IMPERIAL COLLEGE LONDON

REPORT OF PROJECT 3

RootSkel – A software tool to measure the angle of curved plant root tips

---

Author:
Felicia BURTSCHER

Supervisor:
Dr Giovanni SENA

A thesis submitted in fulfillment of the requirements
for the degree of MSc Bioinformatics and Theoretical Systems Biology

in the

Department of Life Sciences

September 1, 2018

Word count: 4,739
(without table of contents, acknowledgements and appendix)

"If you thought that science was certain – well, that is just an error on your part."

Richard Feynman

IMPERIAL COLLEGE LONDON

Abstract

Department of Life Sciences

MSc Bioinformatics and Theoretical Systems Biology

***RootSkel* – A software tool to measure the angle of curved plant root tips**

by Felicia BURTSCHER

Plant morphology studies the form and structure of plants and how these develop over time under different conditions.

One phenomenon that, despite being first identified more than 100 years ago, has not been researched much and we know little about its underlying molecular mechanisms, is *electrotropism*. Electrotropism describes the response of a plant to an electric field; one common response is the curving of a plant's root tip. Unlike gravitropism, there exist to our knowledge no tools specifically designed to assist biologists in studying this effect, most importantly the angle resulting from the curving of the root tip. This is especially relevant as the experimental setup for electrotropism is more complex and the images tend to be more error-prone, which is often the reason why standard gravitropism study tools fail and biologists compute the angles resulting from the curved root tip manually.

Here we present *RootSkel*, a novel intuitive and flexible stand-alone software for image processing developed in *MATLAB*, whose pipeline was optimised for noise-intensive electrotropism images. Unlike when doing the angle computation to measure the curvature of a root tip manually, our tool ensures a standardised version of the angle computation. To make the tool more user-friendly, we developed a graphical user interface (GUI) that will help the user to process the images and compute the angles in a standardised and controlled fashion.

Additionally, we computed the angle of 3 *Arabidopsis thaliana* root tips at 11 time points over one experiment each (approximately 5 hours) and evaluated the results by comparing it to previously manually computed angles of the same roots. The results show similar patterns in the computed angles and two sample t-tests on the data show they are not significantly different ($\alpha = 0.05$). Statistics on our data sets suggest that our tool can improve the precision of the manually computed angles by up to 12.4%. However, further validation on more images needs to be done; measurement errors of both the manually as well as computationally computed angles need to be obtained in order to show robustness of our method. Unlike the manually computed angles, with using a standard definition of the angle, our software delivers angles with less human bias which makes the results comparable and reproducible. Previously computed angles can be checked by using our software, and more angles of curved roots can be computed in the future and will hopefully reveal interesting insights into electrotropism. Moreover, our tool is not limited to roots but could theoretically be used on any curved object.

Acknowledgements

This project was conducted under the supervision of Dr Giovanni Sena, whom I thank for his advice and guidance. Thanks also to Nick Oliver for his help and expertise from the biological side and other members from the Plant Morphogenesis Laboratory at Imperial College London, as well as Suhail A Islam for fixing technical issues and his incredible patience. Finally, thanks to Prof Michael PH Stumpf, Prof Michael Sternberg and others involved in conducting and overseeing the MSc in Bioinformatics and Theoretical Systems Biology at Imperial College London.

Contents

List of Abbreviations

Arabidopsis	***Arabidopsis** thaliana*	
EF	**E**lectric **F**ield	
e.g.	*latin* **e**xempli **g**ratia	for example
GUI	**G**raphical **U**ser **I**nterface	
i.e.	*latin* **i**d **e**st	that is
LHS	**l**eft-**h**and **s**ide	
RHS	**r**ight-**h**and **s**ide	

For Yaron Efrat – the person I admire the most.

Chapter 1

Introduction

In the following chapter we will briefly familiarise the reader with the biological background of this work, summarise recent literature regarding the problem at hand to put our work into context and show in which way it contributed to the field.

1.1 Biological background

An important biological phenomenon studied by plant morphologists is *tropism*, which is used to indicate the directional movement of a biological organism, here of a plant, when exposed to different environmental stimuli [33]. Usually the stimulus involved is added to the name, e.g. *phototropism* as a reaction to sunlight; it can be either *positive*, i.e. towards the stimulus, or *negative*, i.e. away from the stimulus. Various types of tropism are shown in figure 1.1.

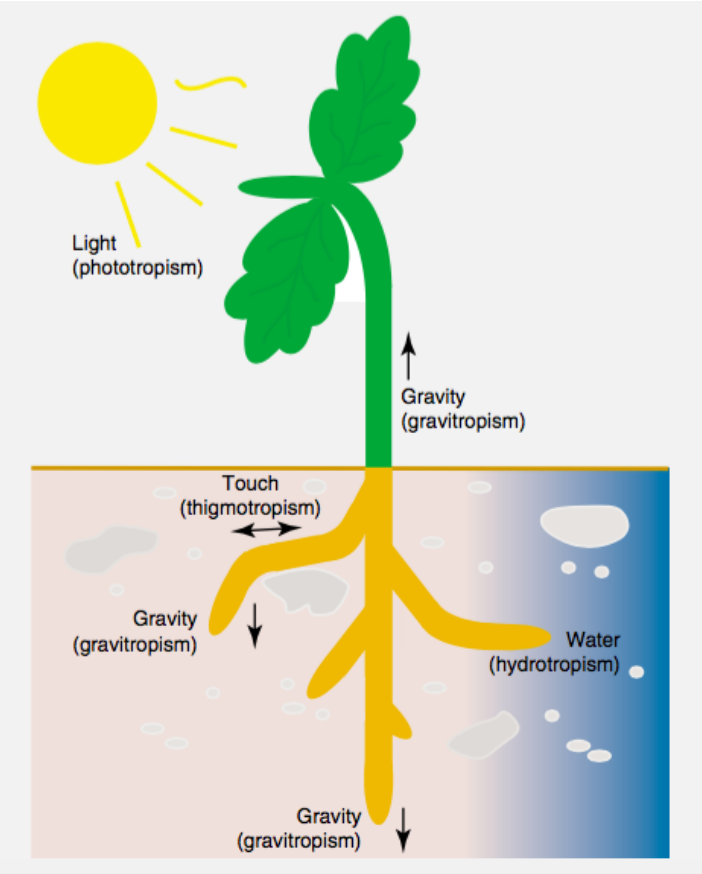

FIGURE 1.1: Different forms of tropism: Phototropism, thigmotropism, hydrotropism, and different effects of gravitropism; taken from [14].

The most frequently observed and best studied tropism is *gravitropism*, which describes the process of how plants grow as a response to gravity. It was firstly scientifically documented by Charles Darwin [27] and can be observed in many plants as well as other organisms: Roots show *positive gravitropism*, i.e. they grow in the direction of the gravitational pull whereas stems grow in the opposite direction, see figure 1.1.

A far less studied process is *electrotropism* which describes the growth or movement of a plant when exposed to an electric field (EF) and which is the tropism under study in this project.

A high-overview explanation of the experimental setup and data collection to study electrotropism can be found in sections 2.1 and 2.2.

1.2 Literature review

The majority of traditional root development bioassays only consider a small number of points [11, 28]. They are informative in terms of long-term effects on root growth; however, small and temporary changes can not be captured [28]. Recently developed tools have considered a higher number of time points which allows a better study of how the growth process develops and the plant's responses [7, 11, 17, 24, 36].

Image sequences can contain a representative "snapshot" description of a plant's developmental stage; also they can be generated at high speed (see section 2.2). From manual time-lapse photography [23, 32] to measure lengths of seedlings after applying different external stimuli, digital camera technology has improved and low digital storage cost has become available which has made it comparatively easy to collect large, time-stamped digital image data sets monitoring root growth [11]. Once the root growth changes have been extracted from the image data, one can correlate their timing with the impact of different external signals including hormonal and environmental on processes such as cell division and cell expansion [11]. This will help us to understand root development in general.

