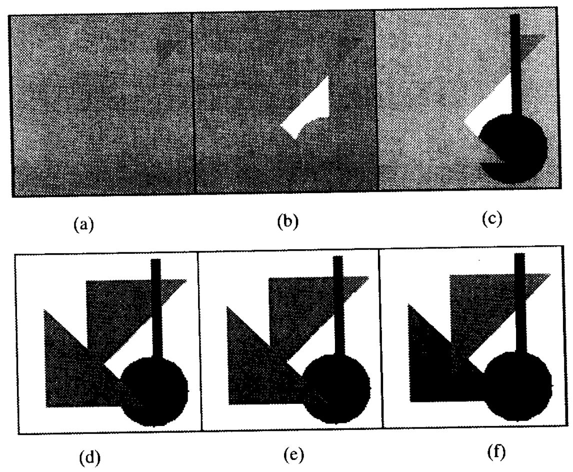


Fig. 2. Resulting image partitions at each step of patch clustering. All pixels are displayed as their respective subtree's meanintensity.This figure shows regions formed after removal of 1 (Fig. 2(a)), 2 (Fig. 2(b)), 3 (Fig. 2(c)), 4 (Fig. 2(d)), 5 (Fig. 2(e)), 6 (Fig. 2(f)) arcs of the spanning tree $\mathcal{T}^*$shown in Fig. 1(d).

tion of the clustering method could comfortably handle graphs with up to a couple of thousand vertices and a comparable number of arcs, i.e., computation times on the order of ten minutes. Unfortunately, a graph constructed from a 256 × 256 image, with each vertex containing only one pixel, will have 65 536 vertices.

The hierarchical implementation developed here is based on the observation that most of the minimum cuts found in the equivalent tree are never used since their values are sufficiently large that the arcs in those cuts will not be removed to form subgraphs. Consequently one may consider the problem of identifying these minimum cuts and condensing the vertices linked by them *before* constructing the equivalent tree. This would greatly reduce the size of the graph to which the Gomory–Hu algorithm is applied, without compromising the overall optimality of the clustering algorithm.

A. Extensions on Network Flow Theory

The maximum flow between any pair of vertices in a connected graph $\mathcal{G}$ is a global characteristic of $\mathcal{G}$. Its value must be computed using the complete graph or an appropriately condensed graph. The Gomory–Hu algorithm provides a way to construct a condensed graph, but this condensation process itself has to start with the complete graph. It would clearly be desirable to know if a subgraph may be condensed without processing the complete graph $\mathcal{G}$. The following new theorem and its corollary will provide sufficient conditions for condensation of $\mathcal{G}$ using arbitrary subgraphs of $\mathcal{G}$.

Theorem 4: Let $\mathcal{G}_o$ be a subgraph of an undirected graph $\mathcal{G}$, and $\overline{F}_{o,\min}$ be the minimum value of the maximum flows between all pairs of vertices in $\mathcal{G}_o$. Let s and t be an arbitrary pair of vertices in $\mathcal{G}$ such that the value of the maximum flow between s and t, $\overline{F}_{st}$, is smaller than $\overline{F}_{o,\min}$. Then $\overline{F}_{st}$ can be equivalently computed from the graph, $\mathcal{G}_c$, which is obtained by condensing all vertices of $\mathcal{G}_o$ into a single vertex.

Proof: Let us first show that the minimum cut separating s and t, $(\mathcal{V}_m \to \tilde{\mathcal{V}}_m)$, may not contain any arcs in $\mathcal{G}_o$. Assume the contrary to be true, i.e., there exists at least one arc $a_{rr'}$ in $(\mathcal{V}_m \to \tilde{\mathcal{V}}_m)$ such that both r and r' belong to $\mathcal{G}_o$. Then by definition, $(\mathcal{V}_m \to \tilde{\mathcal{V}}_m)$ is also a cut (not necessarily minimum) separating r and r'. However, its cut value, $\overline{F}_{st}$, is smaller than $\overline{F}_{rr'}$, the maximum flow between r and r', because $\overline{F}_{st} < \overline{F}_{o,\min}$ by assumption, and $\overline{F}_{o,\min} \leq \overline{F}_{rr'}$ by definition. This contradicts the definition of $\overline{F}_{rr'}$ as the value of the minimum cut between r and r'. Consequently, $(\mathcal{V}_m \to \tilde{\mathcal{V}}_m)$ cannot contain any arc in $\mathcal{G}_o$ and therefore $\mathcal{G}_o$ should be entirely contained by either $\mathcal{V}_m$ or $\tilde{\mathcal{V}}_m$. It then follows that condensing vertices in $\mathcal{G}_o$ will have no effect on the minimum cut $(\mathcal{V}_m \to \tilde{\mathcal{V}}_m)$ which separates s from t. □

Theorem 4 can be modified to provide a result which is more useful in developing the hierarchical clustering algorithm. This result is contained in the following corollary.

Corollary 3 Let $\mathcal{G}_o$ be a subgraph of an undirected graph $\mathcal{G}$. Let $\overline{F'}_{o,\min}$ be the minimum value of the maximum flows between all pairs of vertices in $\mathcal{G}_o$ computed using an arbitrary subgraph, $\mathcal{G}_o'$, which contains $\mathcal{G}_o$, and let s and t be an arbitrary pair of vertices in $\mathcal{G}$. If the value, $\overline{F}_{st}$, of the maximum flow between s and t, computed using $\mathcal{G}$, is smaller than $\overline{F'}_{o,\min}$, then $\overline{F}_{st}$ can be equivalently computed from the condensed graph $\mathcal{G}_c$ with all vertices in $\mathcal{G}_o$ condensed into a single vertex.

Before we proceed to prove the corollary, let us introduce the following lemma. The lemma establishes an important relationship between the maximum flows computed using the complete graph and those using a local subgraph. Proof of Corollary 3 then follows by combining this lemma with Theorem 4.

