Abstract—Plant identification is an interesting and challenging research topic due to the variety of plant species. Among different parts of the plant, leaf is widely used for plant identification because it is usually the most abundant type of data available in botanical reference collections and the easiest to obtain in the field studies. A number of works have been done for plant leaf identification. However, it is far from user expectation. In this paper, we propose a new plant leaf identification based on kernel descriptor (KDES). KDES is recently proposed by Bo et al. This is proved to be robust for different object recognition problem. In this paper, once again, the experimental results obtained on two plant leaf datasets show that this approach outperforms the state of the art.

Keywords—Plant leaf identification, kernel descriptor.

I. INTRODUCTION

Plant identification method based on information technology goes from semi-automatic to automatic. Most semi-automatic method [1], [2] imitate the plant identification process of botanist with aid of a multimedia database and a set of questions. The automatic approaches try to extract a number of descriptive features from raw visual plant data and use them in order to determine and output the corresponding plant species.

A plant can be identified by using different parts such as leaf, flower and stem. Among these parts, plant leaf is widely used for plant identification because it is usually the most abundant type of data available in botanical reference collections and the easiest to obtain in the field studies. The production of flowers and fruits occupies a relatively short part of the life of the plant while leaves are present for most or all of its existence.

There are a number of works that have been proposed for plant leaf identification. However, this is still an active and challenging topic for researchers in computer vision community due to the huge variety of leaf. Recently, Bo et al. have proposed a new image feature named kernel descriptor [3]. The authors have proved that kernel descriptor archives the state of the art results on different benchmarks. In this paper, we propose to use this descriptor for plant leaf identification.

II. RELATED WORKS

While working with plant identification based on leaf image, the most crucial part is leaf representation, in which, we need to define and to decide the robust features in order to obtain the best leaf representation. Wu et al. [4] focus on extracting 5 basic geometric features including diameter, physiological length, physiological width, leaf area and leaf perimeter. Based on these geometric features, they also defined 12 digital morphological features used for leaf recognition. For data dimension reduction, they use Principle Component Analysis (PCA) method. Then, a Probabilistic Neural Network (PNN) is implemented for classifying feature vectors. Each feature vector is classified to specific class which has the maximum probability to be the correct class. The authors in [5] also proposed to use two morphological features for plant identification. The first is the morphological covariance on the leaf contour profile and the second is an extension of the recently introduced circular covariance histogram, capturing leaf venation characteristics. Lin et al. [6] calculated some basic descriptors (area, perimeter, length, width, convex hull) and some dimension less shape factors (compactness, roundness, elongation, roughness), they also extracted Fourier descriptor and Bezier descriptors. The authors in [7] analyzed all morphological parameters and combined them with Hu invariants. In [8], the authors apply HOG to plant leaf identification. Since, HOG is a high dimensional feature, the authors proposed to use maximum margin criterion (MMC) method to reduce dimension of feature vector. In [9], Pham et al. apply also HOG and SVM (Support Vector Machine) to plant identification. The authors have compared HOG with Hu moments and the obtained results show that HOG is more robust than Hu moment for plant leaf identification. The authors in [10] analyze the use of invariant keypoint and keypoint descriptor such as SURF for plant leaf identification. Based on this, the authors have developed a plant leaf identification system for Android.
The overview of Kernel based approach for plant leaf identification is illustrated in Fig. 1. This approach consists of two phases: training and testing. In the training phase, for each image in the training phase, firstly we perform preprocessing. The main aim of this step is to normalize training images and extract leaf regions from this image. Then, we do patch-level feature extraction, dictionary building, image-level feature extraction. Finally, each image is represented by a feature vector. We use SVM (Support Vector Machine) to learn models from training images. In the testing phase, for each image, we do the same process as the training phase except that for image-level feature, we reuse the dictionary that is determined in the training phase. In the next section, we describe in detail five main components of our approach that are preprocessing, patch-level feature extraction, dictionary building, image-level feature extraction, classification.

