Available online at www.sciencedirect.com
ScienceDirect
Computer Vision and Image Understanding 110 (2008) 245–259
Computer Vision and Image Understanding
www.elsevier.com/locate/cviu

A similarity-based leaf image retrieval scheme: Joining shape and venation features ☆

Yunyoung Nam [a], Eenjun Hwang [b,*], Dongyoon Kim [c]

[a] *Center of Excellence for Ubiquitous System, Ajou University, Suwon, South Korea*
[b] *School of Electrical Engineering, Korea University, Seoul, South Korea*
[c] *Visual Computing Laboratory, Ajou University, Suwon, South Korea*

Received 18 January 2007; revised 2 July 2007; accepted 19 August 2007
Available online 14 September 2007

Abstract

In this paper, we propose a new scheme for similarity-based leaf image retrieval. For the effective measurement of leaf similarity, we have considered shape and venation features together. In the shape domain, we construct a matrix of interest points to model the similarity between two leaf images. In order to improve the retrieval performance, we implemented an adaptive grid-based matching algorithm. Based on the Nearest Neighbor (NN) search scheme, this algorithm computes a minimum weight from the constructed matrix and uses it as similarity degree between two leaf images. This reduces necessary search space for matching. In the venation domain, we construct an adjacency matrix from the intersection and end points of a venation to model similarity between two leaf images. Based on these features, we implemented a prototype mobile leaf image retrieval system and carried out various experiments for a database with 1,032 leaf images. Experimental result shows that our scheme achieves a great performance enhancement compared to other existing methods.

Keywords: Similarity-based image retrieval; Shape-based retrieval; Leaf image retrieval; Venation; MPP; Mobile computing

1. Introduction

Efficient image retrieval tools can assist people in making efficient use of digital image collections. Traditional image retrieval systems in the late 1970's were mainly based on keyword annotation. This approach suffers from many difficulties including vast amount of human labors in generating annotations for images and efforts for maintaining their consistency. In order to overcome these difficulties, in the last decade, Content-Based Image Retrieval (CBIR) has been researched. This is one of the major approaches of image retrieval, and has drawn significant attention from various communities. Examples of prominent systems based on this approach are VIRAGE [1], QBIC [2], Photobook [3], and VisualSEEk [4].

In many CBIR systems, an image is represented by some low level features such as colour, texture, shape, and structure. Relevant images are retrieved based on the similarity of their low level features. However, depending on the data domain, all such features are not equally useful. For instance, leaves of most plants have similar color feature such as green or brown. That is, color feature is not much useful in querying and retrieving leaf images. However, leaves have distinct shape and venation features and thus can be used for representing and retrieving leaf images. Shape-based image retrieval has been already regarded as an efficient and interesting approach. For example, shape recognition methods have been proposed and implemented into face recognition, iris recognition, and fingerprint recognition.

☆ This research was supported by University IT Research Center Project and a grant (no. BDM0100211) to JRL from the Strategic National R& D Program through the Genetic Re sources and Information Network Center funded by the Korean Ministry of Science and Technology.
* Corresponding author. Fax: +82 2 921 0544.
E-mail address: ehwang04@korea.ac.kr (E. Hwang).

1077-3142/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2007.08.002

The effectiveness of a shape-based image retrieval system depends on the types of shape representation used, the types of queries allowed, and the efficiency of the shape matching techniques implemented. As in the typical content-based image retrieval systems, three steps are required to retrieve relevant images based on shape features; (1) The first step is to detect edge points. (2) The second step is to represent shapes in such way that it is invariant to translation, rotation, scale, and viewing angle changes. (3) The last step is to evaluate shape similarity, which determines how similar images are to a given query image in terms of shape feature.

Among those three steps, effective shape representation is a central problem in visual information systems, computer vision, and pattern recognition. Shape representation methods are classified into two categories: boundary-based and region-based. The former describes a region of interest using external characteristics [5] while the latter represents a region of interest using internal characteristics [6–9]. In this paper, since the primary concern is shape characteristics such as length of boundary, straight line orientation, extreme points to join, and number of concaves, external representation is used. With regard to shape representation, the Minimum Perimeter Polygons (MPP) algorithm can be used [10,11]. MPP is a polygonal approximation method for identifying curvature descriptions [12]. However, this method only uses the outside boundary of a strip of cells. Nevertheless, it may take long time to retrieve due to the fact that it may consider many unnecessary points. Another important issue regarding shape-based image retrieval is the shape matching method, on which retrieval performance is heavily dependent. There are several approaches [13,14] to the shape-matching problem.

In the earlier efforts for leaf image retrieval [15,16], they considered leaf contour for shape similarity measurement. Recently, researchers proposed diverse approaches for extracting venation feature from a leaf. Sollie [17] applied morphological filters to extract leaf veins. Fu and Chi [18] developed an efficient two-stage approach for leaf vein extraction. In the first stage, a preliminary segmentation is carried out based on the intensity histogram of the leaf image to estimate the rough regions of vein pixels. The second stage performs a fine checking using a trained artificial neural network classifier. They demonstrated that their approach was capable of extracting more precise venation modality of the leaf than the conventional edge detection methods. Li, Chi, and Feng [19] developed a venation extraction technique by using Independent Component Analysis (ICA).

However, the leaf venation feature has not yet been exploited for similarity measurement of leaves. Interestingly, plants of same species have nearly same contour pattern. For example, Nuphar japonicum, Nuphar pumilum, Nuphar coreana Nakai, Nymphaea tetragona, and Nymphaea minima Nakai are members of the Nymphaeaceae family. Their common venation feature can be fully exploited for shape-based similarity measurement. In this paper, we propose a new similarity-based leaf image retrieval system that utilizes both contour and venation features. Especially, in order to provide access to the system for the leaf search_in the field, we extend our system to operate in the mobile environment.

The major contributions of this paper are as follows:

- *Contour similarity matching and measurement*. An improved MPP algorithm and a new Adaptive Matching (DM) algorithm are proposed for contour similarity measurement of leaves. The improved MPP algorithm is used to identify curvature descriptions. DM, which is based on the NN (Nearest Neighbor) search algorithm, computes a minimum value of a constructed bipartite matrix for the similarity degree between two leaves, by guaranteeing one-to-one mapping. In order to improve the computation time, we performed two optimizations: (i) point merging based on the angle between points and (ii) leaf complexity reduction based on the symmetric feature of leaves.
- *Venation similarity matching and measurement*. A graph matching algorithm, called Venation Matching (VM), is proposed for venation similarity measurement of leaves. A weighted bipartite graph is constructed to model the similarity between two leaves: every vertex in a bipartite graph represents one node in a leaf, and the weight of an edge represents the distance between two nodes. In venation similarity measurement, a venation graph is constructed to represent the leaf nerve; the venation similarity is measured by the pattern and distance between two venation graphs. Final similarity is calculated based on the weighted sum of venation similarity values.

Another important issue of shape-based image retrieval is the query execution time. Several approaches have been proposed to reduce query execution time. In this paper, we propose a grid-based shape matching algorithm to reduce matching time.

The rest sections of this paper are organized as follows. Section 2 describes the outline of the proposed system. In Section 3, image segmentation is described, and Section 4 presents how to match and retrieve images. In Section 5, experimental results are presented. Section 6 concludes this paper.

2. System Overview

In this chapter, we describe an overview of our prototype system "CLOVER." CLOVER is an effective and robust leaf image retrieval system that works even in a mobile environment.

Since CLOVER represents and retrieves leaf images based on the shape and venation features, the system has typical components for image representation and retrieval including noise filtering, edge detection, shape representation and matching. Noise filtering estimates original image

information by removing or reducing noises from the image. Due to the imperfection of image sensors or imperfections created during photography, digital images are often contaminated with noises. Such noises affect perceptual quality of the image, decreasing not only the appreciation of the image but also the performance of the task for which the image was intended. Therefore, noise filtering is an essential part of many image processing systems and should be performed prior to subsequent processing steps such as image sharpening, edge detection, shape representation and matching.

Edge is another important feature since it implicates the shape of an object in the image. Therefore, edge detection is a common component in many image processing systems. Numerous higher levels of functions such as object recognition, robot vision, and segmentation rely on accurate edge detection.

