Entry to the Stockholm Junior Water Prize 2024

Czech Republic

Root Tracker

Tadeáš Fryčák

Gymnázium Jana Opletala,
Litovel, Czech Republic

Acknowledgements

I would like to thank my colleagues, Mgr. Jan Humplík, Ph.D. and RNDr. Tomáš Fürst, Ph.D., for the opportunity to participate in this research and for providing valuable advice. I would also like to thank Mgr. Jan Humplík, Ph.D. for comprehensive and long-term testing of Root Tracker on thousands of root system images.

List of abbreviations

Abbreviation	**Meaning**
DLSR	Digital Single-Lens Reflex
SW	Software
ROI	Region of Interest
LED	Light-Emitting Diode
SNR	Signal-to-Noise Ratio
RGR	Relative growth rate
HSV	Hue, Saturation, Value
PETG	Polyethylene terephthalate glycol
CSV	Comma-separated values

Abstract

Scientists around the world are studying the effects of various chemicals on root systems under increased stress conditions – for example, the increasingly frequent long-term droughts. Plants defend themselves against drought by various mechanisms, such as storing water in their root systems. The bulkier and more extensive a plant's root system, the more water it can retain, increasing its chances of survival during prolonged droughts. Therefore, the researchers' work aims to identify a chemical that significantly stimulates root growth.

In order to determine the effect of the chemicals studied on root systems, it is necessary to quantify the extent of the root system. Researchers therefore grow plants, for example, in transparent nutrient media in *in vitro* modules (petri dishes hermetically isolating the internal environment from the external environment) or using so-called rhizotrons (devices filled with a substrate with one transparent side for observing the roots).

Since the preparation of a single *in vitro* module is complicated and economically demanding in large numbers, researchers plant, for example, 6-12 plants per module. Due to the limited size of the petri dish, from a certain point in time, individual plant root systems inevitably start to cross until they later reach a state where it is complicated to even observe them manually, let alone measure and evaluate them.

Therefore, I have developed a mathematical methodology for fully automatic processing of root growth over time for different biological environments that implements the extraction of root parameters (e.g. thickness, length, number of adventitious roots, ...), or also the evaluation and comparison of data between control and groups with added test chemicals. The solution I have proposed is, in contrast to all existing programs in the world (e.g. Ez-Rhizo, Segroot), able to work in complex conditions where root systems often cross each other (e.g. in *in vitro* modules), or where parts of the roots are hidden behind the substrate (e.g. in rhizotrons) and need to be predicted for proper evaluation.

For the evaluation of the effect of chemicals, I came up with an innovative metric (interpretation of the parameters of the logistic curve) that eliminates all the disadvantages of RGR (RGR is not suitable for use on logistic phenomena; RGR is always calculated for only one variable from two time points – it is therefore discrete and not comparable when using different time scales). In addition, there are several other advantages to using a logistic curve evaluation - each analysis always yields three values for the entire growth curve, regardless of the number of data points measured or whether the growth curve was followed to the end of its growth period.

I implemented these methods in the Python programming language and created the Root Tracker software, which has been used by biologists at Palacký University in about 60 scans per day since January 2023, saving them several hours of manual work every day. During this period of more than a year, they have successfully used Root Tracker to identify several stimulants and have tested it on tens of thousands of images.

Contents

1 Image acquisition **5**

1.1 Rhizotron 5

1.2 In vitro 5

1.2.1 3D model 6

2 Program modules **7**

2.1 ROI 7

2.1.1 Manual 7

2.1.2 Cropping by thresholding 7

2.1.3 Cropping by median 8

2.2 Origin detection 9

2.2.1 Using green area center detection 9

2.3 Image processing 9

2.4 Root detection 12

2.4.1 Detection by thresholding 12

2.4.2 SNR detection 14

2.5 Missing root prediction 18

2.6 Property extraction 19

2.7 RGR 19

2.8 Logistic function 20

Conclusion **22**

List of sources **23**

List of figures **24**

List of equations **25**

1. Image acquisition

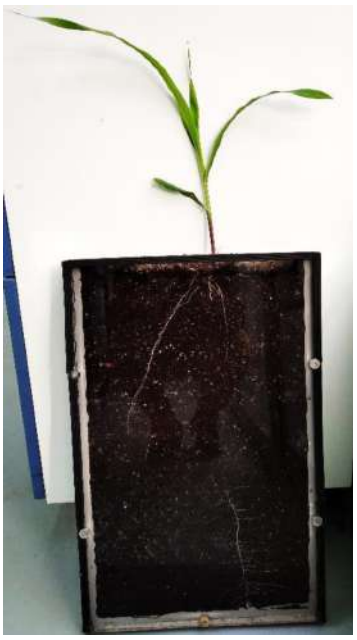
Image acquisition deals with the process of correctly capturing the desired real phenomenon for subsequent computer processing. The quality of the image capture is very important for later processing – imperfections in the image capture (such as bad reflections, ...) can degrade the performance of the model. [1]

1.1 Rhizotron

For regular rhizotron scanning, we decided to use a non-contact scanner placed in an indoor environment to minimize reflections from the transparent glass of the rhizotron.

1.2 In vitro

When taking pictures of the *in vitro* modules, we used a classic and affordable DSLR placed in a tripod with the possibility of fast data transfer to a computer.


(a) Rhizotron

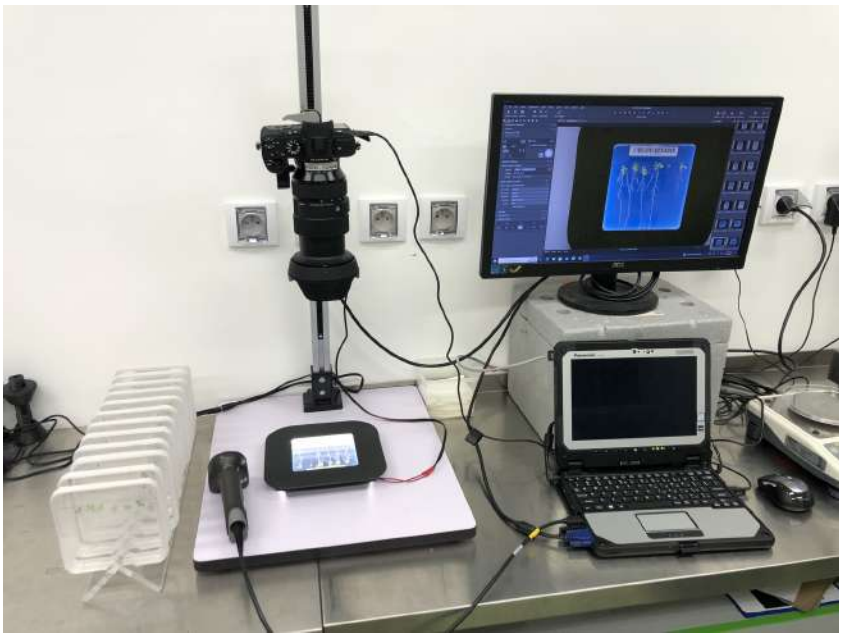

(b) *In vitro* module in the scanning setup

Figure 1.1: Rhizotron and *in vitro* module in a scanning setup

1.2.1 3D model

In order to capture the highest quality images, we decided to take advantage of 3D printing and the translucency of the media. I designed a 3D model of the holder for the *in vitro* modules. When designing the model, I emphasized that it would be easy to print on any 3D printer, that it could be attached to a substrate, and that it would be possible to adequately illuminate the *in vitro* roots using an LED strip placed around the outer edge of the holder.

