Computers and Electronics in Agriculture 164 (2019) 104924

Contents lists available at ScienceDirect

Computers and Electronics in Agriculture

journal homepage: www.elsevier.com/locate/compag

Original papers

Fully automatic segmentation method for medicinal plant leaf images in complex background☆

Liwen Gao[a], Xiaohua Lin[b,*]

[a] School of Medical Information Engineering, Guangzhou University of Chinese Medicine, Guangzhou, China
[b] School of Chinese Materia Medica, Guangzhou University of Chinese Medicine, Guangzhou, China

ARTICLE INFO

Keywords:
Complex background
Leaf
Image segmentation
Vein
Fully convolutional network

ABSTRACT

Vein is a vascular bundle in a leaf which often extends from the leaf center to the edge. In this paper, an accurate and fully automatic segmentation method for medicinal plant leaf images in complex background is proposed by taking vein enhancement and extraction in the image as the core. It has laid a solid foundation for the non-destructive machine identification of medicinal plants taking a mobile phone as the terminal.

Gradient magnitude image and gradient angle image are obtained by directly solving color RGB images. By fully applying the objective rule that the veins are linear and the consistency of gradient angles of adjacent vein points is relatively high, the veins in the gradient magnitude images are further enhanced by the standard deviation of the gradient angles to obtain the vein enhancement image. Based on this image, the OTSU method is used to obtain a binary image taking the veins as the foreground, and the main veins are detected from it. In other areas beyond the main veins in the vein enhancement map, fine veins are detected and then connected to the main veins. Then, a foreground marker image targeting the veins is obtained. The background marker image is segmented by using the OTSU method in each component image of the color image and then screened after a specific ratio is calculated. Based on the foreground markers and the background markers, the marker-controlled watershed method is applied to obtain the binary segmentation result.

A series of experimental tests based on a self-built database and another widely used database show that the accuracy of the proposed method is better than many main fully automatic image segmentation methods including deep learning FCN. Its running speed is also fast and slightly lower than that of the OTSU threshold segmentation.

1. Introduction

Medicinal plants are the main source of Chinese herbal medicines and a material basis for disease treatment by traditional Chinese medicine. In recent years, medicinal plant resources have shrunk significantly due to the deterioration of ecological environment (Yang et al., 2000). To strengthen the protection of medicinal plants is extremely urgent.

If we can thoroughly figure out the geographical distribution of endangered medicinal plants and build a geographic information resource library, it will play an important supporting role in protection, introduction and utilization of wild medicinal plants. As it is difficult for ordinary people to identify the plant types in a complex ecological environment, rough sampling can only be made for current resource survey, and there is still a certain distance from the comprehensive and in-depth investigation. Even so, a lot of manpower and resources have been consumed.

In order to break this deadlock, we propose to photograph the image of the plant leaf and then carry out the machine identification so that more people with only a preliminary basis can accurately identify the current plant species in the wild by simple mobile phone operation. Like most of the image identification problems, segmentation of leaf images is the first difficulty. In the early stage, we proposed an accurate segmentation method for the image of medicinal plant leaf in complex background (Gao and Lin, 2018). However, the method requires manual participation.

In order to further improve the convenience and try to ensure the accuracy of segmentation, we continue to carry out in-depth research

☆ This paper is supported by the Higher Education Teaching Reform Project of Guangdong Province (143), the Elite Youth Education Program of the Guangzhou University of Chinese Medicine (QNYC20140203).

* Corresponding author.
E-mail addresses: glw@gzucm.edu.cn (L. Gao), networkspace898@163.com (X. Lin).

https://doi.org/10.1016/j.compag.2019.104924
Received 28 March 2019; Received in revised form 18 July 2019; Accepted 23 July 2019
Available online 30 July 2019

and propose a fully automatic segmentation method for the image of medicinal plant leaf in complex background. The method firmly grasps the morphological characteristics of the unique leaf vein structure in the leaf area, extracts it as accurately as possible and obtains the foreground markers of the image. Finally, the image segmentation is completed by the marker-controlled watershed method combining with additionally extracted background markers.

The experimental results show that the proposed method has made significant progress in the automatic segmentation of medicinal plant leaf images in complex background. Its segmentation accuracy is superior to many automatic segmentation methods including FCN (Fully Convolutional Networks).

In this paper, Section 2 explains relevant researches; Section 3 introduces and details our method; Section 4 shows the segmentation result and discusses it; finally, Section 5 makes a brief summary.

2. Related works

There have been many reports on the research of plant machine identification based on leaf images. According to the mode of image acquisition, it can be divided into two types. The first one is to pick off the leaf and then a simple background leaf image is obtained by shooting or scanning. Its advantage is that the image is easily segmented and the segmentation accuracy is high. Its disadvantage is that it damages the plant. This method is used for most of the existing researches (Larese et al., 2014; Larese et al., 2014; Cem and Önsen, 2015; Aimen and Faisal, 2015; Naresh and Nagendraswamy, 2016). The second one is to directly shoot a leaf on the branch to get a complex background leaf image. Its advantage is that it does not cause any damage to the plant. Its disadvantage is that besides the target leaf, the image also contains background objects such as branches, soil and other leaves. With difficult segmentation and low segmentation accuracy, it seriously affects the accuracy of subsequent classification and identification.

Due to the non-destructive advantage, we focus on researching machine identification based on leaf images in complex background. Then, the accurate segmentation of leaf images in complex background is the primary problem to be solved. Although the current deep learning classification method can map from image to category, it is difficult to evenly expand the plant leaf image samples to a large number. A small number of types are common, but most of the types are difficult to be found, and some are rarely available. In this case, the applicability of the deep learning classification method needs to be verified. Furthermore, if the image segmentation can be accurately conducted to eliminate the image background (namely disturbance term), it is advantageous no matter what the classification method is used later.

There are many methods to segment images. Some of them require manual participation. At the previous stage, we carried out in-depth research and proposed a high-accuracy image segmentation method in complex background with manual participation (Gao and Lin, 2018). However, manual participation still affects the user experience. Obviously, the fully automatic method is more popular.

There are many fully automatic image segmentation methods at present, and the famous ones include:

OTSU (1979) is a threshold segmentation method for binarizing images proposed by the Japanese scholar OTSU in 1979. The method selects the optimal segmentation threshold based on the maximum variance between clusters. The proposed method is very sensitive to noise and target size and only produces a good segmentation effect on the images with obvious foreground and background contrast.

Mean shift (Comaniciu and Meer, 2002), a hill-climbing algorithm based on kernel density estimation, can be used for clustering, image segmentation and tracking. Image segmentation is to find the class label for each pixel. Class label depends on the cluster its eigenspace belongs to. For each cluster, a class center should be available. It is considered by mean shift that the maximum point of the probability density function is the class center. Similar to OTSU, mean shift produces a good segmentation effect on the images with obvious foreground and background contrast. Otherwise, the effect is poor.

The GraphCut (Boykov and Jolly, 2001) associates the image segmentation with the min cut of the graph. The essence of the segmentation method based on graph theory is to remove the specific edges by minimizing cost and divide the graphs into several subgraphs to achieve segmentation. Cost contains zone item and boundary item. It is advantageous in dealing with the images whose pixel values have a significant difference, but it is less effective to deal with the images in complex background and with closer foreground and background.

In recent years, deep learning has developed rapidly. The FCN (Fully Convolutional Networks) method for deep learning image segmentation also emerges at the right moment (Shelhamer et al., 2017). It has repeatedly demonstrated a superior performance on various complex image segmentation problems. FCN is trained end-to-end and pixels-to-pixels on semantic segmentation to achieve pixel-level classification of images and solve semantic segmentation. If time and memory limitations are not considered, it can accept input images of any size in theory. However, FCN also has its shortcomings: ① The obtained results are still not fine enough and insensitive to the details in the image. ② Pixels are classified individually, and the relationship between pixels is not fully considered.

There is no such a method that can solve all the problems. A specific practical research is still required for specific problems. During research, we noticed that the leaf has veins. At this moment, we think of another classic segmentation method – marker-controlled watershed segmentation (Roerdink and Meijster, 2000).

It is not limited by the shape of the target area, and it is suitable for segmentation of leaf images of various shapes and types. Moreover, it is very suitable for the situations in complex background. As long as the foreground markers and the background markers are input accurately, it can usually produce a good segmentation result. However, how to accurately obtain the foreground markers and the background markers is a big problem. In many cases, only manual participation can be used.

We insisted on taking fully automatic segmentation as goal. After attempts for times, we have extracted the main veins more accurately to form a foreground marker image. Combining the obtained background marker image, the marker-controlled watershed segmentation method is applied to realize the automatic segmentation of the leaf images in complex background. The specific process for the method and the comparison of the segmentation results for various methods will be explained in detail below.

3. Methodology

3.1. Databases

In order to support the research on medicinal plant machine identification, we spent more than a year on shooting and collecting the leaf images in complex background on Yaowang Mountain, Guangzhou University of Chinese Medicine, Guangzhou University Town, Panyu District, Guangzhou, China. There are more than 1600 species of medicinal plants on the mountain, all of which have been migrated from all over the country. After many years of hard work by the staff of Guangzhou University of Chinese Medicine, the plants have been flourishing.

So far, a total of 88 species have been collected, in which each category is provided with 100–115 images. For the information on the plant species name and medicine name, please refer to (Gao and Lin, 2018).