Lemma 1: Let $\mathcal{G}_o'$ be a subgraph of an undirected graph $\mathcal{G}$, and $\overline{F'}_{st}$ be the value of the maximum flow between a pair of vertices, s and t, in $\mathcal{G}_o'$ computed using $\mathcal{G}_o'$. Let $\overline{F}_{st}$ be the maximum flow between s and t computed using $\mathcal{G}$. Then

$$\overline{F'}_{st} \leq \overline{F}_{st}.$$

Proof: Let $(\mathcal{V}_m \to \tilde{\mathcal{V}}_m)$ be a minimum cut separating s and t computed in $\mathcal{G}$. The subset of $(\mathcal{V}_m \to \tilde{\mathcal{V}}_m)$ whose arcs are in $\mathcal{G}_o'$ is also a cut separating s from t in $\mathcal{G}_o'$. By definition, the value of this cut should be greater than or equal to $\overline{F'}_{st}$, the value of the minimum cut between s and t computed in $\mathcal{G}_o'$. □

Proof of Corollary 3: If $\overline{F}_{st} < \overline{F'}_{o,\min}$, then $\overline{F}_{st} \leq \overline{F}_{o,\min}$, since $\overline{F'}_{o,\min} \leq \overline{F}_{o,\min}$ from Lemma 1. Therefore, the conditions for Theorem 4 are satisfied. □

The implication of Corollary 3 is significant since the condition under which a subgraph, $\mathcal{G}_o$, may be condensed can be checked using only a local subgraph which contains $\mathcal{G}_o$. Using this result a hierarchical clustering method may be developed which uses subgraphs for condensation (local processing) and the Gomory–Hu algorithm for optimal clustering applied to the condensed graph (global processing). The theoretic foundation of such a clustering technique is presented below as a theorem.

Theorem 5: Let $\mathcal{G}_o$ be a subgraph of an undirected graph $\mathcal{G}$, and $\overline{F'}_{o,\min}$ be the minimum value of the maximum flows between all pairs of vertices in $\mathcal{G}_o$ computed in a subgraph $\mathcal{G}_o'$ which contains $\mathcal{G}_o$. Let $\mathcal{G}_c$ be the graph formed from $\mathcal{G}$ with all vertices in $\mathcal{G}_o$ condensed into a single vertex and let $\mathcal{T}_c^*$ be an equivalent tree constructed from $\mathcal{G}_c$. Then $\mathcal{T}_c^*$ is a partially cut and flow equivalent tree of $\mathcal{G}$, at threshold $\overline{F'}_{o,\min}$, in the sense that the maximum flow, $\overline{F}_{st}$, and the corresponding minimum cut can be computed from $\mathcal{T}_c^*$ for all pairs of vertices s and t in $\mathcal{G}$ such that $\overline{F}_{st} < \overline{F'}_{o,\min}$. Furthermore for all vertex pairs, s and t, in $\mathcal{G}$ with flow $\overline{F}_{st} \geq \overline{F'}_{o,\min}$, the maximum flow value computed in $\mathcal{T}_c$ is also greater than or equal to $\overline{F'}_{o,\min}$.

Proof: Let s and t be an arbitrary pair of vertices in $\mathcal{G}$, and assume the maximum flow between s and t, $\overline{F}_{st}$, is less than $\overline{F'}_{o,\min}$. From Corollary 3, $\overline{F}_{st}$ and the corresponding minimum cut separating s and t may be computed from $\mathcal{G}_c$. Since $\mathcal{G}_c$ and $\mathcal{T}_c^*$ are flow and cut equivalent, the maximum flow $\overline{F}_{st}$ and the minimum cut can also be computed from $\mathcal{T}_c^*$. Now assume that $\overline{F}_{st} \geq \overline{F'}_{o,\min}$. From Lemma 1, the maximum flow computed in $\mathcal{G}_c$ is at least as large as $\overline{F}_{st}$. □

So far, we have derived sufficient conditions for subgraph condensation when we are only interested in finding minimum cuts with small cut values. In the remainder of this subsection, we will derive additional conditions for subgraph condensation with no cut value constraints attached.

Let $\mathcal{G}_o'$ be a subgraph of an undirected graph $\mathcal{G}$ and $\mathcal{T}_o'^*$ be an equivalent tree computed from $\mathcal{G}_o'$. We wish to construct an equivalent tree $\mathcal{T}^*$ for $\mathcal{G}$ using as much information from $\mathcal{T}_o'^*$ as possible. Specifically we want to identify those minimum cuts in $\mathcal{T}_o'^*$ which are also minimum cuts in $\mathcal{T}^*$. If successful, we can initialize the Gomory–Hu algorithm at a much more advanced stage, since these identified minimum cuts need not be recomputed, and thus greatly reduce the computation time.

Definition: An interior vertex of a subgraph $\mathcal{G}_o'$ of $\mathcal{G}$ is a vertex such that all of its neighboring vertices also belong to $\mathcal{G}_o'$. A non-interior vertex is called a boundary vertex.

Theorem 6: Let $\mathcal{T}_o'^*$ be an equivalent tree of $\mathcal{G}_o'$, a subgraph of an undirected graph $\mathcal{G}$, and $\mathcal{T}_o^*$ be a branch of $\mathcal{T}_o'^*$ consisting exclusively of interior vertices of $\mathcal{G}_o'$. Let $\mathcal{G}_c$ be the graph formed from $\mathcal{G}$ with all vertices in $\mathcal{T}_o^*$ condensed into a single vertex v_o^*. The equivalent tree $\mathcal{T}^*$ of $\mathcal{G}$ can be constructed as follows: construct an equivalent tree $\mathcal{T}_c^*$ for the condensed graph $\mathcal{G}_c$, and then replace the vertex v_o^* by $\mathcal{T}_o^*$ connected at its root vertex.

Here, the root vertex of the branch $\mathcal{T}_o^*$ refers to the only vertex in $\mathcal{T}_o^*$ which neighbors a boundary vertex in $\mathcal{T}_o'^*$. The proof of Theorem 6 is based on the Gomory–Hu algorithm and its optimality.

Proof: Let s denote the root vertex of the branch $\mathcal{T}_o^*$, and t denote the parent vertex of s in $\mathcal{T}_o'^*$. Since all vertices in $\mathcal{T}_o^*$ are interior vertices, the minimum cut (in $\mathcal{G}_o'$) separating s and t, which is equivalent to the arc $a_{st}'^*$ in $\mathcal{T}_o'^*$, should also separate s and t in the complete graph $\mathcal{G}$. Denote this minimum cut as $(\mathcal{G}_o \rightarrow \tilde{\mathcal{G}}_o)$, where $\tilde{\mathcal{G}}_o$ is the subgraph of $\mathcal{G}$ which contains all vertices not in $\mathcal{G}_o$.

