Automated Image Processing Pipeline for Separating Flower Material from Background in Botanical Research

Sofia Saeed Ahmed
School of Computer Science
University of Nottingham Malaysia
hcysa3@nottingham.edu.my

Anjelie Kandasamy
School of Computer Science
University of Nottingham Malaysia
hcyak4@nottingham.edu.my

Afra Adnan Qadir
School of Computer Science
University of Nottingham Malaysia
hcyaq1@nottingham.edu.my

Zaid Faisal
School of Computer Science
University of Nottingham Malaysia
hcyzs2@nottingham.edu.my

***Abstract*—This research project addresses the automation of flower material segmentation from background images using image processing techniques, aiming to meet the demand for precise segmentation methods in various fields, including biological sciences and industrial applications. The proposed methodology involves a three-stage segmentation process: pre-processing, Otsu's thresholding calculation, and post-processing.**

Pre-processing involves median filtering and Gaussian blur to optimise input images, while Otsu's thresholding ensures efficient binary conversion. Subsequent post-processing utilises morphological operations to refine segmentation results, enhancing accuracy and quality. By systematically integrating these techniques, the research achieves effective segmentation of flower images, laying the groundwork for further analysis and applications in diverse fields.

***Keywords*—Image processing, flower segmentation, Otsu Thresholding, Median Filter, mIoU metrics, OpenCV, color conversion, HSV, morphology**

I. Introduction

In recent years, the intersection of image processing and biological sciences has seen significant advancements, driven by the growing need for precise analysis of biological structures and phenomena. Among the various applications of image processing, the segmentation of biological materials from complex backgrounds stands out as a critical challenge with broad implications. [1]

Our coursework addresses this challenge by focusing on the automation of flower segmentation—a crucial task for advancing genetic research and agricultural innovation.

With the advent of advanced imaging technologies, biologists now have access to high-resolution images that provide unprecedented insights into the genetic structure and behaviour of plants. However, the lack of efficient methods for analysing these images hampers efforts to understand the complex interplay of genetic factors influencing plant traits. [2]

We use OpenCV-Python for shape and color detection, taking advantage of its capabilities across various fields. We assess the method's effectiveness through both qualitative and quantitative metrics, including mean Intersection over Union (mIoU), to gauge the accuracy and reliability of segmentation results. [3]

Our coursework focuses on creating a pipeline that segments flower material from backgrounds in images. We leverage techniques such as color space conversion, noise reduction, thresholding, and binary image processing to automate the identification of flower regions. This pipeline aims to provide biologists with a valuable tool for extracting quantitative data

facilitating informed research.

II. Literature Review

Over the years, flower segmentation through image processing techniques has shifted towards more sophisticated methods such as Otsu's thresholding and graph cuts. While earlier methods focused on edge-based algorithms and color similarity measures, these often struggled with background clutter and complex imagery.

Subsequent strategies pivoted towards leveraging the color attributes of flowers as segmentation criteria. For instance, in a study, color patches from flower images served as reference points for measuring color similarity, facilitating flower extraction based on obtained similarity scores. Statistical color models, derived from color combinations, were iteratively utilized to segment flowers. [1]

In the late 2000s, researchers delved into more intricate segmentation methods. Maria-Elena Nilsback developed an iterative segmentation scheme capable of segmenting flowers solely based on a general color model, without specific image information. This scheme incorporated a geometric model and continually updated the color model to enhance segmentation precision.

Several studies utilize OpenCV and Python for flower segmentation. For instance, one showcases an image processing pipeline that uses these tools to effectively segment a flower from a plant image, removing noise and isolating the flower from the background. Moreover, OpenCV's ability to perform thresholding in a single line allows it to be an efficient library.[4]

Other than thresholding, pre and post processing play a huge role in shaping the image quality. Several studies show the use of Mean or Median filters for noise reduction. Moreover, morphological operations such as closing have been used to improve the after affects of thresholding.[5]

In summary, the trajectory of flower segmentation using image processing methods has progressed from basic edge-based approaches to sophisticated color-based methods using Python functions, and most recently, to deep learning-based techniques. These advancements have significantly enhanced the accuracy and resilience of flower segmentation, fostering applications in domains like agriculture and computer vision.

