Root_Processing Documentation

Release 1.1.9

Keita DeCarlo

May 10, 2019

Contents

CHAPTER 1

Introduction

Here provide a step-by-step procedure for downloading and implementing the Python package ‘rootprocessing’. This library provides a suite of image preparation and processing tools used for images outputted by the CG-1D beamline at ORNL. Although primarily designed for data at the beamline, in particular the pre-processing procedures, all individual analyses can be run on any external image, and are free to use.

CHAPTER 2

Getting Started

I. SETTING UP

First, download the 'Root_Processing' library, available on the PyPi website, onto your computer:

```
pip install rootprocessing
```

We will then create a fake dataset in order to run and test the analyses. In your window, type the following:

```
import rootprocessing
wd = '/Users/johnsmith'  #Specify where your data files will be.
from rootprocessing.sampledata import sampledata
sampledata(wd)
```

This will create a 'Sample_Data' subdirectory in your library, which will contain a 'raw' subdirectory with a set of 6 images, as well as a dark field and open beam image.

You will also notice a 'user_config' file created in the 'Sample_Data' file - this contains all the necessary parameters for each of the analyses conducted by this library. Please check the documentation within each module for details.

Also, be sure to *only change the entries following the colon for each parameter!* Do not add any extra lines or modify the headings for each section.

II. RUNNING ANALYSES WITH 'RP_RUN'

From here, we will use the 'RP_run' module, which will act as the top-level program for running any analyses of interest:

```
from rootprocessing.RP_run import RP_run
```

We will then specify the analyses of interest. You can run these in any order, but make sure that you have the necessary images and analyses completed first. Below is the suggested order of the analyses:

```
analysis_list = ['RP_stitch', 'RP_crop', 'RP_wc', 'RP_mask', 'RP_imagefilter', 'RP_
↪distmap', 'RP_radwc', 'RP_thickness', 'RP_rootdiameter']
```

We will also need to specify where the user_config.txt file will be. In our case, this is the same location as where our data files are:

```
wd_userconfig = wd+'/Sample_Data'
```

Once this is complete, then simply run the module, and the outputted subdirectories/data will automatically be placed in the 'Sample_Data' subdirectory:

```
RP_run(wd, wd_userconfig, analysis_list)
```

As noted earlier, this code can run each component separately (assuming all necessary input files are already in place). So for example, if a user is only interested in the root diameter distribution of their image, they can run either:

```
analysis_list = ['RP_stitch', 'RP_crop', 'RP_mask', 'RP_imagefilter',
'RP_rootdiameter']
RP_run(wd, wd_userconfig, analysis_list)
```

Or:

```
RP_run(wd, wd_userconfig, ['RP_crop'])
RP_run(wd, wd_userconfig, ['RP_mask'])
RP_run(wd, wd_userconfig, ['RP_imagefilter'])
RP_run(wd, wd_userconfig, ['RP_rootdiameter'])
```

Both of these formats will output the same files.

Specific tutorials for each analysis will be outlined, using the sample dataset provided, so be sure to run that code when following through the guides.

CHAPTER 3

Getting Started: stitch

I. OVERVIEW

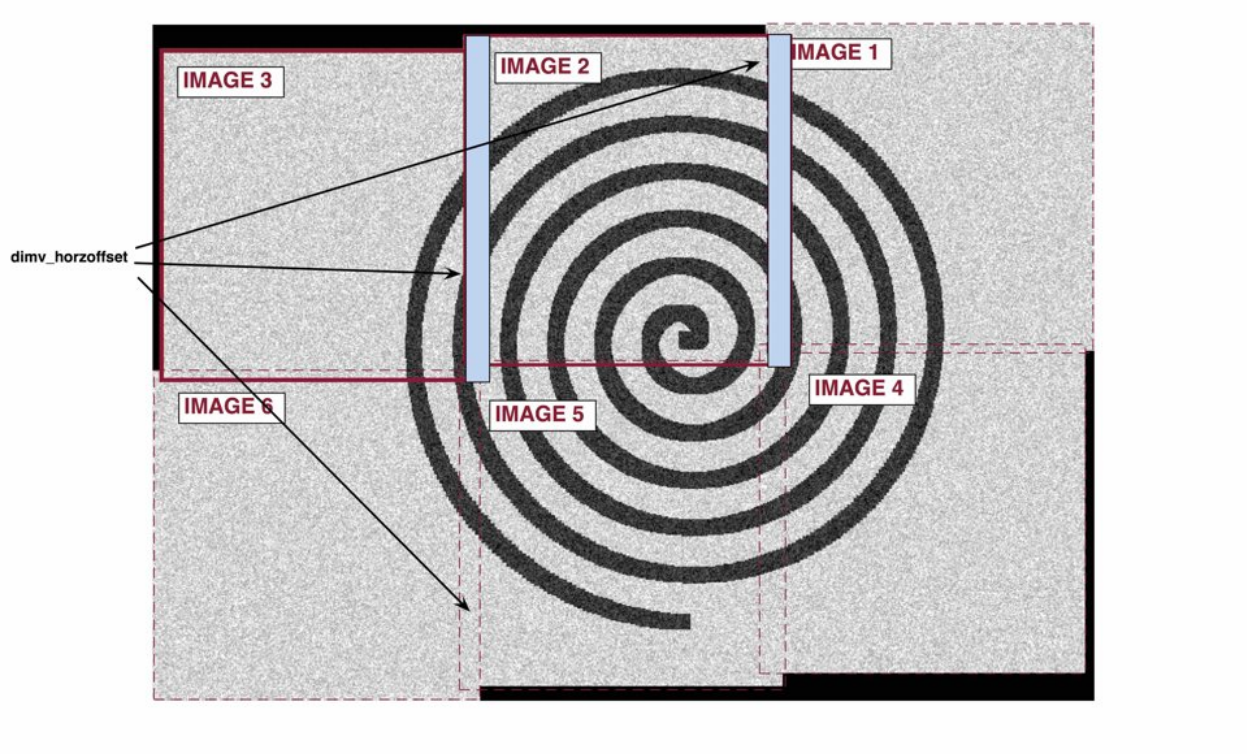
The 'RP_stitch' analysis takes the raw images outputted from the CG-1D beamline and stitches the images together, as well as applying the necessary image corrections.

