MAGYAR AGRÁR- ÉS ÉLETTUDOMÁNYI EGYETEM

Columella

Volume 10, Number 1, 2023

Columella
Journal of Agricultural and Environmental Sciences

Monitoring of plant growth through methods of phenotyping and image analysis

Nezha KHARRAZ[1] – István SZABÓ[2]

1: Doctoral School of Mechanical Engineering – Hungarian University of Agriculture and Life Sciences, 2100 Gödöllő, Páter Károly u. 1., Hungary, e-mail: Nezha.Kharraz@phd.uni-mate.hu

2: Institute of Mechanical Engineering - Hungarian University of Agriculture and Life Sciences, 2100 Gödöllő, Páter Károly u. 1., Hungary

Abstract: With the rapid development of imaging technology, computing power, and algorithms, computer vision has revolutionized thoroughly plant phenotyping and is now a major tool for phenotypic analysis. Those reasons constructed the base for developing image-based plant phenotyping methods, it is a priority for the complementary or even alternative to the manual measurement. Nonetheless, the use of computer vision technology to analyze plant phenotypic traits can be affected by a lot of factors such as research environment, imaging system, and model selection. The field of plant phenotyping is developing rapidly at the moment. Image-based plant phenotyping has stated proven to be in precision agriculture, providing a quantitative basis for the description of plant-environment interactions.

Keywords: phenotyping, image-analysis, plants' growth, agricultural

Received 15 February 2023, Revised 20 June 2023, Accepted 20 June 2023

Introduction

Open-source image analysis software with plant phenotyping using computer vision is a key source in hydroponic cultivation. The former is performed in an outdoor and indoor installation or in a greenhouse under controlled conditions (Quigley et al., 2009). It may be carried out within a facility that includes providing light for plant growth, accessibility to the nutrient solution, and electrical power. The growing plants are set in a growth chamber and periodically soaked with nutrient solution. In addition, the hydroponic system gives the chance to control the entire growth chamber environment precisely. The hydroponic system is a modern technique of agriculture that is still under development. Recently, limited studies have been performed which suggested that hydroponics is performed without soil or any solid media; thus, the main observed problems to tackle the growth process. Zhou et al. (2016) conducted a study titled "ROSCC: An Efficient Remote Sensing Observation-Sharing Method Based on Cloud Computing for Soil Moisture Mapping in Precision Agriculture." This research was published in the IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. The paper focused on the development of a method called ROSCC, which utilizes cloud computing to enhance remote sensing observation-sharing for soil moisture mapping in precision agriculture. The authors present their findings and discuss the efficiency and effectiveness of the proposed method in accurately mapping soil moisture levels, which is crucial for optimizing agricultural practices and improving crop productivity.

The groundbreaking work of Castelló Ferrer et al. (2017) introduced the Personal Food Computer, a novel device designed for precise monitoring and management of environmental conditions to enhance crop cultivation. It was reported that the hydroponic system provides better control of plant growth and nutrient availability and prevents the plant from various diseases and root rot. However, during plant growth from sowing to harvest time, the method adopted in the hydroponic system requires expertise in domain knowledge of plants (Fox, 2006), environment control, and operations to maintain and control the growth of the plant.

In the Proceedings of the Future Technologies Conference (FTC) 2018, Ferrer et al. (2018) introduced the Personal Food Computer as an innovative device for controlled-environment-agriculture, revolutionizing food production methods.

Materials and Methods

An open-source image analysis software with an aimed package for plant phenotyping using computer vision was solicited for our experiment. The software is called PlantCV and is built based on modular functions, it is applicable to a wide range of plant types and different imaging systems, and it has multiple functions where the use of each one is centered on the context of an overall image-processing workflow. Nonetheless, the software is new and under continuous development where new functionalities are added on a regular basis. PlantCV currently supports the analysis of standard RGB color images, standard grayscale images, thermal infrared images, grayscale images from chlorophyll fluorescence imaging systems, and hyperspectral images. Support for additional image types is under development. The modular functions of which PlantCV is composed can be rearranged and adjusted quickly and easily. Workflows do not need to be linear. A global variable "debug" allows the user to print out the resulting image. The debug has three modes: either None, 'plot', or 'print'. If set to 'print' then the function prints the image out to a file, or if using a Jupyter notebook we could set debug to 'plot' to have the images plot to the screen. Debug mode allows users to visualize and optimize each step on individual test images and small test sets before workflows are deployed over whole datasets. In order to run the pant CV code, we need to previously have two required inputs:

1. **Image:** Plant CV process the Images regardless of what type of camera was used, particularly in our project, we used Raspberry PI camera. The processing works better if the images are of good lighting and the background's color is different from the plant's material.
2. **Output directory:** We need to select and name an output directory where the output images from each step will be saved.

