RootNav User Documentation

The University of Nottingham

2013

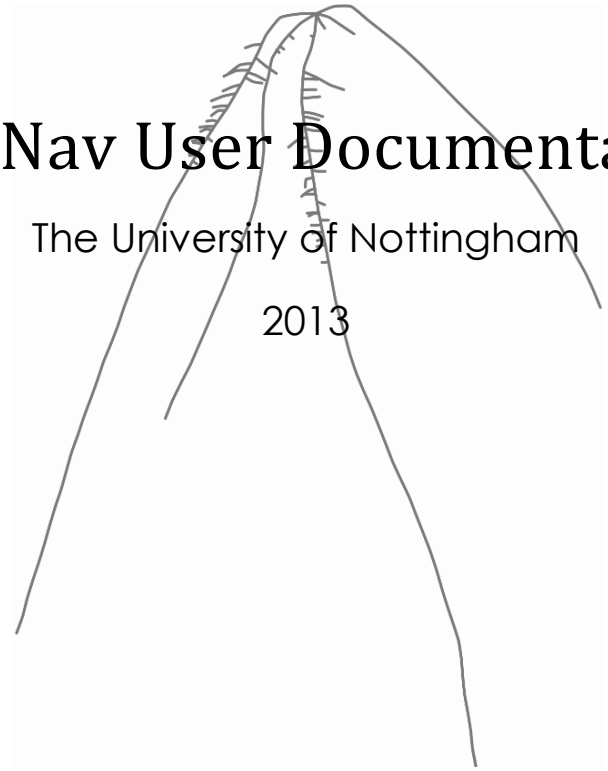

Table of Contents

Overview 2
Introduction 2
Compatible Input Images 2
The Program Interface 3
Using RootNav 5
The RootNav Processing Pipeline 5
Expectation-Maximisation 6
A Discussion of the Existing Presets 8
Locating Root Terminals 9
Manipulating Root Architectures 11
Root Measurement 13
Measurements within RootNav 13
Exporting to a Database 14
The RootNav Viewer 16
Appendix 18
Customising E-M Presets 18
Programming Additional Trait Measurements 19

Overview

Introduction

RootNav is a tool designed to speed up the analysis of root architectures, captured through a variety of techniques. Complete architectures, including both primary and lateral roots can be extracted from a number of plant species, and the software is easily adapted to other input types.

Some RSA traits can be calculcated within RootNav, and an additional Viewer application can be used to extract a larger variety of measurements. Further measurements can be added by those with programming knowledge.

RootNav is written in C# .NET, and makes extensive use of the .NET libraries. For this reason a modern Windows machine is essential. There are currently no plans to re-implement this system for other operating systems.

Compatible Input Images

RootNav can load most common image file formats. For reasons of computational efficiency file size is limited to 7 megapixels, in most cases image dimensions of ~2000 pixels will be more than adequate to accurately capture root systems. RootNav expects images to have a bright foreground, and darker background, any images that have a bright background, for example IR plate images, should be inverted before processing begins. Two example images that will work in RootNav are shown in Figure 1.

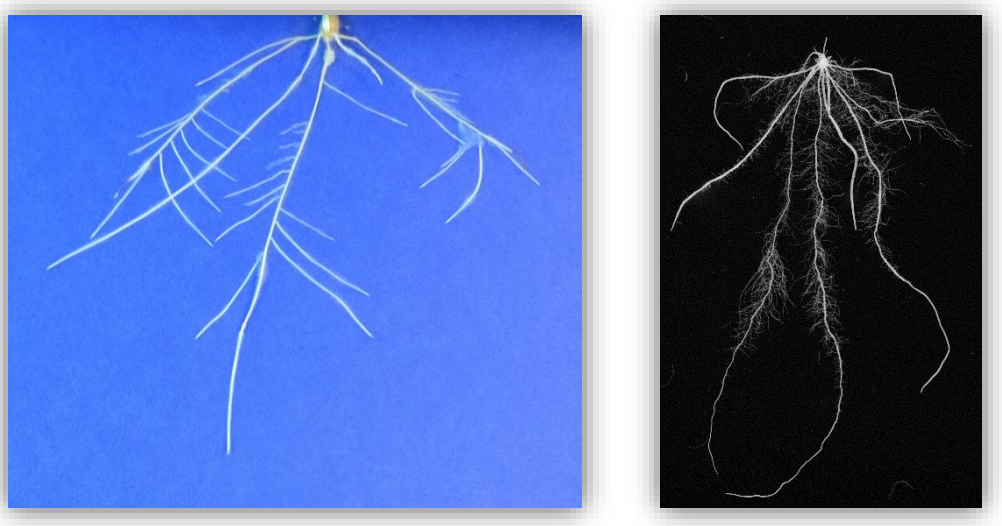

Figure 1: Two example images that can be loaded into RootNav. (Left) A wheat seedling grown on germination paper. (Right) A rice root system scanned in a flatbed scanner. In both cases the foreground (root material) is brighter than the background.

The Program Interface

The RootNav interface consists of 3 major components, as seen in Figure 2.

