

Additional File 2

```
1 //*****
2 //      Script name: CalloseMeasurer.script
3 //      Version: 1.0 - polished for publication
4 //
5 //
6 //      The Silke Robatzek Lab
7 //      The Sainsbury Laboratory, Norwich Research Park, Norwich, UK
8 //
9 //      Dr Ji Zhou
10 //      email: ji.zhou@tsl.ac.uk
11 //      Tel: + 44 (0) 1603 450316 (2316 internal)
12 //
13 //
14 //      System Requirements:
15 //          To allow use of this script, the Acapella framework (v2.0 or above) and Windows XP are required.
16 //
17 //
18 //      User Manual:
19 //          This software solution is designed to analyse spreading callose deposits on a plant leaf tissue
20 //          and, if necessary, to construct spreading callose networks. Output fields (in a variety of CSV files)
21 //          can be used to quantify features such as size, shape, fluorescent signal intensity, and pathogen growth
22 //          patterns (e.g. total length and spreading area). In order to measure callose deposits as well as detect
23 //          callose spreading networks, users need to follow the following instructions:
24 //
25 //          *      Selecting "Image Directory" selection to tell CalloseMeasurer where to read microscopic images
26 //
27 //          *      If users would like to only detect only big callose deposits, please tick the "Include large callose" selection
28 //
29 //          *      If users would like to only detect only small callose deposits, please tick the "Include small callose" selection
30 //
31 //          *      If users would like to only detect both small and big callose deposits, please tick both "Include small callose"
32 //          and "Include large callose" selections
33 //
34 //          *      If users would like to preview analysed images within the Acapella, please tick "Show Illustration"
35 //          selection, otherwise analysed images will be exported to the result folder, which will be generated by
36 //          CalloseMeasurer during the batch process
37 //
38 //          *      If users would like to remove detected objects attached to the image border, please tick the "Remove objects
39 //          attached to the image border" selection.
40 //
41 //          *      If users would like to detect spreading callose networks, please tick the "Detect Callose Network" selection -
42 //          be aware that this function is time-consuming. Although computation time depends on computer hardware,
43 //          the version of the Acapella system used for the analysis, and the complexity of the pathogen growth patterns,
44 //          CalloseMeasurer normally needs to spend 5-6 minutes to construct a modestly complicated callose network.
45 //          However users could always enter a smaller distance threshold (in pixel value) before constructing callose
46 //          networks. The threshold can be entered in the input box "The Distance Threshold of Callose Networks".
47 //          The default value is 25 pixels, which can be changed by the software according to the callose diameters
48 //          and full length of recognised objects of callose deposits.
49 //
```

