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

August 30, 2018

Word count: INSERT #WORDS HERE

"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 condition.
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 [DO I HAVE TO EXPLAIN IT HERE?] 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; statistics on our data sets suggest that our tool improves the accuracy of the manually computed angle 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 in 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 Figures

List of Abbreviations

Arabidopsis	***Arabidopsis** thaliana*	
EF	**E**lectric **F**ield	
GUI	**G**raphical **U**ser **I**nterface	
ie	*latin* **i**d **e**st	that is

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* [ADD TO GLOSSARY], which is used to indicate the turning movement of a biological organism, here of a plant, when exposed to different environmental simuli [INSERT REFERENCE]. Usually the stimulus involved is added to the name, eg *phototropism* as a reaction to sunlight; it can be either *positive*, ie towards the stimulus, or *negative*, ie away from the stimulus. Various types of tropism are shown in figure [INSERT REFERENCE HERE].

The most frequently observed and best studied tropism in *gravitropism*, which describes the process of how plants grow as a response to gravity. It was firstly scientifically documented by Charles Darwin [INSERT REFERENCE] and can be observed in higher and many lower plants as well as other organisms [INSERT REFERENCE, INSERT FIGURE DIFFERENT TROPISMS FROM G SLIDES]: Roots show *positive gravitropism*, ie they grow in the direction of the gravitational pull whereas stems grow in the opposite direction, see figure [INSERT REFERENCE HERE].
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.

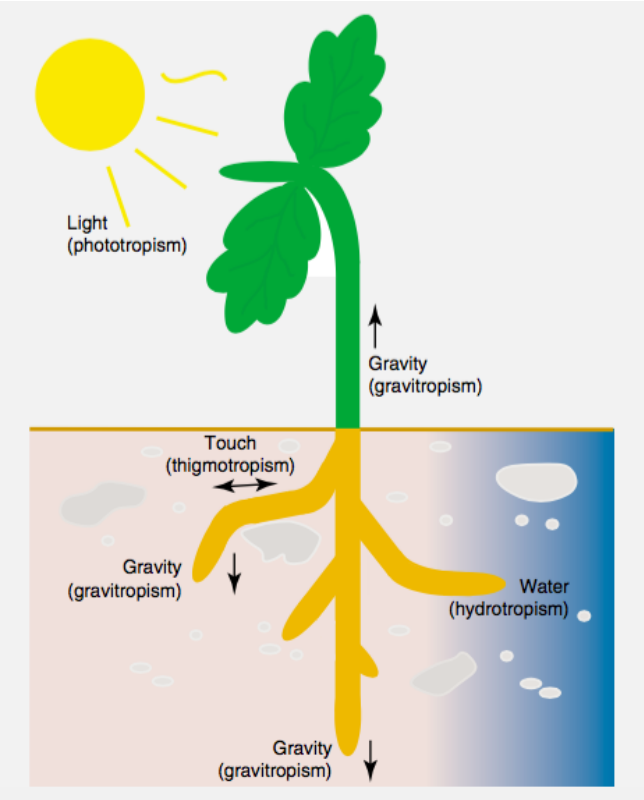

FIGURE 1.1: Different forms of tropism: Phototropism, thigmotropism, hydrotropism, and different effects of gravitropism; taken from [INSERT REFERENCE].

A high-overview explanation of the experimental setup and data collection to study electrotropism can be found in section [INSERT REFERENCE TO SECTION METHOD].

1.2 Literature review

The majority of traditional root development bioassays only consider a small number of points [REFEREENCE PERRY ET AL, 2001 AND MAIN ROOTTRACE PAPER]. They are informative in terms of long-term effects on root growth; however, small and temporary changes can not be captured [REFERENCE MAIN ROOTTRACE]. Recently developed tools have considered a higher number of time points which allows a better study of how the growth process develps and the plant's responses [REFERENCE MAIN ROOTTRACE; ISHIKAWA AND EVANS, 1997; VAN DER WEELE ER AL, 2003; CHAVARRIA-KRAUSER ET AL, 2007; MILLER ET AL, 2007].