Apply the Gomory–Hu algorithm to $\mathcal{G}$ as follows. Choose s and t as the first vertex pair between which the maximum flow will be computed. Since $(\mathcal{G}_o \rightarrow \tilde{\mathcal{G}}_o)$ is a minimum cut separating s and t, the resulting tree at this stage should be two condensed vertices v_o^* and $\tilde{v}_o^*$ linked by an arc with capacity equal to the cut value of $(\mathcal{G}_o \rightarrow \tilde{\mathcal{G}}_o)$, where v_o^* and $\tilde{v}_o^*$ are condensed from $\mathcal{G}_o$ and $\tilde{\mathcal{G}}_o$, respectively. Now proceed with the algorithm by selecting only vertex pairs in $\tilde{\mathcal{G}}_o$ until every condensed vertex except v_o^* has become a single vertex of $\mathcal{G}$. Note that we have just constructed the equivalent tree $\mathcal{T}_c^*$ for the graph $\mathcal{G}_c$, which is obtained from $\mathcal{G}$ by condensing all the vertices in $\mathcal{G}_o$ into a single vertex v_o^*.

Finally, we need to compute the maximum flows between all vertex pairs in $\mathcal{G}_o$. But the resulting maximum flows and the minimum cuts should be identical to those computed using the subgraph $\mathcal{G}_o'$, because all the vertices in $\mathcal{T}_o'^*$ are interior vertices of $\mathcal{G}_o'$. Therefore in order to complete the equivalent tree $\mathcal{T}^*$ of $\mathcal{G}$, the condensed vertex v_o^* can be expanded in exactly the same manner as when constructing $\mathcal{T}_o'^*$ from $\mathcal{G}_o'$. □

B. A Clustering Algorithm Using Hierarchical Implementation

In order to perform clustering of the vertices of a graph $\mathcal{G}$ to minimize the largest inter-subgraph maximum flow, it is not necessary to know the exact arc capacities (minimum cut values in $\mathcal{G}$) for all arcs in the equivalent tree $\mathcal{T}^*$. This is because only those arcs with small capacities will be removed during the clustering process. In other words, if an arc capacity is greater than a given threshold, $\mathcal{F}_T$, we are no longer interested in its exact value since the arc will not be cut. The value of this threshold can be chosen as any value greater than the largest minimum cut which will be made in partitioning the graph into its subgraphs. Using this approach and Theorem 5, a partially equivalent tree can be constructed more efficiently than $\mathcal{T}^*$. This produces a saving in computation time for the overall clustering process without any loss in optimality. In this subsection we develop an efficient clustering technique based on hierarchically constructing a partially equivalent tree for the graph $\mathcal{G}$.

Consider an arbitrary subgraph, $\mathcal{G}_o'$, of an undirected graph $\mathcal{G}$, and let $\mathcal{T}_o'^*$ be an equivalent tree constructed using $\mathcal{G}_o'$. If the vertices in $\mathcal{G}_o'$ are clustered by removing all the arcs corresponding to cuts whose capacities are smaller than $\mathcal{F}_T$, then the maximum flow (computed in $\mathcal{G}_o'$) between any pair of vertices belonging to the same cluster must be at least $\mathcal{F}_T$. From Theorem 5, the equivalent tree $\mathcal{T}_c^*$, computed from the graph with each of the clusters in $\mathcal{G}_o'$ condensed into a single vertex, can be used in place of $\mathcal{T}^*$ for clustering vertices in $\mathcal{G}$. Here $\mathcal{T}_c^*$ is referred to as a *partially* equivalent tree of $\mathcal{G}$, because $\mathcal{T}_c^*$ is cut and flow equivalent to $\mathcal{G}$ only between vertex pairs with maximum flow values smaller than $\mathcal{F}_T$.

The construction of this partially equivalent tree $\mathcal{T}_c^*$ can also make full use of the minimum cuts in $\mathcal{T}_o'^*$ that enclose interior vertices of the subgraph $\mathcal{G}_o'$. A branch of $\mathcal{T}_o'^*$ that contains only interior vertices is referred to here as an interior branch of $\mathcal{T}_o'^*$. A maximal interior branch of $\mathcal{T}_o'^*$ is then an interior branch that cannot be contained by any other interior branch of $\mathcal{T}_o'^*$. From Theorem 6, all vertices in any maximal branch of $\mathcal{T}_o'^*$ can be condensed into a single vertex when computing $\mathcal{T}_c^*$. Furthermore the minimum cut between these condensed vertices and the rest of $\mathcal{T}_c^*$ is identical to that computed in

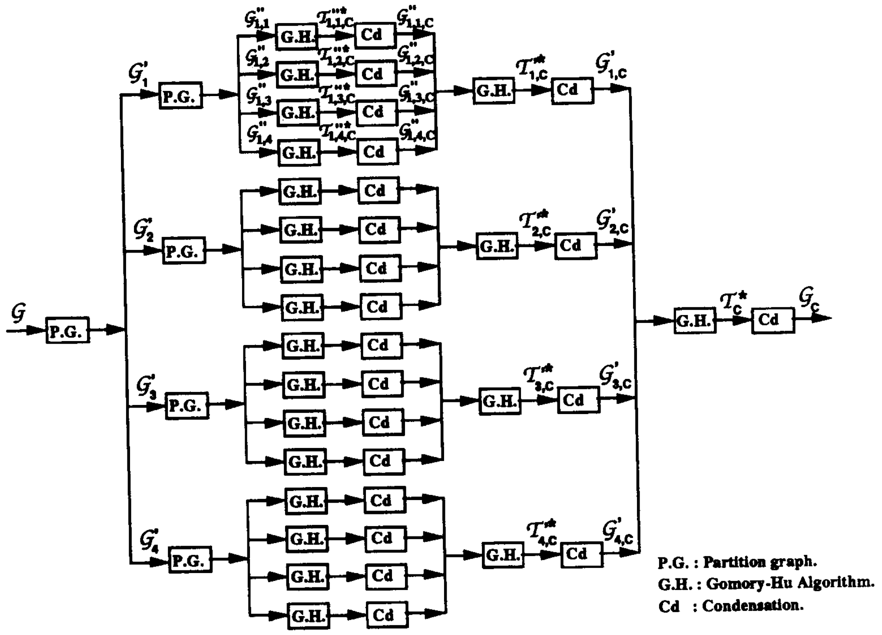


Fig. 3. Block diagram for the construction of a partially equivalent tree using a three level hierarchy.

