BIULETYN INSTYTUTU HODOWLI I AKLIMATYZACJI ROŚLIN Nr 289/ 2020 : 31–35
E-ISSN: 2657–8913 DOI: 10.37317/biul-2020-0019

Original research paper
Oryginalny artykuł naukowy

Automation of septoria disease severity assessment using digital image analysis in Python programming language

Automatyzacja oceny nasilenia objawów chorobowych septorioz zbóż z wykorzystaniem komputerowej analizy obrazu w języku programowania Python

Sławomir Bartosiak ✉

Instytut Hodowli i Aklimatyzacji Roślin – Państwowy Instytut Badawczy,
Zakład Fitopatologii, Pracownia Hodowli Odpornościowej,

✉ e-mail: s.bartosiak@ihar.edu.pl

The assessment of the severity of symptoms of septoria disease is both time- and labour consuming. The paper presents a means of automating the evaluation of cereal septoria disease symptoms using open-source applications developed in the Python programming language. The software enables computerized documentation and reading of study subject items and the assessment of individual leaf disease symptoms, making it possible to remove outliers from the analysis.

Key words: Python, digital image processing, septoria, Stagonospora Nodorum Blotch, leaves, wheat, triticale.

Ocena objawów chorobowych septorioz na liściach zbóż, opisywanie i tworzenie dokumentacji fotograficznej poszczególnych liści jest czasochłonnym i pracochłonnym zadaniem. W opracowaniu przedstawiona została automatyzacja oceny objawów chorobowych septorioz zbóż za pomocą aplikacji open-source stworzonych w języku Python. Oprogramowanie umożliwia automatyzację odczytywania nazw obiektów doświadczalnych oraz ocenę objawów chorobowych poszczególnych liści, dzięki czemu istnieje możliwość np. usuwania obserwacji odstających z analizy.

Słowa kluczowe: Python, analiza obrazów cyfrowych, septoriozy, liście, pszenica, pszenżyto

Introduction

Rapid disease severity assessment is one of the key elements of plant resistance breeding (Singh, Misra; 2017). In the past, the most common approach was to use visual assessment of the severity of disease symptoms based on an adopted scale; however, with the development of information technologies, digital imaging and advanced multispectral imaging, old and subjective visual methods are rarely employed today. This is because visual assessment is prone to errors and is slower in comparison to computationally efficient digital image analysis (Thomas et al., 2018). Digital image processing can be carried out using both commercial and open source software solutions. One of the major disadvantages of commercial solutions concerns their additional cost, which can be easily reduced by using open source software. Open source solutions, such as ImageJ or ImageJ2 are useful tools; however, in many cases they may be difficult to employ due to complicated graphical user interfaces and numerous features that users need to familiarise themselves with. Batch processing and macros are available in these solutions; however, creating them can be a daunting and arduous task (Easlon and Bloom 2014; Rueden *et al.* 2017). Early versions of ImageJ also did not utilise the full computational power of modern multi-core CPUs. With development of high level programming languages and free access to standard libraries, software development has been facilitated. One of the most popular and user-friendly high level programming languages is Python. This can be readily applied in developing computer vision applications for plant breeding (Easlon, Bloom; 2014).

Pathogenicity tests in controlled environments in the seedling stage facilitate rapid disease symptom severity assessment, leading to shortening of plant breeding programs. These tests are widely employed in research and plant breeding (Thomas et al., 2018, Easlon and Bloom, 2014; Singh and Misra, 2017).

Managing editor: Anna Linkiewicz

One of the work-intensive and time-consuming tasks in these kinds of environmental chamber experiments comprises creating photographic documentation and labelling image samples. This paper presents the methodology for using Quick Response (QR) codes for automated label generation and reading of experimental objects.