Image sequences can contain a representative "snapshot" description of a plant's developmental stage; also they can be generated at high speed [REFERENCE TO EXPERIMENT SETUP IN CHAPTER ....]. From manual time-lapse photography [REFERENCE VAN DER LAAN, 1934; MICHENER, 1938] 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 [REFERENCE MAIN ROOTTRACE]. Once the root growth changes have been extracted from the imge 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 [REFERENCE MAIN ROOTTRACE]. This will help to understand root development in general.

Analysing image data manually, however, is time-consuming, very subjective and thus error-prone [REFERENCE MAIN ROOTTRACE]. 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 [REFERENCE ROOTTRACE].

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 figure [INSERT REFERENCE HERE].

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, what programming languages, definitions and methods 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, ie every 10 minutes a photo was taken. The images were named after the date and time they were taken; information on the strength of the EF in voltage was captured in each folder containing one experiment data set. All the other parameters except for the applied voltage remain the same (eg media, temperature, reservoir size). The specifications are available on request.

It should be noted that these images had already been collected in a previous project and are not subject but only the basis of the work presented here.

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 [INSERT REFERENCE HERE].

[SEE SLIDES, THESIS]

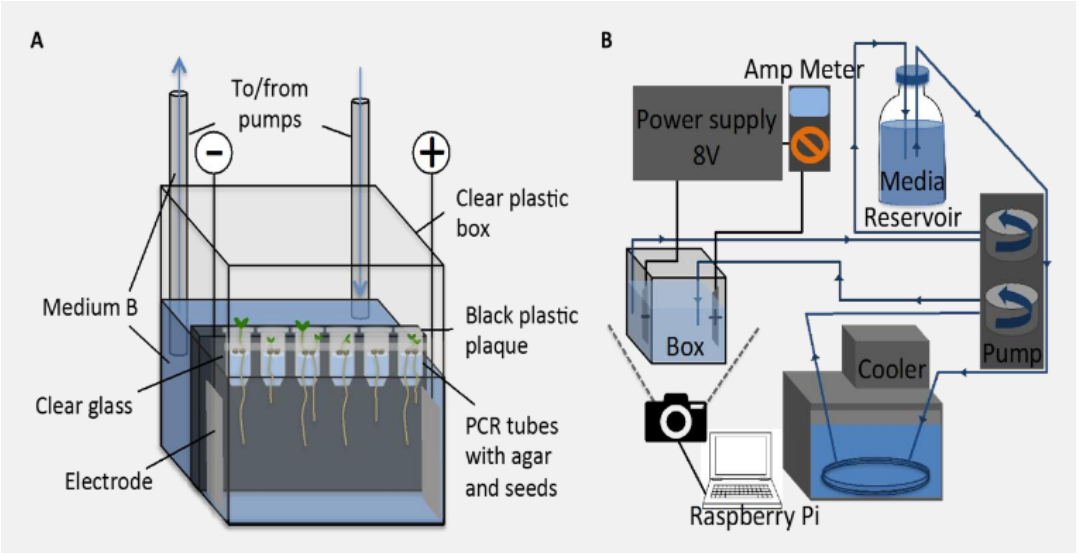

FIGURE 2.1: The experimental setup for root electrotropism: Subfigure A shows a V-box containing plants in a medium, stabilised by PCR tubes with agar and seeds. To the left a submerged negative electrode (cathode); to the right a submerged positive electrode (anode); inflow and outflow tube controlling 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 intervals. Taken from [INSERT REFERENCE].

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 information in a form that is suitable for computer processing [INSERT REFERENCE HERE].

Figure [INSERT REFERENCE HERE] 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.

[INSERT FLOWCHART FIGURE HERE, adapted from....]

2.3.1 Digital images

In image processing, we operate on digital (discrete) images. An image refers to a 2D light intesity 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 [INSERT REFERENCE HERE]. Thus, a digital image is an image $f(x, y)$ that has been discretised both in spatial coordinates and brightness, ie we

- Sample the 2D space on a regular grid
- Quantise each sample, ie 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.

