Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Graphics and Algortihms
Vienna University of Technology
Favoritenstr. 9/186-3
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18661
Fax: +43 (1) 58801-18697
E-mail: ines@prip.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-131 October 23, 2013

TR Examination of root development by means of topological image analysis

Ines Janusch

Abstract

The focus of this work is to examine root plant development by means of topological image analysis. As roots grow and develop they change their shape (for example when branches are formed). Due to this aspect this application is well suited for a topological analysis. The aim of the scientific practical is to represent the root images as Reeb graphs. A method to compute critical points using a height function and to build a Reeb graph based on these points, is presented. All nodes in the graph are classified according to their type and their color. There are four different types of nodes in a Reeb graph: minima, maxima, saddle nodes and regular nodes. When storing color information with the nodes and interpret the graylevels as levels of elevation, the nodes can be further classified according to their elevation.

1 Introduction

Shape description by means of topological image analysis is a good choice in this context, as roots change their shape when they are growing (for example branches are formed). A shape description based on contours and shading is due to this not suitable here. But topological attributes are necessary to allow for an abstraction of shape according to invariant aspects of geometry. Topology can be seen as a method for shape understanding. A topological exploration of the shape according to quantitative geometric properties provided by real-valued functions defined on the shape is performed. Different choices of functions allow for different properties to be considered [5]. The first milestone in the examination of the root development using topological persistence is to build Reeb graphs representing the dataset images."Reeb graphs are compact shape descriptors which convey topological information related to the level sets of a function defined on the shape."[4, p. 1] Therefore these Reeb graphs are compact representations of the underlying images. An analysis of the Reeb graphs allows for categorization of the nodes of the graph according to the neighboring nodes.
These Reeb graphs can be of use in several applications: Plant characteristics, for example the number of branches in a root or the position and number of branch-endings, can be read off the graphs. Moreover with the projection of a 3-dimensional structure (here the roots) to a 2-dimensional space, overlaps of this structure with itself may arise in the 2-dimensional representation. To distinguish between a branch and an overlap may not be easy when using an analysis simply based on contours. Using a topological analysis an overlap and a branch can be easily distinguished, as for an overlap the Reeb graph representing the structure holds a loop that comes along with the overlap. Another application for Reeb graphs lies in the comparisons of images. The Reeb graph can be seen as a skeletonized and therefore simplified representation of the image content. Such an approach is described in [7].
The paper is organized as follows: the dataset is presented in Section 2, the segmentation approach used is discussed in Section 3 as this approach uses combinatorial maps, a brief insight into this topic is given in Section 4. Section 5 discusses the data-structure used. A theoretical overview of critical points and Reeb graphs is given in the Sections 6 and 7. The actual computation of these critical points, the building of the Reeb graph and the classification of nodes within the graph are shown in the Sections 8, 9, 10 and 11. Section 12 and 13 show the results.

2 Data set

There are two different datasets analyzed in this paper: a synthetical test dataset and the root dataset. The test-dataset was created to serve as ground truth and to allow for a quick evaluation of this approach.
The approach is tested and evaluated on the test dataset before examination of the root data is done.

2.1 Test dataset

A test dataset containing possible types of regions was assembled. The images are synthetically created and range from simple images of only one foreground region to more complex ones with several foreground regions of different gray-values. To generate these more complex images a distance transformation was applied to some of the simpler images. In order to obtain clear regions, the distance transformed image is spread so that it covers the whole 8bit intensity interval of [0,255]. Next the regions are formed according to gray values; all values of [0,255] are mapped to the nearest of the six values: 0, 50, 100, 150, 200 and 255.

The dataset consists of 23 gray-scale images showing 1 to 7 foreground regions and up to 2 holes in the foreground structure. The whole test-dataset used here is shown in Appendix A.

2.2 Root dataset

The root images dataset was acquired by the research group of Dr. Wolfgang Busch at the Gregor Mendel Institute of Molecular Plant Biology. For this dataset images of the plant Arabidopsis thaliana are taken. This plant is a model organism which is widely used in plant sciences, due to the small size of its genome, the small size of the plant itself and its rapid life-cycle[10]. The plants are grown in liquid between glass plates. All plants in one plate belong to one dataset. One dataset/plate consists of 2 rows of 12 plants. The plates are placed in a growth chamber that allows for controlled conditions as constant temperature or humidity.

There are two types of data obtained:

Type 1: Microscopy images on cellular resolution showing the root only: A root needs to be colored using a fluorescent contrast agent that is not able to enter the cells but stays in the cell membranes, coloring them in this way. The data is then obtained by means of confocal microscopy. Therefore the object is at no time pictured as a whole, but it is sampled by a laser in points. Only one point of the object is illuminated at a time. All points pictured in this way are then combined to one image of the object. This approach increases the optical resolution and the contrast. As the laser is able to enter the root (the laser can enter a medium for a maximum of 150 μm, the roots are about 120 μm thick) 3D data is acquired, as there is a stack of 2D images of different layers of the root.

Type 2: Images of the whole plant (including root and leafs): These images are taken using an image scanner. A special fixture allows for two datasets to be placed in an exact known position inside the scanner. The images

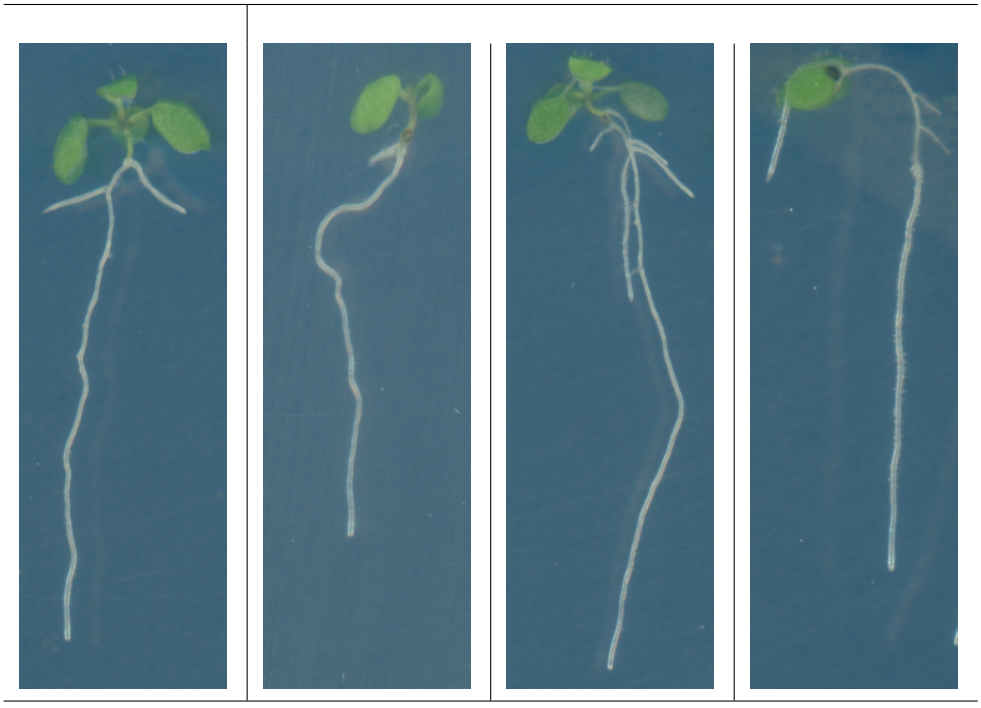


Figure 1: Example images of root dataset

are acquired with a scan at 1200 dpi resolution with 8bit color depth, therefore one image is of approximately 6000x6000 pixels in size. The images are stored as bmp files of about 150MB. Along time several images are acquired this way as each plate is scanned at several days of the growth process. A 3D stack of 2D images over time is thus created for each root.

This work considers the scanned images of the whole plants (type 2 as mentioned above) only. Moreover the images used are pre-processed to simplify the work with such large images. The 24 plants per plate are cropped to single images: one image per plant with an image size in the range of 500x1300 to 800x1300 pixels resolution and a file size of 1,5-2,5Mbyte. Example images of this dataset are shown in Figure 1.

The whole set of plant images used here consists of 9 sets of time series. Each set holds 6 images of one plant taken over time (day 1, day 4, day 8, day 12, day 16, day 20). Of these 54 images, 34 images are analyzed, the other images are too early in the growth process and therefore to small in structure to be represented by a non-trivial Reeb graph. All images analyzed consist of 2 foreground regions (leaves and root) and up to 3 holes in the foreground structure. The dataset is shown in Appendix B.

3 Image segmentation

An image segmentation is needed first, as all following steps are performed on the segmented data. The segmentation results are checked and corrected manually to generate ground truth.

There are several approaches available for this segmentation step. A pyramid based approach is used here. Image pyramids describe the content of an image at multiple levels of resolution; from a high resolution at the base to a coarse resolution at the top of the pyramid. Lower resolutions are computed using a reduction function which takes as input the cells in the reduction window[13].

The approach given in this paper was done using graph pyramids (based on [13]):

Images can be represented as graphs, each pixel can for example be described as a vertex, connected by edges to neighbouring vertices (=pixels). A graph pyramid is a pyramid where each level is a graph consisting of vertices V and edges E. Graph pyramids can be used for image segmentation. On the base level each pixel (each vertex) is a homogenous region. On the next level certain homogeneity criteria for regions are checked. If these conditions are fulfilled regions are merged. „Using 2D images, combinatorial maps may be understood as a particular encoding of a planar graph, where each edge is split into two half-edges called darts.“[9, 3] Using a stack of region adjacency combinatorial maps an irregular combinatorial pyramid is built [11]. „A combinatorial map in two-dimensions is composed of vertices, edges and faces, respectively defined as cells of 0, 1 and 2 dimensions and noted i-cells.“[9, 2]

Definition for a two-dimensional combinatorial map:
“A two-dimensional combinatorial map (or 2-map) M is a triplet $M = (D, \beta_1, \beta_2)$ where:

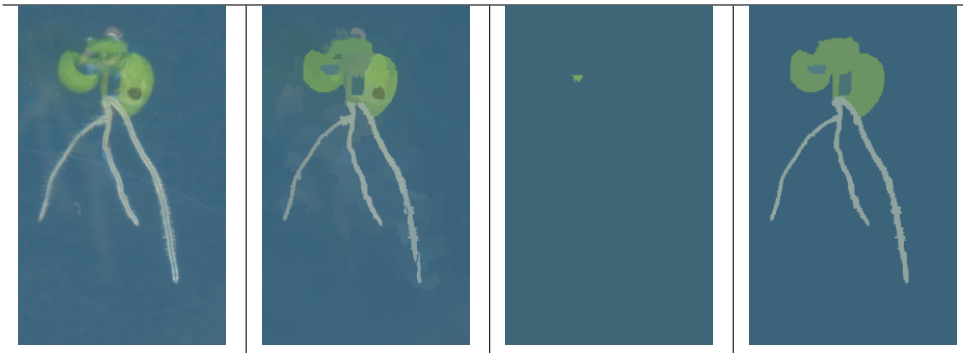
- D is a finite set of darts;
- β_1 is a permutation on D;
- β_2 is an involution on D. “[9, 2]

" Each dart belongs to a single vertex, edge and face of the map. The β_1 permutation links each dart of a face to the next dart encountered while turning clockwise around the face. The β_2 involution links each dart of an edge to the other dart of the edge which has an opposite orientation. " [9, 3]

One advantage when using a segmentation approach based on combinatorial maps is, that there is no need to detect region borders as these are already defined by the darts in the combinatorial map. The positions of such darts in the image can be obtained by exploiting the hierarchy of the segmentation pyramid. At the bottom level each pixel forms a single region. The position of the darts describing the one-pixel region are therefore known.

The implementation of the segmentation used, takes an arbitrary image as input and returns a hierarchy of segmented images. The segmentation based on an irregular graph pyramid can be changed by user input: in a graphical user interface the user can place its modifications[8]. The user has the possibility to merge regions or prevent them from being merged in higher levels of the hierarchical segmentation (as shown in Figure 2). For changes made by the user the pyramid is recomputed to adapt the underlying combinatorial maps [8]. A metadata file for each level of the hierarchy is generated as well. This file contains labeled regions (one label for each connected component) which correspond to the segmented regions in the image.

As illustrated in Figure 2, the automatic segmentation is not able to segment the given root correctly. On the coarsest level of the segmentation (level 31) the root is even entirely missing. With the possibility to inhibit some segments on a level of fine segmentation by user input, it is possible to correctly segment the root and the background as shown in the manual segmentation example. Therefore by using this segmentation approach ground truth in terms of a distinction between foreground and background can be generated. This user interaction was needed for all images in the dataset as a correct segmentation was not possible based on the automatic segmentation alone (see Figure 2). The segmentation takes about one hour and a half per root image, this sums up to a total of 50 hours for the segmentation step.
Despite using the interactive segmentation, some region borders were not



original image	automatic segmentation level 18	automatic segmentation level 31 (top level)	manual segmentation

Figure 2: Levels of segmentation

segmented correctly but appear frayed due to the segmentation process (see Section 10.1). This problem appears mostly for images in the dataset that were taken on day 16 of the plant's growth period. For the images taken on day 16 the glass surface appears stained when compared to images of other days which leads to these segmentation artifacts.

4 Correspondence: image - combinatorial map

The combinatorial map is based on vertices, edges and faces which describe the underlying image. However these vertices, edges and faces are not the vertices, edges and faces a human would use to describe an image.

Figure 3 shows a white rectangle on a black background. Describing this image using vertices, edges, faces we can describe this image with 4 vertices (the 4 corner points of the rectangle), 8 edges (the 4 edges bordering the white rectangle and 4 edges forming the border of the black area) and two faces (the white region of the rectangle and the black region of the background).
Using a combinatorial map this image is described with 2 vertices, 3 edges and 3 faces. This map is shown in figure 4. Vertices are marked by circles and labeled with red numbers, edges are marked by lines, labeled with blue numbers and faces are white regions labeled with gray numbers (there are 3 faces, as the combinatorial map extends the image by a virtual background face labeled 0 in this example). The basic element of a combinatorial map, the darts, are labeled using green numbers here [14].
"Each dart belongs to a single vertex, edge and face of the map. The β_1 permutation links each dart of a face to the next dart encountered while turning clockwise around the face. The β_2 involution links each dart of an edge to

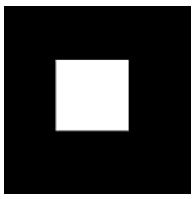


Figure 3: simple image: white rectangle on black background

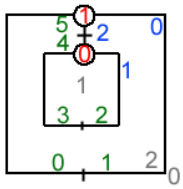


Figure 4: corresponding combinatorial map

the other dart of the edge which has an opposite orientation.[9]"
Although darts describe the borders of region, these darts do not directly contain information about the location of these borders in the image. By using a hierarchical segmentation approach it is possible to trace the darts back to the bottom level. On this level each pixel forms one region described by a dart and the location of these darts is known. Using the pyramidal structure the border locations thus can be obtained at higher levels as well.

"There are two possible methods for the representation of the region-boundaries:

1. The boundaries may be inside-pixel-boundaries or outside-pixel-boundaries. However, this approach has a severe drawback:

   - 2 adjacent regions do not share the same boundary element

2. Alternatively define the boundaries on a set that is retrieved by translating (shifting) the original image domain Z^2 by $(-1/2, -1/2)$. This new set is called half-integer plane or boundary plane as opposed to the image plane. The boundaries defined in the half-integer-plane are called inter-pixel-boundaries." [12]

In the following we will use inter-pixel-boundaries. Details on the implementation are given in Section 5.

5 Data structure

The segmentation pyramid is saved layer-wise as a label-image. These files are imported into the Matlab environment.
At the bottom level of the pyramid each pixel is seen as a region and therefore represented by linked darts. As pixels and regions are merged darts are removed/contracted.
The whole pyramid can be stored as a list in the size of the maximum number of darts (number of darts on the bottom level).

However, for this approach the darts are not linked and resorted for all levels of the pyramid:
A list of all pixels is generated for the bottom level. The location of such a pixel in the image can be obtained easily due to the naming convention used: the pixels are labeled by successive numbers. For the bottom level of the pyramid, we start with the pixel in the upper left corner of the image and label it 1, we then label the whole first row with ascending numbers, move to the next row and label it with further ascending numbers in a left to right order. Therefore the top left pixel is named 1 while the bottom left pixel is named $n * m$ for an image of size n x m.Next the final segmentation on the top level is analyzed. For each region in this segmentation result, the pixels forming the inside-pixel boundary are located. These pixels are now linked in a clockwise order around the region, all pixels that do not belong to a border are left unlinked.

6 Critical points

„A point (a, b) is called a critical point (or a stationary point) of a function $f(x, y)$ if both derivatives $f_x(a, b) = 0$ and $f_y(a, b) = 0$, or if one of these partial derivatives does not exist. If f has a local maximum or minimum at (a, b), then (a, b) is a critical point of f. However, not all critical points give rise to maxima or minima. At a critical point, a function could have a local maximum or a local minimum or neither.“[15, p. 959]
Such a critical point can either be a degenerate or a non-degenerate critical point. These two cases can be distinguished via the Hessian matrix. The Hessian matrix is the square matrix of second-order partial derivatives of a function. The determinant of the Hessian matrix at a critical point x is then called the discriminant. If this determinant is zero then x is called a degenerate critical point of f (or non-Morse critical point of f). Otherwise it is non-degenerate (or Morse critical point of f).

„Let M_d denote a d manifold with or without boundary. A smooth, real-valued function $f : M_d \to \mathbb{R}$ is called a Morse function if it satisfies the following conditions:

- all critical points of f are non-degenerate and lie in the interior of M_d,
- all critical points of the restriction of f to the boundary of M_d are non degenerate
- for all pairs (p, q) of distinct critical points of f and its restriction to the boundary, $f(p) \neq f(q)$.“[6, p.4]

Critical points of such a real-valued function are those points where the gradient becomes zero. At regular points no topology changes occur. Topological changes occur at critical points only. In 2D these are minima, maxima or saddles [6].

7 Reeb graph

"Reeb graphs are compact shape descriptors which convey topological information related to the level sets of a function defined on the shape."[4, p. 1] A Reeb graph is a topological graph. $\mu : M \to \mathbb{R}$ is a continuous, scalar function on a compact manifold. The Reeb graph M concerning μ is the quotient space of M[4]. Vertices of the Reeb graph correspond to critical points of the function (points where the topology of M changes), edges describe topological persistence[4].
These critical points / nodes in a Reeb graph are shown in Figure 5. In order to categorize a critical point all neighboring points in the Reeb graph are analyzed. "In the case of saddles, the corresponding node has degree 3 if the saddle merges/splits components, and degree 2 if it is a genus modifying saddle."[6, p. 5]

Degenerated critical points (= non-isolated critical points such as plateaus or flat areas) represent a typical problem. Therefore extended Reeb graphs were introduced[4]. The Extended Reeb Graph is able to faithfully represent the surface shape, even in case of degenerate critical points[4]. The concept of critical points is hereby extended to critical areas. There are simple and complex maximum / minimum areas as well as simple and complex saddle areas[4]. The distinction among the different types of critical areas is done by analyzing the coordinates of the vertices which are edge-adjacent to the

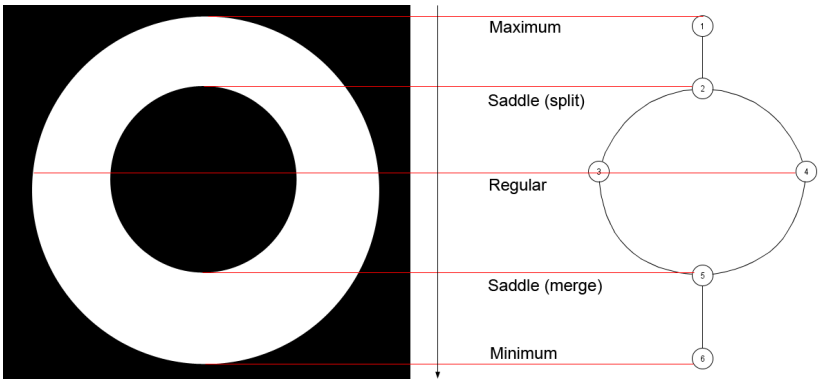


Figure 5: Different types of critical points and corresponding nodes in the Reeb graph.

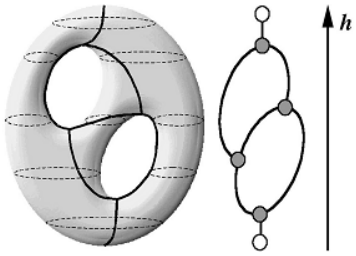


Figure 6: "The Reeb graph of a bitorus and some cross sections. Each contour determines an equivalence class which is represented in the Reeb graph by a single point."[2, p. 5]

region boundary that is, by checking the ascending/descending directions[3].

Once the critical points are computed the Reeb graph will be built based on these points as its nodes (see Figure 6 for an example of a Reeb graph). As the critical points occur with changes in topology (changes in the number of components when analyzing the root in direction of growth), a Reeb graph built on these nodes can be seen as a kind of skeleton representation of the root.

When organizing the nodes and arcs in the Reeb graph according to their position in the image, proportions of the roots characteristics are taken into

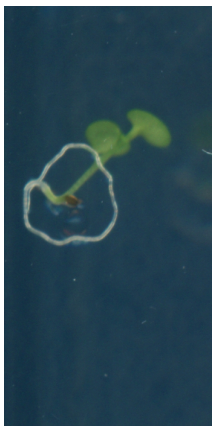


Figure 7: Example of a root with changing direction of growth, dataset "complex"- root001 day 5

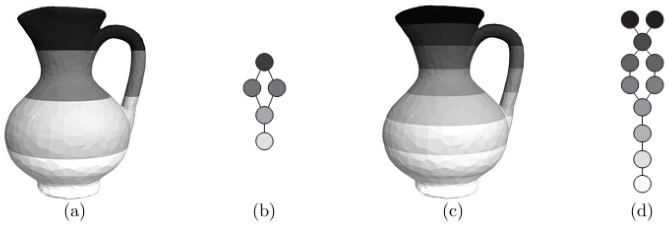


Figure 8: "Construction of the Reeb-graph encoding topological features of a surface with respect to the elevation of surface points. (a) Level sets induced by quantization of elevation values into 4 levels. (b) The Reeb-graph of (a). (c) Level sets induced by quantization of elevation values into 8 levels. (d) The Reeb-graph of (c)."[1, p. 12]

account. This allows for a more reliable representation of the root images.

For roots with a main direction of growth that is not perpendicular to the rows of the image or with changing direction of growth (as shown in Figure 7), the Reeb graph built in this way might not describe branches and merges in the root in a correct way (due to overlappings). Using the height function to find critical points might describe several independent components although they form only one component.
By changing the base function of the filtration this issue can be handled. Fore example Eccentricity Transform or a strategy similar to step 1 in [7] could be used instead of the height function to compute the critical points.

The nodes of the Reeb graph can be attributed with color / graylevel information. This allows for further differentiation between saddle points. In order to describe changes of color in a component, regular nodes will be needed in such a Reeb graph (as shown in Figure 8).
This approach will be of particular use when working with the test dataset. With the root data, an analysis of the color information is not needed, as the roots are all similar in color and shading. What is of interest though, is the location of the transition of leaves to root. This location can be found by analyzing the color distribution in the root. So storing color information with the Reeb graph nodes can be used here.

8 Computation of critical points

The underlying idea is to analyze the border of flat-regions in the image. Critical points indicate topology changes and these changes might only occur at the border of a region but not within a region. Therefore the border of a region needs to be examined in order to find such critical points.

The two types of critical points that can be used with the root data are:

1. Critical points along the outer edge of the root. These occur with topological changes in the shape of the root like branches.
2. Critical points inside the root. Here the gray levels are understood as encoding of elevation, a kind of terrain model is built this way. These critical points occur in places of zeros of the first derivative.

As all roots are similar in color and gray levels the second approach might not allow for a reliable classification of a root. Therefore we will concentrate on the first type of critical points (along the outer edge of the root) only.

8.1 Critical points along the outer edge of a root

As in the image in Figure 9 the roots are pictured in the direction of natural growth, thus the leaves are in the top part of the image and the roots are in a mainly vertical direction facing downwards. The main direction of growth of roots is downwards and branches occur mainly in this direction. Therefore the height function can be used as measuring function here.

A critical point in this case indicates either a birth or death of a component. There are 4 different cases to distinguish:

1. birth of a component while other components remain unchanged
2. birth of a component by splitting one component into two components

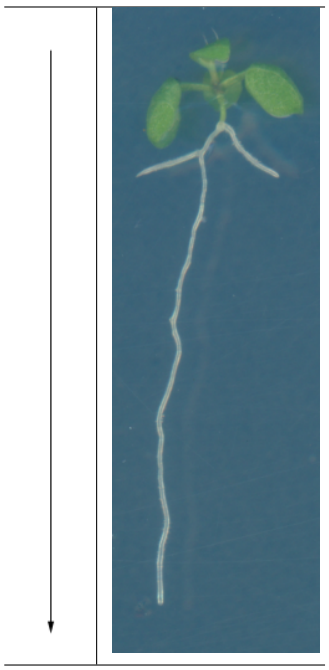


Figure 9: Example of a root with vertical direction of growth

3. death of a component while other components remain unchanged
4. death of a component by merging two components to one component

Figure 10 illustrates these four different types of critical points.

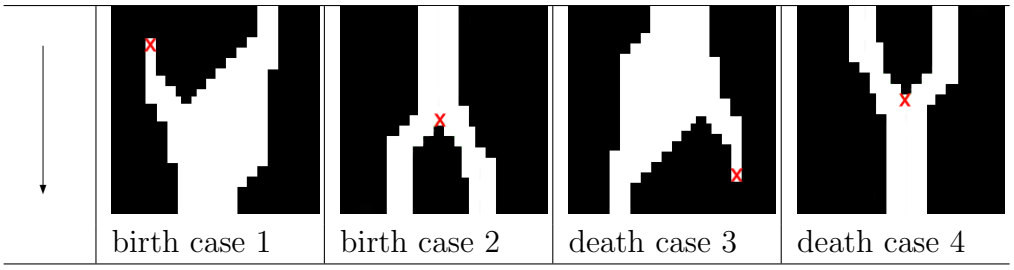


Figure 10: Four different types of critical points: critical points are marked by red crosses.

The critical points are computed based on the region borders obtained during segmentation. As the height function is used to compute the critical points, the region borders are analyzed with regard to horizontal borders as these might describe a change in the number of components. For each horizontal border its endpoints are checked for connected vertical borders:

- no change in topology - no critical point: one endpoint connected to a vertical border going up, the other connected to a vertical border going down
- split or birth: both endpoints connected to a vertical border going down
- merge or death: both endpoints connected to a vertical border going up

The so found critical points are located at the center of such a horizontal border.

As changes in the color information are described as well, a fifth type of node is introduced: changes in color information are described by regular nodes. For each critical point its position, the color at its position and its type (birth, split, merge, death, regular node) are stored.
To distinguish between the five types of nodes, for each node of type birth or death certain neighboring pixels (in a 4-neighborhood) are checked. For nodes of type birth the neighboring pixel in the previous row is checked, if this pixel belongs to a background-region, the critical point is of type birth, if it belongs to a foreground-region the critical point is of type split. Similar to this, for a pixel of type death the neighboring pixel in the following row is checked. If this pixel belongs to a background-region the critical point is of type death, otherwise it is of type merge. The simultaneous occurrence of a critical point of type split and merge indicates a change in the color information and therefore a regular node, in this case the two critical points originally computed (split and merge) are substituted by a regular node.

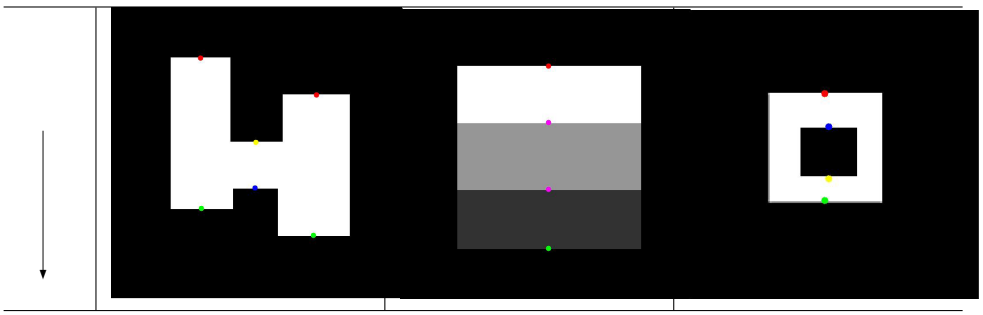


Figure 11: Example images with critical points marked.

Figure 11 shows the critical points computed for three different images. The critical points are marked in five different colors: red for critical points of type birth, green for type death, blue for type split and yellow for type merge, regular nodes due to changes in the color are marked in pink. For more results on the test dataset or the root dataset see Section 12 and 13.

9 Building the Reeb graph

Based on the computed critical points the Reeb graph is built.
Therefore these critical points need to meet the conditions of Morse functions (see Section 6). If these conditions are not met the Reeb graph can not be built.

To determine the connections between two critical points the following approach which resembles a flood fill algorithm is used:

1. The critical points are sorted according to the rows in an ascending order to be processed sequentially (starting with the topmost). For each critical point the successive rows are checked for critical points in a raster scan order.
2. The row of a critical point is followed to the right of it until a pixel belonging to a background region is reached.
3. Here the search is moved one row down and this row is followed to the left until a background pixel or a critical point is found.
4. If a background pixel is found the search is moved to the next row. In case of a critical point, the search terminates.
5. If the critical point that was chosen as start point is of type split, the search for the two connected critical points is carried out to the left and the right of the critical point.