III. Methodology

In this study, we developed an image processing pipeline to separate flower material from the background in a series of images. Our approach consisted of three main stages: pre-processing, thresholding, and post-processing. During the pre-processing stage, we converted the images to a different color space and applied various smoothing techniques. For thresholding, we utilized the Otsu thresholding algorithm to segment the images based on intensity. In the post-processing stage, we employed morphological operations to refine the segmentation results and fill any gaps.

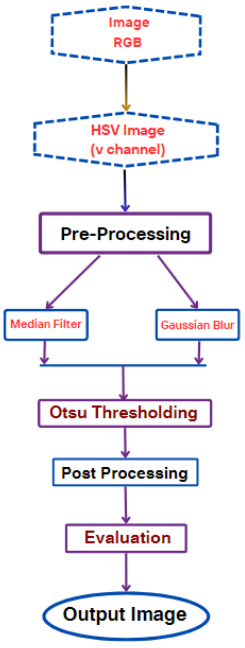

Figure 1. Our Image Processing Pipeline

1) Color Space Conversion

The process begins with assessing the color distribution in the images. Flower images often exhibit vibrant and high-intensity colors, which is beneficial for color-based segmentation. Color can be represented in different color systems, each offering unique advantages. Choosing the appropriate color space is crucial for effective segmentation as it can impact the results significantly.[6]

After extensive research, we considered three options: RGB, LAB, and HSV. Analysis of these color spaces highlighted potential challenges with the RGB space. It creates a nonlinear and discontinuous space, making it difficult to track changes in color hue due to these interruptions. Additionally, the RGB model is sensitive to illumination changes, complicating color tracking and analysis. These factors can pose challenges when working with the RGB color space in image segmentation tasks. Hence, we decided to explore LAB and HSV. LAB color space is designed to be perceptually uniform, meaning that a change in color value corresponds to a change of similar visual importance. This makes it particularly useful for distinguishing colors accurately. HSV color space represents color in terms of hue, saturation, and value, which is often more intuitive and closer to human perception, making it effective for isolating objects in an image based on their color properties.

In one research paper [7], the authors divided the LAB color space into its individual components: L, A, and B. They applied OTSU thresholding to each component separately and evaluated each resulting segmentation. They then pre-processed the final segmentation. The methodology outlined in the research paper is shown in Figure 2.

We experimented with this approach but found the results to be suboptimal. Consequently, we explored a similar methodology using the HSV color space. Instead of applying thresholding to each component separately, our results indicated that the V-channel is the most effective for segmentation. This is due to its representation of brightness. This channel provides a clear distinction between the foreground and background based on light intensity, which aligns well with our goal of isolating flowers from their surroundings. By focusing on the V-channel, we were able to achieve more precise and consistent results in our segmentation tasks.

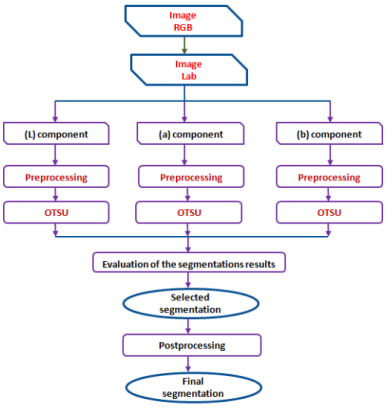

Figure 2. Difference steps of the proposed method

Figure 3. Breakdown of each HSV component

A. Pre-Processing

The pre-processing stage focuses on preparing the images for effective segmentation by separating the foreground flowers from the background. The dataset provided three levels of images - easy, medium, and hard- so we began our methodology by focusing on the easy level.

After extracting the V-channel, we applied a median filter using the 5x5 neighborhood to eradicate the noise. We then applied the Gaussian blur which smoothes uneven pixel values by cutting out the extreme outliers. These techniques help smooth out any imperfections and provide a clear distinction between the flowers and the back-

ground, setting the stage for successful subsequent stages of the pipeline.

B. Otsu Thresholding

Otsu's thresholding is an image segmentation technique that automatically calculates the optimal threshold to convert an image to binary. This characteristic offers an advantage over traditional thresholding methods, which rely on manually inputting threshold values. Since our dataset has multiple images, a threshold value set manually for one image did not coincide with other images. Hence, Otsu was the best way to go about it.