Shape representation involves shape boundary and boundary points called interest points. The shape boundary is coded as an ordered sequence of interest points. For the index structure, encoded feature vectors representing the shape boundary features help form a feature index for the shape database [27]. A Euclidean distance between two feature vectors defines degree of similarity between the two features. And query processing involves query feature selection, formulation of possibly similar shapes, and formulation of the final response set. Since the user sketch may not match exactly in shape with images in the database, the method elastically deforms the user template to match the image contour. An image for which the template needs to undergo minimal deformation or loses minimum energy is considered the best match.

Fig. 1 presents the overall architecture of CLOVER. A typical scenario for the retrieval is as follows. First, a leaf is photographed using a camera-equipped PDA or sketched on the screen. Second, simple characteristics of the leaf such as the arrangement, apex, and base are extracted. Third, the leaf image and its extracted characteristics are sent to the system, where shape relevant characteristics are extracted from the leaf image. These characteristics are combined with user-specified characteristics to formulate a condition for retrieving images from the database and matched results are displayed on the screen.

As we already mentioned above, CLOVER supports two query modes for content-based retrieval: Query-By-Sketch (QBS) mode and Query-By-Photograph (QBP) mode. In the QBS mode, the user sketches the leaf contour on the touch-screen of a PDA or on a Java applet-based web interface. In the QBP mode, the user takes a picture with a digital camera and transmits the image to the server. Finally, CLOVER extracts leaf contour and venation features automatically, and searches the database for similar images.

3. Feature extraction and shape representation

Image segmentation has been a key problem in computer vision. In the past, various techniques such as edge detection and split-and-merge algorithms were the primary focus of research. Image segmentation decomposes images

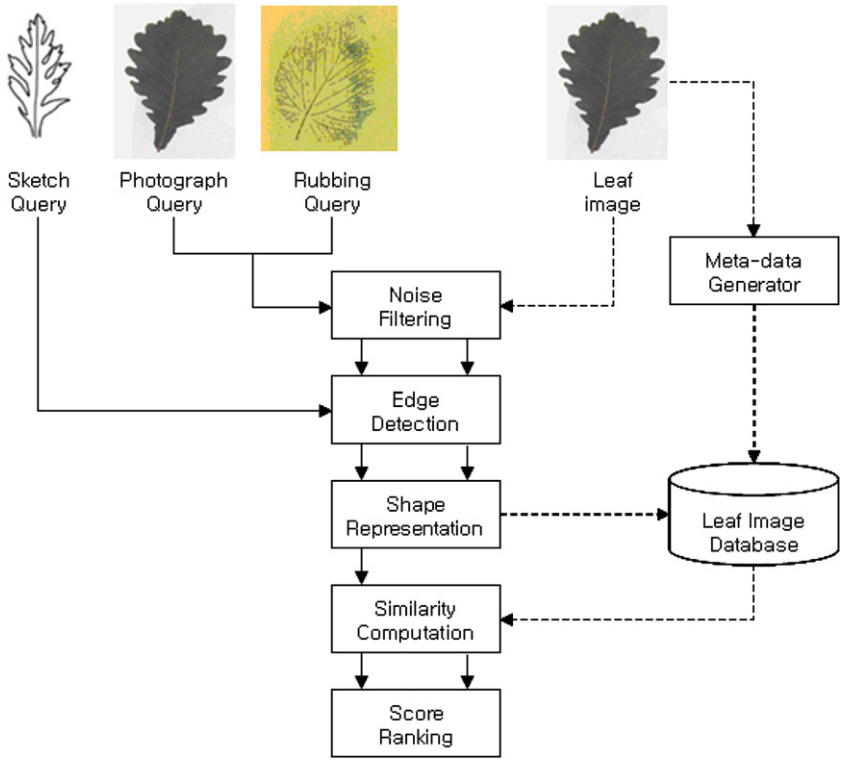

Fig. 1. System Overview.

into multiple regions corresponding to independent objects. It is an essential preliminary step in most automatic pattern recognition and image analysis processes. The goal is to get good image segmentation useful in a variety of situations, much in the way that other low-level techniques such as edge detection are used in a wide range of computer vision tasks. In order to achieve such broad utility, it is important to notice that successful shape feature extraction depends on good image segmentation. In this section, image segmentation methods including edge detection and shape representation for leaf images, are described.

3.1. Shape extraction

Fig. 2 presents detailed steps for extracting an object contour from a photographed image. Every photographed image in the database or in the query should be converted into a gray scaled image for boundary tracing. During this step, small objects are filtered out so that unimportant details are ignored. Next, its grayscale image is made morphologically closed as a single object and holes in the image are filled to estimate the area enclosed by the boundaries. Finally, its contour is extracted by an edge detection method. Through these steps, a photographed image is transformed into a contour detected grayscale image.

3.2. Shape representation

Extracted objects should be represented in a suitable form for efficient processing. There exist various shape representation methods: chain codes [21], Fourier transform [22], and MPP. The chain codes are used to represent a boundary by a connected sequence of straight-line segments of specified length and direction. This representation is based on 4- or 8- connectivity of the segments. Since the chain code method depends on the starting point, it may have following problems: (i) the starting points should be same. If the starting points are different, the code must be rotated to align the different starting points; (ii) even minor difference in the contour could result in different codes. Matching chain codes for a slightly noisy version of the same object can be very difficult; and (iii) the representation is not rotational invariant.

The Fourier transform converts a function from space domain to frequency domain with derived sine wave coefficients describing the given 1-D function. While the transform achieves scale invariance automatically, a change in the rotation angle of the shape results in phase shift of sine coefficients. Due to this property, rotation invariance can also be achieved by reviewing only the frequency magnitude of the frequency coefficients. Despite the robustness in terms of scale and rotation variance, the Fourier transform method(s) do not perform well under noisy conditions.

Another method for compacting the boundary information is to fit line segments or other primitives in the boundary. It is only required to store the parameters of these primitives, instead of discrete points. This is useful because it reduces the effects of the discrete pixelization of the contour. In general, a boundary can be approximated with arbitrary accuracy by a polygon. In the case of a closed curve, the approximation is exact when the number of segments in the polygon is equal to the number of points in the boundary, making each pair of adjacent points define a segment in the polygon. MPP is used for defining curvatures when a change of the slope occurs with control points approximately uniformly spaced along the curvatures. Algorithm 1 presents the steps for finding the MPP of a region and Fig. 3 presents examples of image representation using MPP.

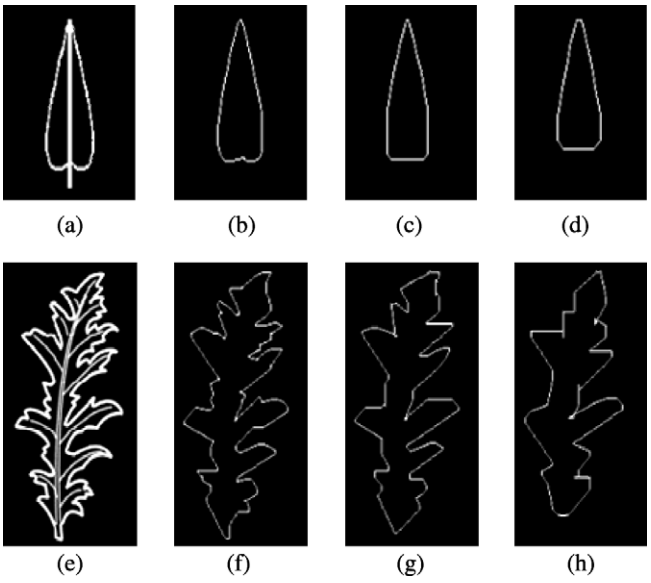

Fig. 3. Examples of image segmentation using MPP: (a) and (e) are original images. (b) and (f), (c) and (g), (d) and (h) are results, respectively, when the cell size is 2, 3, 5.