$\mathcal{T}_o'^*$. Hence, the Gomory–Hu algorithm may be initialized using the following tree: every maximal interior branch of $\mathcal{T}_o'^*$ corresponds to a condensed vertex in this tree, and each is linked to a condensed vertex t_0 that contains the rest of the vertices of $\mathcal{G}_c$. The capacity of each arc in this tree is equal to the value of the minimum cut between the corresponding interior branch and the rest of the graph. The Gomory–Hu algorithm is only used to expand the vertex t_0. The rest of the condensed vertices are expanded using the interior branches of $\mathcal{T}_o'^*$. Note that not only is the Gomory–Hu algorithm applied to a graph of reduced size due to the condensation of the maximal interior branches of $\mathcal{T}_o'^*$, but it is also initialized to solve only as many maximum flow problems as the number of vertices in t_0 minus one. Obviously this same interior branch condensation is also applicable to $\mathcal{T}_{o,c}'^*$, the condensed tree of $\mathcal{T}_o'^*$ obtained by condensing each cluster of vertices linked by arcs with capacity $\geq \mathcal{F}_T$.

This condensation procedure can be applied in a hierarchical fashion, i.e., $\mathcal{G}$ may be partioned into a large number of smaller subgraphs with the above procedure applied to each. The resulting condensed subgraphs are then combined in groups of two or more and the process repeated in a hierarchical fashion as illustrated in Fig. 3. For a clustering procedure in which the maximum inter-subgraph flow is constrained to be less than $\mathcal{F}_T$, the result obtained using this hierarchical approach is identical to that which would be obtained by applying the Gomory-Hu algorithm directly to the complete graph. The computational advantage is significant—in our experiments CPU times for the direct Gomory–Hu algorithm in excess of 12 hours are reduced to the order of 10 minutes using the hierarchical approach.

Clustering Algorithm Using Hierarchical Implementation:

Step 1) Form the adjacency graph $\mathcal{G}$, and then arbitrarily partition $\mathcal{G}$ into a number of small subgraphs, $\{\mathcal{G}'_{0,m}\}$, usually of similar size. Let $i = 0$.

Step 2) For each subgraph $\mathcal{G}'_{i,m}$ find an equivalent tree $\mathcal{T}'^*_{i,m}$ using the Gomory–Hu algorithm.

Step 3) Permanently condense all the subtrees in $\mathcal{T}'^*_{i,m}$ which are linked by arcs with capacities $\geq \mathcal{F}_T$. Also temporarily condense all the maximal interior branches of this resulting tree to form a condensed subgraph $\mathcal{G}'_{i,m,c}$.

Step 4) If there is only one subgraph in $\{\mathcal{G}'_{i,m,c}\}$, then denote the resulting equivalent tree as $\mathcal{T}_c'^*$ and go to Step 6; otherwise continue to Step 5.

Step 5) Group $\{\mathcal{G}'_{i,m,c}\}$ into subsets, each containing anumber of neighboring subgraphs of $\{\mathcal{G}'_{i,m,c}\}$. By connecting the subgraphs in each subset using the inter-subgraph arcs, we form a new subgraph set $\{\mathcal{G}'_{i+1,m}\}$. Let $i = i + 1$, and go to Step 2.

Step 6) Expand back all the temporarily condensed vertices in $\mathcal{T}_c'^*$ to construct the partially equivalent tree $\mathcal{T}_c^*$ of the original graph $\mathcal{G}$.

Step 7) Remove successively arcs in $\mathcal{T}_c^*$ in order of increasing arc capacity until a given stopping condition is met. A clustering constraint may be attached in this last step.

Since constructing an equivalent tree for graphs with few vertices is very fast, it is usually more efficient to start the algorithm with a large number of very small subgraphs. The only restriction in selecting the threshold $\mathcal{F}_T$ is that no minimum cut with value larger than $\mathcal{F}_T$ may be used for partitioning the graph $\mathcal{G}$. Ideally the threshold $\mathcal{F}_T$ should be chosen as the value of the largest minimum cut which will be made in partitioning $\mathcal{G}$ so that the clustering is most efficient. In practice we can choose any reasonable value for $\mathcal{F}_T$ because the additional cost of having selected a larger $\mathcal{F}_T$ is very small compared to the overall saving of the hierarchical implementation.

C. Incorporation of Constraints

So far, we have only discussed unconstrained data clustering. Often we also want the resulting data clusters to possess certain desirable characteristics, e.g., clusters should not be too small.

A direct approach to incorporating these requirements is to impose "hard" constraints on the clustering formulation as follows: find the K-subgraph partition that minimizes the largest inter-subgraph maximum flow, while each of the K subgraphs meets the specified requirements. Unfortunately, the optimal solution would have to be chosen from a possible $C_{M_c-1}^{K-1}$ different combinations, where M_c is the number of vertices in $\mathcal{T}_c^*$. Branch-and-bound algorithms should be a suitable choice for solving this problem.

Alternatively, we may impose constraints on the minimum cuts which are used to partition the graph $\mathcal{G}$. For example, only minimum cuts containing at least a given number of arcs in the original graph may be selected for graph partitioning. For this type of constraint, the optimal solution can still be found in a sequential fashion: remove the arcs of $\mathcal{T}^*$ in order of increasing capacity unless the corresponding cut set in $\mathcal{G}$ does not satisfy the constraint, in which case no action takes place. This process is repeated until $K-1$ arcs have been removed. The resulting solution is locally optimal, since the equivalent tree of $\mathcal{G}$ may not be unique. However if either all or none of the arcs in $\mathcal{T}_c^*$ with the same capacity are considered for removal, the solution becomes globally optimal following similar arguments used to prove Corollary 2.

Another approach is to select arc capacities in the adjacency graph $\mathcal{G}$ in such a way that the minimum cuts become less likely to result in undesirable subgraphs. When the constraints are not strict, this penalty approach may be a preferred one. In addition, we may also want to give special treatment to those clusters that do not meet the constraints. For example, when segmenting images we do not want to form many small regions. In practice, we find that these small regions often correspond to boundary pixels between large objects. Therefore we may simply choose to allow these small regions to be formed in the clustering stage and then merge them into large neighboring regions in a refining step.

IV. Image Segmentation Based on Clustering

In this section, we will demonstrate how we can apply the graph theory based clustering algorithm to the image segmentation problem. The segmentation is achieved by searching for closed contours of edge elements. These edge elements are computed directly from the image prior to construction of the graph.