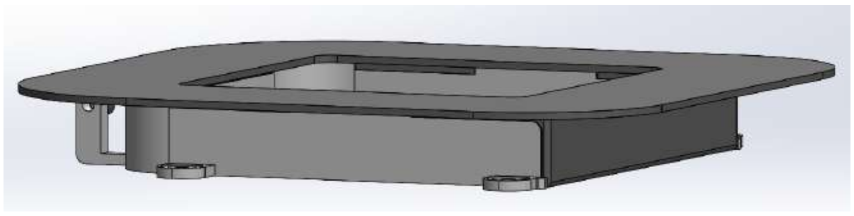


Figure 1.2: Design of an *in vitro* root photography setup

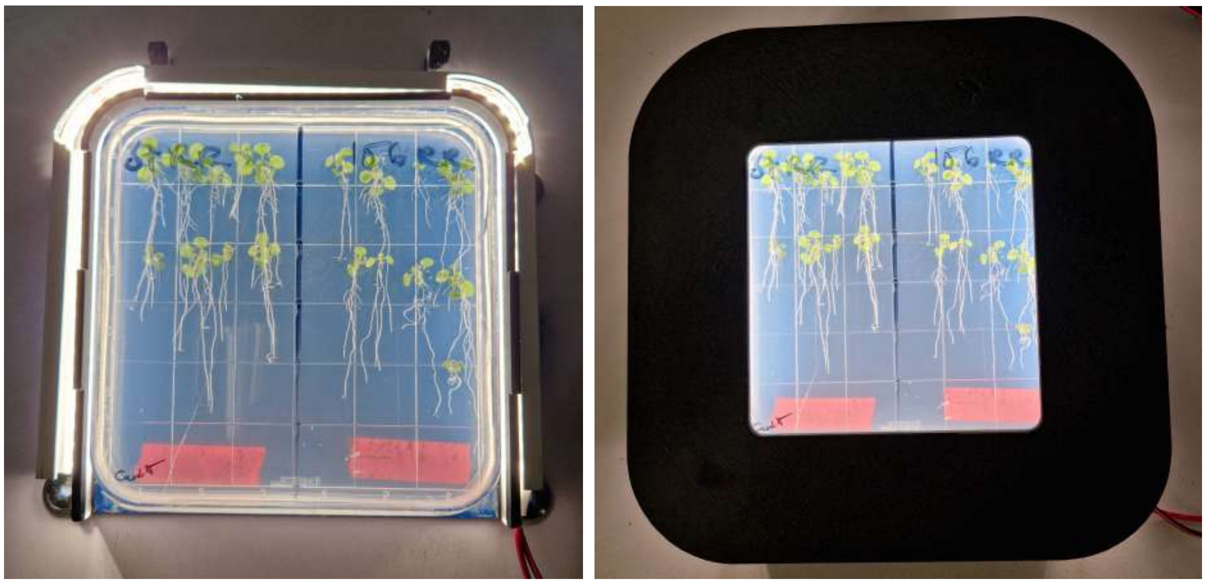

(a) Without cover (b) With cover

Figure 1.3: *In vitro* photography setup

2. Program modules

2.1 ROI

For a proper image analysis, it is necessary to select a region of interest (ROI). [2] Root Tracker offers several options for obtaining ROIs (due to the different assumptions of each method):

2.1.1 Manual

The basic method is to crop the image and rotate the image according to user input. This method can also be used in combination with the other automatic methods described below.

2.1.2 Cropping by thresholding

Root Tracker first performs thresholding in HSV color space to detect an object with a predefined color. If the color of the object is unknown in advance, it is also possible to use the Otsu thresholding method, which segments the object based on pixel brightness. [3]

Then the smallest detected objects are removed using a morphological operation – erosion and dilation. [4] I chose the morphological operation mainly to achieve a higher computational speed than filtering all objects by size using the *for* loop. Then the image is cropped according to the position and size of the smallest possible bounding rectangle so that all detected objects are inside this rectangle.

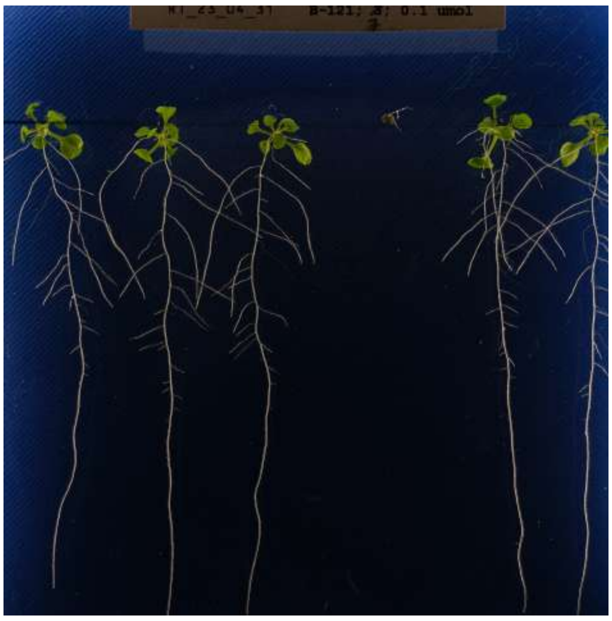


Figure 2.1: Image automatically cropped using thresholding

2.1.3 Cropping by median

This algorithm first calculates the average brightness value of all pixels for each column of the image using formula:

$$g(x) = \frac{1}{N}\sum_{y=1}^{N} f(x,y)$$

Rovnice 2.1: Average value for the column x

Root Tracker then calculates the median of all the average values to obtain the background brightness. To this value, it adds the specified constant determining the background heterogeneity to determine a threshold value (in the case of a substrate background, it is appropriate to use a constant of $\frac{1}{20}$ image bit depth) and then just extracts the first and last column that has a value below the threshold.

The Root Tracker performs the same operation for all rows, thus cropping the image vertically and horizontally according to the extracted information.

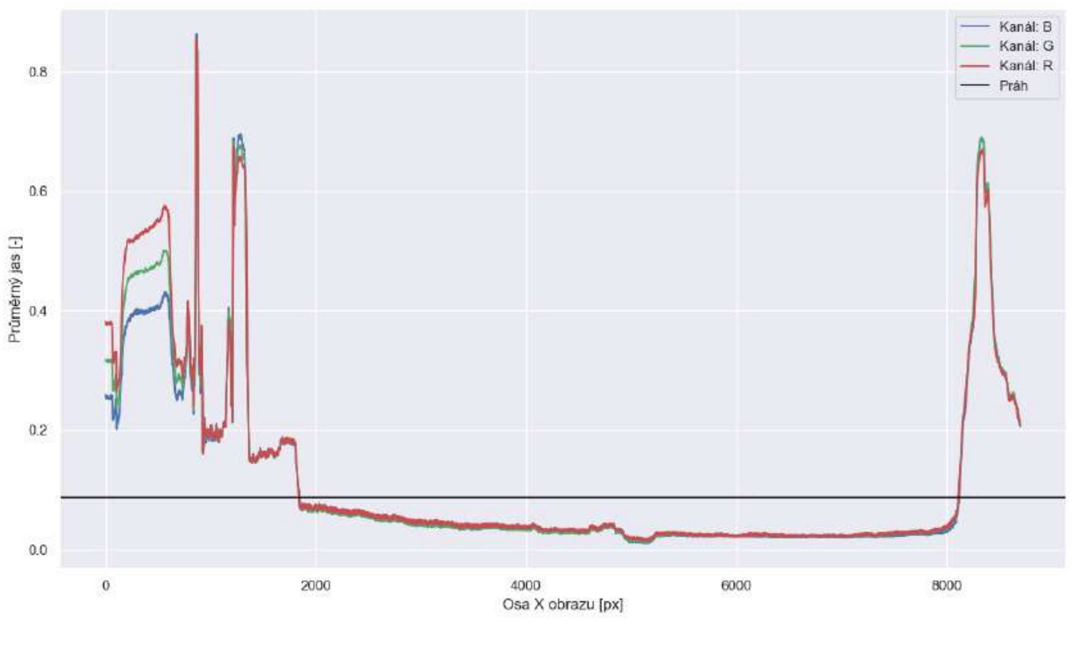


Figure 2.2: Average brightness of the rhizotron columns and the specified threshold

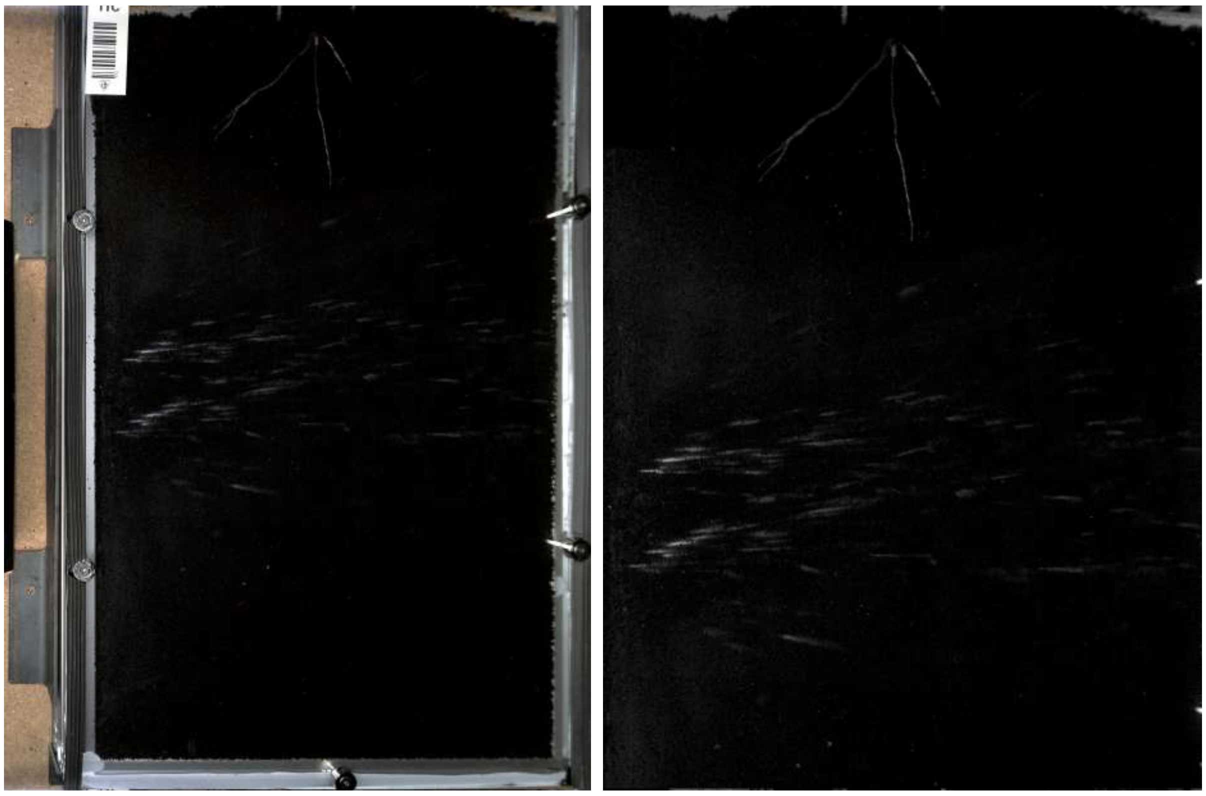

(a) Original image (b) Cropped image

Figure 2.3: Rhitoztron cropping algorithm

2.2 Origin detection

The scanned model may contain different numbers of plants planted in different locations, so Root Tracker automatically finds the points from which the plants grow by using the highest points of the root systems or by detecting the green area.