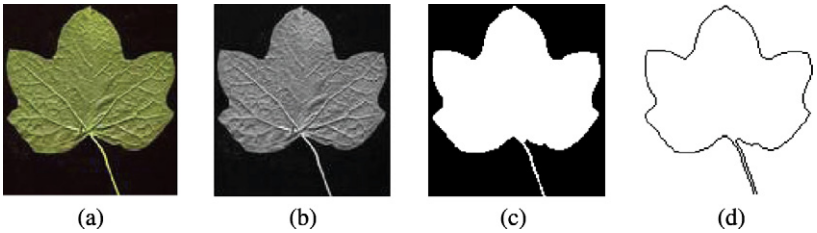

Fig. 2. Object extraction steps: (a) original leaf image. (b) gray-scaled image. (c) image masking. (d) contour extraction.

Algorithm 1. MPP algorithm.

1. Obtain the cellular complex.
2. Obtain the region internal to the cellular complex.
3. Use function boundaries to obtain the boundary of the region in Step 2 as a 4-connected, clockwise sequence of *coordinates*.
4. Obtain the Freeman chain code of this 4-contected sequence using *fchcode* function.
5. Obtain the convex (black dots) and concave (white dots) vertices from the chain code.
6. Form an initial polygon using black dots as vertices, and delete any white dots that are outside this polygon from further analysis.
7. Form a polygon with the remaining black and white dots as vertices.
8. Delete all black dots that are concave vertices.
9. Repeat steps 7 and 8 until all changes cease, at which time all vertices with angles of 180°are deleted. The remaining dots are the vertices of the MPP.

When an image contains many straight lines along the boundary, its segmentation using MPP may include many useless points and result in poor performance with regard to shape matching. In order to overcome this problem, points are merged along the boundary if their angle exceeds a specific threshold. Algorithm 2 presents details of the point merging algorithm.

Algorithm 2. Point merging algorithm.

```
Input
    point []: a set of points in x and y coordinates
    N: number of points;
    threshold: angle boundary for merging;
Output
    result[]: a set of points
Method
    get_distance(): calculates the distance between
    two points;
    add_point(): add a point into a point array;
function find_sequence(point, N, threshold)
{
    for (i = 1; i < N; i++) {
        a = get_distance(point[i-1], point[i+1]);
        b = get_distance(point[i], point[i-1]);
        c = get_distance(point[i], point[i+1]);
        angle = acos((b^2+c^2-a^2)/(2*b*c));
          if (angle < threshold)
              add_point(result, point[i]);
    }
    return result;
}
```

In this algorithm, a, b, and c are the sides of a triangle. Let the angle opposite the side a be A. Then, cosine A can be defined as follows:

Table 1
An example of point merging

Point i	x_i	y_i	angle(°)	action
0	51	16	108	
1	55	16	108	
2	60	31	176	merging
3	65	51	180	merging
4	70	71	166	merging
5	70	85	135	
6	65	90	135	
7	41	90	135	
8	36	85	135	
9	36	71	166	merging
10	41	51	180	merging

$$\cos A = \frac{b^2 + c^2 - a^2}{2bc} \tag{1}$$

Side a can also be calculated from coordinates of B and C.

$$a = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{2}$$

where (x_i,y_i) and (x_j,y_j) are coordinates of B and C, respectively.

Table 1 and Fig. 4 present the results when the points of the segment are merged with the threshold set at 160 degrees.

For the invariance, angles are adjusted with respect to the longest distance between two points, then left, right, top, and bottom points are detected for scale invariance, as presented in Algorithm 3. Fig. 5 presents the steps for adjusting the original image.

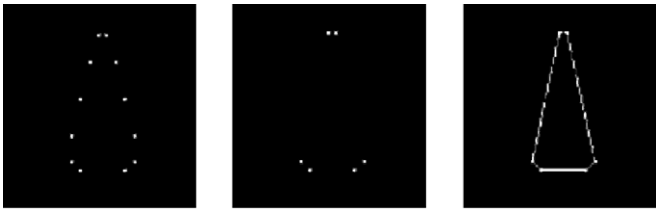

Fig. 4. Reduction of points by point merging.

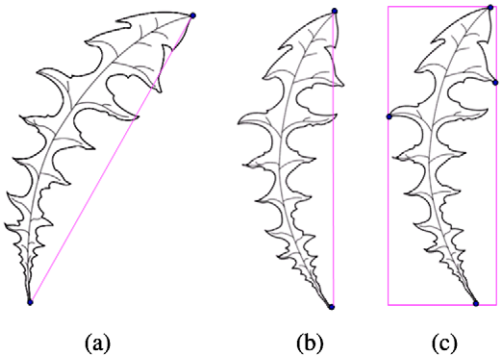

Fig. 5. Image adjustment based on viewing angle and scale: (a) original image. (b) rotational adjustment. (c) 4 edge point detection for scale invariance.

Algorithm 3. Rotational adjustment algorithm

```
Input
    crt1, crt2: two points of Criterion
    N: number of points
 function rotating_point (point, N, crt1, crt2) {
    cos = (crt2.X-crt1.X)/get_distance(crt1, crt2);
    sin = sqrt(1- cos∧2);
    for (i = 0; i < N; i++) {
        point[i].X= cos*(point[i].X-crt1.X)+sin*
(crt1.Y- point[i].Y)-crt1.X;
        point[i].Y=  sin*(point[i].X-crt1.X)-cos*
(crt1.Y-point[i].Y)- crt1.X;
        if (crt1.Y <= point[i].Y)
            add_point(ptlist, point[i]);
    }  return ptlist;
}
```

3.3. Leaf arrangement and venation representation

In order to improve search performance, we use a hybrid-search scheme that utilizes both shape feature and leaf arrangement and venations. Fig. 6 presents different types of leaf arrangement and venations. We consider four different leaf arrangements in this paper: alternate, opposite, verticillate, and radical. While the alternate is an arrangement with one leaf attached at each node, the opposite is an arrangement with two leaves attached on the opposite side of the stem. The verticillate is an arrangement with three or more leaves attached at each node.

There are two types of venations: feather venation and parallel venation. Their characteristics are as follows:

- *Feather venation:* veins arise pinnately from a single mid-vein and are subdivided into veinlets. These, in turn, form a complicated network. This type of venation is typical for dicotyledons.
  - *Pinnately venationed:* the leaf has usually one main vein with veinlets, smaller veins branching off laterally, usually somewhat parallel to each other.
  - *Palmately venationed:* several main veins diverge from near the leaf base where the petiole attaches, and radiate toward the edge of the leaf.
- *Parallel venation:* veins run parallel from the base to the apex. Commissural veins (small veins) are connected to the major parallel veins.

In this paper, we consider venation types 5–8 because venation types 9–11 are similar to venation type 5 and venation type 12 is similar to venation type 8.

Leaf arrangement and venation are extracted from user-sketched images based on the phyllotaxy and number of venations per node. The phyllotaxy is defined as the distribution of leaves in each node, taking into consideration the number of leaves in each node, their position and their distribution along the stem.

To decide the type of leaf venation, we first construct a weighted graph G for the leaf venation. The graph G consists of two sets: a set of vertices V representing intersection and end points of the leaf and a set of edges E connecting vertices in V. An edge in E corresponds to a connection between corresponding two points in the venation. Also, each edge is assigned a number representing the normalized distance between the connected vertices. Typically, a graph can be represented using an adjacency matrix. The adjacency matrix for G is a $|V| \times |V|$ array, say A, with $A(i,j) = A(j,i) =$ distance of edge(i, j) if there exists an edge between i and j in G. $A(i,j) = 0$ if there is no such edge in G. Fig. 7 shows graphs for various leaf venations.

Especially, Fig. 7(e) shows the adjacency matrix for the graph in Fig. 7(d). Also, in the case of Fig. 7(b) and (c), the circumference C of an ellipse is computed in order to calculate the length of the lateral vein.