In order to verify the effectiveness of the segmentation algorithm, 5 images are randomly selected from each of 88 leaf images, and the standard segmentation results are manually identified one by one as a reference. Thus, Database 1 is formed. Eight of these images are shown in Fig. 1. The segmentation result of the proposed algorithm is attached

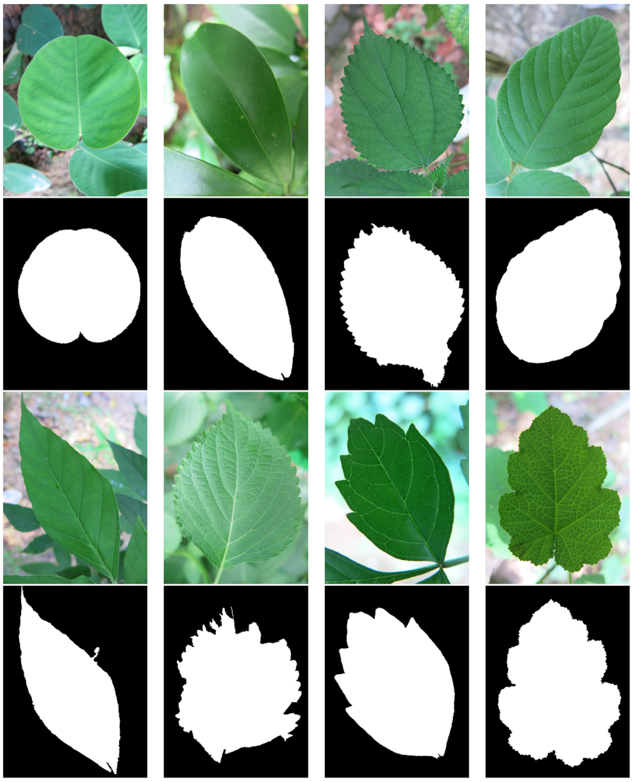


Fig. 1. 8 leaf images of Database 1 and segmentation results after using the proposed algorithm.

with the original image.

In order to train the deep learning algorithm, 5 images are randomly selected from each category of 88 species after the images selected in Database 1 are excluded, and the standard segmentation results is manually identified one by one. Thus, a training set corresponding to the test set Database 1 is formed, which is named as Database 0.

In addition, to make a direct comparison to other good algorithms reported, Database 2 (Grand-Brochier et al., 2015) containing 233 leaf images with complex backgrounds is also used here. The images are extracted from Pl@ntLeaveI (Goëau et al., 2011), with manually identified standard segmentation results. Downloaded from "http://liris.univ-lyon2.fr/reves/content/en/databases.php". And the Pl@ntLeaveI database is just a part of the databases done by the organizers of "Plant Identification" challenge for the conferences ImageCLEF 2011 and 2012. Eight sample images are now demonstrated as shown in Fig. 2. The segmentation result of the proposed algorithm is attached with the original image.

Similarly, in order to train the deep learning algorithm, 117 images are used as a training set, which is named as Database 2(a), and the remaining 116 images are used as a test set, which is named as Database 2(b).

3.2. Process overview

The overall flow for the proposed algorithm is shown in Fig. 3.

Firstly, simple front segmentation is conducted. As the background of some leaf images is not a green leaf or a green grass, and it is a contrasting object when compared with the color of the target leaf, such as bricked pavement. In the first step, OTSU segmentation should be firstly tried on different component images. If it is successful, the algorithm will be over.

Then, the proposed algorithm detects the background markers in each component image and selects one of the background marker images as an optimal background marker image according to our

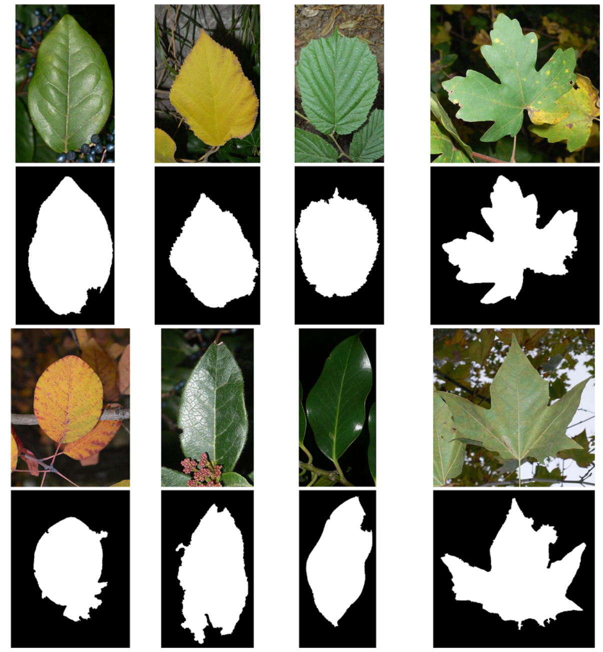


Fig. 2. 8 leaf images of Database 2 and segmentation results after the proposed algorithm.

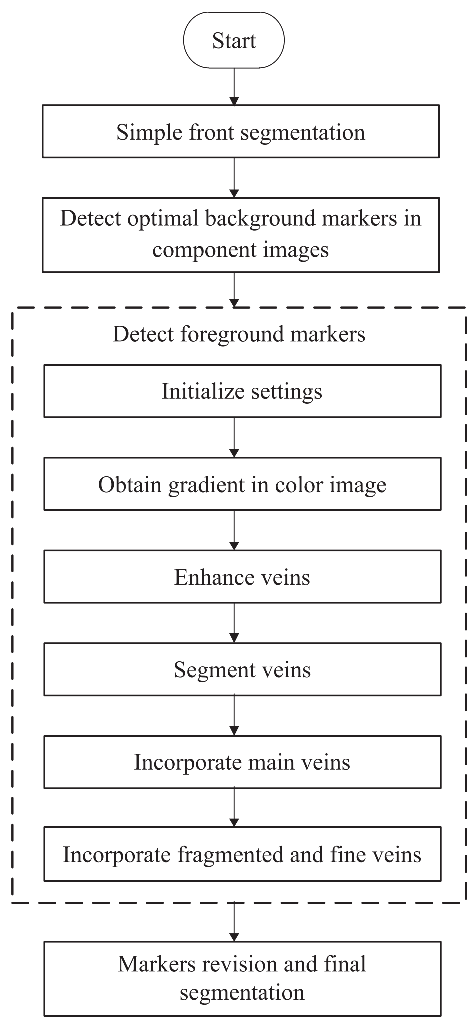


Fig. 3. General Idea of the Algorithm.

established criteria.

Next, the foreground markers are detected. Specifically, after settings are initialized, the gradients are directly calculated on the color image, the gradient amplitude image and the gradient angle image are obtained, and then the gradient angle image is used to enhance the veins on the gradient amplitude image according to such a characteristic that a leaf contains veins. After the veins are segmented, the main veins are combined with a small area in the center of the image and then incorporate fragmented and fine veins to obtain foreground markers.

Finally, the foreground markers and the background markers are revised, and the marker-controlled watershed segmentation method is applied for segmentation.

3.3. Simple front segmentation

Step 1: The color RGB image is reduced to an appropriate scale. Calculate first the ratio by using the equation:

$$\text{ratio} = (1200 \times 900)/(\text{the row number of the image} \times \text{the column number of the image}) \quad (1)$$

If *ratio* is < 1, the image is reduced by a multiple of the square root of *ratio*. Otherwise, there is no need to reduce it. The image is marked as *currentImage*. In practice, if other scales are used, the calculation parameters of *ratio* are adjusted.

Step 2: *currentImage* is converted into the representation of HIS and Lab models respectively. It is preliminarily judged whether the foreground and the background colors are sharply contrasted in *H* component image, *I* component image, *a* component image and *b* component image respectively. If this situation is detected, record the corresponding winning coefficient *coef*, judgement marker *flat* and segmentation result *BW*.

As shown in Fig. 4, Fig. 4(a) is the original image of the leaf *Cyathula prostrata (L.) Bl.* Fig. 4(e–h) are *H* component image, *I* component image, *a* component image and *b* component image, respectively.

The detailed process for preliminarily judging whether the foreground and background colors are sharply contrasted and calculating the winning coefficient is as follows:

① The binary image *BW* is obtained by the maximum variance between clusters (OTSU) for the input component image. Fig. 4(i–l) are *BW* after *OTSU* segmentation of *H* component image, *I* component image, *a* component image and *b* component image.

② Calculate the ratio (*frameCof*) of the pixels whose values are “1” of the four frames in *BW*.

③ Calculate the number (*area*) of the pixels whose values are “1” in *BW*.

④ Delete all small areas in *BW* and leave the largest area only.

⑤ Calculate the number (*targetArea*) of the pixels whose values are “1” in *BW*.

⑥ The ratio of *targetArea* and *area* is used as *foregroundCoef*.

⑦ If *frameCof* < 0.1 and *foregroundCoef* > 0.9, it is considered that the foreground and background colors are sharply contrasted, *flat* is set as 1, and (1 − *frameCof*) * *foregroundCoef* is used as the winning coefficient (*coef*). Otherwise, reverse *BW* and repeat Step ②–⑥. Judge whether *frameCof* < 0.1 and *foregroundCoef* > 0.9. If so, *flat* is set as 1, and (1-*frameCof*)**foregroundCoef* is used as the winning coefficient (*coef*). Otherwise, *flat* is set as 0.

Note: In Fig. 4, the *flat* of *S* component image segmentation result is 1, *coef* is 0.9536, and the outputted *BW* is as shown in Fig. 4(b). The *flat* of *b* component image segmentation result is 1, *coef* is 0.8722, and the outputted *BW* is as shown in Fig. 4(c). The *flat*s of the remaining component images are 0.

Step 3: In the previous step, if *flat* obtained after operation is 0 for each component image, go to Step 8. If more than one is 1, the segmentation image *BW* corresponding to the maximum *coef* is taken as *logicImage*. (In Fig. 4, select the segmentation image *BW* of *S* component image whose *flat* is 1 and the *coef* is the maximum, as shown in Fig. 4(b).)

Step 4: Check how much the foreground area of *logicImage* extends to its four frames. If there are more than 3 places, go to Step 8.

