Using Dendronal Signatures For Feature Extraction and Retrieval

Luojian Chen, Michael W. Berry
University of Tennessee
and
William W. Hargrove
Oak Ridge National Laboratory

A dendrone is a hierarchical thresholding structure which can be automatically generated from a complex image. The dendrone structure captures the connectedness of objects and sub-objects during successive brightness thresholding. Based upon connectedness and changes in intensity contours, dendronic representations of objects in images capture the coarse-to-fine unfolding of finer and finer detail, creating a unique signature for target objects which is invariant to lighting, scale, and placement of the object within the image. Sub-dendrones within the hierarchy are recognizable as objects within the picture. Complex composite images can be autonomously analyzed to determine if they contain the unique dendronic signatures of particular target objects of interest. In this paper, we describe the initial design of the Dendronic Image Characterization Environment (or DICE) for the generation of dendronic signatures from complex multi-band remote imagery. By comparing sub-dendrones within an image to dendronic signatures of target objects of interest, DICE can be used to match/retrieve target features from a library of composite images. The DICE framework can organize and support any number of alternative object recognition and comparison techniques, depending on the application domain.

General Terms: Algorithms, Standardization

Additional Key Words and Phrases: dendrogram, feature extraction, image characterization, metadata, query matching

1. INTRODUCTION

Images are being generated at an increasing rate by sources such as defense and civilian satellites, military reconnaissance, and biomedical imaging (see [Gudivada

---

Chen's and Berry's work was supported by the National Science Foundation under Grant Nos. ASC-94-11394 and CDA-95-29459.
Name: Luojian Chen
Address: United Airlines - WHQKB, 1200 E. Algonquin Road, Elk Grove, IL 60007 (Luojian.Chen@ual.com)
Affiliation: University of Tennessee
Name: Michael W. Berry
Address: Department of Computer Science, 107 Ayres Hall, Knoxville, TN 37996-1301 (berry@cs.utk.edu)
Affiliation: University of Tennessee
Name: William W. Hargrove
Address: Computational Physics and Engineering Division, P.O. Box 2008, Oak Ridge, TN 37831-6274 (hnw@fire.esd.ornl.gov)
Affiliation: Oak Ridge National Laboratory

and Raghavan 1995]). New image analysis and retrieval techniques are required to effectively extract and use information from these images. Among the many different methods, previous approaches to image analysis and content-based retrieval have mainly taken two directions. The first approach models image contents as a set of attributes extracted manually and managed within conventional database-management systems, which entails a high level of image abstraction. For example, the *Chabot* system (see [Ogle and Stonebraker 1995]) developed at UC Berkeley uses a relational database to store text information describing certain attributes of images (photos), which are searched when a user enters a query. These attributes include abstract, title, comments, copyright information, location where the photo is taken and date when the photo is taken.

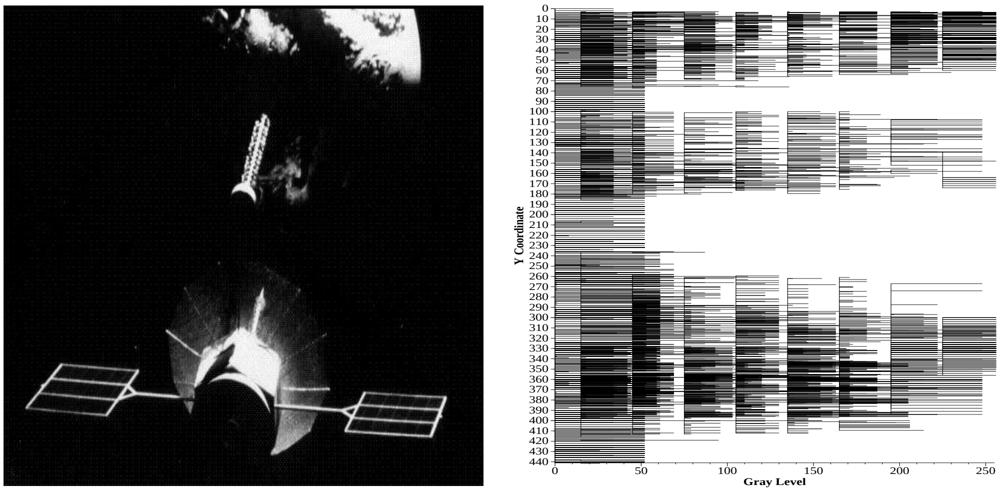
Another approach uses an integrated feature-extraction/object-recognition subsystem to overcome the limitations of attribute-based retrieval. However, this approach is often computationally expensive, difficult, and tends to be domain-specific. One example of this approach is the *QBIC* (Query by Image Content) system (see [Flickner et al. 1995]) developed at IBM Almaden Research Center, which allows queries on large image and video databases based on example images, user-constructed sketches and drawings, selected color and texture patterns, camera and object motion, and other graphical information.

This paper discusses a new technique for image analysis and retrieval based on dendronic image signatures. A dendrone, as described in [Hanusse and Guillataud 1990; Hanusse and Guillataud 1992], is a hierarchical thresholding structure which can be automatically generated from a complex image. Based upon changes in intensity contours, dendronic representations of objects in images capture the coarse-to-fine unfolding of finer and finer detail, creating a unique signature for target objects which is invariant to lighting, scale, and placement of the object within the image. Sub-dendrones within the hierarchy are recognizable as objects within the picture. Complex composite images can then be autonomously analyzed to determine if they contain the unique dendronic signatures of particular target objects of interest. While the dendrone itself does not impose external restrictions to the image, certain attributes could be incorporated into the dendrone. For example, text information describing the image could be added to the dendrone to facilitate content-based retrieval. Different feature description attributes could also be computed and stored in the dendrone for shape-based object retrieval.

A recent implementation of dendrones called **DICE** (**D**endronic **I**mage **C**haracterization **E**nvironment) is described in this work. DICE provides an integrated tool for image feature extraction and object retrieval based on dendronic image signatures. In particular, DICE can be used to create overlays containing *skeletons* of identified objects of interest within the image, created from image data associated with the nodes in the sub-dendrones. Using an object-oriented (C++) software design and implementation, DICE demonstrates the utility of the dendronic signature for like-feature detection across multiple images (perhaps stored in large digital libraries).

As a *visual* introduction to the concept of dendrones, Figure 1 illustrates the dendronic tree representation (Figure 1(b)) of a black and white image of an orbiting satellite (Figure 1(a)). Even without knowing how the dendronic tree representation is obtained, an observer can see from the dendronic signature (Figure 1(b)) that the image contains three primary objects on a noisy background.

The remaining sections discuss the theory, implementation, and evaluation of dendronic image characterization. Section 2 reviews the underlying concepts of dendrones and discusses the algorithms for dendrone construction and sub-dendrone matching. Section 3 is a description of the design and implementation of the DICE software environment including a demonstration of how text (metadata) can be integrated into dendronic data structures for future text/image search retrieval. Section 4 examines the efficiency of dendrone construction and effectiveness of sub-dendrone matching, and Section 5 summarizes the overall effectiveness of dendronic image characterization.



(a) (b)

Fig. 1. Black and white satellite image (a) with dendronic signature (b). Image Courtesy of the USAF Phillip's Laboratory's Satellite Control and Simulation Division.

2. DENDRONE DATA STRUCTURE AND ALGORITHMS

An image may be considered as a two-dimensional intensity field. In addition, if the intensity values of each pixel in the image are considered as elevations, an image can be viewed as a three-dimensional intensity terrain. The brighter the pixel is, the higher the elevation. Brighter pixels form *mountains* with the brightest pixel as the peak. Darker pixels form *valleys* with the darkest pixel as the bottom. However, this imaginary terrain differs from real-world geographical terrain in that the imaginary terrain is *solid*, which means it has no subsurface features such as *holes* or *caves*.