Additional File 2

```
50 //      *      Users can select the input image format on the "Image Format" dropdown list - TIFF, PNG, BMP, and JPG
51 //      formats can be read and processed by CalloseMeasurer. The default input image format is TIF/TIFF files
52 //
53 //      This software solution performs image analysis functions such as image enhancement, object segmentation, filtering
54 //      noise objects, splitting fused callose signals, edge-based measurement, and construct networks for detecting pathogen
55 //      growth patterns in leaf tissues. New functions can be added as external procedures or internal functions.
56 //
57 //
58 //
59 //      Copyright: (C) The Sainsbury Laboratory
60 //
61 //
62 //*****
63
64
65
66 //// Establish input parameters
67 set( formats= " \".bmp\", \".jpg\", \".png\", \".tiff\" " )
68
69 //*** INPUT PARAMETERS ***//
70 Input(filepath, NO, type="p", prompt="Image Directory: ", description="Please select the directory of input callose images...")
71 input(ShowIllustrations, NO, "Show Illustrations", "y", "YES- Output illustrations are depicted. No- Output illustrations are not shown.")
72 input(LargeCallose, Yes, "Include large callose", "y", "YES- Analysis will retain large callose. No- Objects will remove large callose objects during the detection.")
73 input(SmallCallose, Yes, "Include Small Callose", "y", "YES- Analysis will retain small callose (callose with low contrast will be removed as required). No- Objects will remove small and dim callose objects during
74 the analysis.")
75 input(RemoveBorderObj, Yes, "Remove Objects Attached to The Image Border", "y", "YES- Analysis will remove objects attached to the image border. No- Objects attached to the image border will be retained.")
76 input(Callose_Network_Trigger, NO, "Detect Spreading Callose Networks", "y", "YES- Export an image that contains spreading callose network together with quantifiable attributes such as area and full length. No-
77 No callose networks will be constructed.")
78 input(Network_Threshold, 25, "The Distance Threshold of Callose Networks", "i", "Enter the maximum distance in pixels between two callose objects when establishing callose networks - default value (25 pixels) might
79 be changed according to the recognised callose objects")
80 input(IN_image_format, "TIFF", "Image format: Select Input Image Format", "s", "Possible values: \".PNG\", \".TIFF\", \".JPG\", \".BMP\". Format \".tiff\" is optimal for analysis and \".png\" images will be generated as
81 processed images.")
82 //*** End of Input Parameters Section ***//
83
84
85 //*** READ IMAGES, PREPARE FILE SYSTEM FOR OUTPUT CSV FILES ***//
86 //*** READ in images into a vector ***//
87 set( imageformat = IN_image_format )
88 // Find the format of input images
89 If(at("p", IN_image_format, 1))
90     Glob(filepath & "/*.p*", ignorecase=yes)
91     // Only compressed png or jpg or bmp files will be read into the Acapella system
92 End()
93 If(at("t", IN_image_format, 1))
94     Glob(filepath & "/*.tif*", ignorecase=yes)
95     // Only tif/tiff files will be read into the Acapella system
96 End()
97
98 // Display a warning message in case no image can be found in the directory
```

Additional File 2

```
99 If(files.size == 0)
100     Warning("No image has been found - please check the image directory as well as the selected image format!")
101     // Finish analysis as no image has been loaded
102 Else()
103     If(!SmallCallose && !LargeCallose)
104         Warning("At least one type of callose deposits should be selected - please check your selection for detecting callose deposits!")
105         // Finish analysis as no image has been loaded
106     Else()
107         /*** Prepare a Result folder for saving CSV results and processed images
108         set(IMG_Name = " ")
109         Set(current_Date = __date__)
110
111         /*** Prepare output Result folder
112         set( namelength = length( files[0] ) )
113         set( pathname = substr( files[0], 1, at("/", files[0], -1)))
114
115         Set(CreateDir = pathname & current_Date & "_Results")
116         Set(Output_File = current_Date & "_Overall_Results")
117         Delete(FileInfo)
118         FileInfo(CreateDir)
119
120         If(!fileinfo.exists) // Make a directory for CSV files
121             Mkdir(CreateDir, 555) // Set the directory attributes - only required in case we run it on Linux machine
122         End()
123
124         /*** Prepare overall output CSV file - average callose size, intensity, spreading network
125         printfopen(CreateDir & "/" & Output_File & ".csv")
126         Printf("Image_Name#")
127         Printf("Callose_Number#")
128         Printf("Average_Callose_Size#")
129         Printf("Average_Callose_Intensity#")
130         If(Callose_Network_Trigger) // If callose network is required
131             set(Network_Length = 0)
132             Printf("Calculated_Spreading_Length#")
133         End()
134         Printf("Total_Spreading_Callose_area#\n")
135         printfopen()
136         /*** End of preparing the CSV file for saving overall output results
137
138         /*** Start to process input images in the vector ***/
139         foreach(0..(files.size-1), file_counter)
140
141             Printf("Start analysing image %s \n", files[file_counter])
142
143             /*** Start to prepare CSV files for every processed image ***/
144             Delete(IMG_Name)
145             // Start to produce the output csv
146             set(IMG_Name = substr( files[file_counter], at("/", files[file_counter], -1) + 1))
147             Set(Experiment_Name = substr(IMG_Name, -1, (at(".", IMG_Name, -1) + 1)))
```