Let the image be represented by an undirected graph $\mathcal{G}$: each pixel corresponds to a vertex of $\mathcal{G}$, and an arc is placed between two vertices if their associated pixels are longitudinal or vertical neighbors of each other. In this paper, only a first order neighborhood system is used although there is no inherent restriction on using a higher order neighborhood. For each pair of neighboring pixels, we define an edge element as a separating element between the pixel pair in a similar manner to the line processes used in Markov random field models [22]. The edge strength is computed as a function of intensity differences between the pixel pair and between pixels in their vicinity, by using masks similar to those shown in Fig. 4. Let $x_{i,j}$ be the grey scale intensity of the pixel at coordinate (i,j), and let $D_{i,j}^H$ and $D_{i,j}^V$ be the edge elements defined between the pixel pairs $(x_{i,j}, x_{i,j+1})$ and $(x_{i,j}, x_{i+1,j})$ respectively, then

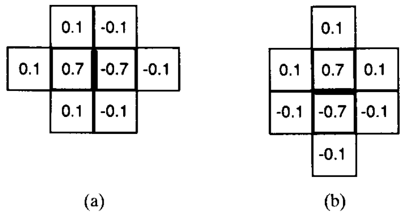


Fig. 4. Masks for computing the edge strength: (a) edge between horizontal neighboring pixels; (b) edge between vertical neighboring pixels.

$$D_{i,j}^H = \Big|\delta \cdot (x_{i,j} - x_{i,j+1}) + (x_{i-1,j} - x_{i-1,j+1}) + (x_{i+1,j} - x_{i+1,j+1}) + (x_{i,j-1} - x_{i,j+2})\Big/(\delta + 3) \cdot \sigma\Big|, \tag{3}$$

$$D_{i,j}^V = \Big|\delta \cdot (x_{i,j} - x_{i+1,j}) + (x_{i,j-1} - x_{i+1,j-1}) + (x_{i,j+1} - x_{i+1,j+1}) + (x_{i-1,j} - x_{i+2,j})\Big/(\delta + 3) \cdot \sigma\Big|, \tag{4}$$

where δ and σ are control parameters. The parameter δ controls the smoothing effect of the edge masks. For the MR brain images used in our experiments, we have observed that $\delta = 7$ seems to be a good compromise between reducing noise sensitivity and preventing thick edge lines. For noiser images, $\delta = 1$ and/or larger edge masks may be necessary. The parameter σ is proportional to the smallest intensity difference that indicates the potential presence of a region boundary in the image.

For an arbitrary cut in the graph $\mathcal{G}$ which partitions $\mathcal{G}$ into 2 subgraphs, the edge elements corresponding to the arcs belonging to the cut form a closed edge contour. The value of a cut is equal to the sum of the capacities of its arcs. Hence, if a small arc capacity is assigned to a strong edge element and *vice versa*, then the cuts with small value correspond to closed contours which contain strong edge elements. Conversely, isolated strong edges will not produce boundaries in the segmented image, since there is a high cost associated with the inclusion of weak edges (large arc capacities) necessary to form a closed boundary through these isolated edges. Ideally, the capacity function for a cut should not penalize long boundaries, but rather penalize the presence of gaps in the boundaries. This is particularly important in the segmentation example for an MR image of the brain presented next. In that case, the boundary between the cortex and CSF and between the cortex and the white matter is highly convoluted. Previously we used the negative exponential of the square of the edge strength to define the arc capacity [23]. Although the results have generally been satisfactory, we note that the capacity function drops to zero too fast as the edge strength increases, which results in indistinguishable capacities (10^{-12} is used as the lower bound for arc capacities to prevent underflow). Consequently isolated regions with very few pixels are more likely to occur when clustering pixels. Of course this problem may be alleviated by increasing the value of the control parameter σ. However a small value for σ is necessary to make a clear distinction between strong and weak edges. Based on our experiments, we found the following arc capacity function to be a more appropriate choice. Let $v_{i,j}$ be the vertex corresponding to pixel $x_{i,j}$, then the capacities for the arcs connecting $v_{i,j}$ to $v_{i,j+1}$ and to $v_{i+1,j}$, $C^H_{i,j}$ and $C^V_{i,j}$ respectively, are defined as

$$C^p_{i,j} = \begin{cases} e^{-(D^p_{i,j})^2}, & \text{if } 3 > D^p_{i,j}, \\ e^{-3D^p_{i,j}}, & \text{if } D^p_{i,j} \geq 3, \end{cases} \quad p = H \text{ or } V. \qquad (5)$$

The arc capacity function defined in this section is simple and often robust. Its simplicity helps to illustrate the idea of using clustering to segment images via closed edge contour finding, without the need for much additional knowledge. At the same time we also acknowledge the *ad hoc* nature of the definitions of edge strength and arc capacity. Attempts have been made to alleviate this weakness. In [24] we incorporate prior information about the regions in the image and then use the likelihood ratio between "no-edge" and "edge" to define the arc capacity. However, the rather lengthy description required prevents us from including it in this paper, where our emphasis is on presenting a novel data clustering strategy and its efficient implementation.

V. Experimental Results

A. Edge Based Segmentation of a MR Image

Fig. 5 shows a typical cross-sectional MR image of the brain for a patient with a large lesion on the left side. Several other edge based segmentation techniques have been tested for the image in Fig. 5. Among them, region finding using the Marr–Hildreth operator is the most promising alternative to the method described in Section IV. By trial and error, the best result was achieved using a Marr-Hildreth operator with $\sigma = 1.35$, implemented using a 9×9 window. Fig. 6(a) shows the filtered image displayed in black (negative pixels) and white (positive pixels). All connected regions in Fig. 6(a) are then extracted, and the three largest internal regions found using this method are shown in Fig. 6(b)–(d). While this method is effective at finding certain boundaries, it also mislocates many other boundaries. The most noticeable problem is that one must break connections between collections of pixels with distinct features. For example, in Fig. 6(b) pixels corresponding to white matter and the ventricles are mistakenly linked; these connections must be broken for accurate segmentation and labeling. This problem is mainly due to an intrinsic drawback of techniques based on zero crossings. All closed contours have to be shared by exactly two connected regions and thus more complex structures, where three or more regions are connected, will be segmented into at most two regions.

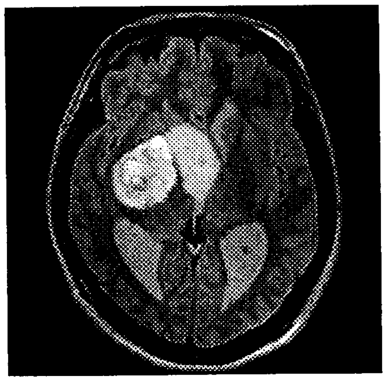


Fig. 5. An axial MR image of a brain with white matter lesions.