Given an image, a unique dendrone structure can be constructed from the imaginary three-dimensional terrain. The dendrone structure captures the connectedness of objects and sub-objects during successive brightness thresholding. The construction process can be visualized as if the terrain is first flooded with water and then the water is slowly drained away. Initially, the water level is so high that no *land* is visible above the water level. As the water level decreases, *mountains* associated

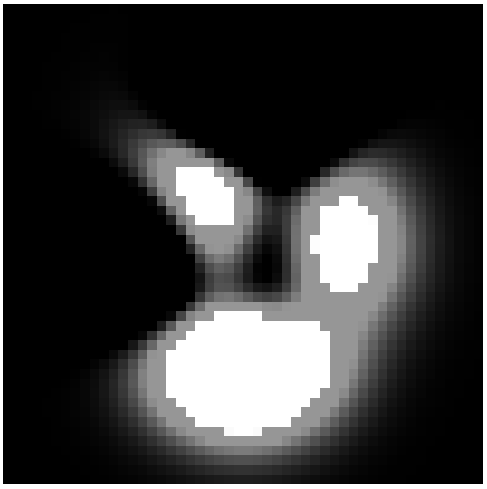


Fig. 2. Image showing three objects.

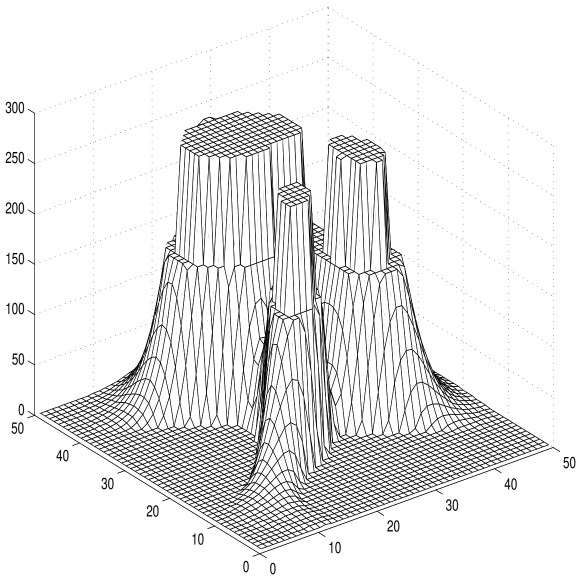


Fig. 3. Imaginary three-dimensional intensity terrain generated from the image in Figure 2.

with the higher elevations appear first, then *plains*, and finally *valleys*. At any particular water (intensity) level, the image is segmented into *islands* (objects). When the water (intensity) level decreases, only three kinds of events occur:

(1) New isolated islands (objects) appear above the water level.
(2) Existing islands (objects) grow in size.
(3) Multiple islands (objects) merge or coalesce and form larger islands (objects).

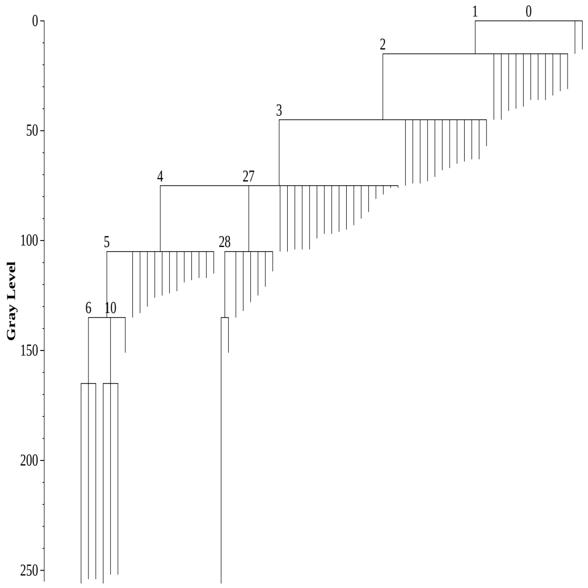


Fig. 4. Dendrogram of the image in Figure 2.

Figure 2 is an artificially generated greyscale image containing three objects, and Figure 3 illustrates the three-dimensional intensity terrain. In the original image, the brightest intensity value is 255 and the darkest intensity value is 0. Accordingly, in the imaginary three-dimensional intensity terrain, the highest elevation is 255 and the lowest elevation is 0. Figure 4 illustrates the tree-like **dendrogram** corresponding to the dendrone generated from Figure 2. This figure illustrates one of many possible ways to draw the dendrone graphically. The dendrone, in this case, is generated with the intensity level decreasing at an intensity resolution increment (stride value) of 30. The smaller the stride value, the more detailed the dendrone is in terms of the number and size of sub-trees.

It is clear that there are three distinct sub-dendrones within the dendrone (see Figure 4), which correspond to the three distinct objects in Figure 2. In this dendrogram, the horizontal axis is arbitrary and the vertical axis indicates the intensity value from 0 to 255. Each vertical line of the dendrogram corresponds to one object within the image. The length of the line indicates the intensity of the object. Each horizontal line connects one vertical line above with several other vertical lines below. The object[1] represented by the vertical line above the horizontal line is formed by those objects (the sub-objects or child objects) represented by the vertical lines beneath the horizontal line. The position of the horizontal lines indicates the intensity level at which the image becomes segmented.

The intensity level decreases from 255 to 0 at a specified stride value. When the intensity level reaches 0, the entire image forms one object, which is represented by the root or the trunk of the dendrone. Despite the flooding/draining allegory, the dendrogram cannot be drawn or constructed until all of the water has been drained and the sequential thresholding is complete. Until then, it is not known where on the arbitrary horizontal axis the vertical lines should be drawn so that they can be connected properly. It is the connectedness as one traverses the dendrone tree from root to leaves that contains information about the relationships between objects.

Figure 5 shows both images and dendrograms generated from the original image in Figure 2 when the water level is three different values. Figure 5(a) shows that when the water level starts at 255, three separate objects begin to appear. Correspondingly, in Figure 5(b), there are three vertical lines representing these three objects. Since they are distinct objects, the three vertical lines have no connections among them. As the water level decreases to 165, two of the three objects have grown larger (see Figure 5(c)) as more components appear. In Figure 5(d), the larger objects are represented by the vertical lines identified by integers *6* and *10*. They link their corresponding sub-objects together by the horizontal lines whose y-coordinate positions are 165. The other object's size does not increase at this intensity threshold. Figure 5(e) shows that when the water level is 135, which is below the elevation of a connected portion shared by two of the three objects, these two objects merge and form a larger object, which is identified by the integer *5* in the dendrogram in Figure 5(f). At that water level, the third object also grows and becomes a larger object, which is identified by integer *28* in the dendrogram. In real-world (more complex) images, the segmentation and merging processes are much more complicated and the three events described above may simultaneously occur, and may involve more than two objects.

2.1 Dendrone Properties

Dendrones generated from images are invariant to scaling. The information stored in the nodes in the dendrones may be different, but the overall structure of the dendrone does not change. Dendrones are also invariant to rotation in terms of connectedness (although they may not necessarily be depicted identically). The relationships among child objects and parent objects will not change although the order of objects within the dendrones may be different. Figure 6 illustrates an enlarged and rotated version of the image in Figure 2. The dendrogram generated from this image is exactly the same as the dendrogram shown in Figure 4. Dendrones are invariant (from a connectedness standpoint) to the placement of objects within the image, and to intensity changes as long as the relative intensity relationships among the objects remain the same.

The dendrone data structure provides a useful and powerful computational framework for image analysis and visual information retrieval. Each node (branching

[1]Actually the parent object identified by the number above the horizontal line.