Analysing image data manually, however, is time-consuming, very subjective and thus error-prone [11]. When the analysis is done "by eye", it becomes difficult to reproduce measurements as they are not standardised by any automatised approach; the computations lack objectivity, and are subject to a significant human bias. Also, subtle phenotypes, such as a delay in the response, might be missed [11].

Different groups have developed tools for gravitropism and have shown the power of automatised image-analysis techniques compared to manual methods. An overview was created in the appendix D.

1.3 Motivation

The work described here is motivated by mainly three factors: definition and standardisation or objectivity of the so far manually computed angle, flexibility and user-friendliness in the pre-processing step, and adaptability to standard consumer cameras. The software tool was designed to be used by a user, typically a biologist, with no need of specific knowledge in image processing nor plant morphology.

Chapter 2

Methods

The purpose of this methods section is to give an overview on, first, the data used to develop the software tool and how the data were collected, second, image processing basics, and third, which programming languages, definitions and methods were chosen when developing the tool. This will help the reader to replicate, understand and modify the methods and source code of the tool.

2.1 Data set

Our data set consisted of high-throughput time-lapse images of *Arabidopsis thaliana* (Arabidopsis) roots taken by a standard Raspberry Pi V2 camera from 5 experiments with 5-6 roots each containing between 32 and 36 images over a period of at least 5 hours, i.e. every 10 minutes a photo was taken. All experimental parameters remained the same across experiments, except for the applied voltage.

It should be noted that these images had already been collected in a previous project, and were not obtained as a part of the work described herein.

2.2 Experimental setup

To better understand the nature of the data, we give a high-level explanation of the experimental setup used for data collection in figure 2.1.

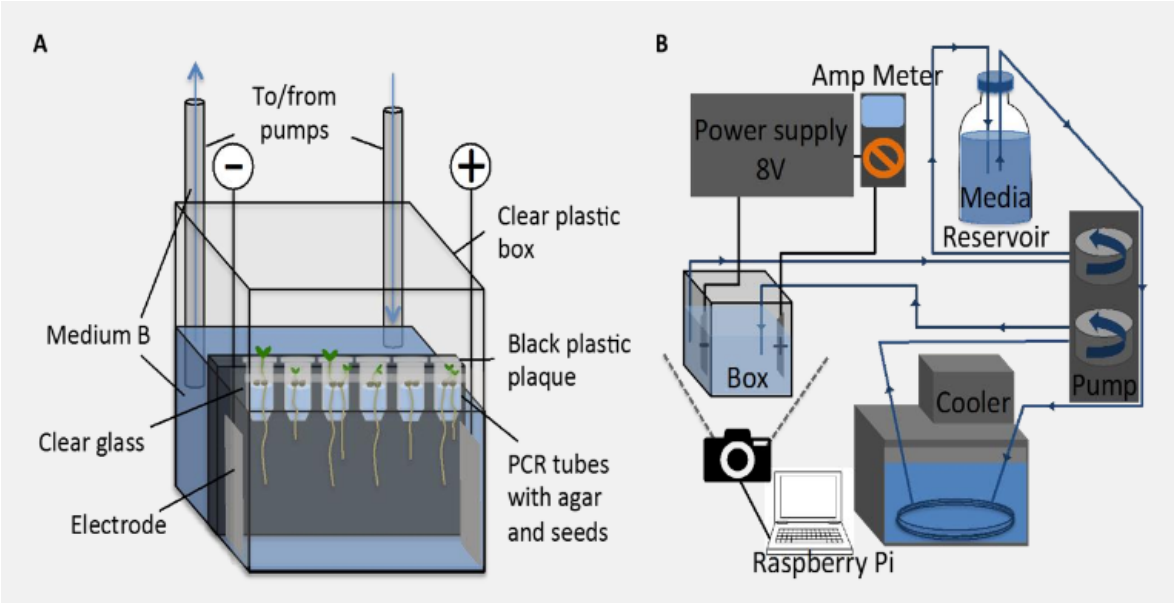

FIGURE 2.1: The experimental setup for root electrotropism: Subfigure A shows a V-box containing plants in a medium, stabilised by PCR tubes with agar and seedlings. To the left a submerged negative electrode (cathode); to the right a submerged positive electrode (anode); inflow and outflow tube controlling the medium. Subfigure B shows the components of an EF experiment: The V-box connected to a power source, a media reservoir connected to the V-box via two pumps controlling the medium inflow and outflow, and a camera controlled via a Raspberry Pi facing the V-box to to take time lapse images of the roots at 10 minute intervals. Taken from [30].

2.3 Image processing

Having these images as a basis, the nature of this work was mainly an image processing one.

Image processing pools together a lot of different domains including physics (optics), signal processing and pattern recognition/ Machine Learning (ML); the idea behind it is to extract information from an image in a form that is suitable for computer processing [4].

Figure 2.2 shows fundamental steps in image processing, that served as a guideline for the presented work. However, every single image processing case is different and there will always be slight variations to this generic image processing workflow.

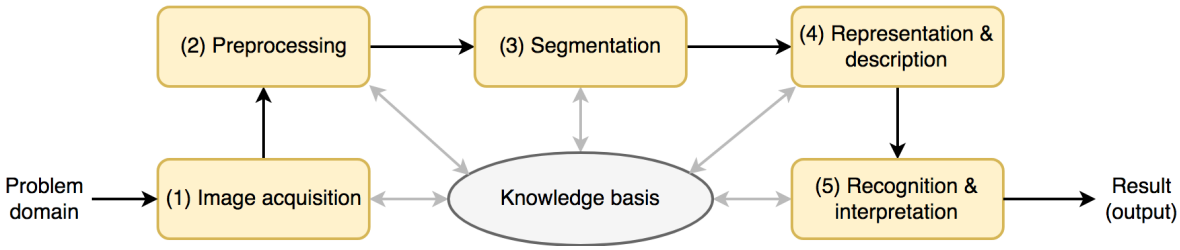

FIGURE 2.2: Image processing steps: (1) Digital image acquisition; (2) Preprocessing: To improve the image in ways that increase the chances for success of the other processes; (3) Segmentation: To partition an input image into its constituent parts or objects; (4) Representation and description: To convert the inut data to a form suitable for computer compressing, to extract features that result in some quantitative information of interest or features that are basic for differentiating one class of objects from another; (5) Recognition and interpretation: Understanding, to assign a label to an object based on the information provided by its descriptors, to assign meaning to an ensemble of recognised objects. All steps interact with the knowledge about the problem domain. Adapted from [6].

2.3.1 Digital images

In image processing, we operate on digital (discrete) images. An image refers to a 2D light intensity function $f(x,y)$, where (x,y) denote spatial coordinates and the value of f at any point (x,y) is proportional to the brightness or gray levels of the image at that point [6]. Thus, a digital image is an image $f(x,y)$ that has been discretised both in spatial coordinates and brightness, i.e. we

- Sample the 2D space on a regular grid
- Quantise each sample, i.e. round to nearest integer.

What we get is an image represented as a matrix of integer values; the elements of such a digital array are called image elements or pixels.

2.3.2 On spatial resolution, gray levels and coloured images

The storage and preprocessing requirements increase rapidly with the spatial resolution and the number of gray levels. For instance, a 256 grayscale image of size 256×256 occupies 64K bytes of memory. Figure 2.3 shows images of different spatial resolution.

FIGURE 2.3: Images of different spatial resolution: Decreasing the spatial resolution by a half in each step, starting from the top left corner to the bottom middle image. Images of very low spatial resolution produce a "checkerboard effect". Taken from [6].

Since we deal with coloured images, we chose the RGB colour model which is an additive colour model in which red, green and blue light are added together in various ways to reproduce a broad spectrum of colours [16].

2.3.3 Image processing operations

An image processing operation typically defines a new image g in terms of an existing image f. We can:

- transform the range of f

$$g(x,y) = t(f(x,y))$$

- transform the domain of f

$$g(x,y) = f(t_x(x,y), t_y(x,y)).$$

From an image processing point of view, we performed point as well as local operations in our work.

