Segmentation of single and overlapping leaves by extracting appropriate contours

Conference Paper · November 2019

DOI: 10.5121/csit.2019.91323

2 authors:

Rafflesia Khan
University of Limerick
13 PUBLICATIONS **18** CITATIONS
SEE PROFILE

Rameswar Debnath
Khulna University
41 PUBLICATIONS **360** CITATIONS
SEE PROFILE

SEGMENTATION OF SINGLE AND OVERLAPPING LEAVES BY EXTRACTING APPROPRIATE CONTOURS

Rafflesia Khan and Rameswar Debnath

Computer Science and Engineering Discipline, Khulna University,
Khulna, Bangladesh.

ABSTRACT

Leaf detection and segmentation is a complex image segmentation problem as leaves are most often found in groups with natural background. Edges of leaves cannot be clearly defined from image because of their color similarities.Also,separating every single as well as overlapping leaf individually is even more challenging as leaves share almost same color, texture and shape. In this paper, we propose a new automatic approach for leaf segmentation from image. Our leaf segmentation process uses efficient techniques for processing an image to obtain contours of every individual objects. Then, it selects the best appropriate connected contours that represent region of every leaves appearing in an image. Our model archives an overall 90.46% segmentation rate where segmentation rates for single and overlapping leaves are 95.34% and 86.73%, respectively.

KEYWORDS

image processing, leaf object segmentation, overlapping leaves, connected contour, object boundary detection.

1. INTRODUCTION

Plants are one of the most essential parts of nature and human lives. As almost every plant is identified by its leaf, proper plant-leaf identification is essential for agricultural productivity as well as industries such as drug, chemical, cosmetics, etc. Leaf identification is also used in crop disease identification and identification of rear as well as endangered plants. With the help of image processing and object detection based automatic models now a days we do not need experienced botanists and hedge effort for leaf identification task. Before identifying a leaf from image, using an automatic model, finding its location and segmenting the leaf region from image are the initial tasks. Leaf segmentation includes two major procedure: (1) segmenting foreground leaf region from natural background and (2) segmenting each single leaf and each occluded or overlapping (i.e., object on object) leaf individually from image. Figure 1 shows these two procedure of leaf segmentation process on an example image. Detecting leaves from complex natural background and separating every occluded leaves of same color and texture make leaf segmentation problem challenging. So, now a days, leaf segmentation from image is an area of growing research interest with significant applications.

Natarajan Meghanathan et al. (Eds) : CSEIT, CMLA, NeTCOM, CIoT, SPM, NCS, WiMoNe, Graph-hoc - 2019
pp. 287-300, 2019. © CS & IT-CSCP 2019 DOI: 10.5121/csit.2019.91323

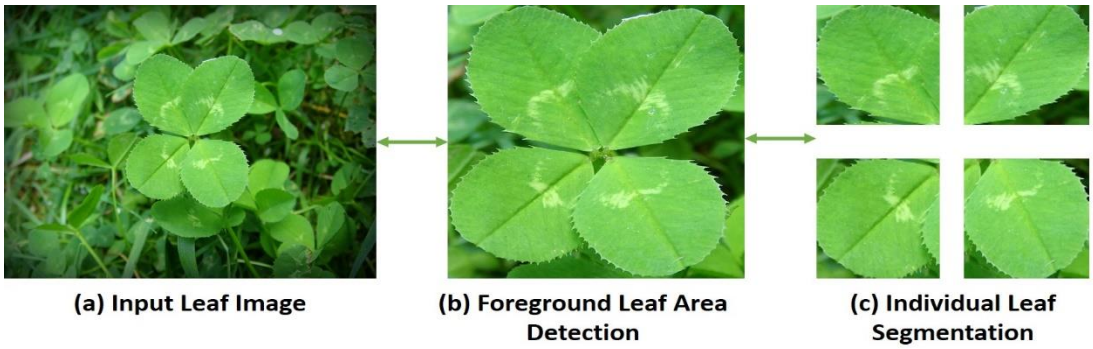

Figure 1. Major procedures of leaf segmentation.

Considering the significant importance and applications, numerous effective methods for leaf segmentation have been proposed [1] since the 1980s. The frequently used image object segmentation methods include edge-based, cluster-based, region-based and deep learning based methods. Niu et al. [2] proposed a model for cotton leaf segmentation using improved watershed algorithm. Dornbusch and Andrieu [3] developed a thresholding algorithm for estimating winter wheat's lamina boundaries. Combining global information with local statistical information Peng et al. [4] introduced a Chan Vese model for boundary detection of given leaf images. Although these models give good performance in case of segmenting a single leaf, accurate and non-destructive leaf segmentation is still a difficult task. These difficulties are caused by the uncertainties of overlapping condition and complex natural background of leaf surroundings. Considering the still existing complexities of leaf segmentation, number of segmentation techniques needed to be combined. Z. Wang et al. [5] presented an overlapping leaves image segmentation technique based on the Chan Vese model and Sobel operator. In [6], Cerutti et al. retrieved the leaf contour of image from a complex natural background by applying a two-step active contour algorithm using polygonal leaf model. Kenta Itakura and Fumiki Hosoi [7] proposed retrieval of plant structural parameters and methods for automatic and accurate leaf segmentation using 3D information point-cloud images. They combine distance transform and watershed algorithm. Chunlei, Wand et al. [8] used mean shift segmentation for segmenting foreground leaf region. Then they have implemented automatic initialization of active contour model (ACM) by calculating the center of divergence (CoD) and finally segmented occluded leaves individually using ACM. Daniel D. Morris [9] proposed a pyramid convolutional neural network with multi-scale predictions that finds and discriminates leaf boundaries from interior textures. Then using a watershed-based algorithm they estimate closed contour boundaries around individual leaves using previously detected boundaries. A comparative study of 14 unsupervised and 6 supervised segmentation models using Pl@ntLeaves dataset [10] was shown in [11].