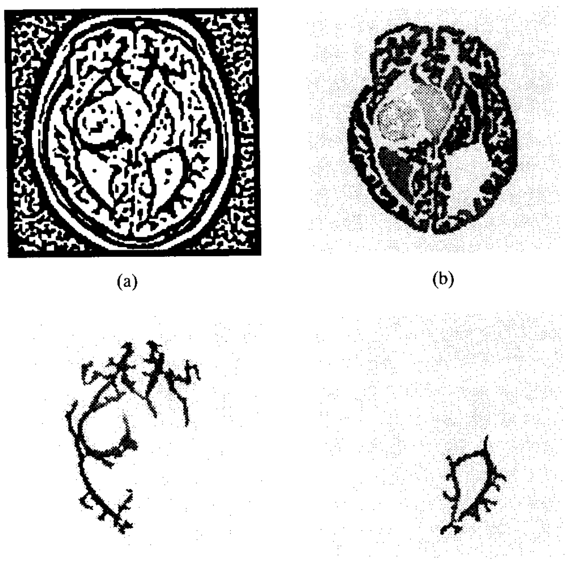


Fig. 6. (a) Regions from the Marr–Hildreth operator;(b)–(d) three large connected regions from (a) after region growing.

The same image, Fig. 5, was also segmented using the new graph theoretic segmentation algorithm. An adjacency graph $\mathcal{G}$ was constructed for the image in Fig. 5, as described in Section IV. The edge elements were computed using (3) and (4) with control parameter $\sigma = 3.0$ (Fig. 7(a)). Since there are almost twice as many edge elements as there are pixels, the following convention is used to display the edge image in Fig. 7(a). At each pixel site (i, j), the maximum of $D^H_{i,j}$ and $D^V_{i,j}$ is displayed. The capacity function for the arcs in $\mathcal{G}$ is defined according to (5). A partially equivalent tree $\mathcal{T}^*_c$ of $\mathcal{G}$ is then constructed at threshold $\mathcal{F}_T = 0.1$. Clusters of pixels are formed by removing all arcs in $\mathcal{T}^*_c$ with capacity smaller than $\mathcal{F}_T$. When no constraint is attached to these clusters, there may be a large number with very few pixels. In our experiment, the minimum size for a cluster is restricted to five pixels. All clusters with less than five pixels, except those clusters connected with other clusters in $\mathcal{T}_c^*$ through arcs with capacities much smaller than $\mathcal{F}_T$, are merged to the most similar neighboring cluster which has more than five pixels, using the merging procedure described in [25]. For the type of images shown here, that procedure is equivalent to region growing from the larger regions ($\geq$ 5 pixels) into their neighboring small clusters based on the difference between the mean intensity of the regions. Unmerged clusters that have less than five pixels will be labeled as unclassified. The algorithm produced a total of 161 clusters (connected regions) excluding the unclassified clusters. Fig. 7(b) shows the segmented image with the pixels in each cluster replaced by their sample mean and Fig. 7(c) shows the unclassified pixels which are also displayed in black within the image of Fig. 7(b). Using the equivalent tree of the adjacency graph of the segmented image, the skull and the external tissue can be easily removed from the image. The remaining 79 clusters correspond to brain tissue (Fig. 7(d)). Eight of these clusters are shown in Fig. 8. Fig. 7(e) shows the edge image associated with the 79 pixel clusters, where the grey level of edge elements reflects their strength. To obtain the final tissue classification, these pixel clusters are further labeled interactively into one of five types. Figs. 9(a)–(d) show the segmented regions corresponding respectively to (a) grey matter; (b) white matter; (c) ventricle; and (d) tumor. In each figure, only pixels classified to that label are displayed while the rest are set to white. The clusters that do not fit in any of the four tissue types are labeled as the null type, "unclassified." The union of the four regions is shown in Fig. 7(f), with the black spots within the image indicating unclassified pixels.

By removing fewer arcs from the partially equivalent tree $\mathcal{T}_c^*$, we could obtain fewer than 79 clusters. However, we have found that this results in the clustering, and hence incorrect labeling, of inhomogeneous regions. We believe that this problem reflects the limited nature of the information present in an MR image, i.e. without providing any form of supervision to the algorithm (the algorithm does not "know" that the image is an MR scan), it is conceivably not possible to obtain an accurate segmentation and clustering into only four regions (tumor, white matter, grey matter and ventricles). The results shown in Fig. 9, were obtained by interactively labeling the 79 regions found after clustering. Rather than attempting to force the current algorithm to form only the desired four regions, our goal is to use the results of the patch clustering algorithm as the input to a tissue labeling algorithm which compares the features of the regions, such as size, shape, location and spatial relationship with the other regions, with a stored brain map and label the regions accordingly.

B. Edge Detection of An Airport Image

The edge based segmentation algorithm has also been applied to edge detection for aerial photographs. Fig. 10.a shows a photograph of an airport, and Fig. 10(b) shows its edge image computed using (3) and (4). An adjacency graph is constructed using the arc capacity defined in (5) with $\sigma = 3.5$, and a partially equivalent tree is then computed at threshold

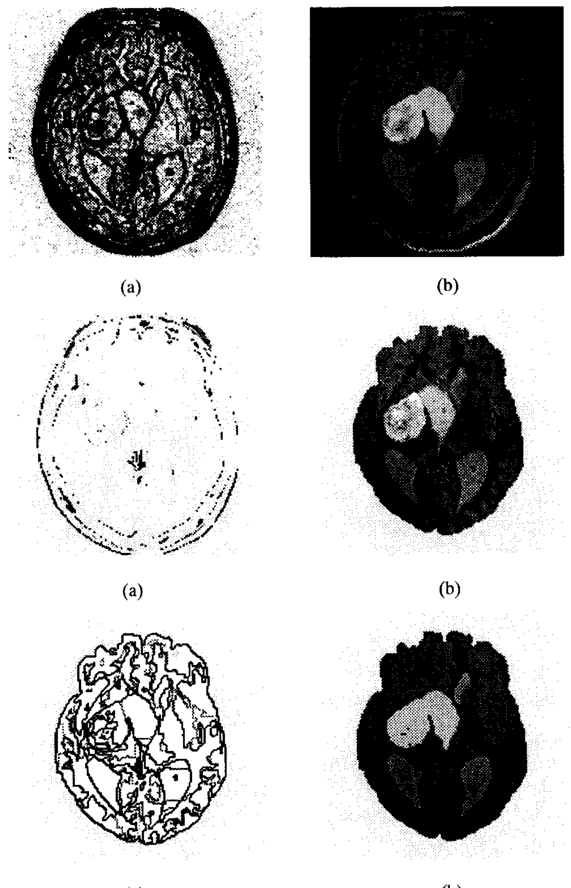


Fig. 7. Graph theoretic clustering results: (a) edges before clustering; (b) segmented image with each pixel cluster displayed in its mean; (c) unclassified pixels; (d) pixels corresponding to brain tissue; (e) edges after clustering; (f) combined tissue map.

