

Automatic Leaf Recognition from a Big Hierarchical Image Database

Huisi Wu,^{*} Lei Wang, Feng Zhang, Zhenkun Wen[†]

*College of Computer Science and Software Engineering, Shenzhen University,
Shenzhen, People's Republic of China*

Automatic plant recognition has become a research focus and received more and more attentions recently. However, existing methods usually only focused on leaf recognition from small databases that usually only contain no more than hundreds of species, and none of them reported a stable performance in either recognition accuracy or recognition speed when compared with a big image database. In this paper, we present a novel method for leaf recognition from a big hierarchical image database. Unlike the existing approaches, our method combines the textural gradient histogram with the shape context to form a more distinctive feature for leaf recognition. To achieve efficient leaf image retrieval, we divided the big database into a set of subsets based on mean-shift clustering on the extracted features and build hierarchical k -dimensional trees (KD-trees) to index each cluster in parallel. Finally, the proposed parallel indexing and searching schemes are implemented with MapReduce architectures. Our method is evaluated with extensive experiments on different databases with different sizes. Comparisons to state-of-the-art techniques were also conducted to validate the proposed method. Both visual results and statistical results are shown to demonstrate its effectiveness. © 2015 Wiley Periodicals, Inc.

1. INTRODUCTION

As the largest population and the most widely distributed species, plants are ubiquitous in the world. They play an important role for human survival and development by generating oxygen, food, as well as medicines. Botanists usually rely on leaves, flowers, or vein structures to recognize different kinds of plants. Since it is impossible for a botanist to know all the species and their names, they have to identify plant species according to their experiences or on the basis of books. But owing to the limitations of human memory, manually plant recognition turns out to be error prone and time consuming. Thus, developing an automatic plant recognition system is an important and urgent task using the artificial intelligent areas, especially when the number of people engaged in plant classification are showing a clear downward trend and more and more precious plant species are on the edge of extinction.

^{*}Author to whom all correspondence should be addressed; e-mail: hswu@szu.edu.cn.

[†]e-mail: wenzk@szu.edu.cn.

Compared with the flowers or vein structures of plants, leaves can be easily captured throughout the year. More importantly, they can also provide enough discriminative features for plant recognition. Recently, automatic leaf recognition has become a research focus in the computer vision and intelligent system area. Several representative methods were proposed to recognize plants automatically. Relying on shape features and a hypersphere classifier, Wang et al.¹ first proposed an automatic method for recognizing leaf images. Im et al.² also developed an improved method by hierarchically representing shapes of contours using polygons with critical points of curvature. To implement a general automatic leaf recognition system, Wu et al.³ further employed probabilistic neural network with image processing techniques for automatic leaf recognition. On the other hand, Uluturk and Ugur⁴ proposed a simpler leaf recognition method based on bisection of leaves. Based on a linear discriminative analysis method, Shabanzade et al.⁵ also combined both local descriptors and global features for leaf recognition. Zulkifli et al.⁶ obtained the most effective moment for leaf recognition using the general regression neural network by comparing the effectiveness of Zernike moment invariant, Legendre moment invariant, and Tchebichef moment invariant (TMI) features in extracting leaf features. More recently, Wu et al.⁷ proposed a rotation invariant shape context for automatic leaf recognition. However, most of existing automatic leaf recognition methods^{8–14} usually only focused on a simple data set, which contains no more than hundreds of species or at most thousands of leaf images. None of state-of-the-art methods^{15–20} reported a stable performance in either recognition accuracy or recognition speed when compared with a huge image data set. Since there exist more than 8.7 million species of plants known around the world,²¹ it is important to develop an automatic leaf recognition method for a big data set containing millions of leaf images. Considering that changes in lighting, background, and position of the leaves can also create dramatically different images for the same leaf, so the advantages of using bigger data set would almost definitely yield better accuracy performances in automatic leaf recognition. But, on the other hand, the challenges are also obvious. First, special treatments such as precautions or indexing must be taken to tackle the strict hardware constraints when dealing with millions of leaf images simultaneously. Second, more sophisticated feature extraction schemes should be designed to accurately represent clustered or indexed leaf image in the big data set.

In this paper, we introduce a novel automatic leaf recognition method for big hierarchical image database. To the best of our knowledge, our method is the first to tackle automatic leaf recognition from a big image database, which contains millions of leaf images. To solve this new problem, our technical contributions primary included two aspects: a novel leaf feature representation and a novel efficient parallel matching scheme. Different from the existing approaches, we develop a more distinctive feature for leaf recognition by taking both the textural histogram and the shape context into account. To tackle efficient feature matching from millions of images, we divide the big database into a number of subsets based on mean-shift clustering on the extracted features. And each cluster leaf feature is indexed with a hierarchical k -dimensional trees (KD-trees) in parallel to achieve efficient leaf image retrieval. Finally, MapReduce architectures are employed to implement the proposed parallel indexing and searching schemes. The proposed method was

evaluated with extensive experiments on different databases with different sizes. We also compared our method with the state-of-the-art methods in terms of performance statistics. Experimental results demonstrated that our method generally outperformed all competitors in both accuracy and running time when compared with a big database.

2. METHODOLOGY

2.1. Data Collections

It is reported that there exist more than 45 thousand species of plants known in China.²¹ The South China has more species as there is plenty of rain all the year round. In this paper, we focus on the automatic leaf recognition for the species found in the South China. Our database is collected from The Southern China Botanical Garden (Guangzhou, China). It contains 23,025 species of plant in the South China. Since leaves of the same species have similar patterns, and more images for a species would almost always represent better its discriminative pattern, about 80 images for each species were collected under different position, rotation, orientation, view angle, and lighting conditions. Therefore, we have about 1.5 million leaf images to form a big database for automatic leaf recognition. Typical leaf image examples in our big database are as shown in Figure 1. Note that, such a big database is used the first time for testing of the leaf recognition system to the best of our knowledge.