The Figures 12, 13 and 14 show examples for such a search: The white region is the foreground region, the critical points are labeled and highlighted in purple. The green and blue marked regions show the regions that are visited during the search for a critical point. The numbers describe the order in which the pixels are checked. A search starting with critical point 1 (Figure 13) and starting with critical point 2 (Figure 14) is shown.

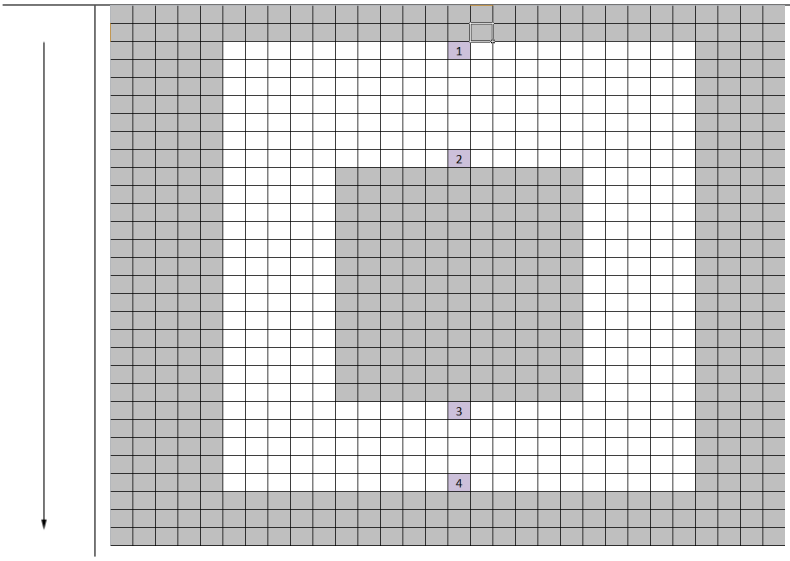


Figure 12: Example image, foreground area marked white

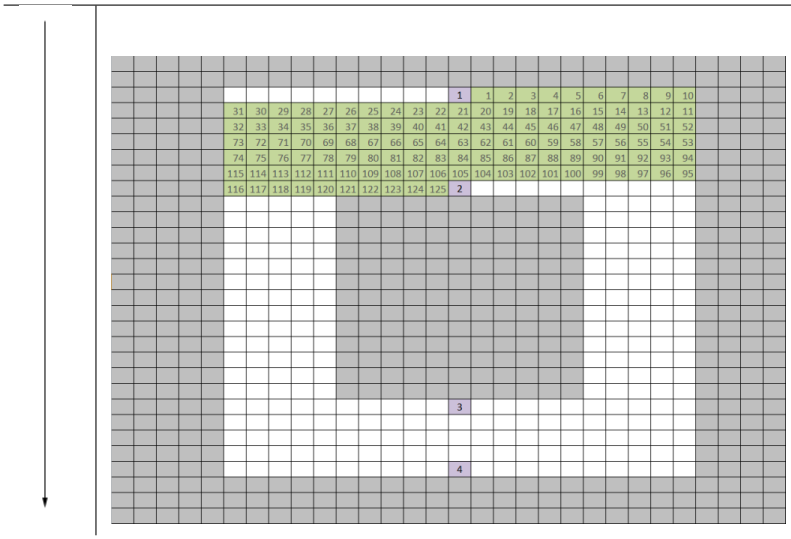


Figure 13: Search performed when computing adjacency of critical points.

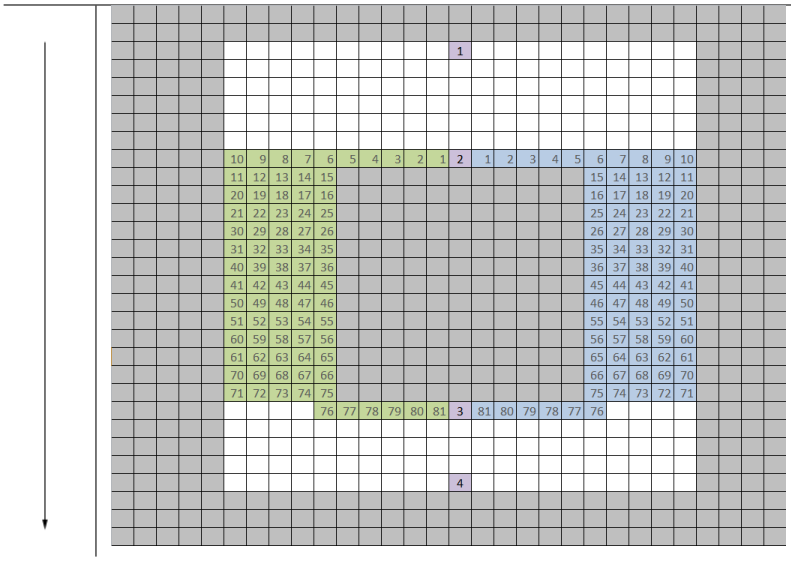


Figure 14: Search performed when computing adjacency of critical points.

An adjacency matrix is created for the critical points and the connections are marked in the adjacency matrix. As the Reeb graph is an undirected graph it suffices to only put one entry in the adjacency matrix for an edge, this means for an edge between point i and j the adjacency matrix is set to 1 for the element (i, j) but not for (j, i). In case of a cycle in the graph, a critical point of type split is connected to a critical point of type merge by two edges. This cycle is described in the adjacency graph, by setting both coordinates describing the split and the merge node in the matrix to 1.

Based on this adjacency matrix the Reeb graph is drawn. There are two types of graphs generated as output:

1. Reeb graph shown as a layer on top of the segmented image: the graph is drawn as simple nodes connected by arcs. The default color is set to red for both arcs and nodes.

2. Reeb graph without the image layer, the nodes are drawn as circles showing the color stored with the critical point and are labeled. The default color of the arcs is set to black.

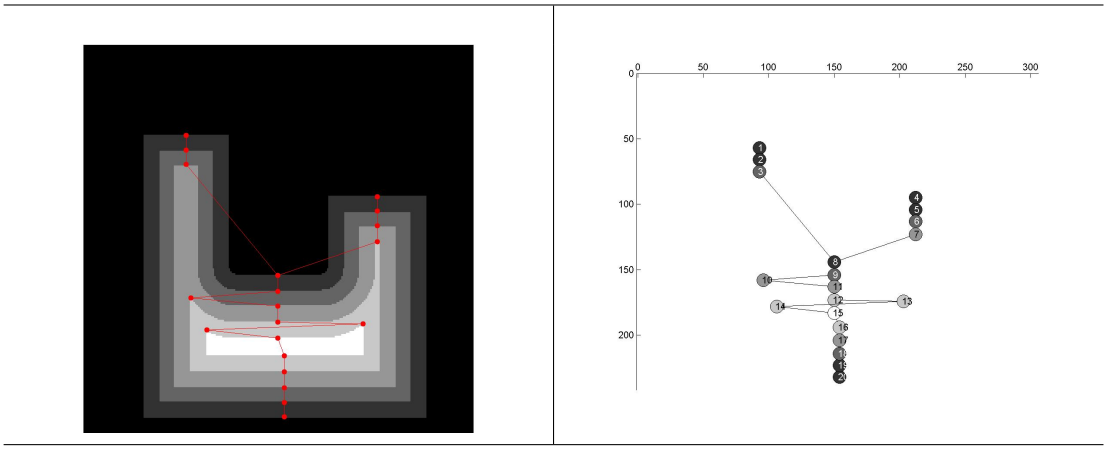


Figure 15: Example image with Reeb graph built on it.

Figure 15 shows examples for both types of Reeb graph built this way. For more results on the test dataset or the root dataset see Section 12 and 13.

10 Problems on the rootset

Two problems were encountered when analyzing the root dataset:

1. Additional critical points
2. Two or more critical points at same height

10.1 Additional critical points

In the final foreground/background image borders of image regions might be frayed due to the segmentation process. Figure 16 shows an example of such a frayed border. For these frayed borders additional critical points that describe no actual split or merge of the root are computed. These critical points alter the Reeb graph and complicate a comparison or matching of these graphs.

10.2 Two or more critical points at same height

Due to the resolution of the images, the discretization of the root to be analyzed and the further distortion during the segmentation process it is possible that several critical points at different horizontal positions in the image are at the same vertical position (same height) in the image (see Figure 17 for an example). Thus the third condition for Morse functions (see Section 6) is not met and no Reeb graph can be built. This problem arises for 32 out

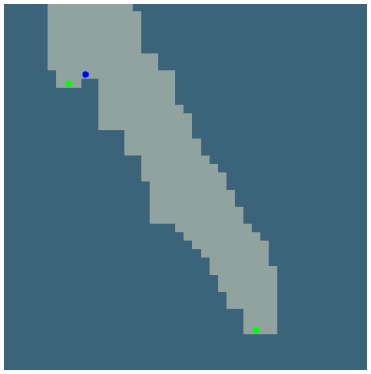


Figure 16: Detail dataset “older age” - root09, day 20: Example of frayed border and additional computed critical points

of the analyzed 34 root images in the root dataset.

10.3 Problem handling

Especially the problem of two or even more critical points at the same height, described in section 10.2, needs to be handled, as it occurs with 94% of all images in the root dataset. This problem is overcome by implementing the following solution:
Due to the discrete pixel-space, the coordinates (x, y) of a pixel $p = (x, y)$ are integers. Critical points at the same height (same y-coordinate) are shifted. The height of such critical points is changed by an added factor f, $0 \leq f < 1$. A critical point $p = (x, y)$ is shifted to $p' = (x, y + f)$, f is computed using the following formula: $f = \frac{1}{w} * (x - 1)$, with $w =$ width of the image. The y-coordinate is thereby changed from an integer to a decimal number. Critical points at the same height are moved downwards in a left-to-right order, thus for two critical points $p_1 = (x_1, y)$ and $p_2 = (x_2, y)$ with $x_1 < x_2$, it is valid that, after shifting the points to $p'_1 = (x_1, y_1)$ and $p'_2 = (x_2, y_2)$, $y_1 < y_2$ holds. The actual order of heights is preserved by this correction procedure as only critical points that were primarily at the same height are changed. All critical points are at different heights, although when rounding down the y-coordinate of the critical points to an integer, they stay in the actual pixel line. A Reeb graph can therefore be built.
It is important to shift the heights in a fixed approach. A random decision choosing one of two critical points at the same height when building the

Figure 17: Detail dataset “older age” - root09, day 20: Example of several critical points at same height

Reeb graph cannot be used, as the results may vary with repeated tests. Reeb graphs built on such random decisions are not unique and therefore useless for i.e. comparison of two images.

Considering the problem of additional critical points described in section 10.1 a solution is not implemented in this approach as it does not influence the possibility of building a Reeb graph. This problem is based on the resolution of the root images, the fine structures in the images and the segmentation needed as a pre-processing step when computing the critical points. As these additional critical points occur with frayed borders of segmented regions, smoothing these borders in a post-processing step could reduce these additional critical points. However, thin structures that actually describe the root, could be lost as well.
Persistence diagrams might pose another possible solution as such frayed borders (and the critical points caused by them) are comparable to noise in images.

11 Categorization of nodes

The five different types of critical points computed result in four different types of nodes in a Reeb graph: minimum nodes, maximum nodes, saddle nodes, regular nodes. Figure 5 illustrates these different types of critical points together with the corresponding nodes in the Reeb graph.

When additionally considering the gray-values (or color-values) of regions, even more types of nodes can be distinguished. For each critical point the color information is stored together with this point. Considering gray-values, these values can be understood as dimension of elevation. The gray-values range between black (value 0) and white (value 255 in an 8-bit image and 65535 in a 16-bit image). White, as the maximum value, is considered to be the highest elevation.

Taking this into account, we can define 12 categories to classify the critical points:

1. critical point of type maximum with:
   (a) maximum elevation
   (b) minimum elevation
   (c) intermediate elevation
2. critical point of type saddle with:
   (a) maximum elevation
   (b) minimum elevation
   (c) intermediate elevation
3. critical point of type minimum with:
   (a) maximum elevation
   (b) minimum elevation
   (c) intermediate elevation
4. regular node with:
   (a) maximum elevation

(b) minimum elevation

(c) intermediate elevation

This classification is based on the type of the critical point/node computed and the color/gray-value of each node compared to the colors/gray-values of all nodes.
As output a list of all nodes holding the classification of each node is generated.

12 Results on the testset

Table 1 shows the occurrences of the 12 categories described in Section 11 in the test dataset.

For each image in the test-dataset (Appendix A) four images are shown:

- The input image
- Critical points marked in the image (as described in Section 8). The critical points are marked in five different colors: red for critical points of type birth, green for type death, blue for type split and yellow for type merge, regular nodes due to changes in the color are marked in pink.
- Reeb graph drawn on the image (as described in Section 8)
- Reeb graph with nodes colored according to the image regions(as described in Section 11)

All critical points and connections in the Reeb graphs were checked and verified manually.

types of nodes	types of elevation		
	maximum	minimum	intermediate
maximum	16	9	1
minimum	15	8	3
regular	15	23	54
saddle	15	3	2

Table 1: Occurrences of the 12 categories described in Section 11 in the test dataset