$\mathcal{F}_T = 0.1$. Clusters of pixels are formed by removing all arcs in $\mathcal{T}_c^*$ in order of increasing arc capacity with capacity smaller than $\mathcal{F}_T$, unless doing so will result in a cluster with less than 5 pixels. This procedure produces a total of 321 clusters. These are shown in Fig. 10(c) with pixels in each cluster displayed in the cluster's sample mean intensity. The edge image corresponding to Fig. 10(c) is shown in Fig. 10(d). Again the edge images only display the maximum of $D_{i,j}^H$ and $D_{i,j}^V$ at each pixel site (i, j).

VI. Conclusion

We have presented a novel graph theoretic approach for data clustering, and demonstrated an application to the problem of image segmentation. We have also developed a fast algorithm for computing the maximum flows in an undirected graph, which is much more efficient than the Gomory–Hu method.

The data to be clustered are represented by an undirected adjacency graph $\mathcal{G}$. A flow capacity is assigned to each arc in $\mathcal{G}$ to reflect the similarity between the neighboring data points linked by that arc. Given $\mathcal{G}$, clustering is achieved by removing arcs corresponding to minimum cuts between

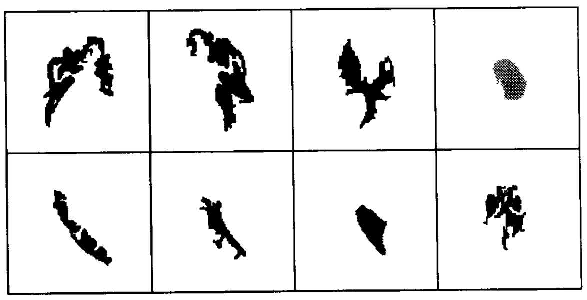


Fig. 8. Examples of regions (pixel clusters) formed by linking edge elements using clustering.

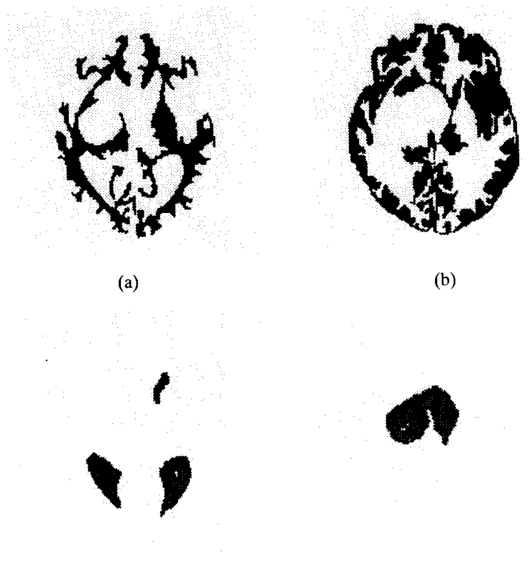


Fig. 9. The final tissue classification after interactive labeling of regions found using edge linking. (a) White matter. (b) Gray matter. (c) Ventricles. (d) Tumor.

vertex pairs in $\mathcal{G}$ to form mutually exclusive subgraphs. For an unconstrained optimal K-subgraph partition, the arcs selected for removal are those corresponding to the $K-1$ minimum cuts with the smallest $K-1$ cut values among all minimum cuts between every pair of vertices in $\mathcal{G}$. The resulting K-partition minimizes the largest inter-subgraph maximum flow among all possible K-partitions of $\mathcal{G}$, hence minimizing the similarity between the subgraphs (clusters). At the same time, this procedure also results in a nested sequence of partitions which are optimal for cluster numbers ranging from 2 to K, respectively. This is a desirable characteristic of a clustering technique, especially when the cluster number has to be determined from the data.

The minimum cuts of $\mathcal{G}$ are computed from a flow and cut equivalent tree $\mathcal{T}^*$ of $\mathcal{G}$, which can be constructed using the Gomory-Hu algorithm. Given $\mathcal{T}^*$, the optimal K-partition can be equivalently obtained by disconnecting the $K-1$ arcs in $\mathcal{T}^*$ with the smallest $K-1$ arc capacities. Based on the observation that most of the minimum cuts found in $\mathcal{T}^*$ are

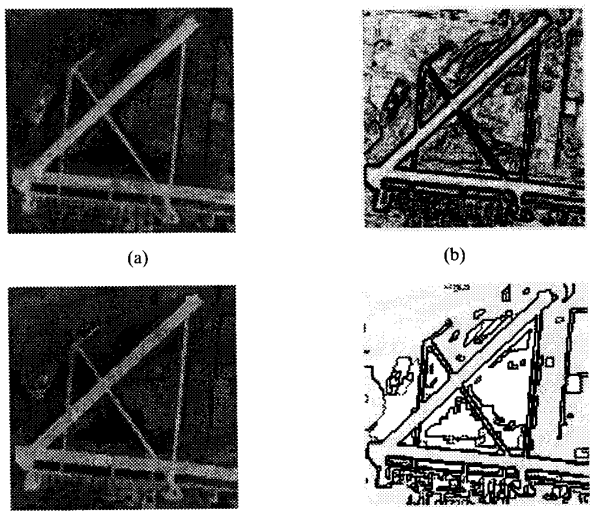


Fig. 10. Results after clustering: (a) original airport image; (b) computed edges of the airport image before clustering; (c) segmented airport image; (d) edges after clustering.