2.2. Preprocessing

To extract discriminative features from a leaf image, an image preprocessing step should be conducted before the actual analysis of the specific pattern in the leaf image. To obtain better quality images for leaf segmentation, we first use Gaussian convolution to eliminate the noise and enhance the degraded images. Since we need the distinctive shape, color distribution or texture information as the cues for leaf



Figure 1. Typical leaf image examples in our big database (1.5 million images from 23,025 species in the South China).

recognition, leaf segmentation becomes extremely crucial to obtain accurate shape curves. In this paper, we take a graph-based approach²² for leaf segmentation, which is efficient and well-valid method for image segmentation. In our implementation, we used an undirected graph G to represent the regions to be segmented, written as

$$G = (V, E) \tag{1}$$

where vertices $v_i \in V$ are the basic elements to form the undirected graph G , and edges $(v_i, v_j) \in E$ are corresponding to pairs of neighboring vertices. We also used a nonnegative weight value $W(v_i, v_j)$ to measure the dissimilarity between two neighboring elements v_i and v_j . For the case of our leaf segmentation, the vertices are pixels in the original image and we simply employed the color and gradient distances as the weight of an edge to define the dissimilarity between two pixels. In our graph-based segmentation, a segmented region S is defined as a partition of V , where the objective function of the optimization is to minimize the weights between two vertices in the same segmented region and maximize the weights between two vertices in different regions. In our implementation, we finally define the segmented regions based on pairwise region comparison.²² This approach employed a simple greedy decision to produce segmentations neither too coarse nor too fine based on the particular region comparison function. Typical segmented results are as shown in Figure 2. Note that, our segmentation runs in $O(n \log n)$ time for n graph edges, which is also fast in practice and generally only cost a fraction of millisecond to segment a leaf image.

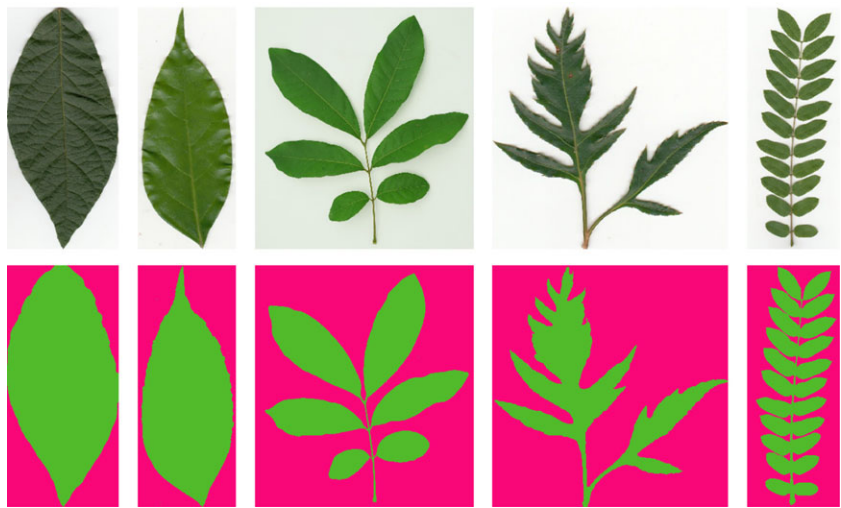


Figure 2. Leaf segmentation. First row: original images. Second row: segmented results using a graph-based approach.

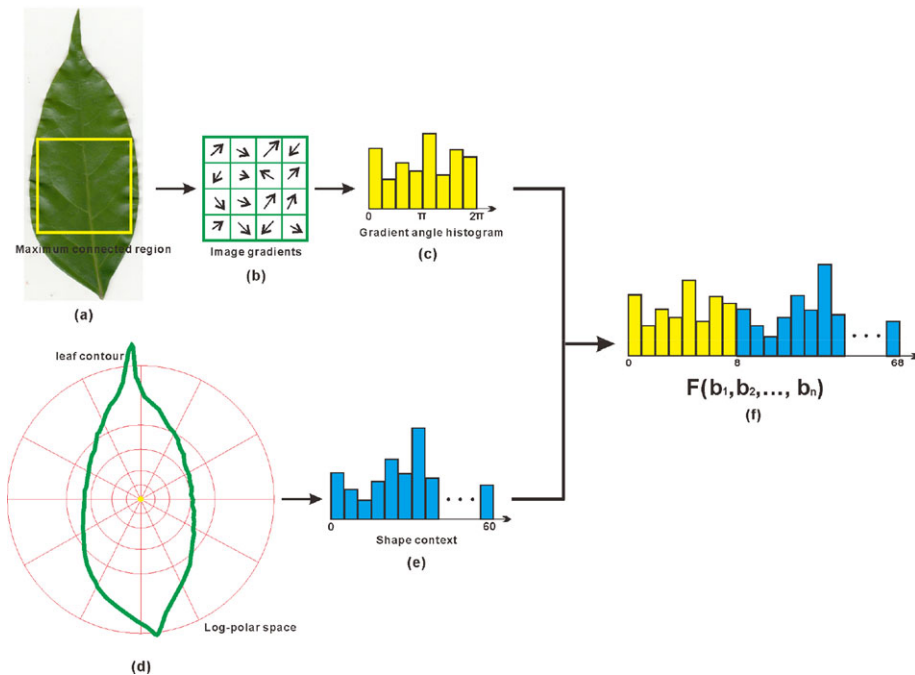


Figure 3. Feature extraction. (a) Maximum corrected region cropped from the segmented leaf. (b) Gradients calculated from the 4×4 subareas. (c) Gradient angle histogram. (d) log-polar space assigned to the center of the extracted leaf contour. (e) Calculated shape context. (f) Final leaf feature extracted by concatenating the shape context to the gradient angle histogram.