II. HOW TO USE

First, open the 'user_config' text file in your 'Root_Processing' directory. The first line will be '1. STITCH', with the following 9 lines specifying the parameters used in the analysis. In order, they are:

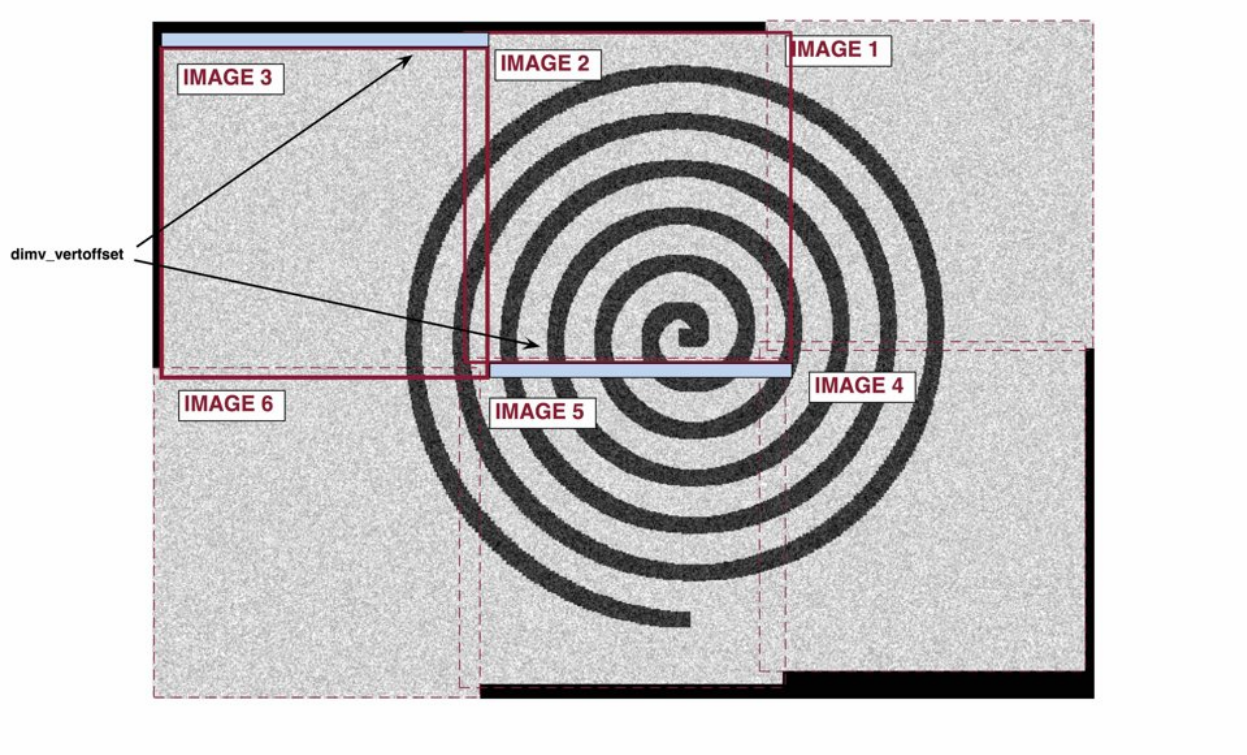
1. image_filename: this is the directory name where the raw files are to be found. Our raw files are in the 'Sample_Data/raw' directory. These are the images directly outputted by the CG1D beamline, and will have integer values (i.e. not corrected to transmission values).
2. output_filename: this is the directory name where the images will be saved. The program will automatically make a 'stitched' directory if not already present, so there is no need to create one manually.
3. output_fileformat: this is the filename format you would like your images to be saved as.
3. fileformat: this is the fileformat of the image to be outputted. The CG1D beamline outputs images in a 'YYYYMMDD_filename_secondsexposed' format, so this is what you will write here.
4. dimv_horzoffset: this is the *horizontal* offset value (i.e. overlap) between two images in the *vertical* dimension.

Essentially, this is the overlap between two horizontally adjacent images:



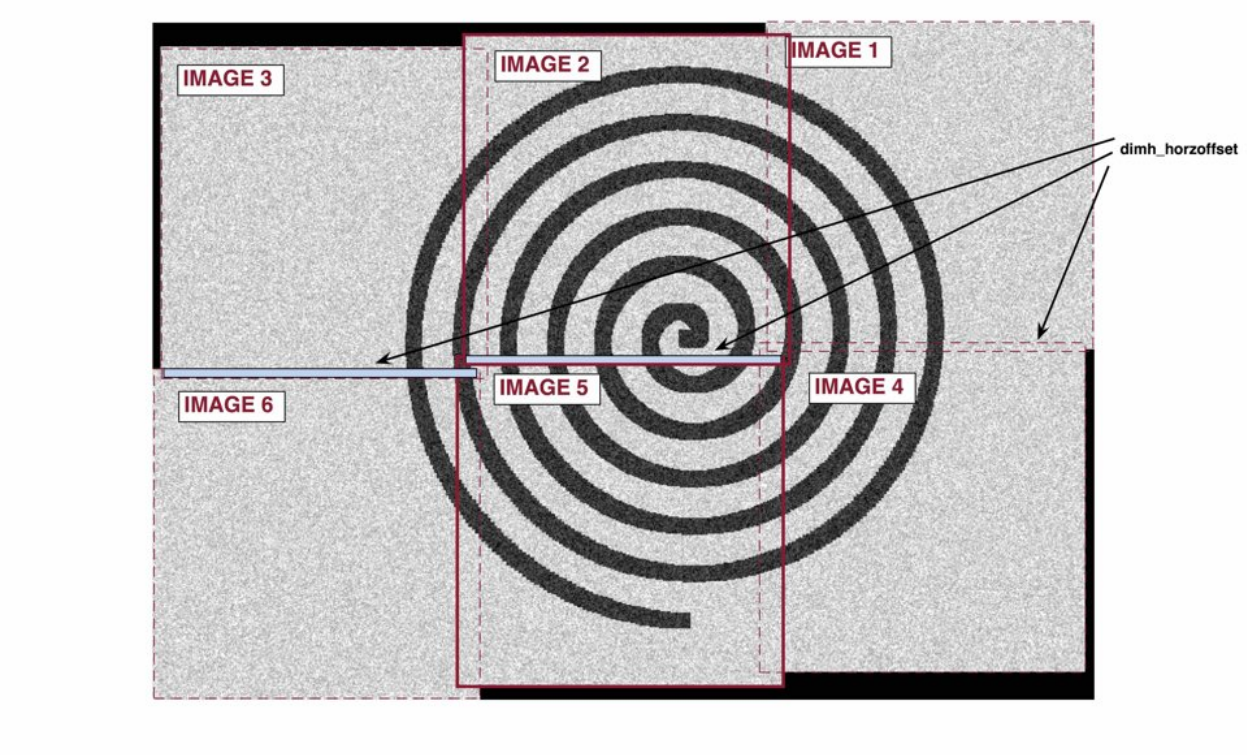
5. dimv_vertoffset: this is the *vertical* offset value between two images in the *vertical* dimension.

Essentially, this is how much an image "drifts" downward with each table shift:



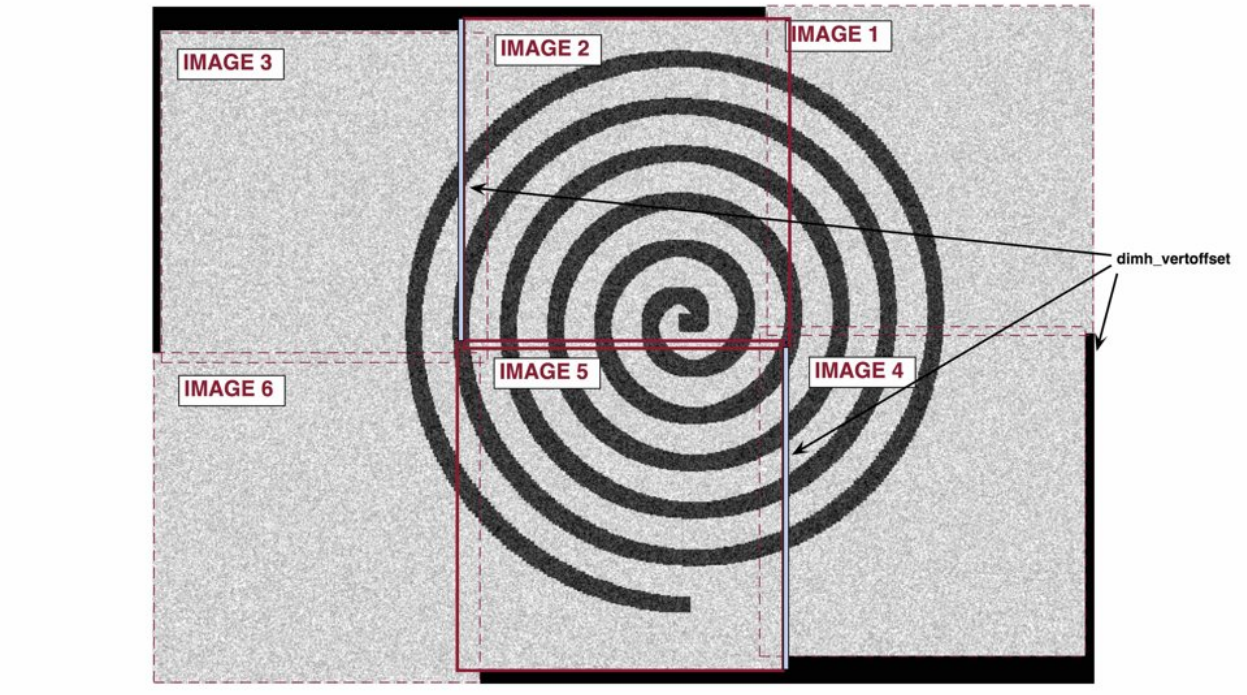
6. dimh_horzoffset: this is the *horizontal* offset value between two images in the *horizontal* dimension.

Essentially, this is the overlap between two vertically adjacent images:



7. dimh_vertoffset: this is the *vertical* offset value between two images in the *horizontal* dimension.

Essentially, this is how much an image "drifts" leftward between each table shift:



8. stitch_order: this is the array that specifies the order in which you would like to stitch your images. The first two values indicate the number of (1) rows and (2) columns of raw images that your final stitched image will have. The following numbers will specify the number label for each position, starting from row 1, column 1 (top left corner), and moving column by column, then moving on to the next row.

