Pruning and Matching Scheme for Rotation Invariant Leaf Image Retrieval

Yoon-Sik Tak and Eenjun Hwang
School of Electrical Engineering, Korea University
136-701, Anam-dong Seongbuk-gu, Seoul, Republic of Korea
[e-mail: life993@korea.ac.kr, ehwang04@korea.ac.kr]
*Corresponding author: Eenjun Hwang

Received September 20, 2008; revised November 2, 2008; accepted November 10, 2008; published December 25, 2008

Abstract

For efficient content-based image retrieval, diverse visual features such as color, texture, and shape have been widely used. In the case of leaf images, further improvement can be achieved based on the following observations. Most plants have unique shape of leaves that consist of one or more blades. Hence, blade-based matching can be more efficient than whole shape-based matching since the number and shape of blades are very effective to filtering out dissimilar leaves. Guaranteeing rotational invariance is critical for matching accuracy. In this paper, we propose a new shape representation, indexing and matching scheme for leaf image retrieval. For leaf shape representation, we generated a distance curve that is a sequence of distances between the leaf's center and all the contour points. For matching, we developed a blade-based matching algorithm called rotation invariant - partial dynamic time warping (RI-PDTW). To speed up the matching, we suggest two additional techniques: i) priority queue-based pruning of unnecessary blade sequences for rotational invariance, and ii) lower bound-based pruning of unnecessary partial dynamic time warping (PDTW) calculations. We implemented a prototype system on the GEMINI framework [1][2]. Using experimental results, we showed that our scheme achieves excellent performance compared to competitive schemes.

Keywords: Shape-based image retrieval, dynamic time warping, sequence matching, rotation invariance, K-NN, range search

This work was supported by the Korea Science and Engineering Foundation (KOSEF) (2008-05074), Ministry of Education, Science and Technology, the Korean government.
Parts of this paper were presented at IEEE CIT 2007, Aizu, Japan, Oct. 18, 2007.

DOI: 10.3837/tiis.2008.06.001

1. Introduction

With the development of Information Technology and the popularity of digital devices such as digital cameras, a great number of digital images have been generated and shared. Considering their volume, an effective image retrieval scheme is critical for the satisfactory utilization of such images. We have observed two different approaches for image retrieval: annotation-based and content-based. The former has several problems such as being labor intensive, and lack of consistency due to the large amount of manual annotations involved. The latter provides an intutive query interface and satisfactory retrieval performance. Popular systems in this category include QBIC, Virage, RetrievalWare, Photobook, VisualSEEK, WebSEEK, Netra, MARS, and ART MUSEUM [3]. There are many different visual feature representation schemes for content-based retrieval such as a set of average values, a few feature points, or a sequence of data. For example, Kokare *et al.* designed a new set of two-dimensional (2-D) rotated complex wavelet filters (RCWFs) to represent texture information of images [4] and proposed a formulation of new texture-retrieval algorithm based on the filters. In [5],Tieu *et al.* proposed an image retrieval scheme based on a large number of highly selective features and efficient learning of queries. In particular, they proposed an algorithm to learn a classification function depends on a small number of the most appropriate features at query time. In [7][8][9], they represented the shape feature of an object into an approximated polygon using the minimum perimeter polygon (MPP) method. More related to our work, [6] calculated a distance sequence for each object to represent its shape feature and calculated a feature vector for matching purpose. In particular, for the scaling and rotation invariant matching, they normalized the feature vector first and then calculated the Euclidean distance to measure the similarity between feature vectors. In particular, for similar but locally out-phased sequences, dynamic time warping is known to be effective [2].

In this paper, we propose a new shape representation, indexing, and matching scheme for effective leaf image retrieval. We first represent a leaf shape using a distance curve, which is a sequence of distances between its center point and every point along the leaf contour. Many leaves consist of more than one blade, and their distance curves show almost same number of hops called unit curves. Based on this observation, we propose a new blade-based matching scheme which guarantees scaling and rotation invariance. In addition, to further accelerate the matching process, we propose two more techniques: i) priority queue-based pruning to avoid unnecessary computation for rotational invariance, and i) lower bound-based pruning to avoid unnecessary computation. For performance evaluation, we implement a prototype leaf image retrieval system on the GEMINI framework [1][2]. This framework has little chance of producing a false dismissal when we index sequence data. Under the framework, we choose a high level representation of the data and define a lower bounding measure on it. This technique is simple, intuitive, and competitive with more complex approaches. In the experimental results, we see that our scheme outperforms any other competitive method.

The outline of this paper is as follows. In Section 2, we introduce some related works about shape feature representation and sequence retrieval. In Section 3, we describe how to calculate distance curves from leaf images for shape representation. In Section 4, we explain our sequence matching algorithm, RI-PDTW, and show how to establish a lower bound measure and queue-based incremental evaluation for pruning. In Section 5, we present our matching framework based on the GEMINI framework. In Section 6, we describe some of the experiments performed to evaluate our scheme, and we conclude the paper in Section 7.

2. Related Works

In shape-based image retrieval, a crucial element for performance is good shape representation. Shape can be represented using either partial shape information or full shape information. The former approach has the advantages of fast matching and direct indexing. MPP [7][8][9], Fourier descriptor [10][11], and wavelet transform [12] belong to this category. The latter approach has the advantage of accuracy. However, this approach has a major drawback in that indexing performance could drop rapidly due to high dimensionality. To avoid this dimensionality problem, the whole shape is not directly indexed, but transformed into a low-dimensional feature [11]. Many dimensional reduction algorithms for sequence indexing are available such as fast Fourier transform (FFT) and discrete wavelet transform (DWT). These algorithms are known to be effective in Euclidean distance matching. However, for dynamic time warping (DTW) matching, these algorithms might give false dismissals since DTW does not satisfy the triangle inequality. To remove false dismissals, new dimensional reduction algorithms such as piecewise aggregate approximation (PAA) [2] and enhanced PAA [13] have been proposed.

To speed up sequence matching, several lower bounding functions have been suggested. A lower bounding function takes less computation time, while guaranteeing a shorter distance compared to the original matching method. Therefore, a lower bounding function can be used beforehand to prune some of the inevitable false hits. Examples include LB_Kim [14], LB_Yi [15], and LB_Keogh [16]. LB_Yi uses sequence points that are larger (smaller) than the maximum (minimum) of the other sequence. The squared difference between their values and the maximum (minimum) value of the other sequence can be the lower bound distance of DTW. LB_Kim extracts four-tuple feature vector which is the first and last element of the sequence, together with the maximum and minimum values. The maximum squared differences of corresponding features can be the lower bound distance of DTW. And LB_Keogh uses Upper and Lower envelops that are defined according to allowed range of warping for each point in a sequence. The difference of query envelops and data sequence is the lower bound distance of DTW.

A sequence indexing and matching scheme guaranteeing image rotation is another important factor in retrieval accuracy. If the same images are aligned differently, then sequence-based matching might consider that they are different. To guarantee rotation invariance in image retrieval, the matching method should consider the shapes of all the rotated images. In terms of sequence, this means that rotation invariance can be obtained by considering all possible circular shifts of the sequence. Therefore, a brute force way to enforce rotation invariance is to use rotational alignment. However, checking all possible cases takes $O(n^3)$ time at matching for a sequence of length n.