B. Kernel based approach

1) Preprocessing

Since leaf images contain not only leaf but also the background, before extracting leaf information, we have separated leaf regions from other regions in the image. In this work, we apply the Watershed algorithm [13] for image segmentation. This technique can help reduce noises caused by the background. Besides, we apply some techniques seeking to eliminate unexpected influences of petioles as well as random positions of leaves when taking photos. In this sense, petioles are mostly cut off and leaves are rotated to a unified orientation. Fig. 2 illustrates the images of Acer platanoides species before and after preprocessing.

In the Kernel Descriptor (KDES), images are converted into grayscale ones. We then resize it into a defined size range (with fixed maximum and minimum size for both dimensions) retaining its respect ratio. This technique helps reduce the effects of the size-dependence in terms of features without altering the shape of leaves which plays an essential role in the identification.

![Fig. 2. Images of Acer platanoides species before and after preprocessing](image)

2) Patch-level features extraction

The preliminary step of KDES is to extract patch-level features from images. A reduced number of pixels on the image is selected with the aid of a uniform grid. A patch with predefined size will be then taken around each pixel as the area where the computing is done. Patches may overlap one another. Three types of kernels are taken into account: gradient (grad), local binary pattern (lbp) and color (rgb). We now consider gradient as a good example.

For this kernel, in pixel level, first we apply a Gaussian Filter on image, then take gradient values of each pixel through both x and y axis of the image. Once these values are normalized (divided by the magnitude of gradient vector of that pixel), we have a 2-dimensional vector for each interest point on the grid in the form of \( \theta(z) = [\sin\alpha \cos\alpha] \) where \( \alpha \) is the angle of gradient vector at that point.

The main idea of KDES is that we build a metric to evaluate the similarity between two image patches. The exponential metric of Euclidean distance between pixel-level features is selected. For example considering two patches P and Q, the match kernel between their gradient features can be calculated as follows:

\[
K_{\text{grad}}(P,Q) = \sum_{z \in P, z' \in Q} m(z) m(z') k_p(\theta(z), \theta(z')) k_p(z', z')
\]

(1)

where:
- \( z, z' \): denote pixels from two corresponding patches P and Q
- \( m(z), m(z') \): magnitude of gradient vector at \( z, z' \)
- \( k_p(\theta(z), \theta(z')) = \exp[-\gamma \| \theta(z) - \theta(z') \|^2] \): orientation match kernel between two pixels
- \( k_p(z, z') = \exp(-\gamma \| z - z' \|^2) \): position match kernel between two pixels

(Here \( \|a\| \) denotes L2-norm of vector \( a \)).

This match kernel shows a principled way to measure the similarity but can be very computationally expensive when patch sizes are large. An approach to extract low-dimensional features from match kernels is proposed. This process is achieved by two steps:

- apply sufficient finite-dimensional approximation by projecting features vector into a set of basis vectors
- apply KPCA (Kernel Principle Component Analysis)

Since pixel attributes are low-dimensional vectors, an approximation can be well achieved by sampling sufficient basis vectors using the fine grid over a support region. After this selection, we project our kernels \( k_p \) and \( k_p' \) into the grid using kernel distance function (exponential function of Euclidean distance). Now match kernel between patches can be seen as:

\[
k_o(\theta(z), \theta(z')) = [G k_o(\theta(z),X)]^T [G k_o(\theta(z'),X)]
\]

(2)

where:
- \( X \): set of sampled basis vectors
- \( G \): coefficient matrix (constructed from basis vectors)

This equation shows us an effective way to build a special type of features which can be easily used for matching. In order to calculate match kernel between two patches, each pixel of a patch needs to be matched to all the ones of the other. Hence a Kronecker product appears in the following formula showing how to compute patch-level features:

\[
F_{\text{grad}}(P) = \sum_{z \in P} m(z) \Phi_o(\theta(z)) \bigotimes \Phi_p(z)
\]

(3)

where \( \Phi_o, \Phi_p \) denote orientation and position match kernel of the pixels in a patch with the selected basis vectors (simply understood as a projection). If the basis vectors are kept fixed, all the features vectors are of the same dimension.

Considering the high dimension of features vectors (due to Kronecker product), KPCA is applied with the learned eigenvectors.

3) Dictionary building

After extracting patch-level features, we apply K-means algorithm [14] to build the dictionary. This building consists of two stages:
Sample a number of patches in each image. This number must be limited by a threshold for the avoidance of too large data.