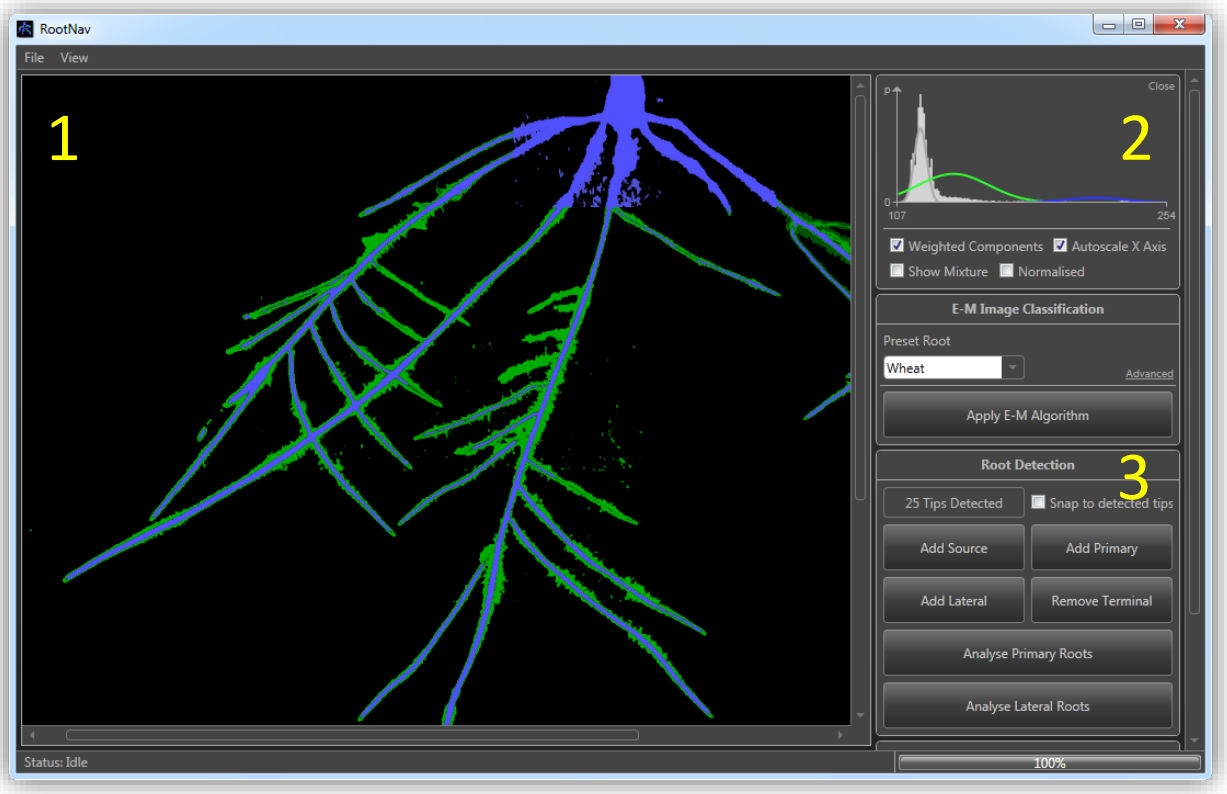

Figure 2: A standard view of the RootNav interface

1. Image Window

The image window shows the current image being analysed by the software. In addition, the majority of manual interaction with the root architecture will involve mouse input in this region. For example, during the tip marking process, it is possible to click a point to add a root tip or source there. During the shortest path process, root paths can be intuitively moved by dragging them.

You can navigate around a loaded image or root system in this window using the mouse. The left mouse button interacts with items, such as adding or removing root tips, or dragging root paths. The right mouse button

2. Expectation-Maximisation Controls

Clicking on an area of the image will bring up the Gaussian Mixture Model for this region. This is only necessary when adjusting which components fall into the background of foreground mixtures, something that is discussed in the original RootNav paper. Below this, it is possible to select pre-configured E-M settings, or open the advanced menu for more control. Again, this will only be necessary in specific circumstances, such as a new type of image capture technique resulting

in a different kind of input image. Details on the controls found in the advanced menu are discussed later in this document, when the use of Expectation-Maximisation is described in detail.

3. Toolbox

The toolbox contains all the interface controls necessary for a given stage in the processing timeline. The controls you see here will vary depending on what task is currently being undertaken. In Figure 2 the controls are concerned with adding or removing various the root markers that will form the end-points of the shortest path algorithm in the following stage.

Using RootNav

The RootNav Processing Pipeline

RootaNav operates using a processing pipeline. Each stage of processing should be completed before the next can begin. In some cases the user interaction at a given stage will be minimal, this will depend on the quality of the input images, and the results that the user requires. In most cases it is possible to return to a previous stage to perform additional tasks, or correct errors. An overview of the RootNav pipeline can be seen in Figure 4.