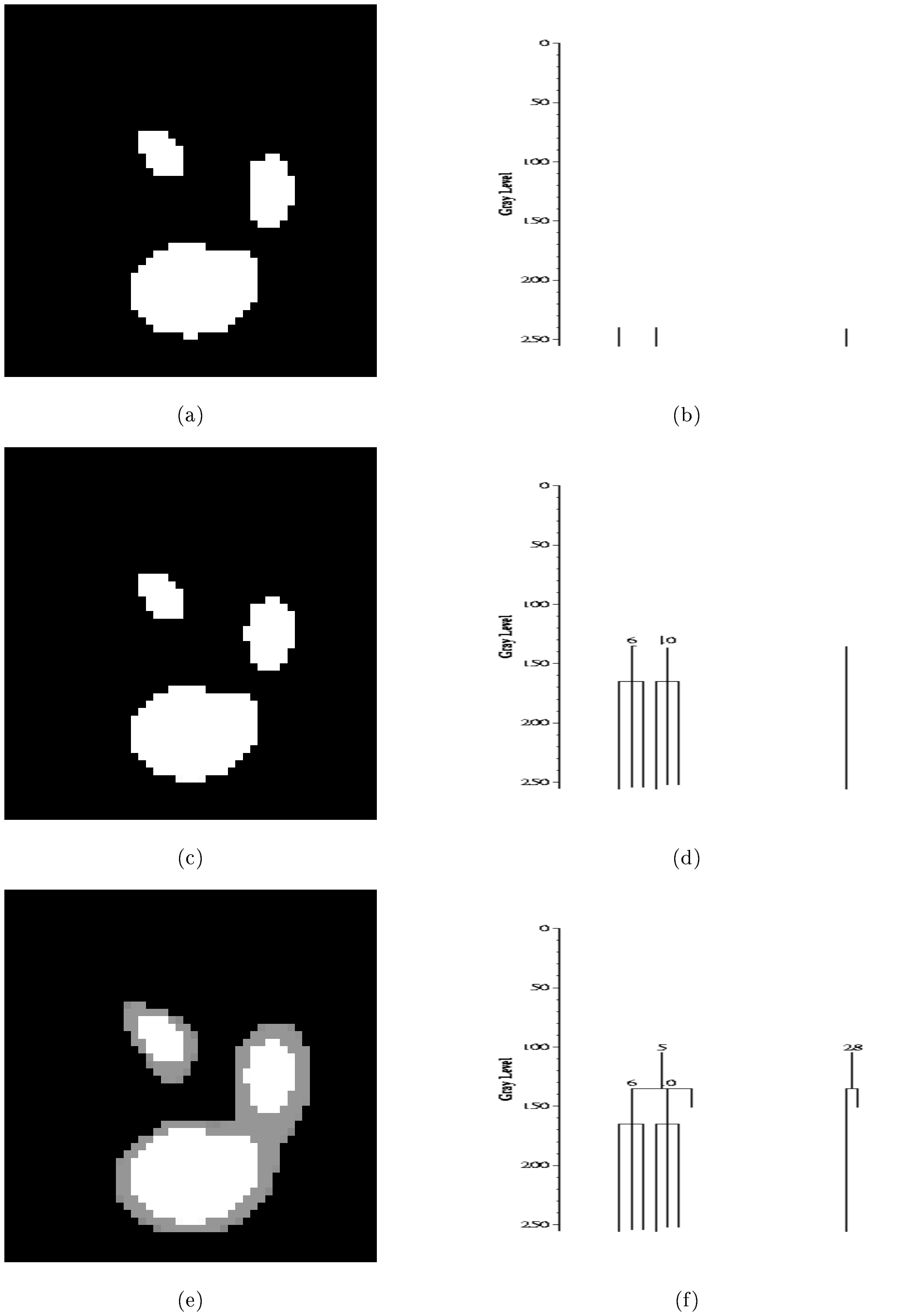


Fig. 5. Image showing three separate objects appearing when the water level is 255 (a) and its sub-dendrogram (b); the growth of two of the three objects (at water level 165) is illustrated in (c) with corresponding sub-dendrogram (d); merging of two objects at water level 135 is shown in (e) along with the sub-dendrogram in (f).

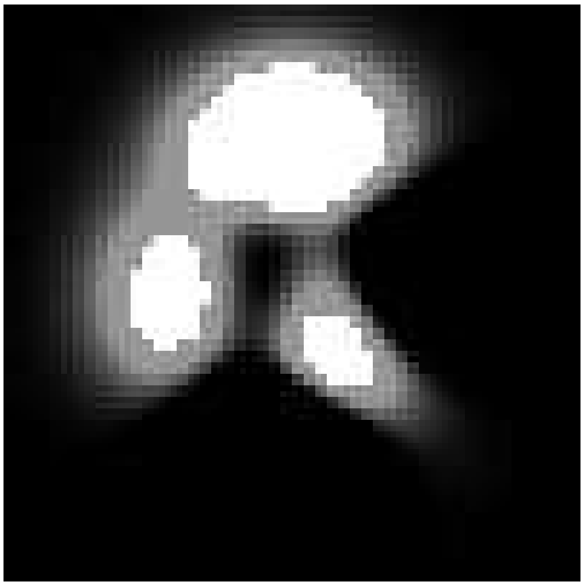


Fig. 6. Enlarged and rotated version of the image in Figure 2.

point) of the dendrone can be used to store ancillary information such as the position of the object within the image, the size of the object, its eccentricity, axis orientation, and information that is directly available from the image itself (e.g., the intensity values of the pixels).

If the dendrone stores information about objects (see Section 3.1) such as pixel coordinates and intensity values, the original image can be reconstructed from the dendrone. Furthermore, one can detect individual objects from the image by extracting sub-dendrones from the dendrone. As demonstrated in Section 4, dendrones can be used to detect objects with similar shape and/or similar brightness topology from multiple images.

2.2 Algorithms

The construction of dendrones involves segmenting the image into isolated objects and building the tree structure from these objects. The matching of dendrones is much more complicated. In developing DICE, our primary intent was to match dendrones representing objects with similar shape and/or brightness topology from multiple images. While the structure of the dendrone is generally sufficient to match objects with similar brightness topology, matching objects with similar shape is more difficult. The development of algorithms in image analysis for shape matching constitutes a major research effort in computer vision. The initial prototype of DICE (Section 3) uses a simple but effective **distance-to-centroid** [Rauber 1994] signature matching algorithm. However, the object-oriented design of DICE facilitates other possible matching algorithms (see Section 3).

2.2.1 *Construction.* The construction of dendrones is accomplished by thresholding the image in a repetitive fashion. At one particular threshold intensity level, the image is processed in two stages: image segmentation, and object merging.

The pixel labeling (or connected components) algorithm presented in [Jain 1989] is used for segmenting a (thresholded) image into isolated objects. The image is first scanned from left to right and then top to bottom, and the current pixel is labeled according to its intensity value. For example, consider the collection of five pixels

$$\begin{pmatrix} A & B & C \\ D & X & \end{pmatrix},$$

where the current pixel is X. The pixels above and to left of the current pixel have already been labeled if they are within the current threshold range. If the intensity value of the current pixel is within the current threshold range, pixels A, B, C, and D are examined and one of the following situations can occur:

(1) None of these pixels are labeled; pixel X is assigned a new label. A new object is created with one pixel X in it.

(2) One of the pixels (A through D) is labeled and X inherits that label. For example, if C is the only pixel that has been labeled, C's label is assigned to X and pixel X is added to the object with that label.

(3) There are two or more qualified labels; these labels are declared to be the same and updated with a new label which is assigned to X. The objects associated with these labels are merged to become a new object (with the new label) and pixel X is added to the new object.

Other possible image segmentation algorithms include Connectivity Filling [Pavlidis 1982], Amplitude Thresholding or Window Slicing [Jain 1989], and Run-length Connectivity Analysis [Jain 1989]. More details on the performance of the pixel labeling algorithm is presented in [Chen 1998].

After all pixels have been scanned, the image is segmented into isolated objects, each with a distinct label. An object *merging* algorithm is then used to compare each object generated from the current segmentation process with any object generated at the last intensity threshold. If two objects (islands) are connected or *touch*, they are merged. Newly generated objects are stored together and will be examined when the image is segmented at a lower threshold intensity level. The newly generated objects also link their respective sub-objects together. If an object does not touch any other object, it is also stored with other newly generated objects and will be examined in future segmentations. Methods for determining whether two objects touch each other are detailed in [Chen 1998].

2.2.2 *Sub-Dendrone Matching.* Sub-dendrone matching can be based on the dendrone structure itself, the shapes of objects represented by the dendrones, or a combination of both. As there are many candidate approaches for comparing sub-objects, we will briefly present two approaches (one based on structure and one based on shape) that have been used in the DICE prototype.

The structure-based matching algorithm takes two dendrones as inputs. One is the *target dendrone* computed from the target image. Usually it is a sub-dendrone extracted from a complete dendrone. The other is the *source dendrone*, which is usually computed from a complete image. Using a breadth-first search on the source dendrone, the difference between number of descendant nodes in the target dendrone and the number of descendant nodes in the source dendrone is calculated for each node of the source dendrone. This difference is recursively calculated for each descendant node as well. A similarity value between 0 and 1 is assigned to every child node to indicate how similar two sub-dendrones are. A similarity value of 1 indicates the two sub-dendrones are identical and a similarity value of 0 means the two sub-dendrones have no similarities.

Several simple rules are used in [Chen 1998] to calculate the similarity between two sub-dendrones:

(1) Leaf sub-dendrones (dendrones that have only one node) and non-leaf sub-dendrones (dendrones that have at least two nodes) have a similarity value of 0.