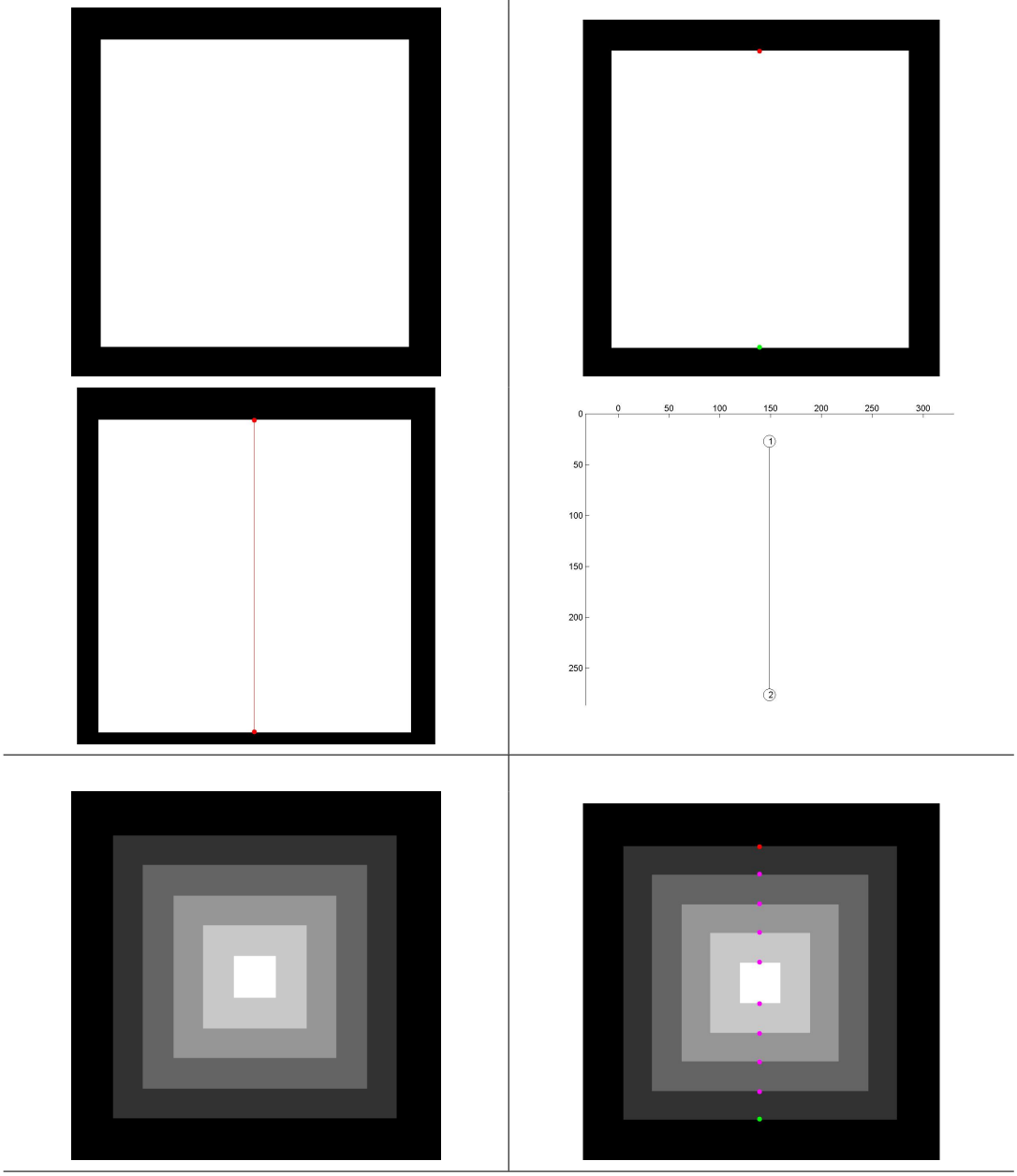


Figure 18: Results on the synthetical test dataset - image 1, 2

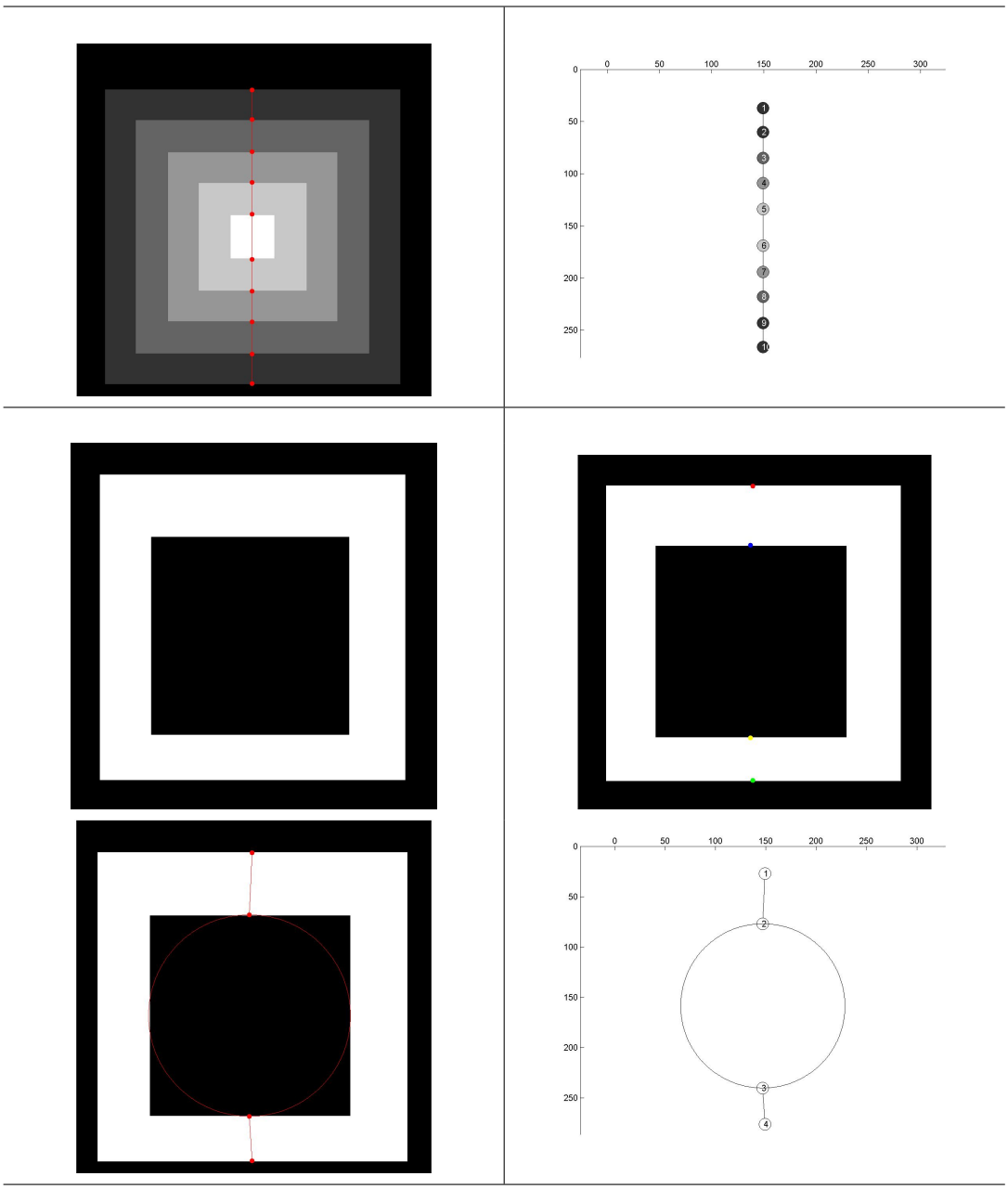


Figure 19: Results on the synthetical test dataset - image 2, 3

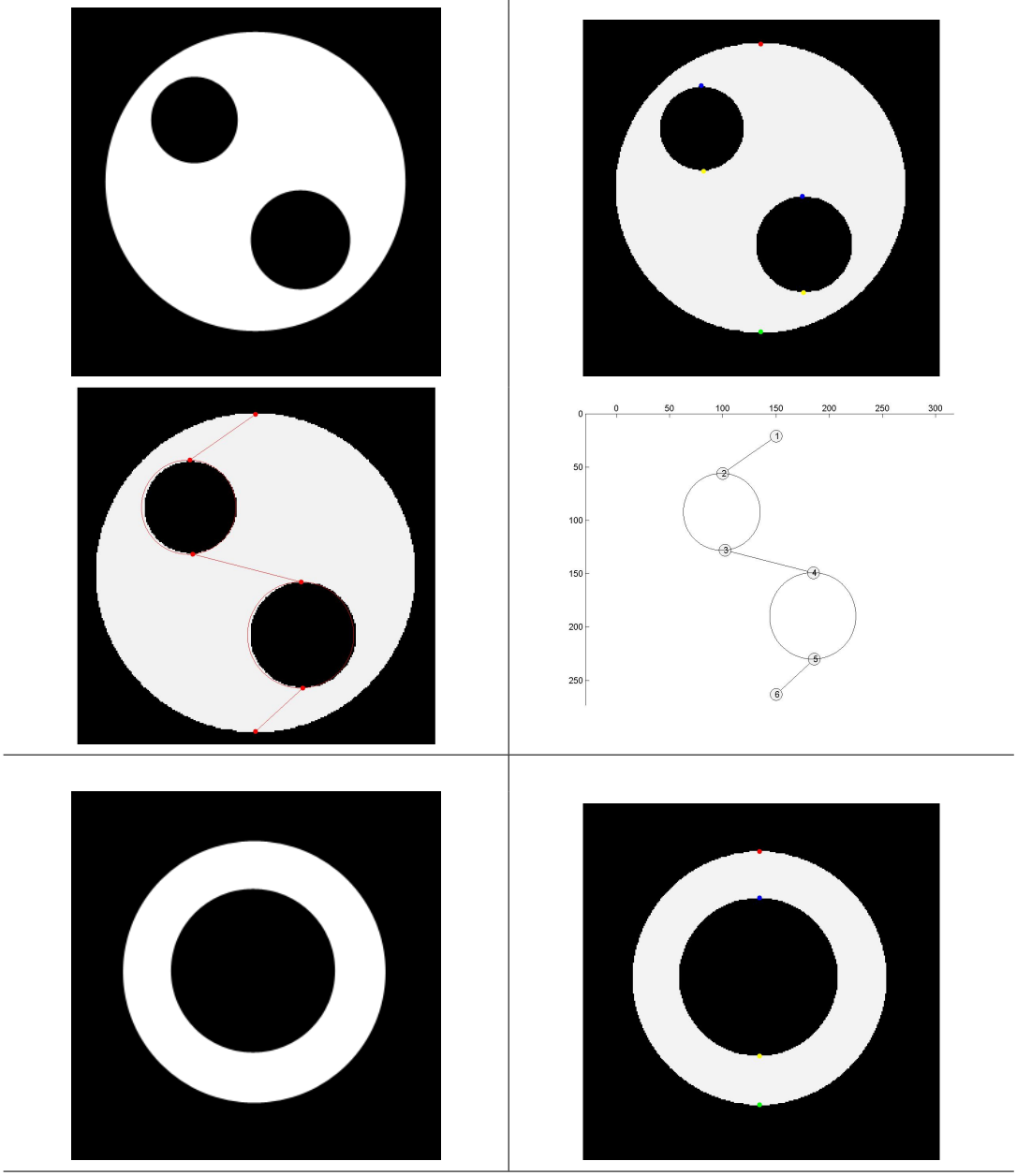


Figure 20: Results on the synthetical test dataset - image 4, 5

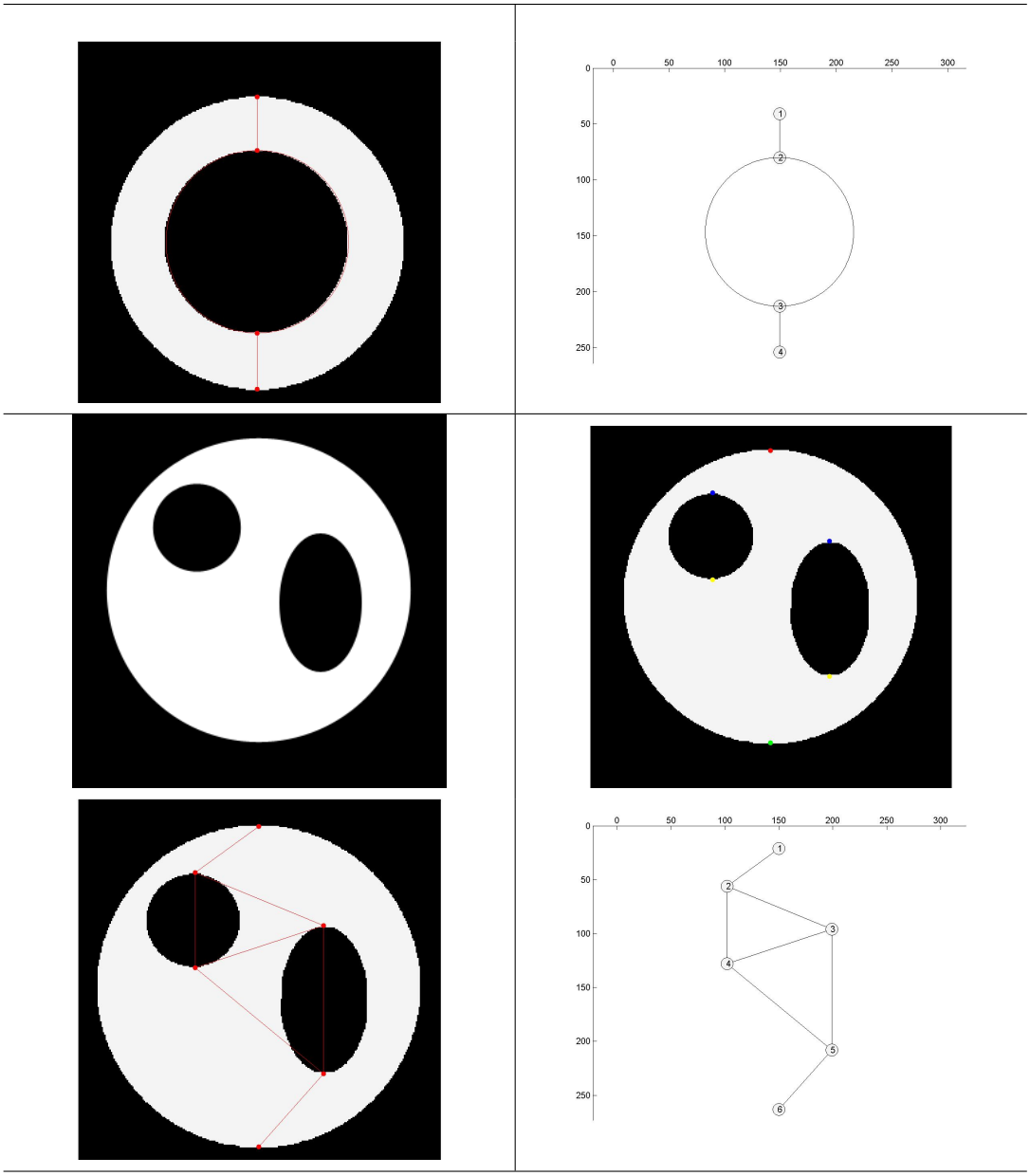


Figure 21: Results on the synthetical test dataset - image 5,6

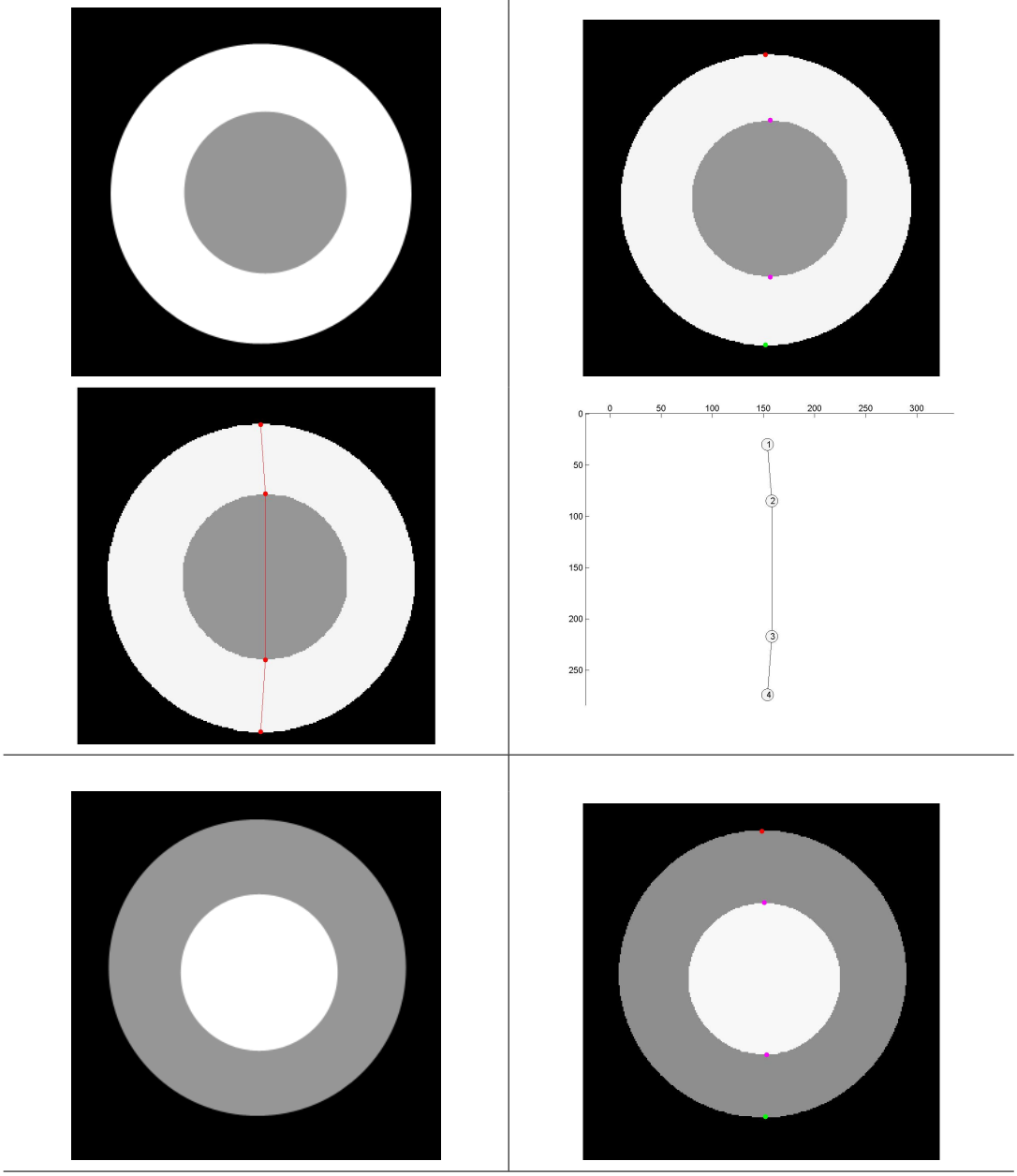


Figure 22: Results on the synthetical test dataset - image 7, 8

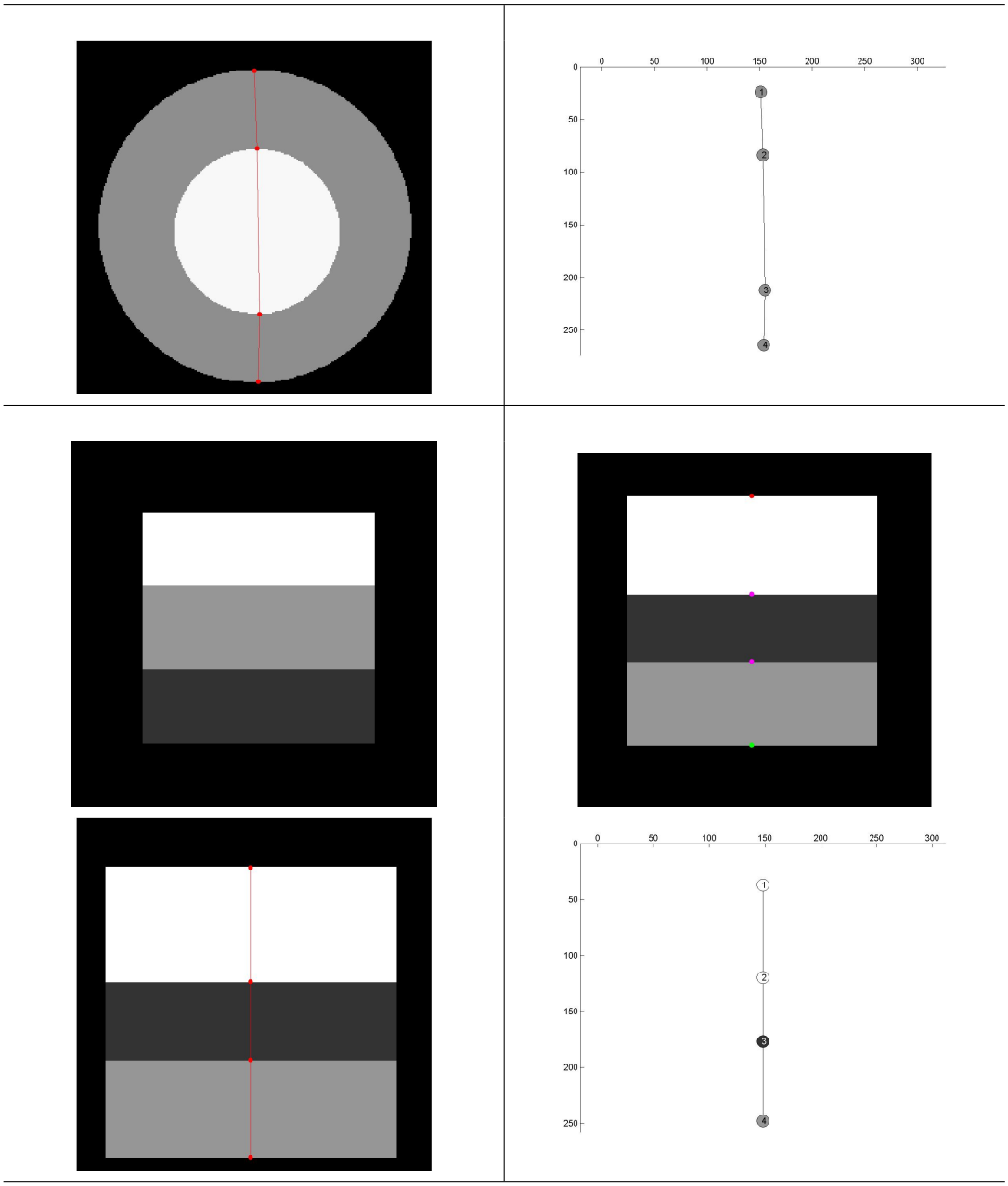


Figure 23: Results on the synthetical test dataset - image 8, 9

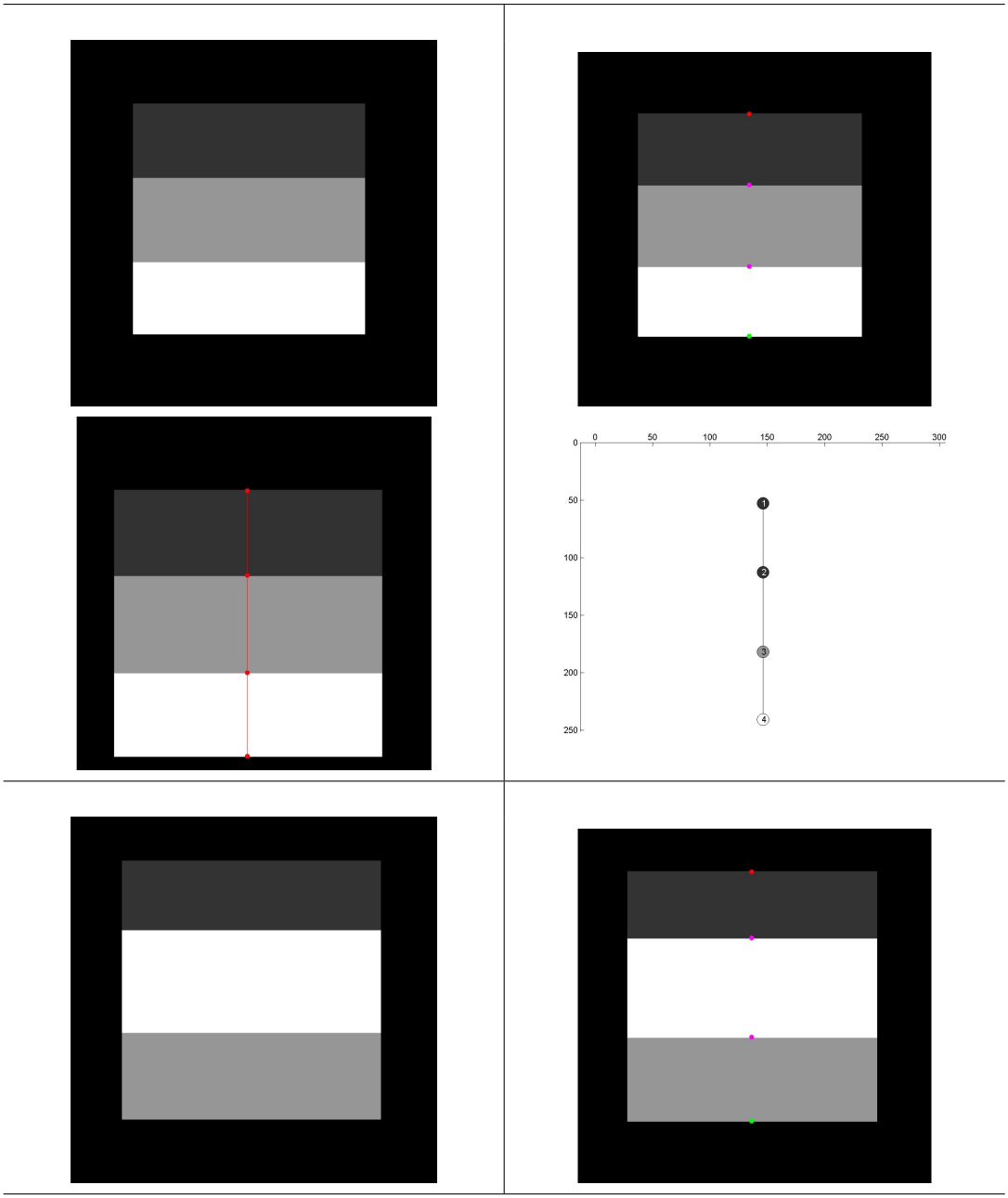


Figure 24: Results on the synthetical test dataset - image 10 ,11

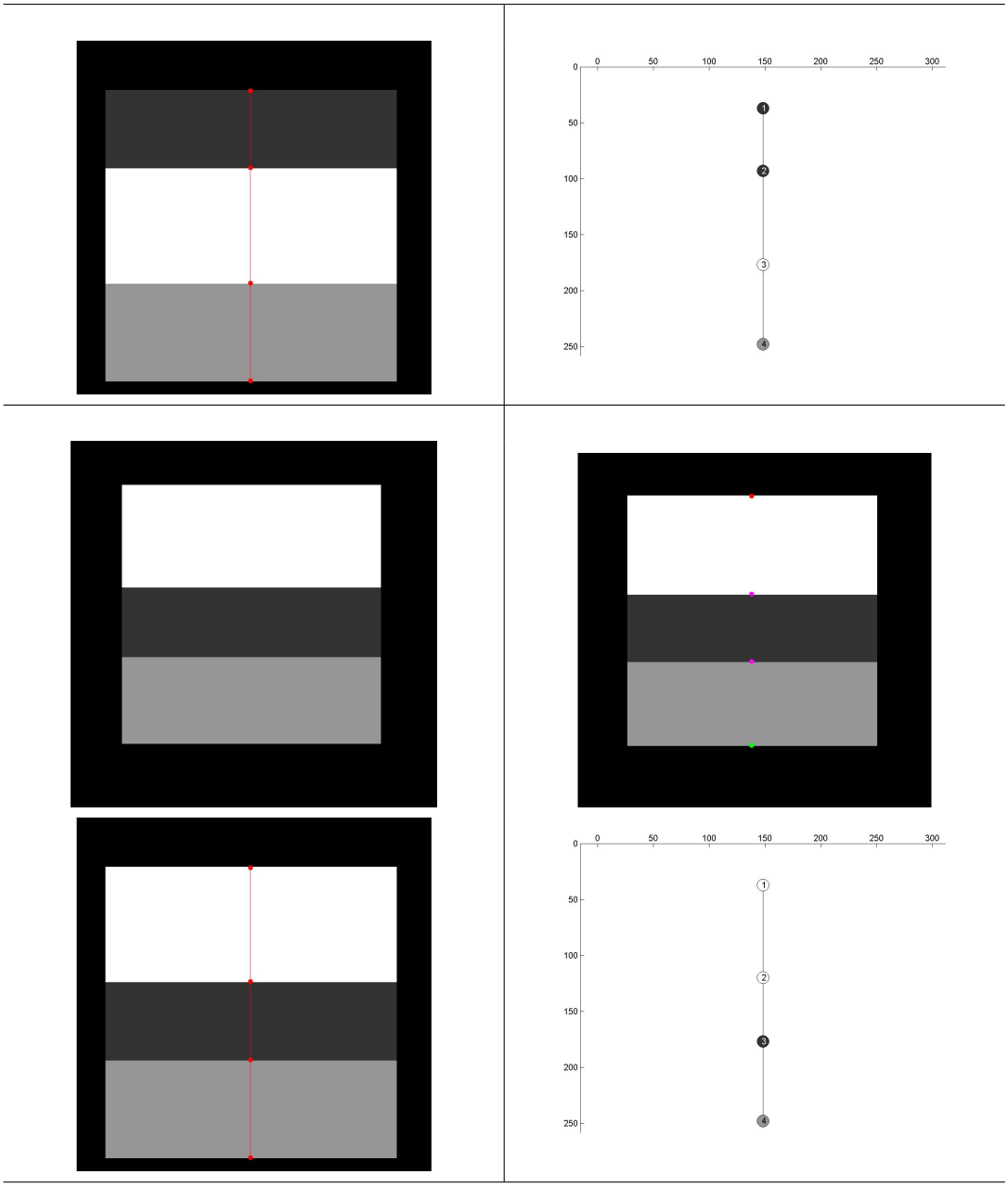


Figure 25: Results on the synthetical test dataset - image 11, 12

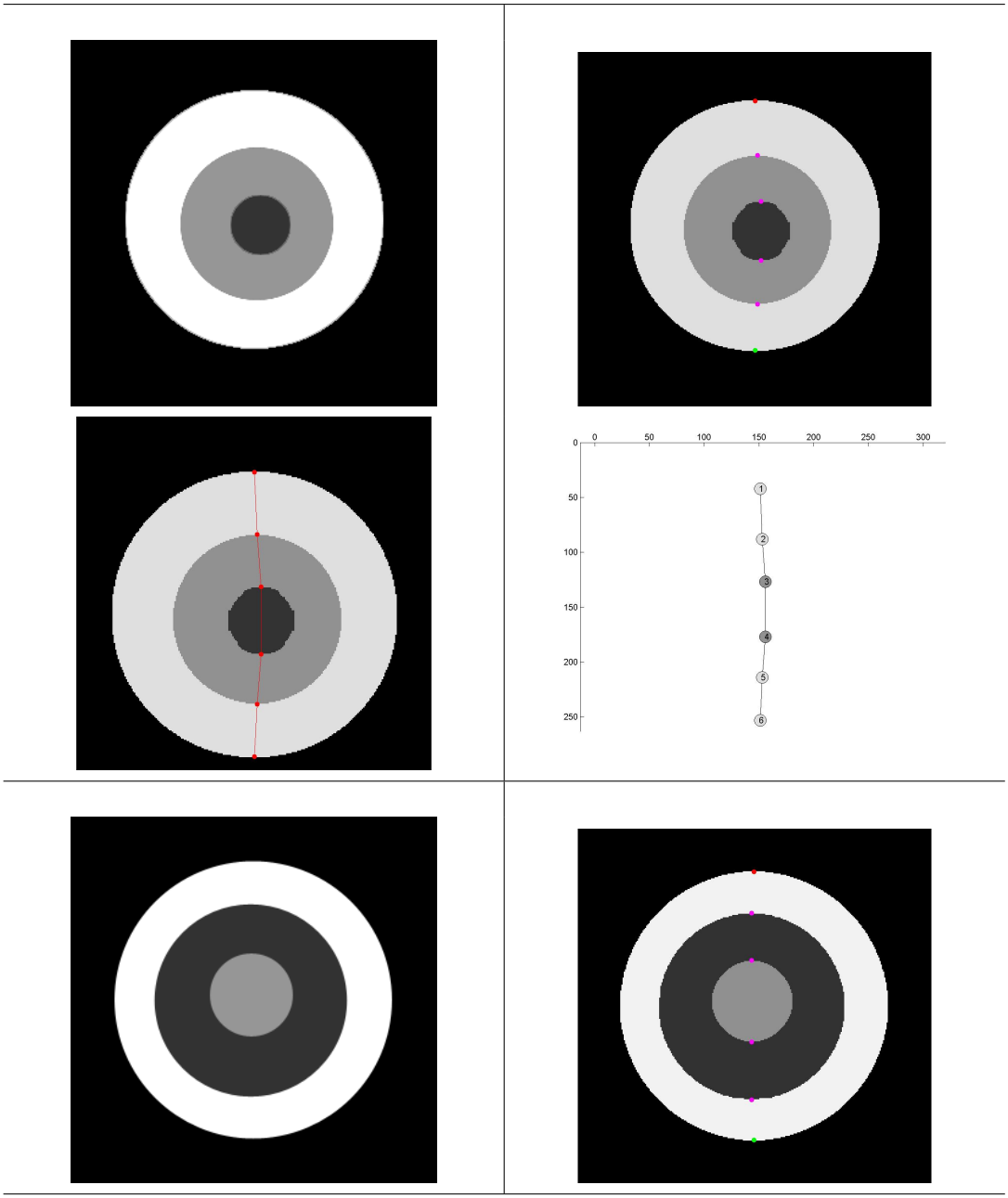


Figure 26: Results on the synthetical test dataset - image 13, 14

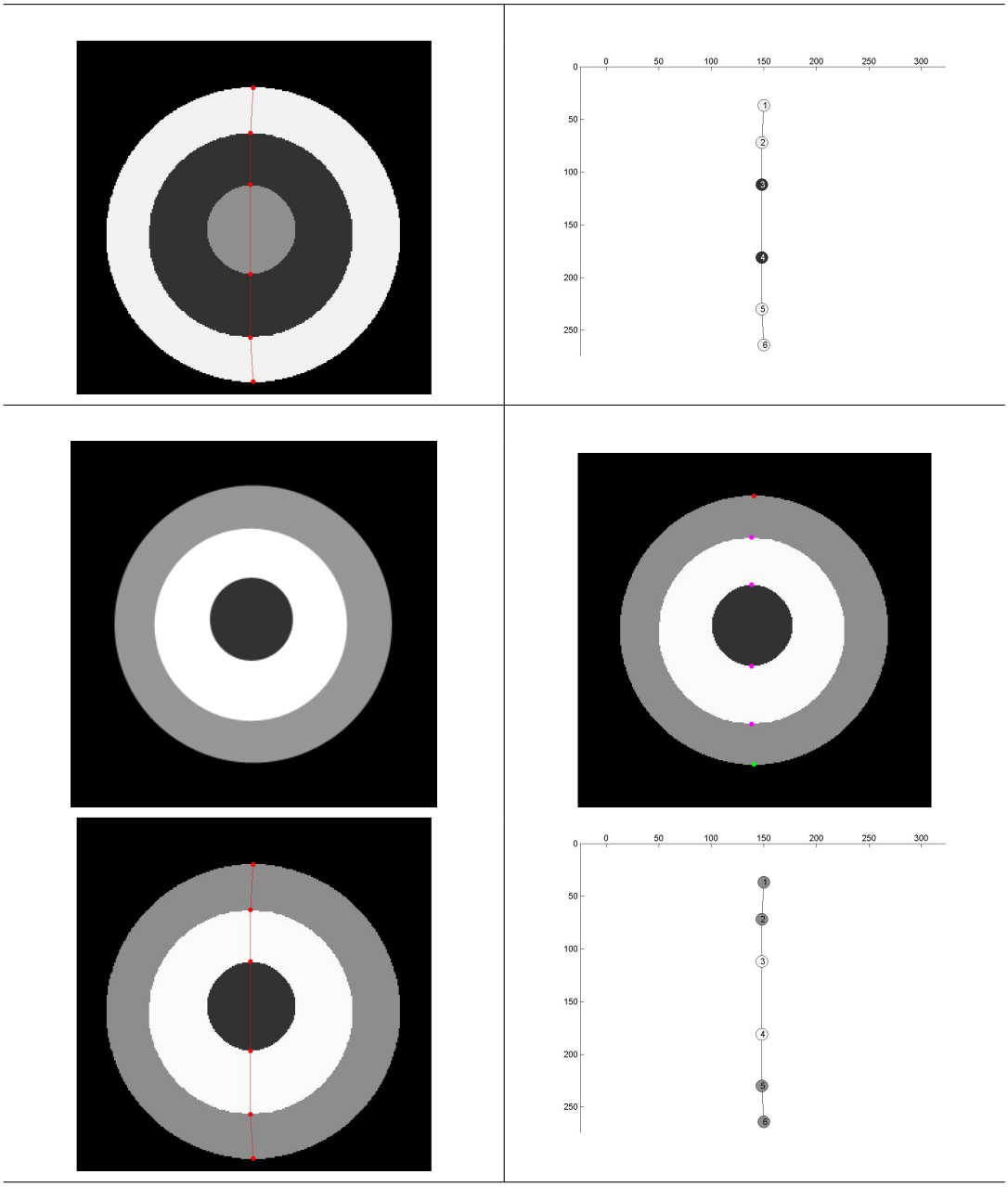


Figure 27: Results on the synthetical test dataset - image 14, 15

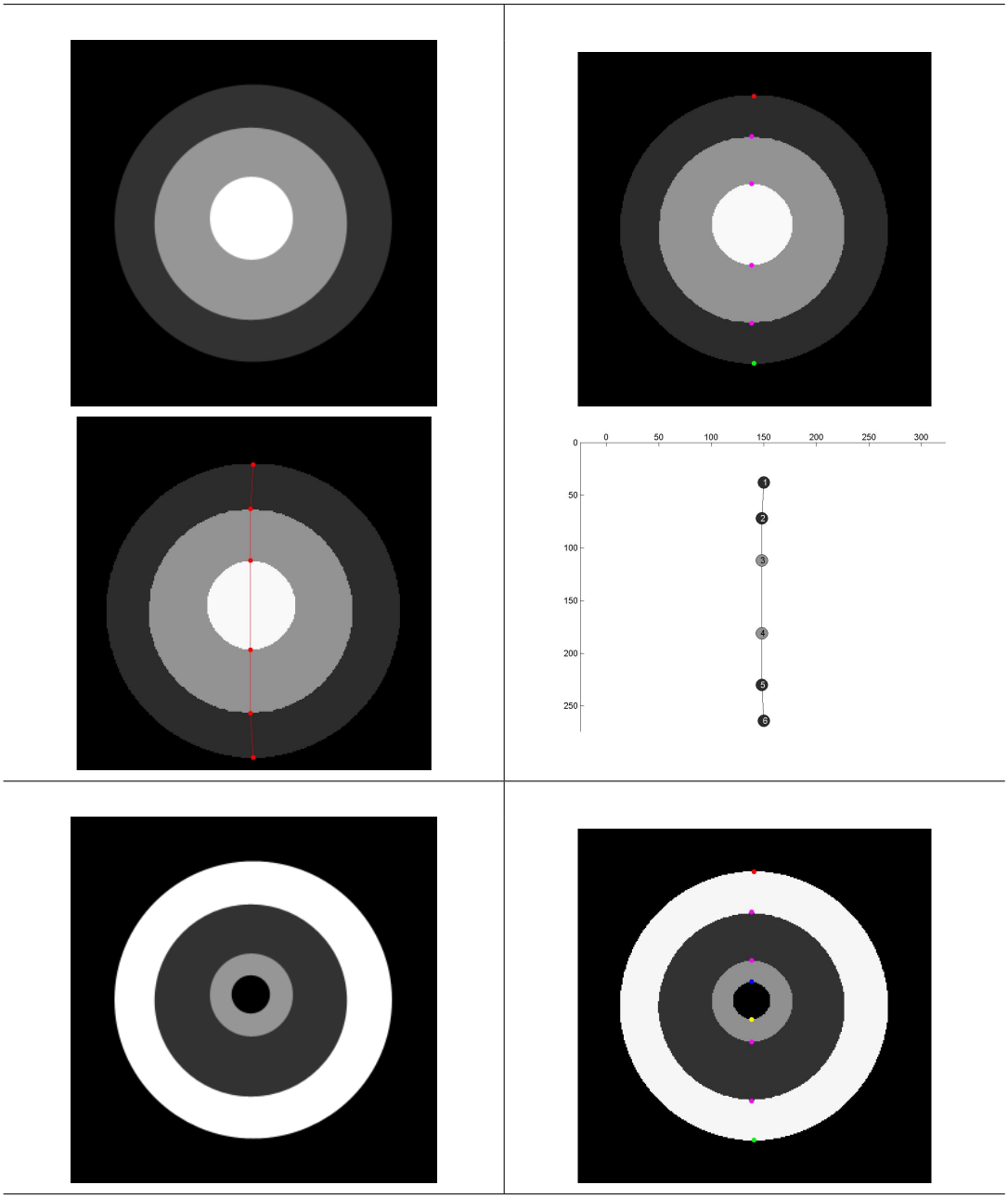


Figure 28: Results on the synthetical test dataset - image 16, 17

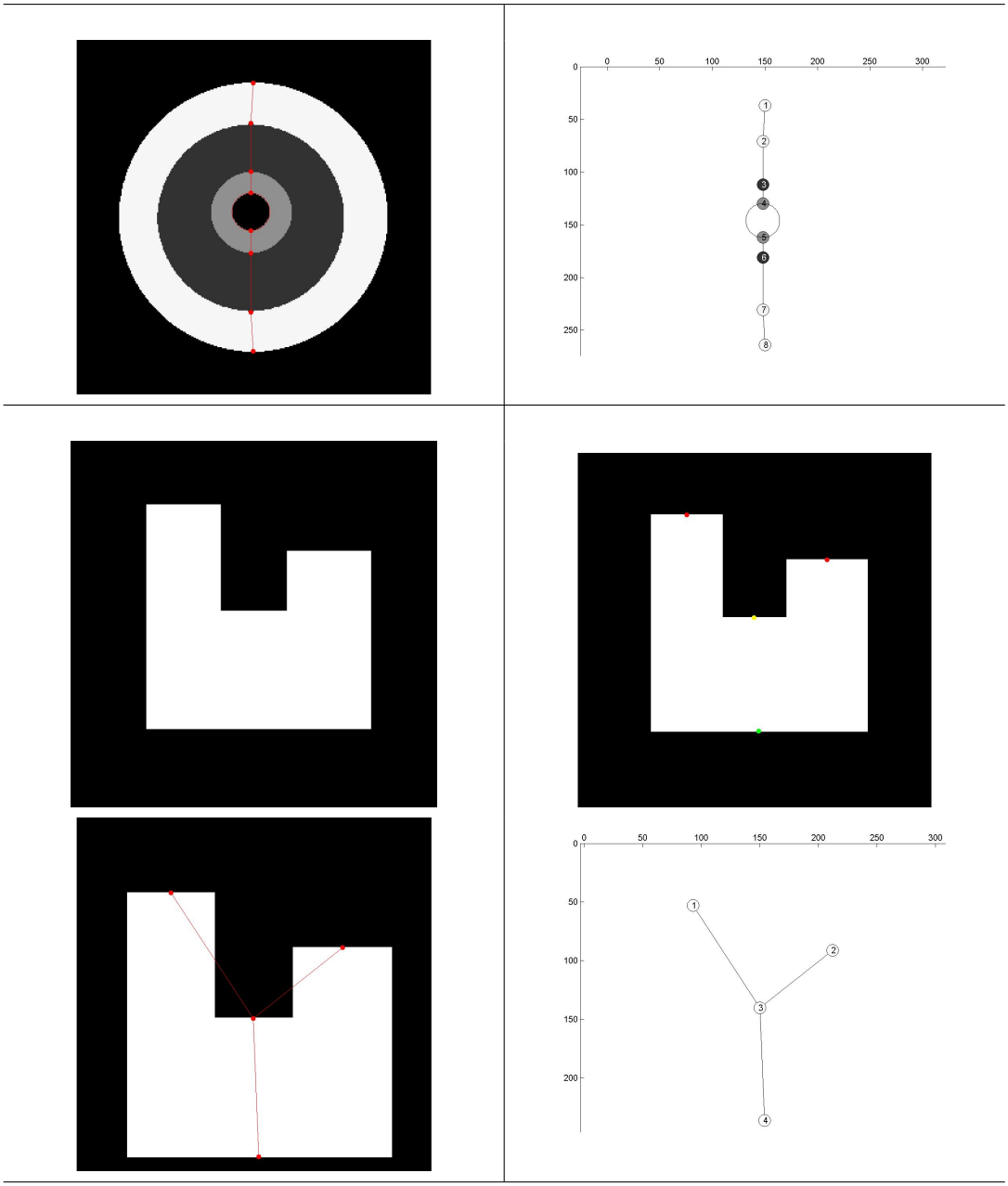


Figure 29: Results on the synthetical test dataset - image 17, 18

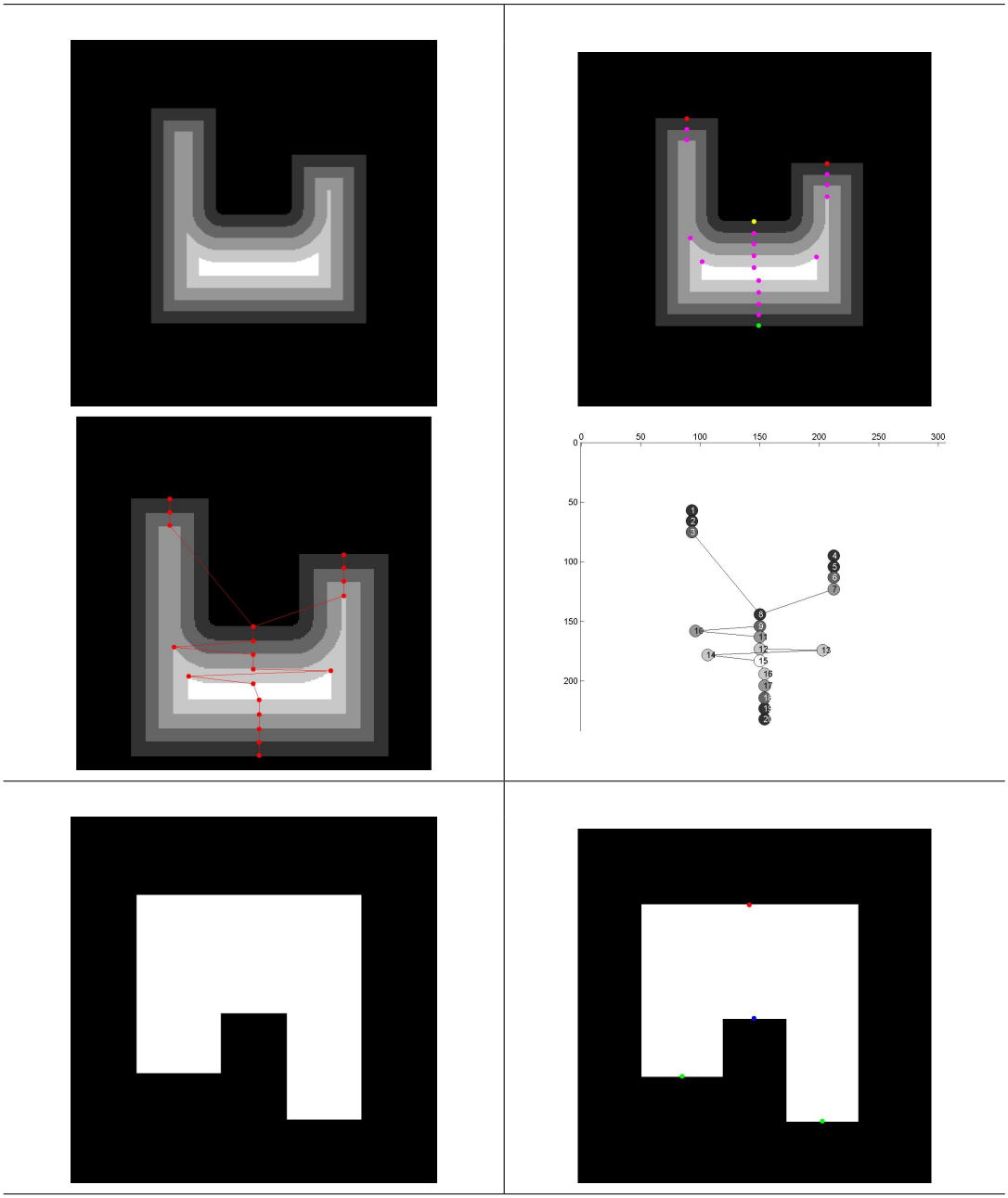


Figure 30: Results on the synthetical test dataset - image 19, 20

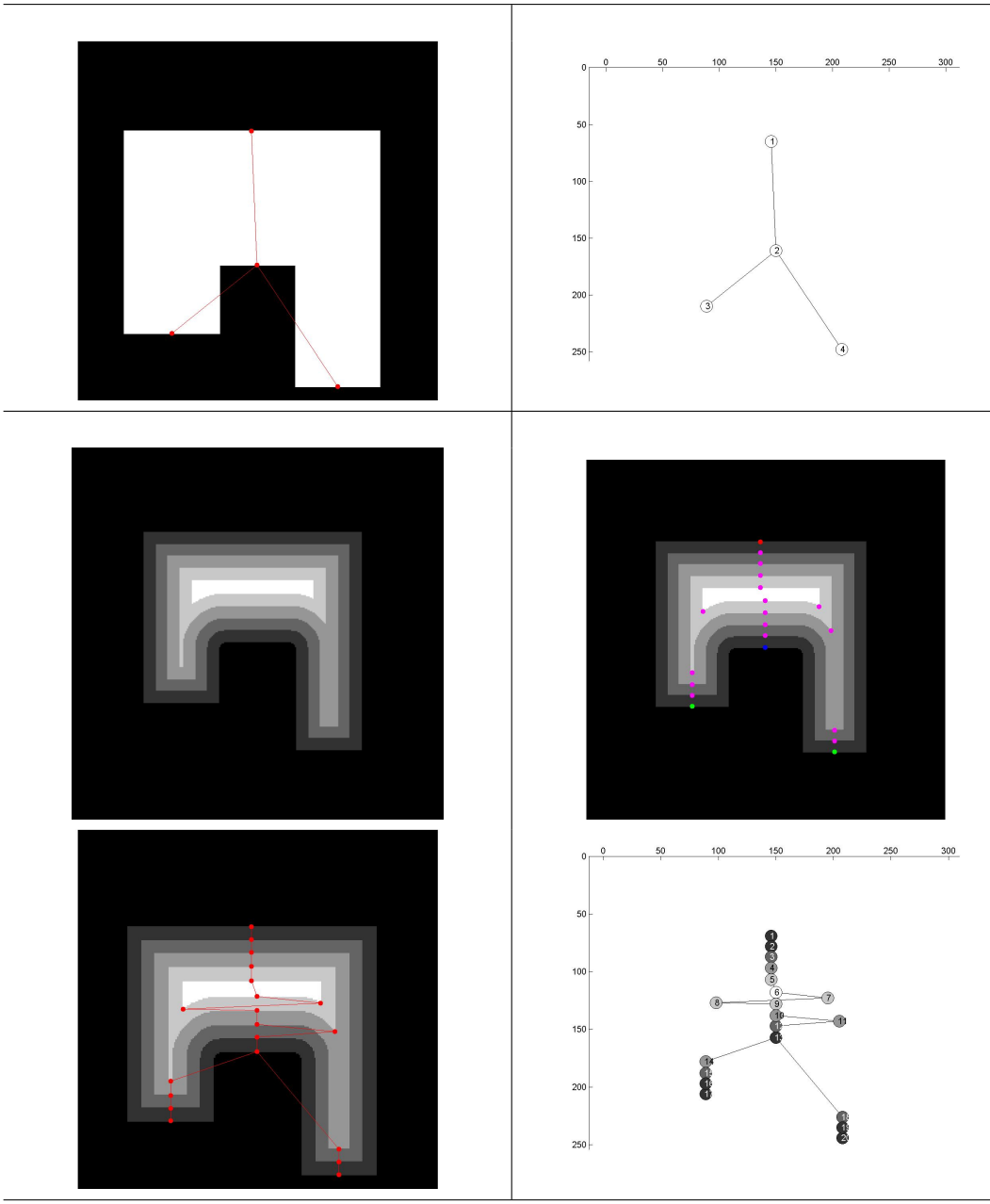


Figure 31: Results on the synthetical test dataset - image 20, 21

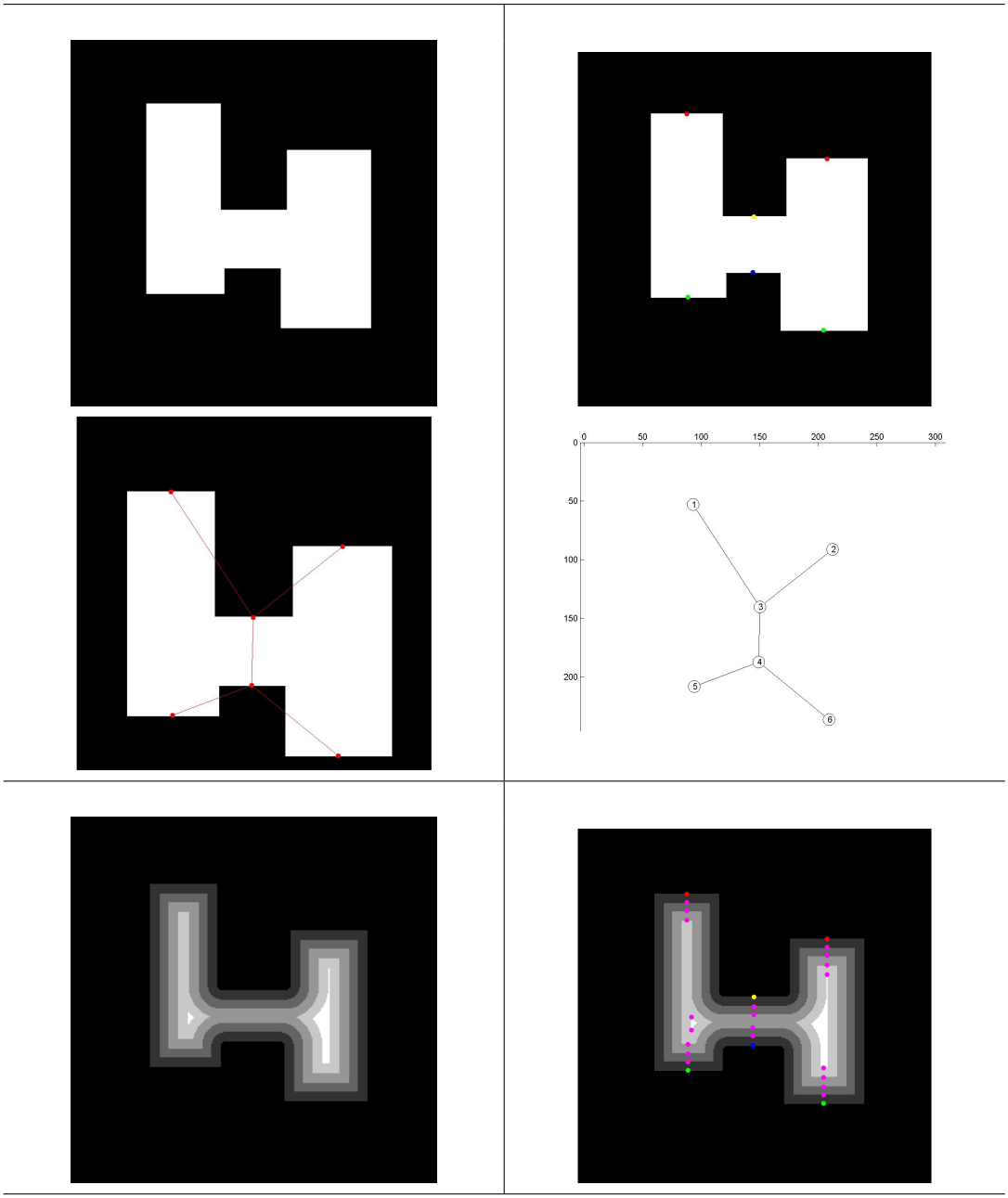


Figure 32: Results on the synthetical test dataset - image 22, 23

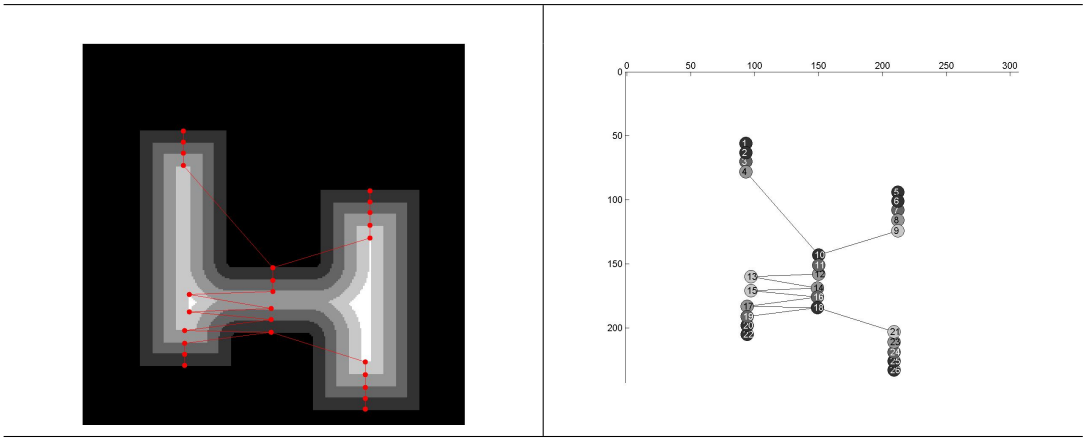


Figure 33: Results on the synthetical test dataset - image 23

13 Results on the rootset

For each image in the root-dataset (Appendix B) three different results were computed:

- The segmented input image
- Critical points marked in the image (as described in Section 8). The critical points are marked in five different colors: red for critical points of type birth, green for type death, blue for type split and yellow for type merge, regular nodes due to changes in the color are marked in pink.