A point operation is a function that is performed on each single pixel of an image, independent of all the other pixels in that image [21]. These include operations like inverting,

changing the brightness or contrast of an image, changing the gamma of an image, binarising an image and logical operations.

Local operations, on the other hand, compute the new value of each single pixel with a neighbourhood around it [10]. They include filters which are operations that convert a source image to a result image by applying some kind of tranformation, be it of linear or nonlinear nature [10]. There are more than 36 different filter functions in the MATLAB Image Processing toolbox [22] that can be applied with different arguments and sensitivity thresholds. Filters used in our work include the 2D median filtering *medfilt2* or the 2D adaptive noise-removal filtering *wiener2* [22].

2.4 *MATLAB* as a programming language

As a language we chose *MATLAB* as it is very popular in academic circles for image/ data processing given the number of built-in functions, and a well documented *MATLAB Image Processing Toolbox* [22].

2.5 Computing the angle

Once the skeleton has been extracted, one can compute the curvature and the angle of the root tip. In fact, we only need to segment the part of the root that is close to the tip including the bending point, i.e. the point with highest local curvature. This will help to discard lots of noise caused by tubes and other objects at the upper part of the root, see subsection 4.2.3 for examples of images.

Various definitions of angles associated to the root tip had been considered; however, in order to make the angles comparable to the so far manually computed angles we chose an emulation of the manual computation of the angle in our final implementation.

The angle Θ we want to compute is the angle between the line through the point of the highest local curvature and the root tip and a line parallel to the electric field (EF). Figure 2.4 illustrates the angle Θ that is computed.

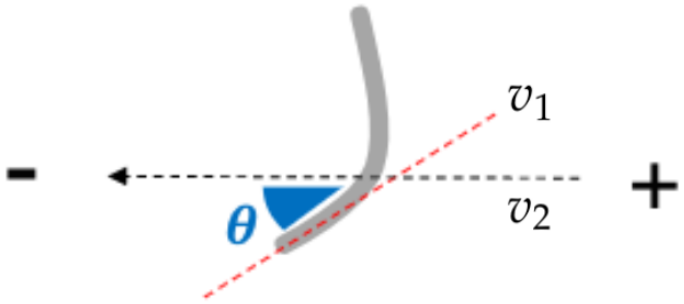

FIGURE 2.4: How the angle in reponse to the EF is measured: The angle Θ represents the angle between the line through the tip of the root and the point of highest local curvature (v_1) and a line parallel to the EF (v_2). Adapted from [30].

Once these two lines are correctly defined, provided the point of highest local curvature is unique, it is straight forward to compute angle Θ.

Let v_1 be the line through the tip of the root and the point of highest local curvature and v_2 a line parallel to the EF. We can compute angle Θ by using simple trigonometric methods

$$\Theta_1 = \cos^{-1} \frac{(v_1 \cdot v_2)}{|v_1| \cdot |v_2|}. \tag{2.1}$$

Assuming the tip of the root only bends positively, i.e. towards the cathode (the negative pole of the EF) in our experimental setup, and v_2 is not positive nor negative, the sign of Θ only depends on v_1. If the root tip crosses the line that is completely aligned with, i.e. parallel to, the EF (where Θ takes the value 0), Θ becomes negative as v_1 will become negative.

2.6 Computing the curvature of the root tip

In order to detect the point of highest local curvature, we need to compute the curvature at each point in the skeleton.

In the following, we will sketch the mathematical framework which describes the curvature of a curve embedded in a plane. This will help the user to understand the rather formula intense implementation computing the analytical curvature using fitted polygons to the points, i.e. the coordinates from the root skeleton.

There exist various different concepts and definitions of curvature in different branches of geometry. Intuitively, curvature in the 2D case captures the amount by which a curve deviates from being a (straight) line. We will only touch upon *extrinsic* curvature here, which is defined for objects embedded in some higher-dimensional, usually Euclidean space. This

allows us to use the radius of circles that touch the object, so-called *osculating circles* (see figure 2.5), to compute the curvature. For the interested reader we refer to Do Carmo's book on differential geometry [9] to learn more about different curvatures.

Geometrically, the curvature measures how fast the unit tangent vector, i.e. the vector tangent to the curve and of length 1, to the curve rotates.

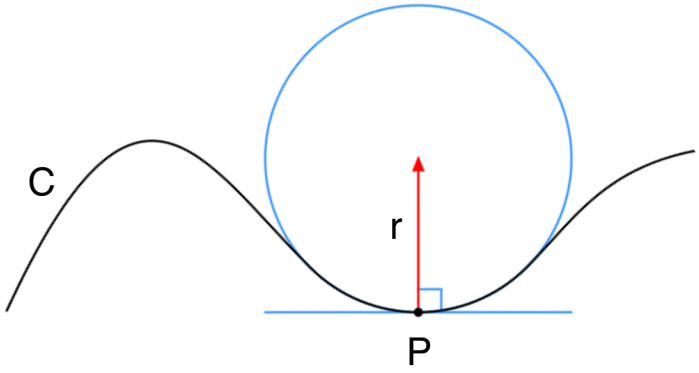

FIGURE 2.5: The curvature and the osculating circle: Let C be a curve and P a point on it. Then there is a unique circle or line which best approximates the curve near C which we call the *osculating circle* at P. We define the curvature of C at P to be the curvature of that circle or line.

If the curve is close to a straight line, and thus the unit tangent vector changes very little, the curvature is small; if the curve contains a sharp turn, the curvature is large. In the implementations we used the difference between edge direction vectors to compute this velocity.

We first compute the first derivative (in between pairs of samples), and we take into account unequal segment lengths. In the same way we then compute the second derivative, we take the average length of two neighbouring edges, i.e. the length of edge in between neighbouring normals if we consider the normals to be halfway between vertices. Once we have the normals, we can compute the rate of change between neighbouring normals to assign the specific curvature value to each point.

If the user does not specify otherwise, we assume the line pieces to connect two neighbouring points. The curvature function outputs a vector of curvature values of the points.

In a previous version, we took a section of the curve (here 9 vertices) and fitted a polynomial to it; this will be our curve model. From this model we can compute the derivative and with the coefficients of the fitted polynomial we can compute the curvature at each point. A large radius of the osculating circle, i.e. a high curvature, means a rather straight part of the curve, while we get a small radius or curvature in parts where the curve is "pointy", i.e.

bends more. The sign of the curvature indicates whether it is right (positive) or left (negative) bending. However, the here used function *polyfit* should not be used on very noisy data. An example of a curvature plot of one of our roots can be seen in figure 2.6.

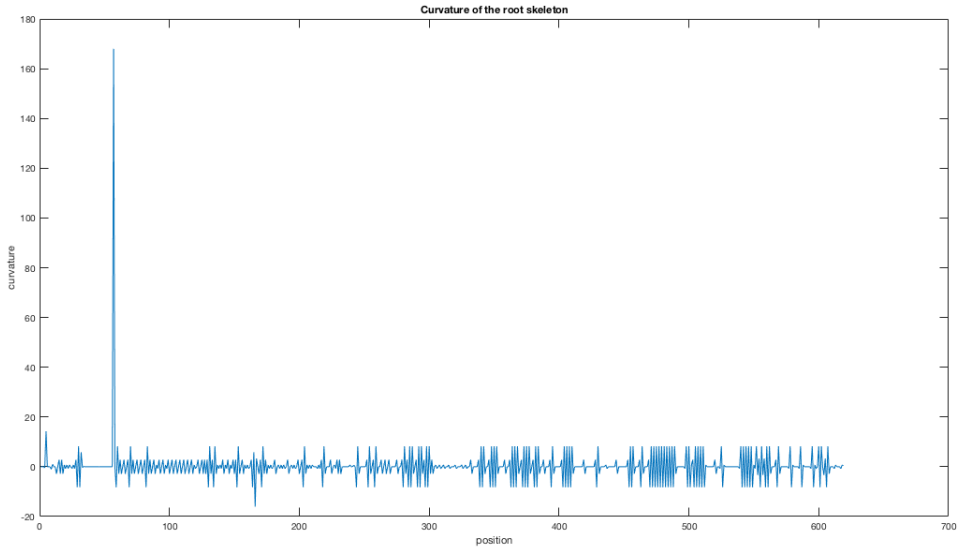

FIGURE 2.6: The curvature of a root skeleton: The peak at position 63 (counted from the root tip) represents the turning point of the root tip; the local (and global) maximum, the positivity of the curvature (ie right-bent) and the position at the tip of the root indicate that this is the point we are looking for. Highly noisy and unreliable pixels close to the tubes had been cut out.