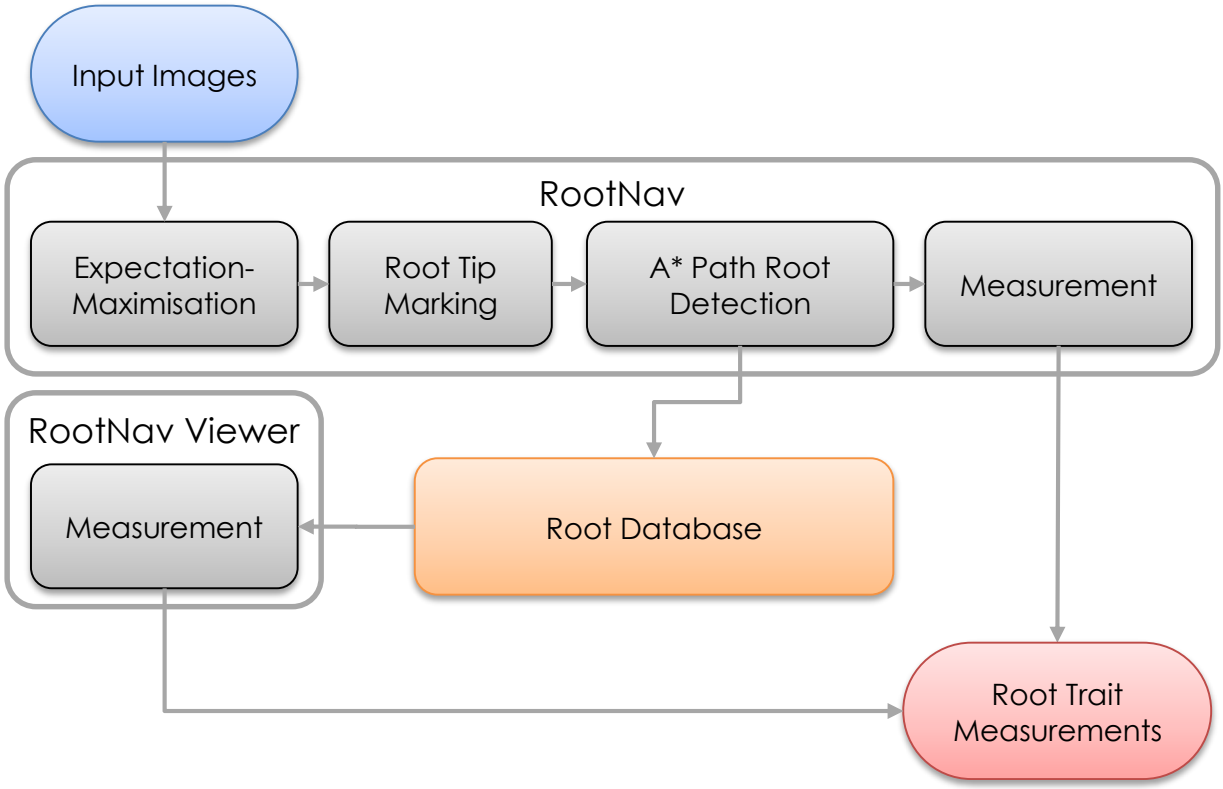

Figure 1: The RootNav processing pipeline

Each component in the pipeline takes the output of the previous stage and passes input into the next stage. In general an image of a plant root system will be loaded into the software, and then analysed for foreground and background pixels using Expectation-Maximisation. Following this, the user may place any number of root sources and tips on the image, and roots will be detected between these points, based on the likely areas of root in the image. Finally, root architectures and traits are measured and either output to a database or table in the measurement stage. Root system architectures saved in the database can be reloaded at a later date, and many additional traits calculated using the RootNav viewer tool.

Expectation-Maximisation

Loading an image into RootNav is done by either clicking on the empty window area when prompted, or by dragging and dropping an image file. When an image is loaded, Expectation-Maximisation begins automatically. The parameters used in E-M are contained in a number of presets, or custom controlled using the advanced tab. These can be seen in Figure 5:

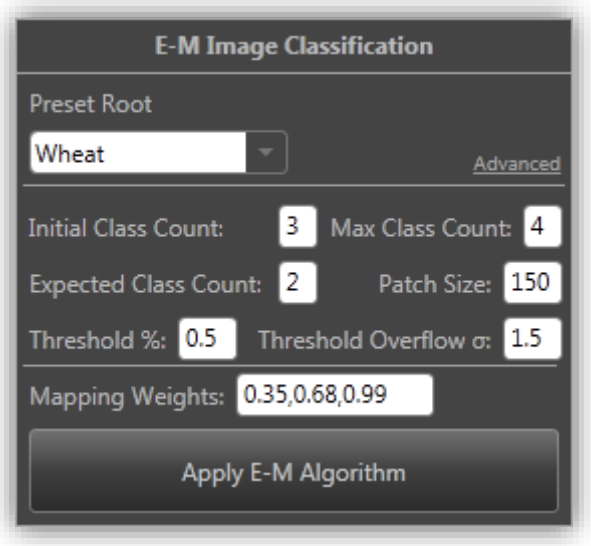

Figure 2: The Expectation-Maximisation classification toolbox.

It is important to note that in the majority of cases, the advanced section need not be used. The drop-down menu labelled "Preset Root" alters these values based on various common image types that may be analysed. These configuration presets are stored in a file named config.xml along with the RootNav software, and can be altered, for example should the user wish a plant that isn't Wheat to be the default selection.

While it is expected that most users will not require any additional information on the exact nature of these parameters, they are provided below for those who are interested, and those who are hoping to run RootNav on very different types of images.

Patch size: The size, in pixels, of the width and height of the image regions used in E-M. Each region is represented by a separate GMM, fitted by a separate instance of the E-M algorithm. However, all instances of E-M use the parameters specified by the other values, such as initial class count.

Initial class count: This specifies the number of components *k* in the Gaussian mixture model, fitted by E-M. This is the total number of components, including foreground and background.

Max class count: This value represents the maximum number of components that E-M classification will try, should the initial count not provide adequate results. This determination is made using the expected class count value.

Expected class count: This is the number of *foreground* classes that are expected in each image region. With an initial class count of 3, this means the algorithm expects 1 background class, and 2 foreground classes. Further details on what these classes may represent can be found in the original paper.

Threshold %: This is the value, between 0 and 1, by which RootNav determines which components in the fitted Gaussian mixture model are background, and which are foreground. This process is discussed in detail in the paper, in most cases this value need not be changed.

Threshold Overflow: This value ensures that should multiple background components be fitted to the same position, that all of these are included in the background distribution. When components are added into the background, once the threshold % has been reached, any additional components within this standard deviation of the mean will be automatically added. This is rare, but helps to compensate when the initial class count is set too high, and multiple background components are produced. In the majority of cases, this value need not be changed. A value of zero removes this functionality entirely.

