Materials and Methods 1

Software 1.1

Image analysis and data extraction were completed through a Python script running on Python v3.12.4, with the following packages: PlantCV v4.2.1, OpenCV v4.10.0.84, natsort v8.4.0, NumPy v1.26.4, XlsxWriter v3.2.0, along with modules from the Python Standard Library. A custom graphical user interface (GUI) was made using CustomTkinter v5.2.2. Data plots were created using Microsoft Excel v2406. All images used were captured at a resolution of 640x480, yielding a total of 307,200 pixels per image.

Light Intensity Normalization 1.2

To eliminate discrepancies in the semantic segmentation process, a method of image classification where every pixel is assigned a class or in our case, assigned to be part of a plant or not, caused by shadows or over-exposure on the input images, the light intensity values of each image were normalized. Input images are converted from the RGB model (Fig. 3), a color model in which the image's red, green, and blue components are separated into three channels, to the YCbCr model (Fig. 4), a color model in which the luminance, blue chrominance, and red chrominance are split into three channels. The luminance channel, Y, is then separated from the chrominance channels, Cb and Cr (Fig. 5). Let I be a column vector representing the width and height of the input image.

$$I = \begin{bmatrix} Width \\ Height \end{bmatrix} \tag{1}$$

Consider the matrix Y, which contains the luminance data for every pixel in the input image:

$$Y = \begin{bmatrix} Y_{1,1} & Y_{1,2} & Y_{1,3} & \dots & Y_{1,I_{\text{Width}}} \\ Y_{2,1} & Y_{2,2} & Y_{2,3} & \dots & Y_{2,I_{\text{Width}}} \\ Y_{3,1} & Y_{3,2} & Y_{3,3} & \dots & Y_{3,I_{\text{Width}}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ Y_{I_{\text{Height}},1} & Y_{I_{\text{Height}},2} & Y_{I_{\text{Height}},3} & \dots & Y_{I_{\text{Height}},I_{\text{Width}}} \end{bmatrix}^{I_{\text{Height}} \text{x} I_{\text{Width}}} \tag{2}$$

Let $Y(i, j)$ denote the element in the i-th row and j-th column of matrix Y. To normalize the light intensity, we utilize a Difference of Gaussians

approach. This involves applying a Gaussian blur to the image, in our case Y, and subtracting the resultant matrix which contains the lower-frequency blurred luminance values from the original image. We derive an approximate kernel for the convolution from Pascal's triangle

$$\begin{array}{ccccccccc} & & & & 1 & & & & \\ & & & 1 & & 1 & & & \\ & & 1 & & 2 & & 1 & & \\ & 1 & & 3 & & 3 & & 1 & \\ 1 & & 4 & & 6 & & 4 & & 1 \end{array}$$

Then we perform element-wise multiplication

$$\begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 1 & 4 & 6 & 4 & 1 \\ 1 & 4 & 6 & 4 & 1 \\ 1 & 4 & 6 & 4 & 1 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{3}$$

Resulting in the kernel ω

$$\omega = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{4}$$

$$\text{Given that } \sum_{i=1}^{5} \sum_{j=1}^{5} \omega(i,j) = 256 \tag{5}$$

However, we want our Gaussian kernel's sum to normalize our kernel's sum to 1 in order to preserve the original brightness of the image, thus the approximate Gaussian Kernel ω is defined as

$$\omega = \frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{6}$$

For the resultant matrix of the convolution between ω and Y have the same dimensions as Y we must implement some sort of edge handling. We chose to pad the input matrix with the following paddings

$$\text{pad}_x = \left\lfloor \frac{\omega_w - 1}{2} \right\rfloor \tag{7}$$

$$\text{pad}_y = \left\lfloor \frac{\omega_h - 1}{2} \right\rfloor \tag{8}$$

$$\text{pad}_x = \left\lfloor \frac{5 - 1}{2} \right\rfloor = 2 \tag{9}$$

$$\text{pad}_y = \left\lfloor \frac{5 - 1}{2} \right\rfloor = 2 \tag{10}$$

Let Y_p be the padded version of Y, where pad_x and pad_y are added to all borders of Y, where the padding is populated with values of 0. The Gaussian blurred matrix $R(x, y)$ is then defined as

$$R(x, y) = (\omega * Y_p)(x, y) = \sum_{i=1}^{5} \sum_{j=1}^{5} \omega(i, j) \cdot Y_p(x - i, y - j) \tag{11}$$

Where x and y correspond to an element in Y, and ω has the indices $[1, 5]$. We remove the padding surrounding R to match the dimensions of matrix Y. Next we compute the Difference of Gaussians by subtracting the resultant matrix of the convolution, R, from the original matrix, Y.

$$Y = Y - R \tag{12}$$

And after performing the Difference of Gaussians, we are left with a grayscale image whose foreground features have been enhanced. Next we perform scalar addition and add 100 to each element in the matrix R in order to increase the brightness for a clearer separation of the background and the subject of the image.

$$Y = Y + 100 \tag{13}$$

Now that the normalization has been performed, we merge all three color channels, Y, Cr, and Cb, back together, and convert the image from the color model YCrCb to RGB. Resulting in the final normalized image (Fig. 6).

