A HIERARCHICAL IMAGE SEGMENTATION ALGORITHM

Wei Yu, Jason Fritts

Washington University
Computer Science Department
St. Louis, MO 63130

Fangting Sun

Iowa State University
Computer Science Department
Ames, IA 50011

ABSTRACT

This paper presents a hierarchical image representation that preserves spatial relationships, and an image segmentation algorithm that efficiently constructs this representation. The overall structure of the hierarchical representation assumes a tree-like organization, in which a node at a higher level represents of a group of nodes from the next lower level. The root node of the representation is the full image, and the leaf nodes are $K \times K$ blocks of pixels (usually 4×4 or 8×8), called *blobs*. The segmentation algorithm builds this hierarchical representation using an efficient clustering algorithm for grouping nodes progressively into larger objects. While performing the union of nodes at one level into a single parent node at the next higher level of the tree, nodes are grouped together according to similarities between their feature vectors, which include such features as color information, orientation, texture, size, energy, and neighbor information. The computation complexity of the hierarchical image segmentation algorithm is $O(MN + \frac{MN}{K^2}\log(\frac{MN}{K^2}))$, where M, N are the height and width of the original image. Experimental results show that this algorithm can segment images and extract objects effectively.

1. INTRODUCTION

Object extraction via image segmentation plays a key role in many applications. Unfortunately however, it remains a difficult problem. There does not exist a single generic algorithm that works well for all applications. Target applications vary in the necessary degrees of precision, efficiency, and feature information required from image segmentation, so different algorithms are typically better suited to different applications. For some application areas, like content-based image retrieval (CBIR) and MPEG-4 video compression, it is desirable to have a segmentation algorithm that preserves information regarding the spatial relationships between objects. Toward this end, we present an efficient image segmentation algorithm that generates a hierarchical image representation that preserves these spatial relationships.

Compared with most non-hierarchical segmentation algorithms, such as the *K-means* algorithm used in SIMPLIcity [1], algorithms based on a hierarchical representation preserve the spatial and neighboring information among segmented regions. For applications like content-based image retrieval and MPEG-4 video compression, the ability to preserve spatial relationships between objects can improve image retrieval and motion estimation efficiency. For content-based image retrieval, segmentation schemes can be used to extract objects and their features from images, and then use the object feature information to increase image retrieval efficiency [2]. The additional feature information regarding neighbor relationships between objects can be used to further improve image retrieval for image queries where spatial information about objects in the image is key to effective image recall.

For MPEG-4 video compression [3], and other content-based video compression schemes, image segmentation is used to divide scenes up into objects. By performing motion estimation and compensation on objects, the minimal temporal variations within each object enables higher compression ratios. The added benefit of spatial information could further improve algorithm efficiency, allowing potentially simpler and more efficient motion tracking and object occlusion detection.

A number of segmentation algorithms for building a hierarchical image representations have been proposed [4, 5, 6, 7]. Although these hierarchical segmentation algorithms perform well in some specific areas, all of them suffer from various drawbacks. Nacken [5] and Shen et al. [6] employ methods similar to ours of iteratively grouping nodes in a hierarchical structure. However, their methods can lead to excessive iterations, generating a large number of hierarchy levels. Montanvert et al. [4] use stochastic analysis for image segmentation, but their segmentation results are not unique due to the stochastic decimation applied at each level. Furthermore, all of the above algorithms use only the original pixel values when performing pixel clustering, which Marr argues is not sufficient [8]. Finally, none of these prior studies has examined the computation complexity of their algorithms. No data is provided in [4, 5, 6] regarding computation complexity, and only empirical results are provided by Marr [7].

In this paper, we propose an efficient hierarchical image representation data structure and a powerful image segmentation algorithm for constructing this representation. We propose three new enhancements to hierarchical image segmentation to alleviate some of the drawbacks of previous approaches. First, we propose an efficient new clustering scheme using a modified UNION-FIND data structure. This algorithm achieves low computation complexity by minimizing both the number of hierarchy levels and the computation complexity for each level of the image hierarchy. Second, when clustering nodes from different regions in each level, besides just color information, the algorithm considers a variety of additional features, including the orientation, texture, size, neighbors, and energy. Finally, this algorithm provides a flexible scheme for specifying when to stop the clustering procedure. The user can specify either the desired number of objects or the desired feature distance (i.e. the maximum feature distance below which clusters may be combined). Or, if the user specifies neither, a full hierarchical representation of the original image will be constructed (i.e. with the root node of the tree being a single object corresponding to the whole image).