(2) Two leaf sub-dendrones have a similarity value of 1.

(3) The similarity value of two non-leaf dendrones is computed as:

$$S_{ij} = \frac{\sum \left(S_{kl} \times \frac{N_k + N_l}{2} \right)}{\sum \left(\frac{N_k + N_l}{2} \right) + \sum N_m}, \tag{1}$$

where S_{ij} is the similarity between dendrone i and dendrone j and the total number of objects in dendrone i is not more than the total number of objects in dendrone j. S_{kl} is the best-fit similarity between sub-dendrone k, which is a sub-dendrone of dendrone i, and sub-dendrone l, which is a sub-dendrone of dendrone j. N_k is the total number of objects in sub-dendrone k, N_l is the total number of objects in sub-dendrone l, and N_m is the total number of objects in sub-dendrone m, which is an unmatched sub-dendrone in dendrone j.

The weighted average value S_{ij} (see Equation (1)) is used to represent the overall similarity between the source dendrone and every sub-dendrone within the target dendrone.

Shape is another important feature used to identify or match objects from multiple images. Some of the many possible techniques for shape representation and matching (see [Jain 1989]) are: boundary representation techniques (chain codes, fitting line segments, B-spline representation, control points, and Fourier descriptors), region representation techniques (run-length codes, quad-trees, and projections), and moment representation techniques.

A distance-to-centroid representation [Chen 1998] was used for the DICE prototype. Similar to the structure-based matching algorithm described above, this shape-based technique also requires a breadth-first search on the source dendrone. At every node, the distance-to-centroid *signature* of the object represented by that sub-dendrone is computed. The signature is then compared with the signature of the object represented by the target dendrone. The details of how these signatures are computed are provided in [Chen 1998]. Essentially, the centroid (or geometrical center) of the object and the distances from all edge pixels to the centroid must be determined. The edges are interpolated in order to produce signatures which are invariant to scaling. Furthermore, it can be shown that the distance-to-centroid edge signature is also invariant to translation and rotation. Translation, in particular, has no effect on the signature since the position of the object is never used to compute the signature.

3. THE DESIGN AND IMPLEMENTATION OF DICE

DICE (**D**endronic **I**mage **C**haracterization **E**nvironment) is an object-oriented computational framework designed for image characterization and retrieval based on dendronic image signatures. The three primary goals of the design and implementation of DICE are *flexibility*, *extendability*, and *portability*. The design of DICE is based on an object-oriented methodology through the use of programming languages such as C++ and Java.

Flexibility is achieved so that users of DICE can incorporate their own image characterization algorithms. For instance, if the user wants to use an image segmentation algorithm other than the pixel labeling algorithm, he or she can write a method in C++ and override or add to the default algorithm. Alternative techniques for matching objects from multiple images can also be incorporated into DICE. Extendability is achieved since the user can extend the functionalities of DICE by adding more C++ classes. For example, since the default image format for DICE is XPM [LeHors and Nahaboo 1991], the user could easily add classes to process images in other formats, such as GIF and JPEG. Portability is achieved because the graphical user interface (or GUI) is implemented in Java which is platform-independent. The GUI can be implemented using other languages or libraries such as X Window/Motif without modifying other modules.

The DICE software environment can be divided into three modules: the dendrone library module (implemented in C++), the GUI module (implemented in Java), and the interface module between the library and the GUI (implemented in C). Figure 7 illustrates the interactions among these three modules and the user. Details of the API (Application Program Interface) methods for the dendrone library classes are provided in [Chen 1998]. The *Dendrone* class which processes the dendrones produced from images is described below.

3.1 The Dendrone Class

The *Dendrone* class in DICE is used to segment the input image, link objects generated from the segmentations to build the dendrone, reconstruct individual object images from the dendrone, and generate PostScript dendrograms.

3.1.1 *Dendrone Data Structure.* In the design and implementation of the dendrone data structure, a dynamically linked tree structure is used. For the variety of images and the large amount of information a dendrone could contain, static memory allocation is neither practical nor efficient. During the construction of the dendrone, every iteration of the segmentation process generates a large number of objects. Some of these objects may be merged immediately with other objects while others may not be merged until much later. Furthermore, the merging process requires efficient access to the parent objects and child objects. After the dendrone is built, the objects are sorted according to intensity value. In order to retrieve the objects quickly, and without using a lot of memory, a pointer hierarchy is implemented (illustrated in Figure 8). The first set of pointers give immediate access to

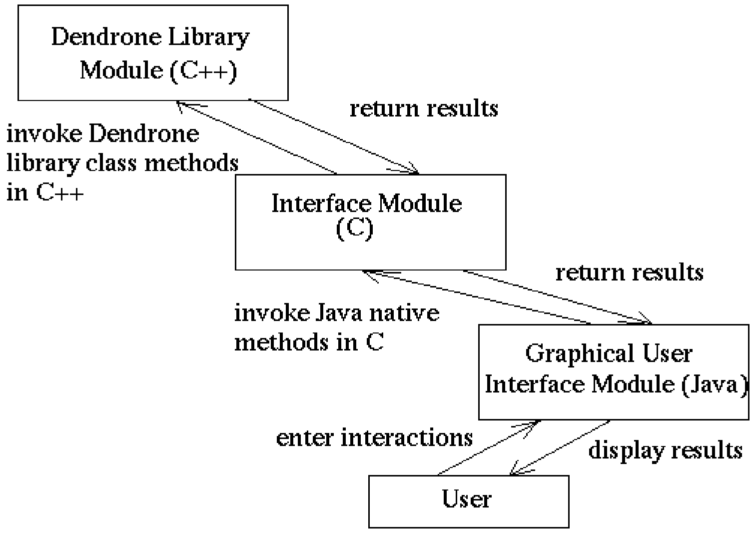


Fig. 7. Interactions among DICE software modules and the user.

the objects in the dendrone through the ids of the objects. On the other hand, the links among related objects give easy access to parent objects and child objects. A third set of pointers group objects with different levels of intensities, which are used during the segmentation and merging processes and also facilitate parallelization of the segmentation process.

3.1.2 *Dendrogram.* For each dendrone, DICE can generate three kinds of dendrograms in PostScript format:

(1) The **non-coordinate** dendrogram (see Figure 4 for an example). The x-coordinate of this dendrogram has no meaning and the y-coordinate indicates the intensity level (ranging from 0 to 255). In addition, the objects in the dendrone are displayed in a certain order:
   (a) Objects with higher intensity values (brighter objects) are displayed to the left side of objects with lower intensity values (darker objects).
   (b) At the same intensity level, composite objects are displayed to the left side of primitive objects.
(2) The **x-coordinate** dendrogram (see Figure 9(b)). The x-coordinate of this dendrogram corresponds to the x-coordinate of the image from which the dendrogram is generated. The y-coordinate indicates the intensity level (ranging from 0 to 255). The individual horizontal and vertical lines in the x-coordinate dendrogram are interpreted as in the non-coordinate dendrogram.
(3) The **y-coordinate** dendrogram (see Figure 9(c)). y-coordinate of this dendrogram corresponds to the y-coordinate of the image from which the dendrogram

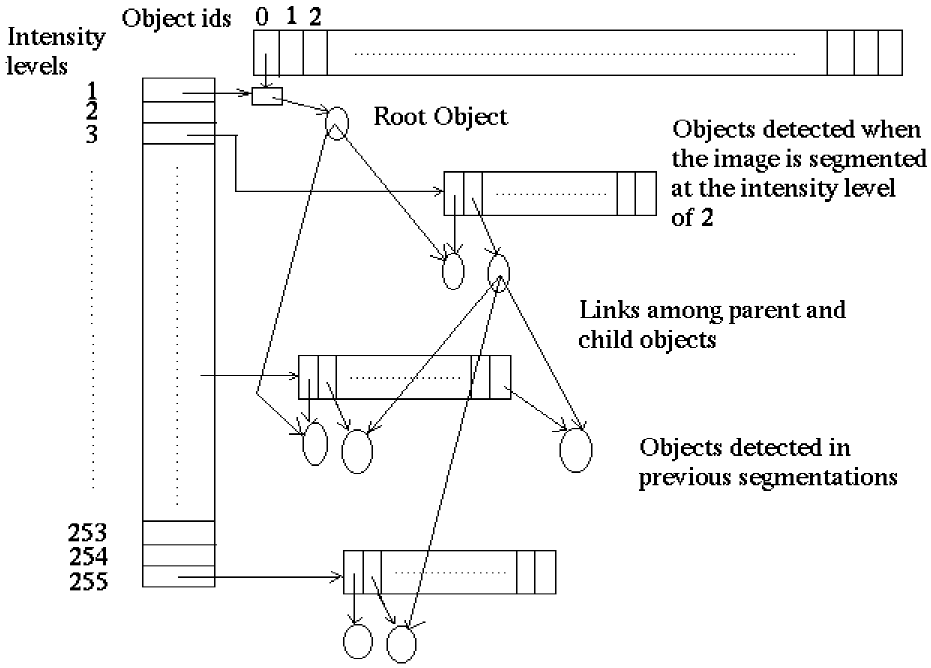


Fig. 8. Dendrone data structure implementation.

is generated, and the x-coordinate indicates the intensity level. As with the x-coordinate dendrogram, the horizontal and vertical lines in the y-coordinate dendrogram designate objects in the image and the intensity levels at which the image is segmented, respectively.