[INSERT EXAMPLE PICTURE HERE]

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 gray-scale image of size 256×256 occupies 64K bytes of memory. Figure [INSERT REFERENCE HERE] shows images of different spatial resolution.

[INSERT FIGURE: Images with decreasing spatial resolution, taken from .....]

Also, an insufficient number of gray levels in smooth areas of a digital image may result in false contouring, see figure [INSERT REFERENCE HERE].

[INSERT FIGURE: Images of different gray-level quantisation]

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

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.

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 [INSERT REFERENCE HERE]. These include operations like inversing, 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 [INSERT REFERENCE HERE]. They include filters which are operations that converts a source image to a result image by applying some kind of tranformation, be it of linear or nonlinear nature [INSERT REFERENCE HERE]. There are more than 134 different filter functions in the MATLAB Image Processing toolbox 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*.

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, including well documented image processing toolbox (*MATLAB Image Processing Toolbox* [INSERT REFERENCE HERE]).

2.5 Pre-processing: Getting the root's skeleton

The goal of the preprocessing step was to extract the skeleton, ie the coordinates of the pixels that represent the root on the image. This task turned out to be highly challenging on noisy image data and can be regarded as the actual achievement of the presented work. The highly elaborate pipeline of the preprocessing step can be found in chapter [INSERT REFERENCE HERE].

2.6 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 bit of the root that is close to the tip including the bending point, ie the point with highest local curvature, not the entire root. This will help to discard lots of noise caused by tubes and other objects at the upper part of the root, see section [INSERT REFERENCE] for example 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 (Gaussian) curvature and the root tip and a line parallel to the electric field (EF).

Figure [INSERT REFERENCE HERE] illustrates the angle Θ that is computed.

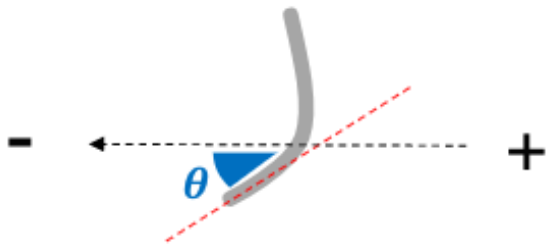

FIGURE 2.2: How the angle in reponse to the EF is measured: The angle Θ represents the angle between the line through the tip of the root and the point of highest local curvature and a line parallel to the EF; taken from [INSERT REFERENCE].

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

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

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

or equivalently

$$\Theta_2 = \tan^{-1}(v_1, v_2) \times \frac{180}{\pi} \tag{2.2}$$

depending on the signedness of v_1 and v_2. Assuming the tip of the root only bends positively, ie towards the cathode (the negative pole of the EF) in our experiment 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, ie parallel to, the EF (where Θ takes the value 0), Θ becomes negative as v_1 will become negative.

2.6.1 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, ie 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 cirlce*, to compute the curvature.

Geometrically, the curvature measures how fast the unit tangent vector, ie the vector tangent to the curve and of length 1, to the curve rotates. [INSERT FIGURE HERE] 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 implementation we will use the difference between edge direction vectors to compute this velocity.

For the implementation we first compute the first derivative (in between pairs of samples); 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, ie 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.

[INSERT FIGURE HERE, DRAW IT]

We assume the line pieces to connect two neighbouring points if the user does not specify otherwise.

The script outputs the curvature values of each point.

In a previous version, we took a section of the curve (here 9 vertices) and fitted a polynomial to it; this will be our cuve 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 curvature means a rather straight part of the curve, while we get a small radius of curvature in parts where the curve is "pointy", ie 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.

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 and we will guide the user through one example to illustrate how the tool is used in practice.

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

The code is open-source and publicly available on [INSERT GITHUB REFERENCE HERE]; all previous versions including log files can be found on [INSERT GITHUB REFERENCE HERE]. [EMPHASISE THIS: MAKE RESEARCH TRANSPARENT]

3.1 Key features

Figure [INSERT REFERENCE HERE: PIPELINE FROM GUI] explains the key components of this image-analysis software tool 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 (GUI) on top of it.