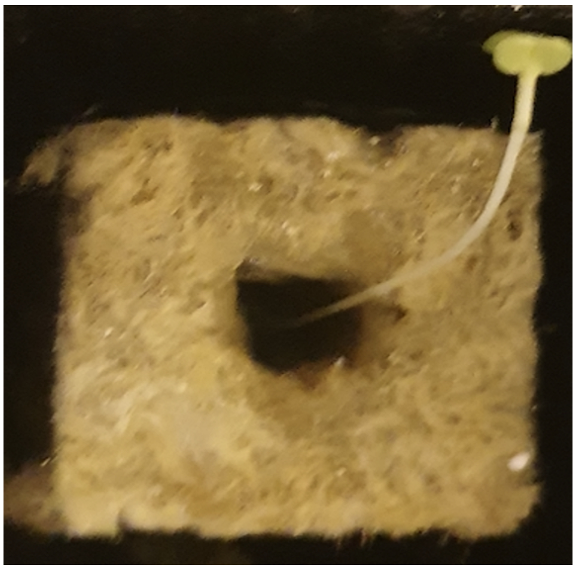

Figure 1: Input image

In the input image as shown in Figure 1, we have the input image. As we mentioned before, our particular image was captured by Raspberry PI camera, this means that Plant CV works on images not captured with specialized VIS image capture conditions. In our project, we used Plant CV to decompose the contours that constitute each plant. Our main interest was to use plant processing, therefore we needed to sort out contours and cluster them together in some way. In order to be able to arrive at the final result, which is color values. There are several steps that we need to follow until color values for each plant pixel are processed.

If the background is foreseeable, the program starts with defining the background. For our particular image, the program did some pre-masking of the background, in order to keep all plant information while removing the background. Thresholding is the simplest method of segmenting images, as it can be used for creating binary images. Generally, we need to select one of the color channels before performing a binary threshold on any image. The plant CV code as shown converts the RGB image to HSV color space and then extract the ’s’ or saturation channel, but if some of the plant’s information is missed or disappeared then the resulting channels may be combined in more steps.

The code below shows the first section of Plant CV code:

```
#!/usr /bin/env python

import os
import argparse
from plantcv import plantcv as
   ↪pcv

### Parse command-line
   ↪arguments
def options ():

parser = argparse.
   ↪ArgumentParser (
   ↪description="Imaging
   ↪processing with o p e n c y )
parser.add_argument ("-i", "
   ↪-image", help="Input image
   ↪ file.    , reguired=True)
parser.add_argument ("-o", "--
   ↪cutdir", help="Output
   ↪directory for image files.
   ↪    , required=False)
parser.add_argument ("-r", "--
   ↪r e s u l t , help=rresult
   ↪file.    , required=False)
parser.add argument ("-w", "-
   ↪  writeimg  ", help="write
   ↪out images.    , default=
   ↪False, action="store true"
   ↪)
```

The code starts by importing necessary packages of libraries, and by defining the inputs. One important thing before running the program is to make sure that plant CV is installed. We easily installed it from PyPI, by running the following command in the terminal as an administrator:

```
<pip install plantcv>.
```

The code of the main code is the following:

```
def main ():
# Get options
args = options ()
pcv.params.debug = args.debug #
   ↪ set debug mode
pcv.params.debug_outdir = args.
   ↪outdir # set output
   ↪directory
# Read image
img, path, filename = pcv.
   ↪readimage (filename=args.
   ↪image)
```

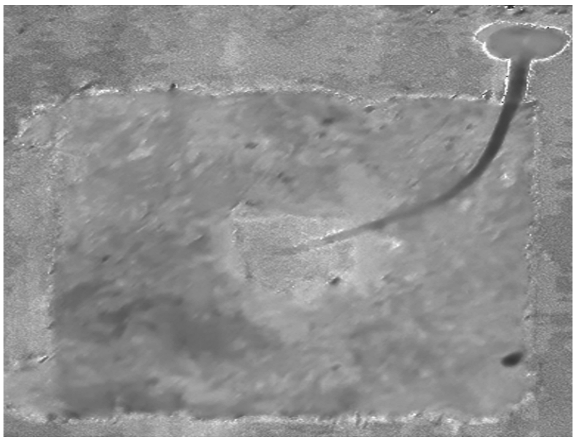
Figure 2: lab-blue-yellow

The returns are img, path, and image filename. The filename is the image file to be read which includes the path. The image presented in Figure 2 of the PlantCV research showcased a lab environment where the plant is visualized using a color representation of blue and yellow. The command reads the image into numpy ndarray(NumPy is an N-dimensional array type called ndarray.) and splits the path and image filename. This is a parameter of the OpenCV function imread.