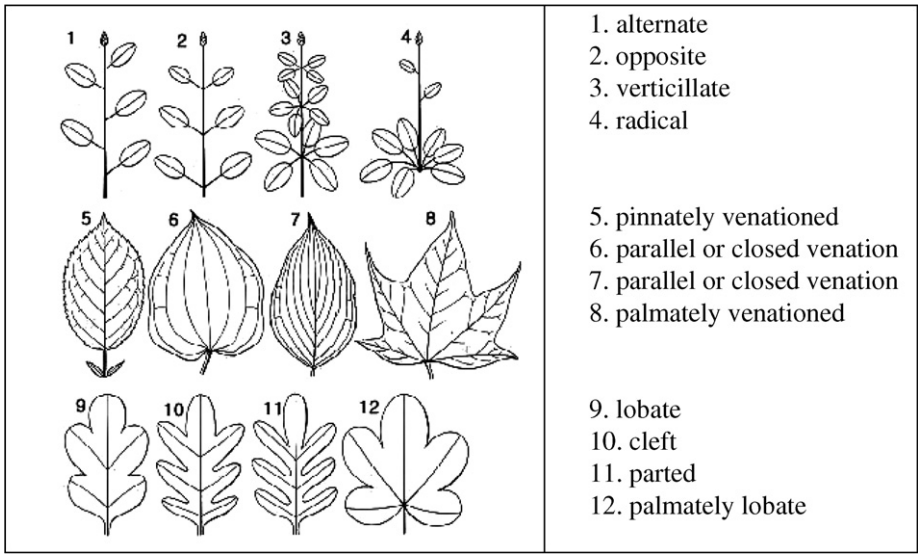

Fig. 6. Arrangements of leaves (1–4) and venations (5–12).

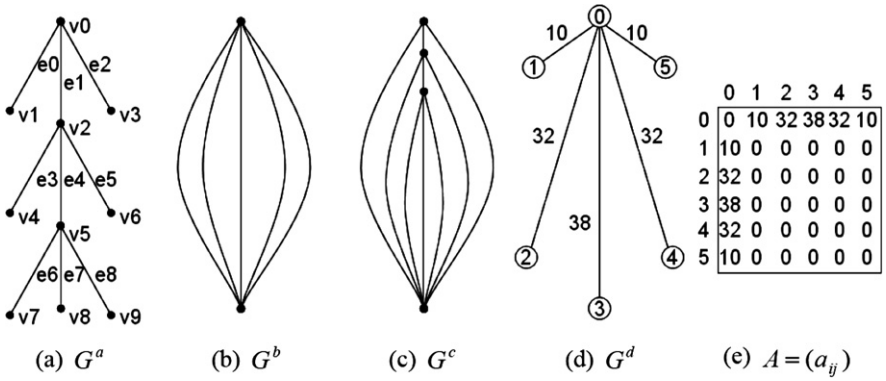

Fig. 7. Graph representation of various venations: (a) pinnately venationed. (b) and (c) parallel or closed venation. (d) palmately venationed. (e) adjacency-matrix for G^d.

$$C = 2\pi\sqrt{\frac{a^2+b^2}{2}} \tag{3}$$

where the x-axis along the major axis has a length of $2a$, and the y-axis along the minor axis has a length of $2b$.

For the venation extraction, intersection and end points of the venation are first detected. As an example, in Fig. 8, the venation extraction algorithm detects intersection points which are 1, 3, 18, 5, 14, 7, 16, 9, 19 (top-bottom) and end points which are 11, 12, 17, 15, 13, 0, 2, 4, 6, 8, 10 (clockwise). In addition, its adjacency matrix can be constructed as in the figure. If the venation has a tree structure as in Fig. 7(a) and (d), its adjacency matrix contains many 0s, which leads to waste of space. To improve the space efficiency, we can use an alternative representation for tree-like venations. The alternative representation, which we call *string representation* in this paper, represents the node and distance information of the graph into a nested list. Let T be a tree with immediate subtrees $T_1,\ldots,T_k$. Then the string notation of T, denoted by $ST(T)$ is defined inductively by

$$\begin{aligned} ST(T) &= NodeID(edge(T,T_1):ST(T_1),\ldots,\\ &\qquad edge(T,T_k):ST(T_k))if\ \ T\ is\ not\ a\ leaf\ \ node.\\ ST(T) &= NodeID\ if\ \ T\ is\ a\ \ leafnode. \end{aligned} \tag{4}$$

As an example, for the tree G^d in Fig. 7, its string representation $ST(G^d)$ is 0(10:1, 32:2, 38:3, 32:4, 10:5).

Image matching is a crucial step in content-based image retrieval and is usually achieved by measuring the distance between corresponding feature vectors. For the efficient search of similar images, many image retrieval systems extract features from the images and index these features using easily searchable descriptors. In this section, we describe an efficient dynamic matching method for generating ranks of all database images in an approximate order of similarity to the query image. Typically, a similarity query is defined as retrieving the most similar data object. In the case of image databases, a similarity query is to

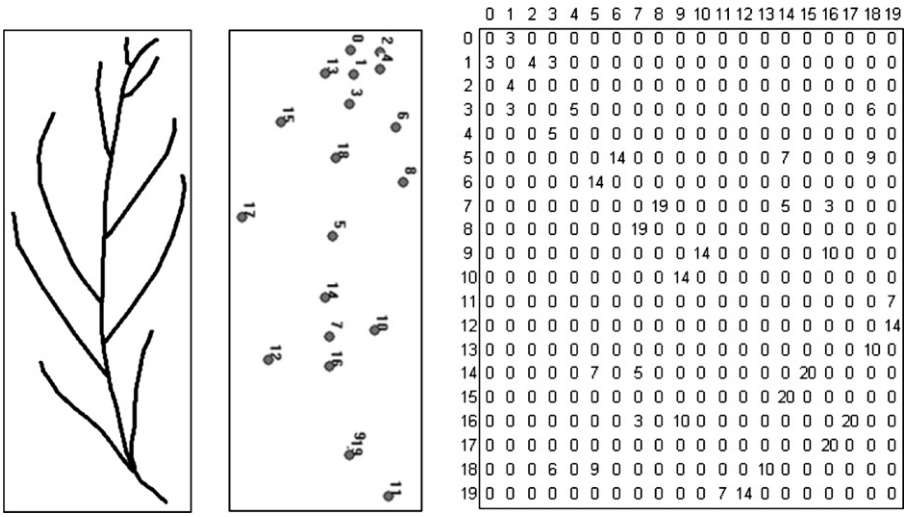

Fig. 8. Venation points extraction and matrix representation.

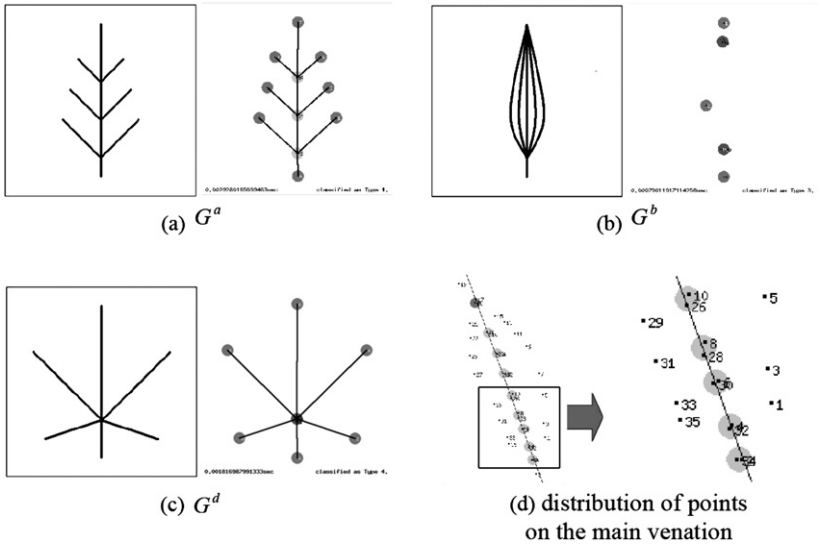

Fig. 9. Venation classification.

retrieve the most similar images to a given image with respect to the given features.

4. Image matching and retrieval

Image matching is a crucial step in content-based image retrieval and is usually achieved by measuring the distance between corresponding feature vectors. For the efficient search of similar images, many image retrieval systems extract features from the images and index these features using easily searchable descriptors. In this section, we describe an efficient adaptive matching method for generating ranks of all database images in an approximate order of similarity to the query image. Typically, a similarity query is defined as retrieving the most similar data object. In the case of image databases, a similarity query is to retrieve the most similar images to a given image with respect to the given features.

