K-PIE: using K-means algorithm for Percentage Infection symptoms Estimation

Vanessa Bueno-Sancho[1*], Pilar Corredor-Moreno[1], Ngonidzashe Kangara[1] and Diane G.O. Saunders[*1]

[1]John Innes Centre, Norwich Research Park, Norwich NR4 7UH, UK
[*]For any queries contact: vbuens@gmail.com; Diane.Saunders@jic.ac.uk

Abstract

The K-means algorithm is one of the most effective clustering methods that has been widely used in plant disease detection. Herein, we developed a script termed K-PIE (K-means algorithm for Percentage Infection symptoms Estimation) that utilises the k-means algorithm to analyse images of both yellow and stem rust infected wheat leaves to estimate the percentage of disease symptoms based on colour analysis.

Introduction

Manual quantification of disease symptoms in plants is laborious and can lead to bias in the results[1]. This has led to machine learning (ML) algorithms becoming increasingly popular for automated image processing of plant disease symptoms[2]. ML algorithms based on image segmentation have been used extensively for stress phenotyping[3], identifying diseases [4] and quantification of infection[5,6]. Image segmentation facilitates the identification of objects in images and the grouping of pixels with similar characteristics. Several techniques for image segmentation have been developed, including those based on thresholds, clusters, edge detection, and neural networks[7]. One of the most effective methods are clustering-based methods, of which the most widely used is the k-means algorithm. This algorithm is an unsupervised method that segments the given data into groups (K-clusters) based on the k-centroids[8]. These k-means algorithms present a simple solution that allows rapid quantification of disease symptoms in images without supervision. Herein, we applied a k-means algorithm for quantifying the percentage of disease symptoms in wheat leaves infected with the yellow rust (YR) fungus (*Puccinia striiformis* f. sp. *tritici*) and the stem rust (SR) fungus (*Puccinia graminis* f. sp. *tritici*).

Material and Methods

A python script was created to process input images and estimate the percentage of infection symptoms, which depending on the colour selected could be pustules and/or include chlorosis. The main script (k_pie.py) calls functions from infection_functions.py, assuming both scripts are in the same directory. The following arguments are included in k_pie.py:

-p: positive control; image file of infected leaf.
-p_rgb: alternatively, RGB colour values as a positive control.
-n: negative control; image file of uninfected leaf.
-n_rgb: alternatively, RGB colour values as a negative control.
-i: input directory containing pictures to analyse.
-f: format of the input images (png, jpg or tiff); default: .png.
-o: output directory for results; default: results.
-k: k value; default: 5.

The python package OpenCV[9] was used for both image processing and the k-means clustering

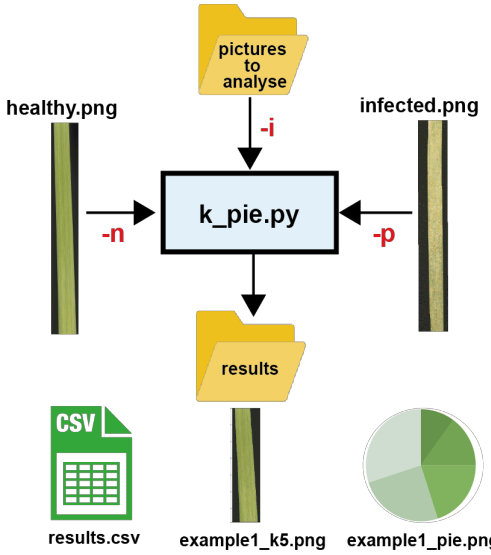

Figure 1. Illustration to show the necessary input files and output files created. A directory containing the images to analyse, and two control pictures are required. An output directory is created containing a CSV file with the results, the processed images and a pie chart representing the colours of the clusters found in the given images.

algorithm. This algorithm classifies the pixels of the image into *k* groups of different colours. To determine the centre colours from a picture, the algorithm uses random centres until the termination criteria are met i.e. when accuracy is equal to 1 or reaches 10 iterations. A total of 10 attempts are used for initial labelling. The python packages NumPy[10] and Matplotlib[11] are also used for managing arrays and plotting figures respectively.

The python script first reads the control pictures, reshaping them to input them into the k-means algorithm using *k*=10 (10 clusters). When the termination criteria are met, the centre colours determined for each of the 10 clusters are saved and the percentage of each colour found in the picture estimated. The most abundant colour for each control is saved as the "standard value" for healthy and infected leaves. Alternatively, RGB colour values can be given as positive and/or negative controls and those would then be used as the "standard value" for healthy and infected leaves. RGB values have to be given as integers and separated by commas as shown in results.

The input pictures are then analysed using the k-means algorithm as previously described with the given *k* to calculate the centre colours. Each one of these values is then classified into "healthy" or "infected", depending on its closeness to the standard values. The percentage of "infected" colours is estimated and output into the results.csv file.