3.1.1 Graphical User Interface

To make the program more user-friendly, we developed a graphical user interface (GUI). Pop-up windows will guide the user through the process (see [INSERT REFERENCE HERE]); 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
- Easy user interaction with mouse clicking
- Flexibility, eg the user can go back at each step without rerunning 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 right values.

3.1.2 Discerning root from background

Gamma correction (imadjust) [TO COMPLETE]

3.1.3 Handling user's mistakes

When we take the user's input, eg choosing samples along the root, we correct for small mistakes by taking an neighbourhood ($3 \times$) 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.

3.1.4 User interaction and optional steps

The software tool was created in a 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 it is necessary. This on the other 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.

3.1.5 Drawing of 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.

3.2 Workflow of root skeletonisation – explained on GUI

The pipeline that has been developed for the preprocessing step of extracting the skeleton is displayed in figure [INSERT REFERENCE HERE] and can be viewed at the bottom of the GUI, see figure [INSERT REFERENCE HERE]. In the appendix [INSERT REFERENCE HERE] we present the tool on one example image guides the reader through the pipeline and can serve as a manual for future users.

3.3 Components of the GUI

With the development of the GUI for more user-friendliness, we split our source code of the tool based on functionality, ie 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 codes get split up into smaller subfunctions and helps debugging. Speaking variables also add to a better understanding of the code.

[INSERT FIGURE FRONT-END BACK-END HERE]

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.

Table [INSERT REFERENCE HERE] 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. The current version of the package can be downloaded from

Even though most of image processing work is empirical, ie based on trial and error and educated guesses on the given data set, we try to explain why certain features were necessary to implement. The pipeline can be seen in the at the bottom of the GUI in figure [INSERT REFERENCE HERE].

[INCLUDE IN GRAPHIC IF EXCEED WORD COUNT]

The GUI includes helpful text along the pipeline as well as pop up windows including instructions or warnings in case the user does not follow them.

In the appendix [INSERT REFERENCE HERE] we guide the user through the different steps of the pipeline and present one example each; this can serve as a manual for users even though the steps are rather self-explanatory by just using the GUI. Also, having a GUI rather than a script with pop-up-windows not only makes the tool more visually attractive but it one can also repeat a step without restarting the whole process or script again. This all adds to the user-friendliness.

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

As a preview of validation we performed our webtool on 3 randomly chosen roots from 4 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 step we also compared the angles to the computed angles with the turning point as user input.

Figure [INSERT HERE] shows the comparison of the angle of interest *Theta*, both of the manual as well as the automatically computed angle at about 11 times steps over a period of 330 minutes.

Differences between the three approaches are in the magnitute of 5 degrees, single points vary up to 12 degrees, see also table [INSERT REFERENCE HERE]. Table [INSERT REFERENCE HERE] compares the autmoatically computed angle by

RootSkel with the manually computed angles on the 3 roots. We compute the absolute difference in the angles and introduce an *improvement score* defined as

$$\left|\frac{\alpha_{\text{RootSkel}} - \alpha_{\text{manual}}}{\alpha_{\text{manual}}}\right|$$

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. For how much our method is really an improvement of the theoretical true value, that cannot be computed as any measurement will be subject to errors, we refer to the discussion section [INSERT REFERENCE HERE]. The improvement score on our 3 example data sets average between 5% and 23% with an ensemble average of 12.4%.

When studying our data sets (manually computed and automatically computed angles), we could oberve that as the angle approaches zero, the values get more volatile, ie the variance increases; this matches up with our observations in presented in table [INSERT REFERENCE HERE]. However, it should be noted that our improvement score is based on relative values and, therefore, can easily blow up for small values (of manually computed angles and increasing or constant difference/ variance to the automatically computed angle).