Existing models combined with various segmentation techniques performs well in case of segmenting both single and overlapping leaf. But, there exists some still unsolved issues. Some models like [5], work with only one species of leaf which does not ensure the performance of model for leaves of different species found in different circumstance. Also, in real-life, randomly captured image can compromise the performance of some models [7, 8] that usually works with high resolution image captured with powerful camera. On the other hand, deep learning based models like, [9] do not work better while detecting leaves with internal texture and weak boundary clues. Total 20 models compared in [11], work for single leaf segmentation where all test images are captured focusing a target leaf at image center. But, in general leaves are most often found in groups where image foreground might contain multiple leaves overlapping one another.

Analyzing all these still existing complexities, in this paper, we propose a contour selection based leaf segmentation approach using various image processing techniques. In this model, our main

aim is to find the region of every individual leaves form image by detecting their appropriate contour. By contour, we indicate the outline that is marking the whole boundary of an object. Also, in this paper our considerable object is leaf. Most of the traditional contour detection algorithms of image either detect contour of leaf region from both foreground and background or detect contour without separating individuals. So, in our proposed model we apply some image processing techniques as pre-processing tasks before contour detection. These image processing techniques not only separate the foreground region from the background but also makes every individual leaf boundaries much easier to detect. So, from the processed image we can detect the contours that best represent all leaf regions individually. Finally, we segment those detected leaf regions (i.e., contours) as individual leaf images.

2. PROPOSED METHODOLOGY

Our proposed model works for segmenting every single as well as overlapping leaves separately. So, our goal is to find every contour (i.e., closed boundary) that precisely represents the outline of all visible regions of leaves in an image. Figure 2 shows the workflow diagram of our proposed model with an example image which visually shows the effect of every step on the input image. On the input image, at first we perform some preprocessing (i.e., section A in Figure 2). Leaf images can be of different types such as image with really complex background with multiple leaves or image with simple background with single leaf etc. Different types of image need different processing for better segmentation. So, we perform two different types of processing (i.e., section B and section C in Figure 2) on the result from A and generate two processed images. Next, we detect two sets of contours from both of those processed images and select one best contour set. Finally, we segment those best contours as individual leaf region from original input image (i.e., section D in Figure 2).

2.1. Preprocessing (Section A)

The preprocessing steps of our proposed model mainly works for preparing an image for leaf boundary detection. This is done by highlighting boundary edges and eliminating unnecessary internal and external edges of leaves.

2.1.1. Input Image

At first the input image is read in BGR (Blue, Green, Red) color format. As we use Python language and OpenCV library for the model implementation, the input image is read in BGR color format instead of RGB. The original version of the input image is stored as *Main image* and a copy of that image is used for further processing. Here, we store Main image because, after all processing finally our model segments or detects the actual contour of leaves appearing in input image from the original version of it. By this the system ensures noise free output images.

2.1.2. Resize

Processing big sized input image requires unnecessary power and computation time. It also sometimes hampers the segmentation accuracy. So, we select an optimal image size, 800 × 700 pixels, for input image. As the size ensures better performance for all our test experiments, after several number of test cases we have selected this size. When input image exceeds this optimal size, our model resizes it to the optimal size and then the resized image is stored as Main image. Otherwise, this step is ignored.

2.1.3. Preserve Edges

Every leaf has some internal textures and some species have even complex ones (e.g., African Blue). These textures create so many unnecessary edges which can manipulate target leaf area segmentation. In our model, for segmenting each leaf separately we wish to eliminate all unnecessary edges from image and preserve only the boundary edges.