- Reeb graph drawn on the image (as described in Section 8)
- Reeb graph with nodes colored according to the image regions(as described in Section 11)

All critical points and connections in the Reeb graphs were checked and verified manually.

Table 2 shows the possible connections between different kinds of critical points and according nodes in the Reeb graph. These connections are given for two nodes $A = (x_1, y_1)$ and $B = (x_2, y_2)$ with $y_1 < y_2$, as a height function was used as basic function to compute the critical points. In addition table 3 describes the numbers of connections (inbound and outbound) each of these

Possible connections between two nodes					
type node A	type of node B				
	maximum	saddle (split)	saddle (merge)	minimum	regular
maximum	⊘	✓	✓	✓	✓
saddle (split)	⊘	✓	✓	✓	✓
saddle (merge)	⊘	✓	✓	✓	✓
minimum	⊘	⊘	⊘	⊘	⊘
regular	⊘	✓	✓	✓	✓

Table 2: Connections from node A to node B. ✓stands for a possible connection, for combinations marked with ⊘ no such connection is possible

nodes has in the graph. These tables are intended to help in understanding the given result images, as some markers for the critical points overlap due to their location close to each other and due to their size. All result images are also saved as images in the Matlab fig format. This allows to zoom in on the images in order to solve the problem of overlapping markers.

Numbers of connections for nodes		
types of nodes	inbound	outbound
maximum	0	1
saddle (split)	1	2
saddle (merge)	2	1
minimum	1	0
regular	1	1

Table 3: Numbers of inbound and outbound connections for different types of nodes

The result images for the root dataset are organized according to the computed results. For all images in the dataset first the computed critical points are shown in section 13.1, section 13.2 shows the Reeb graphs based on these critical points as an overlay on the images. Reeb graphs with labeled nodes are given in section 13.3. A discussion of the results is given in section 14.