2.3. Feature Extraction

As leaves of different plants may be similar in color or shape, only color feature²³ or shape feature²⁴ alone cannot achieve high accuracy in the automatic leaf recognition. Thus, we take into account both color and shape features to identify leaf, which is expected to outperform the existing leaf recognition methods^{25,26} that only took into account the distinctive shape feature as the sole recognition cue.

To extract distinctive features of segmented leaf image in both color domain and shape domain, we used gradient histogram to represent the textural feature of leaves and described the leaf shape using shape context.²⁷ As shown in Figure 3a, we first obtained a maximum corrected region from the segmented leaf image (Figure 2). This maximum correct region contains most of distinctive textural features of a leaf, so we can extract gradient histogram from it. Compared with color histogram, we found that gradient histogram is much more unique for a specific class of leaves, because color distribution may highly variable across different seasons of the same species but gradient distribution is invariant. Thus, we divided the maximum corrected area into 4×4 subareas and computed all of their gradients, as shown in Figure 3b. We designed an angle histogram with eight bins according to the angle

of the gradients. For each subregion, we added the magnitude of its gradient to one of eight bins. These bins form a unique textural feature that can later be used to identify identical leaves. On the other hand, we used a shape context to describe the unique shape of a leaf. As shown in Figure 3d, we obtained a center point by averaging the extracted leaf contour and described the shape context of a leaf under a log-polar space. We also used one-dimensional vector to denote the calculated result by concatenating the original shape context according to the log dimension. As shown in Figure 3e, we obtained a shape context with 60 bins as we set the original log-polar space as 5×12 . Finally, we concatenated the shape context to the gradient angle histogram and finally formed a distinctive feature $F(b_1, b_2, \dots, b_{68})$ for the later leaf recognition, as shown in Figure 3f.

2.4. Indexing

Given the extracted leaf features, we need to design an efficient algorithm to retrieve a leaf feature from millions of images. Currently, Bag of Words (BoW)^{28–31} and Full Representation (FR)^{32–37} are two primary approaches to build large scale image retrieval database. BoW represents each image with a histogram of occurrences of extracted features, which has the advantage of saving an order of magnitude storage. FR retrieves approximate nearest neighbors by searching all the features in the databases, which required more storage in indexing but its accuracy performance is much better than BoW. In our experiments, we chose a FR approach to build a large-scale hierarchical database based on KD-tree indexing. For the KD-tree implementation, we followed the real-time KD-tree construction method built on graphic processing unit (GPU).⁴⁷ Specifically, we built a KD-tree in a greedy, top-down manner via splitting each parent node into two subnodes. First, we evaluate the surface area heuristic costs for all splitting candidates. The optimal candidate with the lowest cost is then picked and split into two children nodes. A KD-tree is finally built by recursively splitting a parent node into two subnodes. We implemented the above KD-tree construction algorithm with Compute Unified Device Architecture (CUDA), which is a parallel programming language provided by NVIDIA.

Most of existing automatic leaf recognition system did not consider a scalable system to handle millions of images due to the storage limits on one machine. Since our goal is to recognize leaf images from a big database, we have to distribute the images on an arbitrary number of machines to achieve the recognition from millions of leaf images. Therefore, we implemented a KD-tree parallelization approach using MapReduce paradigm.^{38,39} MapReduce is a programming framework provided by Google to support distributed processing of large data sets on a set of independent computers or computing nodes. MapReduce paradigm supports two distributed programming functions: map and reduce. Map paradigm usually takes an input pair to produce a number of pairs of keys and values, whereas reduce paradigm takes an intermediate key and a set of values associated with it to merge together for outputs.

Our implementation of parallel indexing leaf images by building KD-trees is as shown in Algorithm 1. Given a large-scale image database, we first extracted a feature $F_i(b_1, b_2, \dots, b_{68})$ for each leaf image I_i in the database. Since similar leaves

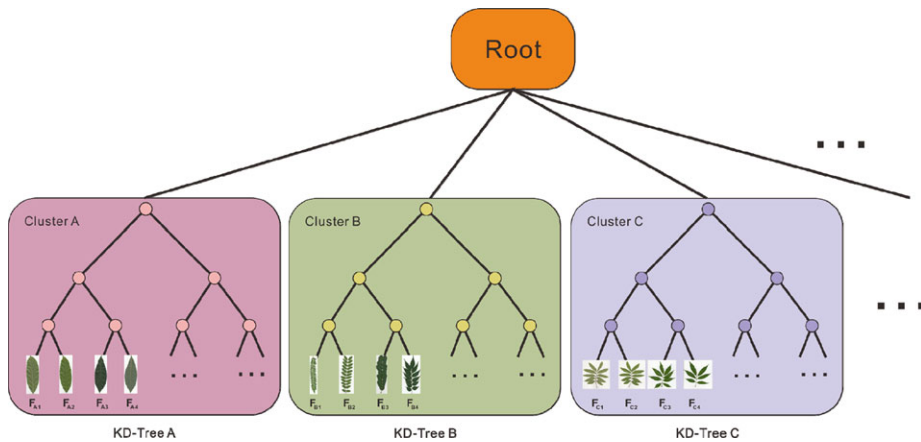


Figure 4. Hierarchical indexing leaf images using KD-trees.