To avoid checking unnecessary cases, several heuristic techniques have been proposed. For instance, the mountain climbing algorithm calculates just one start point from a sequence of data [17]. However, these algorithms cannot guarantee the optimal path since they calculate the starting point using a heuristic algorithm. Another way to achieve rotation-invariant matching is by extracting rotation-invariant features and indexing them with a feature vector [18]. There are dozens of rotation-invariant features such as the ratio of perimeter to area, fractal measures, circularity, min/max/mean curvature, and entropy, to name a few. Also, Keogh *et al.* defines a general algorithm guaranteeing rotation invariance called “H-merge” [16] for sequence data. H-merge accumulates wedges from the rotated sequences of a sequence of data. Then, the nearest neighbor algorithm is applied to these wedges to obtain the optimal path.

Overall, based on the existing sequence indexing and matching schemes, in this paper, we propose a new blade-based rotation-invariant sequence matching algorithm for more efficient leaf image retrieval. Unlike most existing sequence retrieval schemes that require huge CPU computation time for the image rotation invariance, our scheme guarantees the image rotation in a very short time by considering blade-based image rotation only and pruning unnecessary blade sequences in matching. Also, we propose a lower bounding function, LB_RI-PDTW, for RI-PDTW for pruning purpose.

3. Preprocessing

In this section, we consider preprocessing to extract a distance curve from a leaf image. The first step is to detect the contour of a leaf image. Then, distances between the center point and the contour points are calculated. By accumulating these distances along the x-axis, we can generate a sequence as the shape of a leaf image.

3.1 Image Binarization

To detect a contour from a leaf image, we first detected edge information. For edge detection, we applied a well-known edge detecting algorithm, Canny edge detection, to the leaf image. **Fig. 1**(b) shows the edges detected from a leaf image in **Fig. 1** (a) using Canny edge detection. Using the edge information, we found the leaf contour. A detailed decription of contour detection is given in [19]. After detecting the contour, we converted it into a binary image by marking pixels within the edge with black, and marking background pixels with white. For example, the binarized image for **Fig. 1** (b) is shown in (c).

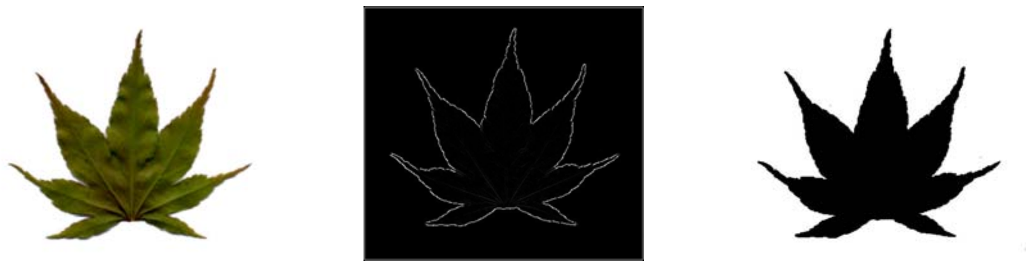


Fig. 1. Image preprocessing: (a) original image, (b) edges by Canny edge detection algorithm, and (c) image binarization.

3.2 Shape Descriptor and Normalization

We can easily generate a distance curve for a binary leaf image. First, we calculate its geometric center point by averaging the x and y coordinates of all the pixels inside the leaf. This point has the property that its relative position within the image does not change during rotation or scaling. A simple equation for calculating a geometric center point is as follows:

$$\left\{ x_c = \frac{1}{N}\sum_{i=1}^{N} x_i,\ \ y_c = \frac{1}{N}\sum_{i=1}^{N} y_i \right\} \qquad (1)$$

, where (x_c, y_c) is the center point of the leaf and x_i and y_i are 2-dimensional coordinates of pixels where N is the number of pixels inside the leaf image.

Then, we calculate the distance of all the points along the contour in a clockwise direction from the center point. Depending on the image size, two distance curves might show different magnitudes for the same leaf image. To avoid this problem, scaling invariance should be guaranteed in the matching. For this purpose, we rescale all the sequences to have some maximum magnitude *m*. For example, for a sequence *s*, we first find its maximum magnitude *n*. Then, the ratio of *m* and *n*, *m*/*n*, is multiplied to all the elements of the sequence. In this way, all the sequences are rescaled to have same maximum magnitude.

4. Blade-Based Matching

A leaf usually consists of more than one blade, and hence its distance curve consists of almost as many hops, called unit curves. In this section, we propose a blade-based sequence matching algorithm. We divide sequences into unit curves, and calculate the distance of sequences by accumulating the distances of their unit curve pairs.

4.1 Unit Curve Detection

As previously mentioned, a distance curve of a leaf can be divided into unit curves, each of which corresponds to a leaf blade. Each unit curve can be described by a triple (Min_Left, Max, Min_Right). For example, for the sample leaf image in **Fig. 2**(a), its distance curve consists of three unit curves which can be described by (P_1, P_2, P_3), (P_3, P_4, P_5), and (P_5, P_6, P_7).

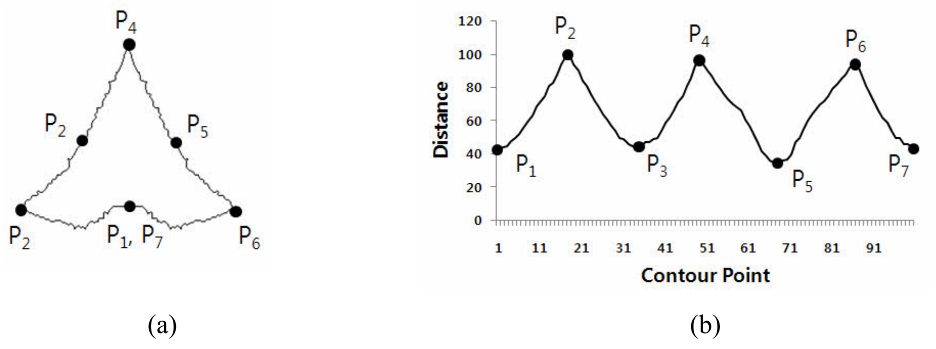


Fig. 2. Generating distance curve and identifying unit curves: (a) sample leaf image, and (b) its distance curve with three unit curves

A detailed algorithm for segmenting a sequence into unit curves is described in [20]. Depending on the starting point of a sequence, part of the first unit curve can be found after the last unit curve. In this case, they must be merged into a complete unit curve by moving the fragment after the last curve to the front. For instance, for a sample sequence in **Fig. 3 (a)**, we can merge two curve fragments into one complete unit curve by moving the last fragment to the front, as shown in **Fig. 3 (b)**. A detailed algorithm for merging the curves can be found in [20].

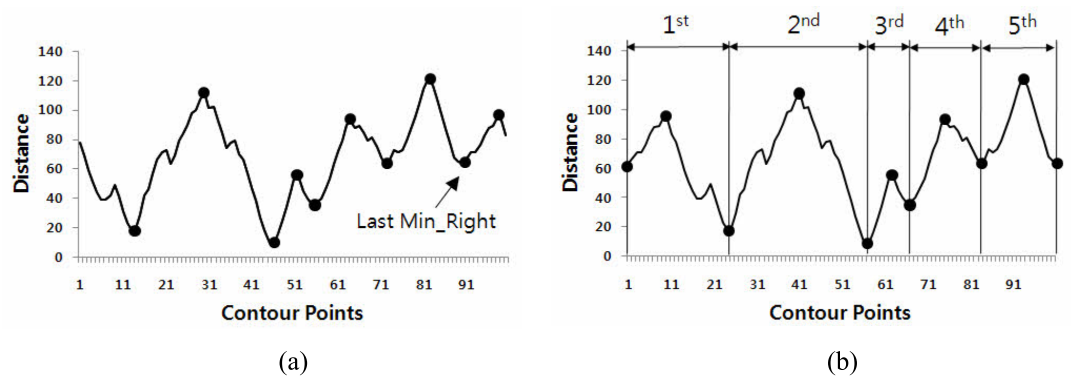


Fig. 3. Detecting unit curves from a sequence data: (a) finding all the max and min points, and (b) complete sequence of unit curves after rotate-shifting all the points after the last Min_Point.