So, using our sample data, we have 6 images to stitch, with 3 images per row, and 2 images per column. So we will specify a 1x8 array, with the first two numbers being 2 and 3. From here, we will specify the image position. Our images are called '19000101_Image_0060_0001.tiff', '19000101_Image_0060_0002.tiff', etc. Our 'fileformat' parameter specifies the image name prefix ('19000101_Image_0060'), so we will only need to specify the numbers here. We want image 3 in the top left corner, with image 2 and then image 1 in the top middle and top right sections, respectively. For the bottom row, we want images 6, 5, and 4 in the left, middle, and right sections, respectively. Placing these numbers after the first two values in the 1x8 array will give us the final 'stitch_order' parameter.

III. RUNNING THE CODE

This analysis can be conducted using the ['RP_stitch'] string in the 'RP_run' module.

CHAPTER 4

Getting Started: crop

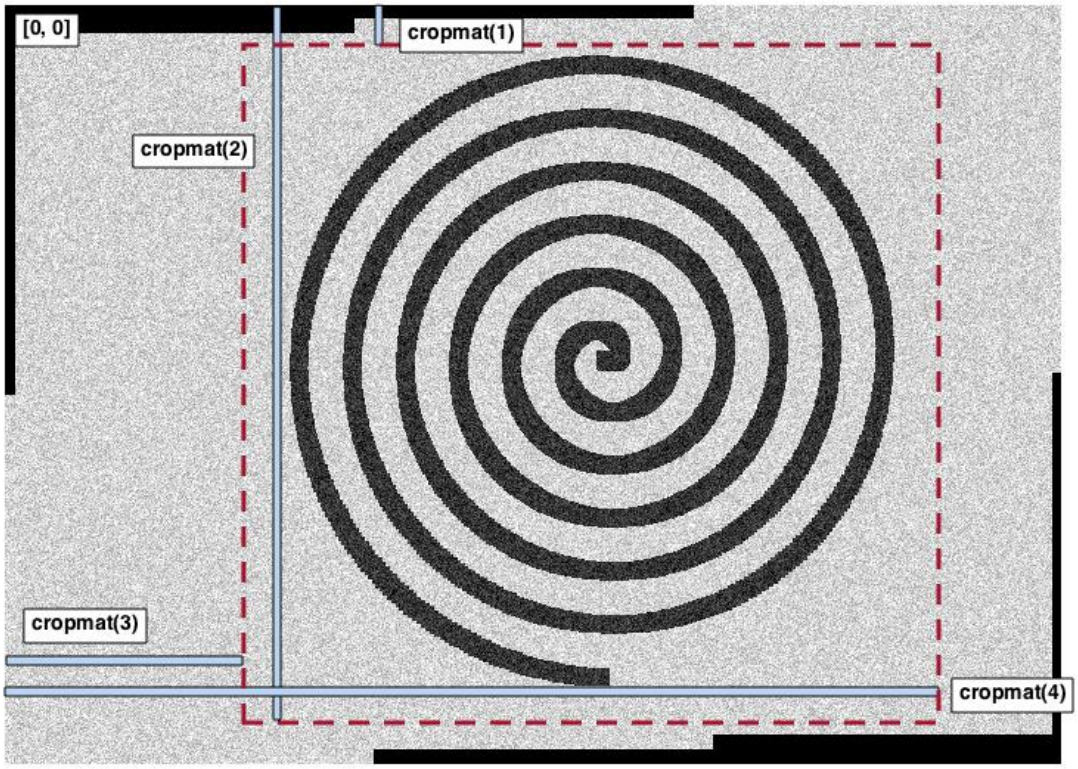
I. OVERVIEW

The 'RP_crop' analysis crops an inputted image. While it can be used in any context, we suggest using it immediately after the 'RP_stitch' analysis to remove any residual blank space left as artifacts from its analysis.

II. HOW TO USE

First, open the 'user_config' text file in your 'Root_Processing' directory. The parameters used in 'RP_crop' are in the 2nd section, and there will be three parameters. In order, they are:

1. image_filename: this is the full image filename (including directory) where the image is to be found.
2. output_filename: this is the full image filename (including directory) where the image is to be saved. If the directory is not present, the analysis will automatically make the directory.
3. cropmat: these are the pixel positions that specify the crop range, specifying the (1) start row, (2) end row, (3) start column, and (4) end column. The origin point (0,0) is in the top left corner. See below for a schematic:



III. RUNNING THE CODE

This analysis can be conducted using the [‘RP_crop’] string in the ‘RP_run’ module.

CHAPTER 5

Getting Started: wc

I. OVERVIEW

The 'RP_wc' analysis creates a volumetric water content map from the specified image using equations outlined in Kang et al., 2013[1]. This analysis assumes that the images are in a quasi-2D (i.e. thin plate) form, with soil held together by two aluminum plates.

Input image must be a neutron transmission image.

II. HOW TO USE

First, open the 'user_config' text file in your 'Root_Processing' directory. The parameters used in 'RP_wc' are in the 3rd section, and there will be eight parameters. In order, they are:

1. image_filename: this is the full image filename (including directory) where the image is to be found.
2. output_filename: this is the full image filename (including directory) where the image is to be saved. If the directory is not present, the analysis will automatically make the directory.
3. b_w: scattering coefficient of water [cm^-2]
4. s_w: attenuation coefficient of water [cm^-2]
5. s_a: attenuation coefficient of aluminum [cm^-1]
6. s_s: attenuation coefficient of silicon [cm^-1]
7. x_s: thickness of soil [cm]. This is the thickness of the soil in the neutron beam direction.
8. x_a: thickness of aluminum [cm]. This is the thickness of the aluminum plates (in total, so both plates) in the neutron beam direction.