should have similar image feature, this suggests that we can perform clustering on the leaf image features before building KD-trees. We used adaptive mean-shift clustering⁴⁰ to group the extracted features $\{F_1, F_2, \dots, F_n\}$, and obtained m clusters $\{C_1, C_2, \dots, C_m\}$ of leaf images. Similar with Zhou et al.,⁴¹ our mean-shift clustering algorithm was also implemented on GPU by using CUDA. The major advantage of parallel mean-shift clustering is it can automatically determine the number of clusters, which otherwise requires knowing the number of clusters in advance by using other techniques, such as k -means. The m clusters $\{C_1, C_2, \dots, C_m\}$ of leaf images were assigned to one of distributed machine using MapReduce, which built a KD-tree independently in parallel based on a best-bin-first (BBF) algorithm. The BBF algorithm is a searching algorithm commonly used to find an approximate solution for the nearest neighbor search problem in high-dimensional spaces. BBF is an approximate algorithm, which backtracks according to a priority queue based on closeness. By searching a fixed number of nearest candidates and stop, it returns the nearest neighbor for a large fraction of queries in an extremely efficient way.

Finally, a hierarchical global tree was formed by simply assigning all KD-trees $\{T_1, T_2, \dots, T_m\}$ to a root node. As shown in Figure 4, we can see that similar leaves are indexed on the same KD-tree to facilitate the following leaf query. The detailed MapReduce scheme and implementation is as shown in Figure 5a. The implementation of distributed KD-trees with MapReduce was quite straightforward. We chose the simplest way of parallelization by dividing the big image database into subsets based on clustering, where each cluster can be fitted in the memory of one machine. Then each machine built an independent KD-tree in parallel for a cluster of images. The Feature MapReduce was run after the initialization of the training stage. The Indexing MapReduce then built the independent KD-trees on each cluster of leaf images.

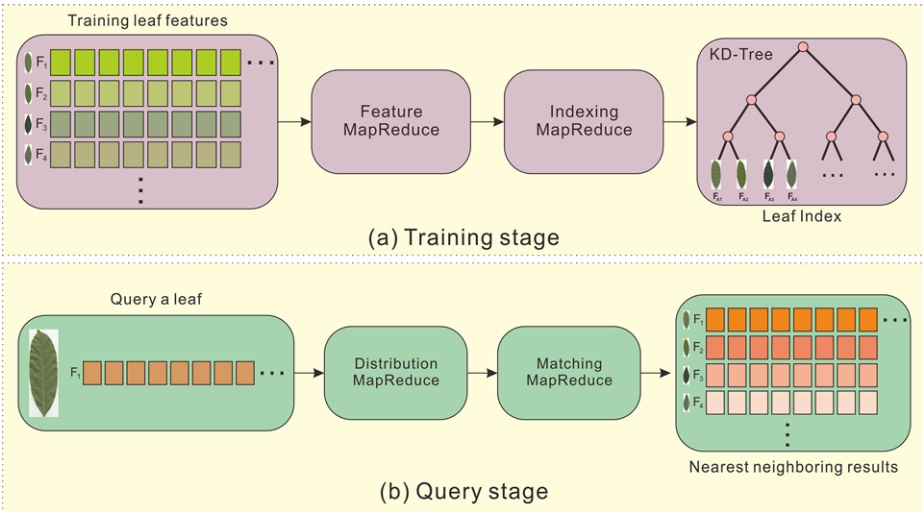


Figure 5. Parallel training and query stage using the MapReduce architecture. (a) Training stage. MapReduce distributes all leaf features to different machines, which build different KD-trees in parallel. (b) Query stage. A leaf to query is input. MapReduce then distributes it to all independent KD-trees for parallel query. All nearest neighboring results are picked up by KNN matching using MapReduce.

Algorithm 1 Parallel indexing leaf images by building D-trees

Input: Leaf images from a big database.
Output: A set of KD-trees (m) hierarchical indexing the leaf image database.
Operation:

1. Extract feature $Fi(b_1, b_2, b_{68})$ for each leaf image I_i in the database.
2. Cluster all leaf features $\{F_1, F_2, \dots, F_n\}$ using Mean-shift.
3. For each cluster C_i in $\{C_1, C_2, C_m\}$: *// in parallel*
Build a KD-tree T_i for all leaf features in cluster C_i using Best-Bin-First algorithm.
4. Form a global tree by assigning all KD-trees $\{T_j, T_2, \dots, T_m\}$ to a root node N.

2.5. Query

Based on the indexed hierarchical leaf images, we designed a parallel querying scheme as shown in Algorithm 2. Given an input image I_q , we first extracted its feature $F_q(b_1, b_2, \dots, b_{68})$, which was assigned to all built KD-trees for parallel searching. For each KD-tree, we can search k most similar leaf images among it using a k -nearest neighbors searching algorithm.^{42–45} Recall that we built m KD-trees in total in the indexing stage, so we retrieved $k \times m$ most similar leaves. By sorting the $k \times m$ nearest neighbors according to their distances to feature F_q , we finally obtained the k most similar leaf images.

The implementation of parallel querying leaf images with MapReduce was also straightforward, as shown in Figure 5b. A single root machine received the

Table I. Five data sets with different number of leaf images for evaluations.

Data set	Number of species	Number of leaf images
1	65	5 K
2	129	10 K
3	2 K	100 K
4	10 K	500 K
5	20 K	1.0 M

query image I_q and passed the query features F_q to all the machines, which then query k -nearest neighbors in parallel. Specifically, given the input leaf feature, the Distribution MapReduce distributed the feature F_q to all machines. The Matching MapReduce searched k -nearest neighbors in parallel on each machine in the map operations and performed the finally leaves counting and feature distance sorting in the reduce operations. Finally, the root machine collected the results and output the sorted list of leaf images.