K-means is applied for a fixed number of loops for a better accuracy. Initially, for each cluster, a center point is chosen at random. Then for each loop, each patch is classified into one cluster of which the center point is the nearest to the patch (based on Euclidean distance) among all the center points. The end of each loop, a new center point is calculated for each cluster as its midpoint.

The definitive center points of the clusters form the dictionary used to extract image-level features.

4) Image-level features extraction

In each layer, image-level features are computed on a learned dictionary. Image-level features are extracted using spatial pyramid matching throughout a number of layers (layer 0, layer 1, layer 2, ...). In layer \( k \), an image is divided into \( 2^{2k} \) cells. The total number of cells generated by a division of \( M \) layers is \( \frac{4^M - 1}{3} \). For each cell, we apply the following strategy:

- Find all the patches of which the center point is located in the cell.
- For each patch found, calculate its distances to all the words in the dictionary, then find its “nearest” visual word. Here we have a list of the corresponding words of the patches.
- For each visual word in the above list, we maintain only one patch which is the nearest to it.
- Average value of distances between the remaining patches and all the visual words form an n-dimensional vector (\( N \) is the number of visual words learned through K-means algorithm).
- A weight is assigned to each layer, the higher layer level it is, the more significant weight it is assigned.

In conclusion, if we build a dictionary of \( N \) visual words and divide an image by \( M \) layers, then its image-level features is represented by a vector of \( N \cdot \frac{4^M - 1}{3} \) dimensions.

Fig. 3. Some images with the red points representing the positions of patches that are used for image-level feature extraction.

5) Classification

Once image-level features are computed, every image has a features vector of the same dimension. An SVM (Support Vector Machine) classifier is learned by “one-vs-all” strategy. For instance we need to build a classifier for \( C \) classes. We first build a hyperplane separating the first class and the others. This process is proceeded continuously for all the classes. Finally, we combine all the hyperplanes to have the final classifier.

IV. EXPERIMENTAL RESULTS

A. Dataset

We have tested our approach with two datasets (see Tab. 1). The first dataset is Flavia dataset [4]. This database consists of 1907 images (1600*1200 pixels) of 32 species. The leaf in this database is taken with simple background (see Fig.4). In order to compare our proposed method with the existing ones, we divide this dataset for training and testing set by the same way as explained in [11]. This means that for each species, we pick 10 images for testing set and the rest belonging to training set.

Fig. 4. Examples of leaf images in the Flavia database

The second dataset is leaf images of the ImageClef 2013. The database of ImageClef 2013 contains images of different organs (leaf, flower, etc.) of 250 plant species in France. In our work, we use only a subset of ImageClef 2013 that contains leaf images taken in a simple background. This subset contains images of 126 species for training dataset and 70 species for testing dataset. The second dataset is much more challenging than the first one because of not only the high number of classes but also low intra-class and high inter-class similarity. The images of the same species can be relatively different because they are taken in different period of time. Figure 5 illustrates several images of the same species. These images are different from each other in term of color and the number of lobes.

(a)

Fig. 5. Several leaf images of (a) Acer negundo and (b) Broussonetia papyrifera species. These images are different from each other in term of color and the number of lobes

Figure 6 shows the high inter-class similarity characteristic of this dataset. There are images of different species with very similar shape and configuration.
Fig. 6. Images of different species with very similar shape and configuration.

Table 1. Flavia and ImageClef dataset

<table>
<thead>
<tr>
<th>Name</th>
<th># species in training dataset</th>
<th>#species in testing dataset</th>
<th>#training images</th>
<th>#testing images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flavia</td>
<td>32</td>
<td>32</td>
<td>1585</td>
<td>320</td>
</tr>
<tr>
<td>ImageClef 2013</td>
<td>126</td>
<td>70</td>
<td>7525</td>
<td>1250</td>
</tr>
</tbody>
</table>

B. Results

We evaluate the performance of the algorithm by calculating the accuracy this is the ratio between the number of right classified images and the number of testing ones.