The non-coordinate dendrogram is useful because it does not contain information related to objects' positions within the image, which makes it ideal for comparing dendrones generated from rotated or scaled images. The x-coordinate and y-coordinate dendrograms are useful when constructing individual object images from the dendrone. From these two kinds of dendrograms, the user can locate an object according to its x-coordinate and/or y-coordinate position within the original image (see [Chen 1998] for other methods used to locate objects). As in the non-coordinate dendrogram, composite objects in the x-coordinate and y-coordinate dendrograms are identified by the integers near the lines representing the objects.

In Section 2, the construction of dendrones was *visualized* as decreasing water level on an imaginary three-dimensional intensity terrain. This approach works particularly well for an image with a dark background and bright foreground. If the image has a bright background and dark foreground, the analogy is simply reversed. Namely, increasing the water level (from empty to full) can reveal the relationships among the objects in the image. DICE allows both a decrease and increase in the water level so that the user can control how the dendrone should be properly constructed.

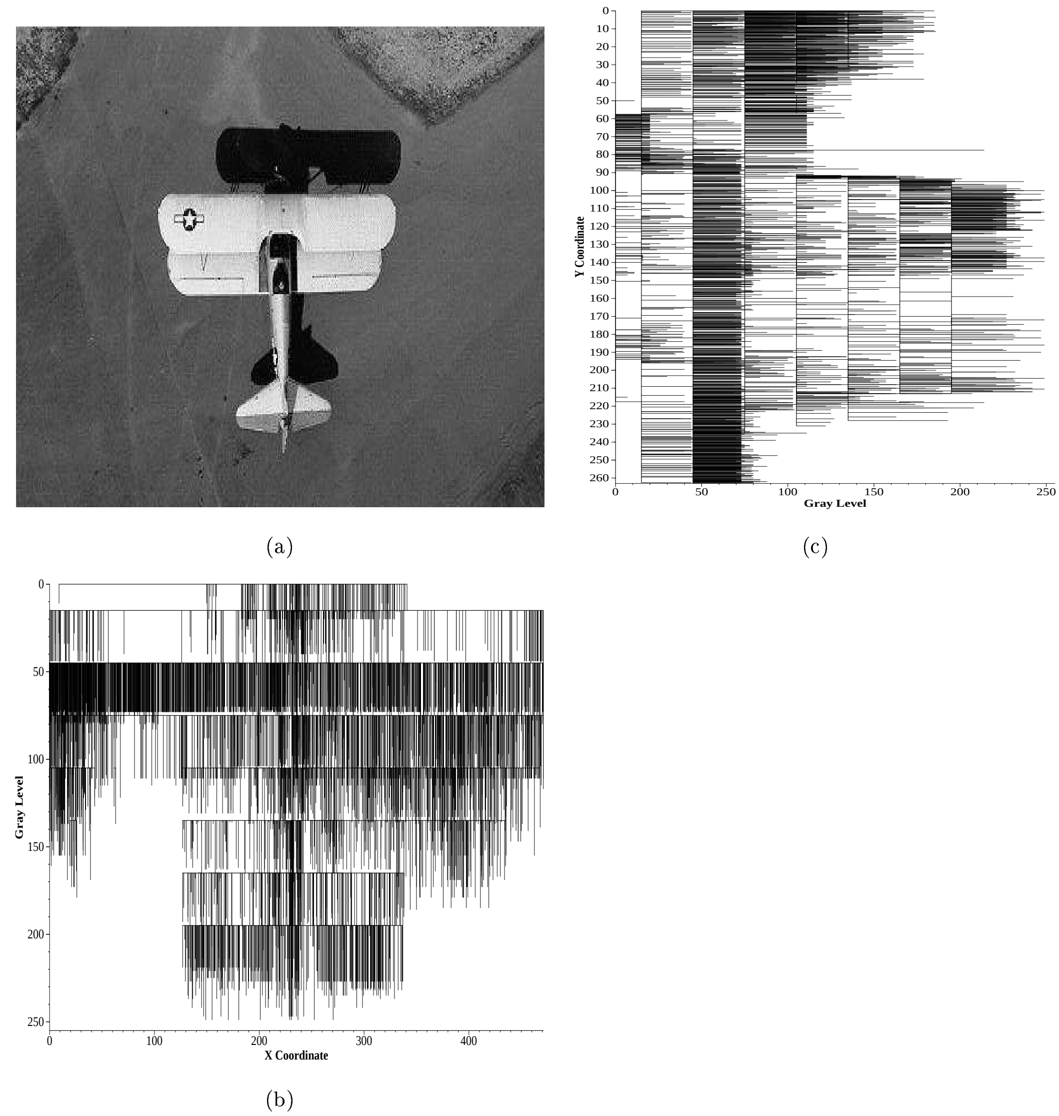


Fig. 9. Bi-plane image (a) with corresponding x-coordinate dendrogram (b), and y-coordinate dendrogram (c).

3.2 Text Files and Metadata

So that a user could save the dendrones generated from images to files for future processing, a simple text format can be defined for dendrone storage. Such a format can incorporate meta-data (e.g., textual descriptions of objects) for alternative retrieval purposes. As an illustration, a header line/record of a DICE-generated dendrone file has the form

$$maxID \quad rows \quad cols \quad left \quad top \quad right \quad bottom \quad stride \quad reverse \ ,$$

where

$maxID$ is the maximum object id in the dendrone,
$rows$ is the number of rows in the image,
$cols$ is the number of columns in the image,
$left$, top, $right$, and $bottom$ are the coordinates of the sub-image from which the dendrone is generated,
$stride$ is the incremental thresholding resolution at which the image is segmented, and
$reverse$ indicates whether the dendrone is generated from the highest intensity value to the lowest or vice versa.

The remaining lines/records of the file contain information about objects in the dendrone. Such objects can be listed according to a *pre-order traversal* of the dendrone. For example, each composite object would be listed as

$$(ID, top, left, bottom, right, cRow, cCol, size, subObjNo, metadata) \rightarrow ,$$

where

ID is the id of the object,
top, $left$, $bottom$, and $right$ are the coordinates of the object's bounding box (i.e., the smallest rectangle containing the object),
$cRow$ is the y-coordinate of the center of the bounding box,
$cCol$ is the x-coordinate of the center of the bounding box,
$size$ is the number of pixels in the object,
$subObjNo$ is the number of child objects, and
$metadata$ is any textual description of the object.

Primitive (or non-composite) objects can be listed under their parent objects respectively, each followed by information about the pixels in the object:

$$\begin{array}{l} (ID, top, left, bottom, right, cRow, cCol, size, 0, metadata) \\ \quad (y_1, x_1, intensity_1) \\ \quad (y_2, x_2, intensity_2) \\ \quad \ldots \\ \quad (y_{size}, x_{size}, intensity_{size}) \end{array}$$

where

y_i $(i \in [1, size])$ is the y-coordinate of one pixel in the object,
x_i $(i \in [1, size])$ is the x-coordinate of one pixel in the object, and
$intensity_i$ $(i \in [1, size])$ is the intensity value of the pixel.

The encapsulation of textual information within the dendronic data structure constitutes a new way of integrating textual and non-textual knowledge. The *metadata* component of each object can be indexed as hypertext so that image databases can be easily linked to other text collections. Conceptual information retrieval models such as Latent Semantic Indexing [Berry et al. 1995; Letsche and Berry 1997] could then be used to match both text and image data to natural language queries. Through the use of dendronic data structures (as shown above), future search engines for information retrieval could permit users to query with both textual and image descriptions:

> *I'm looking for information on tiny blood vessel formations in the macular region of the eye* + [an example or sample image or drawing].