Algorithm 2 Parallel query a leaf image

Input: A leaf image I_q to query.

Output: k most similar leaf images among the big hierarchical leaf image database.

Operation:

1. Extract a feature $Fi(b_1, b_2, b_{68})$ for the leaf image I_q .
 2. Assign the leaf feature F_q to all built KD-trees.
 3. For each KD-tree : T_i // **in parallel**
 Search k most similar leaf images among the KD-tree T_i using a KNN algorithm.
 4. Sort the $k \times m$ nearest neighboring results according to their distance to feature F_q .
 5. Return the k most similar results $\{I_1, I_2, \dots, I_k\}$.
-

3. RESULTS

3.1. Visual Evaluations

We have implemented our proposed method using MapReduce infrastructure based on the open-source Hadoop software.⁴⁶ In our experiments, the number of distributed machines was between 8 and 20, where the memory per machine was limited to 16 GB. To evaluate our method in handling different scales of image databases, we collected five leaf data sets, which include 5 K to 1 M leaf images captured from 65 to 20 K of species, as shown in Table I.

For visually evaluations, we run our proposed method to the collected data sets and visualized the leaf recognition results, as shown in Figures 6 and 7. By setting the $k = 5$ in the K-Nearest Neighbors (KNN) searching, our system automatically retrieved five best recognition results according to the given input leaf image.

Figure 6 illustrates the performance of our system in identifying leaves with approximate convex shapes. We can see that our system can accurately recognize the leaves from the same species of the input leaf. For the species with nonconvex shapes or including inseparable multiple small leaves, our method can also easily

























Input Leaf	Recognition Results				
	1 st	2 nd	3 rd	4 th	5 th
					
					
					
					

Figure 6. Visual evaluation for leaf recognition. The left column: input leaf. The five columns in the right: five best recognition results.

identify them without any modification. From the results shown in Figure 7, we can see that our method performed accurately in the complex leaves retrieval, even for the leaves with complicated shapes.

3.2. Statistical Evaluations

Besides the visual evaluations, we also performed a statistical evaluation for our method. We also implemented three state-of-the-art methods to compare with our method in leaf recognition, including Wu et al.,³ Chaki and Parekh,²⁵ Kumar et al.,²⁶ and our methods. We first compared the ability in handling different scales of databases for different methods. By setting $k = 5$ in the KNN searching, we run different leaf recognition methods onto the five collected data sets. An average leaf recognition rate and running time were collected from 30 times of independent leaf recognitions with different methods.

In our experiments, the average running time only calculated the query time and did not include preprocessing time for each method. The reasons for why we ignored the time of preprocessing for different leaf recognition systems are (a) evaluating time performance of different systems by collecting the time from input a leaf to the recognized results returned is reasonable. More importantly, it is coincidence with the user experience; (b) different methods may have different image processing

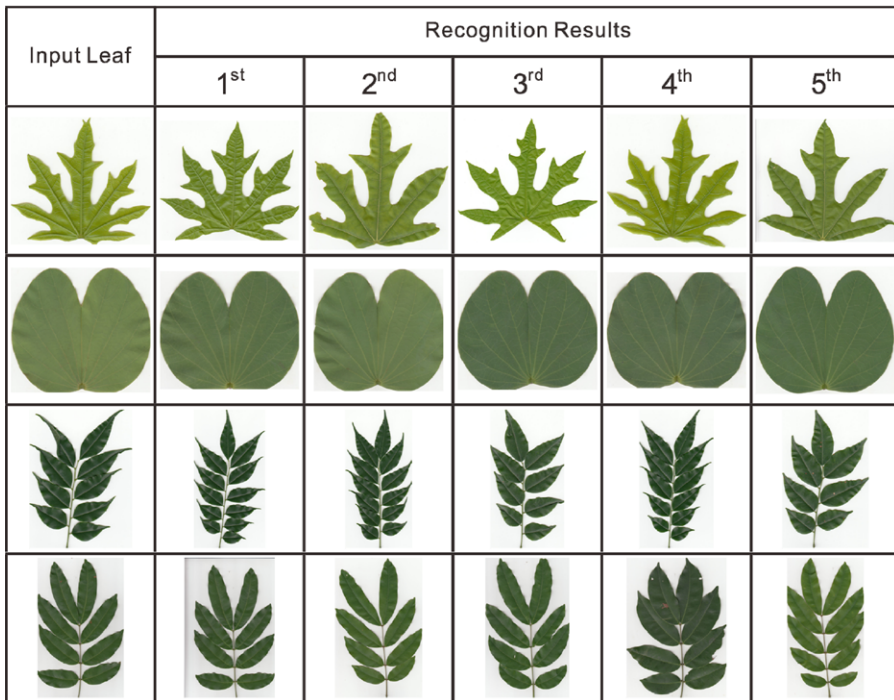


Figure 7. Visual evaluation for leaf recognition. The left column: input leaf. The five columns in the right: five best recognition results.

and feature extraction before the leaf query, which were usually done before leaf recognition in an offline manner. Timing those parts becomes less important from the view of the users; (c) we discarded timing preprocessing steps equally for all competitors in our comparison, so we believe it is still a fair comparison for only leaf recognition. The results are plotted in Figure 8, where we can see that our method generally outperformed all of the three competitors in both accuracy and running time.