4.2 PDTW

Since a distance curve usually consists of more than one unit curve, blade-based matching could be more efficient than whole-shape matching in the following aspects. First, the DTW algorithm restricts the warping window size to prevent pathological warping. Due to this restriction, unit curves cannot be matched correctly if the unit curve is longer than the warping constraint. To avoid this problem, we can expand the warping window size. However, according to [21], when the warping window size is restricted to 5%~10% of the window size, higher accuracy can be achieved. Thus, if we make the warping window wider, we cannot achieve high accuracy. This can be solved by dividing a sequence into unit curves and applying the matching method to the unit curve pairs. A detailed algorithm for this is shown in **Table 1**.

Table 1. Algorithm PDTW (Q, S)

```
1. dist = 0;
2. if NoC(S) == NoC(Q)      //NoC(S) indicates the number of unit curves in S
3.      for i = 1 to NoC(Q)
4.           dist = dist + DTW(ith blade of Q, ith blade of S);
5.      return dist
6. else
7.      return infinity
```

For instance, for the two sequences Q and S with three unit curves in **Fig. 4**, their PDTW distance is the summation of DTW distances of three unit curve pairs.

The difference between DTW and PDTW can be observed from their warping windows. For instance, **Fig. 5 (a)** and **(b)** show the warping windows of DTW and PDTW, respectively. Both use the same warping constraint. We see that DTW makes the wrong warping path from the first unit curve, and accumulates matching error to the end. Moreover, PDTW has a smaller warping window size than DTW. This is because the dimension of the warping window size for the whole sequence is always bigger than that for unit curves under the same warping constraint, which we prove in Theorem 1:

Theorem 1. Assume that we have two sequences of equal length n and warping constraint 10%. Furthermore, they are divided into k equal unit curves. Then, the warping window of

DTW is about 0.1 * n^2. However, for PDTW, the warping window of each unit curve is 0.1 * $(n/k)^2$. Since the sequence has k unit curves, the warping window size of PDTW is k * 0.1 * n^2/k^2. This means that the warping window size of DTW is about k times larger than that of PDTW.

In addition, a smaller warping window size gives the additional advantage of reduced memory size and CPU time during optimal path searching.

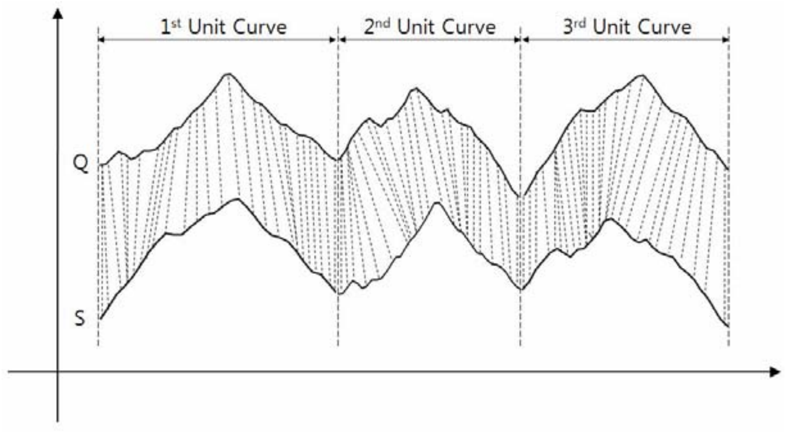
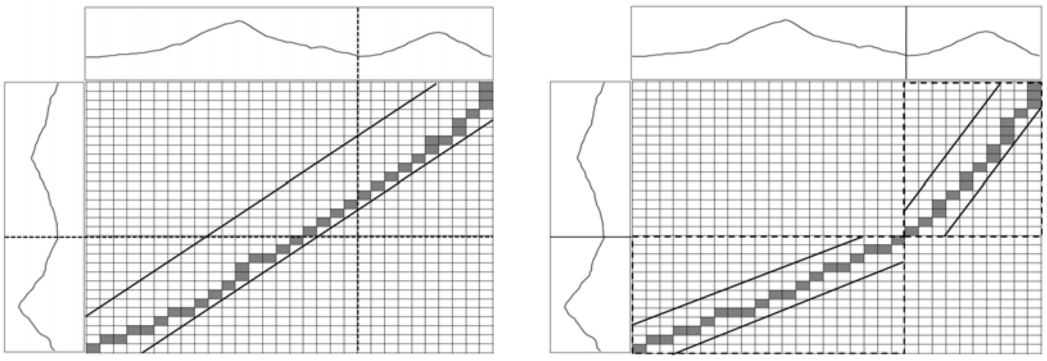


Fig. 4. Example of PDTW matching.

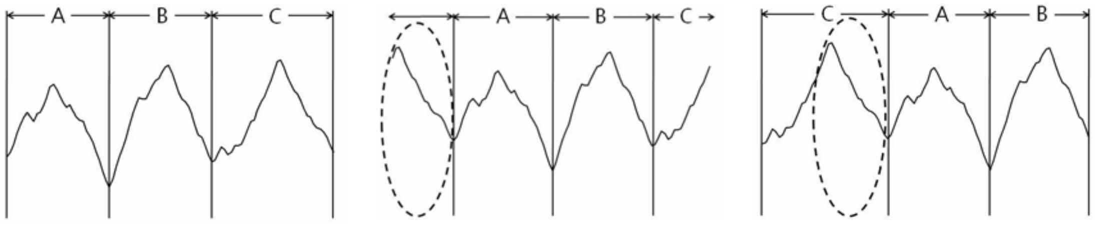


(a) (b)

Fig. 5. Warping paths of (a) DTW, and (b) PDTW.

4.3 Rotation Invariant PDTW (RI-PDTW)

We described how image rotation should be considered for a more accurate matching result. In this section, we describe how to guarantee rotation-invariance in PDTW-based matching. Since we consider blade-based sequence matching, we do not need to consider cases where part of a unit curve is shifted as in the brute force algorithm. Typically, in order to guarantee rotation invariance in sequence-based image retrieval, sequences are aligned to have the same starting point. To reduce the alignment overhead, some heuristic approaches have been proposed. Most heuristic approaches calculated starting point using a mathematical function. These approaches produced acceptable experimental results but they could not explain the approaches always produce optimal path. Unlike these approaches, our RI-PDTW can produce optimal path for any cases in partial dynamic time warping matching. An image and its rotated image have the same but shifted distance sequences, where the shift is propositional to the rotation. Partially shifted sequences are moved around so that their starting point is always the first point of a complete blade. Since our PDTW matching considers only the full blade shifts and calculates distances of blade pairs, it can guarantee rotation invariance. This enables to reduce the number of cases to consider and, in turn, reduce the matching time. For example, for the sample sequence with three unit curves shown in **Fig. 6 (a)**, **Fig. 6 (b)** shows a partially-rotated sequence where unit curve ‘C’ is partially shifted, and **Fig. 6 (c)** shows a fully-rotated sequence.



(a) (b) (c)

Fig. 6. Representation of rotated sequence data: (a) original sequence data, (b) intermediate state of shifting a blade, and (c) shifting one blade ‘C’ from the original sequence data.