Chapter 3

Results

The following section of this report will briefly explain the tool from a technical perspective, but more importantly we will present some highlights of the tool that sets it apart from other tools. In the appendix B we guide the user through one example to illustrate how the tool is used in practice. This can serve as a manual for users even though the steps are self-explanatory by just using the Graphical User Interface (GUI).

Additionally, we will show some validation of the tool by comparing the angle computed by our tool to the one computed manually on 3 time-series image data sets of Arabidopsis roots.

The code is open-source and publicly available on `https://github.com/GiovanniSena/Auto-angle`; all previous versions including log files can be found on `https://github.com/burfel/root-tip-angle`.

3.1 Workflow of root skeletonisation

The pipeline that has been developed for the preprocessing step of extracting the skeleton of the root is displayed in figure 3.1 and can be viewed at the bottom of the GUI (explained in the next section), see chapter B.

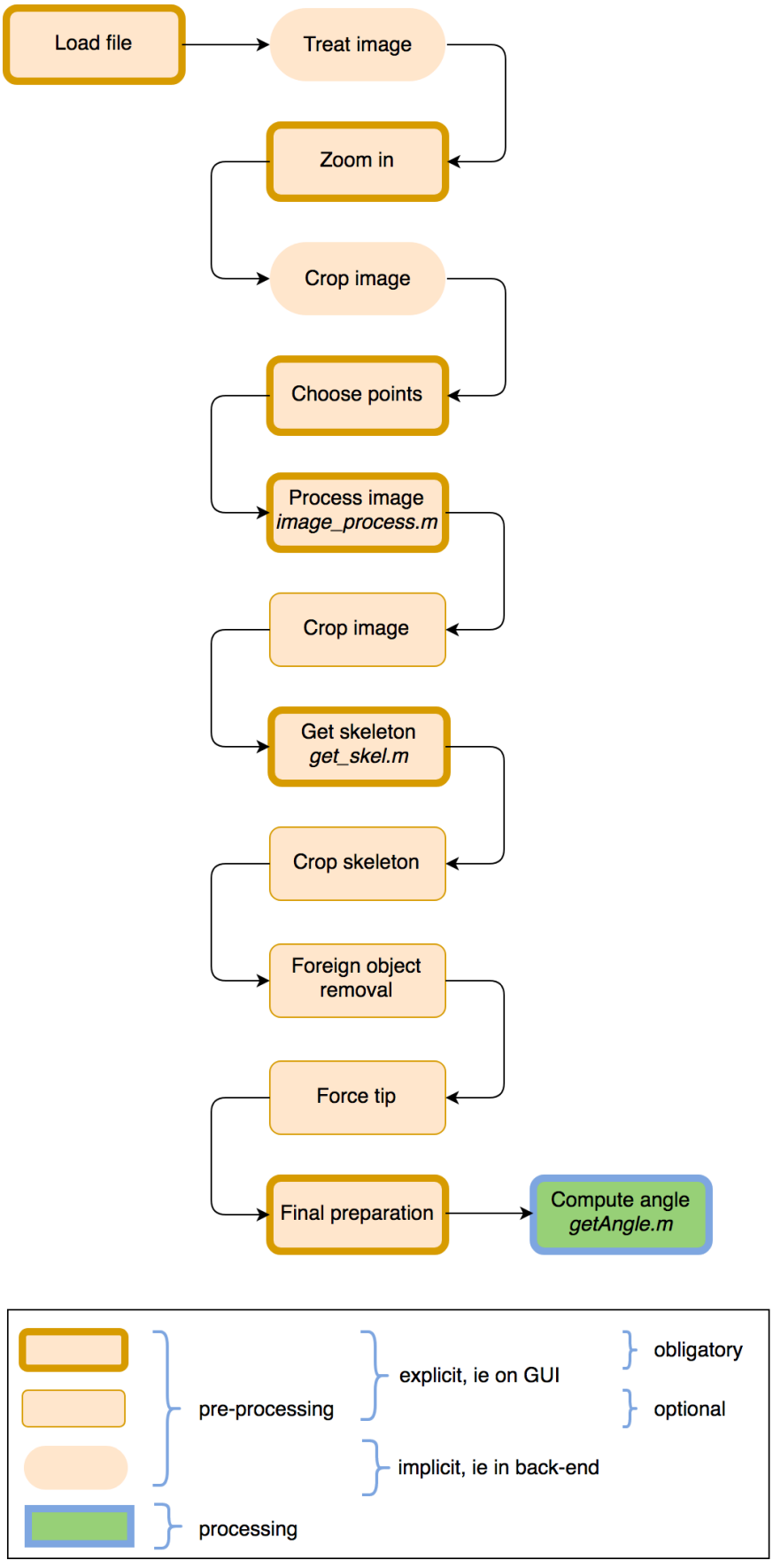

FIGURE 3.1: The workflow of *RootSkel* and different components of the GUI: Pre-processing steps to extract the skeleton are highlighted in orange, the actual angle computation is shaded in blue. Steps that have a front-end, i.e. are visible on the GUI, are framed in red, only back-end components are without a frame.

3.2 Graphical User Interface

To make the program more user-friendly, we developed a graphical user interface (GUI). Pop-up windows, message boxes as well as error boxes, will guide the user through the process (see chapter B); a separate manual is not necessary as the steps are very intuitive and straight-forward. There are various benefits of the GUI such as

- Visualising the process including the pipeline and the angle that is computed
- Easy user interaction with mouse clicking
- Flexibility, e.g. the user can go back at each step without running the whole script from the beginning
- Pop-up windows and mouseover functions on buttons that guide the user through the process and explain the steps
- Error messages if user does not enter allowed values.

With the development of the GUI, we split our source code of the tool based on functionality, i.e. we create separate functions or modules that we then connect to several objects in the GUI. This process, also known as *modularity* in software engineering, has various benefits for developing and maintaining the application: The code is less cluttered but more structured and readable and just by reading the main functions which calls all other subfunctions you get a general overview and understanding of what the code does. Also, it reduces redundancy in the main code if the code gets split up into smaller subfunctions and helps debugging. A clear naming convention also adds to a better understanding of the code.

Here, it allowed us to not only handle front-end and back-end but also the different steps in the pipeline separately:

Every function representing one specific step in the pipeline or objects accessed by various functions is encoded in the back-end which is not visible and relevant for the future user. These modules are connected via callbacks to different objects in the GUI which represent the front-end, see figure 3.2.