Mapping Weights: Mapping weights specify the relative weights given to each class when the shortest path graph is created. Consider a mapping weight value of 0.35,0.68,0.99. This means that the likelihoods produced by the highest intensity component will be further weighted by 0.99, the second highest by 0.68, then all subsequent components by 0.35. These values are discussed below along with the presets for wheat, rice and Arabidopsis.

How these values work together: Understanding these values is made easier when they are seen in relation to the entire RootNav E-M step. Classification of roots proceeds as follows for each image region:

1) Expectation-Maximisation runs using a number of components specified by the initial class count.
2) Threshold % and Overflow are used to split the resulting GMM into background and foreground, with the amount of background corresponding to at least the threshold %.
3) The number of foreground components is compared to the expected class count, if these values are the same, classification is complete.
4) If these values are not the same, E-M may repeat steps 1-3 using an additional component, and continue repeating while the number of components in use is less than or equal to maximum class count. If at any point the expected class count is matched, the process ends. If the maximum

class count is reached, the process also ends, and the current foreground and background mixtures are used.

A Discussion of the Existing Presets

This section explains the existing presets for Wheat, Rice and Arabidopsis Plate images.

Wheat: Many of the wheat images used to test RootNav were photographs of plants grown on germination paper. These were captured in good light, using a high resolution DSLR camera. The images are good enough that the root hair can be seen as a lighter border around the main root tissue. There is enough of a difference in intensity between root, root hair and background to classify this data into three or four separate components. For this reason, an initial class count of 3, and maximum class count of 4 were used. The expected class count is 2 foreground classes (root and root hair).

In the majority of wheat images we did not expect a large amount of root tissue in each region, so a background threshold value of 50% was used, an overflow of 1.5 standard deviations above this accounts for situations where many background components fall on the same position. This is rare, but possible. Given the size of the images being tested, a few megapixels, a region size of 150^2 is reasonable.

Finally, the mapping weights mean that root is weighted by 0.99 (almost not weighted down at all). Root hair is weighted at 0.68, meaning that the shortest path algorithm will travel down root hair, but not if a valid root pixel lies nearby. Finally, the weight of any additional root hair or background is 0.35. These values were chosen mainly on empirical observation, but to an extent they were also chosen to intuitively weight each class as we would expect.

Rice: We have also tested RootNav on flatbed scanner images of rice roots. In these images lateral roots take the place of a single component, rather than root hair, which cannot be seen. We expect a single root class, and a single background class. This provides an initial class count of 3, but we also allow a maximum class count of 4 for noisy regions. The expected class count is 2 foreground classes (root and lateral).

The patch size is larger than that of wheat, because the background is generally less noisy, and some smaller patches would contain a lot of root material. This change allows us to run the same threshold % and overflow values as with wheat.

The mapping weights do not penalise additional root classes like with wheat, because all lateral material should be treated the same. For this reason root material is weighted at 0.99, and all other classes at 0.68.

Arabidopsis Plate: IR images of Arabidopsis on agar plate will often contain multiple seedlings, often at an earlier stage of growth. Root hair is less developed, and is

difficult to detect at these resolutions. For this reason the number of classes fitted using E-M is reduced by 1 throughout. Thus, an initial class count of 2, a maximum class count of 3, and an expected class count of 1 foreground class (all root material). Other variables remain the same as with wheat, except the mapping weights, where like rice we need only two.

Locating Root Terminals

In order to calculate possible root paths, the user must first add root sources and tips onto the classified E-M image. This is done using the root detection toolbox, which can be seen in Figure 4.

Figure 3: The Root Detection toolbox.

The four upper toggle buttons are used to add and remove the root tips, these are used by the shortest path algorithm to find possible root architectures. A **source** represents the beginning of a primary or seminal root, or roots. A source can have any number of roots attached to it. Toggling the "Add source" button and clicking anywhere in the image window will add a source, which will be shown with an S symbol. A **primary** represents any primary or seminal root, in this context any root that is not a lateral. Paths are found between sources and primaries. Primaries can be added in the same way as sources, and are shown with a P symbol. A **lateral** represents any lateral root. In this context this is any root that starts part-way down a root path, rather than from a source. Laterals are represent using an L symbol. Sources, primaries and laterals are known collectively in RootNav as terminals.

In images containing multiple plants, there will be multiple source terminals associated with multiple primary terminals. By default RootNav will calculate an optimum path between each combination of source and primary, then attempt to reconstruct plants using the most likely combination of these terminals. This exhaustive process can be computationally inefficient, with 5 sources and 5

primaries, RootNav will examine all 25 possible path combinations. Where there is a clear mapping between primaries and sources, a user might wish to calculate paths only between these known pairs. Terminals can be linked by first placing a source terminal, then holding the *Shift* key (the "Add Primary" button will be automatically toggled), then clicking to add a primary terminal. A line will appear between the source and primary showing that these two are linked. Pairing up combinations of 5 source and 5 primary terminals will result in only 5 paths being reconstructed, with a 5-fold decrease in the required computation time.

As soon as the E-M step is completed, RootNav calculates positions of possible root tips automatically. These locations are not shown; they are used to guide the placement of root terminals when added manually. If you wish to use this feature, tick the "Snap to detected tips" box that can be found in the Root Detection tool box. If you enable this feature, semi-transparent circles will appear at possible root tip locations. Adding or dragging any source, primary or lateral near these will immediately snap them to that location. If you are dissatisfied with the location you can override this feature by holding the *Ctrl* key when you add or move a terminal.

Once any number of sources, primaries and optional laterals have been added, clicking the "Analyse Primary Roots" button will begin the A* search path reconstruction of the root architecture. Once primary roots have been completed, clicking "Analyse Lateral Roots" will complete the architecture. These two steps have been separated to allow users to manipulate the primary roots. This is discussed in the next section.