In this paper, the term "disease severity" refers to a quotient of diseased tissue area and total area (sum of diseased and healthy leaf tissue), expressed as a percentage. Two approaches can be employed in the assessment of disease severity in triticale and wheat leaves – either weighted disease severity average (sum of diseased area to total area of the leaves) or arithmetic average of disease severity in each diseased leaf. In order to calculate the weighted average, it is necessary to segment the source image into individual leaves. The use of weighted average increases the weight of the leaves with a higher surface area on the assessment of the extent of the disease in the test subject, while the use of arithmetic average lowers the impact of leaf surface area on the disease severity. The arithmetic average approach can be useful in test subjects where leaf surface area variation is significant. The aim of the study was to develop an application using the Python programming language, facilitating rapid severity assessment of septoria disease (*Parastagonospora nodorum* and *Zymoseptoria tritici*) in triticale and wheat leaves.

Materials and Methods

Applications are available for download at Github.com (https://github.com/SlawomirBartosiak/septoria-leafQR.git).

To download applications, click the link, and then select the clone or download button, then unpack the downloaded *.zip file preferably directly to the root directory on the C: drive or to any other folder, making sure that there are no Polish diacritical marks in the folder path, otherwise some of the commands that use paths with Polish diacritical marks will not run and the software may not work properly.

Hardware test platform:

Windows 10, 64-bit operating system; x64 Intel®Core™ i5–4590 CPU; 8 GB of 800 Mhz DDR3 RAM; Intel HD Graphics 4600; Seagate Barracuda 500 GB 7200 rpm 3.5" SATA III HDD (ST500DM002).

Experiment:

In the study, we used photographic documentation from experiments carried out in environmental chambers on cereal Septoria disease. Test subjects were sown in multi trays – 8 x 13 cells per pallet. The temperature in the environmental chamber was set to 22/20°C for 16/8 hours, and lights were on and off for 16 and 8 hours, respectively. Test subjects were inoculated 14 days after sowing with a water suspension of conidial *P. nodorum* spores at concentration of $4x10^{-6}$ spores ml^{-1}. Maximum relative air humidity was provided after the inoculation and for 10 days of the SNB disease symptom developing stage. The infected leaves of 3-week old wheat and triticale seedlings were adhered to a transparent adhesive film, labelled and photographed in a light box, while a second leaf was used to carry out the assessment. Selected varieties of wheat and triticale used in the experiment varied in susceptibility to *P. nodorum*.

Experiment photographic documentation preparation (Fig. 1.):

1. Label – comprising a QR code and the name of the test subject.
2. Cyan backdrop.
3. Diseased leaves attached and evenly spread on a flat surface.
4. Light box providing uniform lighting of the object, lit with a 5500K 2x 6055 Lm light source.
5. Camera or other optoelectronic device set perpendicularly to the photographed surface on a rigid tripod, taking pictures in the sRGB colour space in the resolution of at least 3072x2048 (180dpi), with 24-bit colour depth, set at f/4 aperture, with 1/250 s exposure time and ISO speed of ISO100, equipped with a lens with a focal length of 50 mm. All photographs were taken without flash.

Requirements:

- Windows 10;
- Python 3.8.2 (https://www.python.org/downloads/windows/) – The "Add Python 3.8 to PATH" option needs to be selected if the user intends to run programs with *.py file extension using Windows Command Line.
- The following Python modules were installed:

— pip 19.2.3
— opencv-python 4.2.0.32
— numpy 1.18.1
— pandas 1.0.3

— pyzbar 0.1.8
— pypng 0.0.20
— Pillow 7.1.2
— pyqrcode 1.2.1

lab_gener.py application:

In order to facilitate software reading of digital image labels, QR codes were used. The *lab_gener.py* application generates QR codes from objects listed within the *object_list.txt* text file, located in the main directory.

Preparing the application:

1. Prepare a reading list of labels located in the object_list.txt file located in the main septoria-leafQR directory. Every object code should be added in a new row, as follows:
   Object 1, Repetition 1
   Object 1, Repetition 2
   Object 2, Repetition 1
   etc.
2. Run the *lab_gener.py* application to generate QR code labels in the labels_output directory. To run the application, open Windows Command Line, go to the *septoria-leafQR* directory and type python *lab_gener.py*. User can also run the application by double-clicking the *lab_gener.py* file if Python3 is set to open files with *.py extension by default.
3. When the application displays the message: "Please enter [y] – yes, to add titles to output files or [n] – no, to save files without title", select whether to save or not to save the titles of the labels in the generated graphic files.
4. After selecting Y or N, the application starts running, displaying the names of the saved graphic files in the *labels_output* directory.
5. The *Finished* message will be displayed after successful execution of the application.