13.1 Results: critical points

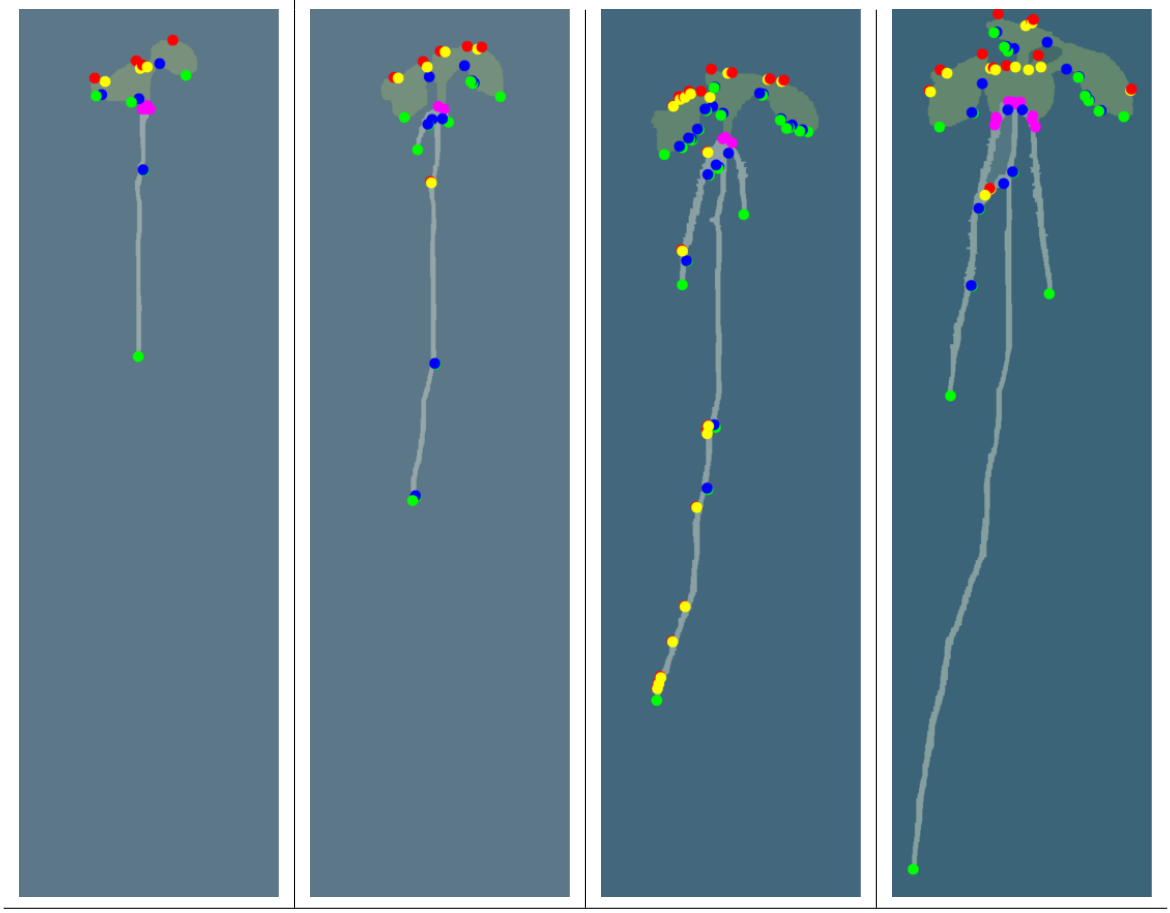


Figure 34: Results: critical points on root04 day 8, 12, 16 and 20

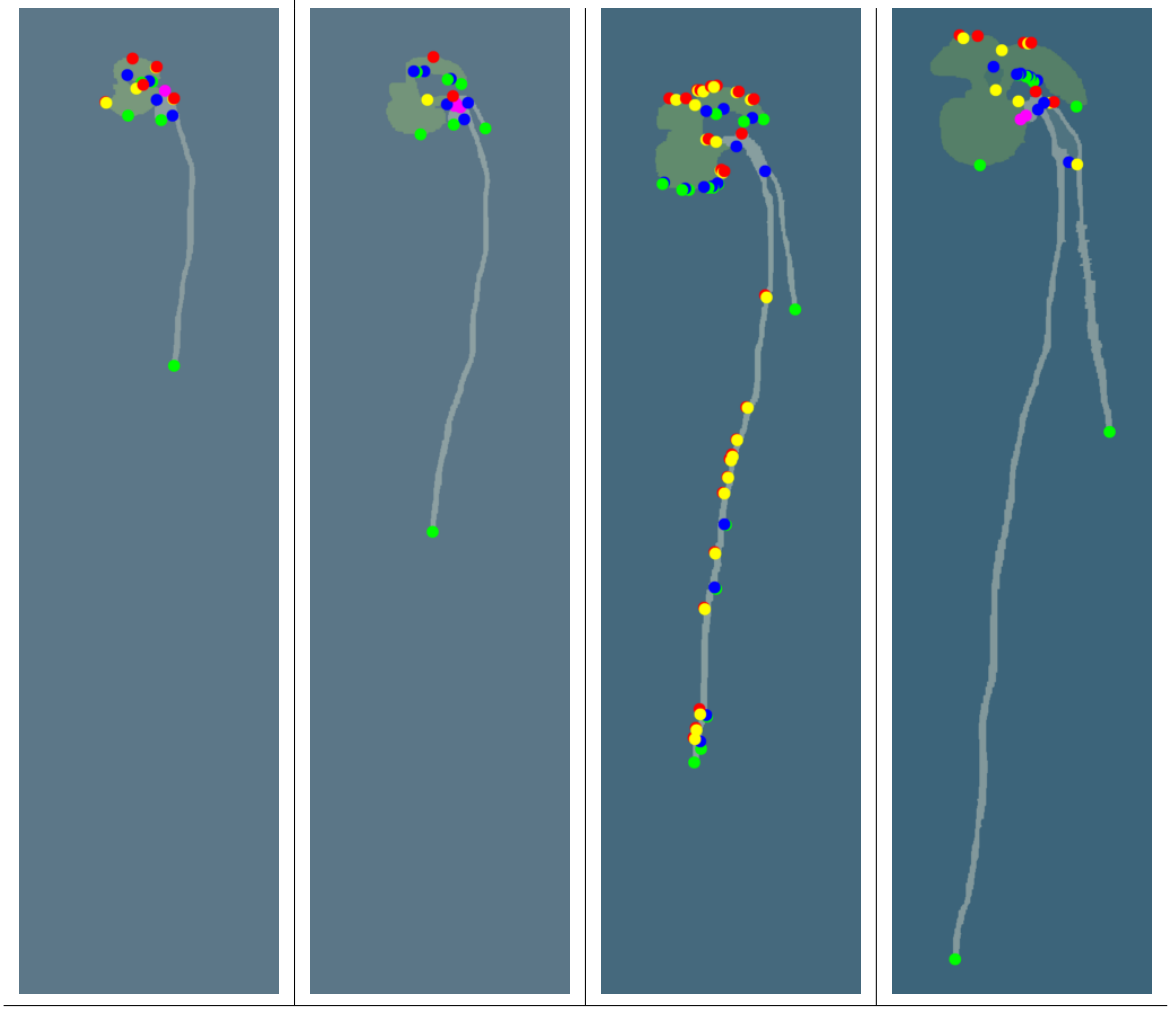


Figure 35: Results: critical points on root05 day 8, 12, 16 and 20

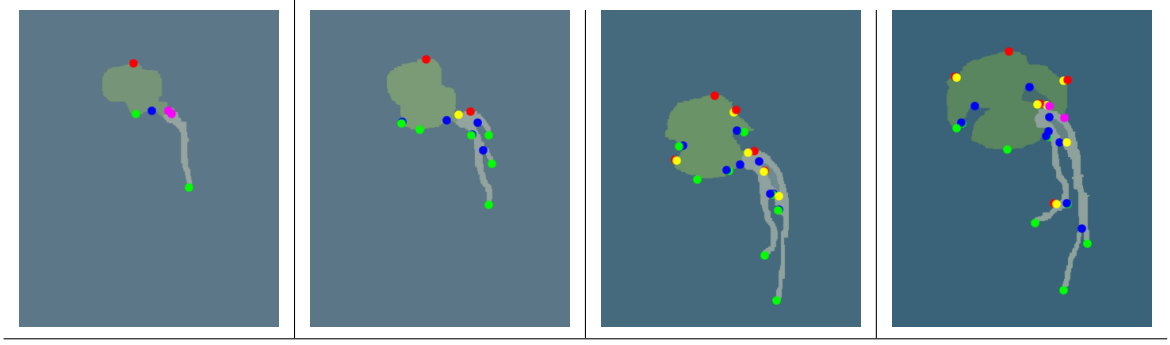


Figure 36: Results: critical points on root07 day 8, 12, 16 and 20

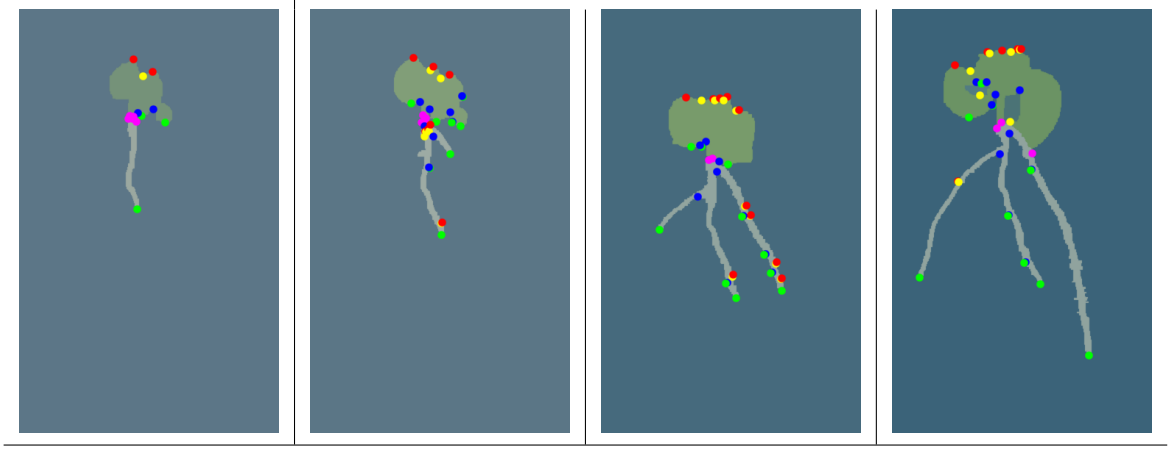


Figure 37: Results: critical points on root09 day 8, 12, 16 and 20

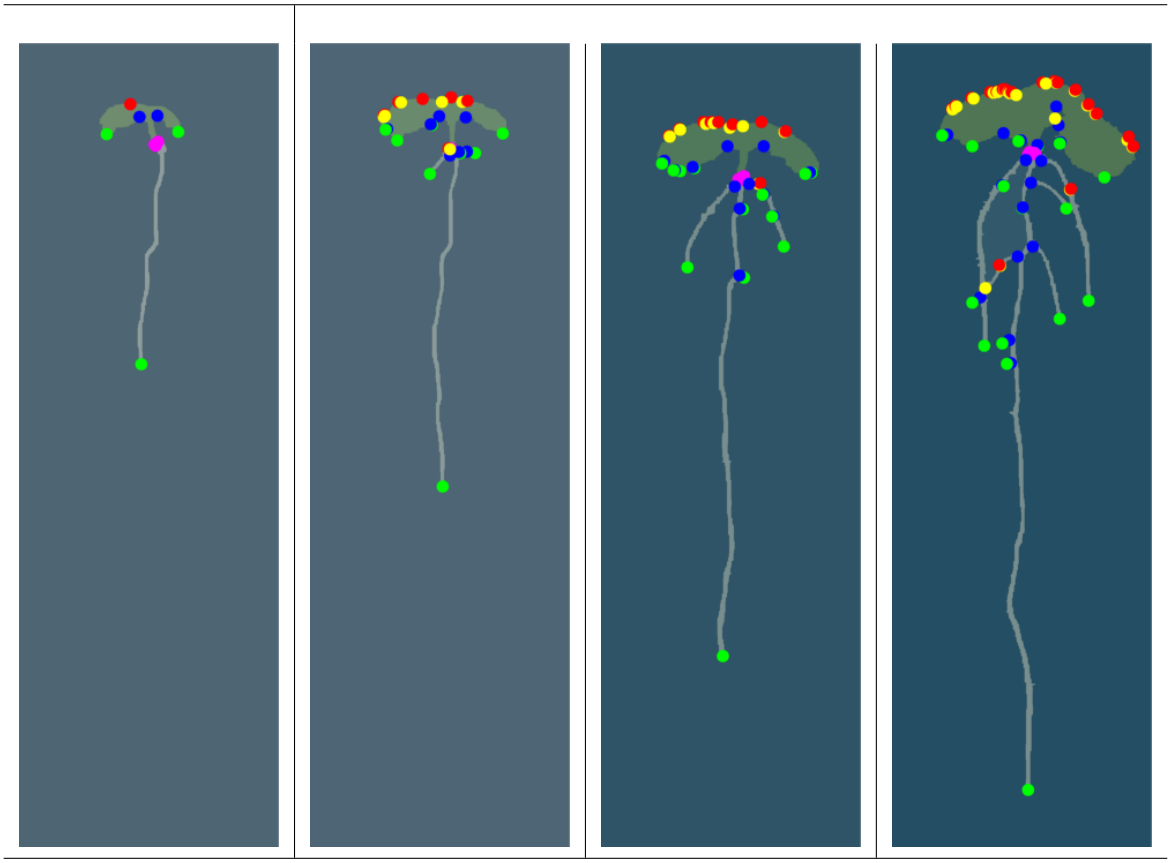


Figure 38: Results: critical points on root12 day 8, 12, 16 and 20

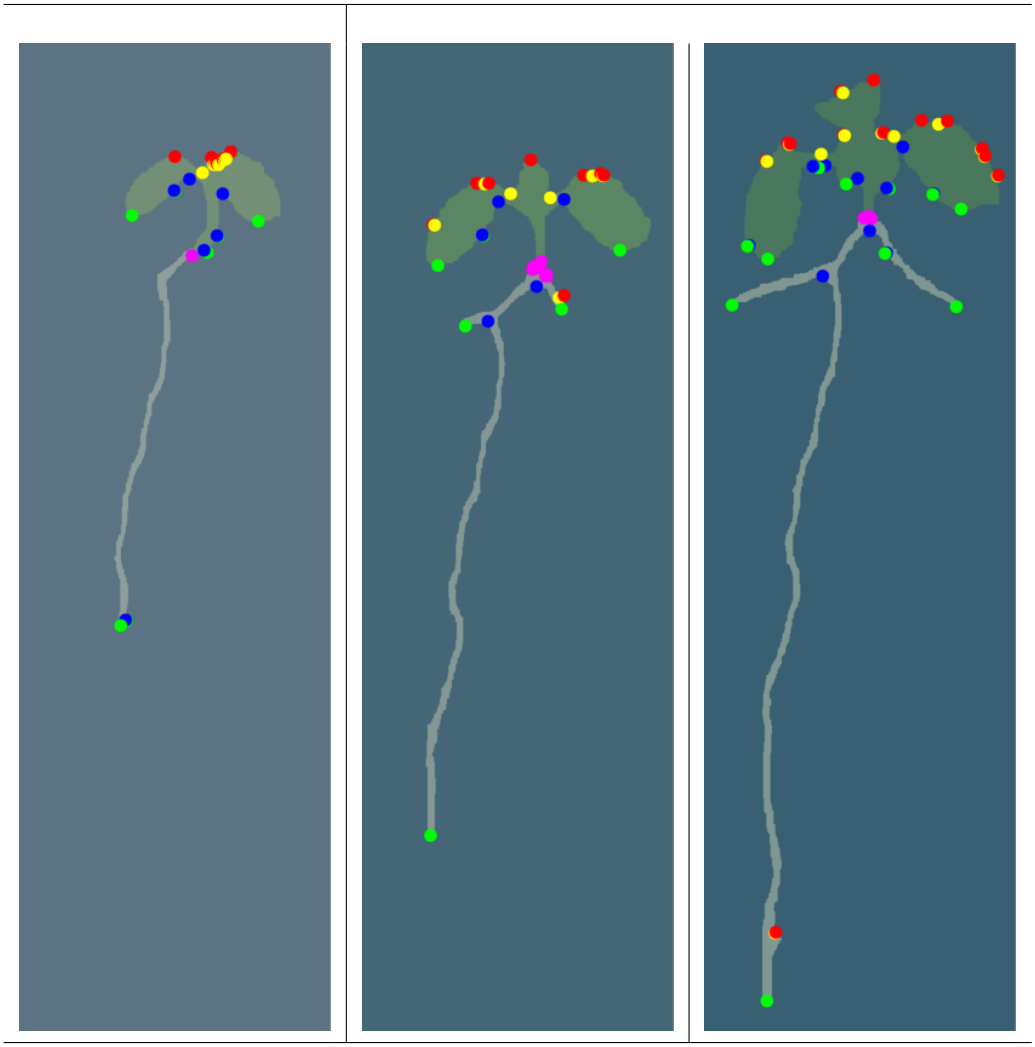


Figure 39: Results: critical points on root17 day 12, 16 and 20

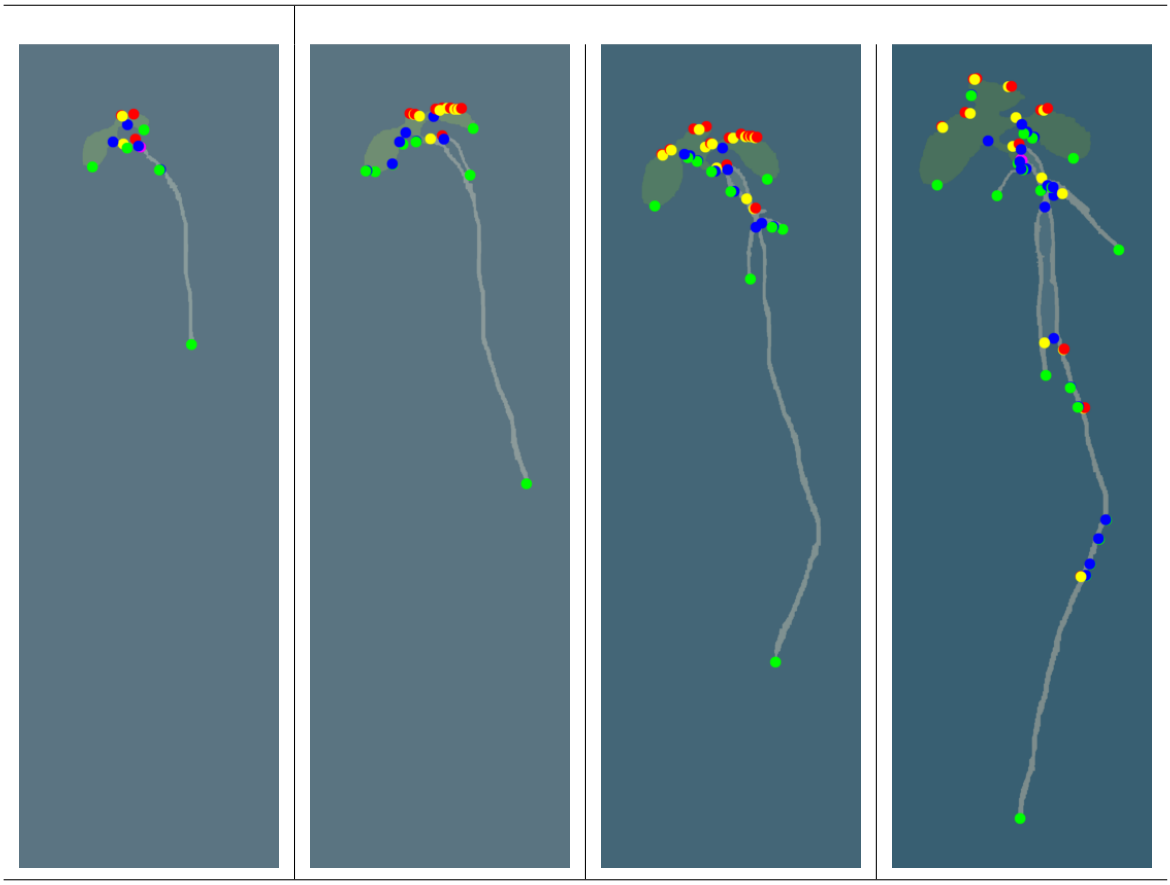


Figure 40: Results: critical points on root19 day 8, 12, 16 and 20

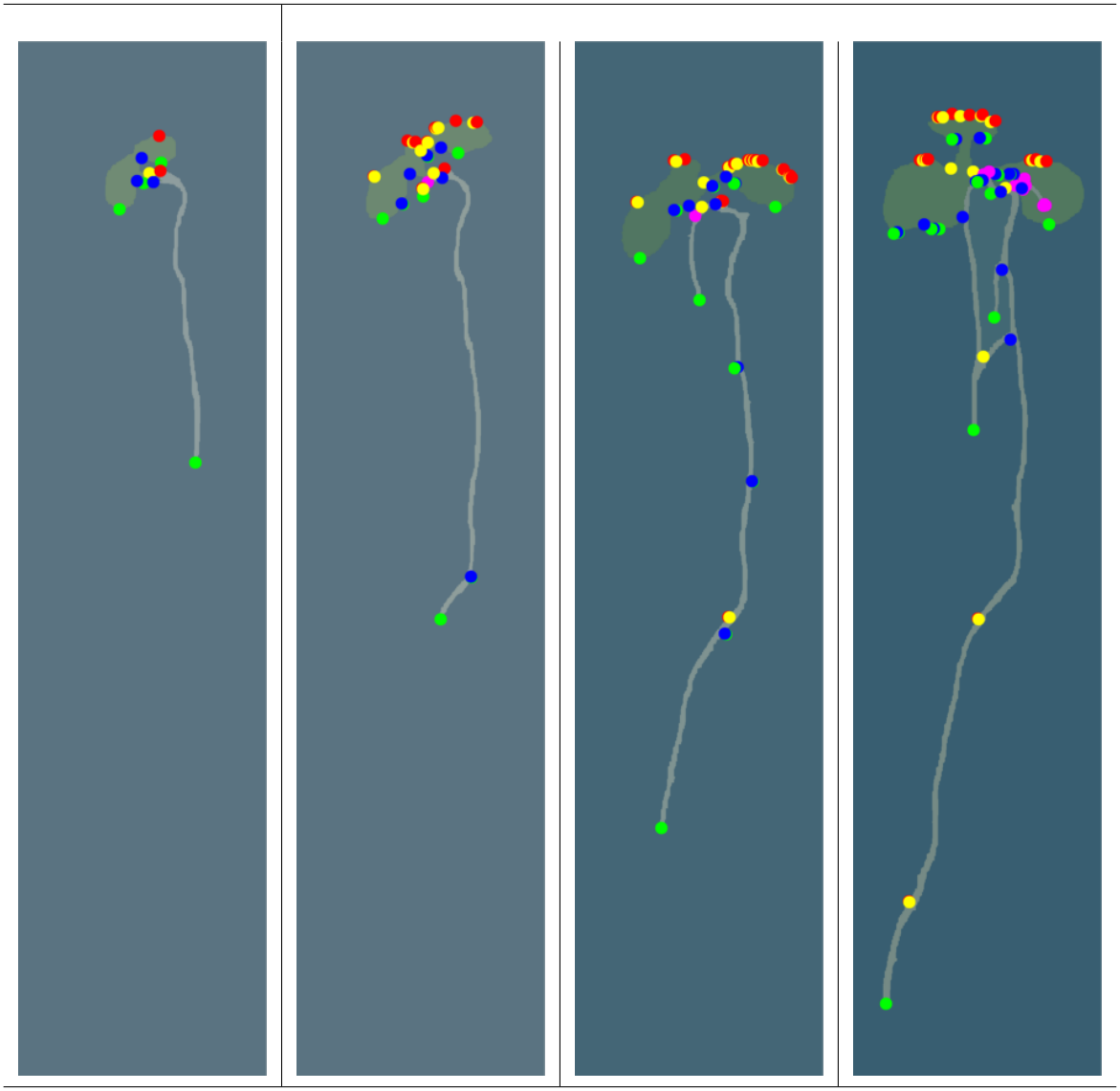


Figure 41: Results: critical points on root20 day 8, 12, 16 and 20

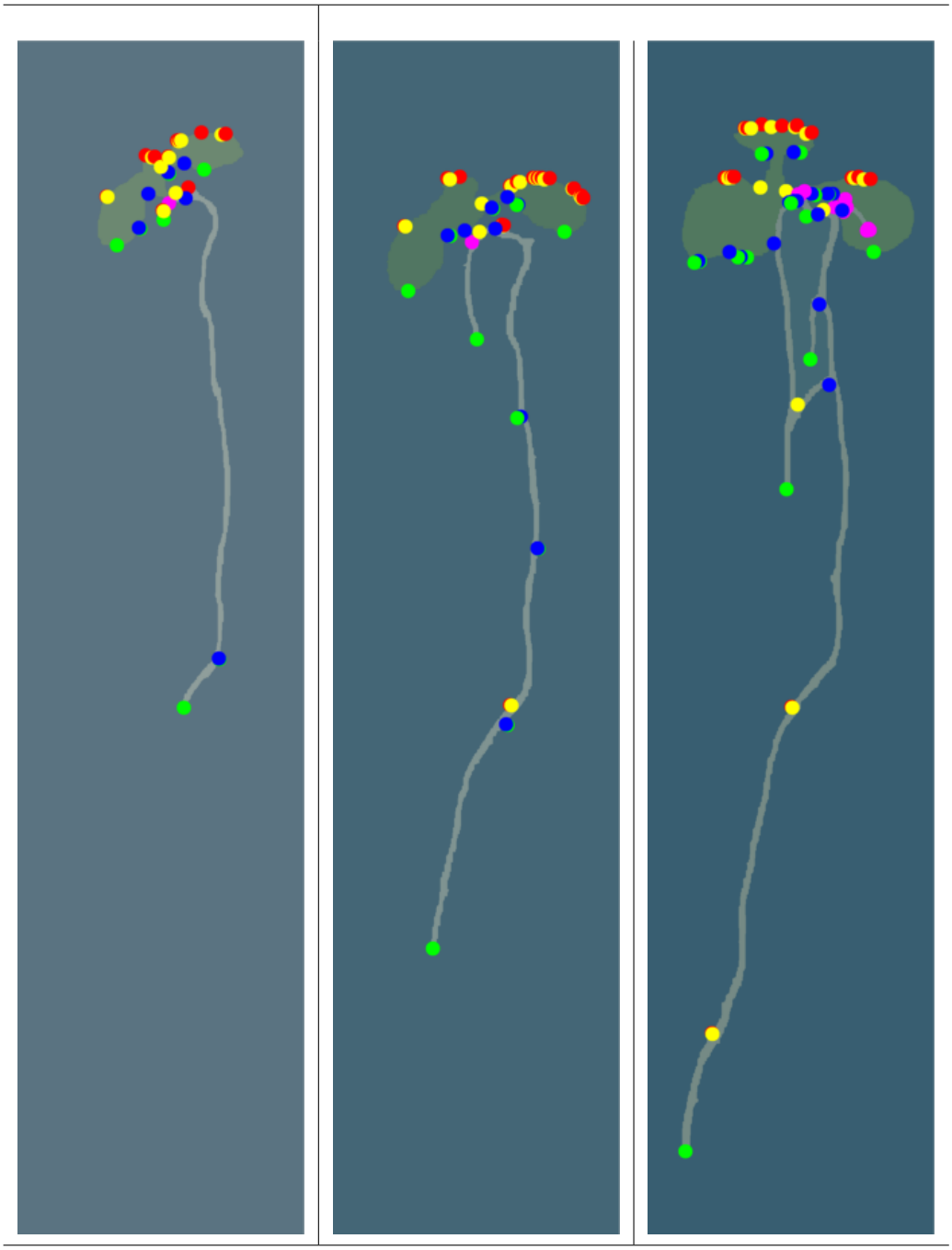


Figure 42: Results: critical points on root20 day 12, 16 and 20

13.2 Results: Reeb graph (image overlay)

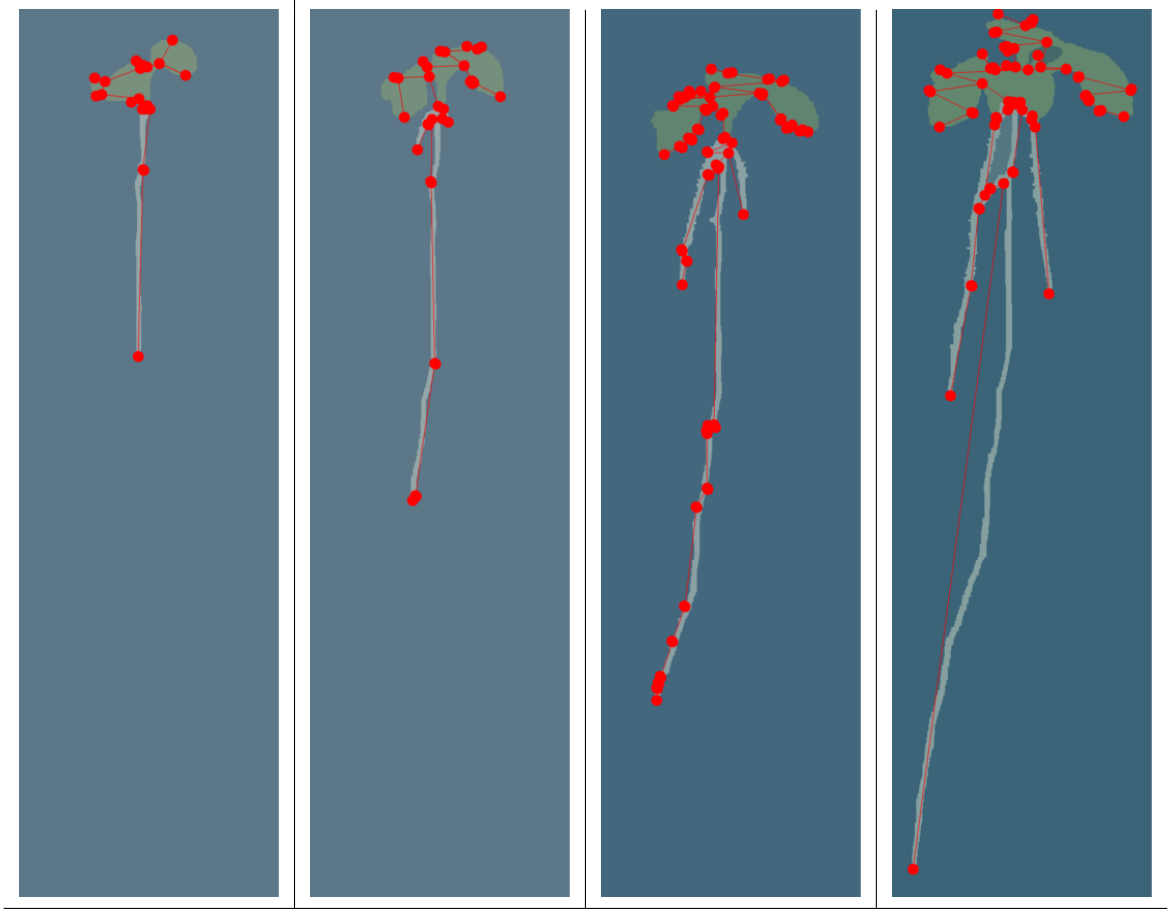


Figure 43: Results: Reeb graph on root04 day 8, 12, 16 and 20

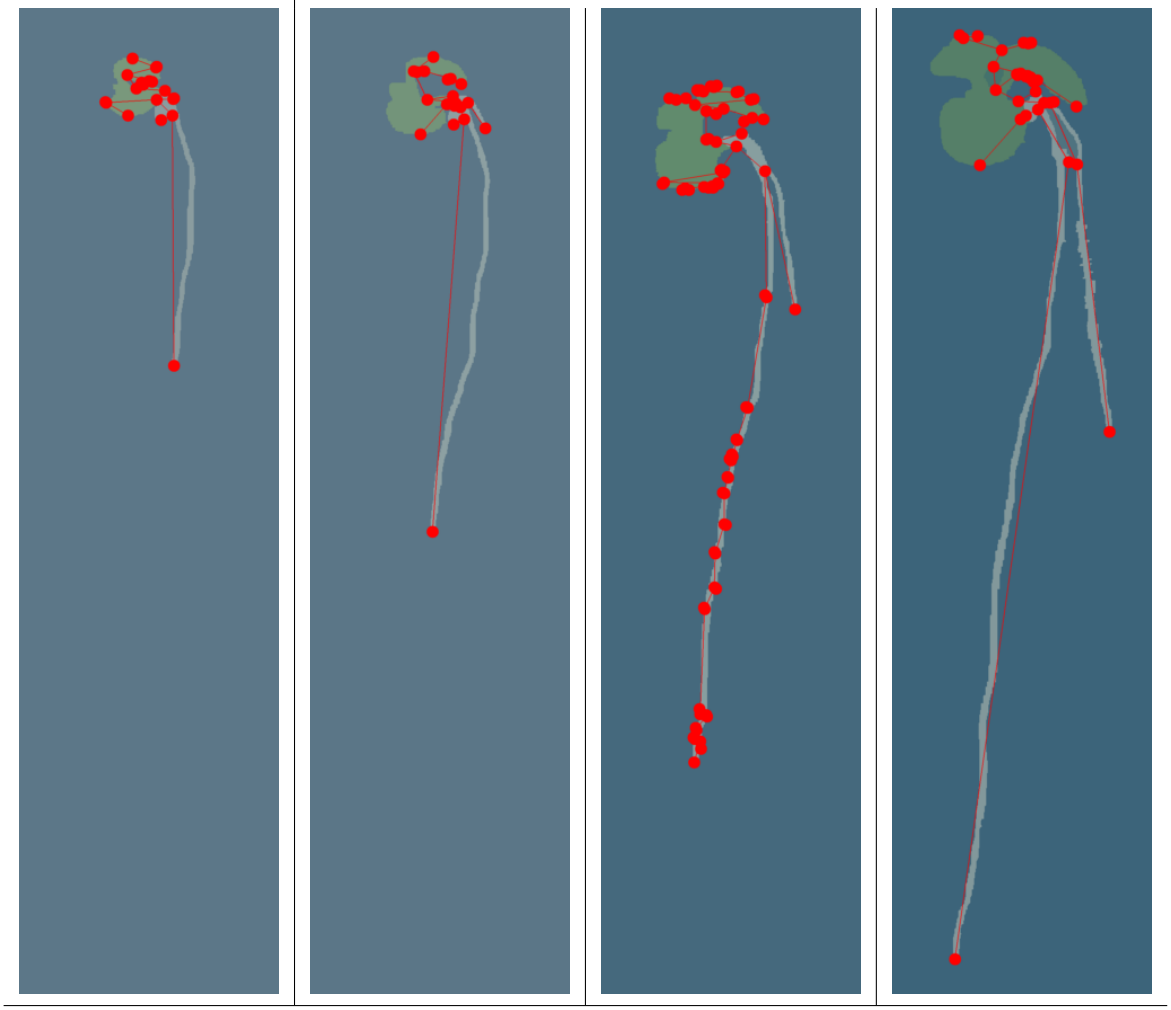


Figure 44: Results: Reeb graph on root05 day 8, 12, 16 and 20

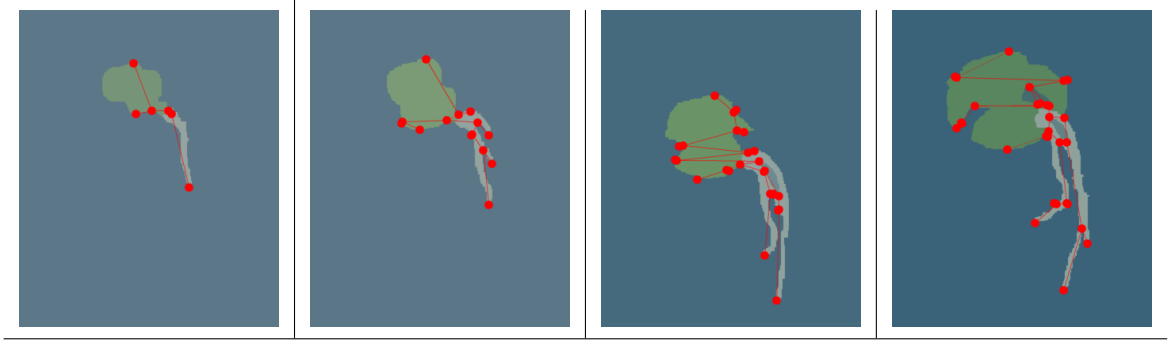


Figure 45: Results: Reeb graph on root07 day 8, 12, 16 and 20

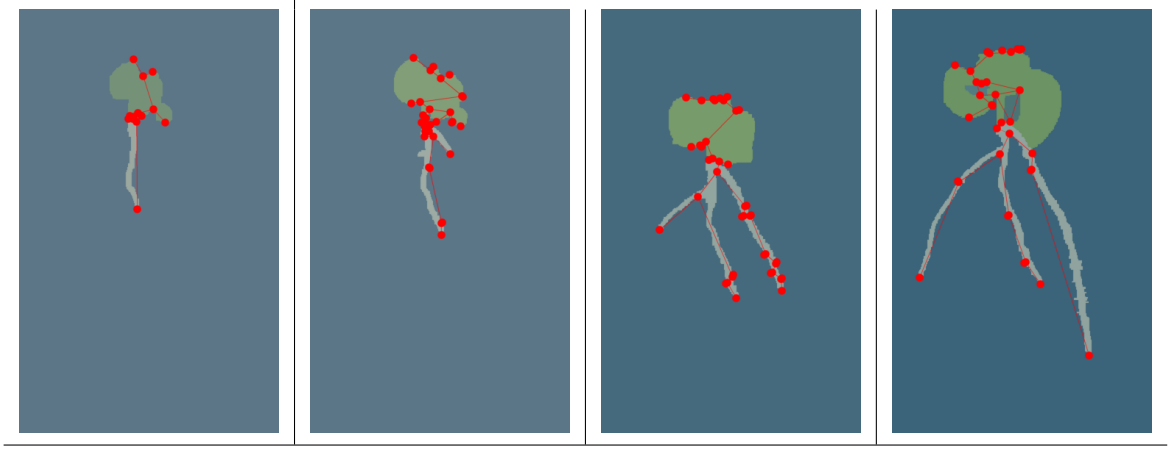


Figure 46: Results: Reeb graph on root09 day 8, 12, 16 and 20

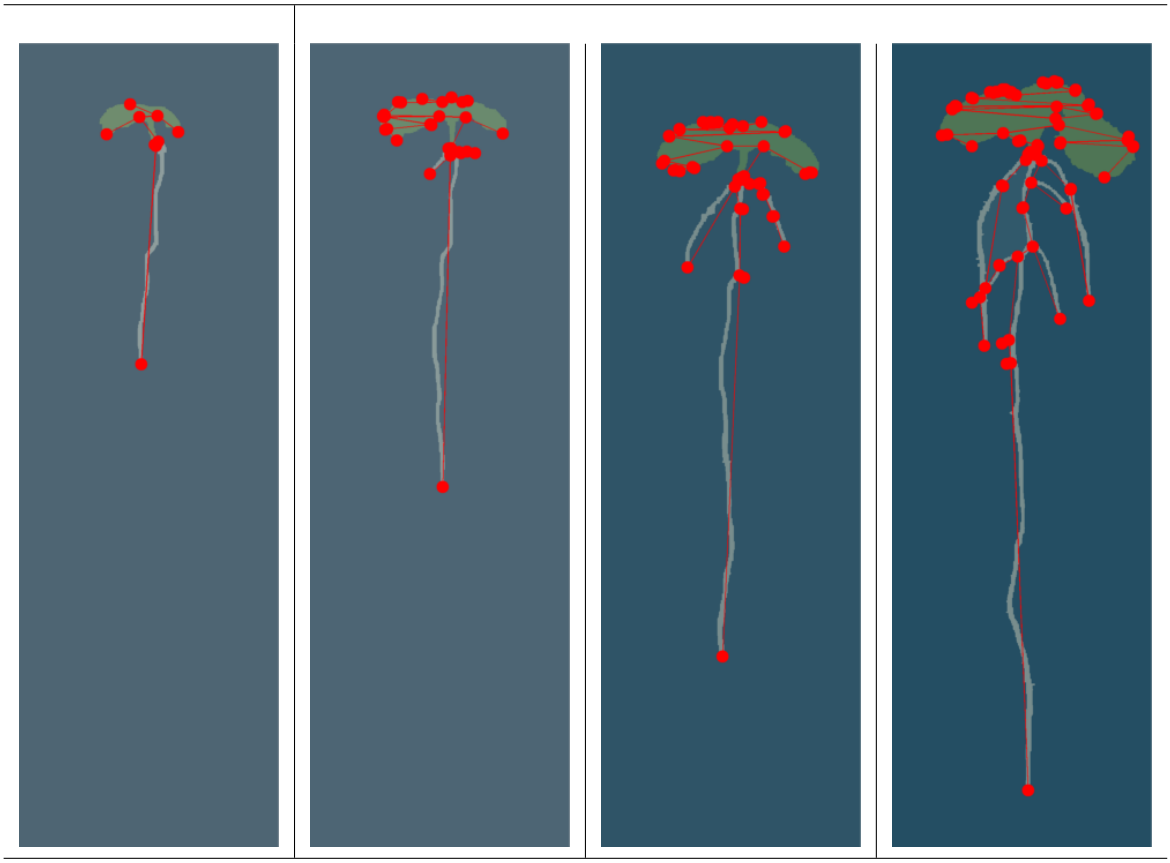


Figure 47: Results: Reeb graph on root12 day 8, 12, 16 and 20

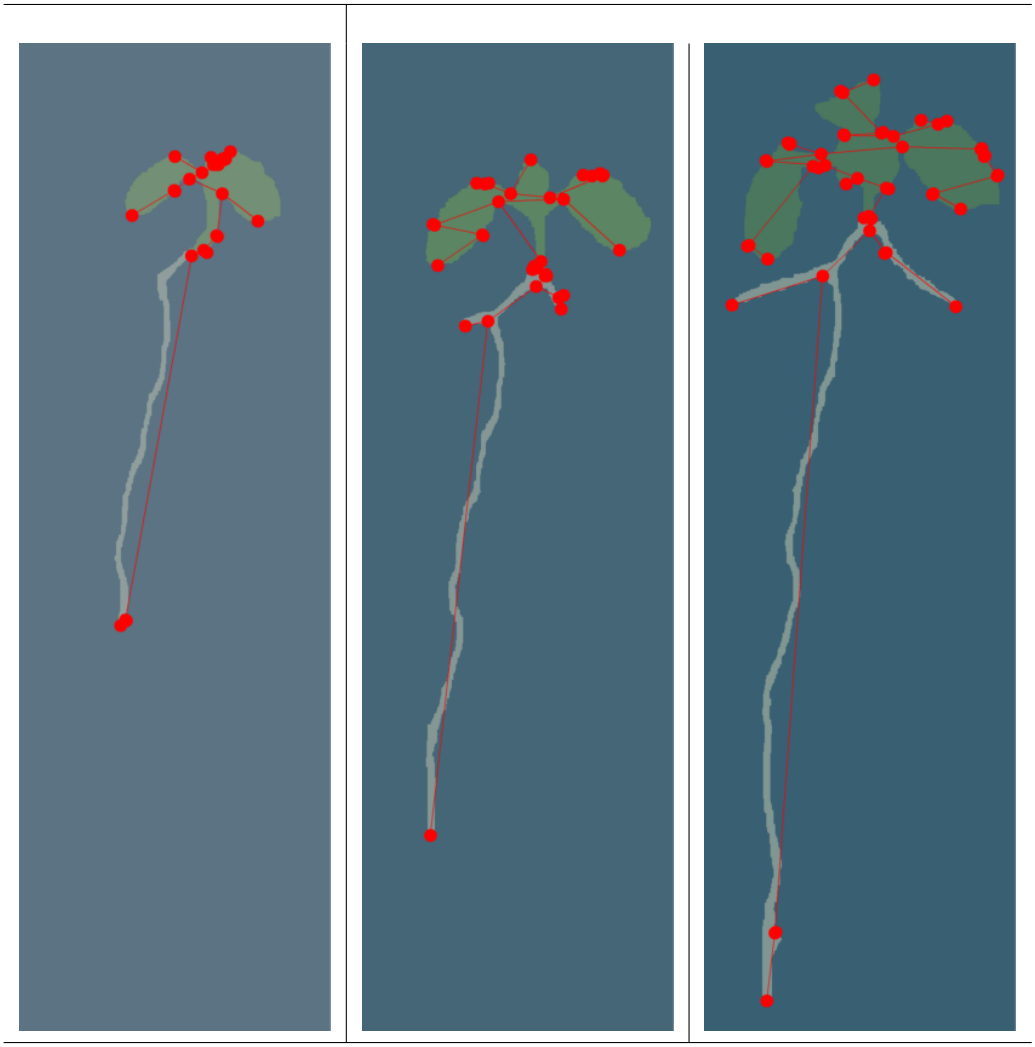


Figure 48: Results: Reeb graph on root17 day 12, 16 and 20

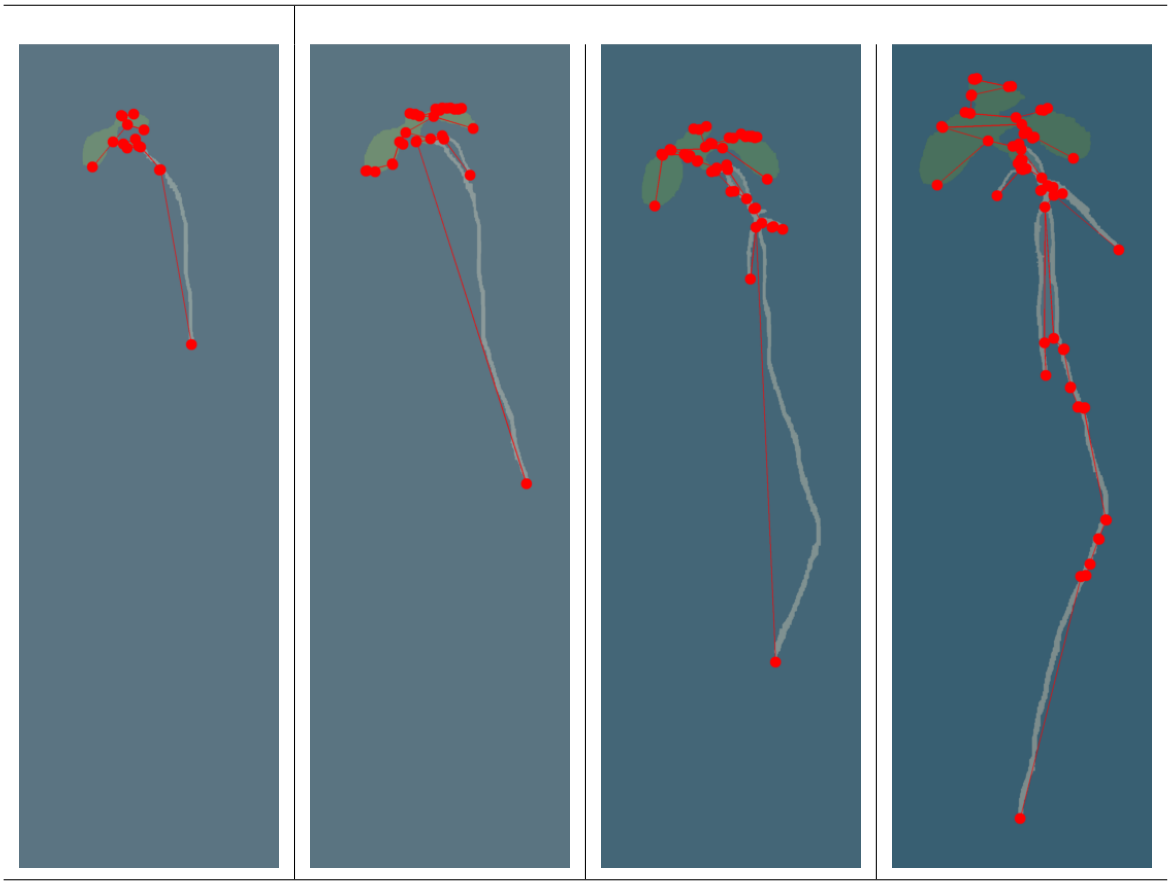


Figure 49: Results: Reeb graph on root19 day 8, 12, 16 and 20

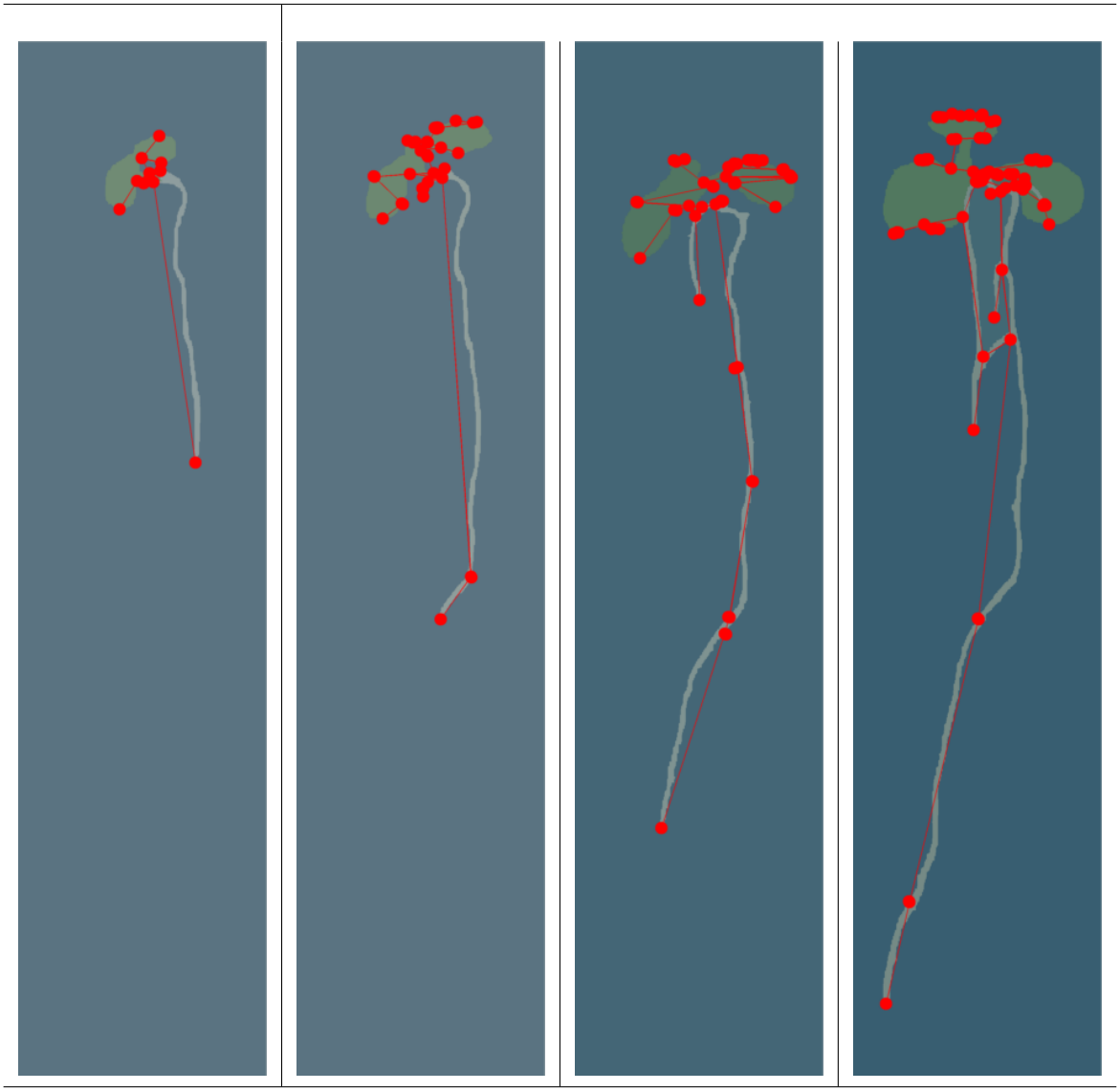


Figure 50: Results: Reeb graph on root20 day 8, 12, 16 and 20

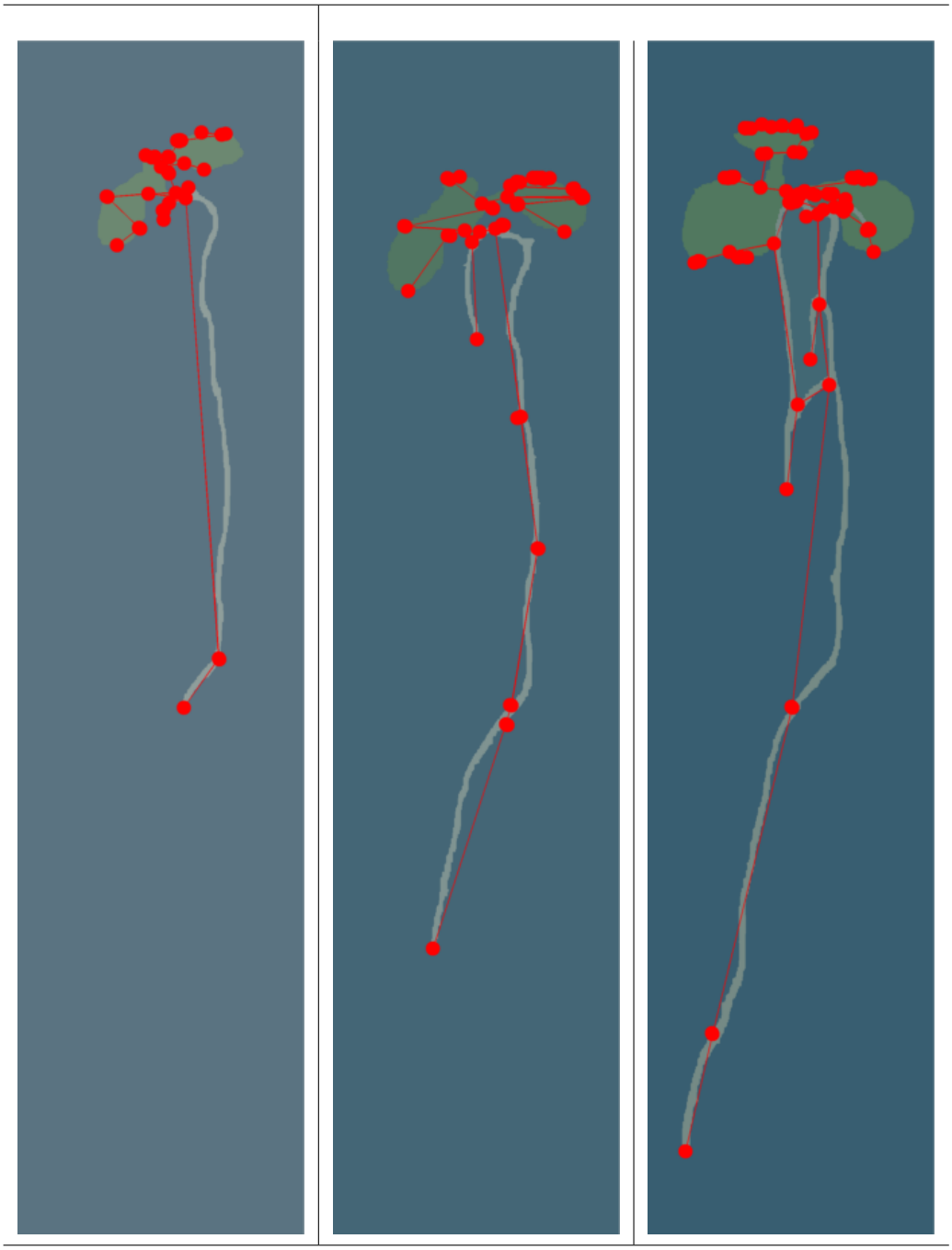


Figure 51: Results: Reeb graph on root20 day 12, 16 and 20

13.3 Results: Reeb graph (labeled nodes)

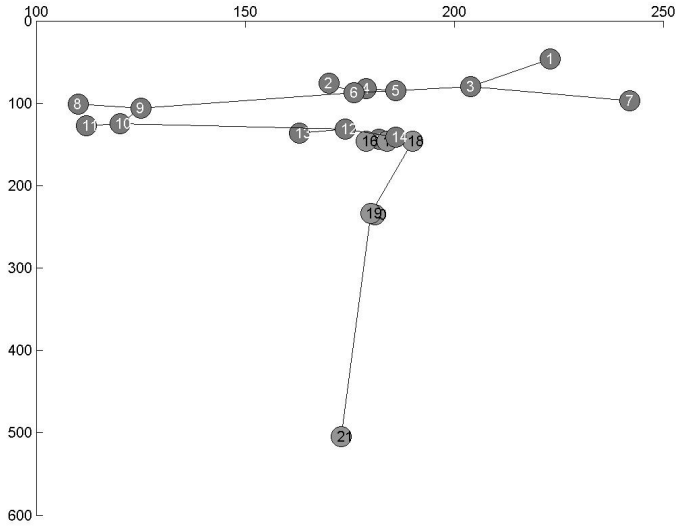


Figure 52: Results: labeled Reeb graph of root04 day 8

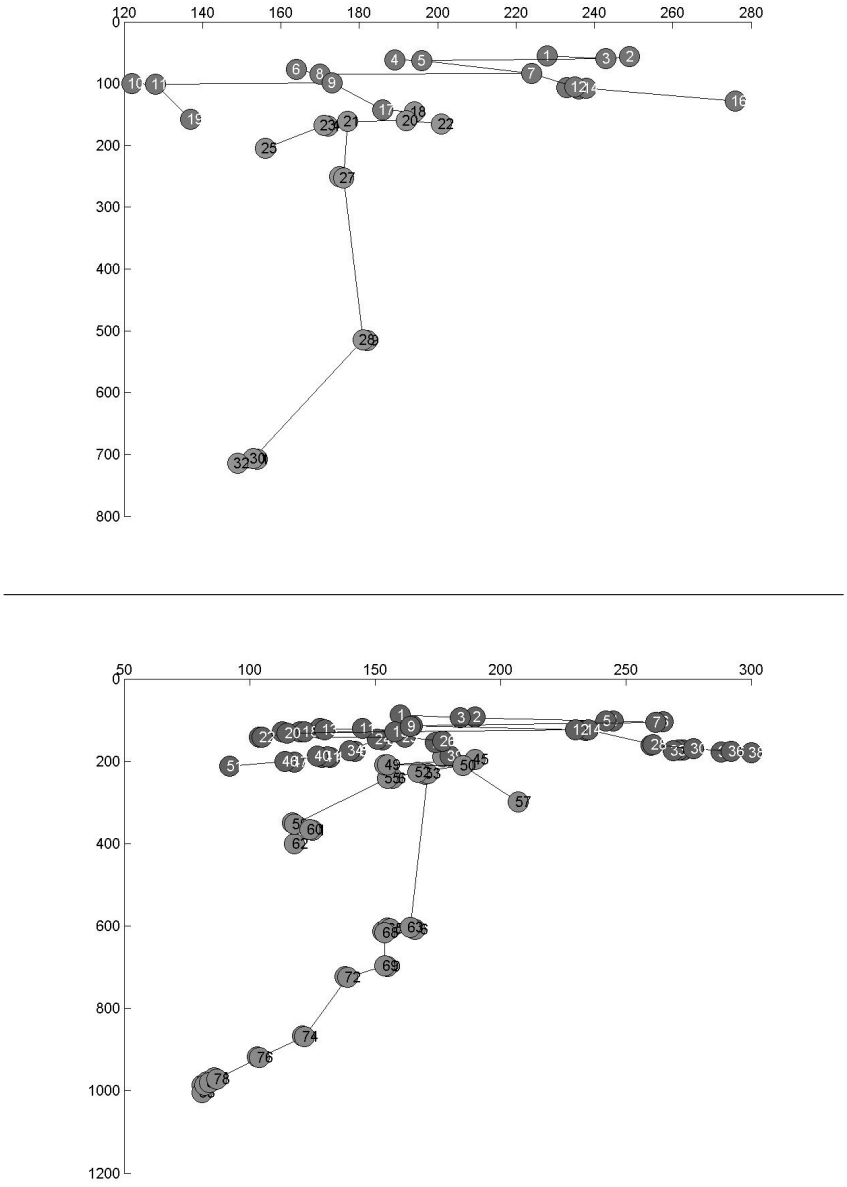


Figure 53: Results: labeled Reeb graph of root04 day 12 and 16

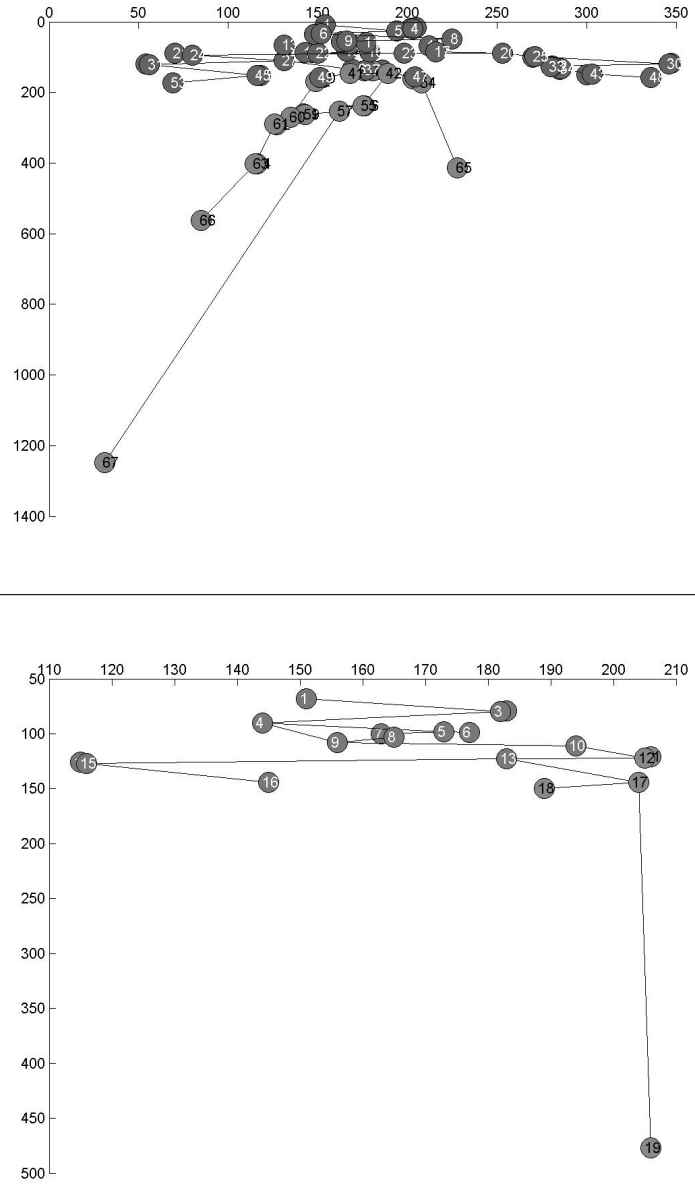


Figure 54: Results: labeled Reeb graph of root04 day 20 and root05 day 8

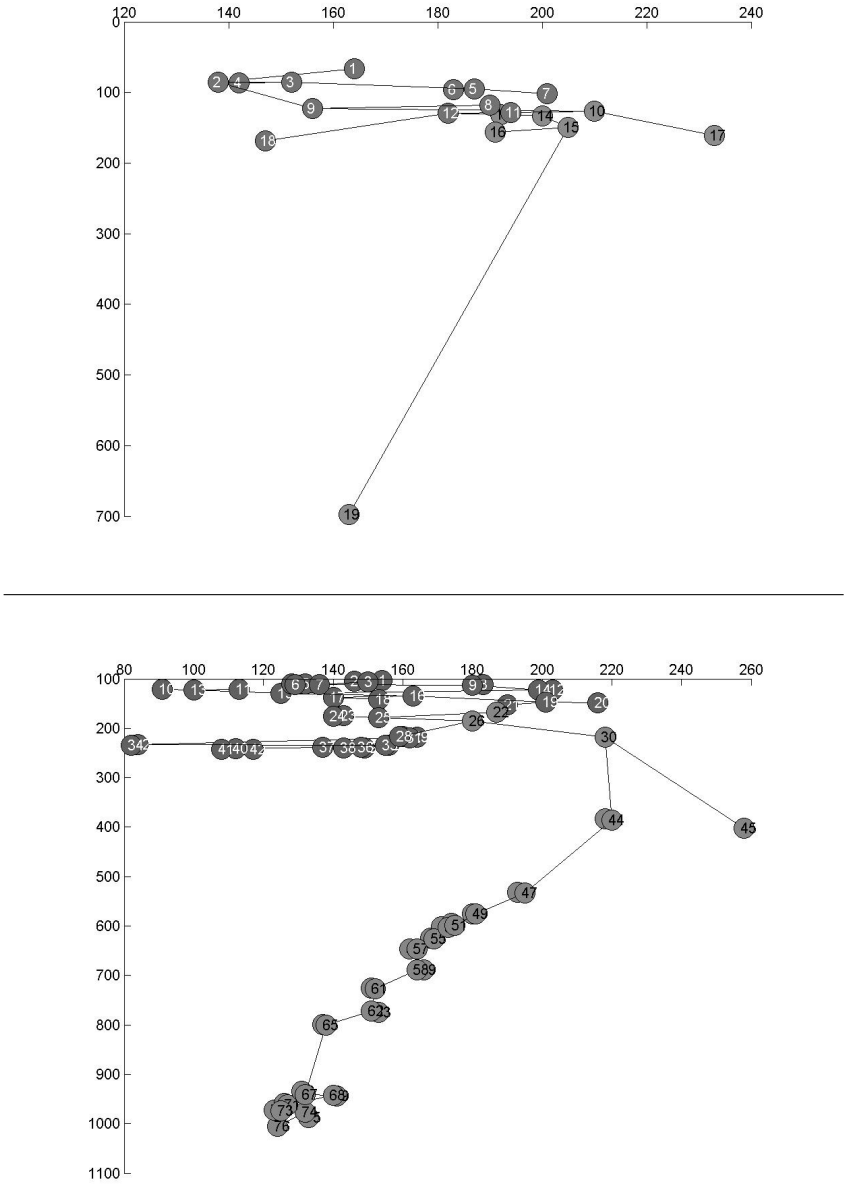


Figure 55: Results: labeled Reeb graph of root05 day 12 and 16