III. RUNNING THE CODE

This analysis can be conducted using the ['RP_wc'] string in the 'RP_run' module.

[1] Kang, M et al., "Water calibration measurements for neutron radiography: Application to water content quantification in porous media." Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 708 (2013):24-31.

CHAPTER 6

Getting Started: mask

I. OVERVIEW

The 'RP_mask' analysis creates a binary segmented image from a specified image using a simple local thresholding technique.

In this analysis, each pixel will have an associated window of pixels in all directions - from this, a mean value will be calculated and compared with the pixel of interest. If the pixel is less than the mean value by a set value 'threshold' (alpha in figure 1), then the pixel will be labeled as TRUE and assigned as a mask pixel. If a pixel does not meet this condition but has a pixel value lower than a global threshold 'globthresh' (alpha_global in figure 1), then the pixel will be assigned as a mask pixel.

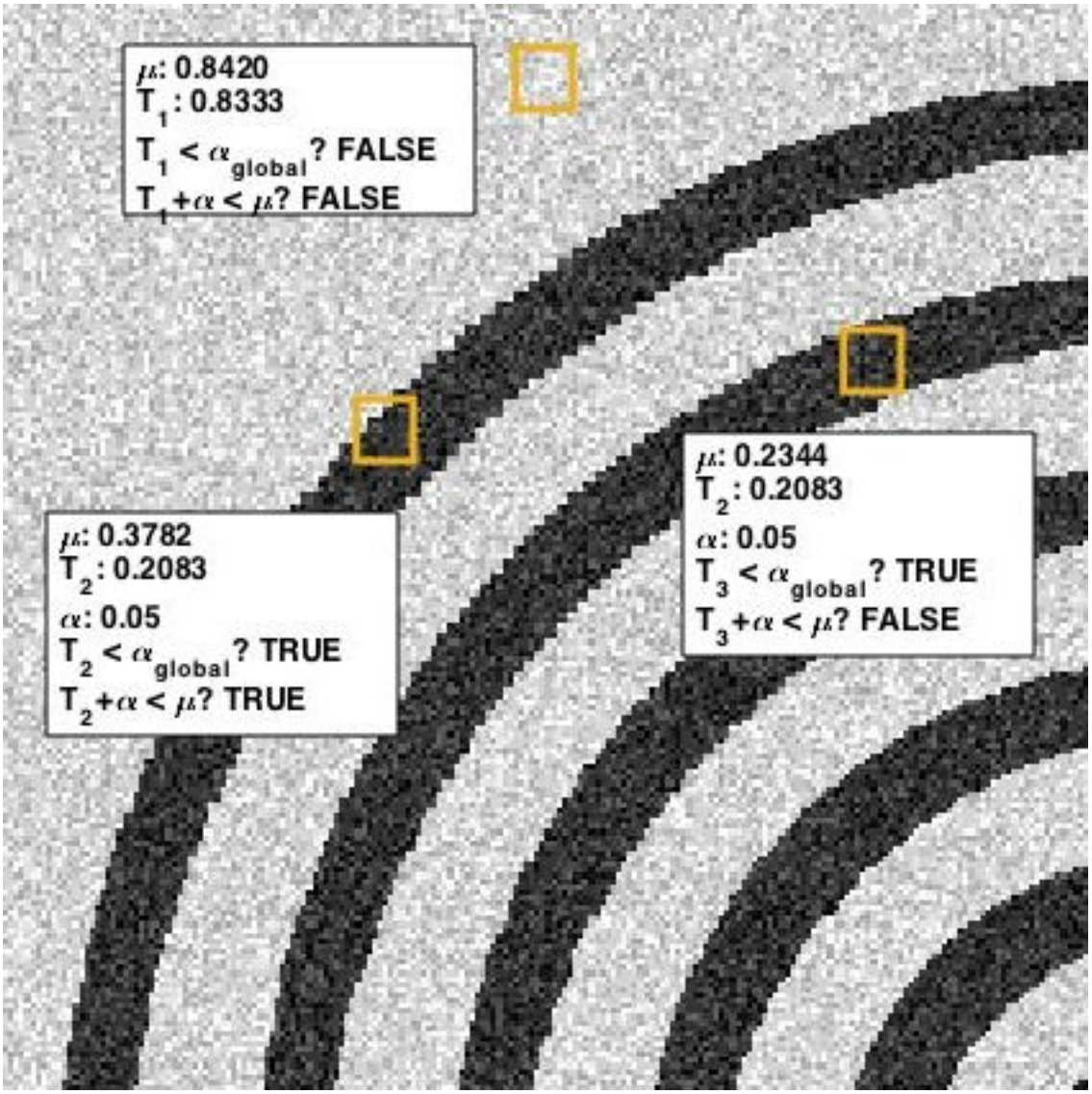


Figure 1: Analysis procedure for three different pixels.

II. HOW TO USE

First, open the 'user_config' text file in your 'Root_Processing' directory. The parameters used in 'RP_mask' are in the 4th section, and there will be five parameters. In order, they are:

1. image_filename: this is the full image filename (including directory) where the image is to be found.
2. output_filename: this is the full image filename (including directory) where the image is to be saved. If the directory is not present, the analysis will automatically make the directory.
3. windowsize: this is the size of the window to be analyzed.
4. threshold: this is the minimum threshold against which the image-mean value difference will be evaluated.
5. globthresh: this is a global threshold value - if the pixel of interest has a value lower than this, then the pixel will be assigned as a mask pixel. This is to avoid an 'outline' effect where the center of objects with a size larger than the window will not be mislabeled due to homogenously dark pixel regions (see T 3 in figure 1).