4.1. Similarity measure

Generally, similarity between two objects is measured by simply evaluating the Euclidean distance [23] between their corresponding points. Let point sets $U = \{u_1,\ldots, u_n\}$ with $u_i = (u_x^i, u_y^i)$ and $V = \{v_1,\ldots,v_m\}$ with $v_j = (v_x^j, v_y^j)$ be the query and a database image, respectively. The distance $D_{ij}(U,V)$ between the two point sets is given by the equation.

$$D_{ij}(U,V) = \sqrt{(u_x^i - v_x^j)^2 + (u_y^i - v_y^j)^2} \quad (5)$$

Also, the minimum distance $D_i(U, V)$ between two point sets is given by the equation.

$$D_i(U,V) = \min\{D_{i1}(U,V), D_{i2}(U,V), \ldots, D_{im}(U,V)\} \quad (6)$$

Based on the Euclidean distance, similarity between query and database image can be defined as follows:

$$S(U,V) = \sum_{i=1}^{n} D_i(U,V) \quad (7)$$

If the brute-force algorithm is used, the time complexity T is $O(|U||V|) = O(nm)$ to search the shortest path between U and V. For the linear time complexity, the $\varepsilon - nearest\text{-}neighbor(\varepsilon - NN)$ search algorithm is used where the time complexity is O(Dpolylog(N)) [24].

4.2. Venation classification

As mentioned in the Section 3.3, venation-based classification can be used to improve the retrieval efficiency and accuracy. The venation classification uses neural networks trained with information obtained through subjective experiments. In order to classify venations, a main venation should be acquired from intersection points.

Fig. 9 shows an snapshot of the venation classification and Algorithm 4 describes how to classify venations. Venations can be classified by considering the number of points and the distribution of points related to each point on the main venation as shown in Fig. 9(d). The venation similarity is measured by evaluating the Eq. (7).

Algorithm 4. Venation Classification Algorithm

```
Input
        CssPoint: points from CSS computation
function Classify_venation (CssPoints)
{
    CssBPs = GetCssBPs(CssPoints);// get CSS
      branching points
    PrimaryVP = GetPrimaryVein-
    Points(CssBPs);//
      get primary vein points
    CoefsOfVeinVect = GetPrimaryVeinLine
      (PrimaryVP);// get primary vein axis
```

```
    CoefsOfVeinNormalVect// slope and
      y-intersection of normal line
        = GetNormalLine(PrimaryVP,
          CoefsOfVeinVect);
    HorizontalDistances// distances of CssBPs
      from primary vein
        = GetDistanceFromLine( CssBPs,
          CoefsOfVeinVect );
    VerticalDistances// distances of CssBPs from
      normal line
        = GetDistanceFromLine( CssBPs,
          CoefsOfVeinNormalVect );
    HorizontalDensity// horizontal density of CssBPs
        = GetDensity ( HorizontalDistances );
    VerticalDensity// vertical density of CssBPs
        = GetDensity ( VerticalDistances );
    HorizontalLocalMaxs// local maximums in
      HorizontalDensity
        = GetLocalMaximums( HorizontalDensity );
    VerticalLocalMaxs// local maximums in
      VerticalDensity
        = GetLocalMaximums( VerticalDensity );
    if (IsParallelVein( VerticalLocalMaximums ) )
        return ParallelWithoutPrimaryVein;
    RelatedBPCnts// No. of related BPs in three
      types: horizontal, vertical, both
        = GetRelatedBPCnts(CssBPs,
          HorizontalLocalMaxs, VerticalLocalMaxs);
    if (RelatedBPCnts[Both] >=
      RelatedBPCnts[Vertical]
       and RelatedBPCnts[Both] >=
          RelatedBPCnts[Horizontal] )
        return Palmate;
    else if (RelatedBPCnts[Vertical] >=
      RelatedBPCnts[Horizontal])
        return Parallel;
    else
        return Pinnate;
}
```

4.3. Shape matching

Even though the time complexity of $\varepsilon -$ *nearestneighbor*$(\varepsilon - NN)$ searching algorithm is linear, it may take a considerable amount of time to match images for a large database. In order to reduce the matching time, we propose a new adaptive matching algorithm. This algorithm is based on a simple observation that a typical leaf shape has a roughly symmetric distribution. Symmetry can occur in any orientation as long as the image is same on either side of the central axis. Using this property, matching scope on the shape can be reduced by $1/2 \times 1/2 = 1/4$ with respect to full matching range. In addition, the matching process may stop when the accumulated similarity value is greater than a predefined threshold.

Even with this reduction scheme, the number of points of interest might be huge for complicated images. To reduce the number of points of interest further, we developed a new function called SMP, which is based on the sampling methodology. Let $|u|$ and $|v|$ be the number of points of interest extracted from the query image and database image, respectively. If $|u|$ is less than $|v|$, the number of interest points can be reduced by $|v|/|u|$ when the $SMP(v)$ function is used. Algorithm 5 below describes the adaptive matching algorithm that reduces the dimension space of the matching scope. Consequently, it reduces the matching time.

Algorithm 5. Adaptive matching algorithm

```
function Adaptive_matching(input_image, db_image,
N, threshold)
{
        input_points = GetCondensingPoints
(input_image);
        db_points = GetCondensingPoints
(db_image);
        if (sizeof(input_points) < sizeof(db_points))
            SMP(db_points);
        for (i = 0; i < N/2; i++) {
            NN_points = NN_search(input_points[i],
db_points);
            sim = MeasureSimilarity   (input_point[i],
NN_points, N/2);
            if(sim > threshold){
                sim = -1;
                break;
            }
        }
         return result;
}
```

We also implemented a grid-based matching scheme into our system to support faster access to the records and handle range queries more efficiently. Most techniques for storing and searching objects hierarchically decompose the multidimensional space into regions. Such a space can be either the data to be stored or the embedding space from which the data is drawn. Grid-based matching adopts this approach. They are especially appropriate when the domains over which the attributes take their values are large and linearly ordered, and the attributes are independent. Under such assumptions, grid-based matching guarantees high data storage utilization, smooth adaptation to the contents to be stored, fast access to individual record, and efficient range query processing [25].

The basic idea of grid-based shape matching is to divide the search space into a grid where each record is represented as a point [25]. More specifically, the record space S is partitioned into blocks, called *grid regions*. The fixed-size physical storage unit is called *cluster*: there is a cluster in correspondence with each grid region, while it is possible that several grid regions are associated with

the same cluster. *Grid directory* is a data structure that provides correspondence between grid regions and clusters: for a k-dimensional search space, a grid directory consists of a dynamic k-dimensional array, called *grid-array*, and k one-dimensional arrays, called *linear scales*. Elements of the grid array are pointers to clusters, and are in correspondence with the grid regions of the grid partition. Linear scales, one for each dimension of the space, contain the bounds of the partition on the corresponding axis. The efficiency of grid arrays is based on the assumption that the linear scales are relatively small and can be maintained in the main memory [25].

In grid-based matching, there are two kinds of searching algorithms: *nearest neighbors searching within a threshold* (*NNWT*) and *nearest neighbors searching (NN)*. *NNWT* can be described as follows: given a record r with valid dimension v and threshold d, it retrieves records comparable with r (i.e. with valid dimension v), which belong to the sphere centered at r with radius d. The idea of the algorithm is quite simple: it accesses, starting from the cluster containing r and moving off in spiral, all the clusters intersecting the sphere with center r and radius d. In order to terminate the search, the algorithm computes the number of steps to be performed before starting the computation.

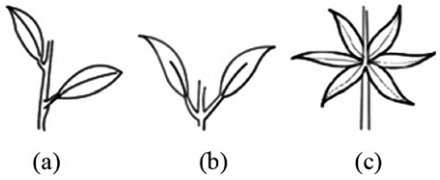

Fig. 10. Leaf arrangement used in the experiment.

Nearest neighbors searching (NN) can be described as follows: given a record r with valid dimension v, it retrieves the record nearest to r with the same valid dimension. The algorithm can be sketched as follows: given a record r with valid dimension v, the algorithm finds one neighbor r' of r with valid dimension v; then, it computes the actual distance d from r to r' and issues a new query centered at r with radius d. In this paper, we used the *nearest neighbors searching* method in grid-based matching.