We can implement rotation invariance into PDTW with a small modification as shown in **Table 2**. If two sequences Q and S have same number of unit curves, then we calculate and accumulate the DTW distances of matched unit curve pairs. Next, we shift the last unit curve of Q and repeat the calculation NoC(Q) times, keeping track of the minimum value. If two sequences have different number of blades, we can abort their matching by returning a value of infinity.

Table 2. Algorithm RI-PDTW (Q, S)

```
1. if NoC(Q) == NoC(S)
2.        for i = 1 to NoC(Q)
3.               $dist_i$ = PDTW(Q, S);
4.               shift last blade of the Q to the front;
5.        set Min_dist to the smallest value of $dist_i$ ($1 \le i \le$ NoC (Q));
6.        return Min_dist
7. else
8.        return infinity
```

4.4 Minimum Priority Queue-Based RI-PDTW

In the previous section, we described how to guarantee rotation invariance while reducing matching time compared to the brute force algorithm. In this section, we reduce the matching time further using minimum priority queue-based pruning. For two sequences with *n* unit curves, our RI-PDTW shifts one sequence circularly by unit curve, and calculates their DTW distance for every pair of unit curves. This requires n^2 DTW calculations. However, if the partial RI-PDTW distance of sequence Q′ to S is greater than the fully calculated RI-PDTW distance of sequence Q to S, we can safely prune Q′ because further calculation for Q′ will just increase its DTW distance. To keep track of partial matches with minimal distance, we use the minimum priority queue. A detailed algorithm for this is shown in **Table 3**. In the algorithm, the entry in the minimum priority queue consists of unit curve identifier *id*, the number of unit curves considered so far *nc*, and the accumulated distance *dist*. The algorithm works as follows:

1) Let $UC_1(S)$ denote the first unit curve of S. For each unit curve i of Q where $1 \leq i \leq NoC(Q)$, calculate the DTW distance *d* between $UC_1(S)$ and $UC_i(Q)$ and make an entry (i, 1, dist) into the queue (lines 1-4).
2) If the first entry in the queue considered all the unit curves, then return the *dist* of the entry as a result (lines 7-8).
3) Otherwise, calculate the DTW distance of the next unit curve pairs of Q and S and update its *nc* and *dist* accordingly (lines 9-13).

Table 3. Algorithm Queue-based RI-PDTW (Q, S)

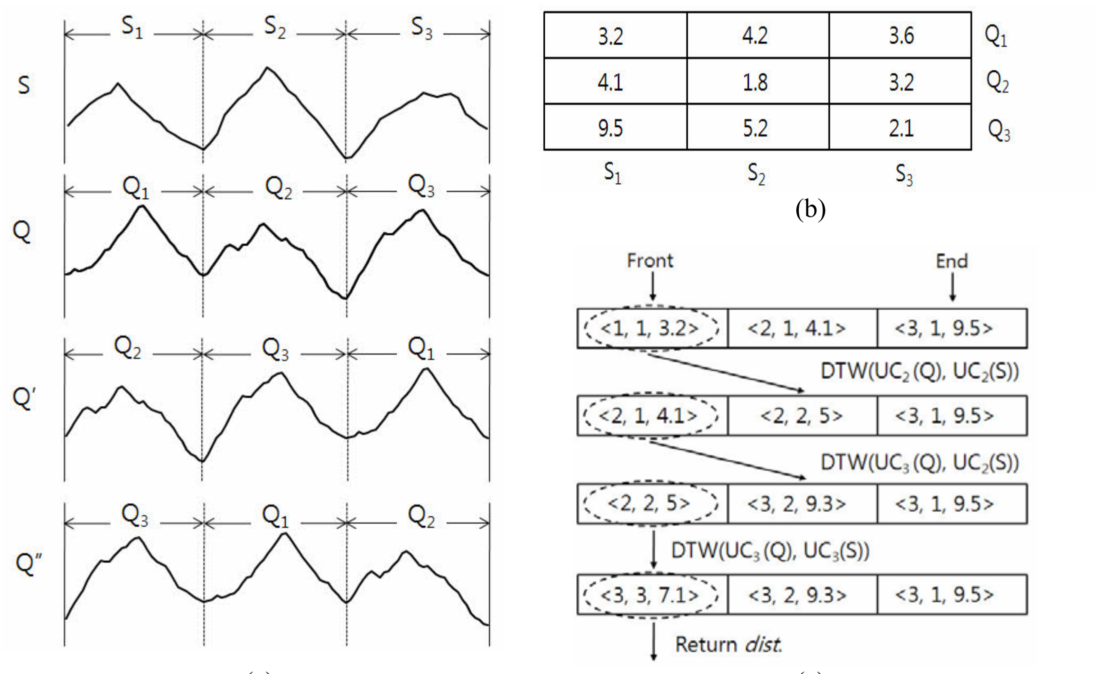
```
Variable queue : MinPriorityQueue;

1.   let $UC_1(S)$ denote the first unit curve of S
2.   for i = 1 to NoC(Q)
3.       dist = DTW($UC_i(Q)$, $UC_1(S)$);
4        queue.push(i, 1, dist);
5.   while not queue.IsEmpty() do
6.       top = queue.Pop();
7.       if (top.nc == NoC(Q))
8.               return top.dist;
9.       else
12.              d = DTW($UC_{top.id+1}(Q)$, $UC_{top.nc+1}(S)$);     //calculate circularly using mod
13.              queue.push(top.i+1, top.nc+1, top.dist+d);
```

For example, for sequences Q and S with three unit curves shown in **Fig. 7**, we need to consider all the rotated sequences of Q for rotation invariance. Since Q has three unit curves, we must consider three rotated sequences Q_1(=Q), Q_2, and Q_3.

1) For all rotated sequence Q_i of Q, calculate the DTW distance *d* between the first unit curve of Q_i and the first unit curve of S, and insert its entry (Q_i, 1, *d*) into the queue. This will create initial queue state (Q_i, 1, d) for i=1, ... , 3.
2) As the first element of the queue is < Q_1, 1, 3.2 >, add the DTW distance between the second unit curves of Q and S to the current distance. This will insert < Q_1, 2, 3.2+1.8 > into the queue.
3) As the first element of the queue is < Q_2, 1, 4.1 >, add the DTW distance between the second unit curves of Q_2 and S to the current the distance. This will insert < Q_2, 2, 4.1 + 5.2 > into the queue.
4) As the first element of the queue is < Q_1, 2, 5 >, add the distance between the second unit curves of Q and S to the current the distance. This will insert < Q_1, 3, (5 + 2.1) > into the queue.
5) The first element of the queue is < Q_1, 3, 7.1 >. Since we considered all the unit curves, we return its distance as the RI-PDTW distance for the whole sequence of Q.

This example shows why our priority queue-based RI-PDTW is faster than naïve RI-PDTW under our blade-based sequence matching. In the naïve RI-PDTW, DTW computation should be performed nine times since both Q and S have three unit curves. However, in our priority queue-based RI-PDTW, DTW computation was performed just six times.



3.2	4.2	3.6	Q_1
4.1	1.8	3.2	Q_2
9.5	5.2	2.1	Q_3
S_1	S_2	S_3	

(a) (c)

Fig. 7. Steps of priority queue based PDTW matching: (a) unit curves of sequences Q and S, (b) DTW distances between curves of Q and S, and (c) applying priority queue-based PDTW with minimum priority queue.