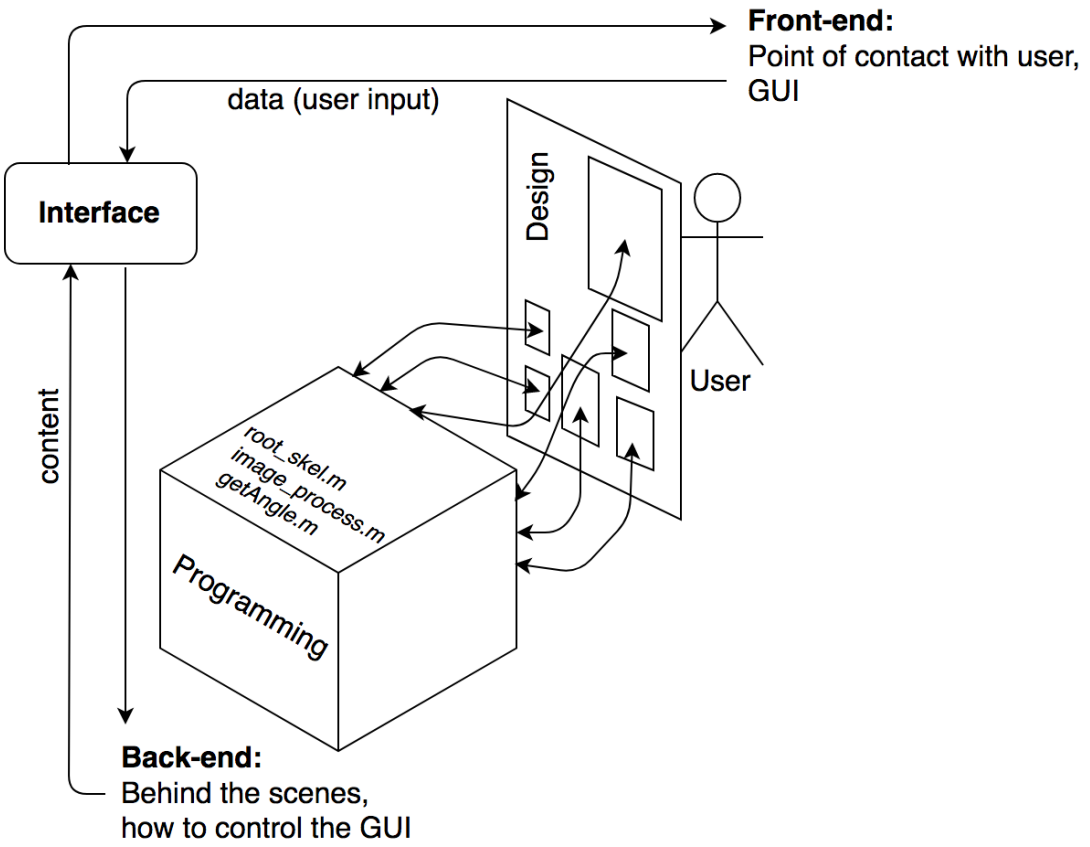

FIGURE 3.2: A scematic figure visualising how back-end and front-end communicate via a well-defined user interface: The back-end containing all the programs acts behind the scenes and controls the GUI; it sends content to the interface that get than transferred to the front-end. The front-end being the point of contact with the user contains the GUI and the design sends the user input data via the interface to the back-end.

Table A.1 gives an overview of the different components and modules our software tool *RootSkel* consists of. It has been developed over a cycle of various iterations, together with an exemplary future user.

3.3 Validation: Comparing the automated calculated angles with the manually calculated angles

As a preview of validation we performed our software tool on 3 randomly chosen roots from 3 different experiments over a time of 330 minutes. We compared both the automatically computed angles with the previously computed manual angle. As a second validation layer we also compared the latter to the computed angles with the turning point as user input.

Figure 3.3 shows the comparison of the angle of interest Θ, both of the manual as well as the automatically computed angle at about 11 times steps over a period of 330 minutes.

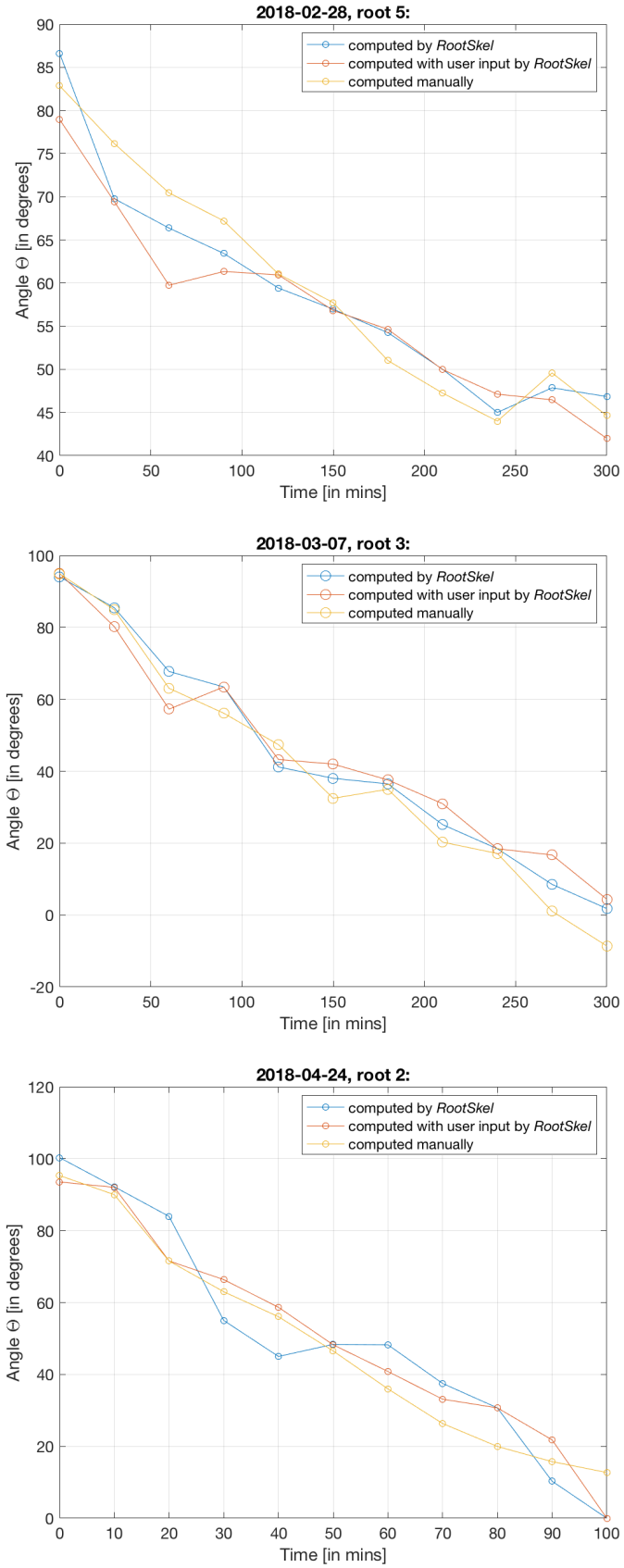

FIGURE 3.3: Comparing the manually computed angle and the angle(s) computed by *RootSkel*; in blue: computed by *RootSkel*, in red: computed by *RootSkel* with the turning point as user input, in yellow: manually computed angle. Each plot is labelled by the data set it was taken from (named after date) and the root number (starting from the left hand side in an image). Computations of all three approaches exhibit similar patterns; the angle Θ continuously decreases.

Differences between the three approaches are in the magnitude of 5 degrees, single points vary up to 12 degrees, see also table 3.1. Table 3.1 compares the automatically computed angle by *RootSkel* with the manually computed angles of the 3 roots. We compute the absolute difference in the angles and introduce an *improvement score* defined as

$$\text{score}_{\textbf{imp}} := \left|\frac{\alpha_{\text{RootSkel}} - \alpha_{\text{manual}}}{\alpha_{\text{manual}}}\right|$$

where α_{RootSkel} denotes the angle computed by *RootSkel* and α_{manual} denotes the manually computed angle; this way we compare to the manually computed angles. How much our method is really an improvement of the theoretical true value, cannot be computed as any measurement will be subject to errors (see also section 4.1). The improvement score on our 3 example data sets average between 5% and 23% with an ensemble average of 12.4%, which can be concluded from table 3.1.

When studying our data sets (manually computed and automatically computed angles), we could observe that as the angle approaches zero, the values get more volatile, i.e. the variance increases; this matches up with our observations presented in figure 3.3. However, it should be noted that our improvement score is based on relative values and, therefore, can easily increase for small values.