2.2.1 Using green area center detection

This method first uses the thresholding in HSV space for the green color. It then calculates the center of the smallest possible circle describing the outermost boundary of a given object, and then uses the clustering method KMeans [5] to determine the centers of individual plants.

2.3 Image processing

For quality image processing, it is advisable to first remove easily filterable types of impurities from the image. For this purpose, Root Tracker applies a median filter to ensure the removal of, for example, glare, shimmering particles in the substrate, or striations in the glass/plexiglass.

Because roots are often hidden, especially for plants in the substrate, to speed up the calculation for each pixel Root Tracker calculates a new value as the average of the 48

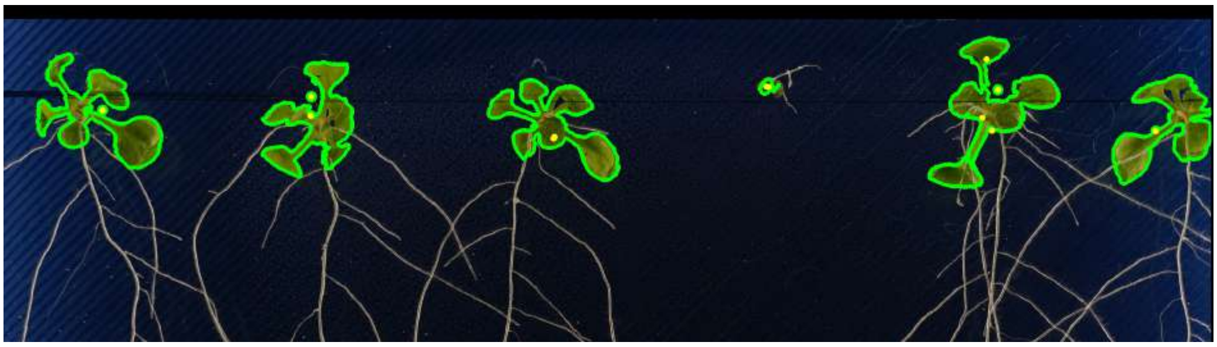


Figure 2.4: Detecting the above-ground part of the plant and calculating the origin

adjacent pixels using convolution, thus combining the smallest gaps. Finally, it applies a Meijering filter to highlight root-like structures. [6]

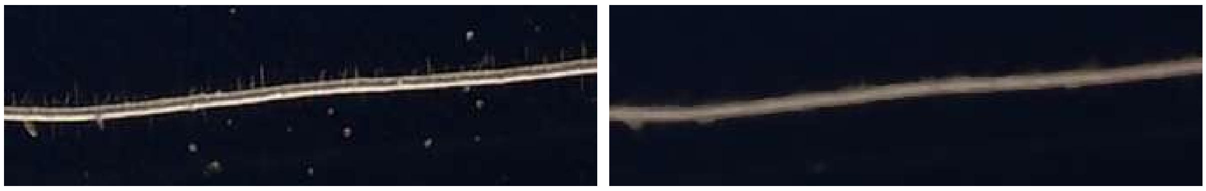

(a) Original image (b) Pre-processed image

Figure 2.5: Pre-processing – applied on a root with lighting defect

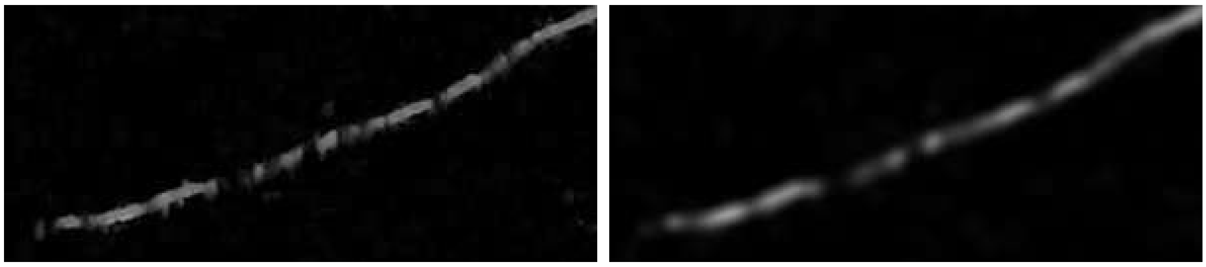

(a) Original image (b) Pre-processed image

Figure 2.6: Pre-processing – applied on a normal root

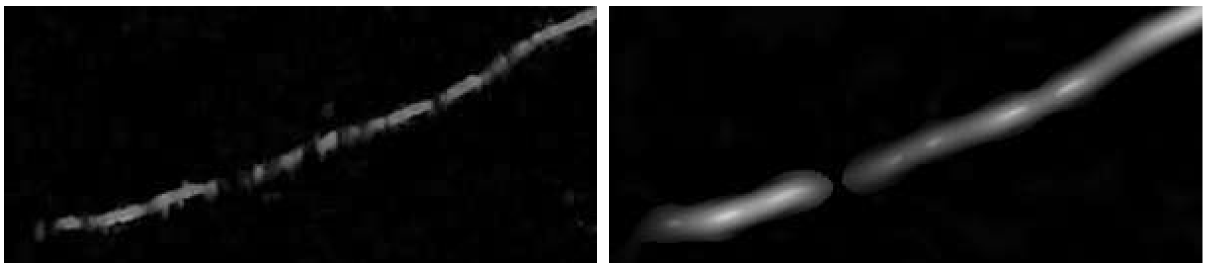

(a) Original image (b) After filter application

Figure 2.7: Pre-processing – Meijering filter

Finally, Root Tracker calculates the background of the image by applying the median using convolution with a large convolution window to obtain the background gradient, which it subtracts from the original image.