4.5 Lower Bounding Technique

In this section, we describe how to further reduce matching time using a lower bound function based on LB_Keogh. A low bound function takes less computation time and returns a smaller value than that of the original matching method. There are many well-known lower bound functions for DTW such as PAA, MINDIST, LB_Keogh, LB_Yi, etc. As a result, if a sequence S has a bigger lower bound distance than the actual distance of sequence S′, we can safely ignore the sequence S. In this paper, we use LB_Keogh as our lower bound function. However, rather than apply LB_Keogh to the whole sequence, we apply LB_Keogh to each unit curve and use their summation as the lower bound of the sequence. If query sequence Q and database sequence S have same number of unit curves, then we calculate and accumulate the LB_Keogh distances for matched unit curve pairs. Next, we shift the last unit curve of Q circularly and repeat the calculation NoC(Q) times, keeping track of minimum value. If they have a different number of unit curves, then we prune the sequence by returning a value of infinity. **Table 4** shows detailed description of this.

5. Matching Framework

For more efficient leaf image retrieval, we constructed an indexing structure for the distance curves of leaves. However, these sequences were too long and their index may give high dimensionality. Therefore, we first transformed the sequences into low-dimensional features using a discrete Fourier transform (DFT) as the dimensional reduction algorithm. Then, we constructed a tree-based index structure on the GEMINI framework. Based on this indexing structure, we revised the K-nearest neighbor (K-NN) and range search algorithms, and show how these algorithms perform under our proposed matching scheme.

Table 4. Algorithm LB_RI-PDTW (Q, S)

```
1.  if NoC(Q) == NoC(S)   // NoC(S) indicates the number of unit curves in S
2.      for i = 1 to NoC(Q)
3.          $dist_i = 0$;
4.          for j = 1 to NoC(Q)
5.              adjust $UC_j(Q)$ and $UC_j(S)$ to have same length;
6.              $dist_i = dist_i + LB\_Keogh(UC_j(Q), UC_j(S))$;
7.          shift  last unit curve of Q to the front;
8.      set LB_dist to the smallest value among $dist_i$ $(1 \le i \le NoC(Q))$
8.      return  LB_dist
9.  else
10.     return  infinity
```

5.1 Low-Dimensional Transform for Indexing

Constructing an index for a full sequence might not be efficient due to high dimensionality. Many dimensional reduction algorithms have been proposed and used, including discrete wavelet transform (DWT), discrete Fourier transform (DFT) and piecewise aggregate (PAA),. PAA is fast in indexing, and shows no false dismissals in DTW matching. However, it may not be appropriate for rotation-invariant image retrieval because it may return different values for the rotated sequence. Therefore, for rotation invariance, we used DFT to transform a sequence into low dimensional data using the first few coefficients. Also, we used R-tree structure for indexing them. In the index, leaf nodes contain the lower and higher endpoints of Fourier-transformed features (Fourier points) from sequences using a minimum bounding rectangle (MBR). These MBRs are recursively grouped in non-leaf nodes, and finally all the MBRs are grouped into the root.

Fig. 8 shows all the steps for constructing an index for leaf images in database. The preprocessing module binarizes each image in the database and calculates its distance sequence as the shape descriptor. For the scale invariance, we normalize distance sequences and then detect unit curves and low level features from sequences for matching. Next, we construct MBRs for the low-dimensional features for indexing. Finally, we complete index structure by setting the lower and higher end points and connecting unit curves to the low-dimensional features.

Detailed descriptions for the index structure are available in [1][2]. In order to support K–NN search and range search more efficiently, we used the GEMINI framework.

5.2 K-Nearest Neighbor Search

The K-NN algorithm returns K similar results. It was applied for sequence data under the GEMINI framework [1][2]. We revise the K-NN search algorithm for our RI-PDTW under the framework. For a query sequence Q and desired number of neighbors K, our revised K-NN algorithm returns an answer set, *Result*, of K sequence data, where for A $\in$ *Result*, B $\notin$ *Result*, RI_PDTW(Q, A) $\le$ RI_PDTW(Q, B). A detailed description for our revised K-NN algorithm is given in Table 5. In this algorithm, we use the minimum priority queue to visit nodes/objects in the index in increasing order of their distances from Q in the indexed

(Fourier) space. We used four different distance measurements according to the index node type. If the node contained a grouped Fourier-transformed feature C from some sequences, its distance was defined by MINDIST (Q,C). When the node contained Fourier-transformed features C from a sequence, the distance was defined by LB_Fourier (Q,C). Moreover, when the node contained a full sequence C, the lower bound distance was defined by LB_RI-PDTW (Q,C) while the actual distance was defined by RI-PDTW (Q,C). A detailed description of distance measurements is shown in [2]. The detailed steps of our K-NN algorithm are as follows:

1) The root of the index is inserted into the queue (Line 1). Next, we pop the first node from the queue and according to the node type, execute an appropriate function (lines 4-18).
2) If the top is a sequence with RI_PDTW distance, we add this sequence into the result. If |*Result*| is equal to K, we return the *Result*.
3) If the top is a leaf node, then for the contained Fourier points, we calculate and record the LB_RI-PDTW distance from the query and insert Fourier points and their distance into the queue (lines 8-10).
4) If the top is a Fourier point, we retrieve a full sequence, and calculate and record the RI-PDTW distance from the query and insert the sequence and distance into the queue (lines 11-12).
5) Otherwise (type is nonleaf node), all the child nodes in the current node are inserted in the queue after calculating MINDIST from the query (lines 16-18).

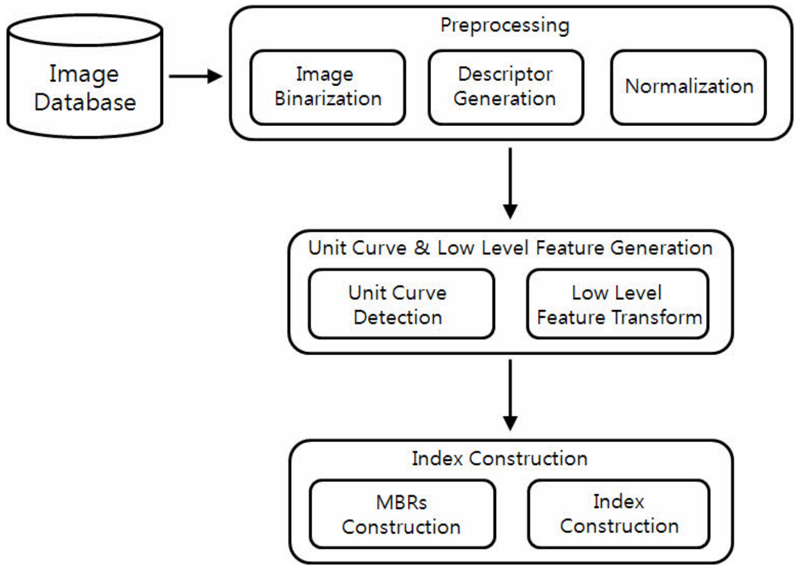


Fig. 8. Overall procedure for index construction

5.3 Range Search

Unlike the K-NN search algorithm, the range search algorithm searches all the images whose distances from the query are equal to or less than given tolerance *e*. The algorithm takes a query sequence Q, tolerance *e*, and the root of an index tree N as input. The range search algorithm works as follows:

1) If the node is a nonleaf node, all the children of the node, whose MINDIST value from the query sequence is equal to or less than the tolerance *e*, call the range search algorithm recursively (lines 1-4).
2) Otherwise (if the node type is leaf node), if the LB_Fourier of the node is equal to or less than the tolerance *e*, retrieve its full sequence. If the both LB_RI-PDTW and RI-PDTW distances from the query are equal to or less than tolerance *e*, add the sequence to the *Result* (lines 5-9).