Objects within images could be retrieved by the semantics of encapsulated *metadata*, by some shape- or structure-based similarity measure (see Section 2.2.2 above) applied to dendronic signatures, or a combination of both.

3.3 Graphical User Interface

The Graphical User Interface (or GUI) module, implemented in Java, provides convenient ways for the user to generate dendrones from images, reconstruct images from dendrones, and retrieve objects in images by matching sub-dendrones. The GUI also provides a customizable configuration environment, which the user can modify according to personal preferences.

After the user selects an input image, an *ImageDisplayer* window (see Figure 10) containing the image is displayed. In the *ImageDisplayer* window, the user can select a sub-image (represented by the rectangle) by clicking and dragging the mouse button. Without a selected sub-image, the dendrone will be built for the entire image. By clicking the *Build Dendrone...* button, the user can bring up a *BuildDialog* window (see Figure 11) in which a number of parameters and options regarding the construction of the dendrone can be selected. The user can also set the stride value using the scroll bar and edit the *Left*, *Top*, *Right*, and *Bottom* text fields for the sub-image coordinates by either entering numbers manually or selecting a sub-image in the *ImageDisplayer* window using the mouse. The radio buttons and check boxes allow the user to generate and display selected PostScript dendrograms.

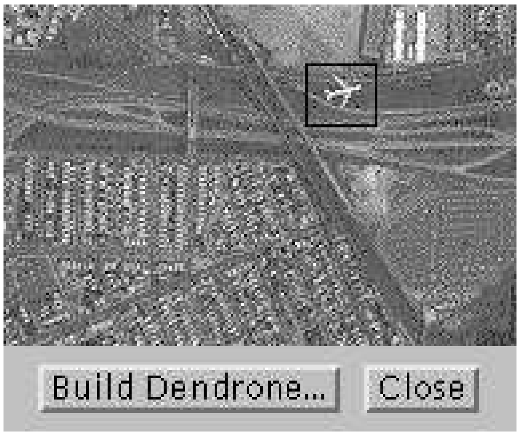


Fig. 10. *ImageDisplayer* window for dendrone construction.

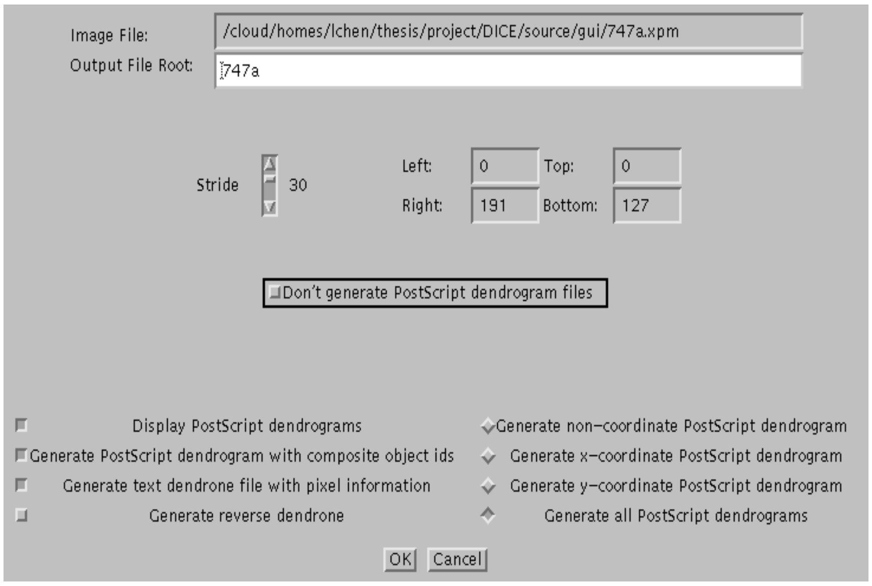


Fig. 11. *BuildDialog* window for dendrone construction.

3.3.1 *Image Reconstruction.* With the help of the DICE GUI, reconstructing images from dendrones can be very convenient. The user first loads the dendrone into DICE by selecting a file storing the dendrone using the *FileDialog*. If the dendrone is successfully loaded, a *DendroneDisplayer* window (see Figure 12) will be displayed. This window contains information about the dendrone, such as the size and coordinates of the (sub-)image from which the dendrone is generated and the stride value used to generate the dendrone. Clicking the *Reconstruct Image...* button will bring up a *ReconstructDialog* window for specifying additional information concerning the reconstruction. An *ImageDisplayer* window (similar to Figure 10) containing the original (sub-)image is also displayed.

3.3.2 *Object Retrieval by Matching Sub-Dendrones.* The *MatchDialog* window (see Figure 13) can be used to retrieve objects with similar brightness topology and/or shape as determined by matching sub-dendrones representing the objects. The user can enter the name of the file containing the dendrone representing the target object to match and the name of the file containing the source image from which objects will be retrieved. To narrow down the number of potential candidate objects, the user can increase the similarity value, or specify the size of the objects to retrieve. Finally, the user can select the criteria for the match. Currently, DICE implements two kinds of matching [Chen 1998] - matching by dendrone structure and by object shape using the distance-to-centroid signature. The user can certainly write his or her own *MatchDialog* class (in Java) to implement alternative matching algorithms. After the user clicks the *Match...* button, the matching results are displayed in a *MatchResultDisplayer* window in which the *scores* are sorted by the similarity value in descending order. The ids and sizes of the matching objects are also displayed so that the user can double-click on an item to obtain a

Fig. 12. *DendroneDisplayer* window.

ReconstructDialog window and reconstruct the selected object's image.

3.4 The Interface Module

Since the dendrone library and the GUI module are written in two different programming languages, an interface module is used to communicate between them. More specifically, in the GUI module, Java objects must invoke methods in the C++ dendrone library. DICE uses the *JNI* (Java Native Interface) mechanism [Campione and Walrath 1998; Eckel 1998] provided by Sun's JDK (Java Development Kit) to accomplish the interactions between Java objects and C++ methods. The interface module, written in C, contains *wrapper* functions which call a C++ method and return results to Java objects. Such functions are required since the JDK only supports native methods written in C. In order to call a C++ method, a C function must be used, which accepts a pointer to a C++ object for subsequent method invocations.

4. PERFORMANCE EVALUATION

To evaluate the performance of dendrone construction algorithms, processing times for a few benchmark images (of different sizes and contents) were recorded. All performance timings were recorded on a Sun Ultra1 SPARCstation with a 167 MHz processor, 32 KB on-chip cache (16 KB Instruction, 16 KB Data), 512 KB external cache, and 256 MB of main memory. A C version of the DICE dendrone library software [Chen 1998] exploiting a command line interface was used applied to Figures 14(a) and 9(a). In Figure 14(b) we illustrate the quality of image reconstruction possible with DICE when a stride-30 dendrogram (storing $8,954$ objects) is used to encode the original image.