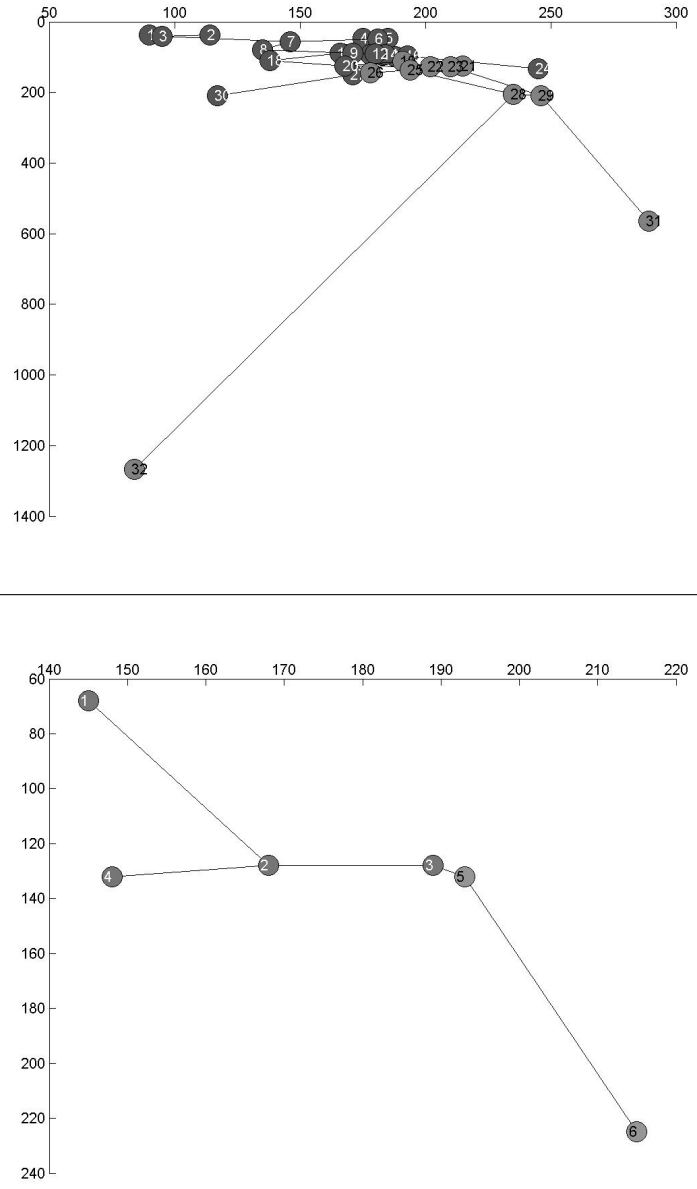


Figure 56: Results: labeled Reeb graph of root05 day 20 and root07 day 8

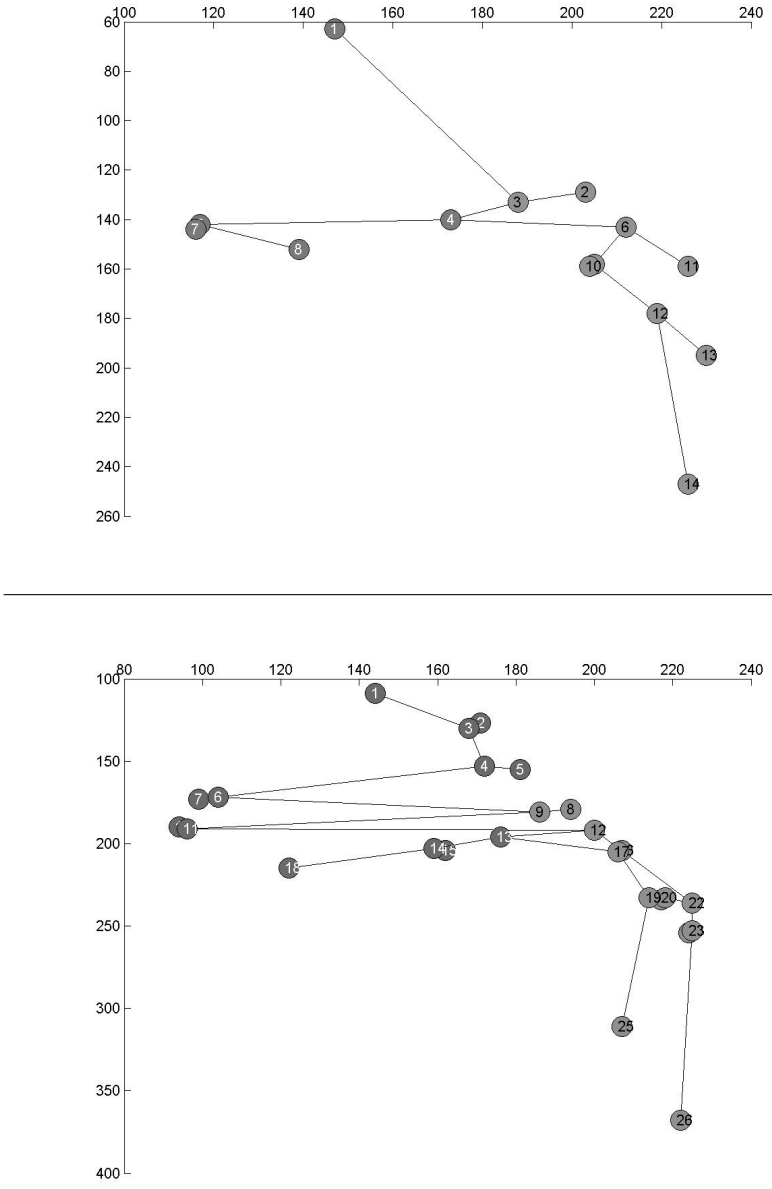


Figure 57: Results: labeled Reeb graph of root07 day 12 and 16

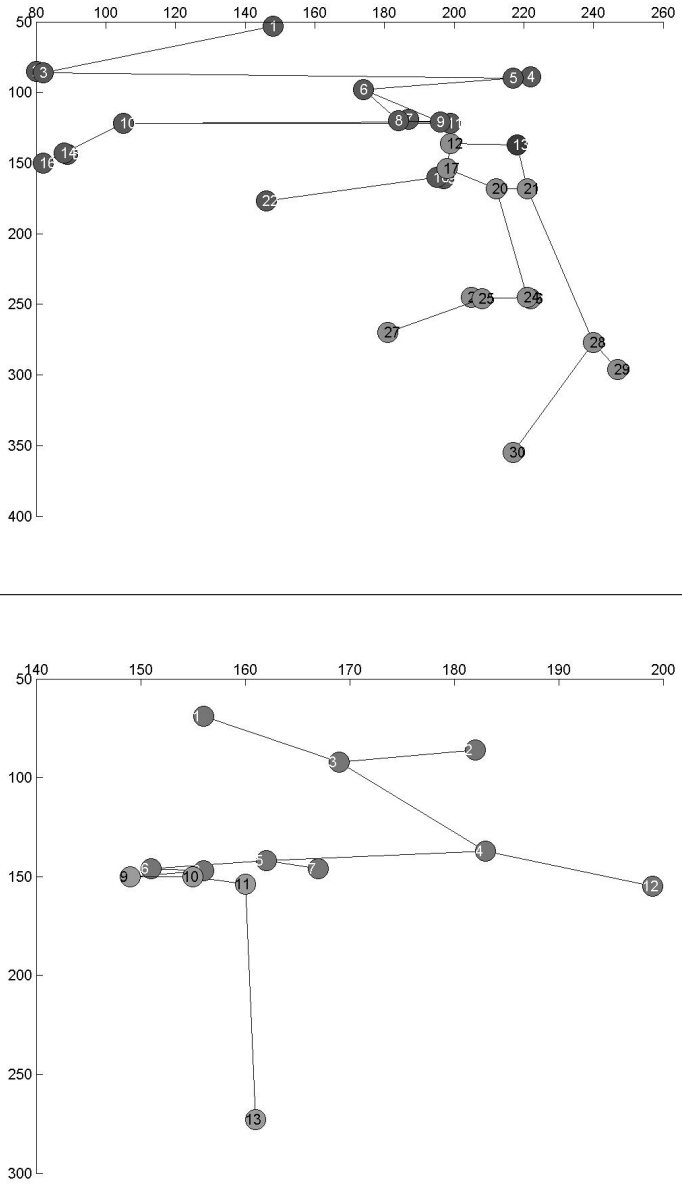


Figure 58: Results: labeled Reeb graph of root07 day 20 and root09 day 8

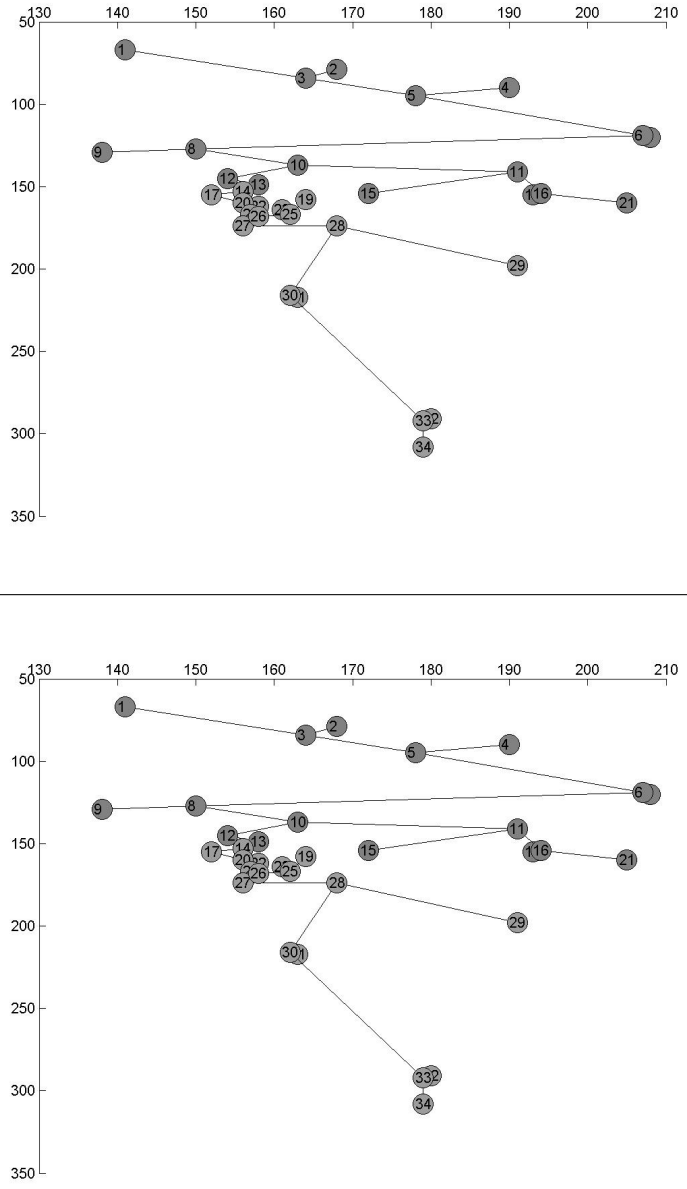


Figure 59: Results: labeled Reeb graph of root09 day 12 and 16

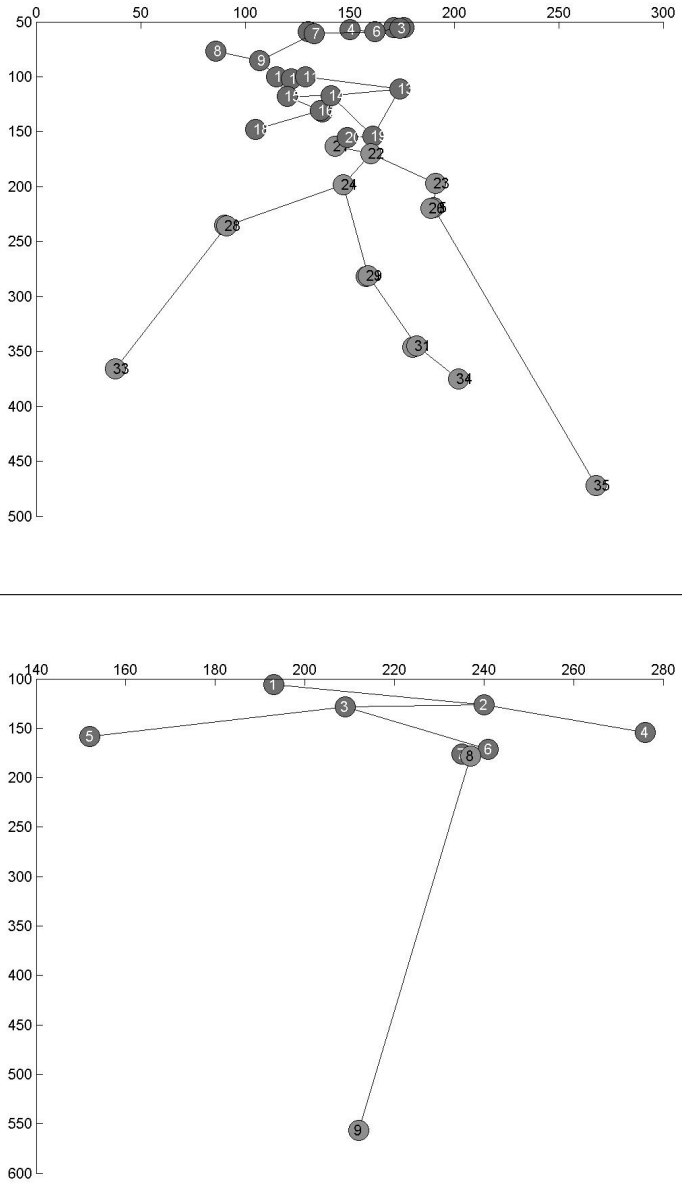


Figure 60: Results: labeled Reeb graph of root09 day 20 and root12 day 8

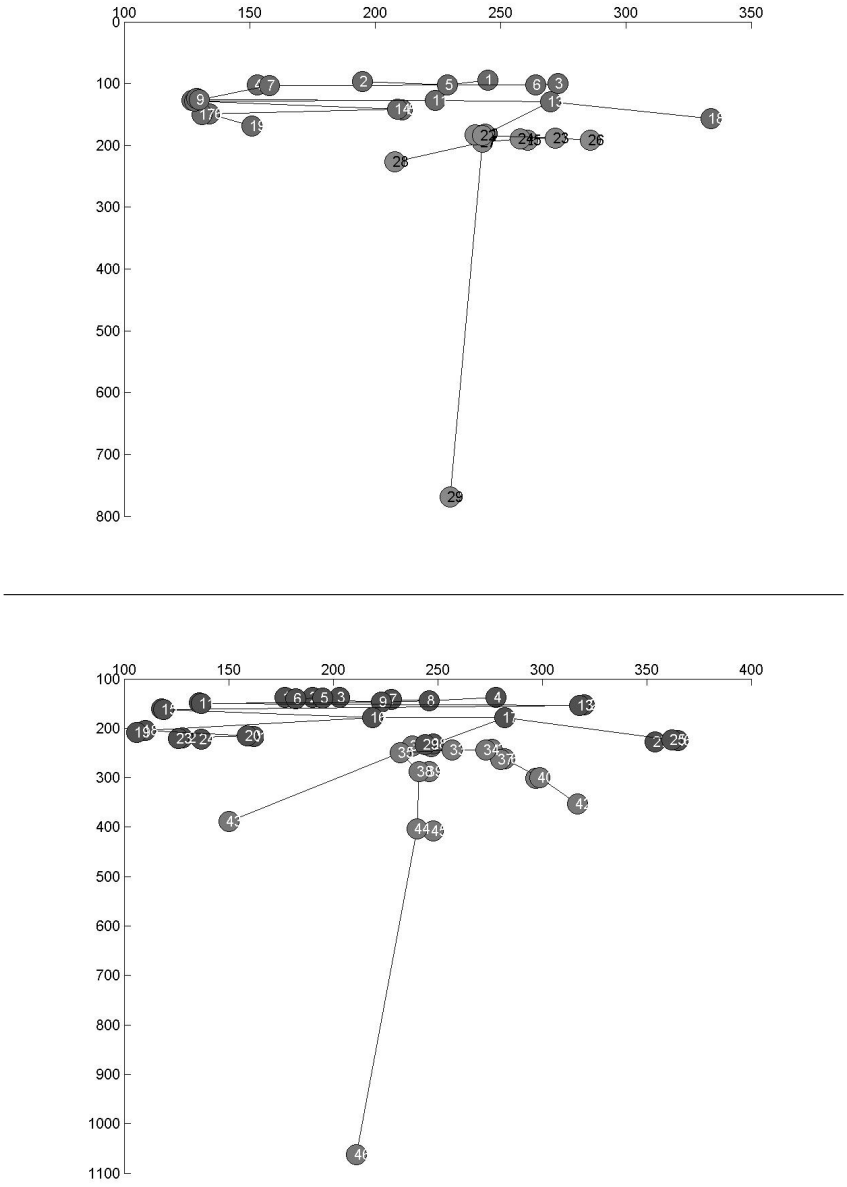


Figure 61: Results: labeled Reeb graph of root12 day 12 and 16

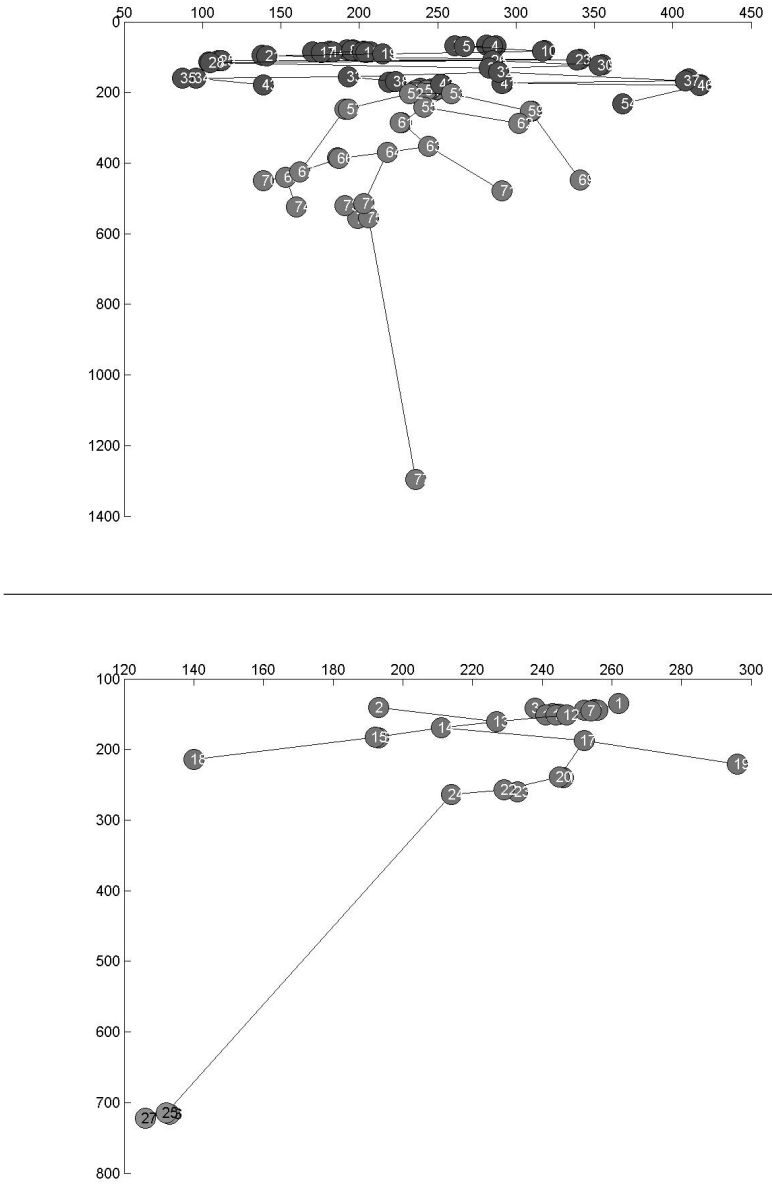


Figure 62: Results: labeled Reeb graph of root12 day 20 and root17 day 12

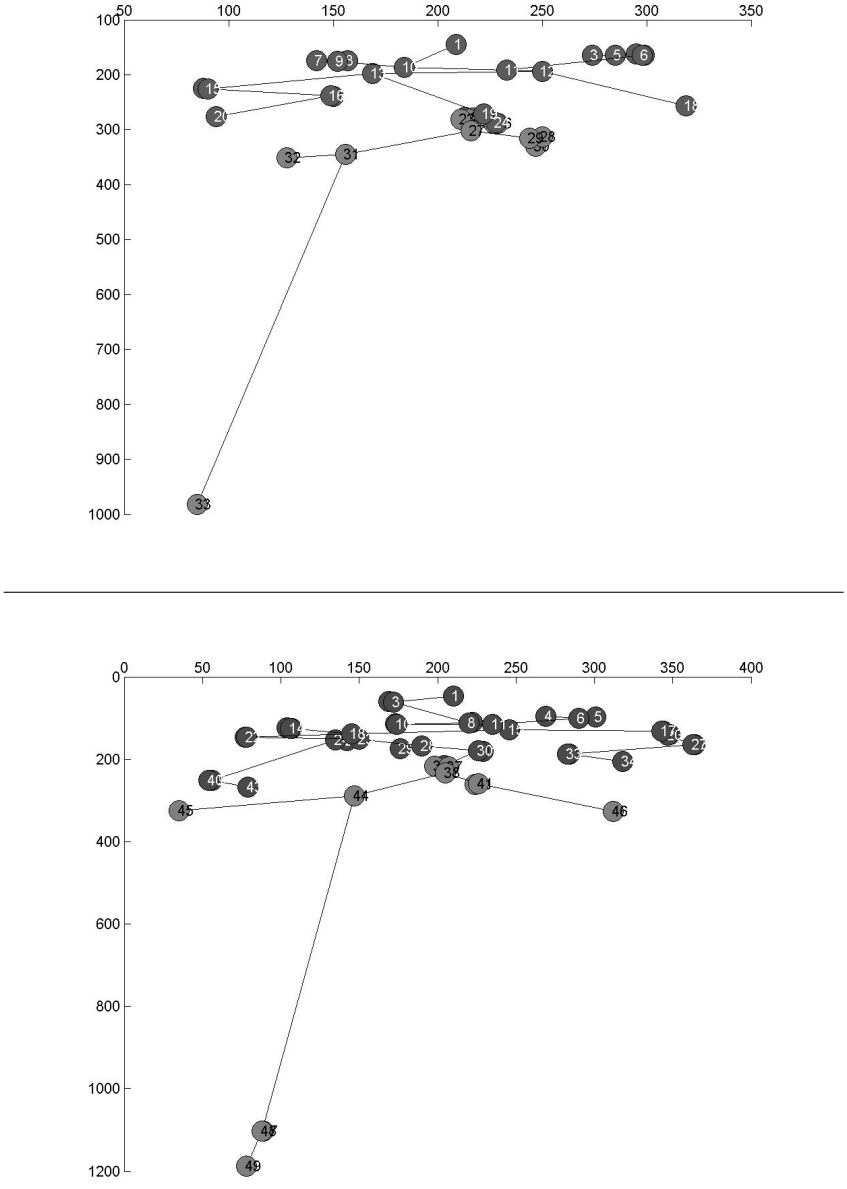


Figure 63: Results: labeled Reeb graph of root17 day 16 and 20

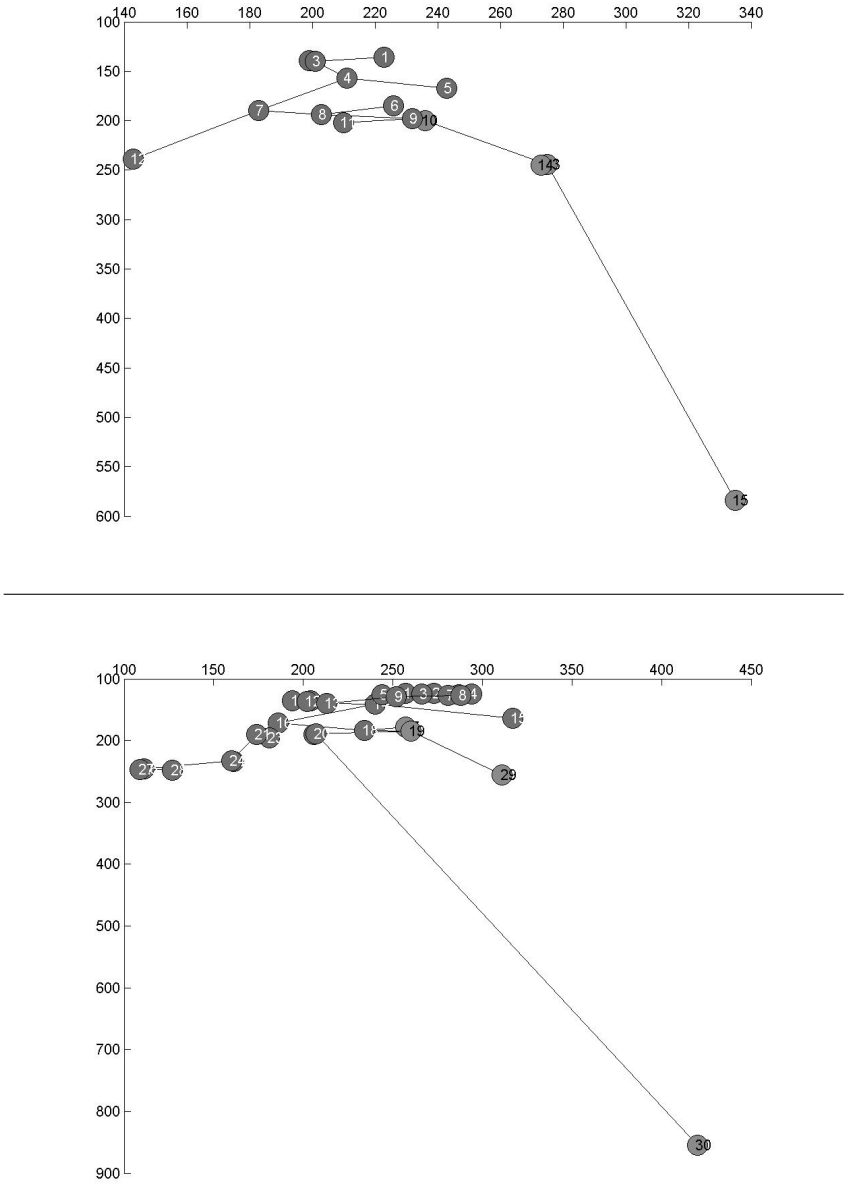


Figure 64: Results: labeled Reeb graph of root19 day 8 and 12

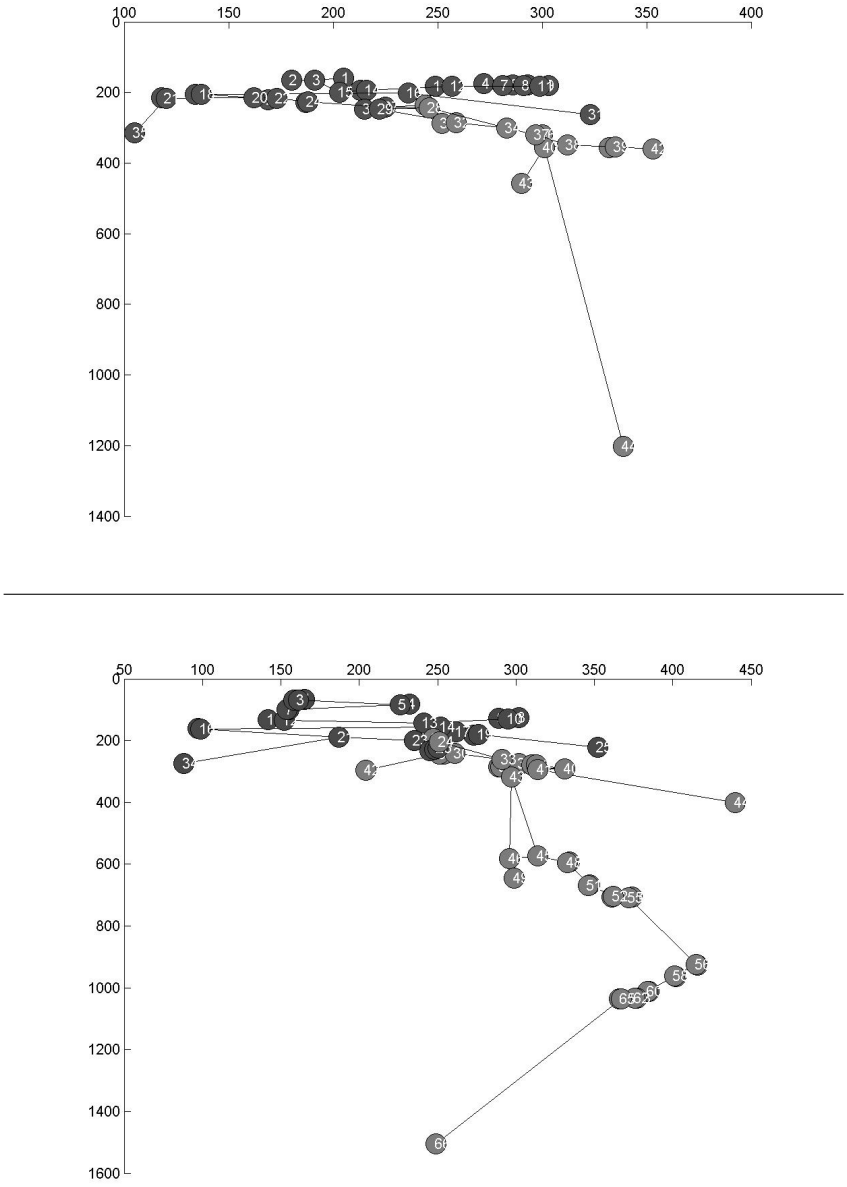


Figure 65: Results: labeled Reeb graph of root19 day 16 and 20

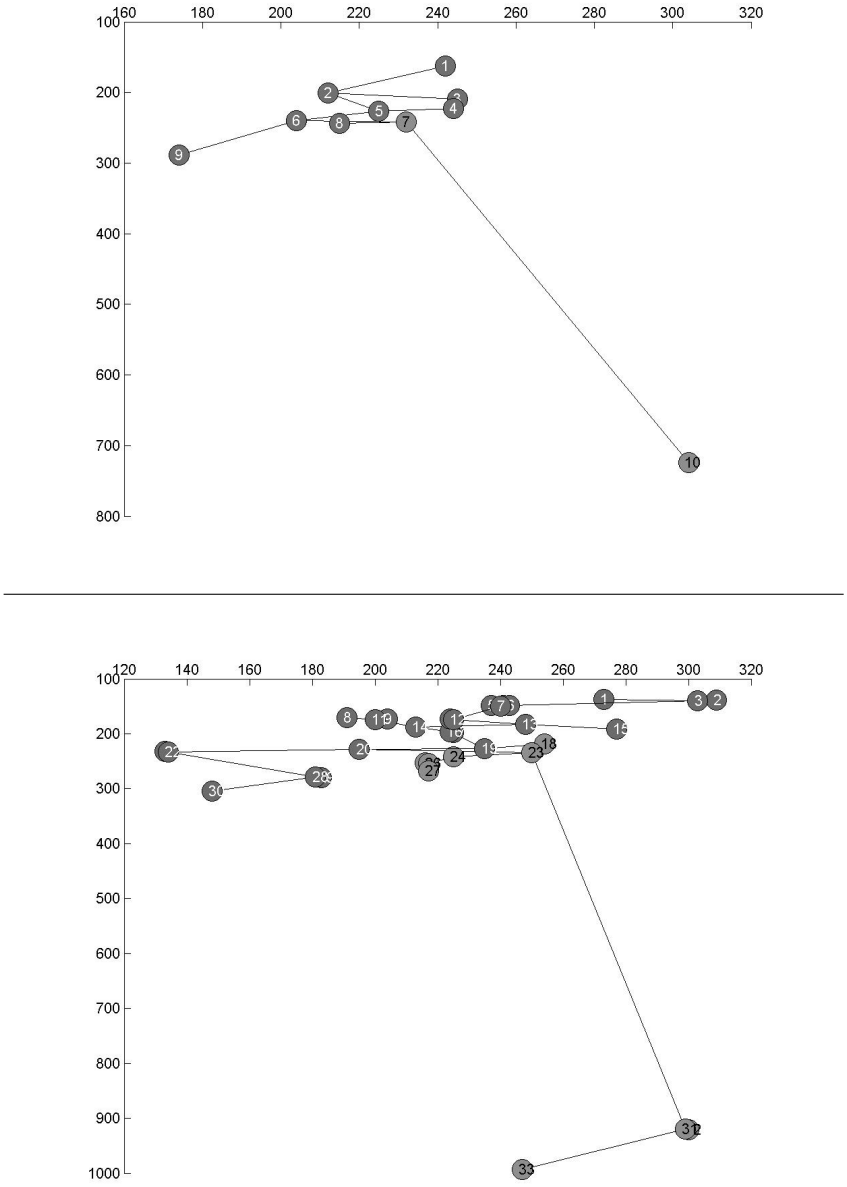


Figure 66: Results: labeled Reeb graph of root20 day 8 and 12

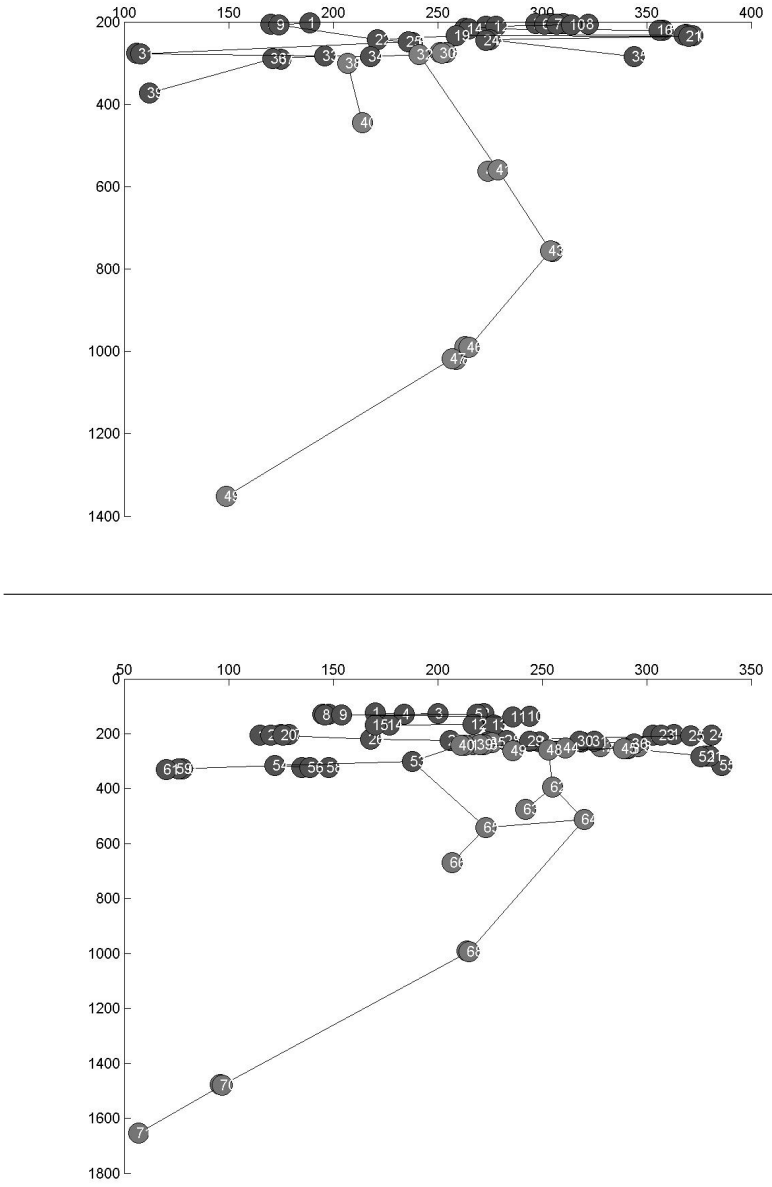


Figure 67: Results: labeled Reeb graph of root20 day 16 and 20

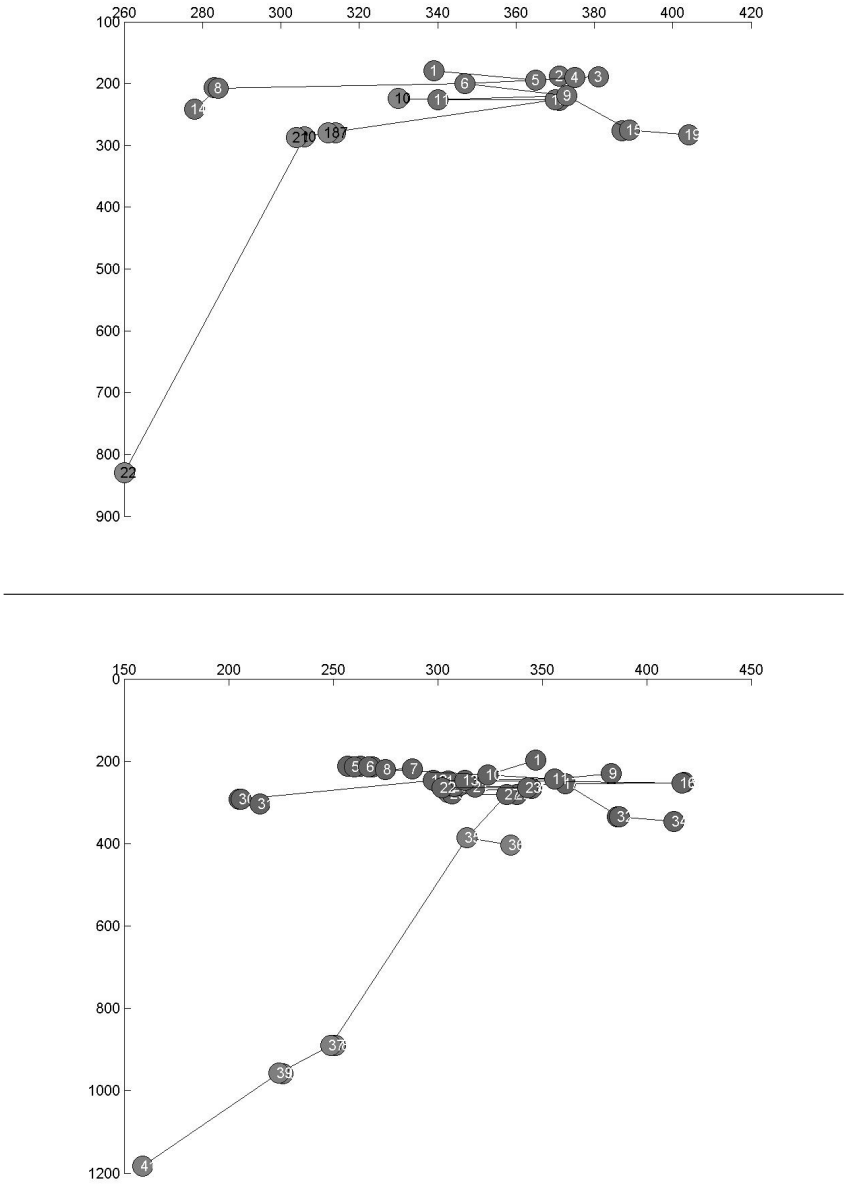


Figure 68: Results: labeled Reeb graph of root24 day 12 and 16

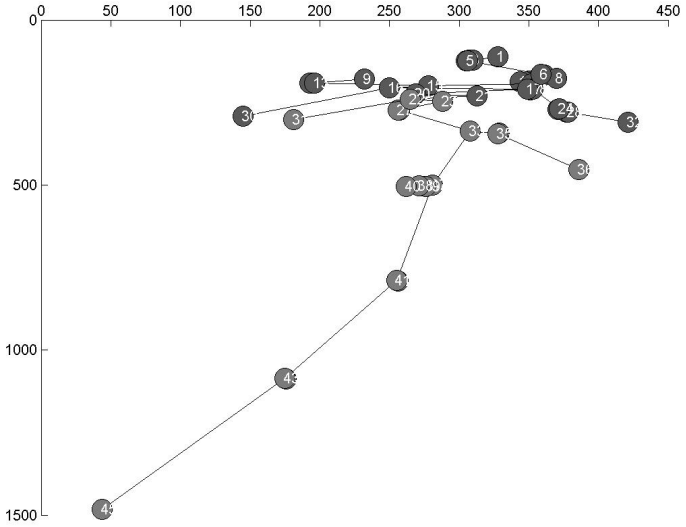


Figure 69: Results: labeled Reeb graph of root24 day 20

14 Discussion of the results

The results show that Reeb graphs, respectively the different types of nodes building the graph, are able to represent and describe image content based on critical points. The results on the test dataset (given in Chaper 12) demonstrate, that even changes in color in the foreground region can be described using regular nodes in the Reeb graph.
Representing the root images by graphs allows for an easy comparison of roots. Moreover characteristics such as distances between branches, numbers of branches or overlappings due to the 2D representation of a 3D structure can be measured directly on the graph.
The description of image content by Reeb graphs is however first of all dependent on the function used to compute the critical points. But image characteristics as resolution and the quality of the segmented images are of importance as well. Some problems on the root dataset that trace back to segmentation and image quality were already discussed in Section 10.