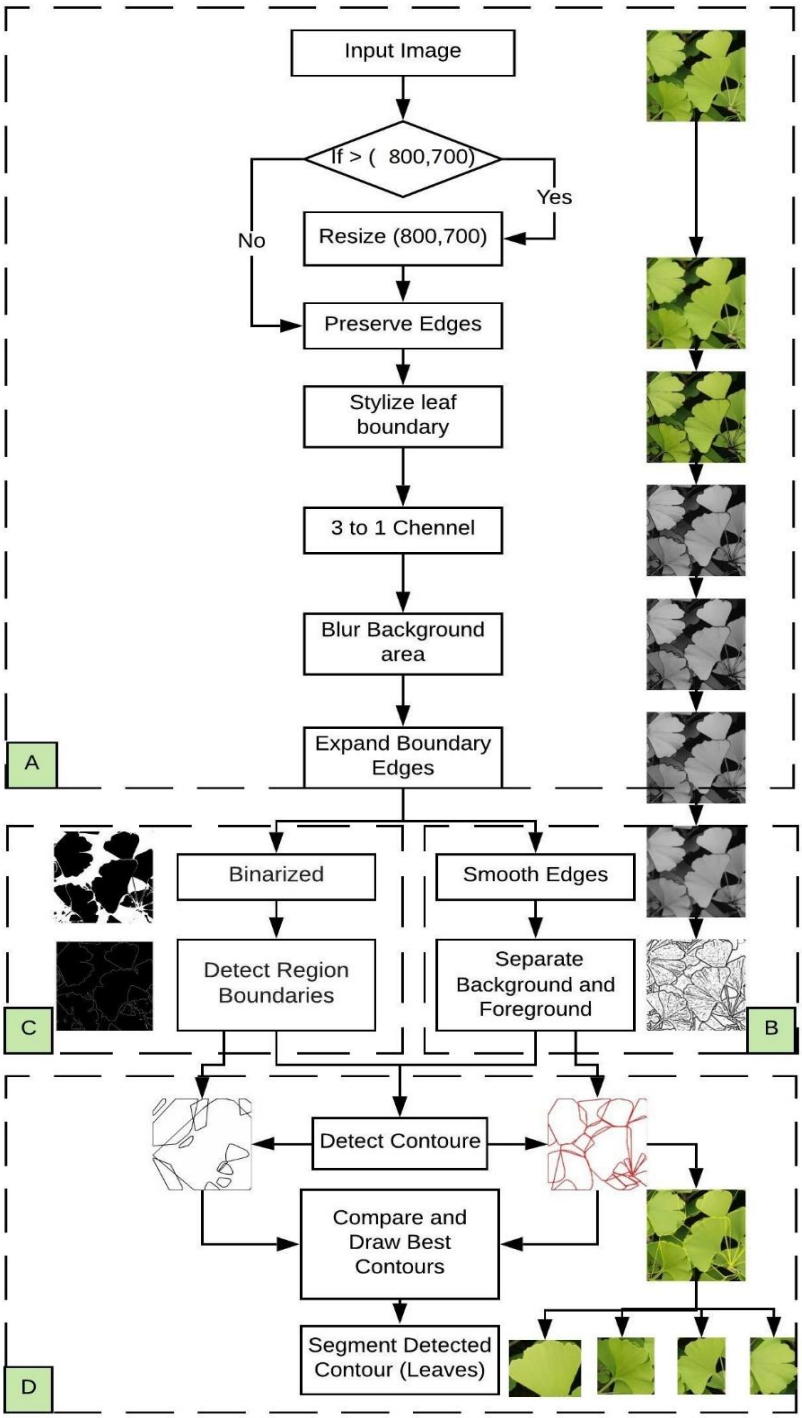

Figure 2. Workflow diagram of our proposed model with example image.

In this paper by unnecessary edges we refer all internal and external edges except the outline edges of every leaf object. Because, our aim is to find the contours or outlines of every leaf and

segment the connected regions within those. At this stage for smoothing internal texture and preserving boundary edges we apply Edge Preserving Filter [12] on image. As the required parameters we use 3rd edge preserving filter flag, value 40 for sigma_s (i.e., Sigma Spatial which controls the amount of neighborhood for smoothing), and value 0.3 for sigma_r (i.e., Sigma Range within the neighborhood which controls how dissimilar colors will be averaged).

2.1.4. Stylize Leaf Boundary

Our model draws thick boundary on the resulting image after smoothing internal texture and preserving boundary. We apply Stylization Filter to produce a watercolor effect on image which makes every object edge or outline or contour smooth and at the same time sharp [13]. It uses Normalized Convolution (NC) filter for providing better accuracy and works faster than other outline sharping filters [14]. Eventually it thickens every outline edges and does not get affected by internal texture. For this procedure, the value of sigma spatial is set to 60 and the value of sigma range is set to 0.07.

2.1.5. Multi-Channel to Single Channel

The next step is to convert the BGR format (3 channel) image into grayscale image (1 channel) using equation (1). In grayscale image we need to process only one-third of the image data compared to BGR image which significantly reduces the amount of computation and memory consumption.

$$G_{g(i,j)} = 0.114I_{B(i,j)} + 0.587I_{G(i,j)} + 0.299I_{R(i,j)} \qquad (1)$$

In equation (1), G represents the gray image and B, G, R represent the Blue, Green and Red channel respectively of image I, and i, j represents the coordinate value in x and y direction respectively.

2.1.6. Blur Background Area

On the stylized grayscale image, the next step is to eliminate the background. But, leaves are most often found in groups and our model works for segmenting multiple leaves individually. Our model does not use any direct background elimination algorithms as most of those target the center of image as foreground which might eliminate some of the foreground leaves which are not within the center of image. In our model we apply Gaussian filter for blurring image background. The Gaussian kernel is defined in 2D using equation (2). This non-linear filter enhances the effect of the foreground pixels and gradually reduces the effects of pixels which are farther from the center [14]. Thus, we get an image with much blurry background area and smoother foreground area.

$$G_{au-2D}(i, j; \sigma) = \frac{1}{2\pi\sigma^2} e^{\left(-\frac{i^2+j^2}{2\sigma^2}\right)} \qquad (2)$$

Here $G_{au\text{-}2D}(i, j)$ represents the Gaussian Kernel function where i, j represents the coordinate value in x and y direction respectively and σ satisfies the width of the Gaussian kernel. In our model we apply a 9×9 kernel with σ value 1.5 in both x and y direction.