5. Experimental results

In this paper, we have implemented a prototype leaf image retrieval system named CLOVER as part of a nationwide project aiming to construct an information bank for all domestic aqua plants. We integrated most algorithms described in this paper into the system and carried out various experiments to evaluate its performance. In the experiments, we used PCs with Dual 2.8 GHz Xeon Processors and 1GB of RAM as hardware platform, and Microsoft SQL Server 2000 as underlying DBMS. In addition, a PDA (CPU: PXA270 CLOCK: 624 MHz) was used as a mobile user interaction device and the query interface was implemented using the C# and Java languages.

In order to demonstrate the effectiveness of CLOVER, 1032 leaf images in the "*Illustrated flora of Korea*" [26] were collected and compared with other methods including Centroid Contour Distance (CCD), Fourier Descriptor, Curvature Scale Space Descriptor (CSSD), Moment Invariants, and MPP.

In addition, we evaluated the effect of considering venation feature together and compared with typical contour

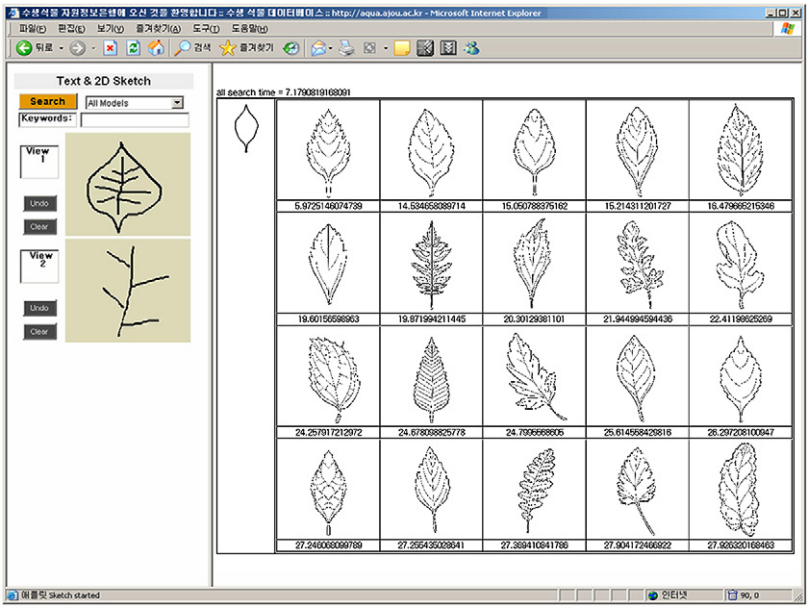

Fig. 11. A screenshot of web based interface.

Table 2
Venation classification results: recall and precision

Venation Arrangement	Retrieved Images	Relevant Images	Relevant Images in DB	Recall	Precision
G^a	185	179	198	0.90	0.97
G^b	17	17	18	0.94	1
G^d	12	7	14	0.86	0.58

Table 3
Average retrieval response time in seconds

Cell size	Response Time		A/B
	NN-search (A)	Adaptive matching (B)	
5	29.57	13.57	2.18
7	16.58	7.26	2.28
9	12.45	5.80	2.15

only-based retrieval. Fig. 10 presents a variety of leaf arrangements, classified into (a) alternate, (b) opposite, and (c) verticillate. Fig. 11 shows the user interface on the web with sample sketch and its result.

In order to evaluate the accuracy of our venation classification scheme, we measured the recall and precision for typical venation arrangements. The recall and precision are defined below. Table 2 shows the results of our venation classification scheme. In the table, each row shows the recall and precision for one of the venation arrangements. As shown in the table, the proposed classification algorithm achieves 90% recall and 85% precision on the average, approximately.

$$Recall = \frac{relevant\ images\ retrieved}{relevant\ images\ for\ the\ query\ in\ the\ database} \quad (8)$$

$$Precision = \frac{relevant\ images\ retrieved}{all\ imsages\ retrieved} \quad (9)$$

Fig. 12 compares the recalls and precisions of the revised MPP, MPP, Fourier Descriptor, CSSD, CCD, and Moment Invariants. The precision is a fraction of retrieved images relevant to a query. In contrast, the recall measures the fraction of the relevant images that have been retrieved. The recall is a non-decreasing function of rank, while precision can be regarded as a function of recall rather than rank. In general, the curve closest to the top of the chart indicates superior performance. In this figure, the proposed algorithm achieves approximately 25% improved precision and recall over MPP. In addition, precision and recall of the proposed algorithm is 2.11 times improved compared to that of the Fourier Descriptor.

Table 3 illustrates the average response time of the NN-search and adaptive matching with different cell sizes. From the table, it can be observed that regardless of matching method used, the response time decreases as the cell size increases. Overall, the proposed method achieves approximately 2.2 times better response time than the NN-search.

In order to evaluate the grid-based matching algorithm, we executed five queries on a database of 422 leaf images and 26,455 records. Each search test has been repeated 100 times to obtain reasonable mean values. The numbers of points in the query images are 31, 25, 23, 11, and 12, respectively. (see Table 4).

Table 5 shows the number of loop iterations in the greedy matching algorithm and the grid matching algorithm. As shown in the table, the grid matching algorithm is about 7.7 times less than the greedy matching algorithm in terms of loop iterations.

Table 6 shows the response time for the five queries. As shown in the table, the grid matching algorithm achieves approximately 2 times better response time than the greedy matching algorithm.

Fig. 13 compares the response times of greedy algorithm and grid-based algorithm for three different cell sizes. The

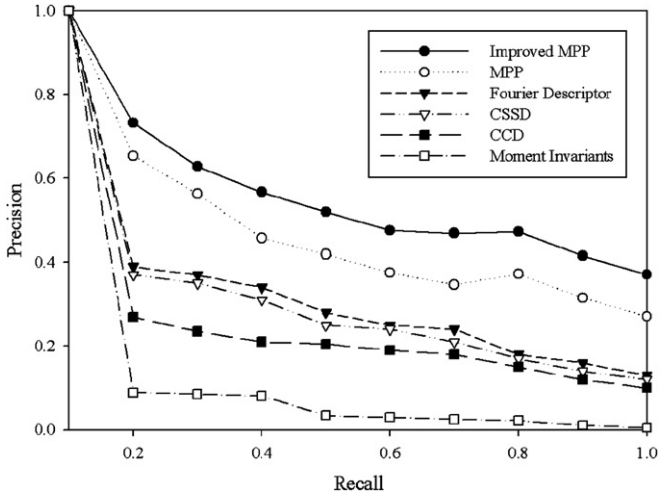

Fig. 12. Precision vs. recall curve.

Table 4
The numbers of points in the query images

Point i	Query 1		Point i	Query 2		Point i	Query 3		Point i	Query 4		Point i	Query 5	
	X[i]	Y[i]		X[i]	Y[i]		X[i]	Y[i]		X[i]	Y[i]		**XH**	Y[i]
1	0	62.674	1	469	293	1	490	268	1	456	248	1	456	248
2	0	68.245	2	431	303	2	485	267	2	225	149	2	26	249
3	132.313	151.811	3	59	209	3	312	345	3	54	247	3	225	149
4	132.312	146.24	4	255	171	4	407	267	4	435	237	4	54	247
5	132.312	158.774	5	341	241	5	232	344	5	431	240	5	435	237
6	139 276	172 702	6	302	192	6	331	264	6	131	247	6	431	240
7	153.203	193.593	7	380	278	7	148	340	7	438	247	7	131	247
8	167.131	207.521	8	373	256	8	253	264	8	217	247	8	438	247
9	167.131	201.95	9	421	306	9	97	198	9	434	262	9	217	247
10	167.131	214.485	10	325	327	10	174	261	10	430	259	10	434	262
11	188.022	228.412	11	351	316	11	50	291	11	437	252	11	430	259
12	194.986	235.376	12	204	306	12	176	261				12	437	252
13	215.877	242.34	13	215	301	13	91	322						
14	250.696	249.304	14	169	277	14	254	260						
15	346.797	249.304	15	405	302	15	153	183						
IS	388.579	242.34	16	158	248	16	331	260						
17	423.39!	235.376	17	227	242	17	235	176						
18	486.072	214.485	18	69	207	18	407	263						
19	500	207 521	19	240	242	19	312	174						
20	500	194.986	20	157	185	20	114	257						
21	430.362	104.457	21	398	298	21	53	227						
22	409.471	83.565	22	198	175	22	26	257						
23	402.507	76.602	23	275	199	23	110	260						
24	367.688	48.747	24	218	330									
25	305.014	6.964	25	277	316									
26	291.086	0												
27	215.877	0												
28	132.312	27.855												
29	118384	34 819												
30	111.421	41.783												
31	20.891	55.71												

Table 5
The number of iterations in the loop

Query	Greedy matching algorithm	Grid matching algorithm
Query 1	978,835	125,010
Query 2	661,375	86,383
Query 3	608,465	80,545
Query 4	291,005	38,941
Query 5	317,460	42,496

Table 6
The response time (milliseconds)

Query	Greedy matching algorithm	Grid matching algorithm
Query 1	70.1	31.3
Query 2	47.4	26.7
Query 3	43.1	24.3
Query 4	20.7	12.2
Query 5	22.5	11.9

top line represents the greedy algorithm and the lines below represent the response times of grid-based algorithm for different cell sizes. From the figure, we can observe the grid-based algorithm always gives better response time than the greedy algorithm.