In general, when a leaf is shot, it will be shot as completely as possible, so it is unlikely that the leaf will touch the frame. Occasionally, the petiole is too long, or the leaf tip is thin and long and stretches to the frame, which is already accepted with great tolerance. If the frame is touched for more than 3 times, then *logicImage* will not be used as a segmentation result.

b component image in Fig. 5(b) of *Lonicerae Confusae (Sweet.) DC.* leaf image in Fig. 5(a) is conducted with OTSU in Step 2 to obtain the binary image in Fig. 5(c). After the maximum foreground area is selected, it is judged that *frameCof* is < 0.1 and *foregroundCoef* is > 0.9. (In this example, *frameCof* is 0.0970, and *foregroundCoef* is 0.9985.) The segmentation result is outputted, as shown in Fig. 5(d). Next, the segmentation result is selected as *logicImage* in Step 3. However, it is obvious that the result has multiple connections with the boundary, and the segmentation effect is not good. This problem is found in Step 4, and the segmentation result was abandoned.

Step 5: In the middle of *logicImage*, cut a subgraph *logicImageCrop* whose length and width are 0.6 times of *logicImage*. The four frames of *logicImageCrop* are paralleled to those of *logicImage*, and their central points are overlapped. The number (*logicImageCropArea*) of the pixels with a value of “1” in *logicImageCrop* is calculated. The number (*logicImageArea*) of the pixels with a value of “1” in *logicImage* is calculated. The ratio of *logicImageCropArea* and *logicImageArea* is calculated. If it is not greater than 0.5, go to Step 8.

The purpose of this step is to detect whether the foreground area is too small and located near the frame of the image in *logicImage*. If so, the segmentation result is abandoned. When a leaf image is shot, the leaf is often shot in the center of the image and will occupy most of the image without the aforementioned appearing.

H component image in Fig. 6(b) of *Clematis chinensis Osbeck* leaf image in Fig. 6(a) is conducted with OTSU in Step 2 to obtain the binary image in Fig. 6(c). After the maximum foreground area is selected, it is judged that *frameCof* is < 0.1 and *foregroundCoef* is > 0.9. (In this example, *frameCof* is 0.0817, and *foregroundCoef* is 0.9993.) The segmentation result is outputted, as shown in Fig. 5(d). Next, the segmentation result is selected as *logicImage* in Step 3. However, it is obvious that the obtained result is completely wrong. Fortunately, in Step 5, it is detected that its foreground area is too small and located near the frame of the image, so it is abandoned.

Step 6: Closing operation is conducted for *logicImage*.

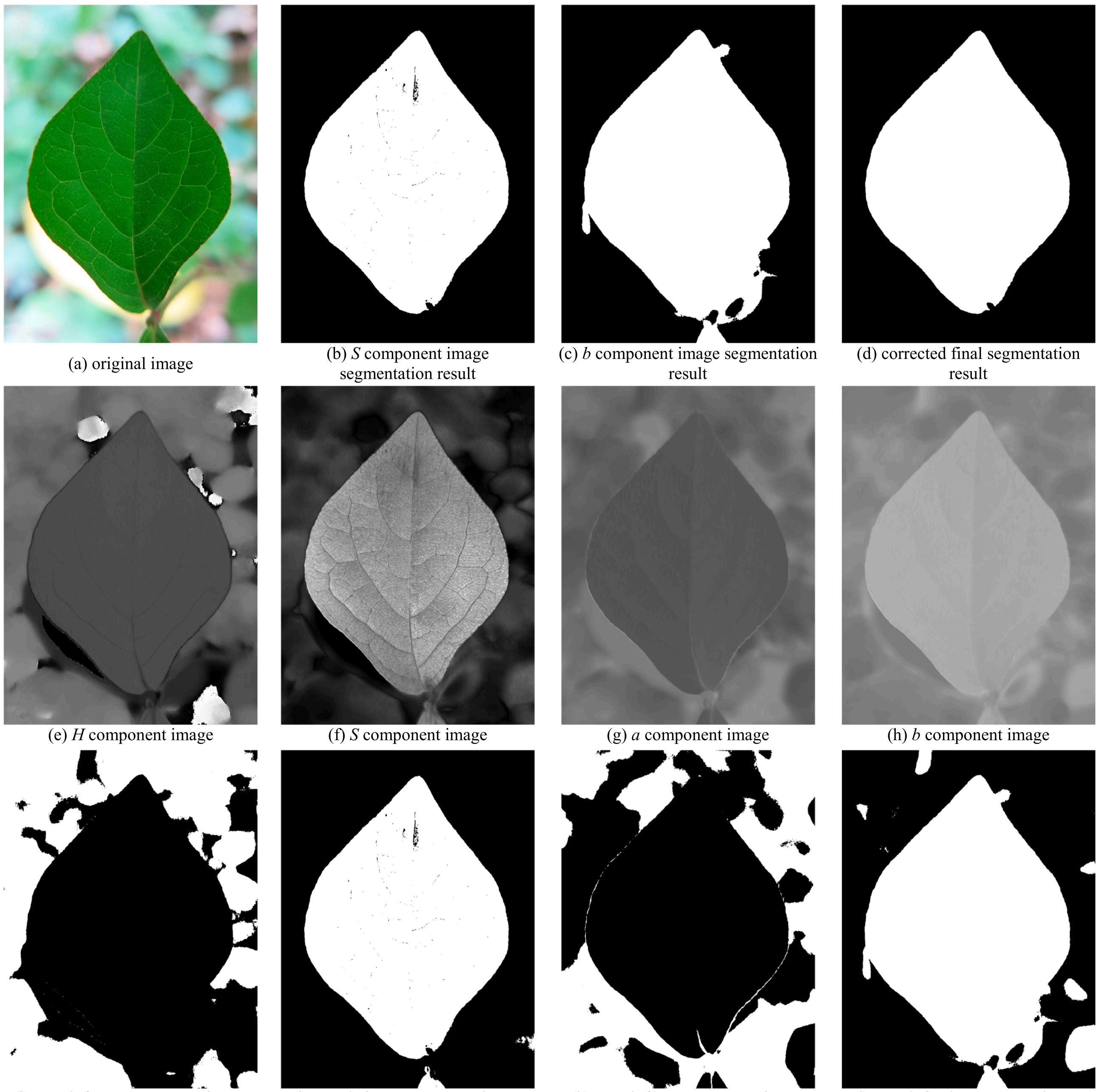
Step 7: Holes are filled for *logicImage* to obtain the final segmentation result. The algorithm is over. *logicImage* in Fig. 4(b) passes the tests in Step 4 and Step 5 and is corrected in Step 6 and Step 7 to obtain the result in Fig. 4(d).

3.4. Detection of optimal background markers in component images

Step 8: Set the credible foreground scale parameter *para* (such as 0.125).

Step 9: Detect the background markers on each OTSU segmentation result *BW* corresponding to *H* component image, *S* component image, *a* component image and *b* component image mentioned above. The specific process is as follows:

(1) In the middle of *BW*, cut a subgraph *credibleForeground* whose length and width are *para* × 2 times of *BW*. The four frames of the subgraph are paralleled to those of *BW*, and their central points are overlapped. As mentioned above, when a leaf image is shot, the leaf is often shot in the center of the image and will occupy most of the image. Therefore, it is highly possible that the target leaf may cover a small area in the center of the image.
(2) Calculate the ratio *coefOfCredibleForeground* for the pixels whose



(a) original image (b) S component image segmentation result (c) b component image segmentation result (d) corrected final segmentation result (e) H component image (f) S component image (g) a component image (h) b component image (i) Result for H component image after OTSU segmentation (j) Result for S component image after OTSU segmentation (k) Result for a component image after OTSU segmentation (l) Result for b component image after OTSU segmentation

Fig. 4. *Cyathula prostrata (L.) Bl.*

values are "1" in *credibleForeground*.

(3) If *coefOfCredibleForeground* is within the interval [0.2, 0.8], the segmentation accuracy of *BW* is considered to be doubtful. (As mentioned above, the target should cover the central small area of the image.), and it is not suitable for detecting the background markers on the *BW* basis. The detection process is over, and detection failure message is returned.

(4) If *coefOfCredibleForeground* is less than 0.2, *BW* will be saved as *backgroundCandidate*, and *BW* is reversed and saved as *BW* again. Otherwise, the *BW* reverse result is *backgroundCandidate*.

(5) Calculate the ratio *frameCof* for the pixels whose values are "1" on the four frames of *BW*.

(6) If *frameCof* is greater than 0.6, the segmentation accuracy of *BW* is considered to be doubtful (the background pixels whose values are "0" on the four frames should be in majority). The detection process is over, and detection failure message is returned.

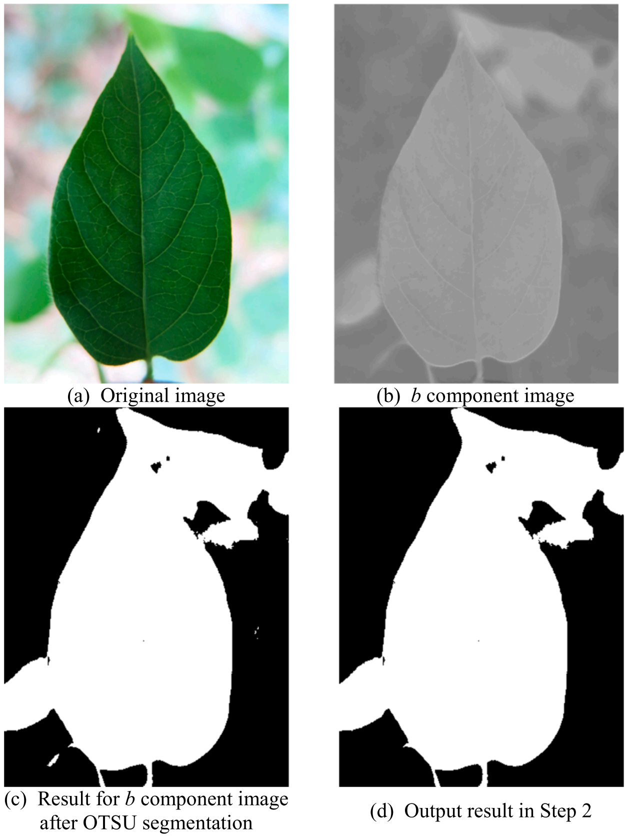
(7) Perform a mathematically morphological eroding operation for *backgroundCandidate*.