III. RUNNING THE CODE

This analysis can be conducted using the [‘RP_mask’] string in the ‘RP_run’ module.

CHAPTER 7

Getting Started: imagefilter

I. OVERVIEW

The 'RP_imagefilter' analysis conducts a simple filtering process of an image for later processing. This is primarily used to 'clean up' the output image from the 'RP_mask' analysis, and as such should be used in parallel.

Two main filters are applied: the first is an area-based filter, where all mask-labeled objects with a connected pixel count smaller than 'bwareaval' will be removed. The second filter is a simple median filter.

II. HOW TO USE

First, open the 'user_config' text file in your 'Root_Processing' directory. The parameters used in 'RP_imagefilter' are in the 5th section, and there will be four parameters. In order, they are:

1. image_filename: this is the full image filename (including directory) where the image is to be found.
2. output_filename: this is the full image filename (including directory) where the image is to be saved. If the directory is not present, the analysis will automatically make the directory.
3. bwareaval: this is the scalar value of the minimum pixel count (i.e. pixel area) to be removed.
4. medfilterval: this is the window size to be used for the median filter.

III. RUNNING THE CODE

This analysis can be conducted using the ['RP_imagefilter'] string in the 'RP_run' module.

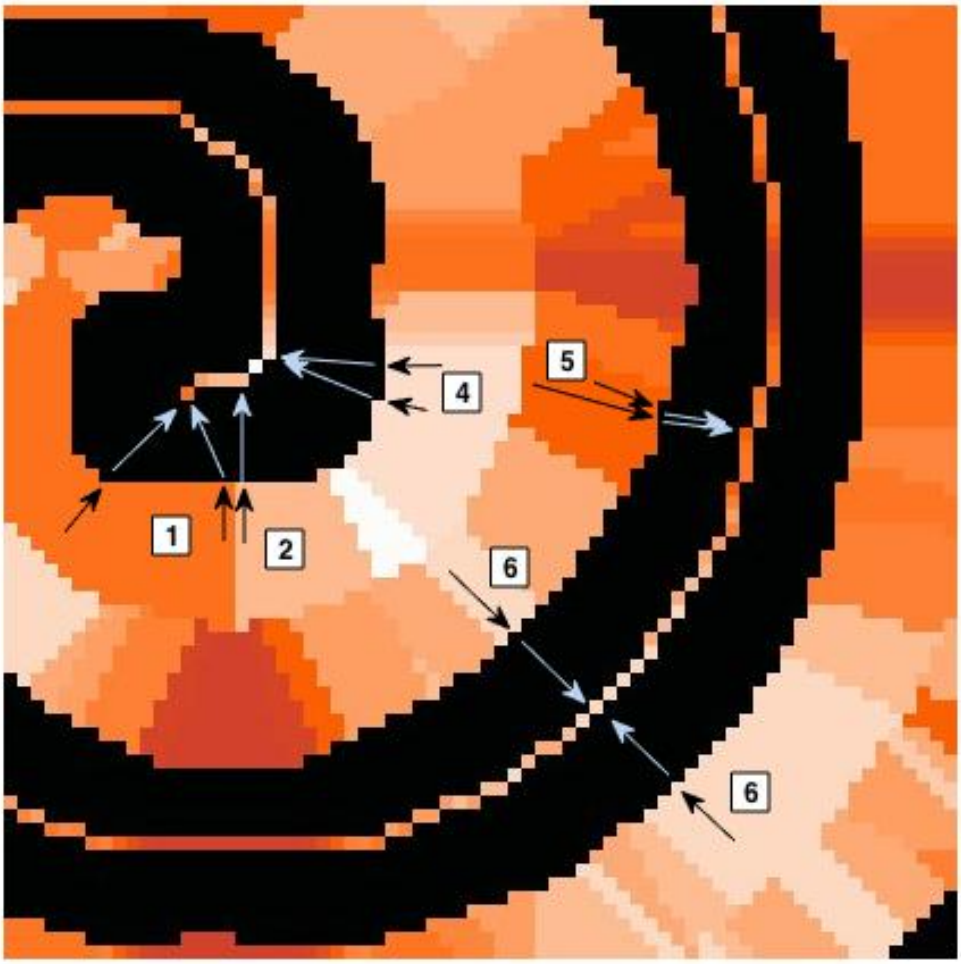
CHAPTER 8

Getting Started: distmap

I. OVERVIEW

Generalizing, the 'RP_distmap' analysis associates every non-mask pixel with the nearest medial axis of a mask object. In the context of a soil-plant system, this means that every soil (i.e. non-segmented, non-root) pixel is associated with a root diameter.

This analysis is done in two steps: firstly, a non-mask (i.e. "soil") pixel is associated with a contour of a mask object. Secondly, this contour pixel is then associated with a medial axis within the mask object. In the image below, note how each set of arrows, grouped by number, operate:



The black arrow indicates the first step, where the pixel is associated with its nearest mask contour. The blue arrow then associates that pixel with the medial axis (colored above for reference). Different colors indicate different medial axis sizes. This effectively creates "zones of influence" for each mask section/medial axis.

II. HOW TO USE

First, open the 'user_config' text file in your 'Root_Processing' directory. The parameters used in 'RP_distmap' are in the 6th section, and there will be three parameters. In order, they are:

1. image_filename: this is the full image filename (including directory) where the image is to be found.
2. output_filename: this is the full image filename (including directory) where the image is to be saved. If the directory is not present, the analysis will automatically make the directory.
3. maxval: this indicates the maximum non-mask/mask contour distance to be evaluated, above which all non-mask pixels will be ignored. This effectively delimits the "zone of influence" for each mask. This is primarily to save time on analyses for very large images.