2018-02-28, root 5

Time [in mins]	Angle computed by *RootSkel* [in degrees]	Angle computed manually [in degrees]	Absolute difference in angles [in degrees]	Improvement score [in percentage]
0	86.6	82.9	3.8	4.6
30	69.8	76.2	6.4	8.4
60	66.4	70.5	4.1	5.8
90	63.4	67.2	3.8	5.6
120	59.4	61.0	1.6	2.7
150	57.0	57.7	0.8	1.3
180	54.2	51.0	3.2	6.3
210	50.0	47.2	2.8	5.9
240	45.0	44.0	1.0	2.3
270	47.9	49.6	1.7	3.5
300	46.8	44.6	2.2	5.0
AVERAGE or mean	**58.8**	**59.3**	**2.8**	**4.7**
p-value of two sample t-test	**0.93**			

2018-03-07, root 3

Time [in mins]	Angle computed by *RootSkel* [in degrees]	Angle computed manually [in degrees]	Absolute difference in angles [in degrees]	Improvement score [in percentage]
0	94.1	94.9	2.2	2.3
30	85.5	84.9	0.6	0.7
60	67.8	63.0	4.7	7.4
90	63.4	56.1	7.3	13.0
120	41.2	47.4	6.2	13.1
150	38.0	32.5	5.6	17.1
180	36.5	35.0	1.5	4.3
210	25.1	20.3	4.9	24.1
240	18.4	17.1	1.3	7.7
270	8.5	1.1	7.5	704.6
300	1.8	-8.6	10.4	120.9
AVERAGE or mean	**40.2**	**40.3**	**4.7**	**83.2**
AVERAGE without considering the measurements highlighted in yellow	**48**	**50.1**	**3.8**	**9.9**
p-value of two sample t-test	**0.81**			

TABLE 3.1: Comparing the automatically computed angle by *RootSkel* with the manually computed angles on the 3 roots. Average values of the difference in the angles and the improvement score are highlighted by a red margin; values that highly skew the results and are advised to be left out when doing the average calculations are highlighted in yellow. The p-values of a two sample t-test are highlighted in green.

It should be noted that we made an educated estimation of the measurement error, i.e. the total error, to be about about 5% of our computed *RootSkel* value. We round this error to two significant digits according to convention and computed the absolute difference in the angles and the improvement score with this precision; we only round once we present the values to avoid rounding errors.

2018-04-24, root 2

Time [in mins]	Angle computed by *RootSkel* [in degrees]	Angle computed manually [in degrees]	Absolute difference in angles [in degrees]	Improvement score [in percentage]
0	100.3	95.3	5.0	5.2
10	92.2	90.0	2.2	2.5
20	84.0	71.5	12.4	17.3
30	55.0	63.0	8.0	12.7
40	45.0	56.0	11.0	19.7
50	48.4	46.5	1.8	4.0
60	48.2	36.0	12.3	34.1
70	37.5	26.3	11.1	42.3
80	30.7	20.0	10.7	53.7
90	10.3	15.7	5.4	34.5
100	0	12.7	12.7	100.0
AVERAGE or mean	**41.0**	**47.3**	**8.4**	**29.6**
AVERAGE without considering the measurement highlighted in yellow	**45.1**	**52.0**	**8.0**	**22.6**
p-value of two sample t-test	**0.90**			

TABLE 3.2: See table 3.1.

As a validation that the computed values of *RootSkel* are similar enough to the manually computed values, we perform a t-test between the two sets of angles. In all three cases we fail to reject the null hypothesis (at a 95% confidence interval), implying that the difference between each of the two samples is not statistically significant. The p-values of the two sample t-test can be found in tables 3.1 and 3.2.

Chapter 4

Discussion

4.1 Validation on more images

Certainly, validation needs to be done on a bigger data set in order for our claims to be valid. As this could not happen in this project due to time constraints we suggest a thorough (statistical) analysis of the collected data and, subsequently, quantification of population heterogeneity in electrotropic responses as future work.

Obtaining standard deviations in the measurements, both in the manual and the computational method, are necessary to make claims on their precision and therefore robustness of *RootSkel*. Quantifying accuracy is almost impossible without having some underlying model or valid assumptions as we do not know the true theoretical values, and any measurements will inevitably contain errors.

4.2 Challenges and limitations of *RootSkel*

During the development process of *RootSkel* various challenges were encountered and trade-offs to overcome them were necessary; there are some limitations of *RootSkel* that users should be aware of and future developers can work on.

4.2.1 Non-planar roots

Another challenge of studying electrotropism compared to gravitropism in an agar gel is that it is much harder to keep the roots in a plane, i.e. keep them from growing in a third direction, due to the experimental set up to capture electrotropic responses. Since we only take 2D images, there will inevitably be an error in the angle calculation by the simple fact that the plants do not bend in a perfect plane.

4.2.2 Skeletonisation of the root

Probably the biggest challenge in the development process was to extract the root skeleton, precisely to discern the root from the background and the noise, mainly due to the low contrast. Our approach of implementing different colour and intensity filters is not optimal as these might not filter out the noise. The most distinct feature of the root is the tubular structure and functions that recognise this have been used; however, this can still be optimised in the future.

4.2.3 Challenging images

The data set we were working on contained highly challenging images, this was due to

- Low contrast between roots and background; the roots were almost not discernible for the human eye without previous treatment of the image
- Lots of noise or objects that we classified as noise as there was no apparent structure to it
- Different noise patterns across the images; the noise was subtle or hiding and could only made visible on filtered images
- Numerous objects that were of no interest, especially the ones interferring with our object of interest
- Low resolution of the images or object of interest itself taking up only a small fraction of the whole image.

Highly pixeled images made not only the preprocessing but especially the curvature computing part challenging; also various MATLAB functions do not work well on sparse data.

[INSERT 4 EXAMPLES HOW CHALLENGING DATA WAS, MAYBE IN APPENDIX]

4.2.4 Scalability and reproducibility

The variation in the noise pattern in the images made it difficult to automate the process of extracting the root skeleton; every image is unique and needs to be treated uniquely. The developed pipeline works well on many images; other images require special and longer treatment. Scalability has been and continues to be a challenge.

Even though our angle computation is deterministic and straight-forward and only depends on the root skeleton, there is still some variability in the preprocessing step due to the user's input. This means, it can happen that for the very same root in the very same image we get slightly different angles due to slightly different root skeletons. However, the error in the angles we observed is tiny and within reasonable bounds to be negligible; we expect the measurement error to be much smaller than in the manual computations. Having said that, reproducibility can not be fully guaranteed.

4.3 Further work

In the following we will outline suggestions for future work including things that need optimisation or features that might be nice to implement in the future.

4.3.1 Effective noise reduction

In order to tackle the problem of noise reduction, it might be advisable to consult an expert not only on image processing but especially in the field of noise reduction to effectively implement filters that get rid of the noise our images contain.

4.3.2 Less user input up to automatisation

Here in the first version of *RootSkel*, the main goal was to standardise the angle computation. If in the future a method for handling the different noise patterns in the images was efficiently handled which required less user input; it would be desireable to automate the whole angle computation.

4.3.3 Robustness

Many iterations on different images of our data set led to an elaborate tool for the preprocessing of the roots; however, we cannot prevent the tool from failing on some images.

If one wants to make the tool available to a wider public and make it more robust so it will work on other, unseen data (which was not the goal of this work), one might want to collect higher-quality, less noisy data in the future and investigate automated approaches such as adaptive thresholding.

4.3.4 Error quantification in the angle computation

Another additional step towards validation or an integral part of taking measurements in general is to quantify the error in the angle computation, e.g. the maximum or average error in the final result due to the changes in the pre-processing step. One can try to do achieve this empirically by trying our tool on a large amount of data to estimate the statistical error; there are ways to also try to quantify the instrumental error.