(8) Set the pixels of the four frames of *backgroundCandidate* as "1".

(9) In *backgroundCandidate*, select and reserve the area that is connected to the point at the upper left corner of the image and delete other areas. The result is named as background. The background markers have been basically obtained as per requirement. There may still be holes in it. Holes can be eliminated by the following three steps, which is helpful to save time consumption of marker-controlled watershed segmentation.

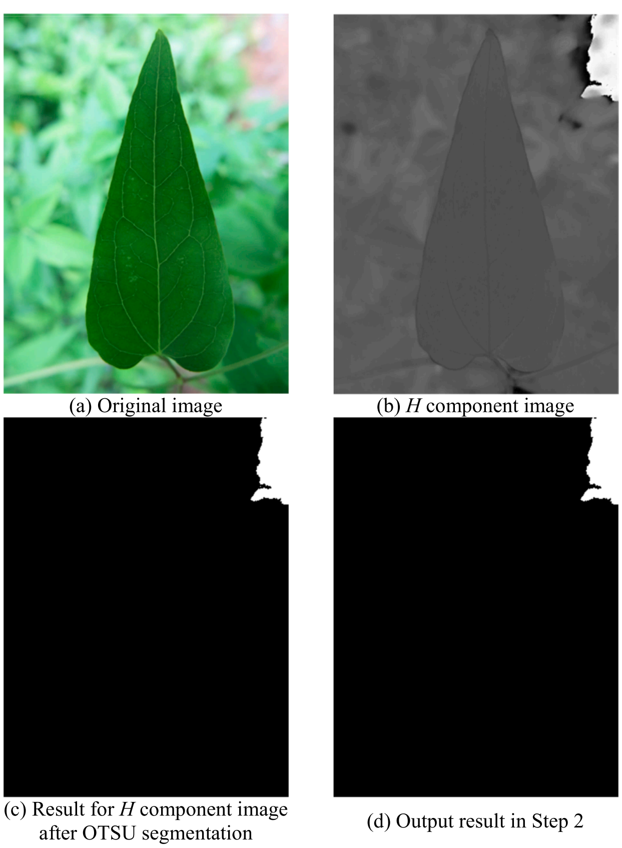
(10) Reverse background and save it as reverseBackground.

(11) In *reverseBackground*, select and reserve the area that is connected to the central point of the image and delete other areas. The result is named as *reverseBackground2*.



(a) Original image (b) *b* component image (c) Result for *b* component image after OTSU segmentation (d) Output result in Step 2

Fig. 5. *Lonicerae Confusae (Sweet.) DC.*



(a) Original image (b) *H* component image (c) Result for *H* component image after OTSU segmentation (d) Output result in Step 2

Fig. 6. *Clematis chinensis Osbeck.*

(12) Save the reversion result of the *reverseBackground2* as *background*.
(13) In the middle of *reverseBackground2*, cut a subgraph *credibleForegroundClean* whose length and width are *para* × 2 times of *reverseBackground2*. The four frames of the subgraph are paralleled to those of *reverseBackground2*, and their central points are overlapped.
(14) Calculate the ratio *coefOfCredibleForegroundClean* for the pixels whose values are "1" in *credibleForegroundClean*.
(15) Calculate

$$backgroundCoef = 1 - (1.01 - coefOfCredibleForegroundClean) \times frameCof \tag{2}$$

Return *background*, *backgroundCoef* and detection success message.

Step 10: In the four background markers detection results obtained in the previous step, select the *background* which is detected successfully and whose *backgroundCoef* is the greatest as *bestBackgroundDetect*, and register *bestBackgroundDetectFlat* as true. If the detection results for four background markers return failure messages, the *bestBackgroundDetectFlat* is registered as false.

Step 11: If *bestBackgroundDetectFlat* is true, correct *bestBackgroundDetect*. The process is as follows: firstly erode *bestBackgroundDetect*, set the pixels on its four frames as"1", select and reserve the area that is connected to the upper left corner of it, and delete other areas.

As shown in Fig. 7, Fig. 7(a) is an original image of *Polygonum chinense L.* leaf. Fig. 7(e–h) are *H* component image, *S* component image, *a* component image and *b* component image. Fig. 7(i–l) are the binary images obtained from these four component images after OTSU segmentation. Fig. 7(m–p) are the four background marker images obtained from the four binary images detected from Step 9. Meanwhile, the corresponding *backgroundCoefs* are measured to be 0.9942, 0.9972, 0.9964 and 0.9986. That is, the background markers (Fig. 7(p)) detected on the binary image of *b* component image correspond to the largest *backgroundCoef*. In Step 10, the background markers are selected as optimal background markers, and they are corrected in Step 11 to obtain *bestBackgroundDetect*, as shown in Fig. 7(b).

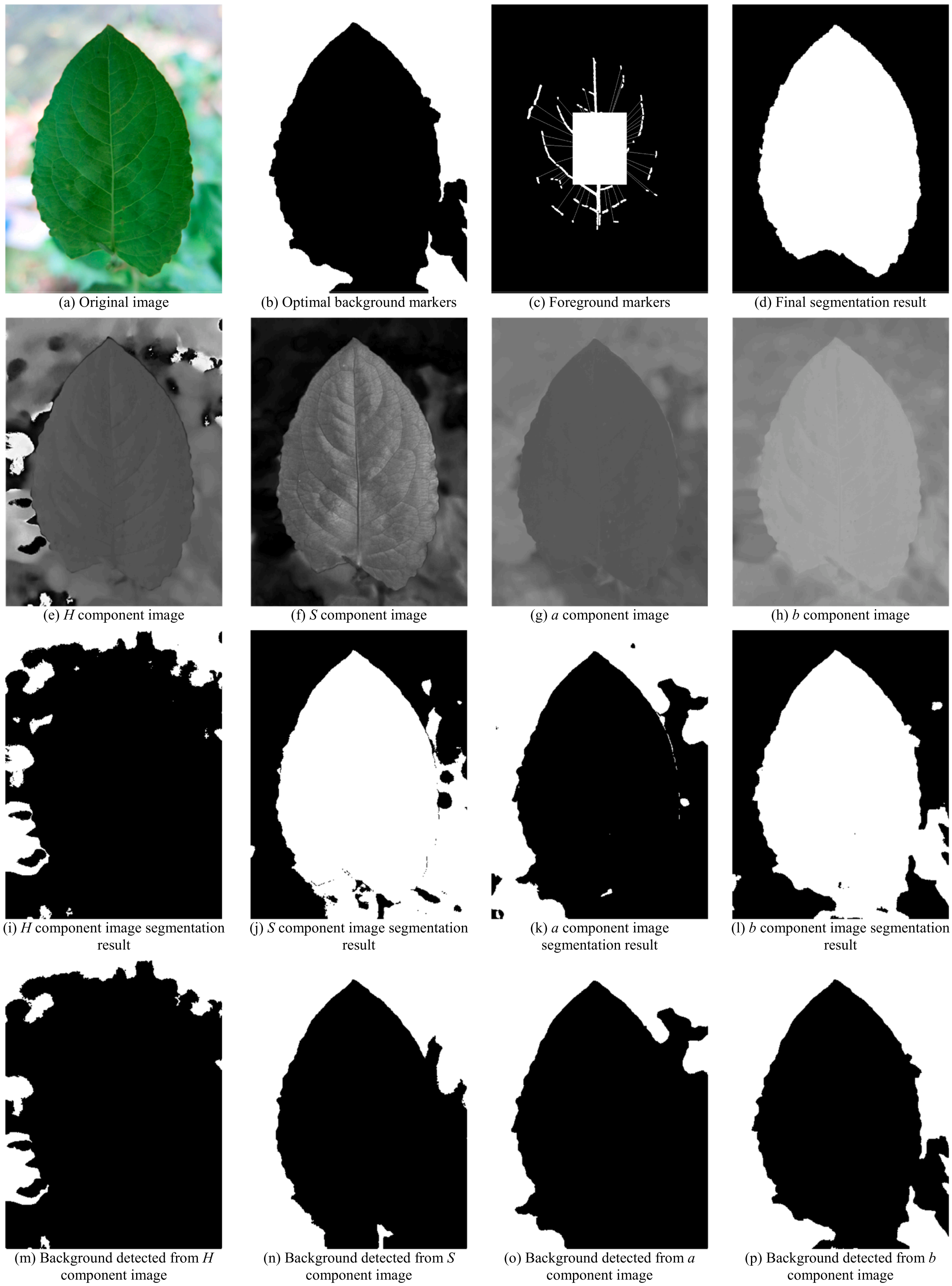
Next, the foreground markers in Fig. 7(c) will be detected. Based on the two marker images, the segmentation result in Fig. 7(d) is obtained.

And, the detection of the foreground markers focuses on the veins. It is divided into five parts: "initialize settings", "obtain gradient in color image", "enhance veins", "segment veins", "incorporate main veins" and "incorporate fragmented and fine veins". The details will be explained below.

3.5. Detect foreground markers

3.5.1. Initialize settings

Step 12: Set the near distance criterion coefficient *nearDistancePara* (such as 0.35), set the near distance criterion *nearDistance* as the average value of the image length and width multiplied by *nearDistancePara* and round it off. Set the near boundary distance criterion coefficient *nearBoundaryPara* (such as 0.1), and set the near boundary row number criterion *nearBoundaryRowDistance* as the total row number of the image multiplied by *nearBoundaryPara* and round it off. Set the near boundary column number criterion *nearBoundaryColDistance* as the total column number of the image multiplied *nearBoundaryPara* and round it off. Set the very near boundary distance criterion coefficient *veryNearBoundaryPara* (such as 0.01), set the very near boundary row criterion *veryNearBoundaryRowDistance* as the total row number of the image multiplied by *veryNearBoundaryPara* and round it off. Set the very near boundary column number criterion *veryNearBoundaryColDistance* as the total column number of the image multiplied by *veryNearBoundaryPara* and round it off. Set the dodge boundary distance coefficient *dodgeBoundaryPara* (such as 0.2. Its value must be greater than that of *nearBoundaryPara*), and set the dodge boundary row number



(a) Original image (b) Optimal background markers (c) Foreground markers (d) Final segmentation result

(e) *H* component image (f) *S* component image (g) *a* component image (h) *b* component image

(i) *H* component image segmentation result (j) *S* component image segmentation result (k) *a* component image segmentation result (l) *b* component image segmentation result

(m) Background detected from *H* component image (n) Background detected from *S* component image (o) Background detected from *a* component image (p) Background detected from *b* component image

Fig. 7. *Polygonum chinense L.*

dodgeBoundaryRowDistance as the total row number of the image multiplied by *dodgeBoundaryPara* and round it off. Set the dodge boundary column number *dodgeBoundaryColDistance* as the total column number of the image multiplied by *dodgeBoundaryPara* and round it off.