Especially the problem of additional critical points that was addressed in Section 10.1 can be observed (in varying degrees of occurance) for all images in the dataset.These additional points are introduced because of frayed borders of image regions due to the used segmentation method. These frayed borders are small structures in the image (compared to the whole root structure). Therefore the resulting critical points are positioned near to each other and the nodes in the Reeb graph might overlap when displaying the Reeb graph. In Figure 55 this can be seen in the lower part of the root. While the segmentation for root05 day20 did not fray at the borders and no additional critical points were introduced in these part, for root05 day16 several additional critical points were computed in the lower part of the root. These nodes are correctly computed and are based on frayed borders in the segmented image. For the root dataset this is a common problem, especially for images taken on day 16, details on this problem are given at the end of Section 3.

A Test dataset

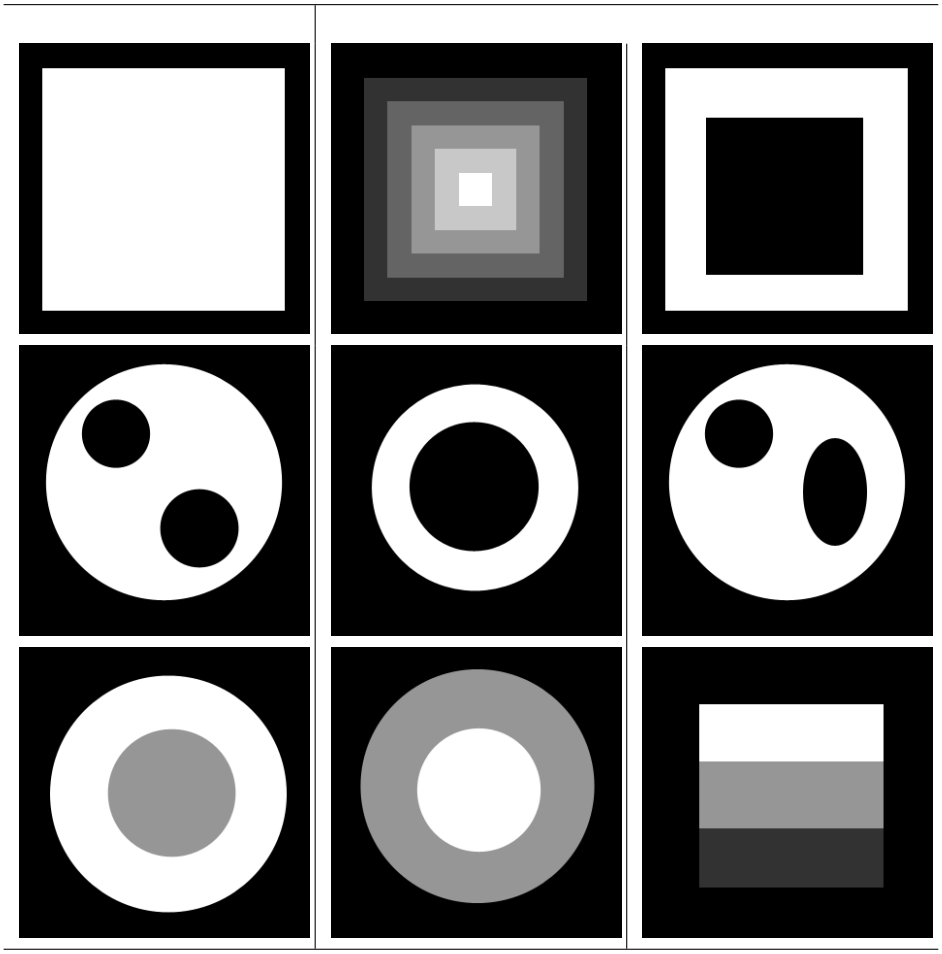


Figure 70: Synthetical test dataset - image 1-9

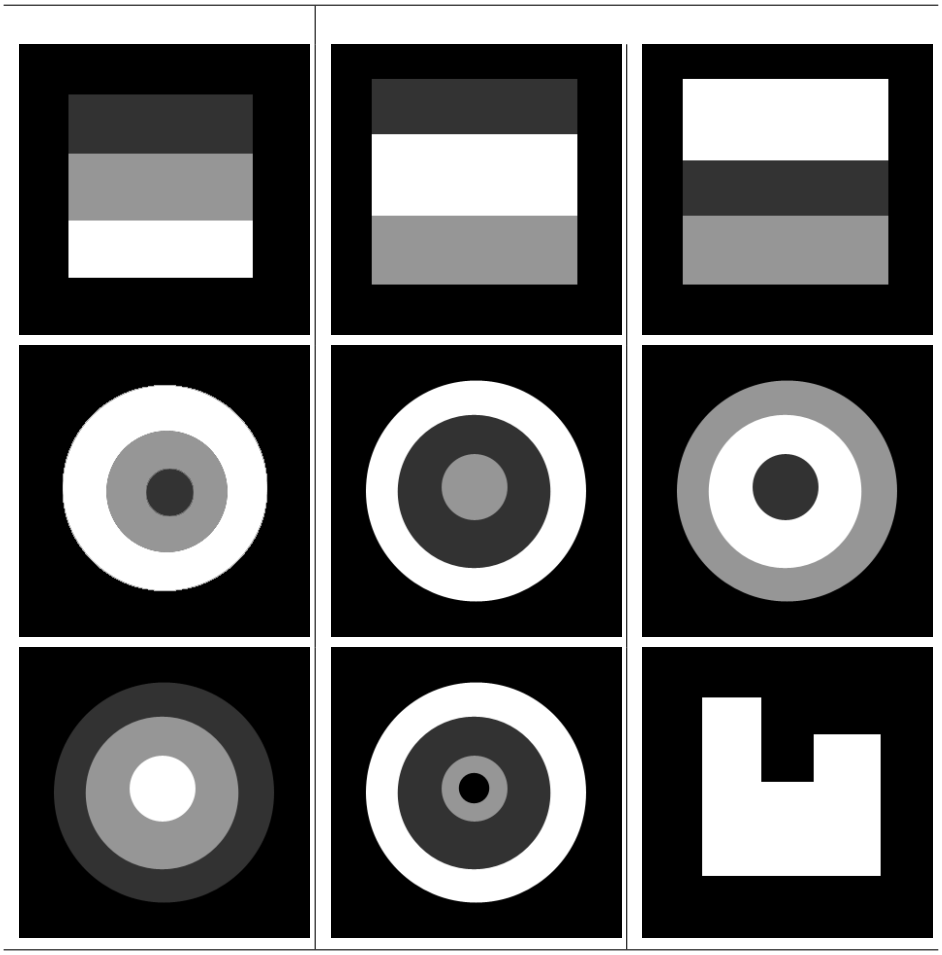


Figure 71: Synthetical test dataset - image 10-18

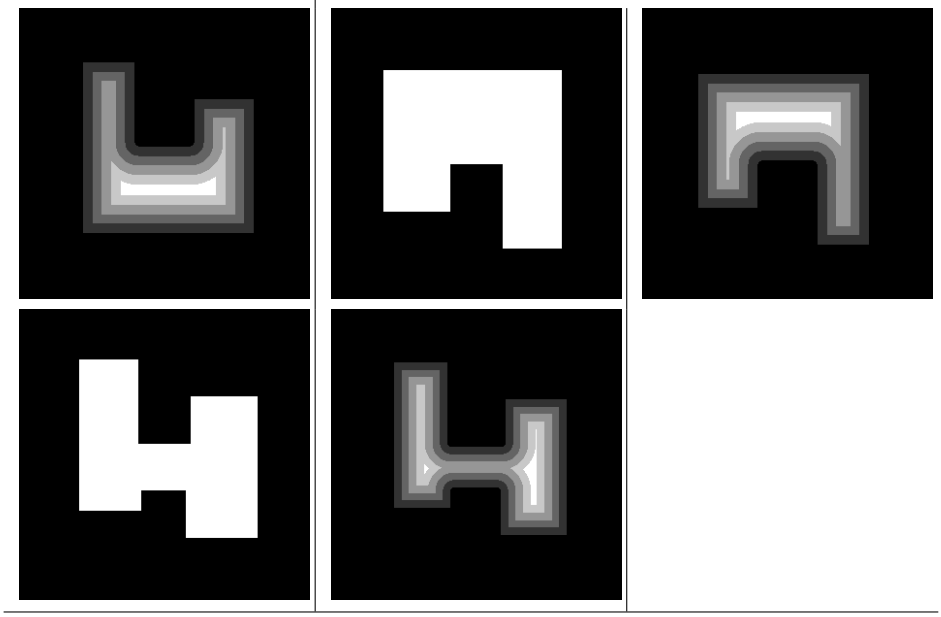


Figure 72: Synthetical test dataset - image 19-23

B Root dataset

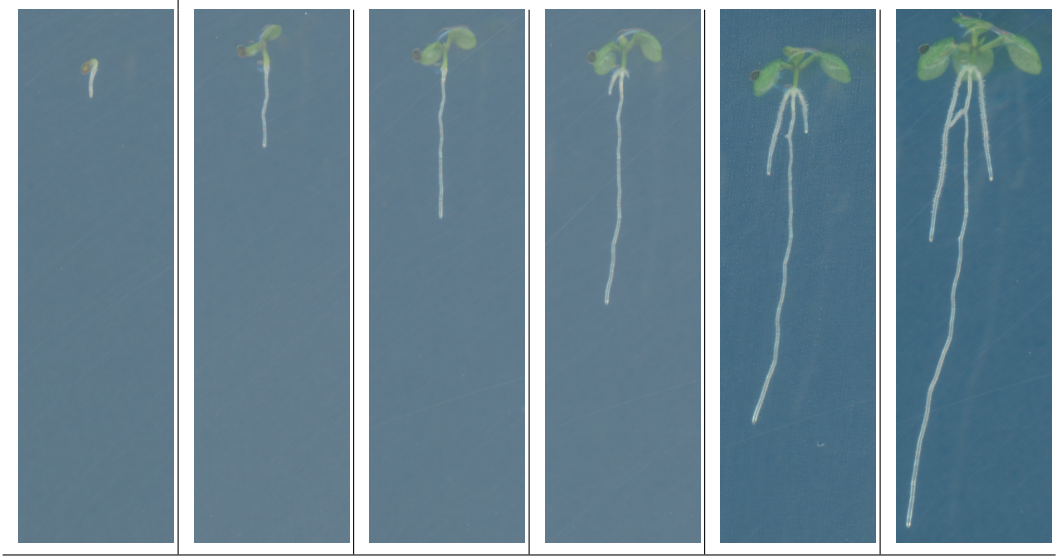


Figure 73: Root dataset “older age” - root004, day 1, 4, 8, 12, 16, 20

Figure 74: Root dataset “older age” - root005, day 1, 4, 8, 12, 16, 20

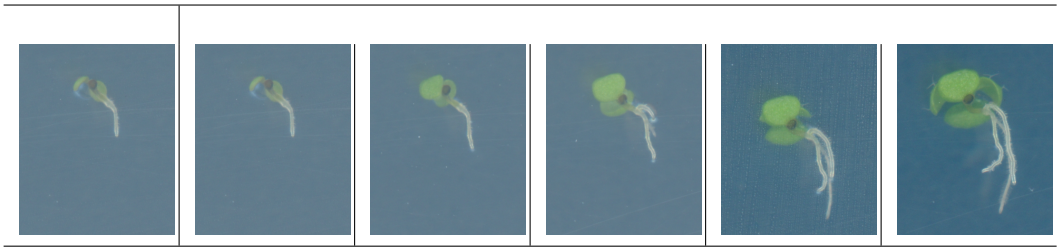


Figure 75: Root dataset “older age” - root007, day 1, 4, 8, 12, 16, 20

Figure 76: Root dataset “older age” - root009, day 1, 4, 8, 12, 16, 20

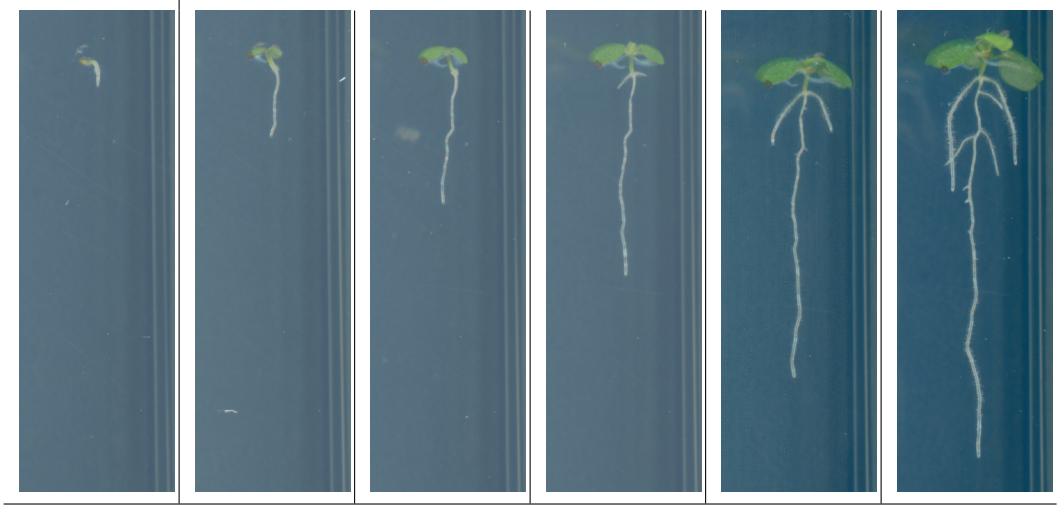


Figure 77: Root dataset “older age” - root012, day 1, 4, 8, 12, 16, 20

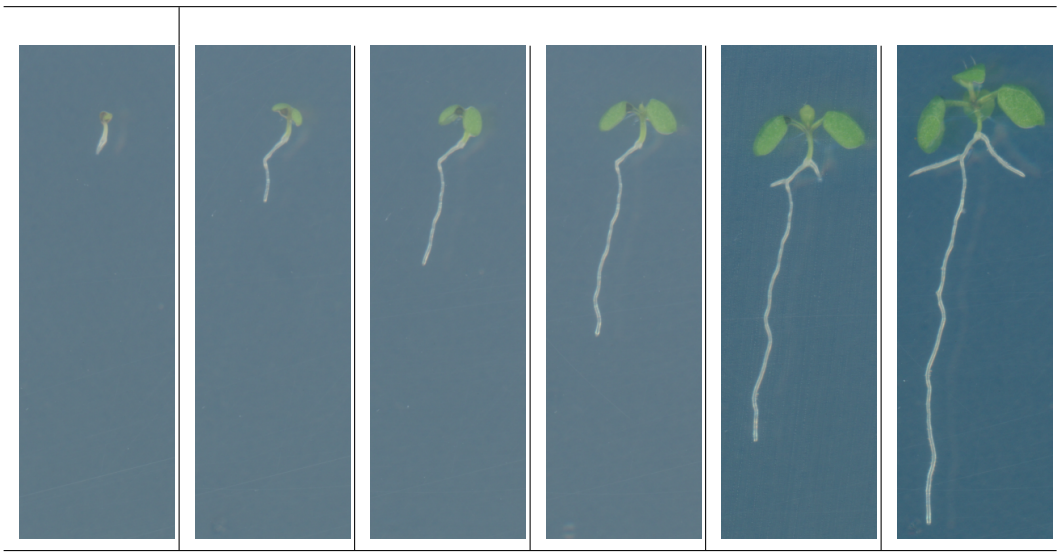


Figure 78: Root dataset “older age” - root017, day 1, 4, 8, 12, 16, 20

Figure 79: Root dataset “older age” - root019, day 1, 4, 8, 12, 16, 20

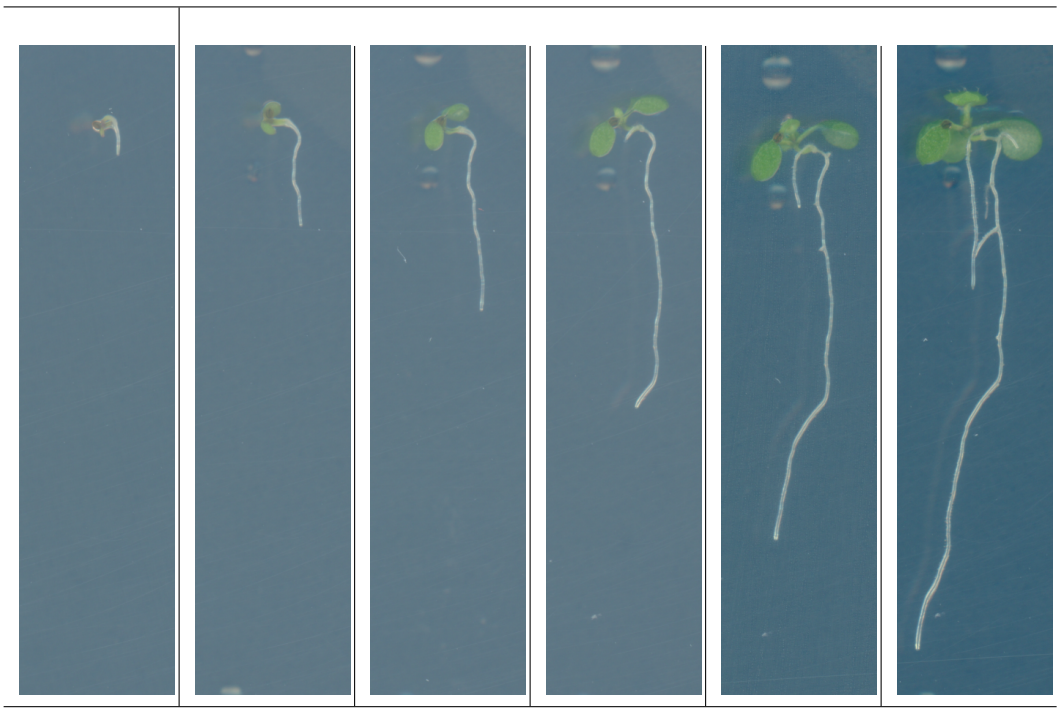


Figure 80: Root dataset “older age” - root020, day 1, 4, 8, 12, 16, 20

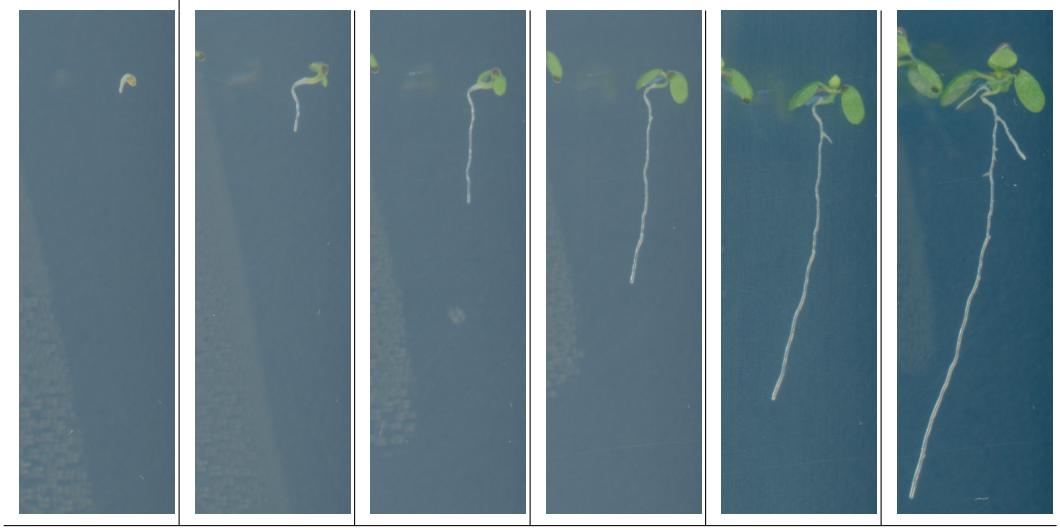


Figure 81: Root dataset “older age” - root024, day 1, 4, 8, 12, 16, 20

References

[1] S. Berretti, A. Del Bimbo, and P. Pala. 3D mesh decomposition using reeb graphs. *Image and Vision Computing*, 27(10):1540–1554, Sept. 2009.