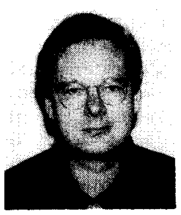
never used, since their values are sufficiently large that the arcs in those cuts will not be removed to form subgraphs, a fast hierarchical algorithm has been developed which constructs only a partially equivalent tree $\mathcal{T}_c^*$ of greatly reduced size. New theorems for subgraph condensation have been derived. Using these theorems, many of the minimum cuts with large values can be identified using local subgraphs, hence the vertices linked by them can be condensed *before* constructing the equivalent tree. Consequently the Gomory-Hu algorithm is applied to graphs of much smaller size without compromising the overall optimality of the clustering algorithm.

This graph theoretic clustering algorithm has been applied to the problem of image segmentation. Segmentation is achieved through seeking closed contours of edge elements of the image. By properly assigning arc capacities as a function of the computed edge strength, a minimum cut with small cut value, found in the adjacency graph formed by the image pixels, should contain mostly strong edges. We have observed that the proposed segmentation approach is able to accurately locate region boundaries while guaranteeing the formation of closed edge contours. The segmentation is unsupervised. This new method also offers the flexibility for incorporating prior information about the image through the arc capacity function. For example when segmenting MR brain images, the pixel intensity ranges for different tissues and their relative position within the brain may be reflected in the capacity function to provide improved segmentation results.

References

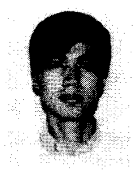
[1] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data.* Englewood Cliffs, NJ: Prentice Hall, 1988.
[2] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.
[3] L. J. Hubert, "Some applications of graph theory to clustering," *Psychometrika*, vol. 38, pp. 435–475, 1974.

[4] D. W. Matula, "Graph theoretic techniques for cluster analysis algorithms," in *Classification and Clustering*, J. Van Ryzin, Ed. New York: Academic Press, 1977, pp. 95–129.
[5] C. T. Zahn, "Graph-theoretic methods for detecting and describing gestalt clusters," *IEEE Trans. Comput.*, vol. 20, pp. 68–86, 1971.
[6] R. Urquhart, "Graph theoretical clustering based on limited neighborhood sets," *Pattern Recognit.*, vol. 15, pp. 173–187, 1982.
[7] W. L. Koontz, P. M. Narendra, and K. Fukunaga, "A graph-theoretic approach to nonparametric cluster analysis," *IEEE Trans. Comput.*, vol. 24, pp. 936–944, Sept. 1976.
[8] Z. Wu and R. Leahy, "Tissue classification in MR images using hierarchical segmentation," in *Proc. IEEE Int. Conf. Medical Imaging*, Oct. 1990.
[9] R. E. Jensen, "A dynamic programming algorithm for cluster analysis," *Oper. Res.*, vol. 17, pp. 1034–1057, 1969.
[10] W. L. Koontz, P. M. Narendra, and K. Fukunaga, "A branch and bound clustering algorithm," *IEEE Trans. Comput.* vol. 24, pp. 908–915, Sept. 1975.
[11] L. P. Lefkovitch, "Conditional clustering," *Biometrics*, vol. 36, pp. 43–58, 1980.
[12] R. E. Gomory and T. C. Hu, "Multi-terminal network flows," *SIAM J. Appl. Math.*, vol. 9, pp. 551–570, 1961.
[13] G. K. Coleman and H. C. Andrews, "Image segmentation bu clustering," *Proc. IEEE*, vol. 5, pp. 773–785, May 1979.
[14] R. L. Cannon, J. V. Dave, J. C. Bezdek, and M. M. Trivedi, "Segmentation of a thematic mapper image using the fuzzy c-means clustering algorithm," *IEEE Trans. Geoscience Remote Sensing*, vol. 24, pp. 400–408, May 1986.
[15] D. A. Ortendahl and J. W. Carlson, "Segmentation of magnetic resonance images using fuzzy clustering," in *Proc. 10th Conf. Inform. Processing in Medical Imaging*, 1988, pp. 91–106.
[16] P. Sahoo, S. Soltani, and A. K. C. Wong, " A survey of thresholding techniques," *Comput. Vision Graphics Image Processing*, vol. 41, pp. 233–260, 1988.
[17] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. Roy. Soc. Lon.*, vol. 207, pp. 187–217, 1980.
[18] L. R. Ford, Sr. and E. Fulkerson, *Flows in Networks*. Princeton NJ: Princeton Univ. Press, 1962.
[19] N. Christofides, *Graph Theory: An algorithmic Approach*. New York: Academic Press, 1975.
[20] L. Lovász and M. D. Plummer, *Matching Theory*. Amsterdam: Elsevier Science Pub. B.V., 1986.
[21] R. K. Ahuja and J. B. Orlin, " A fast and simple algorithm for the maximum flow problem," *Oper. Res.*, vol. 37, pp. 748–759, 1989.
[22] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 6, pp. 721–741, Nov. 1984.
[23] Z. Wu and R. Leahy, "A graph theoretic approach to image segmentation of MR images," in *Proc. SPIE/SPSE's Symp. on Elect. Image Sci. & tech.*, vol. SPIE-1450, Feb. 1991.
[24] Z. Wu, "Hierarchical and graph theoretic approaches to image segmentation and pattern classification," Ph.D. dissertation, Signal and Image Processing Inst., Univ. of Southern California, 1991.
[25] Z. Wu and R. Leahy, "Unsupervised hierarchical segmentation of textured images based on homogeneity testing," USC–Signal & Image Processing Inst., Tech. Rep. #157, June 1990.



Richard Leahy was born in Surrey, England in 1960. He received the B.Sc. and the Ph.D. degrees in electrical engineering from the University of Newcastle upon the Tyne, England in 1981 and 1985, respectively.

In 1985, he joined USC, Los Angeles, CA where he is currently an Associate Professor in the Signal and Image Processing Institute of the Department of Electrical Engineering-Systems. He holds a joint appointment with the Department of Radiology at USC. His research interests include medical image reconstruction and analysis, biomedical signal processing, and other applications of optimization theory and statistics in signal and image processing.



Zhenyu Wu (S'90–M'93) was born in Zhejiang, China, 1961. He received the B.S. degree in computer engineering and the M.S. degree in electrical engineering from the National University of Mexico, Mexico City, in 1984 and 1986, respectively, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, in 1992. From 1984–1986, he was with the Instituto de Investigaciones Electricas (IIE) of Mexico. From 1987 to 1992, he was a Research Assistant in the Signal and Image Processing Institute at USC. Since March 1992, he has been a postdoctoral fellow in the Medical Image Processing Group of the Department of Radiology, University of Pennsylvania, Philadelphia. His current research interests include image processing, pattern classification, graph theory, and their applications to medical imaging.