Fig. 14 shows a few snapshots of QBS and QBP access to the CLOVER. The user can sketch a leaf image or upload a photographed leaf image. On the server, interesting points are sampled using the revised MPP and matched images are retrieved from the database based on the similarity to the query image. Finally, by selecting one from the result, the user can get its detailed information including the habitat information [20].

6. Conclusions and future work

In this paper, we presented an effective similarity-based leaf image retrieval system. One of the novel features of the system is that image retrieval is based on both shape and venation features. In order to improve the efficiency of leaf representation, we revised the MPP algorithm such that we can reduce the number of points of interest for matching. For matching, we proposed an adaptive grid-based matching algorithm and reduced the matching time. Furthermore, by using a hybrid-search scheme that considers both leaf arrangements and venation features, we

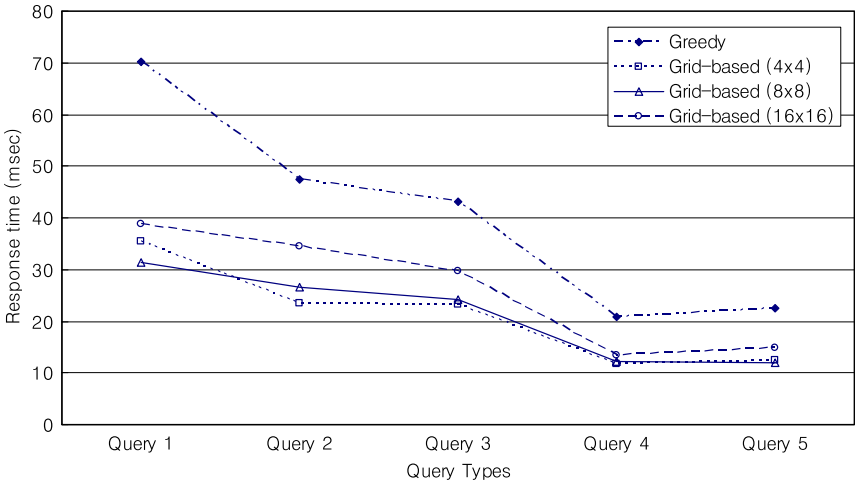

Fig. 13. Response time of greedy vs grid-based algorithm.

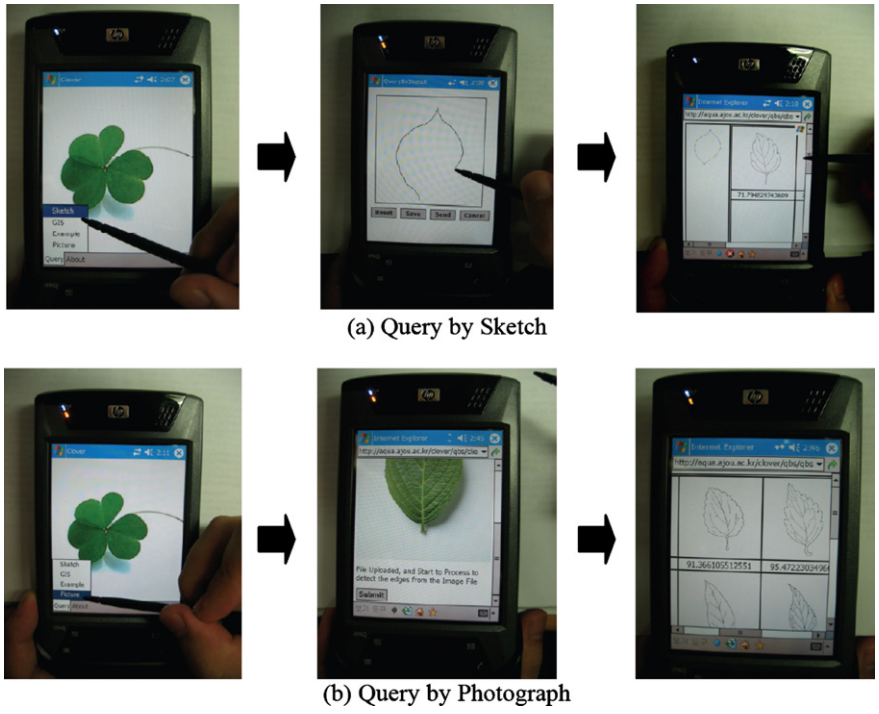

Fig. 14. Snapshots of QBS and QBP access to the CLOVER.

improved the overall system performance. In order to evaluate its effectiveness, we compared our proposed scheme with other methods including CCD, Fourier Descriptor, CSSD, Moment Invariants, and MPP. The experimental results demonstrate that our proposed scheme performs better than the other methods.

We are currently working on faster generation of the query preview. This is important in making our system scalable to larger data collections. We are also planning to study the effects of adding personalization, history, and relevance feedback functionality to the system, and investigate the efficacy of the method on text collections. Also, we plan to expand our prototype system to support more complex types of queries

Appendix A. Representative Subset of Leaf Images

See Figs. 15, and 16.

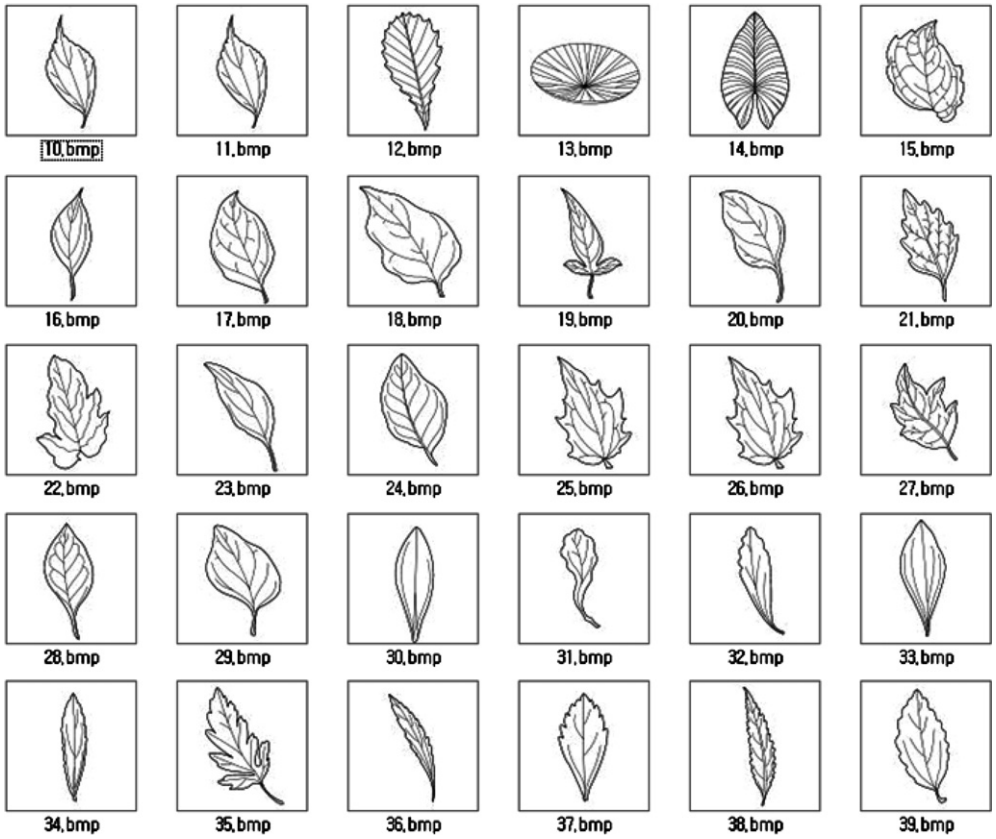

Fig. 15. Sample leaf images in the database.