4.3.5 Other programming languages

Even though MATLAB has several advantages, it might be worth looking into alternative non-proprietary programming languages to make the tool accessible to a wider public or other options of making the tool portable, i.e. without requiring the user to have MATLAB installed. Alternative languages one might want to consider are *Python* and *Julia*. Another recommended language is *OpenCV* as it is very fast and well documented. Other non-open source software such as *ImageJ*, a Java based image processing program, and *Avizo* which is a general-purpose commercial software application for scientific and industrial data visualisation and analysis with a nice GUI, were not investigated further in this work.

4.4 Broader application of this tool

This tool can be reused for many purposes and is not restricted to root detection. It might also be used for "easier" problems like gravitropism.

Chapter 5

Conclusion

Here we presented *RootSkel* – a novel and stand-alone image-analysis-based software tool developed in MATLAB to compute the curvature and angle at plant root tips in a standardised fashion. It was optimised for noise-intensive electrotropism images and comes with a user-friendly and flexible pre-processing step to extract the skeleton of the root.

The software has been designed using an extensive amount of different filtering techniques optimised on the image data set described in this work and can therefore be used to work with standard images from consumer digital cameras.

Automated image capturing as well as the design of the software presented here both aim to reduce the time-consuming process of the biologist quantifying the root tip curvature manually but more importantly, it standardises the computations and makes the results reproducible and comparable over a large amount of data.

It offers the possibility to extend the analysis by including different angle definitions to capture the curvature of the plant root and compare them. Also, the software tool is not limited to computing the angle of root tips but due to the flexibility of the pre-processing step can be used for any other curved or polynomial-like structure.

The robustness and reproducibility of *RootSkel* need to be further validated; suggestions for future work can be implemented.

We hope *RootSkel* will contribute to the understanding of highly complex and poorly-understood phenomenons like eletrotropism in plants and possibly other tropisms.

Bibliography

[1] Abas, L. et al. “Intracellular trafficking and proteolysis of the Arabidopsis auxin-efflux facilitator PIN2 are involved in root gravitropism”. In: *Nature cell biology* 8.3 (2006), p. 249.

[2] Basu, P. et al. “A novel image-analysis technique for kinematic study of growth and curvature”. In: *Plant physiology* 145.2 (2007), pp. 305–316.

[3] Beemster, G. T. and Baskin, T. I. “Analysis of cell division and elongation underlying the developmental acceleration of root growth in Arabidopsis thaliana”. In: *Plant physiology* 116.4 (1998), pp. 1515–1526.

[4] Byrne, J. “Digital Image Processing: Introduction and Applications”. Presentation at 46th Meetup (3 July 2018). PyData London.

[5] Canny, J. “A computational approach to edge detection”. In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), pp. 679–698.

[6] Chan, C. Y. H. “Image and Audio Processing”. Lecture and slides (15 July 2018). The Hong Kong Polytechnic University. URL: http://www.eie.polyu.edu.hk/~enyhchan/a_iap18.htm.

[7] Chavarría-Krauser, A. et al. “Spatio-temporal quantification of differential growth processes in root growth zones based on a novel combination of image sequence processing and refined concepts describing curvature production”. In: *New Phytologist* 177.3 (2008), pp. 811–821.

[8] Comaniciu, D., Ramesh, V., and Meer, P. “Real-time tracking of non-rigid objects using mean shift”. In: *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*. Vol. 2. IEEE. 2000, pp. 142–149.

[9] Do Carmo, M. P. *Differential Geometry of Curves and Surfaces: Revised and Updated Second Edition*. Courier Dover Publications, 2016.

[10] Easton, R. "Basic Principles of Imaging Science II". Lecture and slides (Winter 2004-2005). Rochester Institute of Technology. URL: https://www.cis.rit.edu/class/simg712-90/.

[11] French, A. et al. "High-throughput quantification of root growth using a novel image-analysis tool". In: *Plant physiology* 150.4 (2009), pp. 1784–1795.

[12] French, A. P. et al. "A Probabilistic Tracking Approach to Root Measurement in Images-Particle Filter Tracking is used to Measure Roots, via a Probabilistic Graph." In: *BIOSIGNALS (1)*. 2008, pp. 108–115.

[13] Ge, Z., Rubio, G., and Lynch, J. P. "The importance of root gravitropism for inter-root competition and phosphorus acquisition efficiency: results from a geometric simulation model". In: *Plant and Soil* 218.1-2 (2000), pp. 159–171.

[14] Gilroy, S. "Plant tropisms". In: *Current Biology* 18.7 (2008), R275–R277.

[15] Hodgson, C. "Geometry". Lecture and lecture notes (Winter 2016). University of Melbourne.

[16] Ibraheem, N. A. et al. "Understanding color models: a review". In: *ARPN Journal of science and technology* 2.3 (2012), pp. 265–275.

[17] Ishikawa, H and Evans, M. "Novel software for analysis of root gravitropism: comparative response patterns of Arabidopsis wild-type and axr1 seedlings". In: *Plant, Cell & Environment* 20.7 (1997), pp. 919–928.

[18] Ishikawa, H. and Evans, M. L. "Electrotropism of maize roots: role of the root cap and relationship to gravitropism". In: *Plant physiology* 94.3 (1990), pp. 913–918.

[19] Kimura, K., Kikuchi, S., and Yamasaki, S.-i. "Accurate root length measurement by image analysis". In: *Plant and Soil* 216.1-2 (1999), pp. 117–127.

[20] Kiss, J. Z., Wright, J. B., and Caspar, T. "Gravitropism in roots of intermediate-starch mutants of Arabidopsis". In: *Physiologia Plantarum* 97.2 (1996), pp. 237–244.

[21] Maini, R. and Aggarwal, H. "A comprehensive review of image enhancement techniques". In: *arXiv preprint arXiv:1003.4053* (2010).

[22] MathWorks. *Image Processing Toolbox R2018a*. URL: https://uk.mathworks.com/help/images/. [Accessed 30 August 2018].

[23] Michener, H. D. "The action of ethylene on plant growth". In: *American Journal of Botany* 25.9 (1938), pp. 711–720.

[24] Miller, N. D., Parks, B. M., and Spalding, E. P. "Computer-vision analysis of seedling responses to light and gravity". In: *The Plant Journal* 52.2 (2007), pp. 374–381.

[25] Mullen, J. L., Ishikawa, H., and Evans, M. L. "Analysis of changes in relative elemental growth rate patterns in the elongation zone of Arabidopsis roots upon gravistimulation". In: *Planta* 206.4 (1998), pp. 598–603.

[26] Naeem, A. et al. "High-throughput feature counting and measurement of roots". In: *Bioinformatics* 27.9 (2011), pp. 1337–1338.

[27] Oliver, N. "An Investigation into the Genetic Mechanisms Involved in Root Electrotropism in Arabidopsis thaliana". MRes Molecular and Cellular Biosciences Spring Research Project (2017). Imperial College London.

[28] Parry, G. et al. "Novel auxin transport inhibitors phenocopy the auxin influx carrier mutation aux1". In: *The Plant Journal* 25.4 (2001), pp. 399–406.

[29] Qi, X., Qi, J., and Wu, Y. "RootLM: a simple color image analysis program for length measurement of primary roots in Arabidopsis". In: *Plant Root* 1 (2007), pp. 10–16.

[30] Sena, G. "Measuring effects of electrotropism in Arabidopsis roots". Presentation (19 June 2018). Imperial College London.

[31] Swarup, R. et al. "Root gravitropism requires lateral root cap and epidermal cells for transport and response to a mobile auxin signal". In: *Nature cell biology* 7.11 (2005), p. 1057.

[32] Van Der Laan, P. A. "Der Einfluss von Aethylen auf die Wuchsstoffbildung bei Avena und Vicia". In: *Recueil des travaux botaniques néerlandais* 31.3/4 (1934), pp. 691–742.

[33] Vandenbrink, J. P. et al. "Light and gravity signals synergize in modulating plant development". In: *Frontiers in plant science* 5 (2014), p. 563.