Although three different image segmentation algorithms have been evaluated with

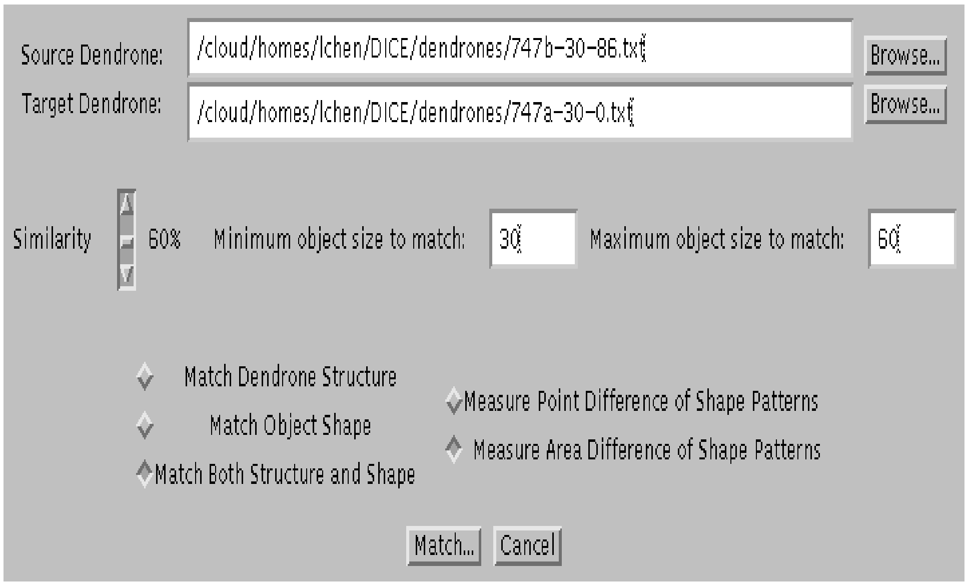
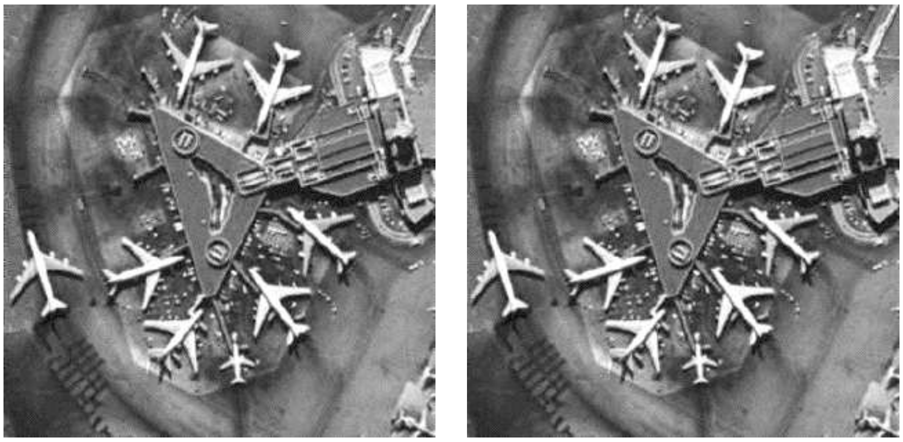


Fig. 13. *MatchDialog* window for object matching.



(a) (b)

Fig. 14. Original high-altitude photograph of a terminal at the Los Angeles Airport (or LAX) in (a), and reconstructed image from a stride-30 dendrogram having 8,954 objects in (b).

DICE (see [Chen 1998]), we present elapsed CPU times only for the pixel labeling algorithm mentioned in Section 2.2.1. For this algorithm, elapsed CPU times were recorded in four key steps of the *buildDendronalTree()* function: image segmentation, object merging, object sorting, and object indexing (i.e., declaring object ids). To reveal the relationship between the time spent in each of these steps and the stride value used to segment the images and construct the dendrones, different stride values ranging from 1 up to 255 were used.

4.1 Processing Time

Elapsed times for constructing dendrones of the 264×462 (in pixels) biplane image (Figure 9(a)) are illustrated in Figure 15 and Table 1. The percentage of time consumed by different steps in the dendrone construction process is also provided in Table 2, where total construction time is broken down at selected stride values.

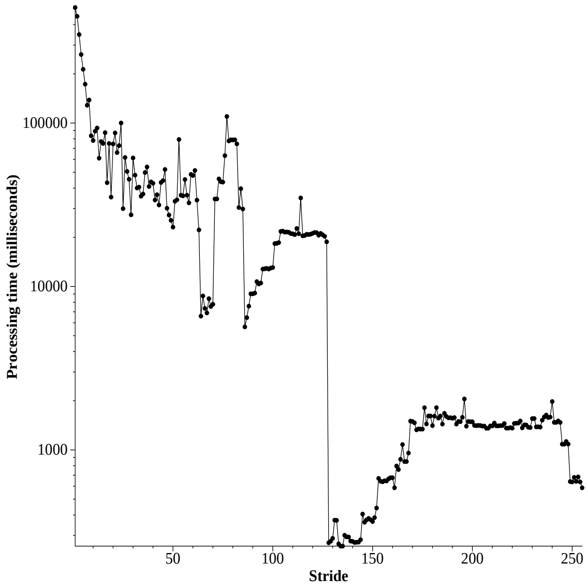


Fig. 15. Total elapsed time of dendrone construction for the image in Figure 9(a) using the pixel labeling algorithm.

Table 1. Elapsed times (milliseconds) of dendrone construction (at selected stride values) for the image in Figure 9(a) using the pixel labeling algorithm.

Stride	Segmentation	Merging	Indexing	Object Sorting	Total
1	15983.31	491851.16	114.31	1654.03	509615.09
5	4612.73	201469.41	76.69	7113.61	213275.58
10	2781.57	65908.59	39.02	9484.02	78214.82
20	1733.67	70949.89	20.09	1756.38	74460.95
30	1419.86	57862.00	12.78	1881.23	61176.66
50	1046.15	19135.43	4.55	2889.49	23076.08
100	695.25	4046.65	0.65	8322.47	13065.35
150	233.76	0.00	1.40	128.96	364.48
200	610.61	0.00	0.10	875.80	1486.73
255	585.97	0.00	0.03	0.07	586.30

Table 2. Percentage time for different steps of dendrone construction (at selected stride values) for the image in Figure 9(a) using the pixel labeling algorithm.

Stride	Segmentation	Merging	Indexing	Object Sorting
1	3.14	96.51	0.02	0.32
5	2.16	94.46	0.04	3.34
10	3.56	84.27	0.05	12.13
20	2.33	95.28	0.03	2.36
30	2.32	94.58	0.02	3.08
50	4.53	82.92	0.02	12.52
100	5.32	30.97	0.01	63.70
150	64.13	0.00	0.39	35.38
200	41.07	0.00	0.01	58.91
255	99.94	0.00	0.01	0.01

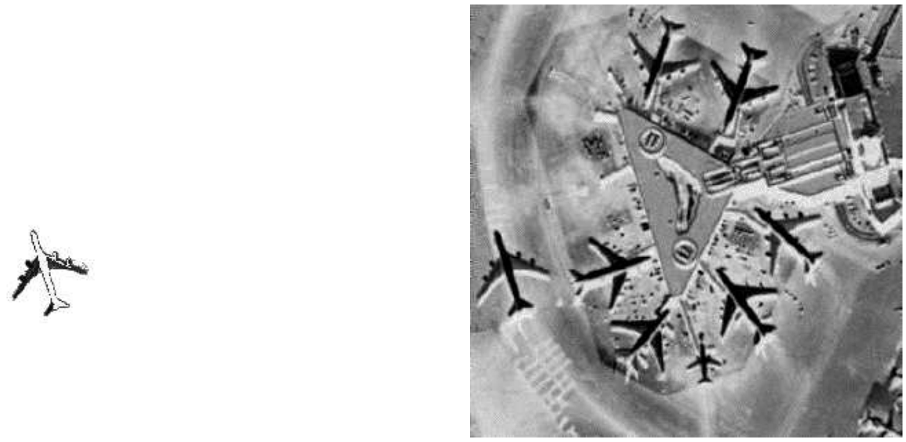
From Figure 15, it is clear that the smaller the stride value, the longer the total time of dendrone construction. As the stride value is chosen to be smaller, relatively more objects are detected during the segmentation processes, although the sizes of the objects may be smaller than those when the stride value is larger. The curve in Figure 15, for example, demonstrates that the majority of objects are segmented between intensity levels 1 and 128. As the number of objects increases, the time for building the dendrone increases dramatically. In fact, the total dendrone construction time increases (regardless of the segmentation algorithm) as the stride decreases [Chen 1998]. Using the pixel labeling approach from Section 2.2.1, Table 2 indicates that dendrone construction time is dominated by the time spent in the object merging step when the stride value is not very large. The object merging step occupies more than 80% of the total time. The performance profile illustrated by Table 2 is just one example of dendronic construction. This profile is certainly image-dependent.

4.2 Memory Requirements

With regard to memory utilization, the pixel labeling version of DICE presented in this section excels in that it does not require any memory beyond that needed to store the dendrone itself. Both the recursive and non-recursive Connectivity Filling versions of DICE (see [Chen 1998]) must use auxiliary memory to store the sorted pixels. In addition, the recursive Connectivity Fill version relies on the operating system for recursive function invocations. It is quite possible that the size and contents of an input image will cause the system's internal stack to overflow. For the biplane image in Figure 9(a), the process virtual memory requirement for dendrone construction (stride set to 30) with Pixel Labeling and recursive Connectivity Filling [Chen 1998] is 3.7 Mbytes and 5.0 Mbytes, respectively.