In the following command, we convert RGB to HSV and extract the saturation channel.

The code to read the image is the following:

```
# Read image
img, path, filename = pcv.
   ↪readimage (filename=args.
   ↪image)
# Convert RGB to HSV and
   ↪extract the saturation
   ↪channel
== pcv.rgb2gray_hsv (rgb_img=
   ↪img, channel='s')
```

The HSV color space has 3 channels: ‘h’ the Hue, ‘s’ the Saturation, and the’v’ Value, or intensity. The Hue channel represents the "color".

The code of the saturation channel from original RGB image converted to HSV color space:

```
# Convert RGB to LAB and
   ↪extract the Blue channel
b = pcv.rgb2gray_lab (rgb_img=
   ↪img, channel='b')
# Threshold the blue image
b_thresh = pcv.threshold.binary
(gray_img = b, threshold-160,
   ↪max_value-255, object_type
   ↪='light')
b_cnt = pcv.threshold.binary
(gray_img=b, threshold-160,
   ↪max_value=255, object_type
   ↪='light')
```

In order to threshold the saturation image, we apply the following code:

```
# Threshold the saturation
   ↪image
s_thresh = pcv.threshold.binary
   ↪ (gray_img=s, threshold
   ↪-85, max_value=255,
   ↪object_type='light')
```

The saturation channel is thresholded, which means the code created a binary image from a gray image based on the threshold values. Our object in this project was the plant which is light. The threshold creates a binary image from the gray image based on the threshold values that can be adjusted depending on the quality of our image, in our case used 85 for the threshold and 255 pixels as a maximum value.

In the next step, again depending on the level of lighting, the code can be modified to better manipulate the background. The original image is converted from an RGB image to LAB color space and the blue-yellow channel is extracted.

The image as shown in figure 3 is then again thresholded and there is an optional fill step that the code runs or not depending on the image, in our case, it was not needed.

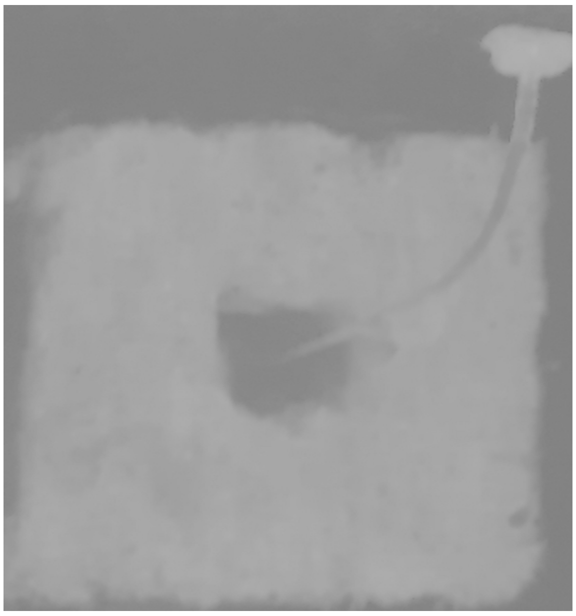
Figure 3: Blue-yellow channel from LAB color space from the original image.

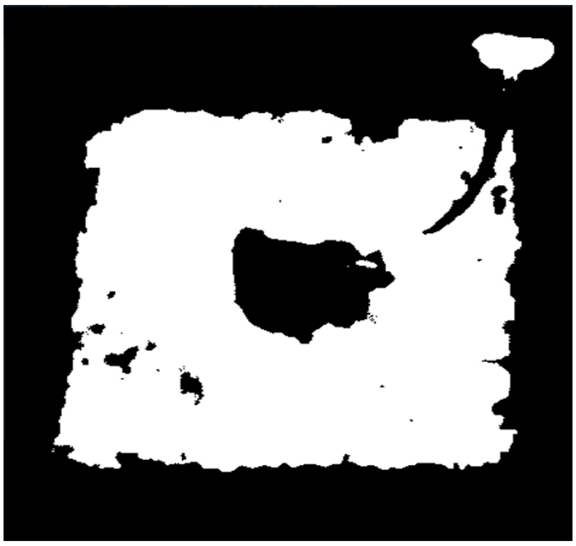

Figure 4: Binary Threshold blue-yellow channel image.