The remainder of this paper is organized as follows. Section

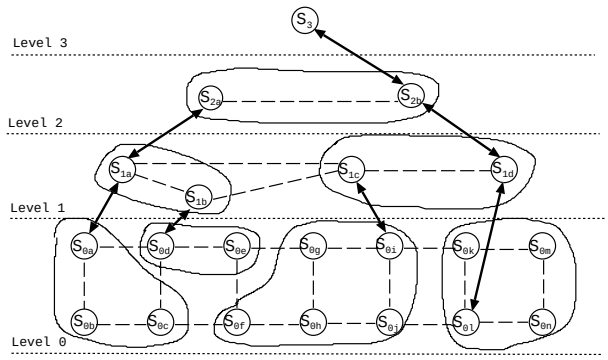


Figure 1: Hierarchical Image Representation

2 summarizes the image representation theory of Marr [8], and discusses the proposed hierarchical image representation and associated image segmentation algorithm. Section 3 then discusses the specific details of the segmentation algorithm. Section 4 analyzes the computation complexity of the algorithm, and section 5 gives some preliminary experimental results. Finally, section 6 summarizes our conclusions and discusses future work.

2. HIERARCHICAL IMAGE REPRESENTATION

In defining an image representation, it is important to recognize that the intensity values in an image are derived from four main factors: geometry, surface reflectance, illumination, and viewpoint. Consequently, it is insufficient to utilize just intensity information during image segmentation. As elaborated by Marr [8], in addition to color intensity information, it is important to define a representation of both surface reflectance and geometrical organization. The representation should contain features that can be derived reliably and repeatedly from images, and to which can be assigned values of attributes like orientation, brightness, size, and position. Further, Marr [8] provides some important underlying physical assumptions for the surface reflectance function of an image: *Existence of surface*, *Hierarchical organization*, *Similarity*, *Spatial continuity*, *Continuity of discontinuities*, and *Continuity of flow*.

Based on these physical assumptions, we propose a hierarchical image representation. This representation embodies a tree-like organization based on the UNION-FIND data structure [9]. The root node represents the full image, and the leaf nodes represent $K \times K$ blocks of pixels (usually 4×4 or 8×8), called *blobs*. This representation defines a hierarchy of levels, where each level consists of two data structures, a weighted graph $G_{level} = (V_{level}, E_{level})$, and a disjoint-set S_{level}. Each vertex of G_{level} represents one region of the original image and has a feature vector characterizing this region. Each set in S_{level} is composed of one or more nodes, and each node has a pointer to one vertex of G_{level}. The two data structures are further defined by Cormen et al. [9]. G_{level} specifies the spatial relationship between regions, where two vertices are connected by an edge if and only if their corresponding regions neighbor each other. The weight of each edge defines the degree of the similarity between two regions. In addition to the standard definition for G_{level}, each vertex of G_{level} has two additional attributes: $child$ and $parent$, which define the relationship between nodes in two adjacent levels.

Algorithm 1 segmentation

```
1: NodeSet := ExtractFeature(image)
2: G_0 := InitGraph(NodeSet)
3: S_0 := InitSet(G_0)
4: level := 0
5: while (StopCondition() = true) do
6:    for (e ∈ E_level) do
7:       if (e.dist < threshold[level]) then
8:          s_1 := FINDSET(S_level, e.LeftNode)
9:          s_2 := FINDSET(S_level, e.RightNode)
10:         if ((s_1 ≠ s_2) then
11:            UNION(S_level, s_1, s_2)
12:         end if
13:      end if
14:   end for
15:   for (s ∈ S_level) do
16:      if (|s| = 1) then
17:         sibling := FindNearSib(S_level, G_level, s)
18:         t := FINDSET(S_level, Sibling)
19:         Link(S_level, s, t)
20:      end if
21:   end for
22:   level := level + 1
23:   G_level := CreateGraph(S_level−1, G_level−1)
24:   S_level := CreateSet(G_level)
25: end while
26: OutputResult
```

Fig. 1 provides an example of our hierarchical image representation. The weighted graph in each level is composed of the nodes and edges within this level. The edges in each level define the spatial relationships between the nodes, and each edge has a weight to represent the similarity between those two nodes. For example, the nodes $\{S_{0a} - S_{0n}\}$, and their associated edges, shown in Fig. 1, compose the weighted graph of level 0. The relationship among the disjoint-set in each level is illustrated by the groupings. For example, in level 2, the four nodes belong to two sets, $\{S_{1a}, S_{1b}\}$ and $\{S_{1c}, S_{1d}\}$, and nodes S_{1a} and S_{1d} represent the canonical elements of these sets, respectively. $child$ and $parent$ attributes of each vertex are illustrated by double-arrow edges, which define the relationship between the nodes of adjacent levels.