Manipulating Root Architectures

One crucial aspect of RootNav is the ability to manipulate primary and lateral root paths once they have been automatically detected. By combining this manipulation with a shortest path approach, the construction of accurate root architectures is quick, and the resulting roots adhere strongly to the image data. In many cases there is no need to manipulate the paths found by the shortest-path approach. In some cases it might be necessary, such as:

1) If two roots grow close to one another, it is possible that two paths may travel along the same root.
2) If part of a root path is missing, for example where there is poor image quality, a root might take an alternate path.
3) If a lateral root travels to an incorrect primary, for example where this unrelated primary intersects the lateral.

In all cases where a path has not adhered exactly to the intended root, dragging the path will re-compute the shortest path and improve the root architecture. The figure below shows how this process works:

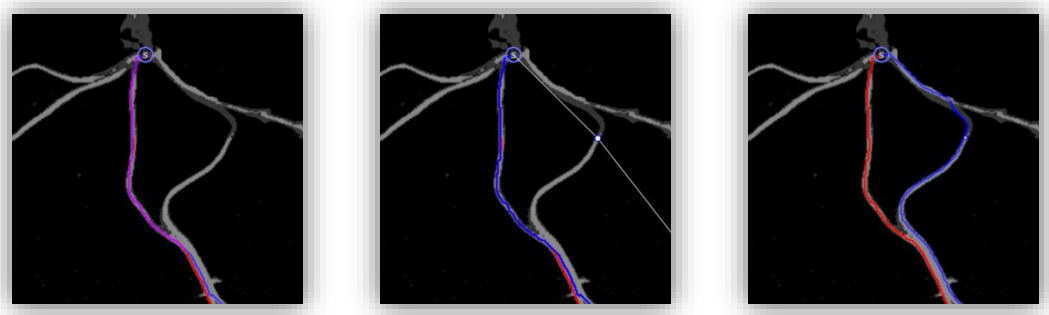

Figure 4: Manipulating root architectures

In figure 5, both the red and blue root paths have travelled down a single path of the image. This will sometimes occur where roots grow next to one another, simply because one root becomes the shortest path. This is easily fixed by dragging a root onto the other path. This creates a control point at this location, which forces the path through this point. The algorithm is still free to travel along the shortest path between terminals and control points, so will still follow the image data closely.

Usually a very small number of control points are sufficient to correct an incorrect path. It should be noted however that there is no limit to the number of control points that can be added. Theoretically any challenging image with any number of roots can be completed with enough manual interaction. It remains for the user to decide if the time required to achieve this on some images is worthwhile.

Manual interaction can also be used to direct lateral roots to other primary roots. This is necessary where an unrelated primary crosses a lateral root, causing the lateral to join with this primary at this point. Dragging the lateral path beyond this junction

towards the true parent root will redirect the path there. This approach can be seen in the figure below.

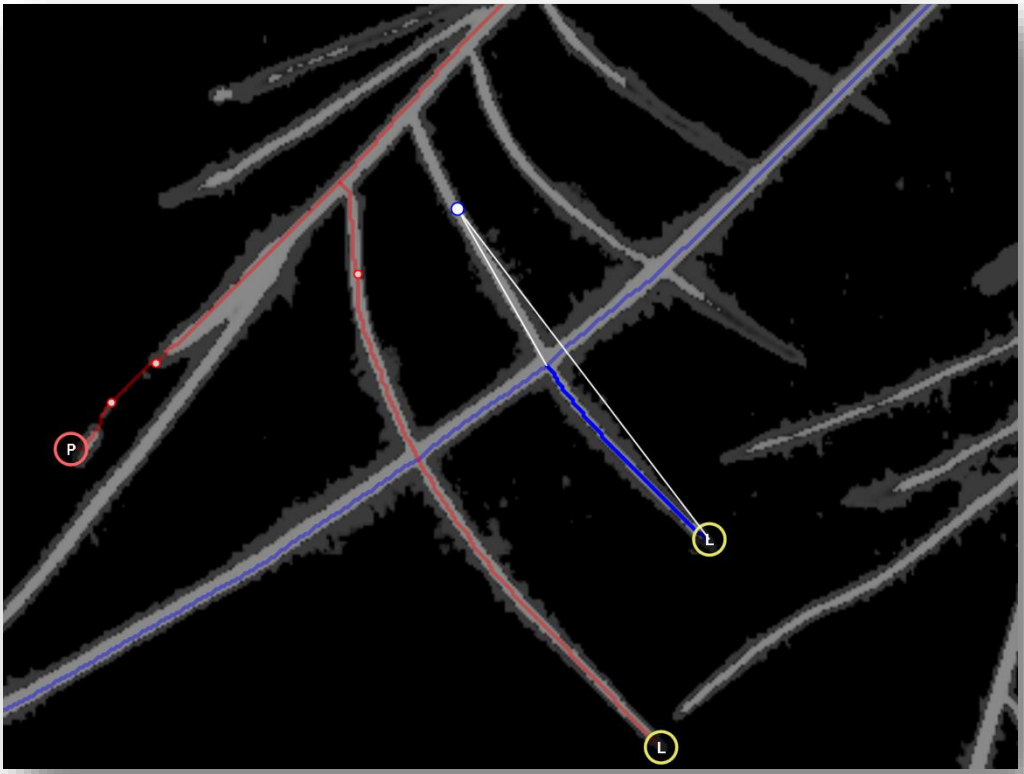

Figure 5: Lateral roots can be redirected past a separate primary towards the true parent root

Root Measurement

Measurements within RootNav

Once the architecture of any required roots has been constructed, measurements can be taken. Simple measures such as root lengths, emergence angle and tip angles are calculated within RootNav. More complex traits can be calculated using the separate RootNavViewer software that is supplied with the installation of RootNav. This software will be discussed in detail later in this document.