4.3 Retrieval Performance

The effectiveness of the object matching algorithm using distance-to-centroid image signatures (see Section 2.2.2) has been evaluated using both artificially generated images and real-world complex images. A number of different images have been used to evaluate the effectiveness of the distance-to-centroid image signature matching algorithm. Figure 16 illustrates two of the images used for the evaluation.



(a) (b)

Fig. 16. Images used to evaluate the effectiveness of the distance-to-centroid image signature matching algorithm: airplane object (a) extracted from Figure 14, and a reverse-video image (b) of the same Los Angeles Airport (LAX) terminal from Figure 14.

Tests using artificially generated images that contain simple shapes have proven that the distance-to-centroid image signature algorithm is effective [Chen 1998]. To further test its effectiveness on real-world (more complex) images, in which the shapes and intensity values of objects are much more diverse and irregular, the aerial photograph of the LAX terminal was used. The goal in this case is to locate that airplane and similar airplanes in the reverse-video source image (illustrated in Figure 16(b)). The query image contains only the airplane (illustrated in Figure 16(a)), which is a reconstructed object image (from the target image in Figure 14) obtained via sub-dendrone extraction. The dendrone for the source image in Figure 16(b) was constructed from lowest to highest intensity value (i.e., water level rises as the initial 3-D terrain is filled).

Compared with simple shape matching, matching objects in complex images can be problematic. In such images, small objects that contain only a few pixels are much more likely to be present. Hence, the minimum and maximum object sizes for matching can be used to filter out very small objects during the matching process. A small object's distance-to-centroid signature often only contains a few pixels, which is not very useful in the computation of the signature image. Similarly, scaling a large object may cause the loss of some important features in the scaled signature image. In the extreme case, if the smaller object has only one pixel, its signature image will also have one pixel. If either the larger or smaller object has only one pixel, the matching algorithm will recognize the two images as a perfect match.

Intensity values also play an important role in object matching. By combining shape matching and object brightness topology matching, better results may be achieved. Figure 17 illustrates the top six matched objects from the image in Figure 16(b) according to similarity in distance-to-centroid signatures (see Section 2.2.2). During the matching, only objects whose sizes were between 400 and $1,000$ pixels were processed, and both object shape and intensity level were used for matching. The best match, as expected, is the target/query airplane itself, but several other objects defining complete or partial airplanes (4.5 of the 7 possible planes, in fact) were highly ranked (within the top 6 matches) in similarity. As with all such approaches, the effectiveness of this matching algorithm depends on the input image.

Since the matching is based on (sub-)dendrones, the construction of the dendrone from the input image, particularly the choice of the stride value, is crucial to the effectiveness of the matching process. If the stride value is too large, some objects may never be presented in the dendrone and consequently will never be matched. On the other hand, if the stride value is too small, there will be many potentially matching objects present in the dendrones, which not only slows down the matching process, but also makes the potential matches difficult to differentiate.

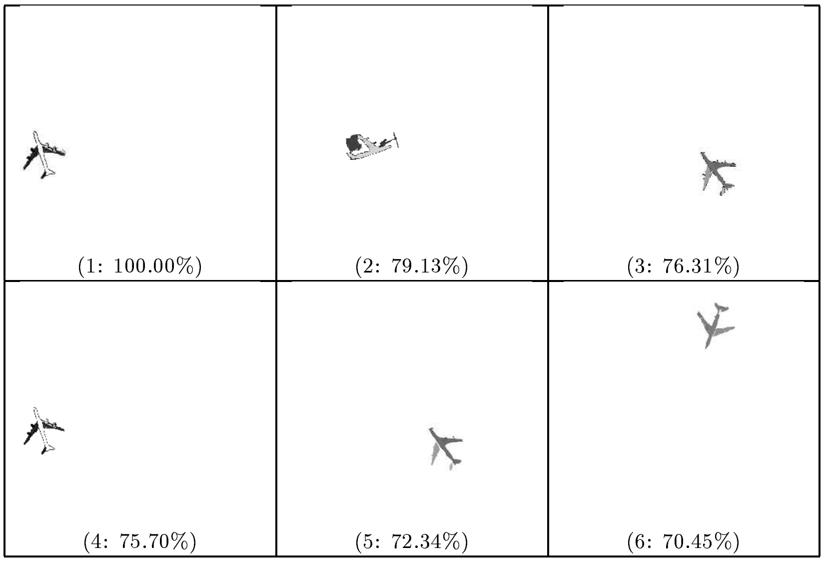


Fig. 17. Matching objects from Figure 16(b).

5. SUMMARY

Dendronic image analysis is a completely neutral, data-driven, self-structuring process. This approach respects the objects within the image, rather than imposing external constraints, divisions, or descriptors onto the image space. Dendronic analysis results in self-definition: a neutral description of objects within the image. At the same time, data compression is achieved. The dendrone is robust to noise, free of contextual information, and invariant under Euclidean geometric transformations. Dendronic signatures have significant potential to provide the basis for an operational capability to detect probable target object identities between multiple images and to recognize and verify similar features in a wide variety of image contexts.

DICE is a flexible, extendable, and portable implementation of dendronic image characterization. The dendrone library and API's allow users to easily extend DICE's functionalities and add alternative algorithms. The graphical user interface provides a simple and straightforward way to analyze images using their dendronic signatures and locate objects with similar shapes (via sub-dendrones) from multiple images. See `http://www.cs.utk.edu/~dice` for on-line information concerning DICE.

Currently, a serial implementation for dendrone construction is used within DICE [Chen 1998]. An optimized parallel approach to image segmentation could speed up the object merging process. Research in shape matching would provide insight into alternative object matching algorithms that could be used for locating objects with similar shapes. Also, the ability to annotate a sub-dendrone with text/metadata describing the corresponding objects could facilitate the integrated retrieval of text and image information from digital libraries.

REFERENCES

Berry, M., Dumais, S., and O'Brien, G. 1995. Using linear algebra for intelligent information retrieval. *SIAM Review 37*, 4, 573–595.

Campione, M. and Walrath, K. 1998. *The Java Tutorial: Object-Oriented Programming for the Internet* (Second ed.). Addison-Wesley, Reading, Massachusetts.

Chen, L. 1998. Feature extraction and retrieval using DICE: Dendronic Image Characterization Environment. Master's thesis, University of Tennessee, Knoxville, Tennessee.

Eckel, B. 1998. *Thinking in Java.* Prentice Hall, Englewood Cliffs, New Jersey.

Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., and Yanker, P. 1995. Query by Image and Video Content: The QBIC System. *Computer 28*, 9, 18–22.

Gudivada, V. N. and Raghavan, V. V. 1995. Content-Based Image Retrieval Systems. *Computer 28*, 9, 18–22.

Hanusse, P. and Guillataud, P. 1990. Object Detection and Identification by Hierarchical Segmentation. In O. Faugeras Ed., *Lecture Notes in Computer Science 427, Proceedings of the INRIA ECCV 90, First European Conference on Computer Vision, Antibes, France, April 23-27, 1990* (Berlin, 1990), pp. 583–585. Springer-Verlag.

Hanusse, P. and Guillataud, P. 1992. Dendronic Analysis of Pictures, Fractals, and Other Complex Structures. In L. Encarnacao, H.-O. Peitgen, G. Sakas, and G. Englert Eds., *Fractal Geometry and Computer Graphics* (Berlin, 1992), pp. 203–216. Springer-Verlag.

Jain, A. K. 1989. *Fundamentals of Digital Image Processing.* Prentice Hall, Englewood Cliffs, New Jersey.

LeHors, A. and Nahaboo, G. 1991. *XPM The X PixMap Format.* BULL Research FRANCE - Sophia Antipolis.

Letsche, T. A. and Berry, M. W. 1997. Large-Scale Information Retrieval with Latent Semantic Indexing. *Information Sciences 100*, 105–137.

Ogle, V. E. and Stonebraker, M. 1995. Chabot: Retrieval from a Relational Database of Images. *Computer 28*, 9, 18–22.

Pavlidis, T. 1982. *Algorithms for Graphics and Image Processing.* Computer Science Press, Rockville, Maryland.

Rauber, T. W. 1994. Two-Dimensional Shape Description. Technical Report GR UNINOVA-RT-10-94, Universidade Nova de Lisboa, Faculdade de Ciècias e Technologia, Lisboa, Portugal.