3. IMAGE SEGMENTATION ALGORITHM

The segmentation algorithm is presented in Alg. 1. Given an RGB color image with height M and width N, the image is initially partitioned into small $K \times K$ blocks of pixels (usually 4×4 or 8×8, but this may vary by image size such that large images have large blobs while small images use small blobs), called *blobs*. Each blob has an associated set of features extracted from the original pixel intensity values of the image, including color, orientation, texture, and energy information. Based on these extracted features, and the neighboring relationships among blobs, the lowest level of the representation is built. Then node clustering and region merging are performed at each level using a bottom-up strategy. At the end of each iteration of the *while* loop, the algorithm has completed one level of the hierarchy, so a new level is constructed and the representation is updated. The procedure is repeated until the stopping condition has been attained, which is defined as either the desired

final number of objects, $SegNumber$, or the maximum feature distance, $DistThreshold$, for merging regions. Or if not specified, the algorithm will continue until a single root node exists that defines the full image. From this representation, objects and features can be extracted easily through a top-down traversal of the final hierarchical representation of the image.

In each iteration of the *while* loop, the first for loop clusters neighboring regions of this level according to some $threshold$ value ($threshold$ may be defined locally for each level, or globally for all levels). This threshold is usually small, no more than 5. To ensure low computation complexity, the second for loop requires that each single-node set merge with at least one other set in this level. So, before a new level is generated, this for loop guarantees that all sets of the current S_{level} include at least two nodes. Consequently, each new level of the hierarchy is guaranteed to have no more than half the number of nodes as the previous level, ensuring fast convergence of the algorithm.

Below are the implementation details of subroutines:

- $ExtractFeature(image)$ partitions an image into $K \times K$ blobs (usually 4×4 or 8×8), with each blob corresponding to a vertex of G_0. Then, the wavelet transform [10] extracts orientation and texture information from each blob, using a similar method to Wang et al.'s [1]. Two kinds of orthonormal wavelet filters are used, the Haar filter and the Daubechies-4 filter [10]. Finally, nine features are extracted for each blob: the first three are the mean color values of the blob; the second three are the orientation values in the horizontal, vertical, and oblique directions (which also implicitly define the texture information in these directions as demonstrated by Wang et al. [1]); the last three are the deviations of the color values of this blob.
- $InitGraph(NodeSet)$ creates and initializes the lowest level's weighted graph, G_0. Two vertices are connected by an edge if and only if the two corresponding blobs in the original image are neighbors of each other. In this paper, the Euclidean distance of the feature vectors for two connected nodes is calculated and assigned to the edge, and each feature has the same weight when calculating distance.
- $InitSet(G_0)$ creates and initializes the disjoint set S_0 of the lowest level, such that each set has only one element and each element points to one vertex of G_0.
- $FINDSET(S_{level}, s)$ and $UNION(S_{level}, s_1, s_2)$ are the standard implementations of disjoint-set operations for the UNION-FIND data structure [9] . Path compression [11, 12] is used in $FINDSET$ to reduce the computation complexity of the set union operations.
- $FindNearSib(S_{level}, G_{level}, vertex)$ returns the vertex with the smallest feature distance to $vertex$ among all the neighboring vertices of $vertex$ in G_{level}.
- $CreateGraph(S_{level-1}, G_{level-1})$ creates a new weighted graph $G_{level} = (V_{level}, E_{level})$, with a number of nodes, V_{level}, equal to the number of sets in $S_{level-1}$. Nodes of V_{level} are only connected by an edge if their corresponding sets in $S_{level-1}$ were likewise connected. Each vertex of G_{level} has a feature vector with six mean values and six deviation values, except the vertices in the lowest level, which have only three deviation values (orientation has no deviation values in the lowest level). For any $v \in V_{level}$, which is mapped from the vertex set s of $S_{level-1}$, the mean feature values are calculated using Equ. 1:

$$v.f_i = \frac{1}{\sum_{u \in s} u.size} \sum_{u \in s} u.size * u.f_i \qquad (1)$$