Table 2. Average accuracy with Flavia dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Average accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method based on Probabilistic Neural Network [4]</td>
<td>90.3</td>
</tr>
<tr>
<td>Method based on SURF and BoW [10]</td>
<td>95.94</td>
</tr>
<tr>
<td>Our proposed method</td>
<td>97.5</td>
</tr>
</tbody>
</table>

Table 3. Average accuracy with ImageClef 2013 dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Average accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method based on SURF and BoW [10]</td>
<td>39.8</td>
</tr>
<tr>
<td>Our proposed method</td>
<td>58.0</td>
</tr>
</tbody>
</table>

We analyze the accuracy of some species in Tab. 5. The species with high identification accuracy appear to have a large number of samples in the training set. Furthermore, they have a distinctive appearance (see Fig. 7). In contrast, the poor performances for some classes are understandable due to their tiny numbers of training images. For some species, the number of training images is 2 or even 1. Moreover, theirs appearance vary largely. Figure 8 illustrates some testing and training images of two species with very low accuracy. As we can see, these images are difficult even for human identification. In addition, the great inter-class similarity of this dataset can be considered as a big challenge.

Table 4. Average accuracy obtained with default parameters and with two modifications on ImageClef 2013 dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Average accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our proposed method with default parameters</td>
<td>58.0</td>
</tr>
<tr>
<td>Our proposed method with two modifications</td>
<td>63.4</td>
</tr>
</tbody>
</table>

Table 5. Identification results of several species

<table>
<thead>
<tr>
<th>Name of species</th>
<th># of training</th>
<th># of testing</th>
<th>Average Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ruscus aculeatus</td>
<td>180</td>
<td>48</td>
<td>100</td>
</tr>
<tr>
<td>Ulmus minor</td>
<td>280</td>
<td>28</td>
<td>100</td>
</tr>
<tr>
<td>Prunus dulcis</td>
<td>32</td>
<td>58</td>
<td>25.9</td>
</tr>
<tr>
<td>Viburnum tinus</td>
<td>129</td>
<td>35</td>
<td>11.4</td>
</tr>
</tbody>
</table>

From the results, we can see that with Flavia dataset, the performance of our proposed approach and that of method [10] is relatively equal. While working with a more challenging dataset like ImageClef 2013, our approach proves its robustness. It obtains 58% of accuracy while the method based on SURF and BOW gets 39.8% of accuracy.

While working KDES, we obtain two observations. The first one is that the number of extracted patches for each image is relatively huge. However, a high number of patches are irrelevant. We can remove these patches by using the gradient value. Therefore, for each image, to calculate good patch-level features, we maintain only a half of patches which have the most significant gradient sums. The second observation is that in image-level extraction, for each visual word, we keep only one patch that is the nearest one. This can lose some important patches. In order to avoid this, we keep two nearest patches for each visual word. We test our proposed approach with two modifications on ImageClef 2013 dataset. The obtained results are shown in Tab. 4. We can see that these two modifications improve clearly the classification results.

For KDES based approach we use 200 eigenvectors for patch-level features extraction. In image-level features calculation, the number of visual words selected is 1000 and “spatial pyramid matching” is applied with 3 layers resulting in a 21000-dimensional vector for each image.

As we can see, the kernel based method obtains the best result with both datasets. Since the ImageClef2013 dataset is much more difficult than Flavia dataset, the kernel method cannot give the result as good as that with Flavia dataset. However, this result is better than the state of the art for ImageClef 2012 [5] even when the ImageClef 2013 is more complex.
In terms of computational time, our proposed approach takes one second for each image on a high-performance computer with 32GB Ram. We use MATLAB version 64 bit on Windows.

V. CONCLUSIONS

This paper presents a kernel descriptor based method for plant leaf identification. The obtained results with both datasets prove the robustness of this method. As we described in the previous section, this method obtains very good result with Flavia dataset. While working with a more challenging dataset such as ImageClef 2013, the result is still limited. In the future, we would like to extend this work by building a classifier for species with similar leaf or by taking into account other information of plant such as flower, stem [12].

ACKNOWLEDGEMENTS

This research is funded by Hanoi University of Science and Technology (HUST) under grant number T2014. CT-12.

REFERENCES

1. IOWA, Needles Sharp or Blunt?, 2012.