2.1.7. Expand Boundary Edges

In our work, we have performed morphological dilation once using a 3×3 integer valued kernel. Basically, image dilation enlarges the boundaries of regions of foreground pixels which also fills the holes within those regions [15] and joins broken parts. In our model, this procedure grows or

expands the areas of bright regions which eventually enlarges the outline edges of leaves. This step of processing also contributes in separating overlapping edges. Also, a bigger or thicker edge separates two connected objects better than a thin one. The formula used in this dilation process is shown in equation (3).

$$D_a(i,j) = I(i,j) \oplus kernel\ (3 \times 3) \quad (3)$$

Here D_a (i, j) represents the output image and I (i, j) represents source image where i and j represents the coordinate value in x and y direction respectively.

2.2. Processing of Type One Image (Section B)

In real-life, randomly captured image do not have object at the center of image with blurred background. These kinds of image contains scattered foreground objects with complex background and more unnecessary edges. These edges make images complex to process and hamper contour detection. So, for these kinds of images we need to do some extra processing before detecting the contour of leaves and running segmentation.

2.2.1. Smooth Edges

In case of type one image, when dilation process enlarges the brighter edges it might enlarge some still existing unnecessary internal texture edges of leaf. To handle that situation after dilation we perform a smoothing operation. For this, we apply a 2D Convolution [16] filter where we use a 5×5 averaging filter kernel to convolve through the image and rapidly smoothen all existing intensity variations within image pixels.

2.2.2. Separate Background and Foreground

In our proposed model, we wish for preparing an input image perfectly ready for individual as well as accurate leaf contour detection. Through all the previous steps we eliminate internal texture edges, thicken outline edges and also smoothen image. At this step, we separate the foreground region's pixels with a single intensity from the pixels on background. For this reason instead of using just a threshold value we use Adaptive Thresholding [17]. It calculates a different threshold for different regions of the same image. Here we use Adaptive Thresh Gaussian as adaptive method which uses threshold value as the weighted sum of neighbourhood values where weights are Gaussian window [18]. Value 1 is set as the subtracting constant from the weighted mean in case of weighted sum calculation. To calculate a threshold value, 11 is used as the size of a pixel's neighbourhood. We also use inverse threshold as thresholding style and this thresholding helps our system to separate background and foreground and make the foreground objects much visible [13]. At the end of this step, we get a background eliminated binary threshold image which is ready for contour detection.

2.3. Processing of Type Two Image (Section C)

An image focusing objects at center with less complex background and less amount of leaves, is another type of image. Removing background and making it ready for contour detection is less complex than type one image. Within our experiments, we find that running these two kinds of images through a same procedure does not result in better segmentation performance. So, we perform both these two types of processing for every image, detect set of contours from each processed image and then select the best set of contour.

2.3.1. Binarize

In type two processing, at first we binarize the resultant dilated image found after pre-processing for foreground and background separation. For this binarization, we apply binary thresholding with Otsu's thresholding. As Otsu's thresholding automatically determines the threshold value that best describes the image, we apply it directly for background removing. But, it do not perform better in case of non-bimodal image. To handle that we use binary thresholding and Otsu's thresholding together where Otsu's method automatically determines the threshold value, pixels below the threshold is turned off and pixels above the threshold value is turned on.

2.3.2. Detect Region Boundary

For detecting the leaves as connected contours from image after background separation we look for region boundary or border extraction. Our model performs border extraction by subtracting dilation result from erosion result. Dilation and erosion mostly affect the pixels that exists between the foreground and background as well as the pixels that exists close to the boundary. The difference between the dilation and erosion generally yields all object's boundary which significantly helps the segmentation task and prepares the image for a perfect individual contour detection. This boundary detection is performed using equation (4), (5) and (6) sequentially.

$$D_a(i,j) = I(i,j) \oplus kernel \quad (4)$$

$$E_r(i,j) = D_a(i,j) \ominus kernel \quad (5)$$

$$B_o(i,j) = D_a(i,j) \backslash E_r(i,j) \quad (6)$$

Here $D_a(i,j)$, $E_r(i,j)$ and $B_o(i,j)$ represents the image after dilation, erosion and border detection respectively, and i, j represents the coordinate value in x and y direction respectively.

2.4. Contour Detection and Segmentation (Section D)

At this step we perform contour detection. We have mentioned earlier that two types of images need different processing for better contour detection. Hence, this contour detection is performed on both the output images found after type one processing and type two processing. From the input image sown in Figure 2, we can see that contour detected from image with separated background foreground (i.e., the processed image of type two processing) is much better than image with region boundary (i.e., the processed image of type one processing).