Image Processing 1.3

Our analysis was strongly supplemented and improved by the use of PlantCV, an open-source Python library for phenotyping plants. PlantCV's modular capability allows for a workflow to be easily created for converting images from RGB to binary. First, we employed the use of rgb2gray_lab method to convert the normalized images to the CIELAB color model. The 'A' channel of the CIELAB image, which contains all green-magenta values, is then isolated (Fig. 7). This channel provides the clearest, sharpest, and most accurate segmentation of the plant from the image's background, compared to the "L" or "B" channels. The next step in the process is to convert the image into a binary image. This is facilitated by means of the thresholding.otsu method, in which the specified object type was "dark" (Fig. 8).

The closing operation uses a specified kernel K, defined as

$$K = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \tag{14}$$

The closing operation was performed on the binary mask of the plant using the kernel K (Fig. 7). The kernel K is convolved with the binary mask to suppress dark noise and refine the segmentation results. The end result is a matrix B which contains all pixel color values, since the image is now binary, the values are either 0, which indicates a pixel not part of the plant class, or 255, which is part of the plant class.

$$B = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}^{I_{\text{Height}} \text{x} I_{\text{Width}}} \tag{15}$$

An opening operation was not needed as each plant's container was covered with black landscaping fabric, as well as the bed the plants were placed in. These precautions helped to mitigate and heavily reduced any noise that may have occured around the plant. PlantCV played an enormous role in facilitating an efficient and elegant method for image processing.

Data Extraction 1.4

We extracted data regarding both the motion as well as the size of the plant. To save on resources and time, if B_i, where B is the matrix of binary pixel data and i is the image index, contains fewer than 500 elements with the value 255, we assume this image does not contain a plant. To compute the motion of a plant, we subtract the previous image's matrix from the current image's matrix:

$$\text{Motion}(B_i):\{i \mid i > 1\} = |S(B_i) - S(B_{i-1})| \tag{16}$$

where $S(B_i)$ is defined as:

$$S(B_i) = \sum_{j=1}^{I_{\text{Height}}} \sum_{k=1}^{I_{\text{Width}}} \begin{cases} B_i(j,k), & \text{if } B_i(j,k) = 255, \\ 0, & \text{otherwise.} \end{cases}$$

Here, $S(B_i) > 0$ if there exist pixels in B_i with intensity 255. This allows us to detect significant changes between consecutive images, indicating motion in the plant. This is repeated for every image in the experiment, yeilding a 1 dimensional matrix containing the pixel sums for each image. Let I_P be the matrix

$$I_P = \begin{bmatrix} S(B_1), & S(B_2), & S(B_3), & \ldots, & S(B_{\text{NumImages}}) \end{bmatrix} \tag{17}$$

We write each element of I_P to an excel workbook by means of XlsxWriter, yeilding the following figures