This method analyzes the histogram of the image's pixel intensities to find the best threshold that separates the image into two classes: foreground and background. By maximizing the separation between these classes, Otsu's method determines the ideal threshold value for dividing the image. This approach is particularly effective when the image has a clear bimodal distribution, representing distinct foreground and background areas. Once the threshold is calculated, the image is segmented by classifying pixels with values below the threshold as background and those above as foreground. This method provides an efficient and automated way to achieve image binarization, making it a valuable tool in image processing.[7]

Although Otsu Thresholding has its formula, the OpenCV Python library streamlines image processing tasks by providing built-in functions for complex operations such as thresholding, filtering, and segmentation. These functions handle the intricate mathematical calculations behind the scenes, making it easy to implement advanced techniques like Otsu's thresholding with just a few lines of code.

We also explored advanced segmentation techniques such as GrabCut and Graph Cut to improve our results. Although effective, these methods require significant user input. GrabCut relies on initial user-defined regions, while Graph Cut requires specific constraints and preferences. Since we aimed for an automated image-processing pipeline, we focused on Otsu Thresholding which requires minimal user input to enhance efficiency and usability.

Figure 4. Orignal Image and Otsu Thresholded Image

C. Post-Processing

Once the image is segmented, we have to post-process it. This is the final stage that refines the segmented image to improve the quality and accuracy of the results. We used morphological operations such as opening and closing to remove the small holes, smoothen the boundary and remove noise. We also used an additional step for further cleaning. This fine-tuning of the output ensures that the final segmented image accurately represents the foreground object. [6]

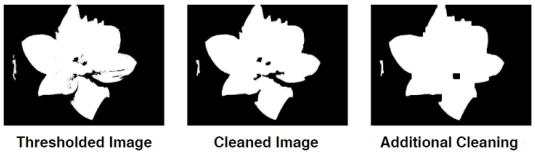

Figure 5. Thresholded Image vs Post-Processed Image

Figure 5. shows how post-processing helps to fill holes and produce a better quality output.

IV. Results and Discussion

A. Evaluation Metrics

We used the mean Intersection over Union (mIoU) metric to evaluate the performance of the segmentation pipeline. It measures the accuracy of the model's segmentation by comparing the predicted segments with the ground truths.

A mIoU score ranges from 0 to 1, with a higher value indicating better model performance. A mIoU of 1 means perfect segmentation, while

a mIoU of 0 indicates poor or no overlap between the predicted and ground truth segments.

B. Equations

To calculate mIoU, we first calculated the IoU of each output image against its ground truth provided in the dataset. We used the following formulas to calculate the mIoU:

1. To calculate intersection and union:

$$\text{Intersection} = \sum(\text{predicted} \wedge \text{ground truth})$$

$$\text{Union} = \sum(\text{predicted} \vee \text{ground truth})$$

2. To calculate IoU:

$$\text{IoU} = \frac{\sum \text{intersection}}{\sum \text{union}}$$

3. Accumulate IoU:

$$\text{total\_iou} = \text{total\_iou} + \text{iou}$$

$$\text{num\_images} = \text{num\_images} + 1$$

4. Calculate mean IoU (mIoU):

$$\text{mIoU} = \frac{\text{total\_iou}}{\text{num\_images}}$$

Figure 6. Equations used

C. Results

We separately printed the IoU of each image in our dataset and the mIoU was calculated using total IoU divided by 9.

The data in the table below consists of Intersection over Union (IoU) and cumulative IoU values for the three categories of images: easy, medium, and hard. Overall, the mean IoU (mIoU) across all categories is 0.795, which suggests that the segmentation process is effective.

Looking at the easy images, the IoU values start strong with 0.791 and 0.763, but then there is a notable drop to 0.274 for the third easy image. This lower performance affects the cumulative IoU and the overall mIoU for the easy category. For medium images, the IoU values are quite high—0.797, 0.903, and 0.892—resulting in a cumulative IoU of 4.42. This indicates the segmentation process works effectively for medium-level images.