2.4.1. Detect Contour

Both the processing of image type one and image type two results a background removed, unnecessary edge removed and boundary extracted image. This is the target image of our model for which we perform all the previously mentioned processing. From this image, we detect the connected regions by detecting their contours. We perform this contour detection using OpenCV Library's findContours() [20] method. The method detects contours from a binary image using the Topological Structural Analysis [21] algorithm. Method findContours() finds connected regions and stores them as individual contours by following object's border or outline of an image. Broken or closely connected outlines inhibits findContours() from detecting multiple object region separately. Also, if overlapping object's edges are not previously separated as individual object edges findContours() will detect one single contour containing all overlapped objects within it. Because of these regions our model passes an input image through all those above mentioned procedures. Which eventually makes the input image a noise free, smooth image with background removed and sharp outline of every individual objects.

2.4.2. Compare and Draw Best Contour

The previous step gives us two sets of contours detected from processed image of type one and processed image of type two. So, it is time to select the best set of contours from these two. Figure 3 shows the comparison process for an example image. For this we simply calculate the area of each contour and store these two sets of contour areas into two individual array, say A and B, in descending order (i.e., from one set of contours the contour with largest area is stored at the first index of corresponding array). Next, we filter these two sets by removing contour areas less than a threshold value. Analysing all our collected test images, we found that the contour area of a leaf object is more likely to be greater than area value 150. So, we found this threshold value 150 optimal for all our experiments. Without filtering, sometimes the model might consider a non-leaf region (e.g., region within two leaves, example shown at the right most contour of set B in Figure 3) as leaf region. After filtering, we compare these two arrays. For comparison at first we check whether the number of contour areas of both A and B is greater than five or not. If both sets has more than five contours, we calculate the total amount of area of first five contours for both set A and B and compare the amount. The set with largest amount is selected as the best contour set.

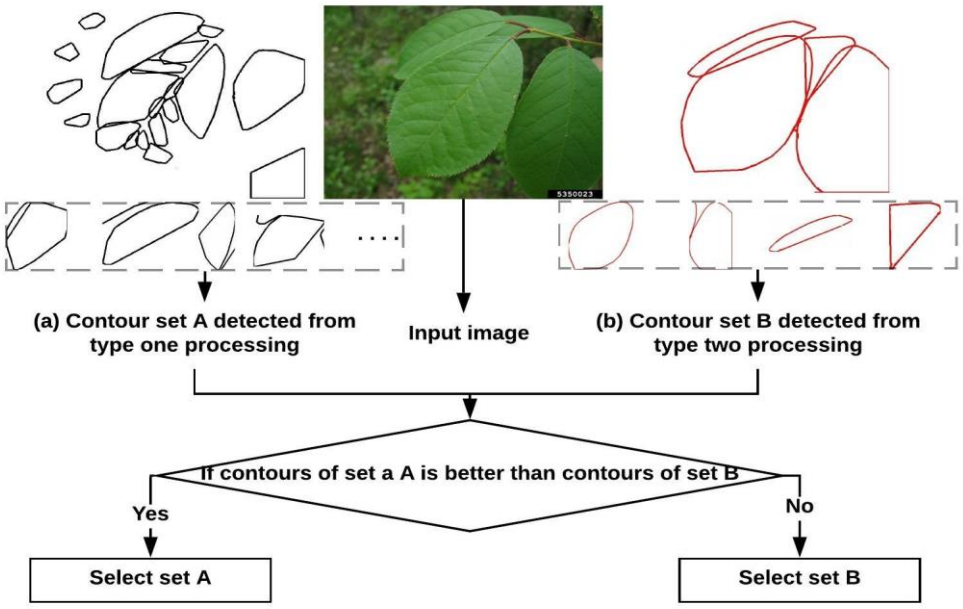

Figure 3. Comparison process between two sets of contours.

If any of set A or B has number of contour areas less than five, we check for the set which has less contour and store the quantity number, say x, of contour areas of that set. Then comparison and best set selection procedure are done as previously using x number of contour areas instead of five. After a large number of test cases we select five as the threshold value for this comparison and it performs well with our experimental dataset. But, we believe this comparison process can be improved with new processes and we are currently working on it. After contour selection, we draw the selected contours on a mask image which has the same size as Main.

2.4.3. Segment Detected Contour (Leaves)

Our model processes an input image following the above mentioned procedures and selects the best set of contours from the image. These contours actually represent the regions of leaves in

main input image. So, the final step is to segment these found contours as regions of image which are actually the regions of individual leaves in Main image.

From our selected set of contour on mask image, for each contour, we consider left-to-right as x direction and top-to-bottom as y direction. Thus we find a point say (a, b) as top left corner of that particular contour. Following that we get a point (say (a, b)) as the top-left vertex and another point (say (c, d)) as the bottom-right vertex for each and every contour region. Using these points we make a rectangular area surrounding each contour within it. Then using these vertex coordinates we crop corresponding region of mask image from the Main image. This cropping is just like array slicing. So, for cropping each contour area from image, we supply the b and d coordinates, followed by a and c coordinates to slice a rectangular portion from Main image that exists within the (a, b) and (c, d) coordinates' surrounding area. Then these rectangular portions are stored as individual images which represent the final leaves segmented from the Main input image.