Results

Using control images to find standard values

The K-PIE script takes three mandatory inputs: (i) An image of a healthy leaf as a negative control (using the argument *-n*), (ii) an image of an infected leaf (argument *-p*), and (iii) an input directory containing all images for analysis (Figure 1). The default format of the input images is set as .png but can also be altered to .jpg or .tiff if appropriate using the *-f* argument.

Execution of k_pie.py using default values (k=5, output=results, image format=.png):

```
python k_pie.py -p infected.png -n healthy.png -i inputpictures/
```

Once complete, a directory containing the results is created. For each picture, a pie chart with colours identified in the picture is created. A file in comma separated value (CSV) format is generated that contains the percentage of infection estimated for each image and the k-value used (Figure 2).

Using RGB colour values as standard values

Alternatively, the script can be executed using RGB values. This is the preferred option when control pictures are not readily available or of poor quality. Alternatively, it can be used for detecting a particular symptom e.g. pustules by colour. In this case, the values are given as integer numbers (no decimals) and separated by commas.

Execution of python k_pie.py using default values (k=5, output=results, image format=.png):

```
python k_pie.py -p_rgb 205,133,63 -n_rgb 137,155,114 -i inputpictures/
```

As before, once the script has completed a directory containing the results is created (Figure 3).

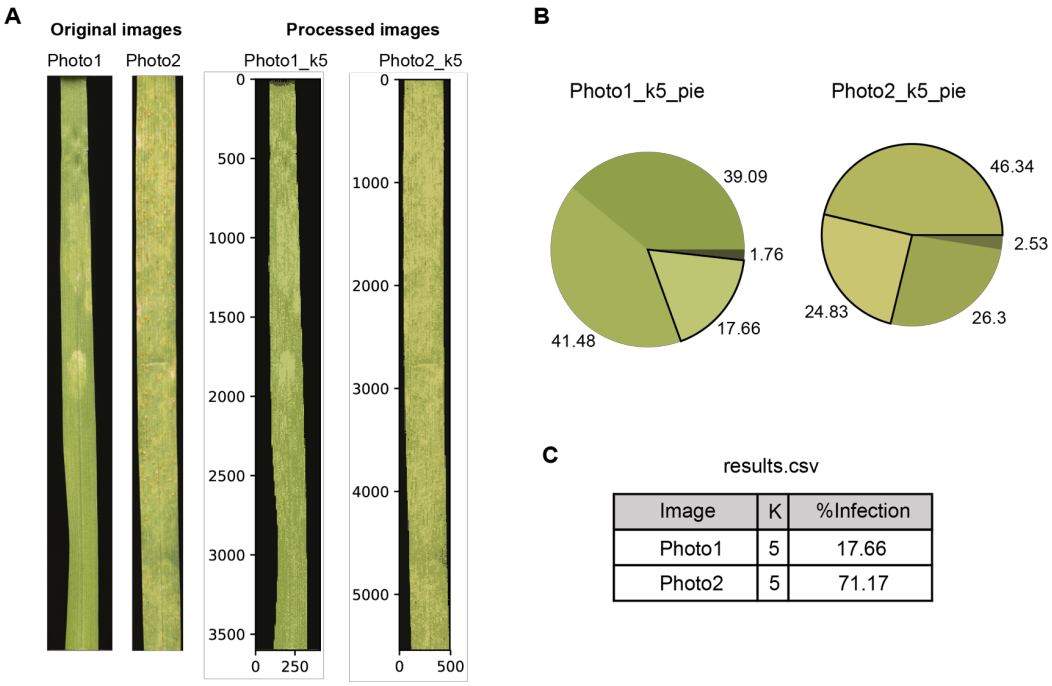

Figure 2. K-PIE output using control images as input. A. Original and processed images. Left, original images; right, images following processing. Axes represent number of pixels. **B.** Pie chart showing the percentage of each colour identified in the image. Black-outlined section highlights the colour considered as infected. Numbers represent percentages **C.** Results in CSV format. From the first picture, using 5 clusters (k=5), the percentage infection was estimated as 17.66 %, whereas 71.17 % infection was estimated for the second leaf.