where $1 \leq i \leq 6$. The deviations are calculated using Equ. 2:

$$v.f_j = \sqrt{\frac{\sum_{u \in s} u.size * (u.f_{i-6} - v.f_{i-6})^2}{\sum_{u \in s} u.size}} \qquad (2)$$

where $7 \leq j \leq 12$. $size$ is an attribute of vertex, which defines how many blobs of the original image this region has. The weight of an edge e, that is, the distance between the connected vertices v_1 and v_2, is calculated using Equ. 3:

$$e.dist = \sqrt{\frac{\sum_{i=1}^{12} (v_1.f_i - v_2.f_i)^2}{12}} \qquad (3)$$

Each vertex in G_{level} has an attribute $least_edge$ pointing to the edge with smallest feature distance among all the edges connected to this node. For each vertex, $child$ points to the set s in $S_{level-1}$ it corresponds to, and the $parent$ of the canonical element of s points to this vertex.

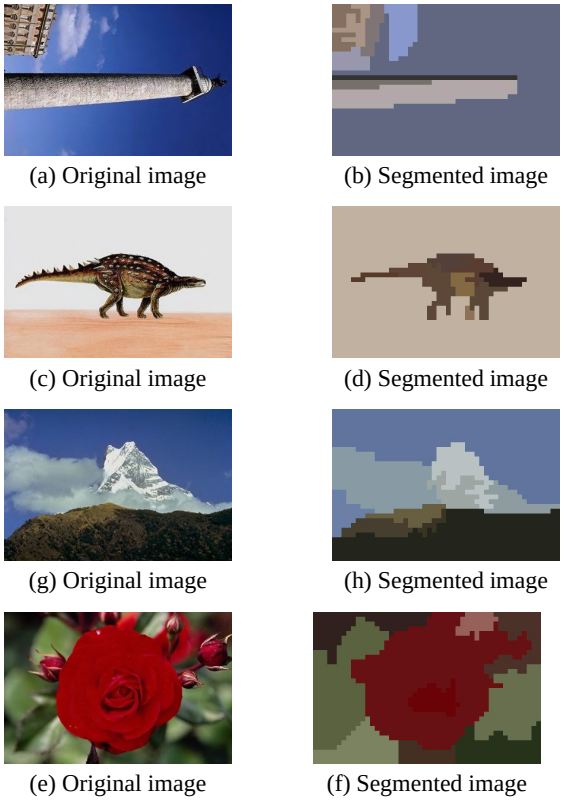
- $CreateSet(G_{level})$ creates a new disjoint-set structure, S_{level}, with each node in V_{level} occupying a single set.
- $StopCondition()$ becomes true only, thereby stopping further merging of nodes in the hierarchical representation, when the stopping condition specified by the user is met. The user can specify the stopping condition in one of three ways: (1) stop when the number of objects at the current level becomes greater than or equal to $SegNumber$, (2) stopo when the distance between any two connected objects nodes is greater than or equal to $DistThreshold$, or (3) stop if neither methods (1) or (2) were specified, and there is only one node remaining (i.e. a single node at the root of the tree defining the complete original image).

4. COMPUTATION COMPLEXITY

To determine the computation complexity of the proposed hierarchical image segmentation algorithm, we need to examine the computation complexity of each step within the algorithm. The subroutine *ExtractFeature* needs time $O(MN)$, since there are $\frac{MN}{K^2}$ blobs, the wavelet transform of each blob requires $O(K^2)$ time, and the time for feature calculation is $O(1)$. The subroutines *InitGraph* and *InitSet* need $O(\frac{MN}{K^2})$ time, because there are $\frac{MN}{K^2}$ vertices and $\frac{2(M-1)(N-1)}{K^2}$ edges in G_0 and $\frac{MN}{K^2}$ sets in S_0 initially. After the end of one *while* iteration, the number of nodes in level l is at most half of the number of nodes in level $l-1$, so there are at most $(\log \frac{MN}{K^2})$ *while* iterations. Because path compression is used, the total time needed for the UNION-FIND operations in one *while* iteration is $\Theta(\frac{MN}{K^2} \alpha(\frac{MN}{K^2}, \frac{MN}{K^2}))$ [13], where $\alpha(\frac{MN}{K^2}, \frac{MN}{K^2})$ is a functional inverse of Ackerman's function and is no more than 5 in general. The subroutine *FindNearSib* needs $O(1)$ time. So the total time needed in one *while* iteration is $O(\frac{MN}{K^2})$, except for the subroutines *CreateSet* and *CreateGraph*. The subroutine *CreateSet(G_{level})* needs $O(V_{level})$ time and subroutine *CreateGraph($S_{level-1}$, $G_{level-1}$)* needs time $O(|V_{level-1}| + |E_{level-1}|)$ time. So the total time needed for a *while* iteration is $O(\frac{MN}{K^2})$ and the computation complexity of the whole algorithm is $O(MN + \frac{MN}{K^2} \log(\frac{MN}{K^2}))$.