3. PERFORMANCE AND COMPARATIVE ANALYSIS

In this section we discuss about the performance of our proposed leaf segmentation model along with some comparative analysis. There exists so many datasets for dense object detection or segmentation. But, most of the well-known leaf datasets contain only single leaf image having white or black background. To analyses the performance of our proposed model we need images with single as well as overlapping leaves in complex natural background. Hence, the experiments of our proposed leaf segmentation model are carried out on leaf images collected from the Internet and Pl@ntLeaves dataset [10]. Dataset [10] includes 233 images but most of them are center focused single leaf images in natural background. To ensure our model's performance we build a dataset containing 190 images of leaves where 153 images are collected from [10] and 37 images are randomly collected from the Internet. These images contain 150 single leaves i.e., leaves without occlusion and 196 occluded leaves i.e., leaves involved in occlusion or overlapping one another.

The performance of our proposed model is presented in Table 1. For explaining this comparison we follow the way explained in [8]. From all our collected 190 images we calculate the percentage of correctly segmented single leaves as well as overlapping leaves. Our model's leaf segmentation performance of every individual leaf is evaluated by segmentation rate and failure rate. In Table 1 the portion of leaves indicates overall 346 individual leaves are found within our collected 190 images, where 150 leaves are found single and 196 are found with occlusion. Segmentation rate refers to the proportion of correctly segmented individual laves from the total number of individual leaves. Failure rate refers to the proportion of incorrectly segmented individual laves from the total number of individual leaves.

Table 1. Segmentation performance for occluded and single leaves individually.

	Portion of Leaves	**Segmentation Rate**	**Failure Rate**
Single Leaves	43.35% (150)	95.34% (143)	4.66% (7)
Occluded Leaves	56.65% (196)	86.73% (170)	13.27% (26)
Overall	100% (346)	90.46% (313)	9.53% (33)

Our model successfully segments 313 leaves from 346 leaves of 190 images which ensures 90.46% overall segmentation rate. Within the 346 leaves 196 leaves were found with occlusion and our model correctly segments 86.73% of them (i.e., 170 out of 190). Table 1 also shows that

the rate of single leaf segmentation of our model is 95.34% and it only fails to segment 7 single leaves out of 150.

Analysis the failure rate of our model we found that in most of the cases our model fails to detect leaves whenever the input image if of very low resolution. Also, image being too much blurry hinders the segmentation process.

Within the 26 leaves with occlusion that our model fails to segment, some were over segmented and some were under segmented. The rate of over segmentation (e.g., one leaf segmented into several parts) and under segmentation (e.g., two or multiple leaves segmented as one) is 42.31% (11out of 26) and 57.69% (15 out of 26) respectively. Figure 4 shows some example of segmented leaf regions with Over-segmentation (OS) and Under-Segmentation (US).

Figure 4. Model output with over segmentation and under segmentation.

The performance of our proposed model is quite satisfactory for some reasons such as: (1) our model can segment both single and occluded leaf individually at a high segmentation rate, (2) it can both separate the closely connected leaves (i.e., leaves touching each other closely) and the overlapping leaves (i.e., leaves overlapping one another), (3) it works better on different types of images as all 190 test images are collected from different resource, (4) it provides good accuracy in segmenting leaves of different species having different shape and texture.

Figure 5 shows how accurately our proposed model performs leaf segmentation procedure. Figure 5(a) shows an image with its final leaf segmentation result which is collected from our 37 Internet images and Figure 5(b) shows an image with its final leaf segmentation result which is collected from dataset [10] 153 images.