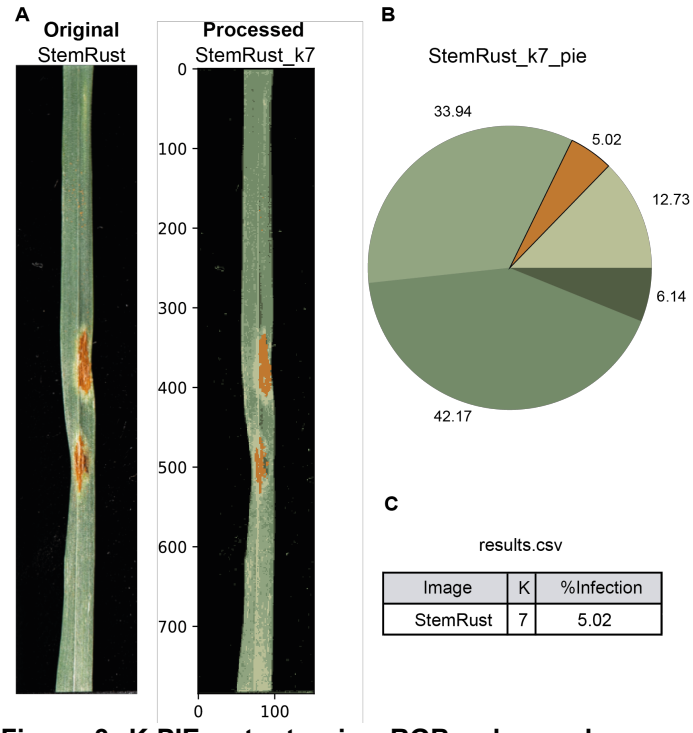

Figure 3. K-PIE output using RGB colour values as input. A. Original and processed images. Left, original image; right, image after processing. Axes represent number of pixels. **B.** Pie chart showing the percentage of each colour identified. The black-outlined section represents the colour considered as infected. Numbers represent percentages. **C.** Representation of estimated results in CSV format. For the given image, using 7 clusters (k=7), 5.02 % infection was estimated.

Conclusion

This script illustrates how the k-means algorithm can be used to estimate the percentage of infection in leaves displaying disease symptoms. In addition, the use of RGB values facilities the precise quantification of sporulating area of a leaf in the case of SR. This presents a very simple solution for estimating the percentage of infection symptoms when quantifying images from a large number of infected leaves. Whereas many current methods require substantial pre-processing by the user (i.e. marking the area to quantify), this tool allows you to rapidly quantify disease symptoms in a high-throughput manner.

Additional information

The python scripts k_pie.py and infection_functions.py can be found in the following GitHub repository: https://github.com/vbuens/k_pie, alongside the requirements for executing the script.

Author contributions

VBS conceived and designed the tool, prepared the figures and wrote the text. PCM and NK provided images and tested the script. DGOS provided feedback on the text. All authors read and approved the final report.

Funding

V.B.S. was supported by the BBSRC Norwich Research Park Biosciences Doctoral Training Partnership (grant number BB/M011216/1), P.C.M. by the European Research Council (no. 715638), N.K. by the European Union's Horizon 2020 research and innovation programme under Marie Skłodowska-Curie (no. 674964) and John Innes Centre Science for Africa Scholarship. Additional funding was provided to D.G.O.S. by the BBSRC Institute Strategic Programmes BB/P016855/1 and BB/P012574/1, and the John Innes Foundation.

References

1. Bock, C. H., Poole, G. H., Parker, P. E. & Gottwald, T. R. Plant disease severity estimated visually, by digital photography and image analysis, and by hyperspectral imaging. *CRC. Crit. Rev. Plant Sci.* (2010). doi:10.1080/07352681003617285
2. Khirade, S. D. & Patil, A. B. Plant disease detection using image processing. in *Proceedings - 1st International Conference on Computing, Communication, Control and Automation, ICCUBEA* (2015). doi:10.1109/ICCUBEA.2015.153
3. Singh, A., Ganapathysubramanian, B., Singh, A. K. & Sarkar, S. Machine Learning for High-Throughput Stress Phenotyping in Plants. *Trends in Plant Science* (2016). doi:10.1016/j.tplants.2015.10.015
4. Camargo, A. & Smith, J. S. An image-processing based algorithm to automatically identify plant disease visual symptoms. *Biosyst. Eng.* (2009). doi:10.1016/j.biosystemseng.2008.09.030
5. Hitimana, E. & Gwun, O. Automatic Estimation of Live Coffee Leaf Infection Based on Image Processing Techniques. *arXiv preprint*. (2014). doi:10.5121/csit.2014.4221
6. Wijekoon, C. P., Goodwin, P. H. & Hsiang, T. Quantifying fungal infection of plant leaves by digital image analysis using Scion Image software. *J. Microbiol. Methods* (2008). doi:10.1016/j.mimet.2008.03.008
7. Bhanu, B. & Lee, S. Image segmentation Techniques. Genetic Learning for Adaptive Image Segmentation. *Springer US* (1994). doi:10.1007/978-1-4615-2774-9_2
8. Pal, N. R. & Pal, S. K. A review on image segmentation techniques. *Pattern Recognit.* (1993). doi:10.1016/0031-3203(93)90135-J
9. Bradski, G. R. & Kaehler, A. Learning OpenCV - computer vision with the OpenCV library: software that sees. *O'Reilly* (2008). doi:10.1109/mra.2009.933612
10. Oliphant, T. E. Python for scientific computing. *Comput. Sci. Eng.* (2007). doi:10.1109/MCSE.2007.58
11. Hunter, J. D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* (2007). doi:10.1109/MCSE.2007.55