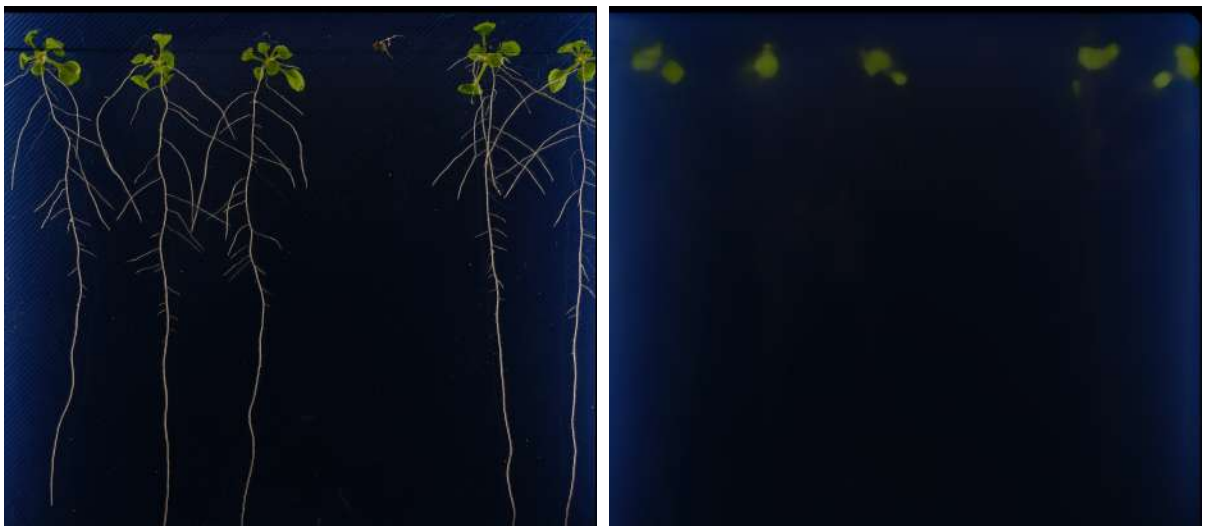

(a) Original image (b) Image background

Figure 2.8: Background extraction for *in vitro*

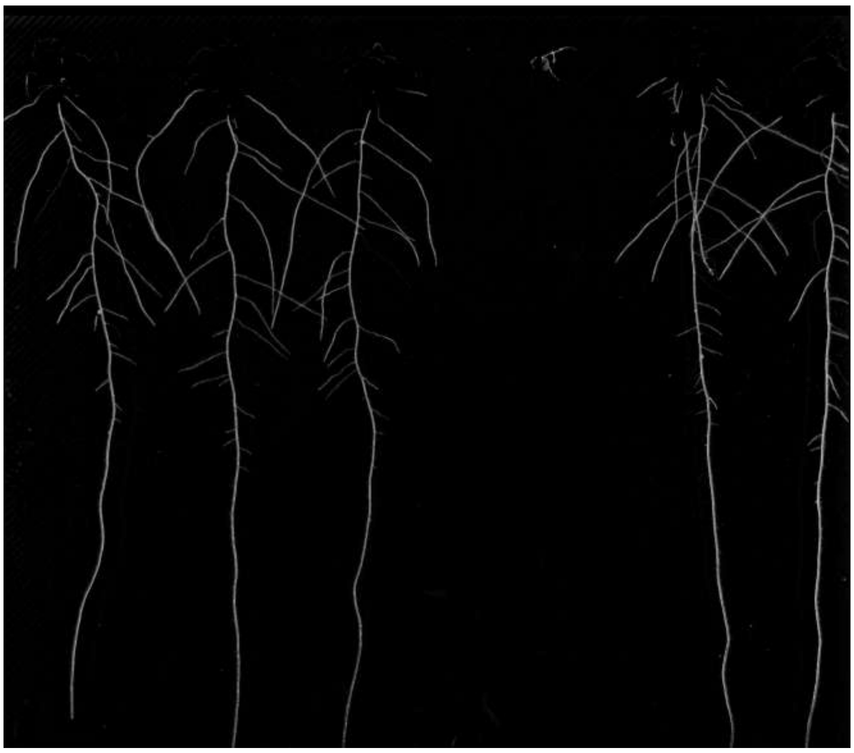


Figure 2.9: Final pre-processed image

2.4 Root detection

Root Tracker contains two algorithms for extracting the root hierarchy:

2.4.1 Detection by thresholding

Thresholding detection alone is not effective due to the frequent occurrence of impurities in the image. Therefore, I have implemented several measures, such as double thresholding – first a high thresholding is performed, which detects only clearly identified roots, and then a low thresholding is performed, which may contain objects other than roots. Root Tracker then discards any contours that are not at least partially present in the high threshold image.

From this image, Root Tracker creates a skeleton [7] to process the subsequent data quickly and easily.

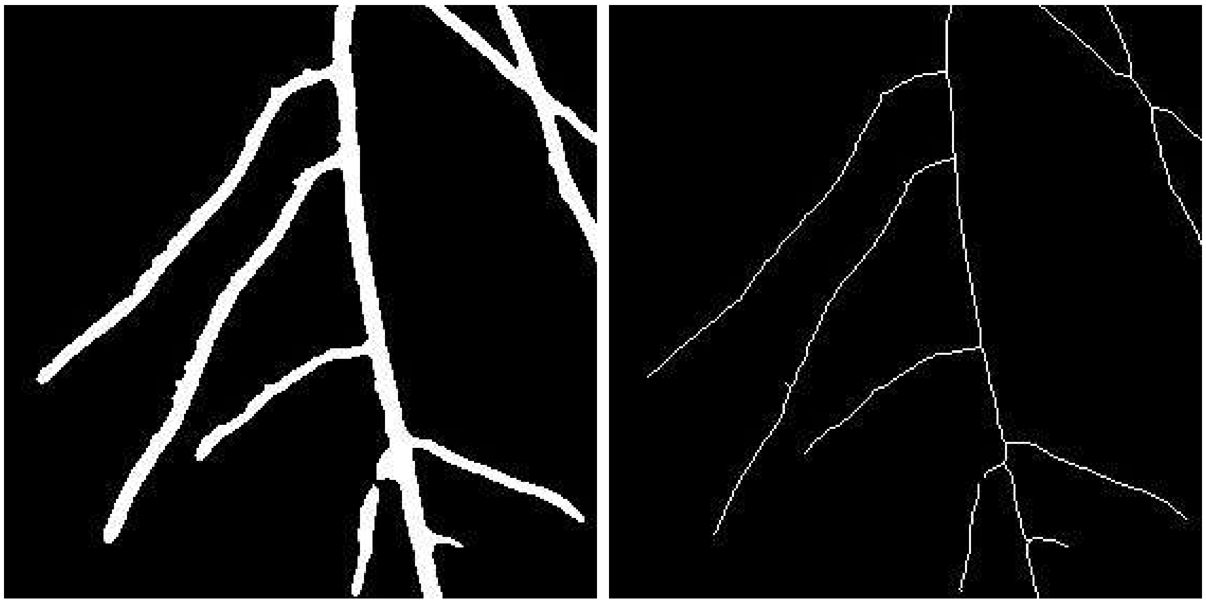

(a) Binarized image (b) Skeleton image

Figure 2.10: Skeleton of binarized image

In order to quickly calculate and find the ends of the roots, I thought of creating my own transformation using convolution:

$$g(x, y) = \max(0, K * f(x, y) - Z),$$

Rovnice 2.2: Transformation to calculate the number of adjacent pixels

where f is the input skeleton, g is the image where the value of each pixel corresponds to the number of adjacent white pixels in its neighborhood 3x3 and $Z > 8$. The maximum is used because of the removal of negative values after subtracting Z, which is implemented to filter all off-skeleton values. The convolution window K looks as follows:

$$K = \begin{bmatrix} 1 & 1 & 1 \\ 1 & Z & 1 \\ 1 & 1 & 1 \end{bmatrix},$$

Rovnice 2.3: Convolution window K

After this operation, the Root Tracker disconnects all connections in the skeleton (resets pixels where $g \geq 3$ and the closest pixels) due to the skeletonization function that connects most root crossings into a single bundle several pixels ahead.

Then, for all ends where $g \neq 1$ (ends that were created after the crossings were removed), it computes the average difference converted to an angle from the last few pixels of both ends of the contour, and then computes the average difference from the average angle again from the last few pixels to capture the curvature. After computing these properties, it combinatorially computes an error function for each of the two ends that can be connected, and selects the connection with the smallest error E:

$$E(t_i, b_j) = ||t_i - b_j||_2 + |a_{t_i} - a_{b_j}| + |d_{t_i} - d_{b_j}|,$$

Rovnice 2.4: Error function $E(t_i, b_j)$

where a are first differences, d are second differences from angles[1], $t_i[x, y]$ and $b_j[x, y]$ are two different ends ($i \neq j$) of disconnected roots, where $t_i[x, y]$ is the upper end (closer to the origin) and $b_j[x, y]$ is the lower root (further from the origin), so only root end connections where b_y is higher than t_y can be considered.

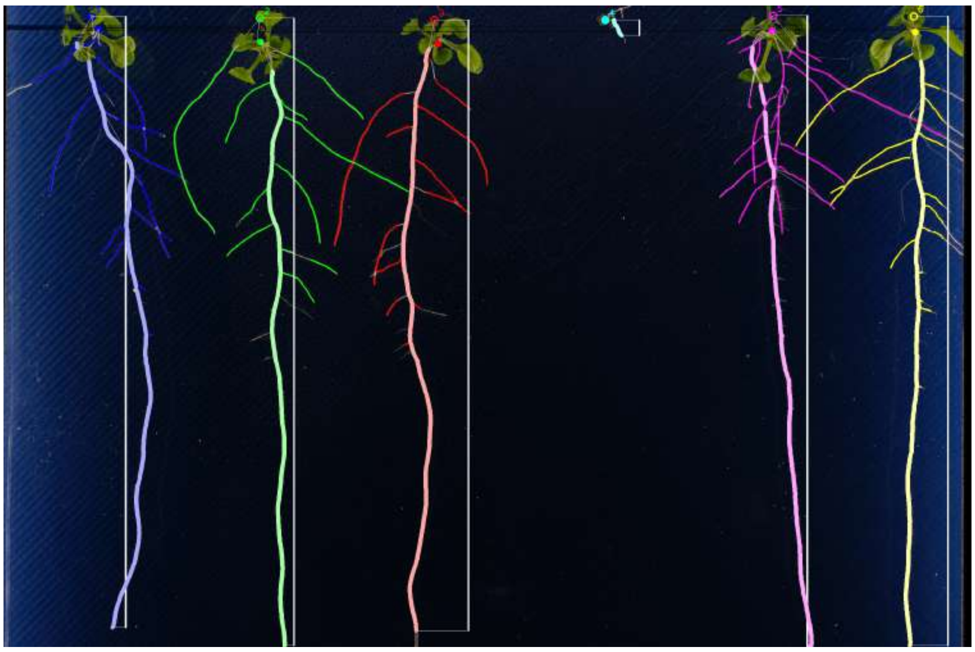


Figure 2.11: Final processed *in vitro* image

[1]The given quantities are first rotated by 180°to be comparable, since the two given ends must always point towards each other, not away from each other

In the output image, the Root Tracker enumeratively labels the root structures from left to right and assigns a unique color to each. Next, it thickly labels the main root and draws a line next to it to indicate the maximum depth reached by the entire root system.