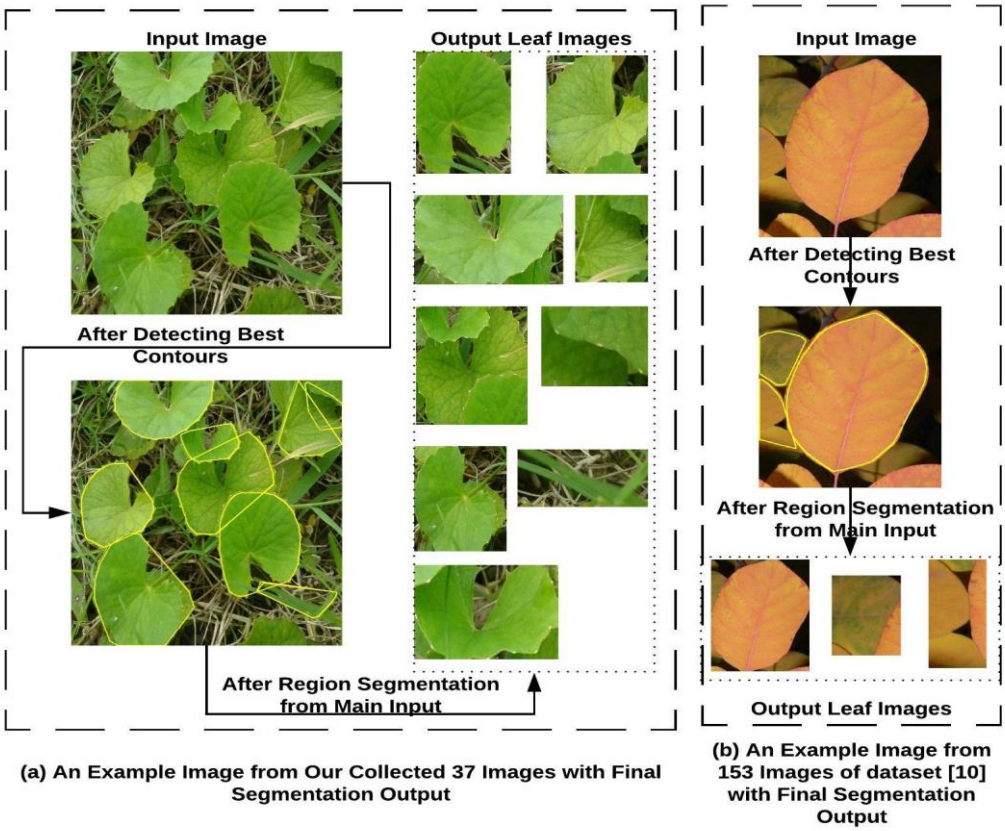

Figure 5. Leaf segmentation example of two images with input and output.

The combination of image processing and best contour selection approach of our model ensures better leaf boundary detection compared to some well-known edge detection algorithms that are usually used for object segmentation. Figure 6 shows a source image along with the output results found after applying watershed, canny, laplacian, sobel and our model's processing on it. Figure 6 also shows the contours detected from watershed, canny, laplacian, sobel and our model's processing applied resultant images which ensures that our model detects better contours, of the two overlapping leaves of Figure 6, individually compared to others.

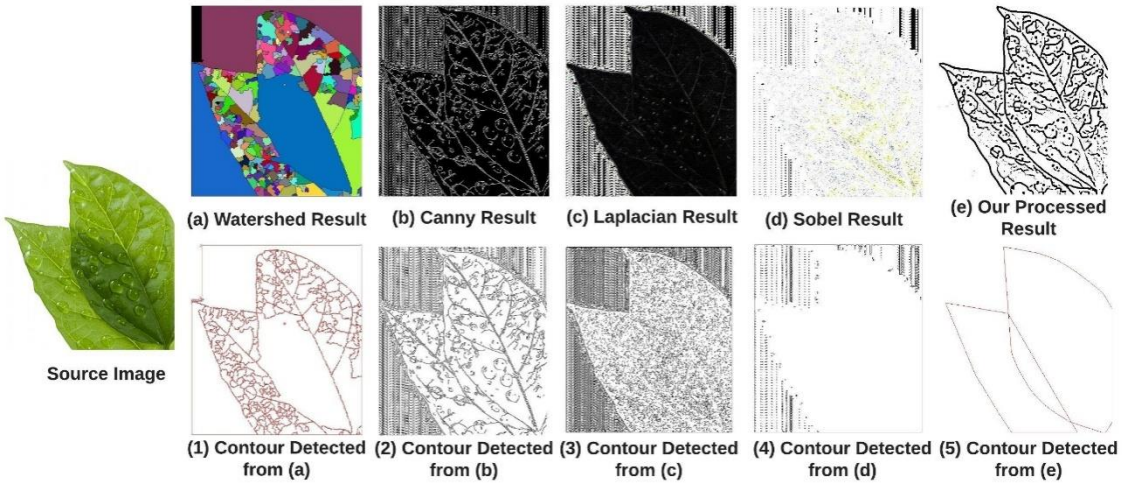

Figure 6. Comparing our model's image processing and contour detection with some other algorithms.

Within our study, most of the existing leaf segmentation models work with different datasets and use different performance measures for evaluating their model. So, instead of comparing accuracy rate, we compare our model on some complex scenario where leaf segmentation becomes tough. Enlisting these scenarios we check whether our model can segment individual leaves, compared to other existing models, covering each scenario.

Table 2. Performance comparison of accurate segmentation on different scenario.

Some complex scenario of segmentation that existing models covers or not covers	**Models**					
	Our model	*Ref. [5]*	*Ref. [7]*	*Ref. [8]*	*Ref. [9]*	*Models of Ref. [11]*
Leaves without having much boundary clue	√	~	~	~	×	√
Image not focusing target leaf object	√	×	×	√	√	×
Image having overlapping leaves at any corner but not always at center of image	√	√	×	√	√	×
Every individual overlapping leaf from image	√	×	√	√	√	√
Image of leaves with different and internal texture	√	√	√	×	×	√

√=Leaf Segmentation Possible, ×= Leaf Segmentation Not Possible, ~ = Criteria not applicable for this model as authors do not clarify.

Table 2 shows a comparative analysis of leaf segmentation from image, of our model with some existing ones, on different complex scenario. It also explains that our proposed model ensures a proper and compelling leaf segmentation from image by covering various complex scenario.

4. CONCLUSIONS

This paper proposes a leaf segmentation system where we aim for segmenting single as well as occluded leaves individually from image. At first we apply some image processing techniques so that the outline or contour edges of every leaf becomes much visible, smooth and sharp which helps in individual object's contour detection. Then our model performs proper filtering and comparison to select the best contour set that matches with the shape of leaves. Finally, our model segments those selected contour areas as leaf regions from main image. From the experimental results, we see that our model archives an overall segmentation rate 90.46% while segmentation rates for single and overlapping leaves are 95.34% and 86.73% respectively, on our created dataset.