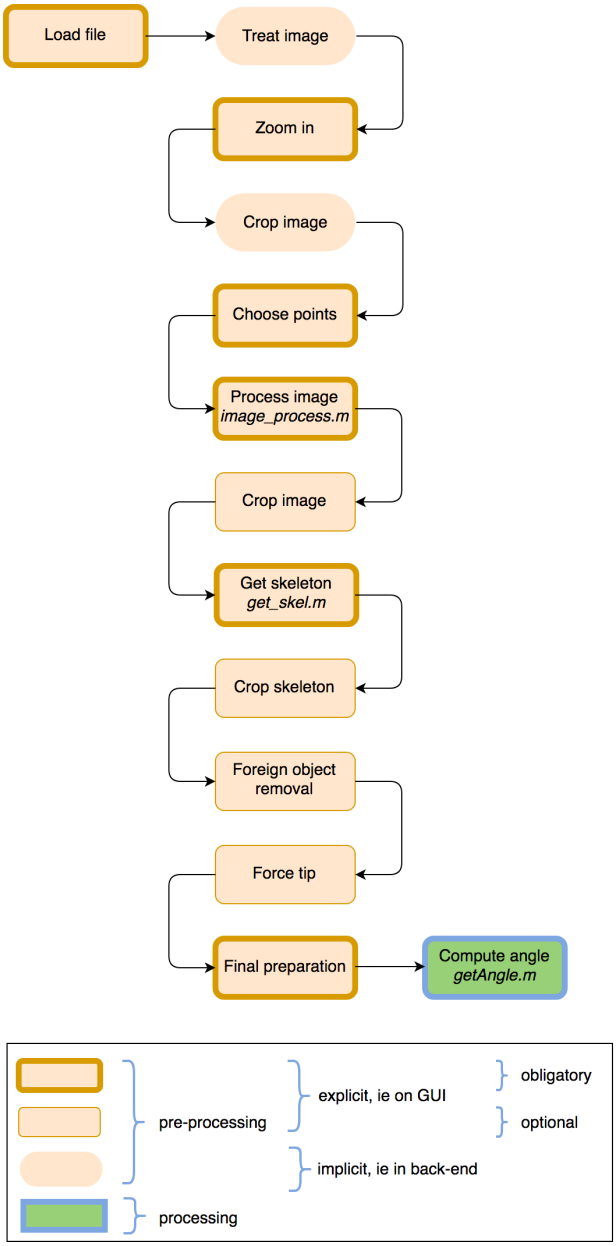

FIGURE 3.1: The workflow of *RootSkel* and different component of the GUI: Pre-processing steps to extract the skeleton are highlighted in orange, the actual angle computation is shaded in blue. Steps that have a front-end, ie are visible on the GUI, are framed in red, only back-end components are without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

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

1

FIGURE 3.2: 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.