Table 5. Algorithm K-NN search (Q, k)

```
Variable queue : MinPriorityQueue;

1.  queue.push(root); Result = {}
2.  while not queue.IsEmpty() do
3.      top = queue.Pop();
4.      if top is a sequence with RI_PDTW Dist.
5.          add top to the Result;
6.              if |Result| = k
7.                  return Result;
8.      else if top is a leaf node
9.          for each Fourier point C in top
10.             queue.push(C, LB_Fourier(Q,C)));
11.     else if top is a Fourier point, C
12.         retrieve a full sequence S from database;
13.         queue.push(C, LB_RI-PDTW(Q,C));
14.     else if top is a sequence with LB_RI-PDTW Dist.
15.         queue.push(C, RI-PDTW(Q,C));
16.     else   // top is a non-leaf node
17.         for each child node C in top
18.             queue.push(C, MINDIST(Q,C));
```

Table 6. Algorithm Range search (Q, ε, N)

```
1.  if N is a non-leaf node
2.      for each child node C of N
3.          if MINDIST(Q,C) ≤ ε
4.              call Range Search(Q, ε, U);
5.  else // N is a leaf node
6.      for each Fourier points C of N
7.          if LB_Fourier(Q,D) ≤ ε
8.              retrieve a full sequence S from database;
9.              if LB_RI-PDTW(Q,D) ≤ ε && RI-PDTW(Q,S) ≤ ε
10.                 add S to the Result;
```

6. Experimental Results

To evaluate the performance of our proposed scheme, we built a prototype leaf image retrieval system and carried out various experiments. We collected about 600 leaf images from diverse sources and used an Intel Pentium 4 3.0 GHz CPU with 2 GB RAM. All the algorithms proposed in this paper were implemented in C#.

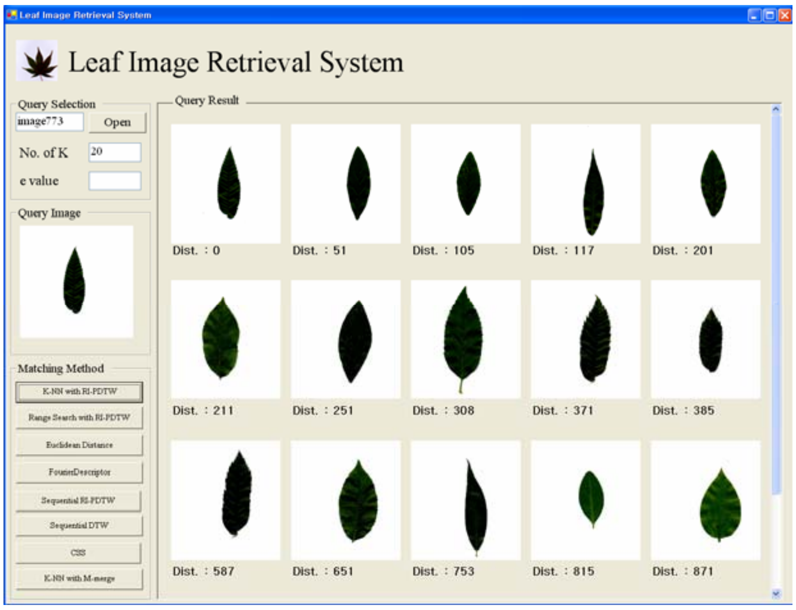


image773

No. of K 20

e value

Fig. 9. User interface of the prototype system.

Fig. 9 shows the overall user interface of the system. The interface consists of three parts: query interface, matching method selector, and result viewer. For sequence matching, we implemented various matching methods including curvature scale space (CSS) [22], RI-PDTW, Fourier descriptor, and Euclidean distance matching. In the first experiment, we compared our RI-PDTW method with DTW without rotation invariance, H-merge [16] and mountain climbing sequence (MCS) [17]. For the experiment, we first generated several rotated images in the range of 0-360 degrees from a query image and inserted them into the database. Next, we tested how many rotated images are included in the result. **Fig. 10** shows the precision-recall graph for our proposed RI-PDTW, DTW, H-merge and MCS. The graph shows that all the rotation invariant schemes are robust to the image rotation.

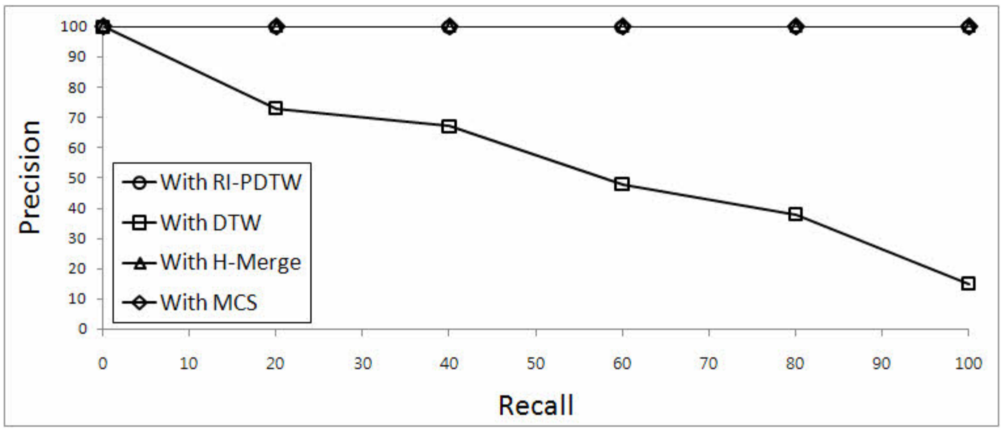


Fig. 10. Precision-recall under image rotation.

Table 7 shows the warping window sizes of DTW and PDTW on the same warping constraint. For this comparison, we used leaf images with two, three, and four blades because they are the most popular shapes. From the table, we can see that PDTW needs a much smaller warping window size than DTW. Since points inside of the warping window size are used to find the optimal path in matching, a smaller warping window size requires less memory and CPU time. Therefore, in that aspect, PDTW is more effective than naïve DTW, especially for multi-blade leaves.

Table 7. Warping window sizes of DTW and PDTW depending on the number of leaf blades.

	Warping window size					
	2 Blades		**3 Blades**		**4 Blades**	
DTW	5868	49.69%	5868	32.92%	5868	28.78%
PDTW	2916		1932		1689	

In the next experiment, we compared RI-PDTW with scale normalization with naïve RI-PDTW without scale normalization. For the test, we first generated several images of different scales from a query image and inserted them into the database. Next, we tested how many scaled images were included in the result. Fig. 11 shows the precision-recall graph for with/without our scale normalization scheme. The graph shows that our scale normalization scheme can guarantee scale invariance while preserving accuracy even with different image scales. Also, we can observe that if we do not use any scale normalization scheme, the accuracy could be dropped seriously.