The working processes described in this paper is applied for detecting contour edges of overlapping leaves from complex background. In future we look forward to improve and apply these techniques on different sectors such as medical cell images where overlapping objects are found within low variance complex background. Our future works will also focus on improving the performance of this model even with less processing.

REFERENCES

[1] Guyer DE, Miles GE, Schreiber MM, Mitchell OR, Vanderbilt VC. Machine vision and image processing for plant identification. Transactions of the American Society of Agricultural Engineers 1986;29(6):1500-1507.

[2] Niu C, Li H, Niu YG, Zhou ZC, BU YL, Zheng WG. Segmentation of cotton leaves based on improved watershed algorithm. In: 9th International Conference on Computer and Computing Technologies in Agriculture, CCTA 2015, Beijing, China, 27-30 September 2015 .

[3] Dornbusch T, Andrieu B. Lamina2Shape-an image processing tool for an explicit description of lamina shape tested on winter wheat (Triticum aestivum L.). Computers and Electronics in Agriculture 2010;70(1):217-224.

[4] Wu P, Li WL, and Song WL. Segmentation of leaf images based on the active contours. International Journal of u- and e- Service, Science and Technology 2015;8(6):63-64.

[5] Z. Wang, K. Wang, F. Yang, S. Pan, Y. Han, Image Segmentation of Overlapping Leaves Based on Chan–Vese Model and Sobel Operator, Information Processing in Agriculture (2017), doi: https://doi.org/ 10.1016/j.inpa.2017.09.005.

[6] Cerutti G, Tougne L, Mille J, Vacavant A, Coquin D. Understanding leaves in natural images - a model-based approach for tree species identification. Computer Vision and Image Understanding, 2013;117(10):1482-1501.

[7] Itakura, Kenta, and Fumiki Hosoi. "Automatic leaf segmentation for estimating leaf area and leaf inclination angle in 3D plant images." Sensors 18.10 (2018): 3576.

[8] Xia, Chunlei, et al. "In situ 3D segmentation of individual plant leaves using a RGB-D camera for agricultural automation." Sensors 15.8 (2015): 20463-20479.

[9] Morris, Daniel. "A pyramid CNN for dense-leaves segmentation." 2018 15th Conference on Computer and Robot Vision (CRV). IEEE, 2018.

[10] H. Goëau et al., “The CLEF 2011 plant images classification task,” Image CLEF 2011 working notes.

[11] Grand-Brochier, Manuel, et al. "Tree leaves extraction in natural images: Comparative study of preprocessing tools and segmentation methods." IEEE Transactions on Image Processing 24.5 (2015): 1549-1560.

[12] E. Gastal, “Non photorealistic rendering using opencv(python, c++) | learn opencv.”[Online], Available: https://www.learnopencv.com/non-photorealisticrendering- using-opencv-python-c/.

[13] Rafflesia Khan, Rameswar Debnath, " Multi Class Fruit Classification Using Efficient Object Detection and Recognition Techniques", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.11, No.8, pp. 1-18, 2019.DOI: 10.5815/ijigsp.2019.08.01

[14] “Image Filtering Techniques in OpenCV.” Packt Hub, 18 Apr. 2018, [Online], Available at: https://hub.packtpub.com/image-filtering-techniques-opencv/.

[15] “Dilation.” Morphology - Dilation, [Online], Available at : https://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm.

[16] “Smoothing Images.” OpenCV, [Online], Available at : https://docs.opencv.org/3.1.0/d4/d13/tutorial_py_filtering.html.

[17] P. O. A. Thresholding, “Adaptive Thresholdings,” (2003). [Online; last accessed 06-April-2019].

[18] OpenCV - Adaptive Threshold, Tutorials Point, [Online], Available at: https://www.tutorialspoint.com/opencv/opencv_adaptive_threshold.htm

[19] S. F. BogoToBogo_K Hong Ph.D.Golden Gate Ave, “Image thresholding and segmentation.” [Online]Available:https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Global_Thresholding_Adaptive_Thresholding_Otsus_Binarization_Segmentations.php

[20] Structural Analysis and Shape Descriptors - OpenCV 2.4.13.7 Documentation, [Online], Available at https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=drawcontours#drawcontours

[21] Suzuki, Satoshi. "Topological structural analysis of digitized binary images by border following." Computer vision, graphics, and image processing 30.1 (1985): 32-46.

AUTHORS

Rafflesia Khan is a student of M.Sc. at Computer Science and Engineering Discipline, Khulna University, Khulna, Bangladesh. She has completed her Bachelor’s degree from Computer Science and Engineering Discipline, Khulna University, Khulna, Bangladesh in 2017. Her research areas of interest are image processing, visual object detection & recognition, machine learning, pattern recognition, and internet of things security.

RameswarDebnath is a Professor of Computer Science and Engineering Discipline at Khulna University, Khulna, Bangladesh. He has completed his Bachelor degree from Computer Science and Engineering discipline, Khulna University, Khulna, Bangladesh in 1997. He has received his Masters in Engineering degree and Ph.D. degree in Computer Science and Engineering from the University of Electro-Communications, Tokyo, Japan in 2002 and 2005 respectively. His research areas of interest are image processing, statistical machine learning and its applications to pattern recognition, visual object detection and natural language processing.