The measurement stage begins when the user clicks the "Measurement" button below the root detection toolbox. The initial measurement controls can be seen in Figure X, along with the subsequent "Root Measurement" toolbox. The spacing of the spline control points can be adjusted; reducing this value can help where roots are particularly curved, such that a spline might oversimplify the complex shape.

The image scale can be specified in pixels per mm, allowing RootNav to adjust results into mm measurements, rather than outputting in pixels. This conversion can also be easily computed afterwards with the output data.

Clicking the Measurement button will show the root tree view and the root measurement toolbox. The main window will also change to remove any path manipulation controls.

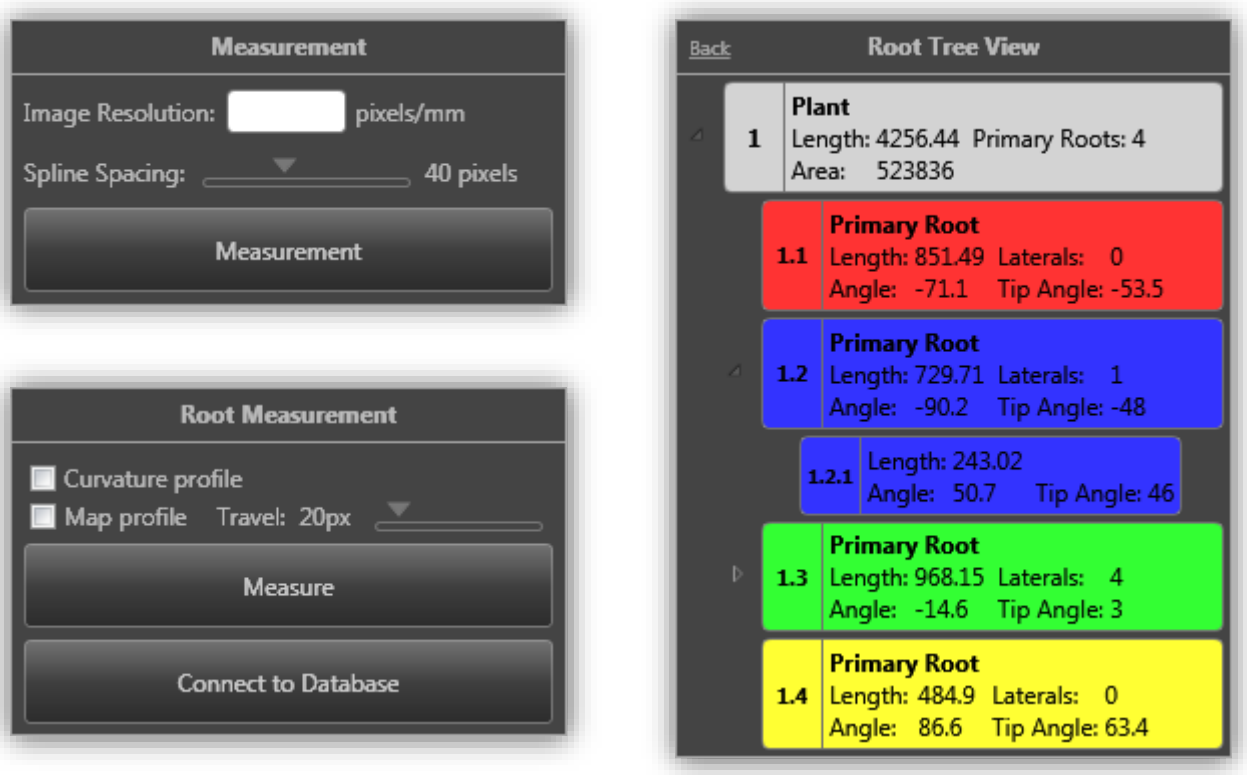

Figure 6: The measurement controls. (Top left) Controls to begin measurement. (Bottom left) The controls to take measurements and output data. (Right) A tree view of all detected plants, roots and laterals in this image

Clicking any branch of the root tree view will highlight the corresponding root in the viewer window. Root architectures are grouped into plants, with each source terminal in the original reconstruction representing a single plant. Clicking a plant in the tree view will also display the convex hull for that plant.

Clicking the Measure button will output measurements such as root length and emergence angles. If the user has checked either the map profile or curvature profile boxes, these additional measurements will also be computed.

Connecting to the database will allow any measurements to be added to it, and analysed at a later date using the RootNavViewer tool. Users who are intending to analyse more than a few images are strongly encouraged to use the database functionality. The measurements computed by the viewer are more extensive, and can also be computed in a batch mode (multiple plants and images at once). New trait measurements can also be added via a plugin interface. Instructions on connecting to the database can be found in the next section.

Exporting to a Database

Storing root architectures in a database for further analysis provides a number of notable advantages when using RootNav over many images and dates. The architectures are stored together in a consistent manner, and can be analysed simultaneously once all quantification is complete. Because the database stores the original spline data, the measurements of new traits can also be retroactively taken on all plants, regardless of whether that trait was originally intended to be measured.

At the time of publication, RootNav will only interface with a MySQL database. It is possible that other database servers will be added into the supported list at a later date. In particular SQLite will be added to allow portable RootnaV installations to be possible with no server software. For the remainder of this section it is assumed that the user has access to a MySQL database on the same or another machine.