III. RUNNING THE CODE

This analysis can be conducted using the ['RP_distmap'] string in the 'RP_run' module.

CHAPTER 9

Getting Started: radwc

I. OVERVIEW

The 'RP_radwc' analysis creates two size NxM text files: (1) the water content of a given root radius (row) at a given distance from the root (column), and (2) the number of pixels analyzed for each entry. The analysis also outputs a size 1xN and 1xM text file of the root radius values and distance from the root, respectively. All distance and radius values are outputted in terms of pixels.

In calculating the water content, for each entry of radius i and distance j, this analysis takes the mean value of all pixels that fit these criteria.

II. HOW TO USE

First, open the 'user_config' text file in your 'Root_Processing' directory. The parameters used in 'RP_radwc' are in the 7th section, and there will be six parameters. In order, they are:

1. wc_filename: this is the full image filename (including directory) where the water content image is to be found.
2. distmap_filename: this is the full image filename (including directory) where the image outputted from 'RP_distmap' is to be found.
3. mask_filename: this is the full image filename (including directory) where the mask image is to be found.
4. output_filename: this is the directory where the outputted files will be saved.
5. fileformat: this is the file prefix that will be attached to the four files outputted by this analysis. So, for example, if 'fileformat' is set as 'SampleImg', then the 4 files outputted will be:
    - 'SampleImg_data_distrange.txt': 1xM text file of distance from the root
    - 'SampleImg_data_num_xrad_ydist_wc.txt': NxM text file of number of pixels analyzed
    - 'SampleImg_data_radrange.txt':1xN text file of root radii analyzed
    - 'SampleImg_data_xrad_ydist_wc.txt': NxM text file of mean water content for each root radius at a given distance
6. pixelbin: the number of pixels to be binned along the 'distance from root' axis when determining water content

III. RUNNING THE CODE

This analysis can be conducted using the [‘RP_radwc’] string in the ‘RP_run’ module.

CHAPTER 10

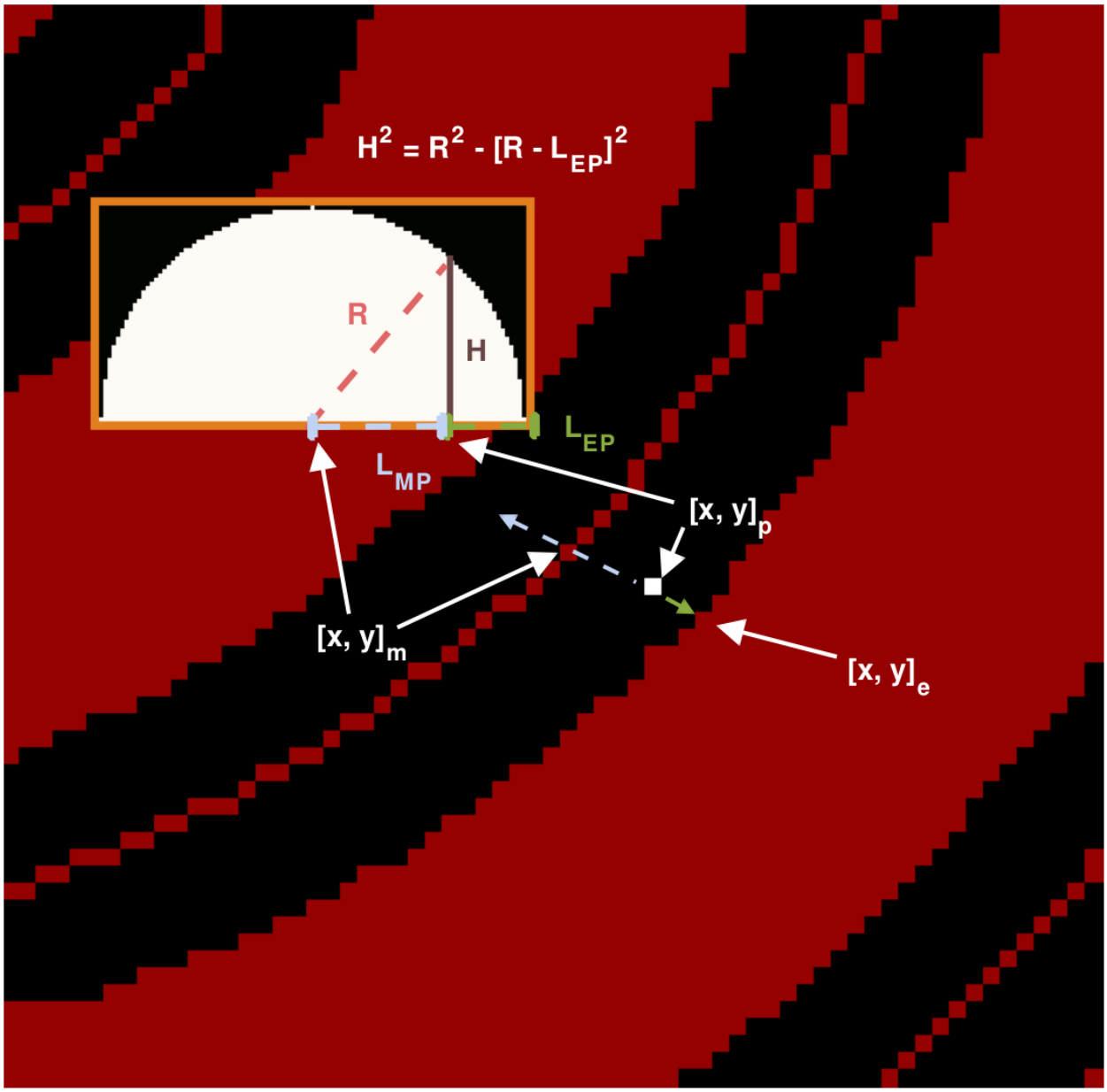
Getting Started: thickness

I. OVERVIEW

The 'RP_thickness' analysis creates a half-thickness image from a binary segmented image, assuming a cylindrical shape.