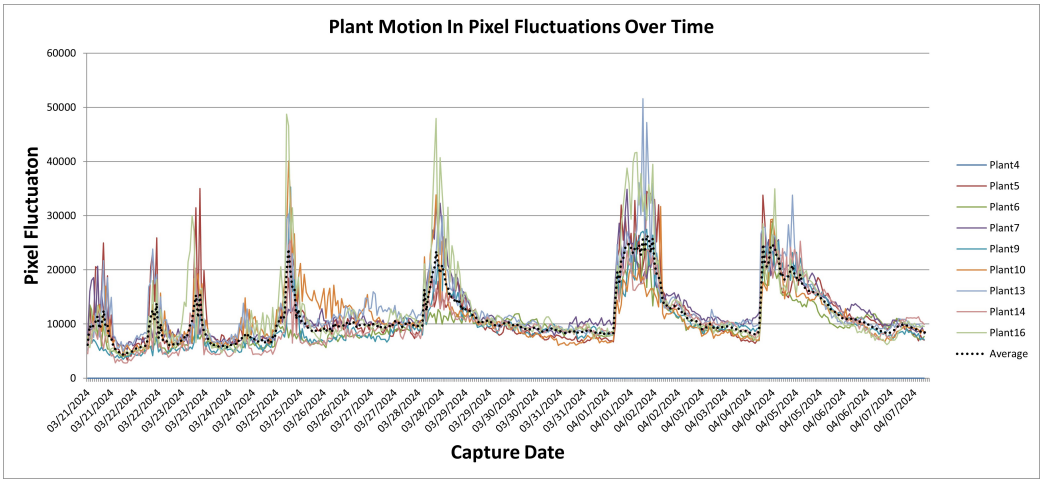

Figure 1: Plant Motion

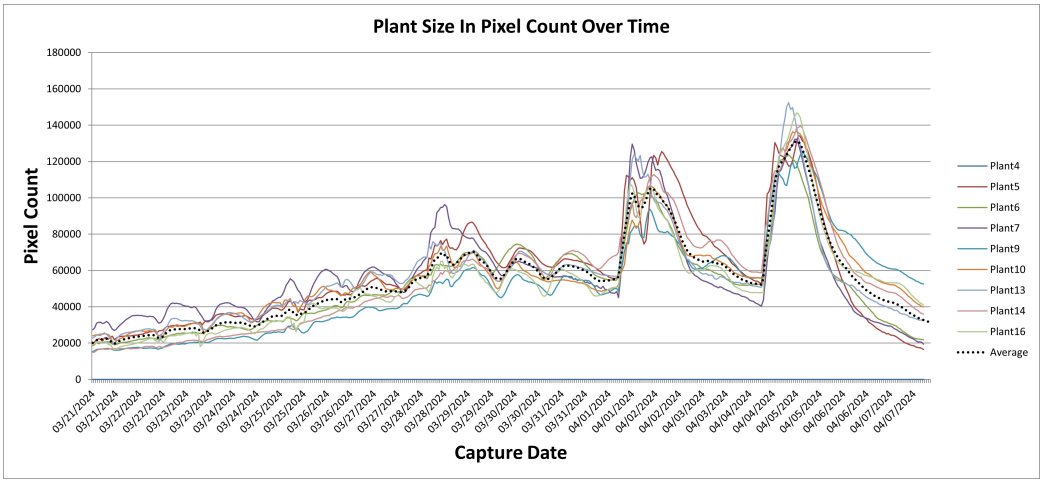

Figure 2: Plant Size

Figure 3: Species Name in RGB

Figure 4: Species Name in YCrCb

Figure 5: Isolated Y channel of YCrCb

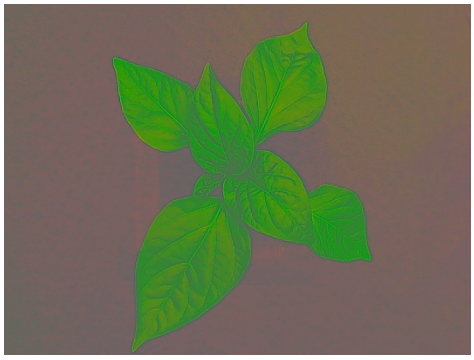

Figure 6: Normalized Light Intensity

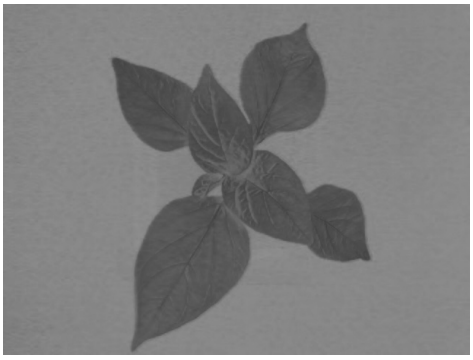

Figure 7: Isolated A channel of CIELAB

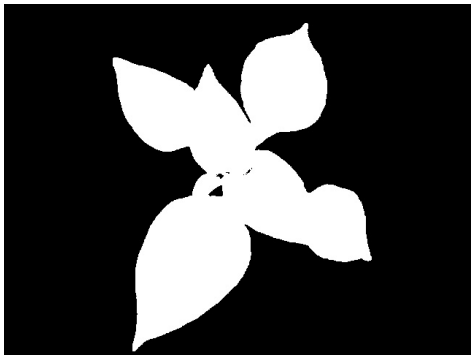

Figure 8: Species Name after otsu thresholding

-

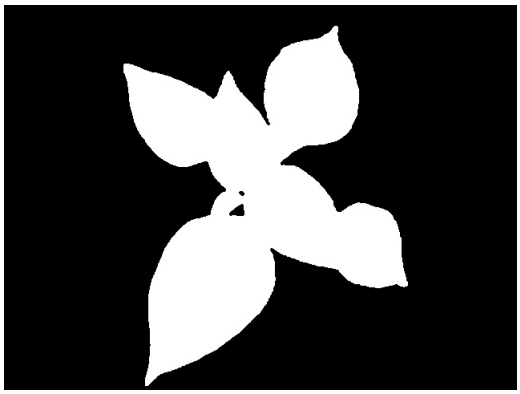

Figure 9: Species Name after closing operation