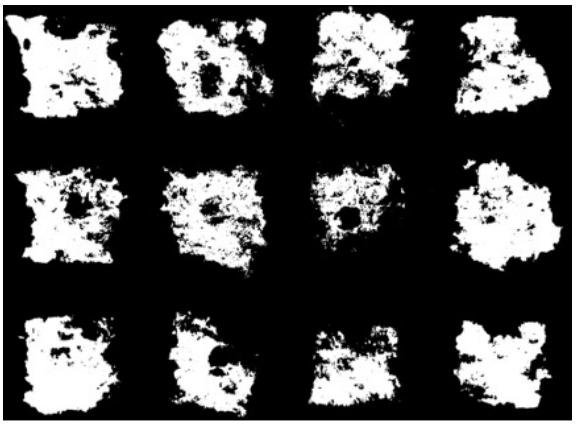

Figure 5: binary_threshold_128

After this step, the resulted in the binary image as in figure 4 was applied as an image mask over the original image, to remove as much background with simple thresholding in figure 5 without missing out any plant material.

The following code is for masked image with background removed.

```
# Apply Mask (for VIS images,
   ↪mask_color=white)
masked = pcv.apply_mask(img=img
   ↪, mask=bs, mask_color='
   ↪white')
```

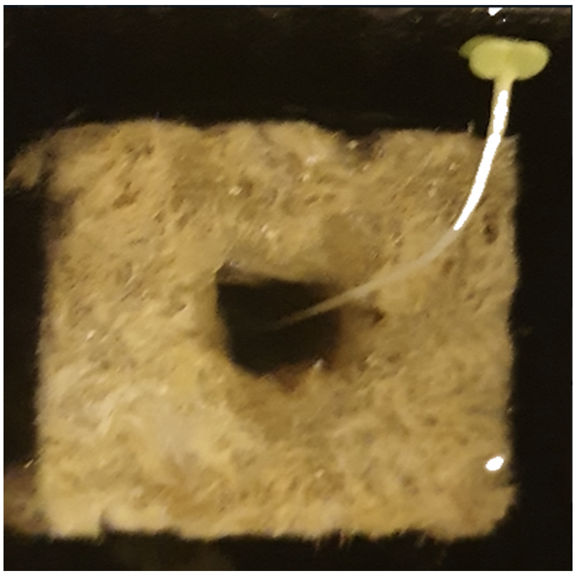

Figure 6: Masked image with the background removed

After getting the masked image in figure 6 with the background slightly removed. The next step from the code is capturing the plant in the masked image from the last figure. The masked green-magenta and blue-yellow channels are taken out. The two channels are thresholded to show different sections of the plant.

The small objects are filled. The image taken has very green leaves, but often (especially with stress treatments) there are yellowing leaves, red leaves, or regions of necrosis. The different thresholding channels capture different regions of the plant, then are combined into a mask for the image that was previously masked. In the following the applied code of RGB to LAB conversion code and the resulted image is shown in figure 7:

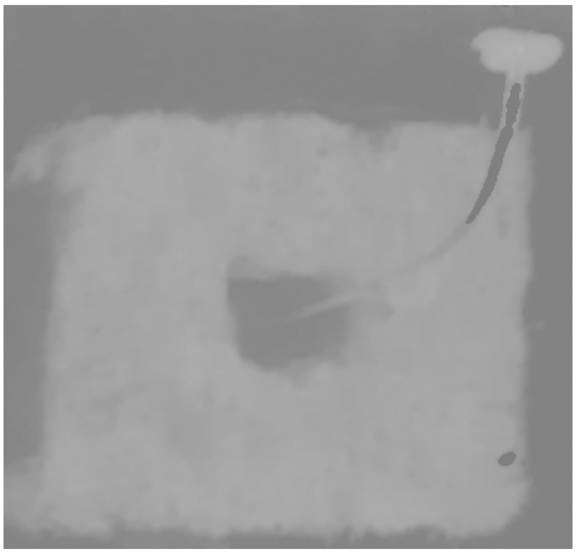

Figure 7: Thresholded LAB channel image

```
# Convert RGB to LAB and
   ↪extract the Green-Magenta
   ↪and Blue-Yellow channels
masked_a = pcv.rgb2gray_lab(
   ↪rgb_img=masked, channel='a
   ↪')
masked_b = pcv.rgb2gray_lab(
   ↪rgb_img=masked, channel='b
   ↪')
# Threshold the green-magenta
   ↪and blue images
maskeda_thresh = pcv.threshold.
   ↪binary(gray_img=masked_a,
   ↪threshold=115, max_value
   ↪=255, object_type='dark')
maskeda_threshl = pcv.threshold
   ↪.binary(gray_img=masked_a,
   ↪ threshold=135, max_value
   ↪=255, object_type='light')
masked_thresh = pcv.threshold.
   ↪binary(gray_img=masked_b,
   ↪threshold=128, max_value
   ↪=255, object_type='light')

# Join the thresholded
   ↪saturation and blue-yellow
   ↪ images (OR)
abl = pcv.logical_or(bin_img1=
   ↪maskeda_thresh, bin_img2=
   ↪masked_thresh)
ab = pcv.logical_or(bin_img1=
   ↪maskeda_threshl, bin_img2=
   ↪abl)

# Fill small objects
ab_fill = pcv.fill(bin_img=ab,
   ↪size=200)

# Apply mask (for VIS images,
   ↪mask color=white)
masked2 = pcv.apply_mask(img=
   ↪masked, mask=ab_fill,
   ↪mask_color='white')
```

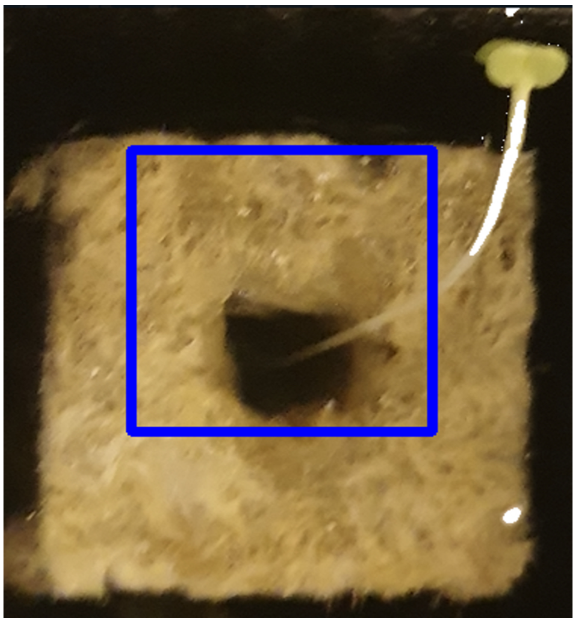

Figure 8: Region of interest drawn into

For the identification process within the image, Plant CV is based on contours in OpenCV. A rectangular region of interest is determined as in figure 8.
The code *defining region of interest:*

```
#Object combine: kept objects
obj, mask = pcv.
   ↪object_composition(img=img
   ↪, contours=roi_objects,
   ↪hierarchy=hierarchy3)
```

Once the region of interest has resulted, the code keeps then the rest of the image overlapping with the region inside the rectangle or cuts the objects to the shape of the region of interest. In our particular case, as seen in the following image. The isolated items now should all be plant material that we are interested in. There can be more than one object that makes up the material of the plant, sometimes leaves appear in the images as separate objects.

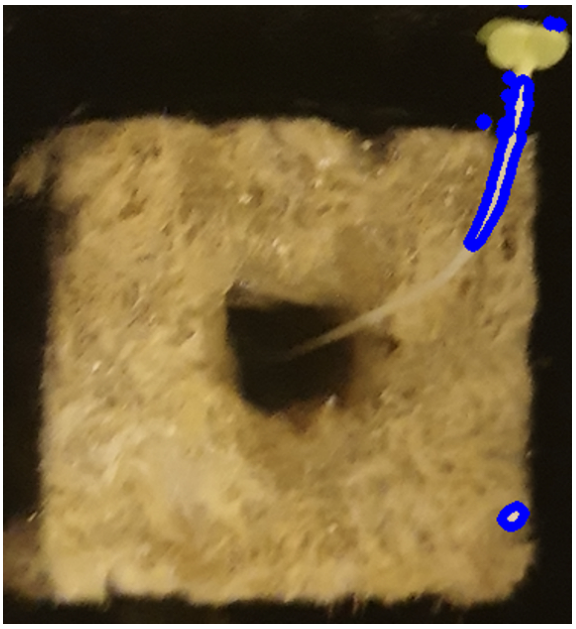
Figure 9: Outline (blue) of combined objects on the image.

The image in figure 9 represents an outline of combined objects, indicated by the blue lines, overlaid on the original image. The combined objects are a result of the previous image processing and analysis technique applied through plantCV to the image data. This outline helps us identify and visualize the boundaries or contours of the objects detected in the image, providing insights into their spatial distribution.