To connect to the database, click the "Connect to Database" button, found in the Root Measurement toolbox. This should be done *before* the measurement button is clicked, otherwise RootNav will assume output is to be in tabular form. If this is the first time a database is being used, or if the server is not available, a connection window will appear where server details should be entered. This window is shown in Figure 8:

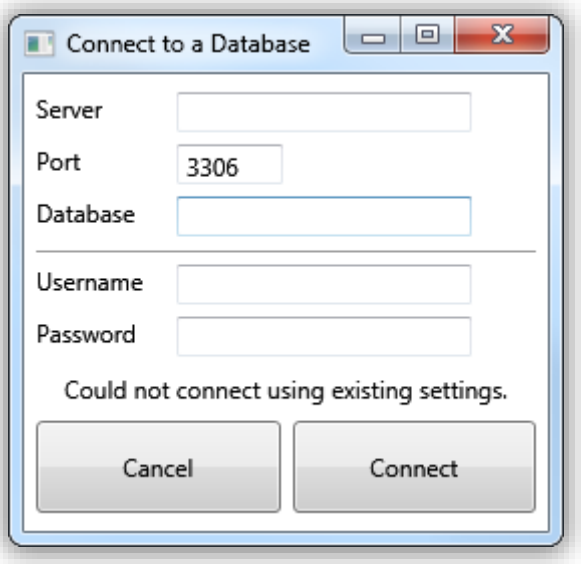

Figure 8: The Database Connection Window. This window will appear when RootNav or the RootNavViewer cannot connect to the server, or the first time either is run. After that, the information will be saved and should not need to be entered again.

Most MySQL installations operate on port 3306, and installations running on the same machine as RootNav can be accessed using Server: *localhost*. Assuming the connection is successful, the Connect to Database button will turn green, as shown in Figure 9.

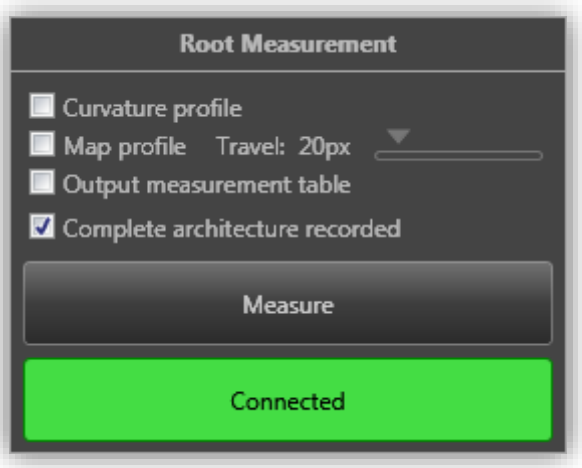

Figure 9: After a database connection is established, the connect button will turn green.

Additional check boxes will also appear once a database connection is established. "Output measurement table" will also output measurements in a tabular form, if the user requires both the measurements now, and for them to be stored in the database. The "Complete architecture recorded" stores a single true or false value

alongside the measurements in the database. This can be useful if some images are difficult to quantify, the user can signal that some roots were omitted, allowing this to be considered when traits are measured at a later date. In cases where only some subset of the root system is being quantified, perhaps because lateral roots are not necessary for the experiment, this value informs any future users that some traits will not be relevant on this data.

The RootNav Viewer

The RootNav Viewer is a standalone application that is provided with RootNav. The motivation for this application is to separate the majority of the trait measurement calculations from the capture the root architectures, using the database as an intermediate storage mechanism. The viewer can be easily extended with additional traits, which can be applied to any number of architectures in the database, regardless of whether the traits were considered when the root system was first captured. A simple plugin interface allows anyone with programming experience to easily measure custom traits on their data.

The RootNav viewer operates only on architectures stored in a database, so a connection window like that shown in Figure 8 must be completed before the software can be used. Once loaded, all plants currently in the database can be viewed, as shown in Figure 10.

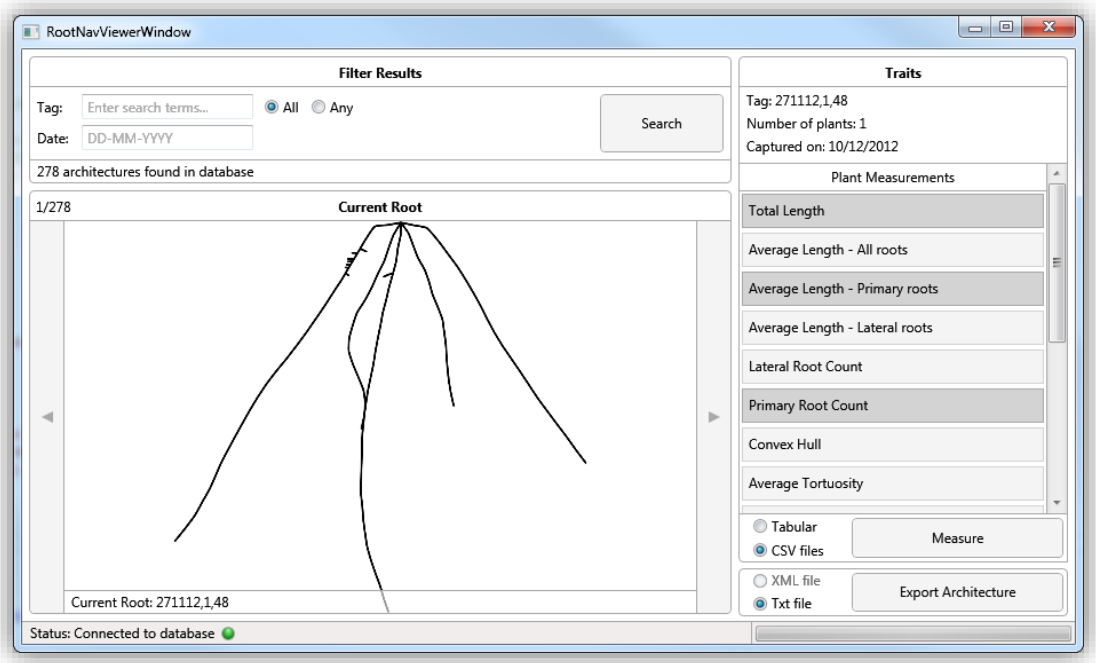

Figure 10: The Rootnav Viewer User Interface. Filters can be used to search for specific plants, the architectures can be viewed, and the required traits chosen and calculated.