2.4.2 SNR detection

For more complex cases (for example, when there are strong reflections in the frame), I developed a unique method for Root Tracker that detects the root hierarchy directly when it is binarized.

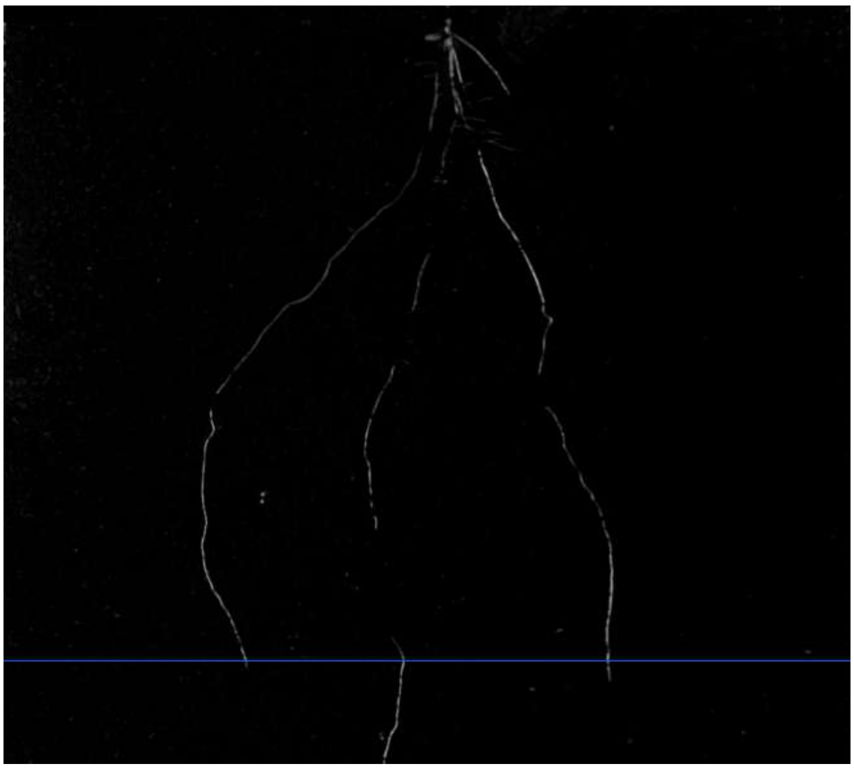
The Root Tracker goes through each row from the last to the first, looking for vertices that have a value higher than the 80% quantile (thus excluding most of the background, but keeping the roots).

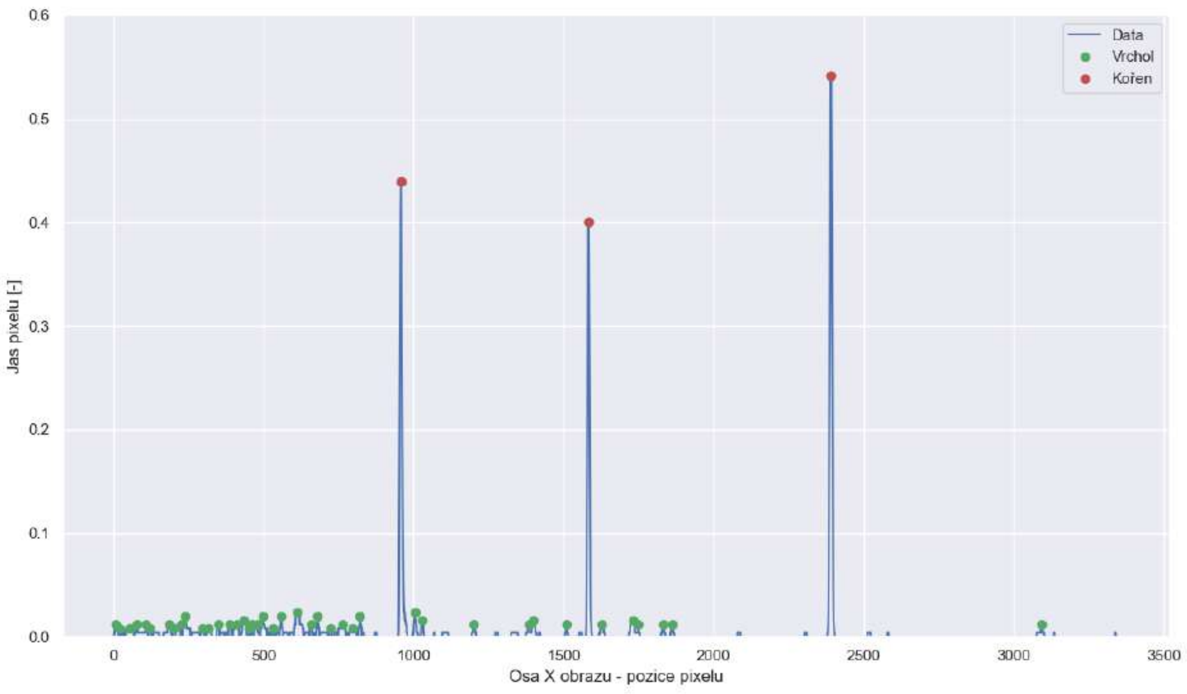
Root Tracker again implements thresholding in an elegant way – in order for a given vertex to be classified as a root and tracked further, it must first exceed a high threshold. Once it has crossed it, it is classified as a root, and if, when the next row is traversed, a vertex is found at a location close to the previous location, it need only cross the low threshold and be in a similar direction to the previous vertices. The signal-to-noise ratio is then calculated as follows:

$$SNR(x) = \frac{x}{NOISE},$$

Rovnice 2.5: SNR calculation

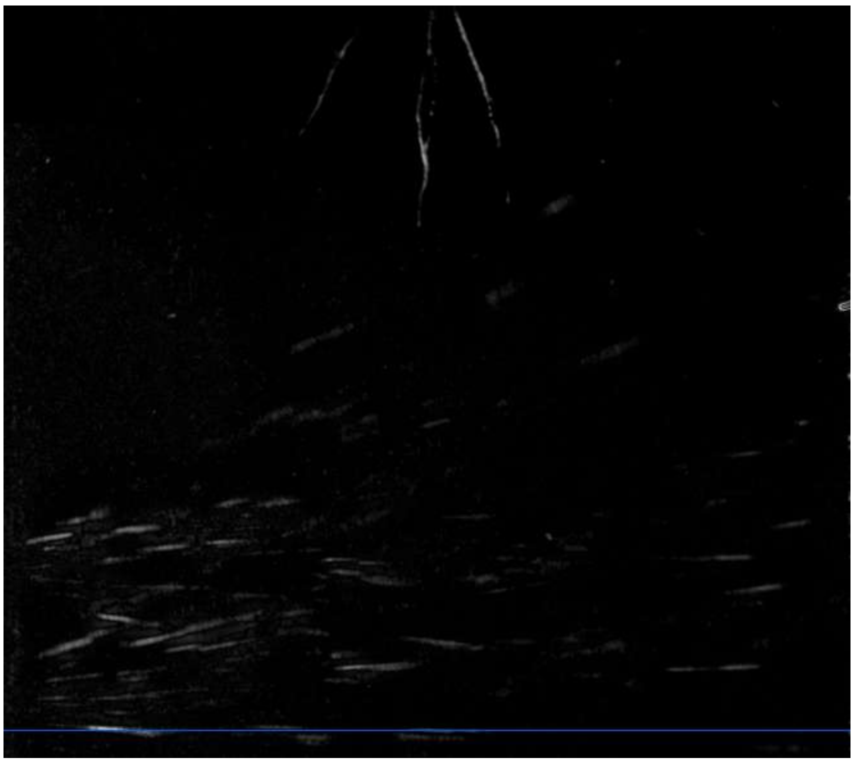
where $NOISE$ is again defined as the 80% quantile of the nearest neighborhood and x is the brightness of the vertex. If this value exceeds the threshold and satisfies the previous conditions, the pixel is assigned to the previous vertex from which the direction and distance were compared. In this way, the entire root hierarchy is created.