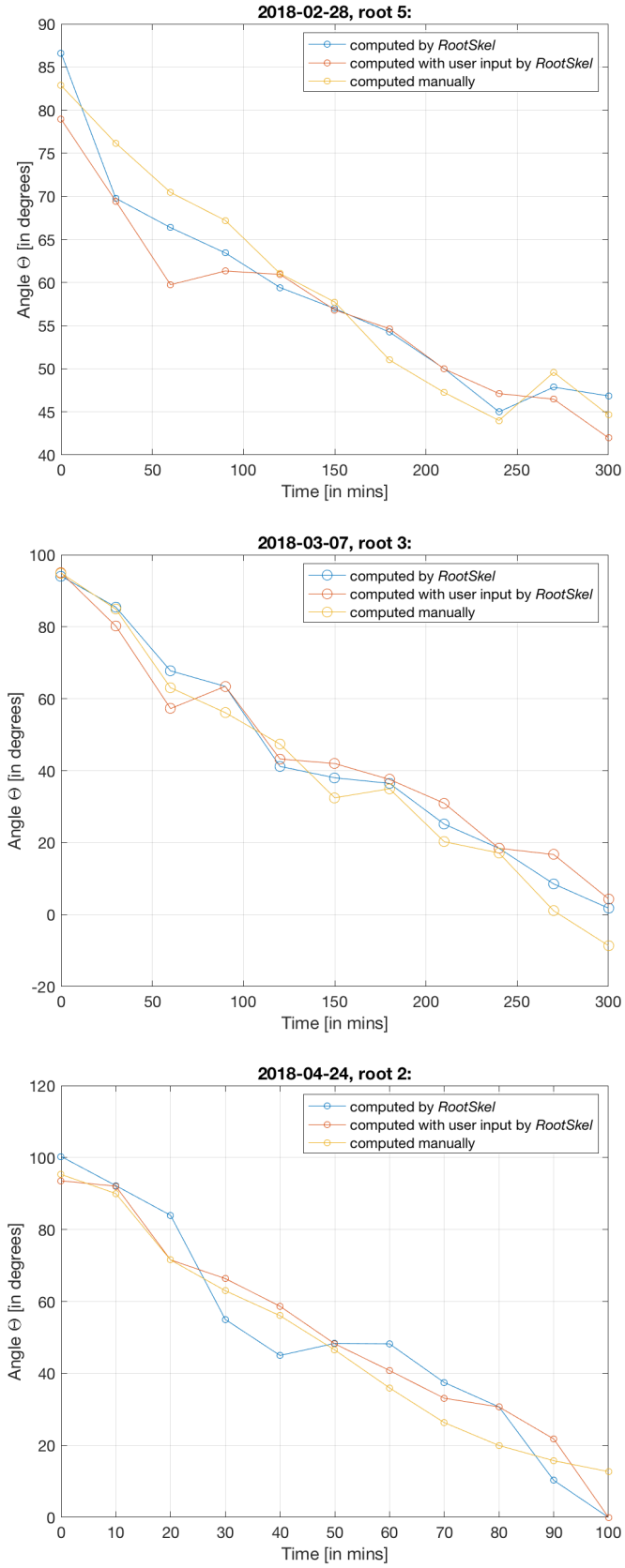

FIGURE 3.3: Comparing the manually computed angle and the angle(s) computed by *RootSkel*; in blue: computed by *RootSkel*, in red: computed by *RootSkel* with the turning point as user input, in yellow: manually computed angle. Each plot is labelled by the data set it was taken from (named after date) 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.

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			**2.8**	**4.7**

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			**4.7**	**83.2**
AVERAGE without considering the measurements highlighted in yellow			**3.8**	**9.9**

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			**8.4**	**29.6**
AVERAGE without considering the measurement highlighted in yellow			**8.0**	**22.6**

FIGURE 3.4: Comparing the autmoatically computed angle by *RootSkel* with the manually computed angles on the 3 roots. Average values are highlighted by a read margin; values that highly skew the results and are advised to be left out when doing the average calculations are highlighted in yellow. It should be noted that we made an educated guess 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.

Chapter 4

Discussion

4.1 Validation on more images

[INSERT ERROR MEASUREMENTS NEED TO BE OBTAINED, CURRENTLY NOT EXISITING IN MANUALLY COMPUTED ANGLES AND ALSO NOT IN ROOTSKEL, IN ORDER TO MAKE VALID ASSUMPTIONS ON ROBUSTNESS AND REPITABILITY]

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* users should be aware of and future developers can wowrk on.

4.2.1 Non-planar root

Another challenge of stuying electrotropism compared to gravitropism is that is is much harder to keep the roots in a plane, ie keep them from growing in a third direction, due to the experimental set up to capture electrotropism. 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 between them. 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 photos

The data set we were working on contained highly challenging photos, 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 itselves 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 reproducability

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 neglectable. Having said that, reproducability can not be fully guaranteed, though.

4.2.5 High user-interaction

Even though *RootSkel* comes with a lot of flexibility and simplicity in the user interaction, it would be desirable to have less user input to guarantee reproducability and save time in the process overall.

This might overall be improved by high-quality images in the future as well as implementing approaches that try to automate the whole preprocessing process.