[34] Walter, A et al. "Spatio-temporal dynamics of expansion growth in roots: automatic quantification of diurnal course and temperature response by digital image sequence processing". In: *Journal of experimental botany* 53.369 (2002), pp. 689–698.

[35] Wang, L. et al. "HYPOTrace: image analysis software for measuring hypocotyl growth and shape demonstrated on Arabidopsis seedlings undergoing photomorphogenesis". In: *Plant physiology* 149.4 (2009), pp. 1632–1637.

[36] Weele, C. M. van der et al. "A new algorithm for computational image analysis of deformable motion at high spatial and temporal resolution applied to root growth. Roughly uniform elongation in the meristem and also, after an abrupt acceleration, in the elongation zone". In: *Plant physiology* 132.3 (2003), pp. 1138–1148.

[37] Wolverton, C et al. "Two distinct regions of response drive differential growth in Vigna root electrotropism". In: *Plant, cell & environment* 23.11 (2000), pp. 1275–1280.

Appendix A

Detailed high-level description of the components of *RootSkel*

Different components of RootSkel

Component	File name	Description
1	Main script	Main file that calls all subfunctions via the internal *MATLAB* callback
	Root_image_GUI.m	
2	Log files	Log files documenting all changes including dates so changes can be undone and future developers can build upon the existing version
	Log.txt	Since the last version
	Old_Versions_log.txt	All previous versions
	CurrentVersion.txt	A shorter version of previous log files including bug fixes
3	Functions	Folder containing the 18 subfunctions
	var_saver.m	★ Creates a variable *varnames* which contains the names of the relevant variables (*skelmatR, skelmatR_simp, max_curv_point, savename*); it then pulls them from the base workspace and lets the user save them in a .mat file. ★ *skelmatR* or *skelmatR_simp* include the skeleton of the root (their x and y coordinates), *max_curv_point* includes the user's input for the possible turning point or an empty set, *savename* includes the name of the image (date and hour) and the number of the roots which is used for names of figures, first column in .csv file and default of *var_saver.m*
	var_loader.m	★ Allows the user to load the .mat files including the relevant objects from the workspace ★ Contains the enabling of appropriate angle calculation buttons; buttons are disabled to avoid bugs and errors (eg angle computation on nothing should not work)
	skel_crop.m	★ Contains the optional free hand cropping of the skeleton
	skel_clean.m	★ Loops on optional additional cleaning, ie bigger and bigger objects are removed, until user is satisfied
	savename_crea.m	★ Saves the label of the root or root number the user chooses in order to keep track of which root is analysed ★ Combines the label with the name of the file and saves it as a folder where the variables (see above) would go
	root_skel.m	★ Takes results from *image_process.m* ★ Applied more fine-tuned filtering ★ Applies more cleaning steps ★ Tries to makes sure that the root tip is in the skeleton ★ Combines approach 1 and 2 ★ Returns the skeleton
	point_get.m	★ Asks the user for points as long as she does not provide the required number (defined as a number of points between minimum and maximum) ★ The user's input is stored in the strings *srcx* and *srcy* are strings with the name of the variable that will receive the data in the base workspace; they tell *assignin* in which variable in the caller to store the data
	point_choose.m	★ Collects the necessary points from the user: 5 points close to the tip, 5 - 10 evenly spaced points on the desired root starting with the tip, the tip of the root ★ Each step can be redone
	image_zoom.m	★ Inverts the image ★ Lets the user zoom in (and zoom out via right click)
	image_process.m	★ Extracts the cropped image ★ Extracts the colours from the sample pixels and averages it with a certain neighbourhood (3x3) ★ Takes a brightness range, an average of the three filters used ★ Approach 1: Colour separation filtering • based on RGB values of points • gray scales image ★ Approach 2: Brightness filtering (intensity-based approach) • enhances brightness • eliminates too bright spots
	image_crop.m	★ Optional free hand cropping
	image_choose.m	★ Allows the user to choose an image ★ Modifies the image using various filter to help the user discern the root
	getAngle.m	★ Takes the skeleton as input ★ Computes the curvature and angle of the root tip
	force_tip.m	★ Prompts user to create an open polygon between the edge of the current skeleton and the tip ★ In order to make sure that the tip of the root is definitely included in the skeleton
	final_prep.m	★ Extracts only the tip of the root and the respective x and y values which are passed on to *getAngle.m* ★ User can choose to select the turning point, ie point with highest local curvature, which can serve as another verification of the computed turning point; it does not have to be exactly on the root as the point in the skeleton that is closest to the chosen point is used
	fig_saver.m	★ Saves the relevant objects upon clicking different buttons
	fig_loader.m	★ Loads respective figures
	angle_file.m	★ Creates a .txt file containing the label of the root (picture name and root number) and the angle ★ Creates a file *root_angles.csv* or *user_assisted.csv* depending on *user_flag* and prints the label of the root and the angle; this file can be appended for consecutive angle calculations of the same root in other images

TABLE A.1: The different components and modules of *RootSkel* with a description of each of them; the most important ones containing the core functionality of *RootSkel* are framed in red, the high-level components are shaded in yellow. A version of an increased font size can be found on https://github.com/burfel/root-tip-angle/blob/master/report/Figures/components.pdf.

Appendix B

Manual of *RootSkel*

[INSERT MANUAL HERE]

B.1 Key features

Figure 3.1 explains the key components of *RootSkel* to address the problem of highly noisy electrotropism consumer camera images of Arabidopsis roots and a standardised way of computing the angle for the curved root tip.

This tool takes the form of a MATLAB program and subprograms with a graphical user interface on top of it.

B.1.1 Handling user's mistakes

When we take the user's input, e.g. choosing samples along the root, we correct for small mistakes by taking a neighbourhood (3×3) average around the pixel. This means the user does not have to take special care when choosing the points as long as it is in the approximate region of the root.

B.1.2 User interaction and optional steps

The software tool was created in ways that it is easy to interact with for a future user. We implemented several optional steps that only need to be performed if the user thinks they are necessary. This on the one hand saves time in the preprocessing but on the other hand also ensures that tricky roots can be tackled by various optional steps in order to extract a skeleton.

B.1.3 Drawing the angle

The GUI lets the user visualise the angle that is computed. This not only helps to make the tool more visual and transparent, but can also assist in debugging.

Appendix C

On the data set

The data set used can be found on https://imperialcollegelondon.app.box.com/folder/50485327378.

C.1 Suggestions for future data acquisition

We will give some suggestions on how to improve the quality of the images in the future.

C.1.1 Stable conditions in the experiment

In general, one should ensure to keep the conditions over one experiment stable as far as possible. It should be noted that this is far harder to achieve in a dynamic system as the one to capture electrotropism compared to a gravitropism setup; the latter usually being done in a stabilising agar gel. This includes a constant number of tubes and roots, constant water level or surface and no external movement to the roots or the tubes to keep the noise distribution as constant as possible. Roots that are affected by change in reflections or illumination (e.g. by people walking into the room) have been proven to be very hard to handle and are often lost in the preprocessing step. Keeping the medium clean of dirt and bubbles as far as possible will also improve the preprocessing step.

C.1.2 No objects interfering with the object of interest

The experimentalist should make sure that there are no objects interfering with the object of interest, be it other roots or any other similar looking objects that are hard to discern even by eye, throughout the experiment.

C.1.3 Increase of the contrast

It is advisable to work towards achieving a higher contrast in the images, so roots can be clearly distinguished from the background, also by eye.

C.1.4 Higher resolution

As the highly pixeled nature of the images did cause problems also in the angle computation, it is recommendable to try to increase the resolution of the images. We could also try to zoom into the roots or even one single root so our actual object(s) of interest take a bigger fraction in the images instead of spending resolution on things that are of no interest.

It might also be the compression step after taking the images that might cause or contribute to the low resolution of the images. One could attempt to use other formats to save the images.

As a last suggestion, different cameras could be compared to see if it does have an effect on the quality of the images.

Appendix D

Literature review