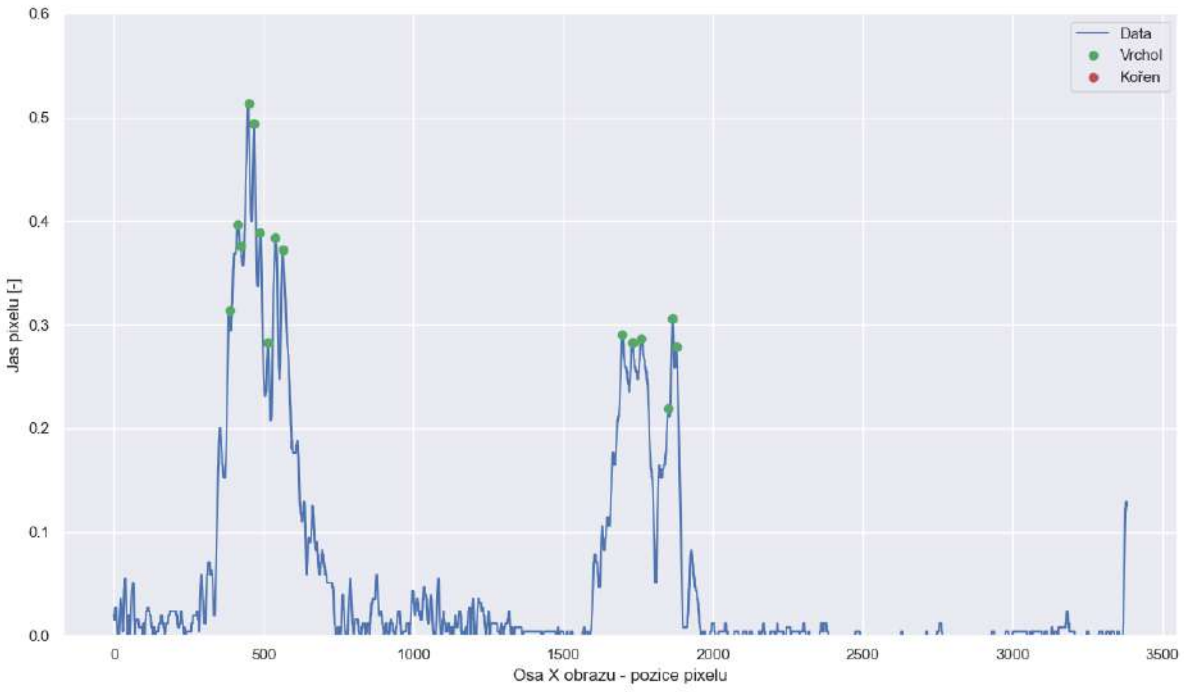

(a) Original image



(b) Image row and vertex detection

Figure 2.12: Row of the image where the roots are present


(a) Original image



(b) One of the rows, where reflections are present

Figure 2.13: Row of the image with strong reflection

This procedure makes the method very robust for root detection. It also does not detect the long horizontal flares that often occur in the image, although individual flares often reach higher luminance values than the roots themselves.

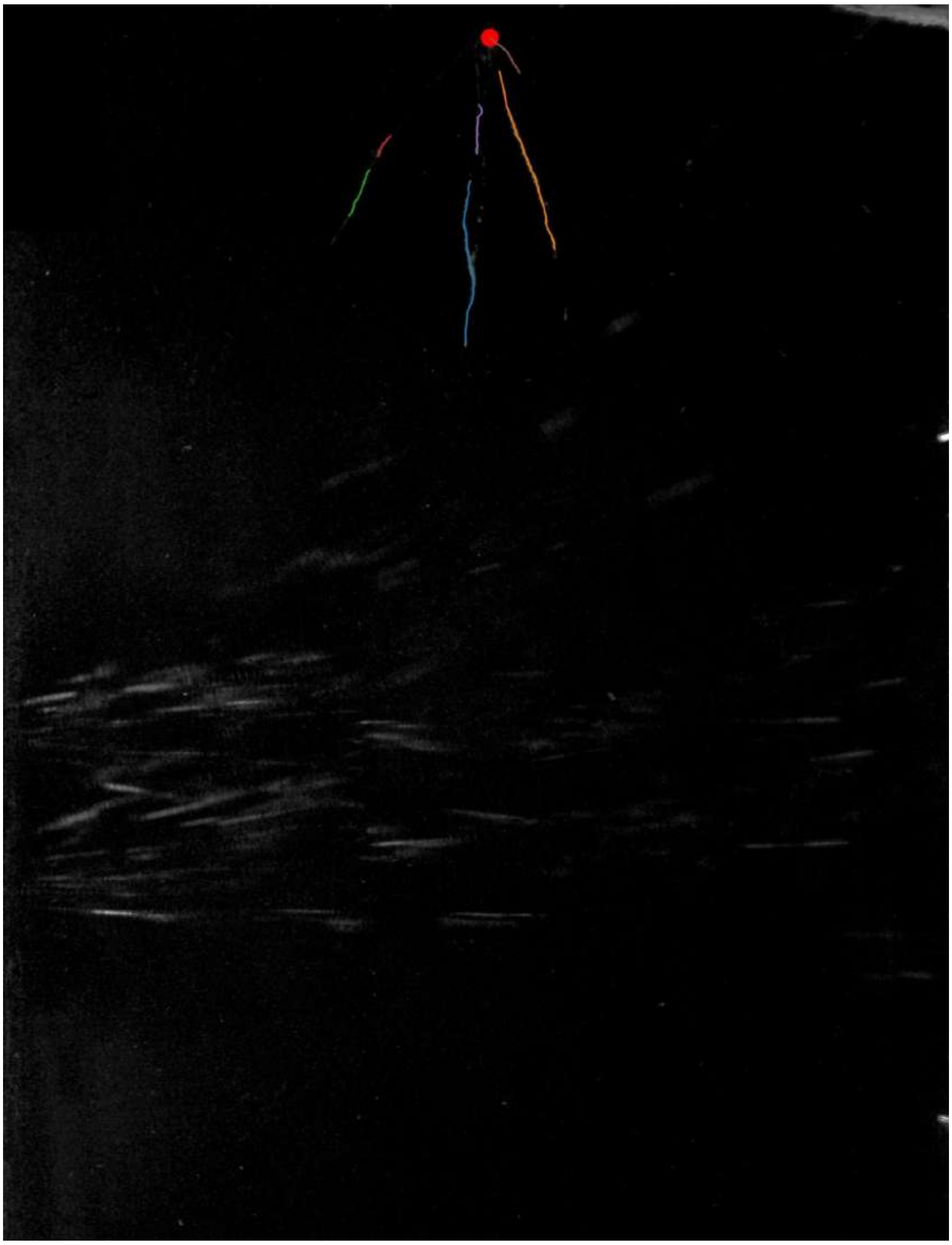


Figure 2.14: Image with strong reflections

2.5 Missing root prediction

For proper evaluation in the case of images with hidden roots, Root Tracker predicts the missing sections between the free ends of the roots using the Bezier curve [8]. The mid-control points of the curve are determined by Root Tracker polarly as points that are located from the ends of the roots at a distance $d = \frac{1}{4}d_0$, where d_0 is the distance of the ends. The choice of roots to be connected is determined by optimization (penalizing the number of free ends, the distance and the direction of the root ends).

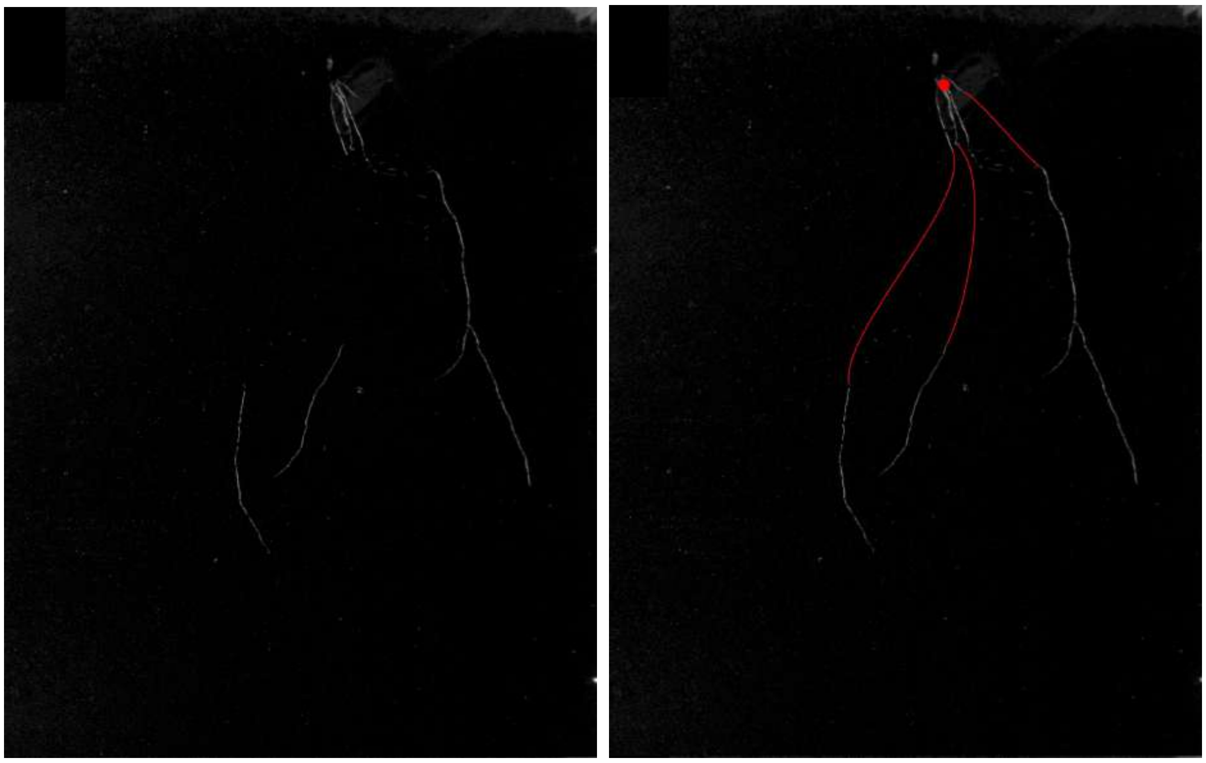

(a) Without prediction of missing roots (b) With prediction of missing roots

Figure 2.15: Prediction of missing roots

2.6 Property extraction

Root Tracker generates information for all roots individually, so it can calculate any desired metric. Commonly used ones include the length and depth (distance of the lowest point of the root system from the origin) of the main root, the number of lateral roots, or the area of the root system.