The final step of the image analysis is to examine the plant material such as horizontal height, shape, or color.

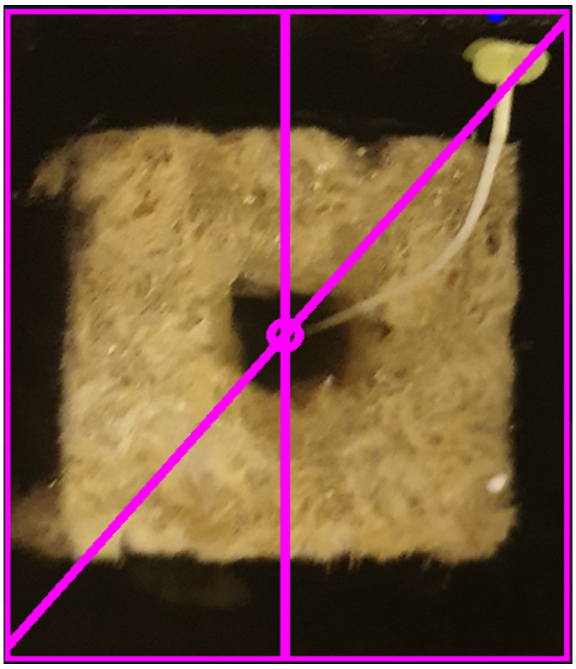
Figure 10: Shape analysis output image.

The image in figure 10 represents the output of shape analysis applied to an image. It typically includes visualizations or graphical representations that provide information about the shapes of objects present in the image. This includes the features as the contour lines and bounding lines that help characterize the shapes and structures within the image. The shape analysis output image serves as a visual representation of the results obtained from analyzing the shapes of objects, aiding in understanding the spatial organization and properties of the objects in the image.

Discussion

Conducting a comprehensive analysis of a plant's characteristics, encompassing factors such as its horizontal height, shape, and colour, yields invaluable insights that can be leveraged in advanced stages of plant recognition and understanding its growth trajectory. By scrutinizing these plant materials, we acquire a wealth of useful information that can be harnessed for diverse purposes, including not only recognizing the plant itself but also elucidating the various developmental steps it undergoes.

In the publication "OpenAG: A Globally Distributed Network of Food Computing" by Harper and Siller (2015), featured in IEEE Pervasive Computing, the concept of a globally distributed network of food computing is introduced. This innovative approach leverages advanced technologies, such as PlantCV and pervasive computing, to revolutionize food production and address challenges in agriculture (Harper & Siller, 2015). The utilization of PlantCV software has revolutionized our ability to analyse images captured through cutting-edge devices like Raspberry PI. This software package, specifically designed for plant phenotyping and the application of phenomics technologies, including PlantCV and image analysis, offers promising solutions to overcome the challenges associated with phenotyping and enhance our understanding of plant characteristics and growth processes (Furbank & Tester, 2011), empowers researchers by providing a cohesive programming and documentation interface. Through PlantCV, an extensive collection of image analysis techniques sourced from diverse packages and algorithms seamlessly converge, facilitating an integrated and comprehensive approach to image analysis.

Exploring the vast potential of an Internet of Things (IoT)-based growing chamber system in tandem with Global Positioning System (GPS) technology represents a fertile area of research. The synergistic amalgamation of these two domains opens up new horizons in plant cultivation. By harnessing the capabilities of IoT, we can accurately detect and map the geographical locations of green growing chamber users across the globe. This spatial awareness not only enables us to identify clusters of green growing chambers but also facilitates the gathering of vital image analysis data. Such data, once recorded and curated within an application, can be effortlessly shared among users, fostering a community of practice where individuals can exchange invaluable insights and best practices pertaining to optimal plant growth. The study conducted by Abdullah et al. (2016) exemplifies the tremendous potential of this approach in enhancing plant cultivation methodologies and promoting sustainable practices.

We can say that delving into the minute details of a plant's material composition, encompassing attributes such as its horizontal height, shape, and colour, provides a rich tapestry of information that can be instrumental in various stages of plant understanding and recognition. With the aid of sophisticated software packages like PlantCV, researchers can harness the power of image analysis techniques, seamlessly integrated from multiple sources, to unlock new insights and correlations. Moreover, the advent of IoT-based growing chamber systems coupled with GPS technology holds tremendous promise for the future of plant cultivation, facilitating global collaboration, knowledge sharing, and the adoption of sustainable practices.