The means of the leaf recognition rate and running time are as shown in Tables II and III. From the statistical results, we can clearly see that none of the existing methods can maintain a stable performance when the size of image database becomes large. Because Wu et al.,³ Chaki and Parekh,²⁵ and Kumar et al.²⁶ did not have indexing and parallelism design, their recognition accuracy suddenly dropped when the image numbers were larger than 10 K. Moreover, their running time generally cost more than 1 h when the image numbers larger than 20 K.

Instead, our method can preserve a stable performance based on our novel parallel indexing and searching framework. By considering both textural and shape feature of the leaves, our method achieved a high recognition rate (>90%) for all collected data sets, as shown in Table II. Moreover, based on the MapReduce settings and our KD-tree indexing, our method also generally outperformed the three other

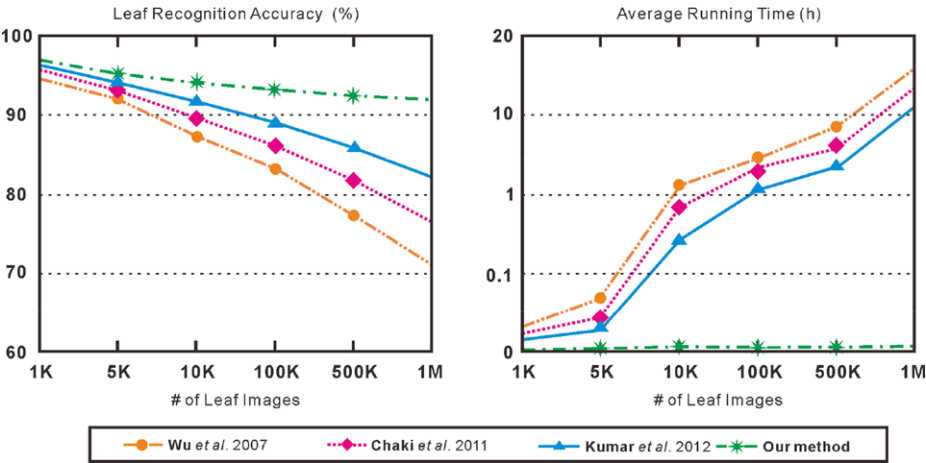


Figure 8. Effect of leaf image numbers. $k = 5$ in the KNN searching. Average leaf recognition rate and running time were collected from 30 times of independent leaf recognitions with different methods. Note that: the average running time only calculated the query time (did not include preprocessing time for each method).

Data set	Leaf recognition accuracy (%)			
	Wu et al. ³	Chaki and Parekh ²⁵	Kumar et al. ²⁶	This study
1	91.51	92.13	93.35	95.26
2	88.38	89.22	91.87	94.69
3	83.40	86.61	89.49	93.18
4	78.83	82.15	86.56	92.21
5	72.39	78.44	83.52	91.75

Data set	Average running time			
	Wu et al. ³	Chaki and Parekh ²⁵	Kumar et al. ²⁶	This study
1	3.31 min	2.19 min	1.06 min	0.81 s
2	1.13 h	0.93 h	0.78 h	0.96 s
3	5.41 h	3.76 h	1.49 h	1.23 s
4	8.85 h	6.43 h	3.86 h	1.52 s
5	14.3 h	12.1 h	10.5 h	1.81 s

competitors in terms of running time, especially for the big image databases. As shown in Table III, our method can retrieve an accurate result within 2 s even when recognizing leaves from the data set 5 (1 million leaf images), whereas the other methods usually cost more than 10 h to retrieve a leaf recognition result as they did not have a hierarchical indexing and parallel searching schemes.

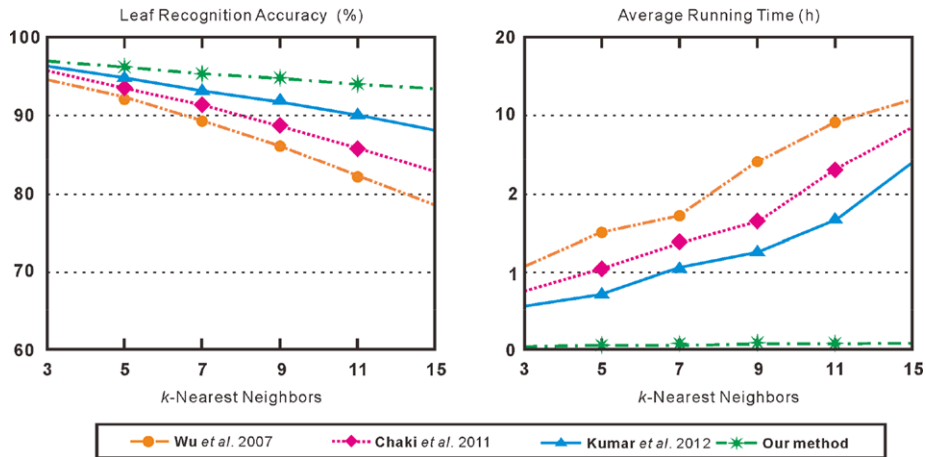


Figure 9. Effect of k in the KNN searching. Data set 3 (including 10 K images) was used in this evaluation. Average leaf recognition rate and running time were collected from 30 times of independent leaf recognitions with different methods. Note that: the average running time only calculated the query time (did not include preprocessing time for each method).

In addition, we also tested the effect of k in KNN searching for leaf recognition. We used the data set 3 (including 10 K images) to evaluate the effect of number k . By setting a set of different values for k (from 3 to 15), we run our method and the three other competitors on the data set 3. Similarly, we also collected both average leaf recognition rate and running time were collected from 30 times of independent leaf recognitions with different methods. The results are plotted in Figure 9, where we can also observe that our method generally outperformed all of the three competitors in both accuracy and running time, especially when the databases become bigger, which indicates that building parallel KD-tree indexing and searching schemes is important for leaf recognition from big image databases.