Step 13: Initialize the foreground marker image - *foregroundMask*. The specific process is as follows: create a binary image *foregroundMask* whose size is consistent with *currentImage* and whose pixel values are all "0". Reset the pixel values in the rectangle of the middle part of *foregroundMask* as “1”. The length of the rectangle is *para* × 2 times of that of *foregroundMask* and its width shares the same ratio. The four frames of the rectangle are paralleled to those of *foregroundMask*, and their central points are overlapped.

Step 14: Initialize the background marker image - *backgroundMask*. The specific process is as follows: create a binary image *backgroundMask* whose size is consistent with *currentImage* and whose pixel values are all “0”. Reset the pixel values on the four frames of *backgroundMask* as “1”.

3.5.2. Obtain gradient in color image

Step 15: Directly evaluate the gradient of color RGB image (*currentImage*) to obtain a gradient amplitude image (*VG*) and a gradient angle image (*A*). The specific process is as follows (Gonzalez et al., 2009) (Di Zenzo, 1986):

Step 15.1: Evaluate the partial derivatives at x and y directions. Suppose the coordinate of any point on the image *currentImage* is (x, y) and the pixel value is (R, G, B) in which R, G and B represent the component values in red, green and blue.

$$\vec{u} = \left(\frac{\partial R}{\partial x}, \frac{\partial G}{\partial x}, \frac{\partial B}{\partial x}\right) \tag{3}$$

$$\vec{v} = \left(\frac{\partial R}{\partial y}, \frac{\partial G}{\partial y}, \frac{\partial B}{\partial y}\right) \tag{4}$$

The sobel operator can be used to calculate six partial derivatives such as $\frac{\partial R}{\partial x}$.

Step 15.2: Solve

$$g_{xx} = \vec{u}\cdot\vec{u} \tag{5}$$

$$g_{yy} = \vec{v}\cdot\vec{v} \tag{6}$$

$$g_{xy} = \vec{u}\cdot\vec{v} \tag{7}$$

Step 15.3: Solve

$$\theta_1(x, y) = \frac{1}{2}\arctan\left[\frac{2g_{xy}}{(g_{xx} - g_{yy})}\right] \tag{8}$$

$$\theta_2(x, y) = \theta_1(x, y) + \frac{\pi}{2} \tag{9}$$

Here, “arctan” indicates arctan function.

Step 15.4: Solve

$$F_{\theta_1}(\mathrm{x, y}) = \sqrt{\frac{1}{2}[(g_{xx} + g_{yy}) + (g_{xx} - g_{yy})cos2\theta_1 + 2g_{xy}sin2\theta_1]} \tag{10}$$

$$F_{\theta_2}(\mathrm{x, y}) = \sqrt{\frac{1}{2}[(g_{xx} + g_{yy}) + (g_{xx} - g_{yy})cos2\theta_2 + 2g_{xy}sin2\theta_2]} \tag{11}$$

Step 15.5: If $F_{\theta_1}(x, y)$ is greater than or equal to $F_{\theta_2}(x, y)$, $F_{\theta_1}(x, y)$ is taken as the gradient amplitude $F(x, y)$, and θ_1 is taken as the gradient angle $\theta(x, y)$. Otherwise, $F_{\theta_2}(x, y)$ is taken as the gradient amplitude $F(x, y)$, and θ_2 is taken as the gradient angle$\theta(x, y)$. They are saved to obtain the gradient magnitude image *VG* and the gradient angle image *A* respectively.

For example, Fig. 8(a) is a color image of *Polygonum chinense L.* leaf, the gradient is directly solved from it, and the gradient magnitude image and the gradient angle image are obtained, as shown in Fig. 8(b) and Fig. 8(c). It can be seen from Fig. 8(b) that although the leaf veins are somewhat dim, they are relatively clear and the brightness of the background area outside the leaf is relatively low.

3.5.3. Enhance veins

Step 16: If *bestBackgroundDetectFlat* is true, the value of any point in *bestBackgroundDetect* will be detected. If it is “1”, reset the point value of the same position in *VG* as “0”. (The background area should not have veins of the target leaf.)

This step eliminates the possibility that some pixels in the background area will remix into the foreground after enhancement. Fig. 8(d) is the result that the pixel values of the background area of the gradient magnitude image are set as “0”. Some of the points that are really in the background are no longer visible. However, some are still there, because *bestBackgroundDetect* (Fig. 7(b), Note: Its original image is the same as Fig. 8) is not entirely correct.

Step 17: Statistic is conducted for the pixel values of *VG*. A threshold *high* is calculated, and 1% pixel values are greater than it. A threshold *low* is calculated, and 1% of pixel values are smaller than it. Reset all points with pixel values greater than *high* as *high*; reset all points with pixel values less than *low* as *low*.

This step removes the points whose pixel values in the gradient amplitude image are relatively extreme, which is helpful for the subsequent OTSU segmentation. The result is shown in Fig. 8(e), and the image is softer.

Step 18: In *A*, the disk with a radius of small scale (such as 3) is used as the neighborhood range for each point, and the standard deviation is solved. The standard deviation image *stdOfA* is obtained. The smaller the standard deviation in the local range, the higher the consistency of the angles of the gradient vectors for the point and those nearby it. As shown in Fig. 8(f), the dim line in the middle corresponds to the main vein.

Step 19: Reset $\alpha = \alpha/(PstdOfA + 0.1)$ for the pixel value α of each point in *VG* in order to enhance the veins in *VG*. Here, *PstdOfA* is the point value in *stdOfA* with the same position as α in *VG*. Fig. 8(g) is the result after vein enhancement. Compared with Fig. 8(e), the veins are bright indeed. However, it has also been seen that the leaf boundaries have also been enhanced. Some points outside the leaf are also enhanced.

3.5.4. Segment veins

Step 20: In the middle of *VG*, cut a subgraph *VGCrop* whose length and width are *para* × 2 times of *VG*. The four frames of the subgraph are paralleled to those of *VG*, and their central points are overlapped, as shown in Fig. 8(h). The purpose of determining the threshold by sampling in central area is to expect that the threshold is not disturbed by pixel values outside the leaf area.

Step 21: Apply the maximum inter-class variance method (OTSU) to get the threshold of *VGCrop* - *level*.

Step 22: Take *level* as a threshold to segment *VG* and obtain *OtsuBW*. The purpose of this operation is to get the veins, as shown in Fig. 8(i).

Step 23: Delete too small regions in *OtsuBW* (for example, delete the regions whose area is 10 or below), as shown in Fig. 8(j).

Step 24: Dilate *OtsuBW* in order to connect the broken veins, as shown in Fig. 8(k).

3.5.5. Incorporate main veins

Step 25: Copy and backup *foregroundMask* as *foregroundMaskBackup*.

Step 26: Conduct logical operation OR for *foregroundMask* and *OtsuBW*. And save the result as *foregroundMask*.

Step 27: In *foregroundMask*, select and reserve the area that is connected to the central point of the image and delete other areas. The result is shown in Fig. 8(l).

Step 28: Copy and backup *foregroundMask* as *foregroundMaskForDel*.

Step 29: Detect whether there is a point with a value of “1” in *foregroundMask* falling in the foremost *nearBoundaryRowDistance* rows, the final *nearBoundaryRowDistance* rows, the foremost *nearBoundaryColDistance* columns or the final *nearBoundaryColDistance* columns. If not, it is considered that the newly incorporated foreground area is far from the image frame, it is likely just the main veins which pass through the central area of the image, and the purpose is reached. Go to Step 36. In Fig. 8(l), foreground near the boundary is unavailable. Therefore, in this example, go to Step 36 directly. The image is also the result after incorporation of main veins.