In our experimer, as mentioned earlier we followed several processes to get a pseudo-colored image as figure 11 of our plant and we were interested in calculating the Excess Green Index (ExG) for a specific pixel within the image. This index serves as a valuable metric for assessing the vegetation's health and stress levels. To illustrate the calculation process, let's consider our scenario where we have a dataset comprising 9 distinct plants(9 rockwool cubes), and our objective is to evaluate the vegetation quality for each individual plant. To achieve this, we need to perform the Excess Green Index calculations separately for each plant, utilizing their respective RGB images.

To calculate the Excess Green Index (ExG) for each plant, we have followed these steps:

1. We began by converting the RGB image of

each plant into the ExG index using the following formula: ExG $= 2 \times$ Green $-$ Red $-$ Blue

By substituting the corresponding pixel values of the Green (G), Red (R), and Blue (B) channels into the equation, we could obtain the ExG index value for each pixel within the plant region of interest.

2. Once the ExG values are computed for all pixels within the plant region of interest, we calculated the average ExG value for that specific plant. This was accomplished by taking the mean of all the ExG values obtained from the pixels within the plant's region.

3. Finally, we compared the average ExG value of each plant to a predetermined threshold or range established for evaluating vegetation health. This threshold serves as a benchmark against which the vegetation quality of each individual plant can be assessed. By analyzing the ExG values in relation to the threshold, we could identify variations or differences in the health of the individual plants.

By performing these calculations for each plant, we gained the ability to assess the vegetation quality on an individual basis. A comprehensive understanding of the characteristics and health status of each plant is achieved. This approach allows for targeted analysis and evaluation of vegetation attributes, as opposed to treating the plants as a collective group.

The Excess Green Index (ExG) serves as a valuable vegetation index, quantifying the excess amount of green color present in an image. By leveraging this index, researchers and practitioners can gain insights into vegetation health and stress levels. It provides a quantitative measure that reflects the abundance of green color in relation to the red and blue channels, offering valuable information about the density and vigor of the plant material.

By incorporating the Excess Green Index into our analysis, we made informed decisions regarding plant health, stress management, and overall vegetation quality. Here's an example of how we use the Excess Green Index:

1. We obtained an RGB image of the plant we wanted to analyze.
2. We convert the RGB image to the LAB color space. This was done using the image processing libraries of PlantCV mentioned earlier.
3. We extracted the individual channels from the LAB image: L (lightness), a (green-magenta), and b (blue yellow).
4. We calculated the Excess Green Index (ExG) using the previously stated formula.
5. We applied a threshold to the ExG image to segment regions with high vegetation density. This was done by selecting an appropriate threshold value to distinguish between healthy vegetation and other objects or background, thanks to the palntCV opensource libraries, this step was not challenging.
6. We analysed the segmented regions to extract relevant information and performed further computations based on our specific objective.

By utilizing the Excess Green Index, we could identify and analyze areas of interest in the image that correspond to healthy vegetation based on their green color characteristics.

1. RGB values for the pixel of interest:
   Red (R) $= 100$
   Green (G) $= 150$
   Blue (B) $= 80$
2. We converted the RGB values to a range of 0-1 by dividing each value by 255:
   $R = 100/255 = 0.392$
   $G = 150/255 = 0.588$
   $B = 80/255 = 0.314$
3. We calculated the Excess Green Index (ExG):
   $\text{ExG} = 2 \times G - R - B$

$$= 2 \times 0.588 - 0.392 - 0.314$$
$$= 1.176 - 0.392 - 0.314$$
$$= 0.47$$

The Excess Green Index (ExG) for the given pixel is 0.47. This value represents the excess amount of green color in relation to the red and blue channels. We performed similar calculations for other pixels in the image to obtain their corresponding ExG values.

The Excess Green Index (ExG) itself does not provide a direct measure of vegetation health or quality. Instead, it quantifies the relative amount of green color present in an image compared to the red and blue channels. It can be used as a vegetation index to assess the density or presence of vegetation in an image.

To determine whether the vegetation is good or not based on the ExG value, we would typically need to establish a threshold or range specific to experiment. This threshold can be determined through experimentation, field observations, or by comparing the ExG values of known healthy and unhealthy vegetation samples.

For example, in the conducted experiment, we determined that vegetation with an ExG value above 0.5 is considered good, and this way we could compare the calculated ExG value of a pixel or region to this threshold. When the ExG value was above the threshold, we classified it as healthy vegetation. Conversely, when the ExG value was below the threshold, we indicated less healthy or sparse vegetation.

It's important to note that the interpretation of vegetation health based on an index like ExG can vary depending on the specific context, plant species, environmental conditions, and other factors. Therefore, it is recommended to validate and calibrate the threshold values based on our specific application and domain knowledge.