5. PERFORMANCE EVALUATION

The segmentation algorithm has been performed on more than 100 reference images from the set used by Wang et al. [1]. Four of them are illustrated in Fig. 2. The first image is a 384×256 RGB color image and the others are 256×384 RGB color images. This implementation used 4×4 blobs and a stopping condition of $SegNumber = 10$. The $threshold$ was globally set to 3 for all levels. All of the images were segmented into the displayed hierarchical representations using no more than 7 iterations.



(a) Original image (b) Segmented image

(c) Original image (d) Segmented image

(g) Original image (h) Segmented image

(e) Original image (f) Segmented image

Figure 2: Image Segmentation Results

6. CONCLUSION AND FUTURE WORK

This paper examined a hierarchical image representation data structure that perserves spatial information between neighboring objects, and an efficient image segmentation algorithm for constructing this representation. The algorithm segments images by iteratively clustering pixels into a hierarchy, using feature information such as color layout, neighbors, size, energy, texture, and orientation. The final result is a hierarchical representation of the original image with each segmented region representing an object. Features characterizing each object can be obtained easily by traversing the final representation.

This hierarchical image segmentation algorithm makes three contributions: First, a novel clustering scheme is propopsed that achieves low computation complexity by minimizing both the number of hierarchy levels and the computation complexity per level. Second, in addition to just pixel intensity information, the algorithm also considers the orientation, texture, size, neighbors, and energy when comparing feature distance during clustering. Finally, this algorithm offers a flexible stopping scheme, allowing the user to stop early by specifying the desired number of objects or maximum feature distance objects, or completing a full hierarchical image representation.

In future research, we intend to combine this segmentation algorithm with multiple instances learning algorithms [2] for further improving image retrieval and classification precision in the CBIR applications. We also plan to comprehensively compare our algorithm with other hierarchical and non-hierarchical segmentation algorithms, and explore use of this algorithm, and future variations, with MPEG-4, as well as CBIR.

7. REFERENCES

[1] J. Z. Wang, J. Li, and G. Wiederhold, "Simplicity: Semantics-sensitive integrated matching for picture libraries," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 947–963, 9 2001.

[2] Q. Zhang, W. Yu, S. A. Goldman, and J. E. Fritts, "Content-based image retrieval using multiple-instance learning," *19th Intl. Conf. on Machine Learning*, July 2002.

[3] P. Salembier and F. Marques, "Region-based representations of image and video: Segmentation tools for multimedia services," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 1147–1169, December 1999.

[4] A. Montanvert, P. Meer, and A. Rosenfeld, "Hierarchical image analysis using irregular tessellations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, April 1991.

[5] P. F. M. Nacken, "Image segmentation by connectivity preserving relinking in hierarchical graph structures," *Pattern Recognition*, vol. 28, no. 6, June 1995.

[6] X. Shen, M. Spann, and P. F. M. Nacken, "Segmentation of 2d and 3d images through hierarchical clustering based on region modelling," *Pattern Recognition*, vol. 31, no. 9, September 1998.

[7] I. Molina, L. J. Roa, F. Arrebola, and F. Sandoval, "Hierarchical image segmentation based on nearest neighbour region chains," *Electronics Letters*, vol. 36, no. 13, June 2000.

[8] D. Marr, *Vision*, W. H. Freeman and Company, San Francisco, California, 1982.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.

[10] I. Daubechies, "Orthonornal bases of compactly supported wavelets," *Comm. Pure. and Appl. Math.*, vol. 41, pp. 909–996, November 1988.

[11] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, MA, 1974.

[12] R. E. Tarjan, *Data Structures and Network Algorithms*, CBMS 44. SIAM, Philadelphia, Pennsylvania, 1983.

[13] R. E. Tarjan, "Efficiency of a good by not linear set union algorithm," *J. Assoc. Comput. Mach.*, vol. 22, pp. 215–225, 1975.