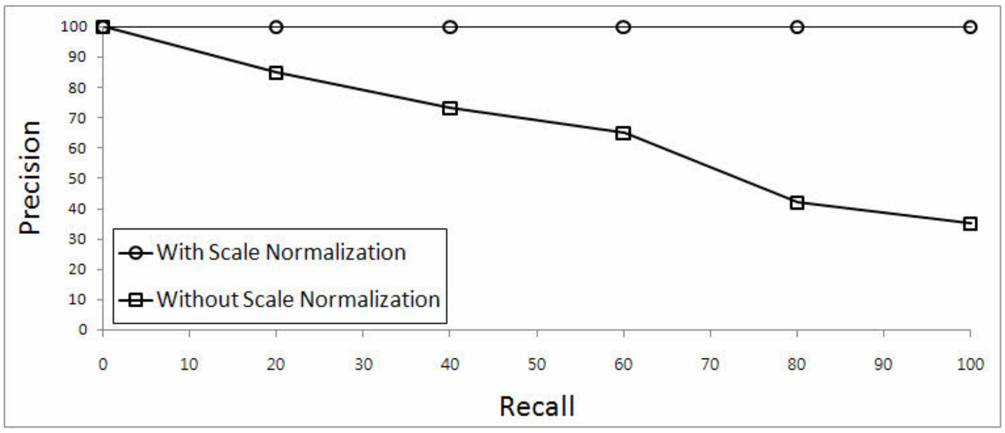


Fig. 11. Precision-recall under image scaling.

In the next experiment, we measured the precision and recall of five different matching methods. In this experiment, we considered both sketched and photographed images because they are the most popular query types. DTW is a very popular sequence matching algorithm and it can match sequences more exactly than Euclidean distance or Fourier descriptor. We concentrated on whether our RI-PDTW could produce more accurate result than DTW and CSS. Fig. 12 and Fig. 13 show precision-recall for the sketched query image and photographed query image, respectively. From the graph, we can see that our RI-PDTW shows the best precision and recall for both types of query images. Moreover, photographed query images gave better accuracy than sketched query images.

In the next experiment, we measured the K-NN search time of six different searching methods. In Fig. 14, the searching time indicates the elapsed time for finding K images in the order of similarity from the database. As shown in the graph, sequential RI-PDTW, sequential

DTW and CSS take huge computation time to find K closest images. In addition, these three schemes and sequential Euclidean distance scheme have the same computation time even though the value of K is changed. On the other hand, our proposed K-NN based DTW and K-NN with priority queue based RI-PDTW can dramatically reduce the searching time compared to sequential schemes. And these schemes require less searching time when we find a small number of closest images like nearest neighbor search.

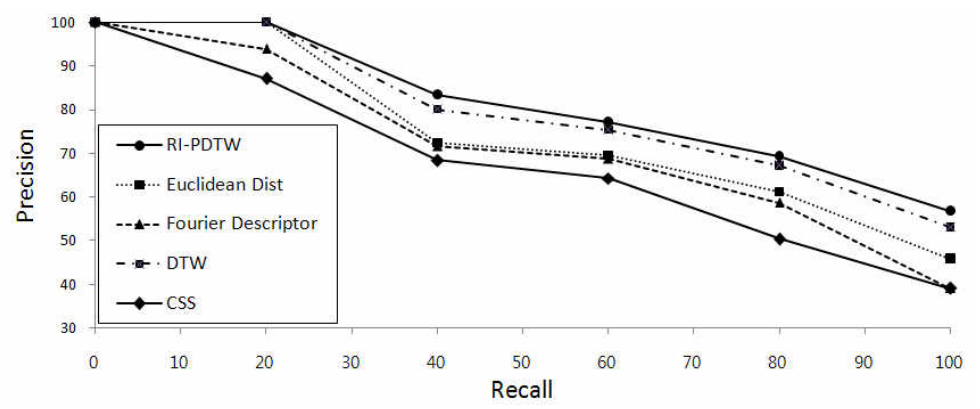


Fig. 12. Precision-recall of matching methods for sketched images.

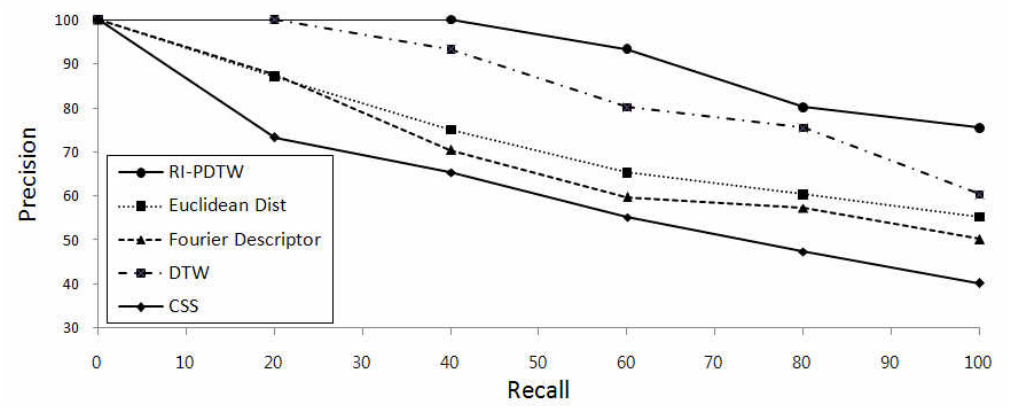


Fig. 13. Precision-recall of matching methods for photographed images.

In the last experiment, we compared six different searching methods in terms of range search. In Fig. 15, the searching time indicates the elapsed time for finding images from the database whose distance from the query is less than given threshold *e*. As shown in the graph, sequential RI-PDTW, sequential DTW and CSS take huge computation time to find similar images. In addition, these three schemes and sequential Euclidean distance scheme have the same computation time even for different threshold *e*. This is because these sequential methods have to calculate distances of the entire images to find images within the given range. On the other hand, our range search with priority queue based on RI-PDTW dramatically reduced the searching time compared to sequential schemes such as K-NN search. And these schemes also require less searching time for the smaller range. Consequently, experimental results show that our proposed RI-PDTW is a very effective matching method for leaf image retrieval because it takes searching time very close to DTW while guaranteeing rotation invariance.

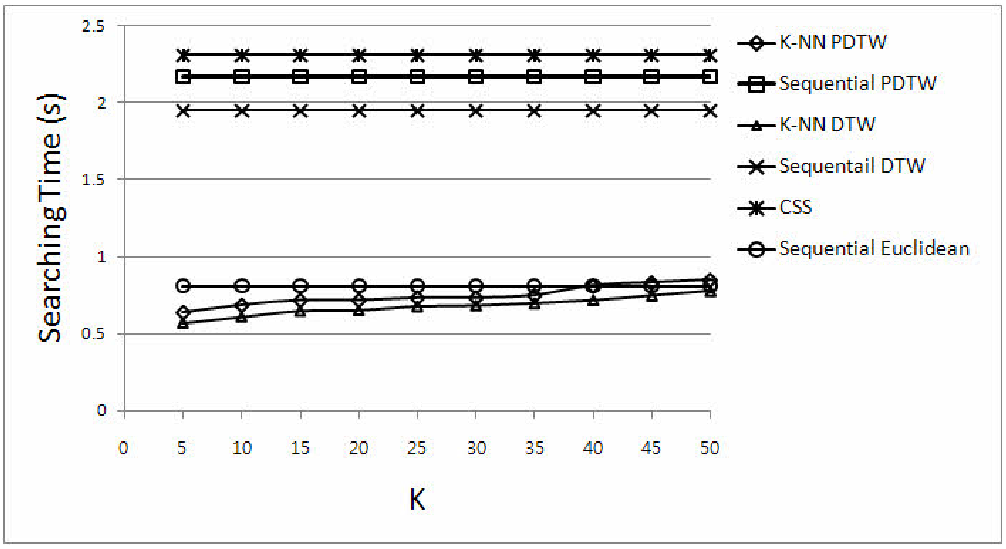


Fig. 14. Searching time of K-NN search.