The measure button will calculate the values of all highlighted traits for all images in the current selection, then will output this data to a table, or CSV file as chosen by the user. Some traits that output 2-dimensional data will be automatically written to CSV file.

Selecting the "Export Architecture" button will output the original spline data in a CSV or Text format for import into other software, such as Matlab.

Appendix

Customising E-M Presets

Depending on the nature of the input images you are using, you may wish to adjust these values. If your images are similar but not identical to the presets, you may find these will work fine without any alteration. If you do wish to change them, altering the file Configurations.xml, found in the same directory as RootNav.exe will allow you to add any number of custom presets. It is also possibly to alter the default preset in this file, if for example you wish to often start with the Arabidopsis preset.

The structure of the Configurations.xml file is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfEMConfiguration
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- EM Configurations -->
  <EMConfiguration>
    <Name>Wheat</Name>
      <InitialClassCount>3</InitialClassCount>
      <MaximumClassCount>4</MaximumClassCount>
      <ExpectedRootClassCount>2</ExpectedRootClassCount>
      <PatchSize>150</PatchSize>
      <BackgroundPercentage>0.50</BackgroundPercentage>
      <BackgroundExcessSigma>1.5</BackgroundExcessSigma>
      <Weights>
        <double>0.35</double>
        <double>0.68</double>
        <double>0.99</double>
      </Weights>
      <Default/> <!-- Optional -->
  </EMConfiguration>
  ... More EMConfigurations
</ArrayOfEMConfiguration>
```

The file is loaded automatically using C#. For this reason the general structure of the file (particularly the header) should not be altered, but the number of *EMConfiguration* tags can be altered. If you wish a tag to be the default, add a *<Default/>* tag into an EMConfiguration tag. If multiple defaults are specified, the first will be used.

Programming Additional Trait Measurements

The RootNav Viewer can be extended using a simple plugin system, allowing additional traits to be measured based on RSAs in the database. Any traits that are added can be calculated on any RSA stored, not just those that are measured after the trait was programmed. This allows all plants to be re-analysed if an additional trait is required at some later date.

RootNav is programmed in .NET, and all plugins should also be programmed in .NET. This means that any .NET language, such as C# or Visual Basic can be used to program additional plugins. This small tutorial on plugins will use C#, however programmers will find it easy to adapt the code to their language of choice.

All viewer plugins are based upon the abstract class MeasurementHandler. This class is defined as:

```
public abstract class MeasurementHandler
{
    public abstract string Name { get; }

    public abstract MeasurementType Measures { get; }

    public abstract bool ReturnsSingleItem { get; }

    public virtual object MeasurePlant(PlantInfo plant)
    {
        throw new NotImplementedException();
    }

    public virtual object MeasureRoot(RootInfo root)
    {
        throw new NotImplementedException();
    }
}

public enum MeasurementType
{
    Plant, Root
}
```

Any plugin must inherit from this class and implement its member functions. To do this, add RootNavMeasurement.dll as a reference in your project, then add a new class inheriting from RootNavMeasurement.MeasurementHandler. The three abstact properties *must* be implemented. One of the two virtual methods MeasurePlant and MeasureRoot should be implemented depending on which type is returned by the Measures property.

The following table shows the function of each of the members of the MeasurementHandler class:

Property or Method	Description	Return Type
Name	Returns the name of the trait for display on the viewer interface	object, usually a string, integer or double
Measures	Returns either MeasurementType.Plant or MeasurementType.Root depending on which structure this trait measures. If a trait could theoretically measure both, e.g. total length, this should be implemented as two separate plugins	MeasurementType
ReturnsSingleItem	Returns true or false depending on whether this trait returns a single value for a single root or plant. For example, total length returns a single value, whereas a curvature profile returns multiple values for a given plant.	A boolean value – true or false
MeasurePlant	The function that implements the measurement of plant information. This is only required if the measures property returns MeasurementType.Plant	object, usually a string, integer or double Or a DataTable if the plugin outputs multiple values
MeasureRoot	The function that implements the measurement of root information. This is only required if the measures property returns MeasurementType.Root	object, usually a string, integer or double Or a DataTable if the plugin outputs multiple values

An example measurement handler that calculates total root length for individual roots is given below:

```
public class TotalLengthRootHandler : MeasurementHandler
{
    public override string Name
    {
        get { return "Total Length"; }
    }

    public override MeasurementType Measures
    {
        get { return MeasurementType.Root; }
    }

    public override bool ReturnsSingleItem
    {
        get { return true; }
    }

    public override object MeasureRoot(RootInfo root)
    {
        return Math.Round(root.Spline == null ?
            0.0 : root.Spline.Length, 2);
        }
    }
```

Once at least one class has implemented the MeasurementHandler, this project should be compiled as a dynamic link library (.dll) file, and placed into the same directory as the RootNav installation. When the viewer loads, any plugins will be automatically found and incorporated into the list of possible traits.

The source code for all current trait measurements is available within the RootNav solution, and interested readers are encouraged to review it if they wish to learn more about implementing plugins for the RootNav viewer.