Root Tracker stores all calculated information in a CSV output file, from which it can then create visualizations suitable for comparison and evaluation.

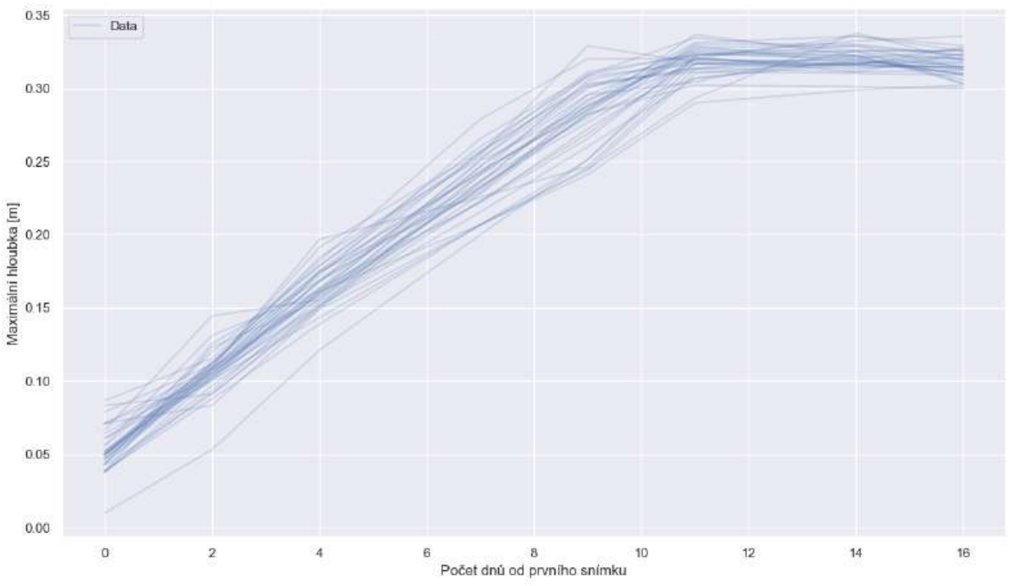


Figure 2.16: Visualization of raw data from CSV. On the x-axis is the number of days since the first image and on the y-axis is maximal depth in meters.

2.7 RGR

Once Root Tracker has evaluated the selected metrics, it can calculate a standard biological evaluation metric for any variable (length, depth, ...) – relative growth rate [9]:

$$RGR = \frac{\ln S_2 - \ln S_1}{t_2 - t_1},$$

Rovnice 2.6: RGR calculation

where S_1 and S_2 is the observed quantity (e.g. size, length, area) of the selected plant part (e.g. leaf, root) measured at two time points t_1 and t_2.

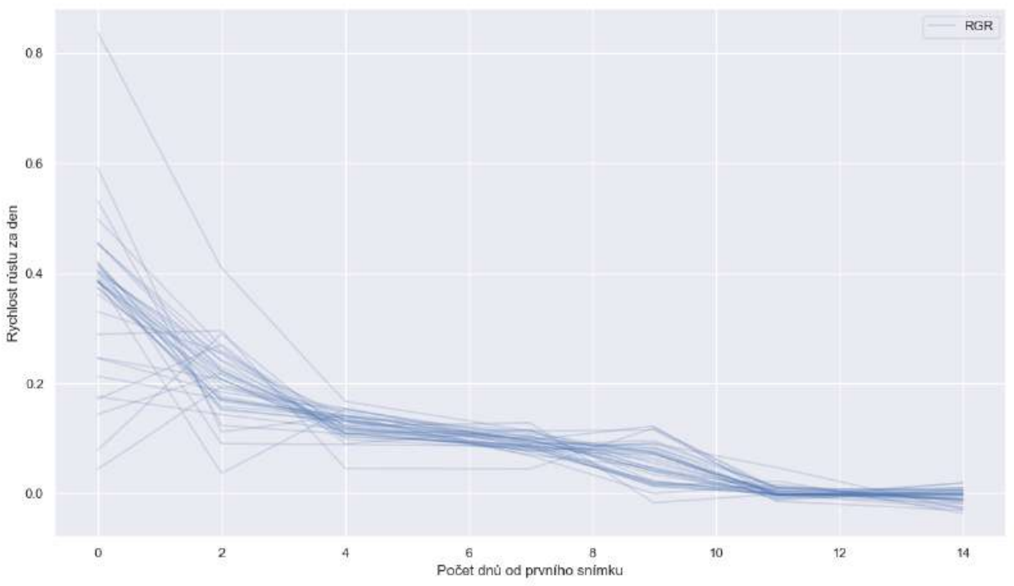


Figure 2.17: Visualization of the dependence of the calculated RGR for depth over time. On the x-axis is the number of days since the first image and on the y-axis is RGR

2.8 Logistic function

Since the root system of plants does not grow exponentially but logistically in long-term monitoring, Root Tracker also comes up with an innovative metric for evaluation. It first passes the extracted data and then interpolates it with the logistic curve [10] using the least squares method [11]:

$$f(x) = \frac{A}{1 + e^{-B(x+C)}},$$

Rovnice 2.7: Equation of the logistic function

where:

- The parameter A represents the maximum value of the monitored variable that the variable has reached, or will reach once it stops growing (in case the scan was stopped earlier).
- The parameter B represents the growth rate over time, not just at individual time points as in RGR
- The parameter C symbolizes the time shift (whether the root system grew earlier/later in time)

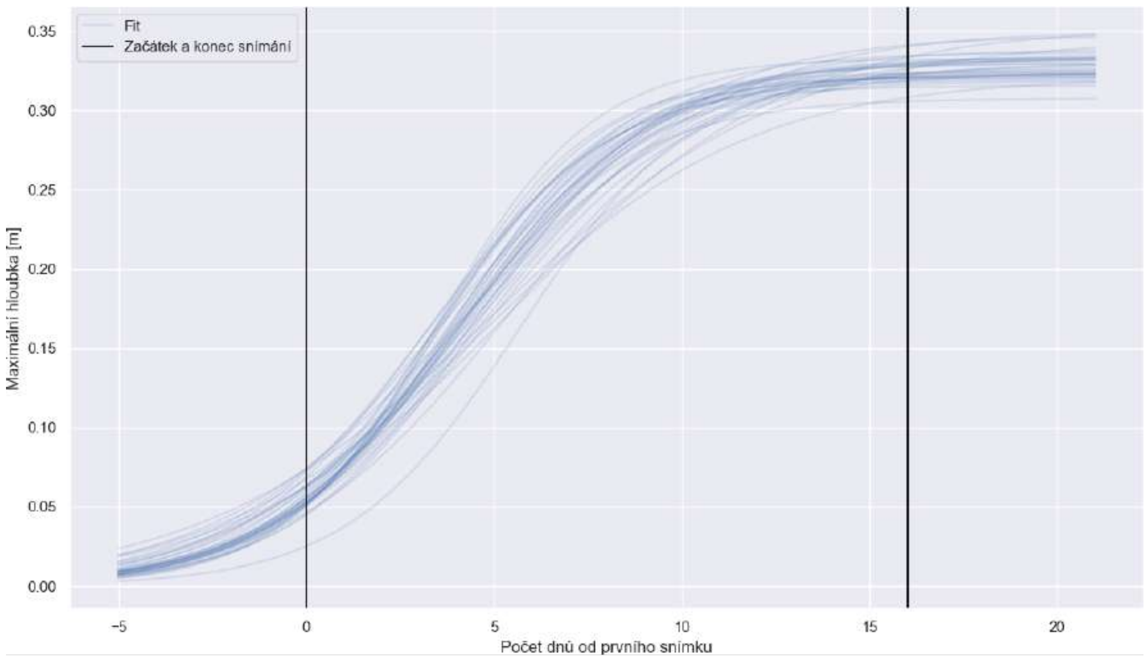


Figure 2.18: Data interpolated using logistic curve. On the x-axis is the number of days since the first image and on the y-axis maximal depth in meters.