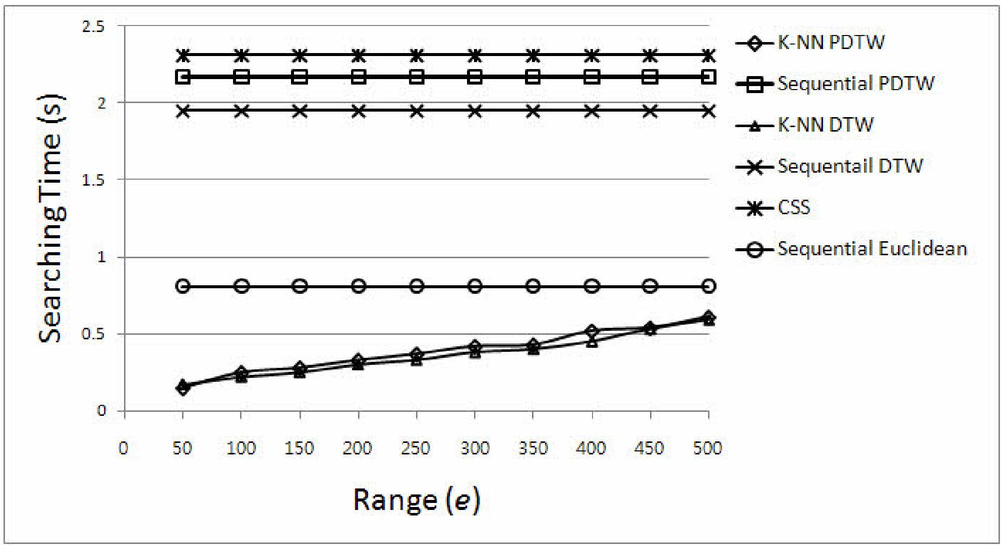


Fig. 15. Searching time of range search

7. Conclusion

In this paper, we proposed a new shape representation, indexing, and matching scheme for effective leaf image retrieval. For the shape representation of leaves, we generated a distance curve by accumulating a sequence of distances between a leaf's center point and all its contour points. For matching, we developed a blade-based, rotation-invariant matching algorithm based on dynamic time warping. Furthermore, in order to speed up the matching process, we proposed two pruning techniques: priority queue-based pruning for unnecessary blade sequence for rotational invariance, and lower bound-based pruning for unnecessary PDTW calculations. To evaluate performance, we implemented a prototype leaf image retrieval system on the GEMINI framework. Throughout the experiment, we showed that our proposed scheme requires less memory and CPU time than DTW, while giving the best precision and recall for K-NN and range queries compared to competitive matching methods.

References

[1] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," *ACM SIGMOD*, pp. 419-429, 1994.

[2] E. Keogh and C. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems*, vol.7, pp. 358-386, 2005.

[3] Y. Rui, T.S. Huang, and S.F. Chang, "Image Retrieval: Current Techniques, Promising Directions and Open Issues," *Journal of Visual Communication and Image Representation*, vol.10, no.4, pp. 39-62, 1999.

[4] M. Kokare, P.K. Biswas, and B.N. Chatterji, "Texture image retrieval using new rotated complex wavelet filters," *IEEE transactions on Systems, Man and Cybernetics – Part B*, vol.35, no.6, pp. 1168-1178, Dec. 2005.

[5] K. Tieu and P. Viola, "Boosting image retrieval," *International Journal of Computer Vision*, vol.56, no.1/2, pp. 17-26, 2004.

[6] Y. Shen, C. Zhoa and K. Lin, "Leaf image retrieval using a shape based method," *IFIP conference on Artificial Intelligence applications and innovation* (*AIAI 2005*), pp. 711-715, Sep. 7-9 2005.

[7] S. Kim, Y. Tak, Y. Nam, and E. Hwang, "mCLOVER: mobile Content-based Leaf Image Retrieval System," *ACM Multimedia*, pp.215-216, 2005.

[8] Y. Nam and E. Hwang, "A Shape-Based Retrieval Scheme for Leaf Image," *LNCS*, vol. 3767, pp. 876-887, 2005.

[9] J. Sklansky, R. L. Chazin, and B. J. Hansen, "Minimum perimeter polygons of digitized silhouetts," *IEEE Transaction s on Computer*, vol.21 no.3, pp.260-268, 1972.

[10] R. Agrawal, C. Faloutsos and A. Swami, "Efficient similarity search in sequence databases," *LNCS*, vol.730, pp. 69-84, 1993.

[11] C.Faloutsos and K.Lin, "FastMap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," *ACM Sigmod*, pp.163-174, 1995.

[12] Z.R. Struzik and A. Siebes, "The Haar Wavelet Transform in the Time Series Similarity Paradigm," *LNCS*, vol.1704, pp. 12-22, 1999.

[13] Y. Zhu and D. Shasha, "Warping indexes with envelope transforms for query by humming," *ACM Sigmod*, pp. 181-192, 2003.

[14] S. Kim, S. Park and W. Chu, "An Index-based approach for similarity search supporting time warping in large sequence databases," *International Conference on Data Engineering*, pp. 607-614, 2001.

[15] B. Yi, H. Jagadish and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," *International Conference on Data Engineering*, pp. 23-27, 1998.

[16] E. Keogh, L. Wei, X. Xi, S. Lee and M. Vlachos, "LB_Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures," *International Conference on Very Large Data Bases*, pp. 882-893, 2006.

[17] H. Lin, Y. Kao, S. Yen, and C. Wang, "A Study of shape-Based Image Retrieval," *International Conference on Distributed Computing Systems Workshops*, pp. 118-123, 2004.

[18] A. Cardone, S.K. Gupta and M. Karnik, "A survey of shape similarity assessment algorithms for product design and manufacturing applications," *ASME Journal of Computing and Information Science in Engineering*, vol.3, no.2, pp. 109-118, 2003.

[19] Y. Nam, E. Hwang and D. Kim, "A similarity-based leaf image retrieval scheme: Joining shape and venation features," *Computer Vision and Image Understanding*, vol.10, no.2, pp. 245-259, 2008.

[20] Y. Tak and E. Hwang, "A Leaf Image Retrieval Scheme Based on Partial Dynamic Time Warping and Two-Level Filtering," *International Conference on Computer and Information Technology*, pp. 633-638, Oct. 2007.

[21] C. Ratanamahatana and E. Keogh, “Three Myths about Dynamic Time Warping Data Mining,” *International Conference on Very Large Data Bases*, pp. 385-394, 2006.

[22] F. Mokhtarian, “Silhouette-based isolated object recognition through curvature scale space,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.17, no.5, pp. 539-544, 1995.

Yoon-Sik Tak received his B.E. degree in Computer Engineering from Dongguk University, Korea, in 2005. Currently he is pursuing the PhD degree in the School of Electrical Engineering in Korea University. His current research interests include image indexing and retrieval framework development, Database, Multimedia Systems and Time Series analysis.

Eenjun Hwang received his B.S. and M.S. degree in Computer Engineering from Seoul National University, Seoul, Korea, in 1988 and 1990, respectively; and his Ph.D. degree in Computer Science from the University of Maryland, College Park, in 1998. From September 1999 to August 2004, he was with the Graduate School of Information and Communication, Ajou University, Suwon, Korea. Currently, he is a member of the faculty in the School of Electrical Engineering, Korea University, Seoul, Korea. His current research interests include database, multimedia systems, information retrieval, ubiquitous computing, and web applications.