Step 30: Erode *foregroundMask* with a small scale (such as applying a

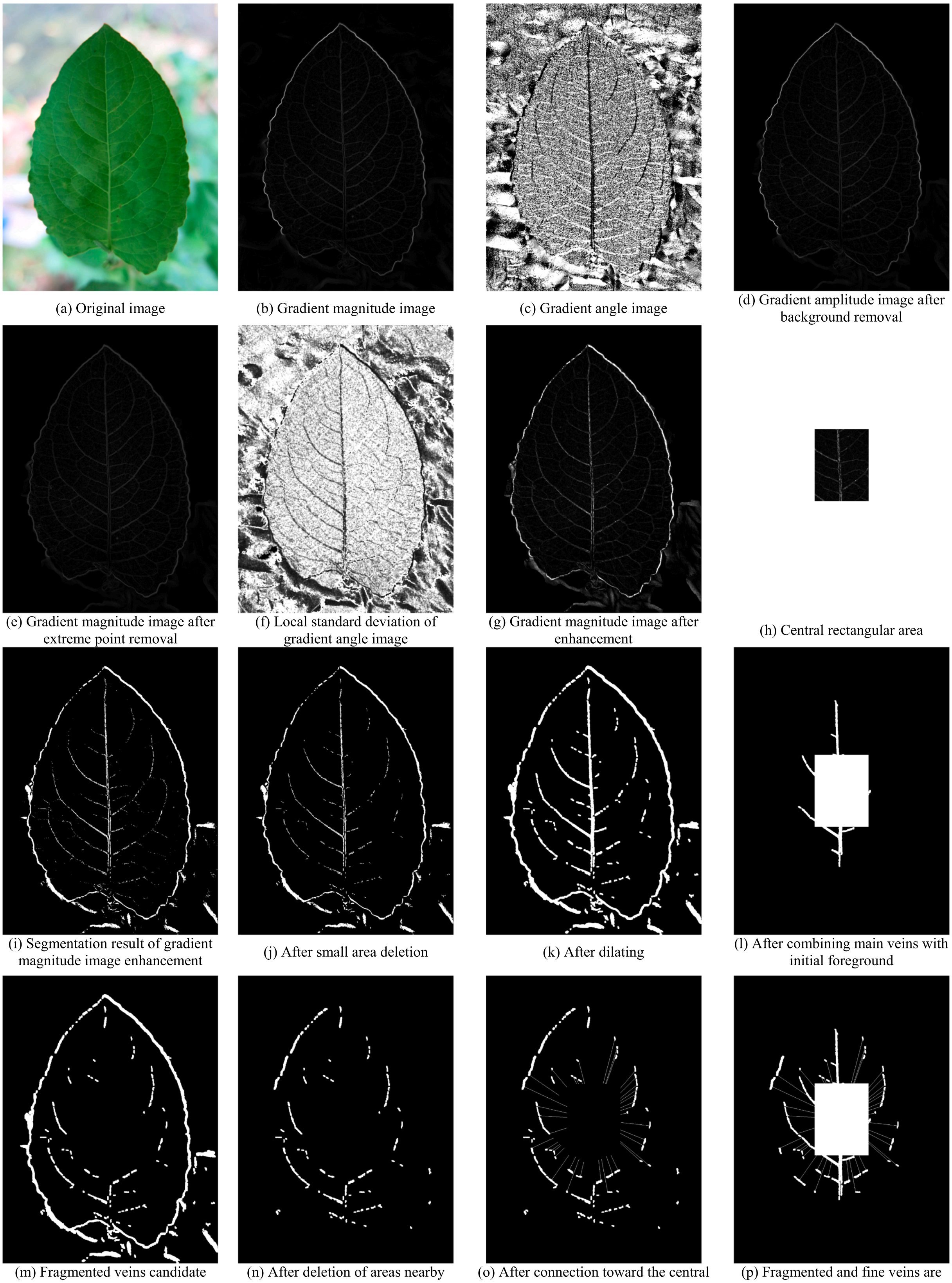
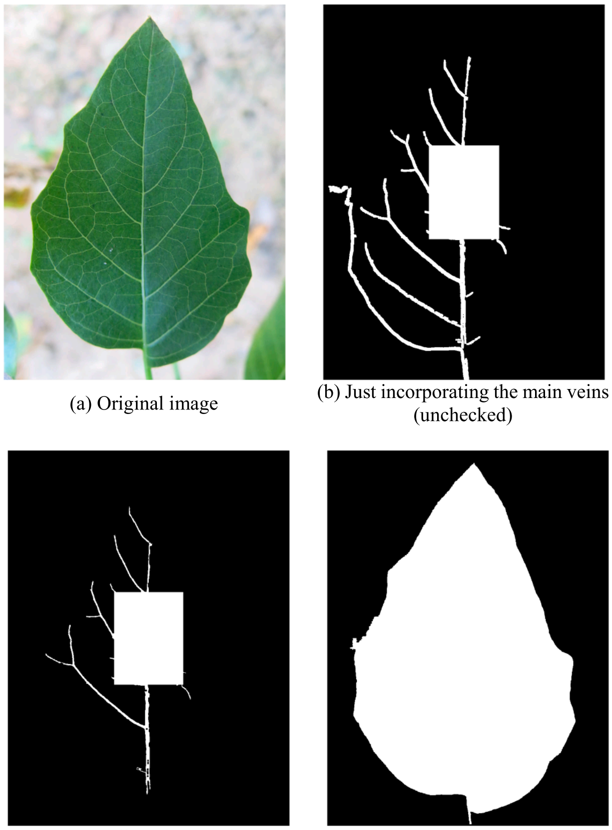


Fig. 8. *Polygonum chinense L.* (the same original image as Fig. 7).

disk with a radius of 3 as a structural element), and then select and reserve the area that is connected to the central point of the image and delete other areas. The function of this step is as follows: the main vein is too long and extends to the scope near the boundary of the image. It is necessary to beware that it incorrectly connects some areas outside the leaf. Therefore, it is necessary to cut it.

For example, the original image is the leaf image of *Datura metel L.* in Fig. 9(a), the foreground marker image after the main veins are



(a) Original image

(b) Just incorporating the main veins (unchecked)

(c) After small cutting for the first time

(d)Final segmentation result

Fig. 9. *Datura metel L.*

incorporated is shown in Fig. 9(b). The main veins extend to the vicinity of the image boundary (actually, the four frames of the image have been reached). Obviously, cutting is required. In this example, go to Step 30 after Step 29 is executed. After cutting, the result in Fig. 9(c) is obtained, and the main veins that were originally incorporated are basically retained. Later, the final segmentation result in Fig. 9(d) is obtained.

Step 31: Further detect whether there is a point with a value of “1” in *foregroundMask* falling in the foremost *veryNearBoundaryRowDistance* rows, the final *veryNearBoundaryRowDistance* rows, the foremost *veryNearBoundaryColDistance* columns or the final *veryNearBoundaryColDistance* columns. If any, it will indicate that the foreground area of the image is very close to the image frame, which means that the previous erosion does not achieve the purpose of trimming and needs to be eroded again. If not, go to Step 35.

Step 32: Erode *foregroundMask* with a small scale (such as applying a disk with a radius of 3 as a structural element), and then select and reserve the area that is connected to the central point of the image and delete other areas.

Step 33: Further detect whether there is a point with a value of “1” in *foregroundMask* falling in the foremost *veryNearBoundaryRowDistance* rows, the final *veryNearBoundaryRowDistance* rows, the foremost *veryNearBoundaryColDistance* columns or the final *veryNearBoundaryColDistance* columns. If any, it will indicate that the foreground area of the image is still very close to the image frame, which means that the previous erosion still fails to achieve the purpose of trimming and needs to be processed specially. If not, go to Step 35.

Step 34: Reset the foremost *dodgeBoundaryRowDistance* rows, the final *dodgeBoundaryRowDistance* rows, the foremost *dodgeBoundaryColDistance* columns, and the final *dodgeBoundaryColDistance* columns in *foregroundMask* as “0”.

Step 35: Conduct logical operation OR for *foregroundMask* and *foregroundMaskBackup*. And save the result as *foregroundMask*. It can be seen by referring to the above steps that *foregroundMask* contains a rectangular area in the center of the image, but there is a loss after erosion. It is replenished here. At this moment, the main veins are incorporated.

3.5.6. Incorporate fragmented and fine veins

Step 36: In *OtsuBW*, delete the foreground area in *foregroundMaskForDel* and save it as the fragmented vein candidate image - *candidates*. The deletion is as follows: For any point in *OtsuBW*, if the point value of the same position in *foregroundMaskForDel* is “1”, the point in *OtsuBW* is reset as “0”, as shown in Fig. 8(m).

Step 37: Scan the whole *candidates* to find out the closest distance *DN* from each area to the center of the image. If *DN* is smaller than *nearDistance*, mark it and record the row number and column number of the point in the area closest to the central point of the image. Judge whether the horizontal distance of each area from the frame of the image is smaller than *nearBoundaryRowDistance* and whether the vertical distance is smaller than *nearBoundaryColDistance*. If so, mark that the area is too close to the frame.

Step 38: Copy and backup *candidates* as *avoidRegions*.

Step 39: Delete the areas in *candidates* that are marked to be too close to the frame, as shown in Fig. 8(n).

Step 40: For each area in *candidates* whose *DN* is less than *nearDistance*, draw a line segment starting from the point of the area which is nearest to the central point of the image and pointing at such central point in an avoiding manner (the row number and the column number of the point are solved in Step 37.). The avoidance rule is as follows: suppose the coordinate of any point on the line segment to be (x, y). If *avoidRegions*(x, y) is “1” and the point is not the starting point of the line segment, the line segment will be canceled. In the process of drawing the line, if *foregroundMask*(x, y) is detected as “1”, it is considered that *foregroundMask* has been connected, the task is completed, and the line segment does not need to be further drawn so as to save time. The obtained *candidates* are shown in Fig. 8(o).

Step 41: Conduct logical operation OR for *foregroundMask* and *candidates*. And save the result as *foregroundMask*.

Step 42: In *foregroundMask*, select and reserve the area that is connected to the central point of the image and save it as *foregroundMask*. The result is shown in Fig. 8(p).