	IoU	Cumulative IoU
Easy 1	0.791	0.791
Easy 2	0.763	1.554
Easy 3	0.274	1.827
Medium 1	0.797	2.625
Medium 2	0.903	3.528
Medium 3	0.892	4.42
Hard 1	0.921	5.341
Hard 2	0.888	6.23
Hard 3	0.922	7.151
mIoU	**0.795**	

Figure 7. Results of Quantitative Analysis

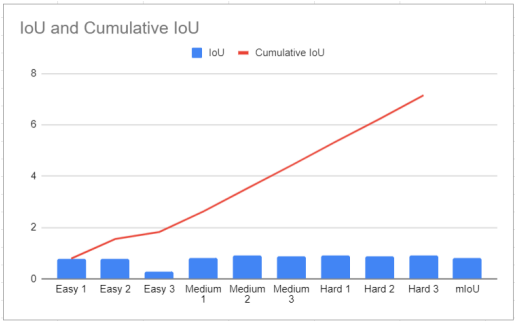

Figure 8. Graphical Representation of Evaluation Results

The hard images yield consistently high IoU values of 0.921, 0.888, and 0.922, resulting in a cumulative IoU of 7.151. These results suggest that even for challenging images, the segmentation approach is successful.

In conclusion, while the medium and hard images show excellent results, the inconsistency in the easy images affects the overall performance. Future improvements may focus on addressing this inconsistency to achieve more uniform performance across all levels of difficulty.

V. Conclusion

In our study, we developed an effective and automated image processing pipeline for flower image segmentation using HSV color space and Otsu's thresholding. The approach we adopted provided efficient results and minimized the need for manual input, streamlining the overall process. Although the method achieved a strong mean IoU (mIoU) across different levels of image complexity, some inconsistencies were observed in the easy-level images. Future work can focus on addressing these disparities to improve performance across all difficulty levels.

Despite the promising outcomes, there is potential for further refinement in the segmentation process, especially in cases where Otsu's method may not be the optimal choice. Exploring advanced algorithms or hybrid approaches may enhance the accuracy and robustness of the segmentation pipeline. Overall, our work provides a solid foundation for future advancements in flower image segmentation, offering valuable insights and pathways for ongoing research in this area.

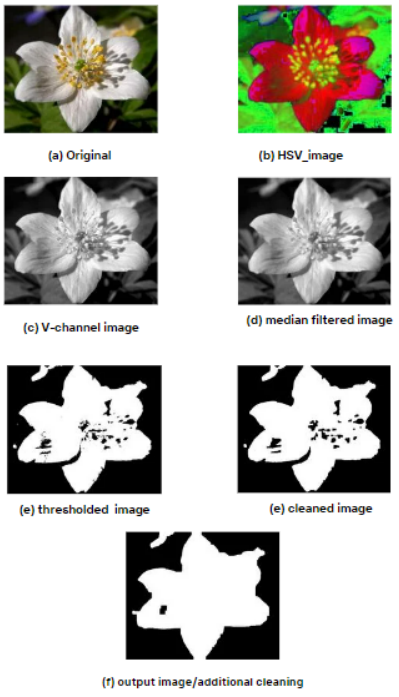

Figure 9. Final Pipeline Results

VI. References

1) Mohammed Khalid Hossen, Sayed Mashequl Bari, Partho Protim Barman, Rana Roy, Pranajit Kumar Das, Application of Python-OpenCV to detect contour of shapes and colour of a real image, Sylhet Agricultural University, Sher-e-Bangla Agricultural University, DOI: 10.5281/zenodo.6576264, Published Date: 24-May-2022

2) Dias, Philipe A.; Tabb, Amy; and Medeiros, Henry P., "Multispecies Fruit Flower Detection Using a Refined Semantic Segmentation Network" (2018). Electrical and Computer Engineering Faculty Research and Publications. 582.

3) Mohammed Khalid Hossen, Sayed Mashequl Bari, Partho Protim Barman, Rana Roy, Pranajit Kumar Das, Application of Python-OpenCV to detect contour of shapes and color of a real image, Journal of Sylhet Agricultural University and Sher-e-Bangla Agricultural University, Published Date: 24-May-2022, DOI: 10.5281/zenodo.6576264

4) https://learnopencv.com/cropping-an-image-using-opencv/

5) B. K. V. Prasad, M. V. D. Prasad, Ch. Raghava Prasad, N. Sasikala and Sunita Ravi Department of Electronics and Communication Engineering, Koneru Lakshmaiah Education Foundation, Green Fields, Vaddeswaram, Guntur DT, India

6) Lodh, Avishikta Parekh, Ranjan. (2017). 2017 : Flower Recognition System based on Color and GIST Features.

7) Flower image segmentation based on color analysis and a supervised evaluation, The 2nd International Conference on Communications and Information Technology (ICCIT): Wireless Communications and Signal Processing, Hammamet, 2012, ISBN: 978-1-4673-1950-8, IEEE.