Print the generated labels, preferably on a cyan background, and place them on adhesive film (as described above) and create a digital photograph (Fig. 1).

sleaves.py application:

The application evaluates the severity of the disease in test subjects prepared as previously described and summarises the results in *.csv tables. The application looks for files for analysis in the *input_images* directory. QR code labels are automatically read by the software. In cases where the application is unable to read the QR code from the image, for example, due to blurriness, it will use the file name for the analysis. The application can also save graphic files with the disease symptoms and individual leaves. Moreover, users can modify parameters via HUE in HSV colour space to optimise selection of the diseased

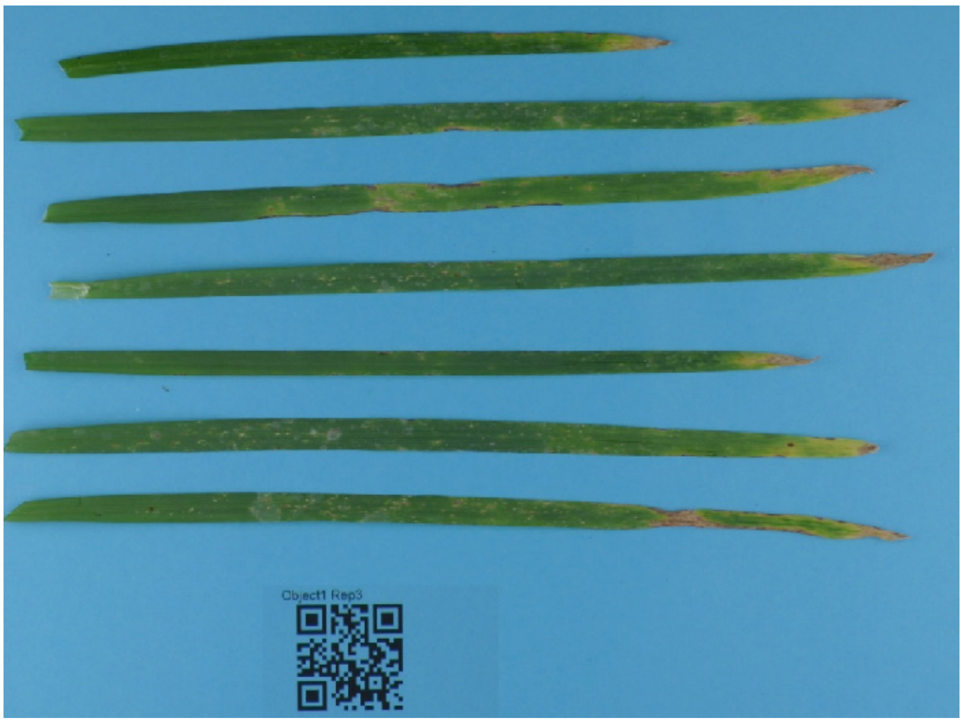

Fig. 1. Example of prepared image sample documentation to analyse.
Rys. 1. Przykład przygotowanej dokumentacji fotograficznej obiektu doświadczalnego.

area and total area of the leaves if the parameters of the equipment used for the analysis differ significantly from those described in this methodology, or if the software is used for another purpose. To run the application and optimise the parameters, follow the protocol below:

Parameter optimisation:

1. Copy image files prepared according to above methodology to the *input_images* directory.
2. To activate the saving of the extracted leaf contours, change the *save_leaf _im* variable in the *sleaves.py* file to *True*.
   save_leaf_im = True # True/False save an extracted leaf image
3. To activate the saving of the extracted diseased areas of the individual leaves, change the *save_diseased_im* variable in the *slaeaves.py* file to *True*.
   save_diseased_im = True # True/False save an diseased image
4. The application saves the extracted images to the *output_analysis* directory.
5. Run the application. In Windows Command Line, change the working directory to *septoria -leaf-QR* and type *python sleaves.py* or double click the application icon.
6. If the contours of the leaves are cut too much, increase the HUE* range using the *leaf_hue_min* and *leaf_hue_max* variables, and if too much of the background is left in the leaf contours, decrease the HUE* range – by default, the HUE variable range is set to 0–90.
   leaf_hue_min = 0 # Min leaf HUE, default 0
   leaf_hue_max = 90 # Max leaf HUE, default 90
7. If the application cuts out too much of the diseased tissue, increase the HUE* range using the *diseased_hue_min* and *diseased_hue_max variables*, and decrease it if the application leaves out healthy tissue.
   diseased_hue_min = 0 # Min diseased HUE, default 0
   diseased_hue_max = 45 # Max diseased HUE, default 45
8. After the parameters are optimised, run the application again to check if the results are correct.

**HUE is represented in HSV colour scale, calculated on the basis of the sRGB colour space. In Python Open CV library, hue can range from 0 to 179.*

Preparing the application:

1. If the prepared image files are located in the *input_images* directory, go to the *Septoria-leaf* directory using Windows Command Line and run the application by typing *sleaves.py*, or by double-clicking the application file.
2. The application will start the analysis, which takes about 20 seconds for 100 images in 3072x2048 resolution – the time depends on the available processing power and the number of analysed photographs.
3. After successful analysis, the *Finished* message will be displayed and analysis summaries in *.csv format will be created in the main application directory – *septoria-leaf.*

a. results.csv – a summary containing the output of individual leaf analysis:
- sample_name – name of test subject, read from its QR-code,
- leaf_area – area of the individual leaf in pixels,
- diseased_area – area of individual leaf diseased tissue in pixels,
- diseased_percent – (diseased_area/leaf_area) *100.

b. pivot_table.csv – pivot table of the summary of the analysis of individual test subjects, included in the *results.csv* file:
- sample_name – name of the test subject, read from its QR-code,
- leaf_area – sum of the areas of all leaves on a single test subject in pixels,
- diseased_area – sum of the diseased tissue of all leaves in pixels,
- diseased_percent – arithmetic average of disease severity based on each individual leaf in the picture of a given test subject.

Results and Discussion

The sleaves.py application will utilize all the resources of a modern CPU. Yet, the analysis of 130 images with a resolution of 3072x2048, using a mid-range computer (detailed specification of hardware test platform can be found in the Materials and Methods section) took only 21 seconds. After the leaf contour saving option was enabled, the analysis time increased to 40 seconds.

There were no statistically significant differences between the arithmetic average and weighted average approach (ANOVA, p=0.62) in the analysis of leaves inoculated by *Parastagonospora nodorum*. Some differences in descriptive statistics were, however, noted. The disease severity arithmetic average approach algorithm increased standard deviation by 0,7% and mean by 1.27%, in comparison to the weighted average approach. The largest average disease severity differences were observed in the case of test subjects where there were distinct