3.6. Markers revision and final segmentation

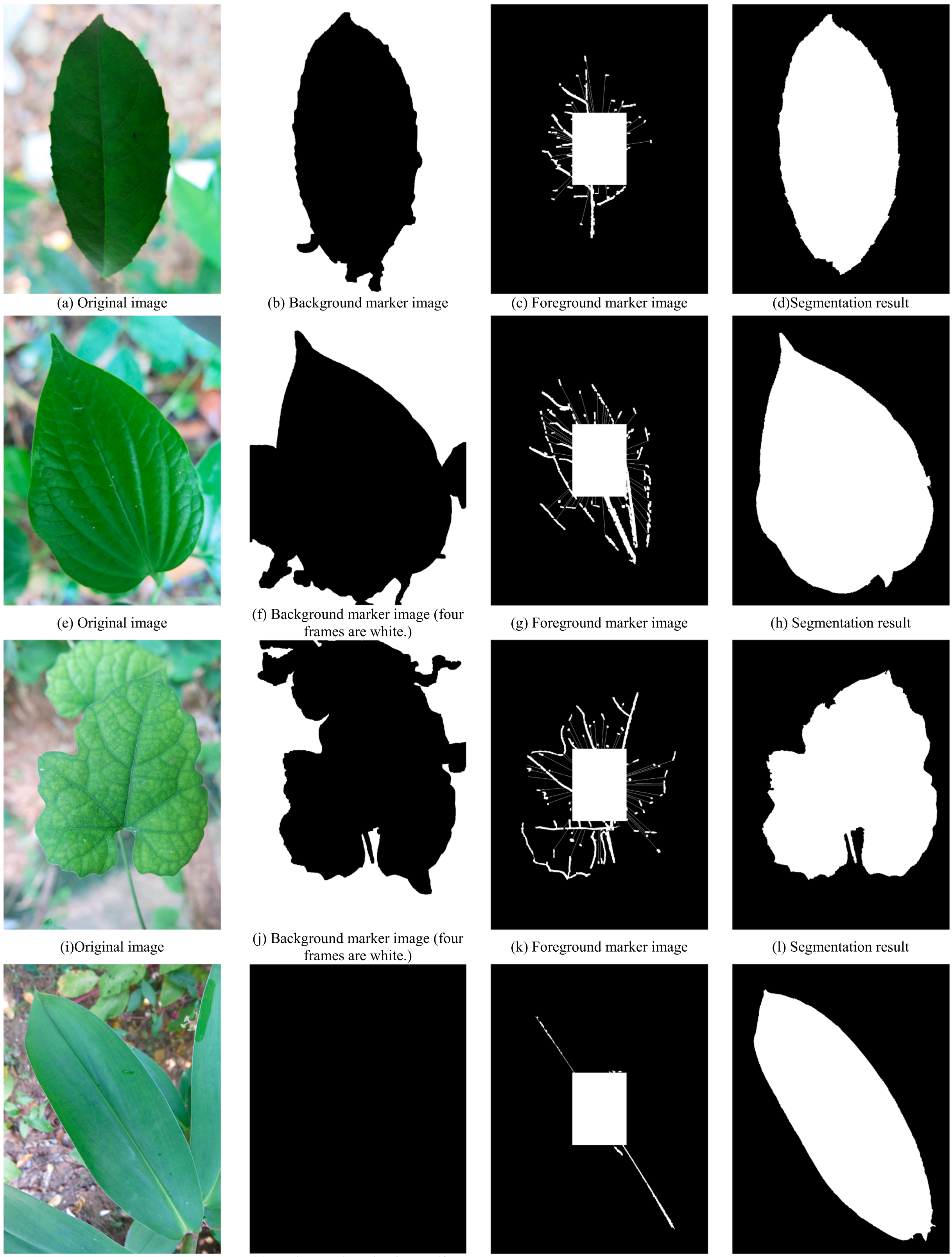
Step 43: If *bestBackgroundDetectFlat* is true, dilate *bestBackgroundDetect* slightly (using the disk operator with a radius of 3) as *bestBackgroundDetectFat*, and then delete the area overlapping with *bestBackgroundDetectFlat* in *foregroundMask*. For *foregroundMask*, reserve the area that is connected to the central point of the image and delete other areas. Finally, save *bestBackgroundDetect* as *backgroundMask*.

Step 44: Conduct marker-controlled watershed segmentation based on the result of logical operation OR for *backgroundMask* and *foregroundMask* to obtain *outPutImage*.

The example for *backgroundMask* is shown in Fig. 7(b), and the example for *foregroundMask* is shown in Fig. 7(c).

Step 45: Take the areas whose pixel values are “2” in *outPutImage* as the foreground, and the remaining as a background to obtain the final binary result *logicalImage*, as shown in Fig. 7(d).

At this moment, the algorithm is over. For more examples, please refer to Fig. 10. The figure provides four leaf images in complex background, foreground marker images (foregroundMask in Step 44), background marker images (backgroundMask in Step 44) and segmentation results respectively. For a quantitative and comparative analysis, please refer to Section 4.



(a) Original image (b) Background marker image (c) Foreground marker image (d)Segmentation result

(e) Original image (f) Background marker image (four frames are white.) (g) Foreground marker image (h) Segmentation result

(i)Original image (j) Background marker image (four frames are white.) (k) Foreground marker image (l) Segmentation result

(m)Original image (n) Background marker image (four frames are white.) (o) Foreground marker image (p) Segmentation result

Fig. 10. Row 1 to Row 4: *Cudrania tricuspidata (Carr.) Bur.ex Lavallee*, *Piper sarmentosum Roxb.ex.Hunter*, *Trichosanthes kirilowii Maxim*, *Costus speciosus (Koen.) Smith*.

4. Results and discussion

4.1. Observation criteria

In order to evaluate the segmentation results, we adopt five well-known observation criteria. In particular, *TP* indicates the number of positive samples (points inside leaf area) being classified as positive samples. *TN* indicates the number of negative samples (background points) being classified as negative samples. *FP* indicates the number of negative samples being classified as positive samples; and *FN* indicates the number of positive samples being classified as negative samples.

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

$$Dice = 2.0 \times \frac{Precision \times Recall}{Precision + Recall} \tag{14}$$

$$Manhattan = \frac{TP + TN}{TP + FP + TN + FN} \tag{15}$$

$$Jaccard = \frac{TP}{TP + FP + FN} \tag{16}$$

Note: *Dice* is equivalent to F1-measure.

4.2. Results based on Database 1

4.2.1. Comparison of multiple automatic algorithms

In order to directly compare with the existing main algorithms, we firstly adopt some traditional methods such as OTSU, MeanShift and GraphCut to segment 440 images in Database 1. Relevant results are recorded in Row 1–3 of Table 1.

Then, we further adopt the deep learning segmentation method FCN. Select imagenet-vgg-verydeep-16 (Simonyan and Zisserman, 2015) as the embedded deep network, use Database 0 introduced above as a training set, train FCN for 50 epochs, and then test Database 1, and finally obtain good results. Relevant results are recorded in Row 4 of Table 1.

Next, the proposed method is used to segment 440 images in Database 1. The segmentation results are recorded in Row 5 of Table 1 (highlighted in bold).

It is noted that as the run time of MeanShift is too long when processing high-resolution images (we tried to process an image with a resolution of 400 × 300, which took more than an hour), the image is compressed to 200 × 150 for processing. Since FCN is unable to accept high-resolution images for training (due to memory and time factors), the image is compressed to 400 × 300 for processing. Other algorithms are tested by default under an image resolution of 1200 × 900. Under this resolution, the image detail, such as the fluff or small thorns on the edge of the leaf, are preserved well.

It can be seen from the comparison of the results of Row 1–5 of Table 1, the proposed algorithm has obvious advantages over another three traditional methods. Compared with the deep learning segmentation method FCN, it is also better slightly.

4.2.2. Comparison of the results of the proposed algorithm under different resolutions

Many details of the leaf are preserved well under an image resolution of 1200 × 900. However, as the deep network is possibly used in the subsequent classification task, the resolution of 1200 × 900 may be too big.

Therefore, we have further tested the performance of the proposed algorithm under the resolutions of 1000 × 750, 800 × 600, 600 × 450 and 400 × 300. Specifically, the calculation parameters of *ratio* in Step 1 for the proposed algorithm is adjusted. Thus, the image in Database 1 with the original resolution of 4000 × 3000 is adjusted to the specified resolution. The test results are recorded in Row 6–9 of Table 1.

It can be seen from Row 5–9 of Table 1 that the performance of the proposed algorithm fluctuates little under such resolutions as 1200 × 900, 1000 × 750, 800 × 600, 600 × 450 and 400 × 300. It can all achieve higher accuracy than other algorithms.

4.3. Results based on Database 2

In order to make more comparisons with the existing fully automatic segmentation methods, the proposed method is used again to segment the widely accept Database 2 in the world. See Table 2 for the results. In particular, the test results of the proposed method and FCN are obtained by us. The results of other methods were given by Grand-Brochier et al. (2015). Taking the five indicators into consideration, the results of the proposed algorithm are better than the other methods listed in the table.

In particular, the sample quantity in Database 2 is too small, and it is difficult to find new samples from the same source to establish a training set. Therefore, an attempt is made to train FCN in three ways.

FCN (a) uses Database 0 as a training set to train 50 epochs and tests with a complete Database 2. In fact, FCN (a) in Table 2 (a) is FCN in Row 4 of Table 1. Their algorithms, initial deep network and training set are the same except the test set. If the test result of FCN in Row 4 of Table 1 is not bad, the same trained network is directly used to test Database 2, and the effect turns out to be terrible. In contrast, the proposed algorithm achieves good results no matter it is used in Database 1 or Database 2. It can be said that the proposed algorithm here reflects its better ability to adapt to new types of samples than FCN.

FCN (b) uses Database 2(a) as a training set to train 50 epochs and tests with Database 2(b) (as explained above, they are from the division of Database 2). It can be seen from Table 2 that the indicators of FCN (b) have been greatly improved when compared with FCN (a), and *Recall* has exceeded 90%. However, its *Precision* is still bad.

As the results of the experiments are not satisfactory, we have further tried the transfer learning. In FCN (c), networks are trained 50 epochs with Database 0, and then transfer to be trained with Database 2 (a). Finally, they are tested with Database 2 (b). The test results are further improved when compared with FCN (b). However, its *Precision* is still low.

Table 1
Average result based on Database 1.

Row No.	Method	Image resolution	*Precision*	*Recall*	*Dice*	*Manhattan*	*Jaccard*	*Time* (second)
1	OTSU (Otsu, 1979)	1200 × 900	70.99%	86.71%	0.773	82.07%	68.94%	0.034
2	MeanShift (Comaniciu and Meer, 2002)	200 × 150	90.90%	86.64%	0.851	90.28%	78.95%	678.455
3	GraphCut (Boykov and Jolly, 2001)	1200 × 900	80.82%	85.61%	0.826	85.58%	72.69%	3.164
4	FCN (Shelhamer et al., 2017)	400 × 300	91.20%	98.29%	0.945	95.57%	89.85%	0.748
5	**Proposed algorithm**	**1200 × 900 (default)**	**96.02%**	**95.39%**	**0.954**	**96.51%**	**91.68%**	**0.949**
6	Proposed algorithm	1000 × 750	96.00%	95.44%	0.955	96.50%	91.74%	0.661
7	Proposed algorithm	800 × 600	96.10%	95.61%	0.956	96.64%	91.96%	0.469
8	Proposed algorithm	600 × 450	95.81%	95.96%	0.956	96.65%	92.04%	0.334
9	Proposed algorithm	400 × 300	95.34%	96.14%	0.955	96.52%	91.82%	0.178

Table 2
Direct Comparison with Existing Methods of Segmentation Based on Database 2.