In conclusion, the Excess Green Index (ExG) and PlantCV software play a significant role in the monitoring of plant growth through advanced methods of phenotyping and image analysis. The ExG serves as a valuable vegetation index, providing insights into the health, stress, and density of plants based on their green color characteristics. By calculating the ExG for specific pixels or regions within RGB images of plants, researchers and practitioners can assess vegetation quality on an individual plant basis.

The PlantCV software, specifically designed for plant phenotyping, enables the analysis of images captured through various platforms like Raspberry Pi. It provides a comprehensive collection of image analysis techniques, integrating algorithms from different source packages. This software offers a common programming interface and documentation, simplifying the implementation of image analysis procedures for plant studies.

By utilizing PlantCV and the ExG index, researchers can extract useful information from plant materials such as height, shape, and color, aiding in plant recognition and understanding growth stages. This integration of advanced imaging techniques with phenotyping methods allows for non-destructive and high-throughput analysis of plant traits. In line with the Food and Agriculture Organization of the United Nations (FAO) report on 'The state of food and agriculture: Climate change, agriculture, and food security' (2016), the utilization of Excess Green Index (ExG), PlantCV, and image analysis techniques play a pivotal role in monitoring plant growth and addressing the challenges of climate change in the agricultural sector (FAO, 2016).

Moreover, the combination of IoT-based growing chambers and GPS systems presents an exciting area of research. By tracking the locations of green growing chamber users on world maps, it becomes possible to collect and share recorded image analysis data. This exchange of information facilitates thedissemination of good practices for growing plants, contributing to advancements in the field of plant science.

In summary, the utilization of the Excess Green Index, along with PlantCV software and emerging technologies, enables efficient monitoring of plant growth. By employing image analysis techniques and phenotyping methods, researchers can extract valuable insights regarding vegetation health, stress levels, and growth patterns. This holistic approach to plant analysis fosters advancements in agricultural practices, crop breeding, and the understanding of plant physiology, ultimately leading to improved plant productivity and sustainable agriculture.

Acknowledgments

This project started during my bachelor's degree a few years ago at MATE University (Former Szent Istvan University). I am grateful to MATE University for welcoming foreign students and awarding them scholarship opportunities. I would like to thank particularly Dr. Szabo Istvan for being my supervisor and helping me through this academic journey since my bachelor's degree.

References

Abdullah, A., Al Enazi, S., & Damaj, I. (2016). AgriSys: A smart and ubiquitous controlled-environment agriculture system. In 2016 3rd mec international conference on big data and smart city (icbdsc) (p. 1-6). doi: 10.1109/ICBDSC.2016.7460386

Castelló Ferrer, E., Rye, J., Brander, G., Savas, T., Chambers, D., England, H., & Harper, C. (2017). Personal Food Computer: A new device for controlled-environment agriculture. MIT Media Lab Open Agriculture Initiative (OpenAg) (), . Retrieved from http://hdl.handle.net/1721.1/110010

FAO. (2016). The state of food and agriculture: Climate change agriculture and food security. Rome, Italy: Food and Agriculture Organization of the United Nations.

Ferrer, E. C., Rye, J., Brander, G., Savas, T., Chambers, D., England, H., & Harper, C. (2018). Personal Food Computer: A New Device for Controlled-Environment Agriculture. In Proceedings of the future technologies conference (FTC) 2018 (p. 1077-1096). Springer International Publishing. doi: 10.1007/978-3-030-02683-7_79

Fox, J. L. (2006). Turning plants into protein factories. Nature Biotechnology **24**(10), 1191-1193. doi: 10.1038/nbt1006-1191

Furbank, R. T., & Tester, M. (2011). Phenomics – technologies to relieve the phenotyping bottleneck. Trends in Plant Science **16**(12), 635-644. doi: 10.1016/j.tplants.2011.09.005

Harper, C., & Siller, M. (2015). OpenAG: A Globally Distributed Network of Food Computing. IEEE Pervasive Computing **14**(4), 24-27. doi: 10.1109/MPRV.2015.72

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., … Ng, A. Y. (2009). ROS: an open-source Robot Operating System. In Icra workshop on open source software (Vol. 3, p. 5).

Zhou, L., Chen, N., Chen, Z., & Xing, C. (2016). ROSCC: An Efficient Remote Sensing Observation-Sharing Method Based on Cloud Computing for Soil Moisture Mapping in Precision Agriculture. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing **9**(12), 5588-5598. doi: 10.1109/JSTARS.2016.2574810