4. CONCLUSION

In this paper, we presented a novel method for leaf recognition from a big hierarchical image database. To the best of our knowledge, our system is the first to handle automatic leaf recognition from a big image database containing millions of leaf images. Different from the existing approaches, we also developed a more distinctive feature for leaf recognition by taking both the textural histogram and the shape context into account. To tackle efficient feature matching from millions of images, the big database was divided into a number of subsets based on mean-shift clustering on the extracted features. To achieve efficient leaf image retrieval, each cluster leaf feature is indexed with a hierarchical KD-trees in parallel. Finally, we adopted MapReduce architectures to implement a novel parallel indexing and searching schemes. We evaluated the proposed method with extensive experiments on different databases with different sizes and also compared our method with

the state-of-the-art methods. Experimental results demonstrated that our method generally outperformed all competitors in both accuracy and running time when compared with a big database.

On the other hand, our method also has limitations: (a) by concatenating textural histogram and shape context, our leaf feature became 68 dimensional, which makes our feature saving, transferring, clustering, and indexing to be a computation-intensive task, especially for a big database with millions of leaves; (b) although our method generally outperformed all competitors, our leaf recognition accuracy is still only about 91% when in fact there are millions of leaves.

There are several directions for our future investigation. We plan to design a more concise and lower dimensional representation to capture both textural and shape distinctive leaf features. We also intend to implement a more accurate system for automatic leaf recognition from a big image database. Finally, we are also interested in developing a more compact framework to recognize leaves from big image databases with mobile computing devices.

Acknowledgments

The authors would like to thank our anonymous reviewers for their valuable comments. This work was supported in part by grants from National Natural Science Foundation of China (numbers 61303101, 61170326, and 61170077), the Natural Science Foundation of Guangdong Province, China (numbers S2012040008028 and S2013010012555), the Shenzhen Research Foundation for Basic Research, China (numbers JCYJ20120613170718514 and JCYJ20130326112201234), the Shenzhen Peacock Plan (number KQCX20130621101205783), and the Start-up Research Foundation of Shenzhen University (numbers 2012-801 and 2013-000009).

References

1. Wang XF, Du JX, Zhang GJ. Recognition of leaf images based on shape features using a hypersphere classifier. *Lect Notes Comput Sci* 2005;3644:87–96.
2. Im C, NiShida H, Tosiya L. Recognizing plant species by leaf shapes—a case study of the acer family. In: 14th Int Conf on Pattern Recognition, Brisbane, Australia; August 16–20, 1998. pp 1171–1173.
3. Wu S, Bao F, Xu E. A leaf recognition algorithm for plant classification using probabilistic neural network. In: IEEE Int Symp on Signal Processing and Information Technology, Cairo, Egypt; December 15–18, 2007. pp 11–16.
4. Uluturk C, Ugur A. Recognition of leaves based on morphological features derived from two half-regions. In: Int Symposium on Innovations in Intelligent Systems and Applications (INISTA), Trabzon, Turkey; July 2–4, 2012. pp 1–4.
5. Shabanzade M, Zahedi M, Aghvami S. Combination of local descriptors and global features for leaf recognition. *Signal Image Process* 2011;2(3):23–31.
6. Zulkifli Z, Saad P, Mohtar I. Plant leaf identification using moment invariants & general regression neural network. In: 11th Int Conf on Hybrid Intelligent Systems, Malacca, Malaysia; December 5–8, 2011. pp 430–435.
7. Wu H, Pu P, He G, Zhang B, Zhao F. Fast and robust leaf recognition based on rotation invariant shape context. In: 8th Int Conf on Intelligent Systems and Knowledge Engineering (ISKE2013), Shenzhen, China; November 21–23, 2014. pp 145–154.
8. Fu H, Chi Z, Feng D, Song J. Machine learning techniques for ontology-based leaf classification. In: IEEE 8th Int Conf on Control, Automation, Robotics and Vision, Kunming, China; December 6–9, 2004. pp 681–686.