4.3 Further work

In the follwing 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 Use more shape-based filters

To overcome the problem of separating the root from the background and the noise, one should use and optimise specific shape-based filters instead of a series of colour and intensity filters that have shown to be not very efficient on our data set. This is due to the observation that the tubular structure is the most distinctive feature of the roots. Tubular structure recognising fulters have been implemented but there is room for improvement. This might help to extract the skeleton more easily and reliably without the need of optional cropping or cleaning.

4.3.2 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 different noise our images contain.

4.3.3 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 pattern in the images was efficiently handled which require less user input, it would be desireable to automate the whole angle computation.

4.3.4 Robustness

Many iterations on different images of our data set led to an elaborate tool for the pre-processing of the root; however, we can not prevent that the tool does fail 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.

Adaptive thresholding

Before opting to take user input in the form of samples of the roots in the image, we investigated adaptive (global) thresholding on each of the root and an adaptive variable setting approach on all of the roots together to extract the root. This however failed, or would have been beyond the scope of this project – the reason why we implemented it the way it was suggested.

4.3.5 Maintenance and bug fixing

Even though *RootSkel* is a tool that is ready to be used, and we tried to cover as many cases as possible of things that could go wrong and these bugs need fixing. As we commented the code generously and kept log files about all changes and decisions

made, a future developer should not have any problems building upon our source code.

4.3.6 More features

More features such as a zooming in function at the beginning of force tip can be included; however, they will not change anything in the core functionality of *RootSkel*.

Also, what would be nice to implement in the GUI or in an extra window from a user perspective is a graph superimposing the skeleton to illustrate what angle is computed. This way, the user can doublecheck that the correct angle is computed. This feature will be provided in the next version of *RootSkel*.

4.3.7 Other programming languages

Eventhouth MATLAB has several advantages, it might worth looking into alternative non-proprietary programming language to make the tool accessible to a wider public or other options of making the tool portable, ie without requiring the user to have MATLAB installed. Alternative languages that one 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, could not be investigated further in this work.

4.3.8 Other ways of curvature and angle measuring

What was implemented as we found that this approach worked best on these data and this resolution. However, MATLAB does have some issues with singularities, eg it whenever the angle is (exactly) 45 degrees one should doublecheck of this is due to an infinity issue in the arctan.

Also, for the angle computation we might want to exploit the fact that we have time series data. It means that technically it does not require do compute the angle on each single image but only the difference in the angle to the previous image, since we assume the turning point does not change over time.

Once we have higher-resolution images or a denser data set, we will also achieve a better approximation of the real curvature. In the meantime, it might be worthwile to investigate more maybe better appraoches to compute curvatures on sparse data sets.

Also, alternative definitions instead of an emulation of the so far manually computed angle, such as the curvature itself and possibly more robust methods of computing it might be worth looking into further.

4.3.9 Error quantification in the angle computation

Another additional step towards verification of our results would be to include an error quantification in the angle computation, that is what is the maximum error in the final result due to the changes in the pre-processing step. This could be computedempirically by trying our tool on a large amount of data.

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 optimised for noise-intensive electrotropism images that is able to compute the curvature and angle at the root tip in a standardised fashion with a user-friendly and very flexible pre-processing step to extract the skeleton of the root from possibly very noise image data sets.

The software has been designed using an extensive amount of different filtering techniques optimised on the image data set described in this work [INCLUDE REFERENCE TO SECTION] and can therefore be used to work with standard images fro costumer 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 compute 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.

Robustness and reproducability of *RootSkel* nees to be further validated; suggestions for future work can be implemented.

We hop *RootSkel* will contribute to understand highly complex and poorly-understood phenomenons like eletrotropism in plants and possibly other tropisms, as well as plant growth in general.

Appendix A

On the data set

The data set used can be found on [INSERT REFERENCE GITHUB HERE].

A.1 Suggestions for future data acquisition

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

A.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 usually done in an 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 (eg 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 also will improve the preprocessing step.

A.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.

A.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.

A.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 wasting resolution on things that are of no interest.

It could 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 user 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 B

Manual of *RootSkel*