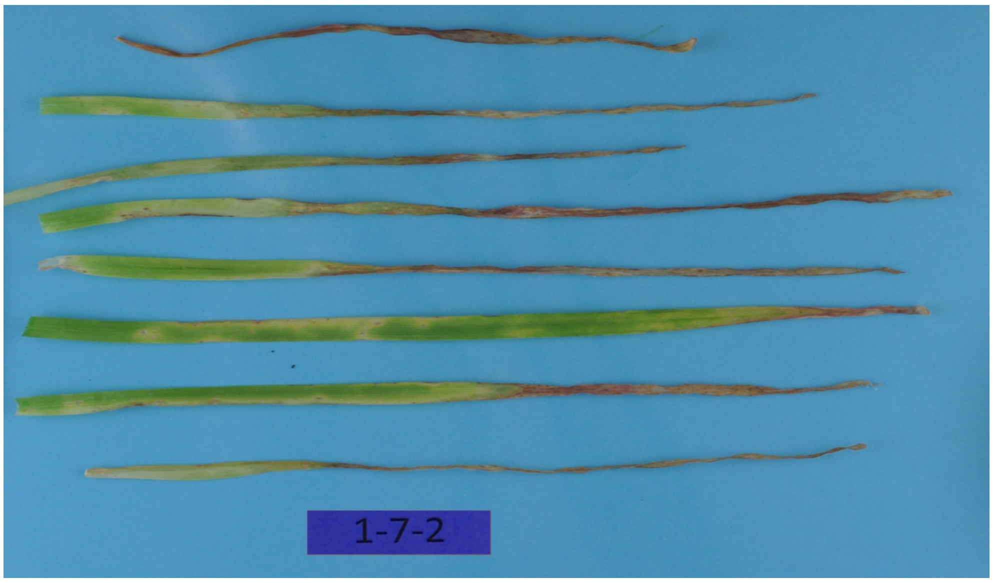

Fig. 2. Septoria disease symptoms caused by *P. nodorum* on second leaves of triticale seedlings.
Rys. 2. Objawy nekroz na drugim liściu siewek pszenżyta wywołane przez *P. nodorum*.

variances between the diseased area and the total leaf area. SNB disease causes a collapse of diseased tissue creating necrosis and chlorosis that can lead to shrinking of diseased leaf area, especially in the seedling growth stage (Fig. 2). This engenders a shrinkage of the diseased area when compared to the healthy area. In test subjects that had large and healthy leaves, the average is lowered by up to 12%. The data on the severity of each of the individual leaves enabled the removal of outliers, thus increasing the precision of the experiment (Fig. 2. – third leaf from the bottom lowers the average disease severity).

Parameter optimisation is required in order to make the application work properly. Undertaking this requires basic knowledge of the Python programming language and basic computer skills. Still, this should not be an obstacle for beginner users. The presented applications can be modified, customised and used for the analysis of other cereal diseases, as well as other plant species. What is more, the modification of the application code enables the extension of its feature set. It needs to be pointed out, however, that the presented application and the algorithms employed are unable to differentiate SNB disease from symptoms of abiotic stresses – such as nutrient deficiencies, as well as biotic stresses – such as other diseases that can also lead to tissue necrosis or chlorosis. Thus, the software is not perfectly suited for field trials. Therefore, it is recommended to use the software for experiments using inoculation or environmental chambers, where the risk of other undesirable biotic and abiotic stresses is minimised. The source code of the application is open and available for free, in line with the MIT license, which means that it can be freely modified and used. While commercial software solutions in general have more intuitive and easier to use interfaces – this, however, comes at an additional cost.

Conclusions

1. The sleaves.py application enables the analysis of the disease severity in each leaf and the elimination of outliers, thus reducing measurement errors.
2. The application permits analysis using the weighted average of the disease severity level.
3. There were no statistically significant differences between weighted average and arithmetic average-based disease severity analysis methods.
4. The application offers batch processing options.
5. The application can utilise all the processing power of modern multithreaded CPUs.

Literature

Easlon and Bloom, (2014). Easy leaf area: automated digital image analysis for rapid and accurate measurement of leaf area. Applications in Plant Sciences. 2 (7): 1400033.

Rueden, C. T., Schindelin, J., Hiner, M. C., DeZonia, B. E., Walter, A. E., Arena, E. T. and Eliceiri, K. W. (2017). ImageJ2: ImageJ for the next generation of scientific image data. BMC Bioinformatics 18:529.

Singh, V., Misra, A. K. (2017). Detection of plant leaf diseases using image segmentation and soft computing techniques. Information Processing in Agriculture (4) 41–49.

Thomas, S., Behmann, J., Steier, A., Kraska, T., Muller, O., Rascher, U. and Mahlein, A. K. (2018). Quantitative assessment of disease severity and rating of barley cultivars based on hyperspectral imaging in a non-invasive, automated phenotyping platform. Plant Methods 14:45.