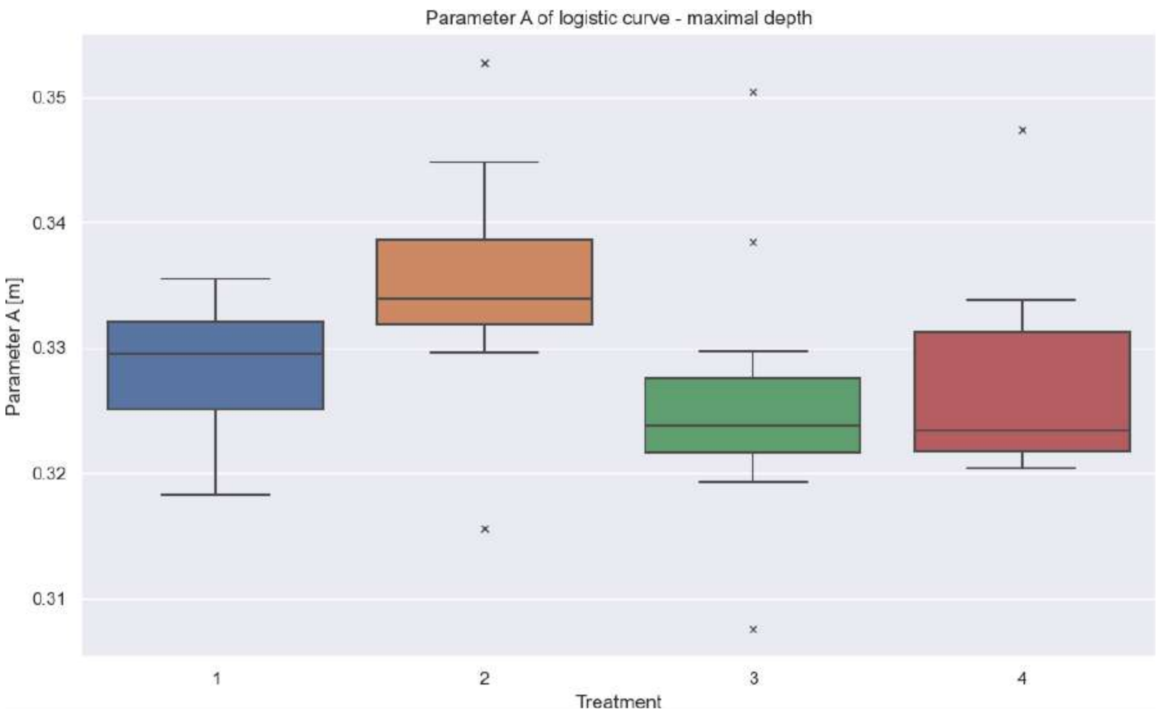


Figure 2.19: Evaluation based on parameter A of the logistic function

Conclusion

I have developed algorithms for processing, extracting parameters (e.g. thickness, length, ...) and evaluating data for images of root systems, as well as for complex cases where complex root systems often intersect, or where parts of the roots are hidden behind the substrate and need to be predicted for proper evaluation. I then implemented this methodology in a modular way in Python (using NumPy, OpenCV and Scikit-Learn libraries). For the visualizations, I used the classical way of evaluation using RGR, but also an i innovative way that interprets the parameters of the logistic curve. I also designed several 3D models (for better fitting of *in vitro* modules, or for printing my own rhizotrons) and then printed them for Palacký University. These models standardize the growing conditions and further simplify the already complicated work of the researchers.

The Root Tracker is currently used by biologists at Palacký University at a rate of about 60 images/day, which saves them several hours/day compared to manual processing. In addition, Root Tracker allows them to achieve an accurate assessment, which would be impossible without it (apart from Root Tracker, there is currently no fully functional solution; individual research groups thus ”extrapolate” the size of root systems using the volume of the above-ground part of the plant, or measure individual root systems approximately with a ruler).

I am currently working on extending this with a user interface to simplify configuration as much as possible using modern web technologies. We plan to publish an peer-reviewed paper on Root Tracker later this year to help simplify the work for other scientific communities studying root systems. At the same time, we are working on extending Root Tracker to include worm detection, which is important for our follow-up project on the effect of parasitic worms on root system growth.

I see further applications of the Root Tracker in aquatic toxicology, where, among other things, the effect of different chemical preparations (or toxic properties of wastewater and aqueous effluents) on plant growth is tested as a proxy for primary producers (seed germination test and growth inhibition of *Sinapis alba*). This may have implications, for example, for root-based wastewater treatment plants, where the size of the root system of aquatic plants with aerobic microflora has a major impact on the efficiency of treatment.

Furthermore, I think the Root Tracker will be a great tool for evaluating chemical additives that increase the bulk in the field of water retention. The ability of the landscape to retain water is being addressed in the present day. It turns out that well-functioning soils can hold a huge amount of water, whereas soils that are compacted by heavy machinery (which lack sufficient organic matter), have a much lower water-holding capacity – water infiltrates poorly into such soils and is more likely to run off through surface or subsurface runoff out of the landscape. Improving soil properties, preventing water runoff – e.g. by planting various grass strips, etc. to increase water retention in the landscape – is the subject of so-called soft or ecological measures. The better the root system of grasses and other plants, the better the water retention of the landscape will be.

List of sources

1. MISHRA, Vikas Kumar; KUMAR, Shobhit; SHUKLA, Neeraj. Image Acquisition and Techniques to Perform Image Acquisition. *SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology*. June 2017, vol. 9, no. 01, pp. 21–24. ISSN 2229-7111. Available from DOI: 10.18090/samriddhi.v9i01.8333.
2. BRINKMANN, Ron. *The Art and Science of Digital Compositing*. Oxford, England: Morgan Kaufmann, June 1999. The Morgan Kaufmann Series in Computer Graphics. ISBN 978-0-12-133960-9.
3. OTSU, Nobuyuki. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*. January 1979, vol. 9, no. 1, pp. 62–66. ISSN 2168-2909. Available from DOI: 10.1109/tsmc.1979.4310076.
4. SAID, K A M; JAMBEK, A B. Analysis of Image Processing Using Morphological Erosion and Dilation. *Journal of Physics: Conference Series*. October 2021, vol. 2071, no. 1, p. 012033. ISSN 1742-6596. Available from DOI: 10.1088/1742-6596/2071/1/012033.
5. HARTIGAN, J. A.; WONG, M. A. Algorithm AS 136: A K-Means Clustering Algorithm. *Applied Statistics*. 1979, vol. 28, no. 1, p. 100. ISSN 0035-9254. Available from DOI: 10.2307/2346830.
6. MEIJERING, E.; JACOB, M.; SARRIA, J.-C.F.; STEINER, P.; HIRLING, H.; UNSER, M. Design and validation of a tool for neurite tracing and analysis in fluorescence microscopy images. *Cytometry Part A*. March 2004, vol. 58A, no. 2, pp. 167–176. ISSN 1552-4930. Available from DOI: 10.1002/cyto.a.20022.
7. ABU-AIN, Waleed; ABDULLAH, Siti Norul Huda Sheikh; BATAINEH, Bilal; ABU-AIN, Tarik; OMAR, Khairuddin. Skeletonization Algorithm for Binary Images. *Procedia Technology*. 2013, vol. 11, pp. 704–709. ISSN 2212-0173. Available from DOI: 10.1016/j.protcy.2013.12.248.
8. MORTENSON, Michael E. *Mathematics for Computer Graphics Applications*. 2nd ed. New York, NY: Industrial Press, January 1999. ISBN 9780831131111.
9. SOUTH, David B. Relative Growth Rates: a Critique. *South African Forestry Journal*. July 1995, vol. 173, no. 1, pp. 43–48. ISSN 0038-2167. Available from DOI: 10.1080/00382167.1995.9629690.
10. KUCHARAVY, Dmitry; DE GUIO, Roland. Application of Logistic Growth Curve. *Procedia Engineering*. 2015, vol. 131, pp. 280–290. ISSN 1877-7058. Available from DOI: 10.1016/j.proeng.2015.12.390.
11. STIGLER, Stephen M. Gauss and the Invention of Least Squares. *The Annals of Statistics*. May 1981, vol. 9, no. 3. ISSN 0090-5364. Available from DOI: 10.1214/aos/1176345451.

List of Figures

1.1 Rhizotron and *in vitro* module in a scanning setup 5
1.2 Design of an *in vitro* root photography setup 6
1.3 *In vitro* photography setup 6

2.1 Image automatically cropped using thresholding 7
2.2 Average brightness of the rhizotron columns and the specified threshold . . 8
2.3 Rhitoztron cropping algorithm 9
2.4 Detecting the above-ground part of the plant and calculating the origin . . 10
2.5 Pre-processing – applied on a root with lighting defect 10
2.6 Pre-processing – applied on a normal root 10
2.7 Pre-processing – Meijering filter 10
2.8 Background extraction for *in vitro* 11
2.9 Final pre-processed image 11
2.10 Skeleton of binarized image 12
2.11 Final processed *in vitro* image 13
2.12 Row of the image where the roots are present 15
2.13 Row of the image with strong reflection 16
2.14 Image with strong reflections 17
2.15 Prediction of missing roots 18
2.16 Visualization of raw data from CSV 19
2.17 Visualization of the dependence of the calculated RGR for depth over time 20
2.18 Data interpolated using logistic curve 21
2.19 Evaluation based on parameter A of the logistic function 21

List of equations

2.1 Average value for the column x 8
2.2 Transformation to calculate the number of adjacent pixels 12
2.3 Convolution window K 13
2.4 Error function $E(t_i, b_j)$ 13
2.5 SNR calculation 14
2.6 RGR calculation 19
2.7 Equation of the logistic function 20