Method	*Precision*	*Recall*	*Dice*	*Manhattan*	*Jaccard*
OTSU (Otsu, 1979)	83.29%	90.02%	0.855	89.51%	76.54%
MeanShift (Comaniciu and Meer, 2002)	78.19%	89.14%	0.816	86.35%	71.56%
Pyr. MeanShift (Li et al., 2010)	81.73%	90.76%	0.846	88.33%	75.29%
Graphcut (Boykov and Jolly, 2001)	75.89%	87.20%	0.79	85.46%	68.34%
Watershed (Beucher and Meyer, 1993)	76.38%	93.67%	0.82	84.93%	72.38%
Felzenszwalb (Felzenszwalb and Huttenlocher, 2004)	86.03%	79.88%	0.793	86.70%	69.31%
Kurtz (Kurtz et al., 2012)	85.07%	87.52%	0.851	89.94%	76.74%
Power Watershed (Couprie et al., 2009)	81.15%	75.79%	0.814	85.08%	64.83%
GAC (With No Input Stroke) (Grand-Brochier et al., 2015)	94.56%	87.47%	0.903	94.06%	83.39%
FCN (a) (Shelhamer et al., 2017)	67.47%	30.54%	0.473	69.39%	23.45%
FCN (b) (Shelhamer et al., 2017)	74.78%	91.02%	0.804	85.18%	69.30%
FCN (c) (Shelhamer et al., 2017)	79.31%	92.65%	0.842	88.01%	74.22%
Proposed Algorithm	92.35%	94.59%	0.929	94.81%	87.54%

Table 3
Run Time of Algorithms Based on Database 2.

Method	Configurations	Time (second)
OTSU	Intel i7-3610QM CPU (2.3 GHz), 8 GB RAM	0.09
MeanShift	Intel i7-3610QM CPU (2.3 GHz), 8 GB RAM	0.6
Power Watershed	Intel i7-3610QM CPU (2.3 GHz), 8 GB RAM	8
GAC	Intel i7-3610QM CPU (2.3 GHz), 8 GB RAM	3.5
FCN (a)	Intel i7-4790 CPU (3.6 GHz), 8 GB RAM	1.582
FCN (b)	Intel i7-4790 CPU (3.6 GHz), 8 GB RAM	1.505
FCN (c)	Intel i7-4790 CPU (3.6 GHz), 8 GB RAM	1.448
Proposed Algorithm	Intel i7-4790 CPU (3.6 GHz), 8 GB RAM	0.389

From the data of Table 1 and Table 2, we also found another interesting phenomenon. FCN has shown extremely powerful performance in Database 1, and it only lags behind slightly the proposed algorithm. However, it does not perform well in Database 2 and its subsets, and it even lags behind some traditional methods. Combining this phenomenon with our superficial understanding on deep network, we can reasonably think that FCN is data-dependent and its performance fluctuates greatly due to different sample sets. Secondly, we can boldly conjecture that FCN is not suitable for processing low-quality sample images, for some of the images in Database 2 have dim and blurred light.

4.4. Run time test

4.4.1. Comparisons based on Database 1

In the last column of TABLE 1, the average time consumption for each algorithm is given in second. In terms of resolution, OTSU takes the shortest time. The proposed algorithm is second to it. MeanShift is too slow.

Under the resolution of 400×300, the time consumption of FCN is not long. However, the proposed algorithm takes shorter time under the same resolution of 400×300. Under the resolutions of 600×450, 800×600 or 1000×750, the proposed algorithm still takes the shorter time.

4.4.2. Comparisons based on Database 2

In TABLE 3, the data of algorithms except the proposed algorithm and FCN is sourced from (Grand-Brochier et al., 2015). It can be seen from the run time test based on the same image database that the performance of the proposed algorithm is acceptable in terms of hardware conditions and run time.

5. Conclusions

In this paper, a fully automatic segmentation method for medicinal plant leaf images in complex background is proposed. Firstly, the gradient is directly obtained from the color image. A more accurate foreground marker image is obtained based on such a feature that the veins are contained in the leaf area and by effectively enhancing and extracting the veins. Finally, the marker-controlled watershed method is used to realize image segmentation.

A series of experimental tests based on a self-built database and another widely used database show that the proposed method is better than many main fully automatic image segmentation methods including deep learning FCN. Its running speed is also fast and slightly lower than that of the OTSU threshold segmentation.

It can be said that the proposed method has basically overcomes the problem of fully automatic segmentation of leaf images in complex background and promoted the identification development of the leaf image in complex background.

Although the proposed method is designed for the leaf, it has certain reference significance for the fully automatic image segmentation with a single target in complex background and the veins in the target area.

References

Aimen, Aakif, Faisal, Khan Muhammad, 2015. Automatic classification of plants based on their leaves. Biosyst. Eng. 139, 66–75.
Beucher, S., Meyer, F., 1993. The morphological approach to segmentation: the watershed transformation. Opt. Eng. 34, 433–481 Chapter 12.
Boykov, Y.Y., Jolly, M.P., 2001. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In: Proc. ICCV, 2001, Vancouver, Canada, pp. 105–112.
Cem, Kalyoncu, Önsen, Toygar, 2015. Geometric leaf classification. Comput. Vis. Image Underst. 133, 102–109.
Comaniciu, D., Meer, P., 2002. Mean shift: a robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. 24 (5), 603–619.
Couprie, C., Najman, L., Grady, L., Talbot, H., 2009. Power watersheds: a new image segmentation framework extending graph cuts, random walker and optimal spanning forest. In: Proc. ICCV, Kyoto, Japan, pp. 731–738.
Di Zenzo, S., 1986. A note on the gradient of a multi-image. Comput. Vis. Graph. Image Process. 33 (1), 116–125.
Felzenszwalb, P.F., Huttenlocher, D.P., 2004. Efficient graph-based image segmentation. Int. J. Comput. Vision 59 (2), 167–181.
Gao, Liwen, Lin, Xiaohua, 2018. A method for accurately segmenting images of medicinal plant leaves with complex backgrounds. Comput. Electron. Agric. 155, 426–455.
Goëau, H., Bonnet, P., Joly, A., Boujemaa, N., Barthelemy, S., Molino, J.M.F., Birnbaum, P., Mouysset, E., Picard, M., 2011. The ImageCLEF 2011 plant images classification task. Image CLEF2011 Working Notes.

Gonzalez, Rafael C., Woods, Richard E., Eddins, Steven L., 2009. Color image processing. In: Digital Image Processing Using MATLAB. PHEI, Beijing, China, pp. 171–178.

Grand-Brochier, Manuel, Vacavant, Antoine, Cerutti, Guillaume, Kurtz, Camille, Weber, Jonathan, Tougne, Laure, 2015. Tree leaves extraction in natural images: comparative study of preprocessing tools and segmentation methods. IEEE Trans. Image Process. 24 (5), 1549–1560.

Kurtz, C., Passat, N., Gançarski, P., Puissant, A., 2012. Extraction of complex patterns from multiresolution remote sensing images: a hierarchical top-down methodology. Pattern Recogn. 45 (2), 685–706.

Larese, M.G., Baya, A.E., Craviotto, R.M., Arango, M.R., Gallo, C., Granitto, P.M., 2014. Multiscale recognition of legume varieties based on leaf venation images. Expert Syst. Appl. 41 (10), 4638–4647.

Larese, M.G., Namias, R., Craviotto, R.M., Arango, M.R., Gallo, C., Granitto, P.M., 2014. Automatic classification of legumes using leaf vein image features. Pattern Recogn. 47 (1), 158–168.

Li, S.X., Chang, H.X., Zhu, C.F., 2010. Adaptive pyramid mean shift for global real-time visual tracking. Image Vis. Comput. 28 (3), 424–437.

Naresh, Y.G., Nagendraswamy, H.S., 2016. Classification of medicinal plants: an approach using modified LBP with symbolic representation. Neurocomputing 173, 1789–1797.

Otsu, N., 1979. A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybernet. 9 (1), 62–66.

Roerdink, Jos B.T.M., Meijster, et al., 2000. The watershed transform: definitions, algorithms and parallelization strategies. Fundamenta Inform. 41 (1,2), 187–228.

Shelhamer, Evan, Long, Jonathan, Darrell, Trevor, 2017. Fully convolutional networks for semantic segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 39 (4), 640–651.

Simonyan, Karen, Zisserman, Andrew, 2015. Very deep convolutional networks for large-scale image recognition. The Third International Conference on Learning Representations.

Yang, Shilin, Zhang, Zhao, Zhang, Bengang, Yulin, Lin, 2000. Counter measures to overcome the present inadequate means for the protection of near extinct species of Chinese medicinal plants. Chin. Tradit. Herbal Drugs 31 (6), 401–403.

Liwen Gao received the B.S. degrees in computer science and technology from South China University of Technology, China, in 2003, the M.S. degree in computer system architecture from South China University of Technology, China, in 2006, and the Ph.D. degree in intelligent detection and instrument for manufacture engineering, from South China University of Technology, China, in 2012. His research interests are in image segmentation and image recognition.

Xiaohua Lin received the B.S. degrees in pharmaceutical engineering from Guangzhou University of Chinese Medicine, China, in 2003, the M.S. degree and the Ph.D. degree in Chinese pharmacy from Guangzhou University of Chinese Medicine, China, in 2006 and 2011, respectively. Her research interest is in medicinal plant identification.