9. Warren D. Automated leaf shape description for variety testing in chrysanthemums. In: Proc IEE 6th Int Conf on Image Processing and Its Applications, Dublin, OH; July 14–17, 1997. pp 497–501.
10. Brendel T, Schwanke J, Jensch P, Megnet R. Knowledge-based object recognition for different morphological classes of plants. Proc SPIE 1995;2345:277–284.
11. Saitoh T, Kaneko T. Automatic recognition of wild flowers. In: Proc Pattern Recognition, Barcelona, Spain; September 3–7, 2000. pp 507–510.
12. Ridler T, Calvard S. Picture thresholding using an iterative selection method. IEEE Trans Syst Man Cybernet 1978;8(8):30–632.
13. Hu M. Visual pattern recognition by moment invariants. IRE Trans Inform Theory 1962;8(2):179–187.
14. Du JX, Huang DS, Wang XF, Gu X. Computer-aided plant species identification based on leaf shape matching technique. Trans Inst Meas Cont 2007;28(3):275–284.
15. Mishra P, Maurya S, Singh R, Misra A. A semi-automatic plant identification based on digital leaf and flower images. In: Int Conf on Advances In Engineering Science And Management, Tamil Nadu, India; March 30–31, 2012. pp 68–73.
16. ArunPriya C, Balasaravanan T, Thanamani A. An efficient leaf recognition algorithm for plant classification using support vector machine. In: Proc Int Conf on Pattern Recognition, Informatics and Medical Engineering, Tsukuba, Japan; November 11–15, 2012. pp 428–432.
17. Hossain J, Amin M. Leaf shape identification based plant biometrics. In: Proc 13th Int Conf on Computer and Information Technology, Bradford, UK; June 29 – July 1, 2010. pp 458–463.
18. Satti W, Satya A, Sharma S. An automatic leaf recognition system for plant identification using machine vision technology. Int J Eng Sci Technol 2013;4(5):874–879.
19. Ling H, Jacobs D. Shape classification using the inner-distance. IEEE Trans Pattern Anal Mach Intell 2007;29(2):286–299.
20. Wu Q, Zhou C, Wang C. Feature extraction and automatic recognition of plant leaf using artificial neural network. Avances en Ciencias de la Computación. Cd. de México, México; 2006. pp 5–12.
21. Mora C, Tittensor DP, Adl S, Simpson AGB, Boris Worm. How many species are there on earth and in the ocean? PLoS Biol 2011;9(8):e1001127.
22. Felzenszwalb PF, Huttenlocher DP. Efficient graph-based image segmentation. Int J Comput Vis 2004;59(2):167–181.
23. Kekre H, Mishra D, Narula S, Shah V. Color feature extraction for CBIR. Int J Eng Sci Technol 2011;3(12):8357–8365.
24. Manay S, Cremers D, Hong B, Yezzi A, Soatto S. Integral invariants for shape matching. IEEE Trans Pattern Anal Mach Intell 2006;28(10):1602–1618.
25. Chaki J, Parekh R. Plant leaf recognition using shape based features and neural network classifiers. Int J Adv Comput Sci Appl 2011;2(10):41–47.
26. Kumar N, Belhumeur PN, Biswas A, Jacobs DW, Kress WJ, Lopez IC, Soares JVB. Leafs-nap: a computer vision system for automatic plant species identification. In: 14th Eur Conf on Computer Vision, Firenze, Italy; October 7–13, 2012. pp 502–516.
27. Belongie S, Malik J, Puzicha J. Shape matching and object recognition using shape contexts. IEEE Trans Pattern Anal Mach Intell 2002;24(4):509–522.
28. Aly M, Munich ME, Perona P. Indexing in large scale image collections: scaling properties and benchmark. In: IEEE Workshop on Applications of Computer Vision, Kona, HI; January 5–7, 2011. pp 418–425.
29. Jegou H, Douze M, Schmid C. Hamming embedding and weak geometric consistency for large scale image search. In: 10th Eur Conf on Computer Vision, Marseille, France; October 12–18, 2008. pp 304–317.
30. Philbin J, Chum O, Isard M, Sivic J, Zisserman A. Object retrieval with large vocabularies and fast spatial matching. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Minneapolis, MI; June 18–23, 2007. pp 1–8.

31. Philbin J, Chum O, Isard M, Sivic J, Zisserman A. Lost in quantization: improving particular object retrieval in large scale image databases. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Anchorage, AK; June 24–26, 2008. pp 1–8.
32. Lowe D. Distinctive image features from scale-invariant keypoints. *Int J Comput Vis* 2004;60(2):91–110.
33. Aly M, Munich ME, Perona P. Bag of Words for large scale object recognition - properties and benchmark. In: Proc 6th Int Conf on Computer Vision Theory and Applications, Vilamoura, Algarve, Portugal; March 5–7, 2011. pp 299–306.
34. Manay S, Cremers D, Hong B, Yezzi A, Soatto S. Integral invariants for shape matching. *IEEE Trans Pattern Anal Mach Intell* 2006;28(10):1602–1618.
35. Jermyn I, Ishikawa H. Globally optimal regions and boundaries as minimum ratio weight cycles. *IEEE Trans Pattern Anal Mach Intell* 2001;23(10):1075–1088.
36. Zitnick C. Binary coherent edge descriptors. In: Eur Conf on Computer Vision, Crete, Greece; September 5–11, 2010. pp 1–14.
37. Branson S, Wah C, Babenko B, Schroff F, Welinder P, Perona P, Belongie S. Visual recognition with humans in the loop. In: Eur Conf on Computer Vision, Crete, Greece; September 5–11, 2010. pp 438–451.
38. Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters. In: 6th Symp on Operating System Design and Implementation, San Francisco, CA; December 6–8, 2004. pp 137–150.
39. Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters. *Commun ACM* 2008;51(1):107–113.
40. Comaniciu D, Meer P. Mean shift: a robust approach toward feature space analysis. *IEEE Trans Pattern Anal Mach Intell* 2002;24(5):603–619.
41. Zhou F, Zhao Y, Ma KL. Parallel mean shift for interactive volume segmentation. In: Machine Learning in Medical Imaging. *Lect Notes Comput Sci* 2010;6357:67–75.
42. Hall P, Park BU, Samworth RJ. Choice of neighbor order in nearest-neighbor classification. *Ann Stat* 2008;36(5):2135–2152.
43. Muja M, Lowe D. Fast approximate nearest neighbors with automatic algorithm configuration. In: Int Conf on Computer Vision Theory and Application, Lisboa, Portugal; February 5–8, 2009. pp 331–340.
44. Chum O, Philbin J, Isard M, Zisserman A. Scalable near identical image and shot detection. In: CIVR, ACM International Conference on Image and Video Retrieval, Amsterdam, Netherlands; July 9–11, 2007. pp 549–556.
45. Perronnin F, Liu Y, Sánchez J, Poirier H. Large-scale image retrieval with compressed fisher vectors. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA; June 13–18, 2010. pp 3384–3391.
46. Chuck L. Hadoop in Action. Manning Publications, Shelter Island, NY; 2010. p 325.
47. Zhou K, Hou Q, Wang R, Guo B. Real-time KD-tree construction on graphics hardware. *ACM Trans Graph* 2008;27(5):126.