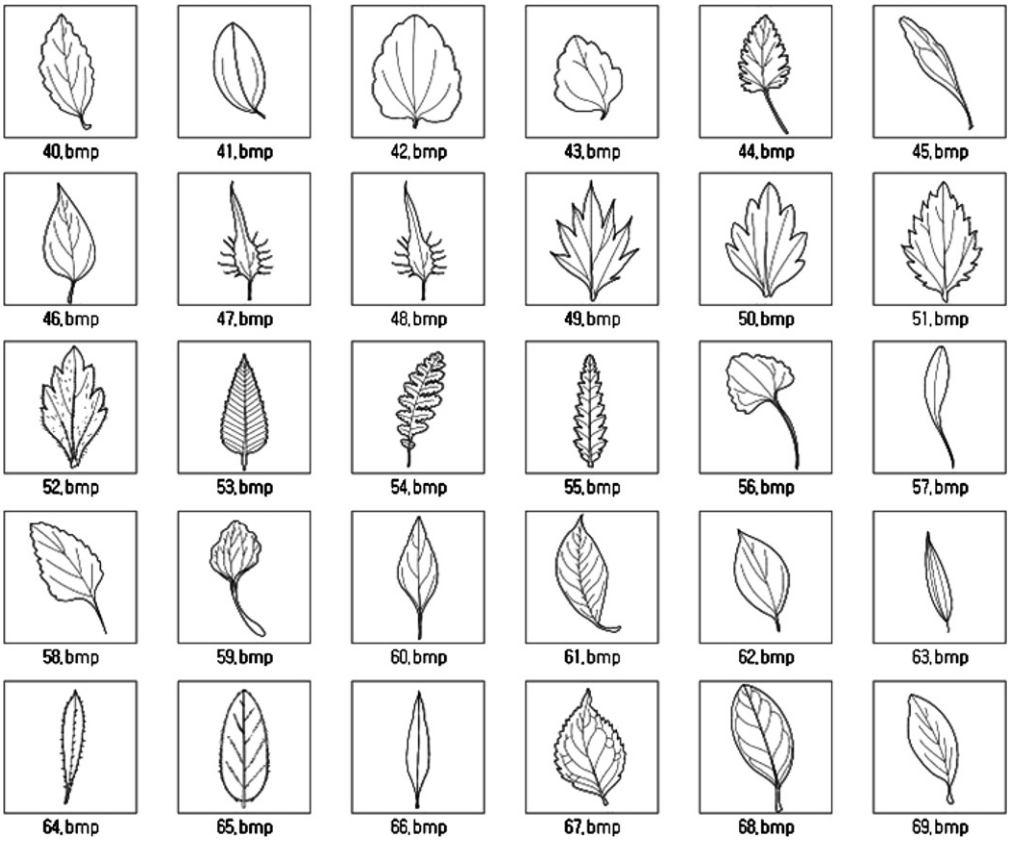

Fig. 16. Sample leaf images in the database.

References

[1] J.R. Bach et al., Virage Image Search Engine: An Open Framework for Image Management. SPIE Conf. On Storage and Retrieval for Image and Video Databases IV, San Jose, CA, 1996, pp.76–87.

[2] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos and G. Taubin, The QBIC Project: Querying Image By Content Using Color, Texture and Shape, Proceedings of SPIE Storage and Retrieval for Image and Video Databases, vol.1908, 1993, pp.173–187.

[3] A. Pentland et al., Photobook: Tools for Content-Based Manipulation of Image Databases, International Journal of Computer Vision 18 (3) (1996) 233–254.

[4] J.R. Smith and S.F. Chang, VisualSEEK: A Fully Automated Content-based Image Query System, ACM Multimedia'96, Boston, MA, November 1996, pp. 87–98.

[5] K.L. Kashyap, R. Chellappa, Stochastic Models for Closed Boundary analysis: Representation and Reconstruction, IEEE Transaction on Information Theory IT-27 (1981) 627–637.

[6] Y. Deng, A Region Based Representation for Image and Video Retrieval. PhD thesis, University of California, 1999.

[7] W.-Y. Kim, Y.-S. Kim, A Region-based Shape Descriptor Using Zernike Moments, Signal Processing: Image Communication 16 (2000) 95–102.

[8] G.J. Lu, A. Sajjanhar, Region-based Shape Representation and Similarity Measure Suitable for Content-based Image Retrieval, Multimedia Systems 7 (2) (1999) 165–174.

[9] C. Chang, L. Wenyin, and H. Zhang, Image Retrieval Based on Region Shape Similarity, 13th SPIE symposium on Electronic Imaging Storage and Retrieval for Image and Video Databases, 2001.

[10] Y. Kurozumi, W.A. Davis, Polygonal approximation by the minimax method, Computer Vision, Graphics and Image Processing, 1982, pp.248–264.

[11] J. Sklansky, R.L. Chazin, and B.J. Hansen, Minimum Perimeter Polygons of Digitized Silhouettes, TC(21), No. 3, March 1972, pp. 260-268.

[12] M. Han, D. Jang, The use of maximum curvature points for the recognition of partially occluded objects, Pattern Recognition 23 (1-2) (1990) 21–33.

[13] H. Sundar, D. Silver, N. Gagvani, S. Dickinson, Skeleton Based Shape Matching and Retrieval, Proceedings of Shape Modeling International, May 2003, pp. 130–142.

[14] C. Chang, L. Wenyin, and H. Zhang, Image Retrieval Based on Region Shape Similarity, Proc. 13th SPIE symposium on Electronic Imaging Storage and Retrieval for Image and Video Databases, January 2001.

[15] Z. Wang, Z. Chi, D. Feng, Q. Wang, Leaf Image Retrieval with Shape Features, Lecture Notes in Computer Science 1929 (2000) 477–487.

[16] F. Mokhtarian, S. Abbasi, Matching Shapes with Self-Intersections: Application to Leaf Classification, IEEE Transactions on Image Processing 13 (5) (2004) 653–661.

[17] P. Sollie, Morphological image analysis applied to crop field mapping, Image and Vision computing (2000) 1025–1032.

[18] H. Fu and Z. Chi, A two-stage approach for leaf vein extraction, Proceedings of International Conference on Neural Networks and Signal Processing, Vol.I, Nanjing, Jiangsu, China, December, 2003, pp. 208–211.

[19] Y. Li, and Z. Chi, and D. Feng, Leaf vein extraction using independent component analysis, IEEE International Conference on Systems, Man and Cybernetics, 2006, pp. 3890–3894.

[20] S. Kim, Y. Tak, Y. Nam, E. Hwang, mCLOVER: mobile content-based leaf image retrieval system, ACM Multimedia (2005) 215–216.

[21] H. Freeman and A. Saghri, Generalized Chain Codes for Planar Curves, Proceedings of the 4th International Joint Conference on Pattern Recognition, Kyoto, Japan, November 1978. pp. 701–703.

[22] C.T. Zahn, R.Z. Roskies, Fourier Descriptors for Plane closed Curves, IEEE Trans. On Computer C-21 (3) (1972) 269–281.

[23] R. Veltkamp, Shape matching: similarity measures and algorithms, Technical Report UU-CS-2001-03, Netherlands, 2001.

[24] P. Indyk, R. Motwani, Approximate nearest neighbors: towards removing the curse of dimensionality, The 30 annual ACM symposium on Theory of computing, 1998, pp. 604–613.

[25] J. Nievergelt, H. Hinterberger, K.C. Sevcik, The Grid File: An Adaptable Symmetric Multikey File Structure, ACM Transactions on Database Systems 19 (1984) 39–71.

[26] C.B. Lee, Illustrated flora of Korea. ISBN-8971871954, Hangmoonsa, (1999).

[27] J.E. Gary, R. Mehrotra, Similar shape retrieval using a structural feature index, Information System, Elsevier Science 18 (7) (1993) 525–537.