[2] S. Biasotti. Topological techniques for shape understanding. In *IN CENTRAL EUROPEAN SEMINAR ON COMPUTER GRAPHICS, CESCG*, 2001.

[3] S. Biasotti. Reeb graph representation of surfaces with boundary. In *Shape Modeling Applications, 2004. Proceedings*, pages 371 – 374, June 2004.

[4] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 392(13):5–22, Feb. 2008.

[5] B. Di Fabio. Shape from Functions: Enhancing geometrical-topological descriptors. Technical report, Ferri, Massimo, 2009.

[6] H. Doraiswamy and V. Natarajan. Efficient algorithms for computing reeb graphs. *Computational Geometry*, 42(67):606–616, Aug. 2009.

[7] X. Ge, I. I. Safa, M. Belkin, and Y. Wang. Data skeletonization via reeb graphs. In J. Shawe-Taylor, R. S. Zemel, P. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 837–845. 2011.

[8] M. Gerstmayer, Y. Haxhimusa, and W. Kropatsch. Hierarchical interactive image segmentation using irregular pyramids. In X. Jiang, M. Ferrer, and A. Torsello, editors, *Graph-Based Representations in Pattern Recognition*, volume 6658 of *Lecture Notes in Computer Science*, pages 245–254. Springer Berlin / Heidelberg, 2011.

[9] R. Goffe, L. Brun, and G. Damiand. A top-down construction scheme for irregular pyramids. In A. Ranchordas and H. Araujo, editors, *Fourth International Conference On Computer Vision Theory And Applications (VISAPP'09)*, volume 1, pages 163–170, Lisboa, Portugal, Feb. 2009. 8 pages, 2 columns.

[10] M. Hayashi and M. Nishimura. Arabidopsis thaliana - a model organism to study plant peroxisomes. *Biochimica et Biophysica Acta (BBA) - Molecular Cell Research*, 1763(12):1382–1391, Dec. 2006.

[11] A. Ion and Y. Haxhimusa. Hierarchical image partitioning using combinatorial maps. In *Proceedings of the 10th Computer Vision Winter Workshop, CVWW 2005*, pages 43–152, 2005.

[12] G. Kaiser. Metasegmentation of remote sensing images based on combinatorial maps, 2003.

[13] W. Kropatsch, Y. Haxhimusa, and A. Ion. Multiresolution image segmentations in graph pyramids. In A. Kandel, H. Bunke, and M. Last, editors, *Applied Graph Theory in Computer Vision and Pattern Recognition*, volume 52 of *Studies in Computational Intelligence*, pages 3–41. Springer Berlin / Heidelberg, 2007.

[14] R. Marfil, L. Molina-Tanco, A. Bandera, J. Rodrguez, and F. Sandoval. Pyramid segmentation algorithms revisited. *Pattern Recognition*, 39(8):1430–1451, Aug. 2006.

[15] J. Stewart. *Calculus*. Cengage Learning Emea, 6th edition. international met edition, Feb. 2008.