In this analysis, a skeleton (i.e. medial axis transform) of the binary image is calculated. A distance transform of the root is then calculated - the distance transform on the medial axis pixel is labeled a "root radius" value R.

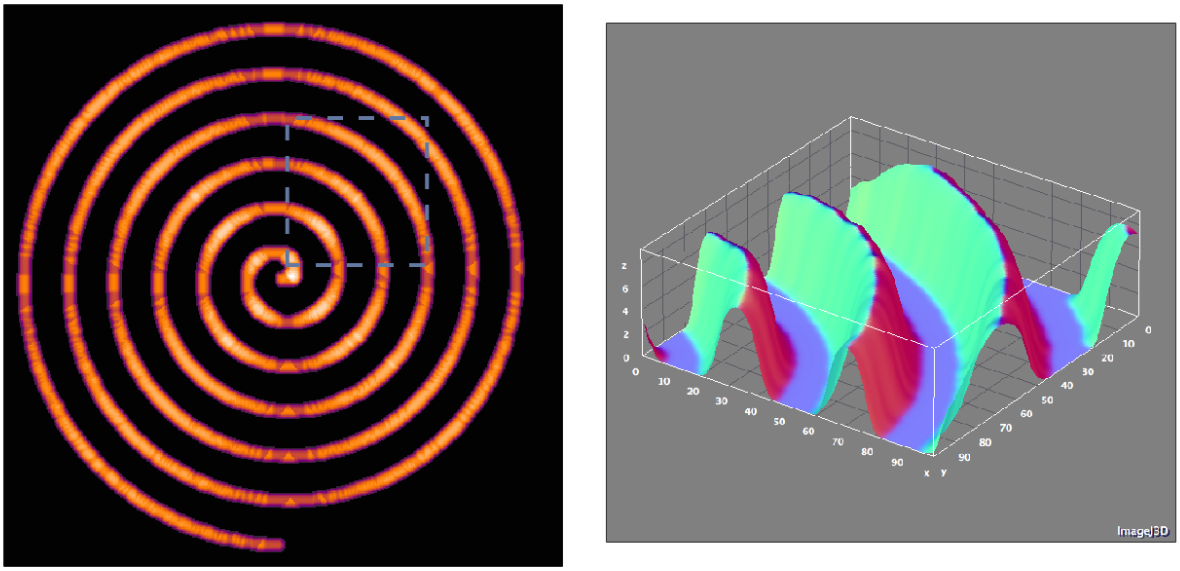
From here, for every pixel [x, y]<sub>p<sub>, the closest root edge pixel [x, y]<sub>e<sub> that doesn't intersect the medial axis is found. We calculate the edge-pixel distance L<sub>EP<sub>. Then, we extend the line between these two points, and extend it in the opposite direction, identifying the closest medial pixel [x, y]<sub>m<sub>, whose path consists entirely of the root (e.g. no medial axes that are located on other root segments). From here, we use the R value assigned to [x, y]<sub>m<sub>. Then, assuming a cylindrical distribution around the medial axis, we can calculate the half-dome height H of the pixel as follows:



Outline of the individual components in the thickness analysis.

We assume that L_MP and L_EP are on an equal plane, thereby making R = L_MP + L_EP. Then, assuming a cylindrical distribution around the medial axis, we can calculate the half-dome height H of the pixel as follows:

$$H^2 = R^2 - (R - L_EP)^2$$



Final product of the thickness analysis, with a 3D surface image of a selected area. Note that the surface image is not 3D due to different scaling between the z and xy axis.

NOTE: the output values are the half-dome height of the root - if the full thickness of the root in the cross-sectional direction is desired, multiply all values by 2.

II. HOW TO USE

First, open the 'user_config' text file in your 'Root_Processing' directory. The parameters used in 'RP_thickness' are in the 8th section, and there will be two parameters. In order, they are:

1. image_filename: this is the full image filename (including directory) where the image is to be found.
2. output_filename: this is the full image filename (including directory) where the image is to be saved. If the directory is not present, the analysis will automatically make the directory.

III. RUNNING THE CODE

This analysis can be conducted using the ['RP_thickness'] string in the 'RP_run' module.

CHAPTER 11

Getting Started: rootdiameter

I. OVERVIEW

The 'RP_rootdiameter' analysis creates a text file of the root diameter distribution using the root mask image. The user has an option to bin the root diameter values, and all results are shown in terms of pixels.

II. HOW TO USE

First, open the 'user_config' text file in your 'Root_Processing' directory. The parameters used in 'rootdiameter' are in the 9th section, and there will be three parameters. In order, they are:

1. mask_filename: this is the full image filename (including directory) where the mask image is to be found.
2. output_filename: this is the full image filename (including directory) where the outputted files will be saved.
3. bincount: the number of bins the root diameter is to be placed into.

III. RUNNING THE CODE

This analysis can be conducted using the ['RP_rootdiameter'] string in the 'RP_run' module.

CHAPTER 12

Indices and tables

- genindex
- modindex
- search