Additional File 2

```
148
149 // Prepare output csv for every callose
150 printfopen(CreateDir & "/" & Experiment_Name & "_Callose_Results"& ".csv")
151 Printf("Image_Name#")
152 Printf("Callose_Index#")
153 Printf("Size#")
154 Printf("Circularity#")
155 Printf("Intensity#\n")
156 printfopen()
157 /** Finish preparing the output CSV file
158
159 //**** Step Zero: pre-process images ****//
160 Delete(Source_Path)
161 set(Source_Path = files[file_counter]) // one channel image
162 ReadImage(Source_Path) // Read image into the buffer
163 set(Callose_IMG = image)
164 HsvSplit(Callose_IMG)
165 Set(Callose_Value = value) // Only use value planes during the image analysis
166
167
168 /// ANALYSIS STARTS FROM THIS LINE ONWARDS ///
169 //*** STEP ONE: Detect ROI ***//
170 //*** Step 1.1: Find ROI using Edge Detection (Gradient) and Convolution ***//
171 //Transforms the image into its gradient
172 Gradient(image=Callose_Value, sobel=yes, directional= 0)
173 Set(Gradient_IMG = image)
174 expand(5, 0.25, image=Gradient_IMG) // Enhance the processed image
175 Set(Processed_IMG = image) // Process the image with Gradient
176 set(convolutionkernel=toimage(vec(1,4,16,4,1, 6,24,36,24,6, 1,4,16,4,1),5,3).image) // build up a convolution kernel
177 set(convolutionkernelfactor=convolutionkernel.sum)
178 convolution(image = Processed_IMG)
179 Set(Convolution_IMG = image) // Process images with a convolution filter, so that the sharp edge can be soften
180
181 // Use bright mask to find the central part of callose objects
182 Bright_Mask(Callose_Value, 5)
183 // using adaptive thresholding
184 ThresholdXX(TuneTH=1.25, Image=result)
185 Mask(Threshold=Threshold * 3.75, Image=result)
186 Mask2Stencil()
187 Stencil2Objects()
188 FillObjects()
189 CalcArea()
190 ObjectFilter(area > 5 && area < 2500)
191 CalcIntensity(Image=Callose_Value, Total= yes)
192 CalcAttr(AVG_Intensity, (intensity * 255)/area)
193 Set(Intensity_Threshold = objects.AVG_Intensity.mean)
194 object_contrast_general(reference = Callose_Value, ContrastDef = "WithoutCyto")
195 ObjectFilter(contrast > 0.25 || AVG_Intensity > Intensity_Threshold * 1.25)
196 set(ROI_Initial_BM=Objects)
```

Additional File 2

```
197 // Initial ROI objects have been detected
198
199 // Start to detect mask based on Gradient image
200 // using adaptive thresholding
201 If(Convolution_IMG.mean / Convolution_IMG.median > 3.25) // image contains big intensity difference
202     ThresholdXX(TuneTH=1.25, Image=Convolution_IMG)
203     Mask(Threshold=Threshold * 6.5, Image=Convolution_IMG)
204 Else() // image has relatively equally spreading intensity values
205     ThresholdXX(TuneTH=1.25, Image=Convolution_IMG)
206     Mask(Threshold=Threshold * 6.25, Image=Convolution_IMG)
207
208 End()
209
210 // Start to build up initial ROI objects list
211 Mask2Stencil()
212 Stencil2Objects()
213 FillObjects()
214 CalcArea()
215 CalcIntensity(Image=Callose_Value, Total= yes)
216 CalcAttr(AVG_Intensity, (intensity * 255)/area)
217 Set(Intensity_Threshold = objects.AVG_Intensity.mean)
218 ObjectFilter(AVG_Intensity > Intensity_Threshold * 0.525 && area > 5)
219 object_contrast_general(reference = Callose_Value, ContrastDef = "WithoutCyto")
220 ObjectFilter(contrast > 0.25 || AVG_Intensity > Intensity_Threshold * 1.25)
221 set(ROI_Initial_Con=Objects)
222 // Initial ROI objects have been detected
223
224 //*** Step 1.2: Detect ROI using Watershed ***//
225 // This step can provide regions with bright pixels
226 // using adaptive thresholding
227 If(Callose_Value.mean / Callose_Value.median > 1.75) // image contains big intensity difference
228     ThresholdXX(TuneTH=1.25, Image=Callose_Value)
229     Mask(Threshold=Threshold * 1.95, Image=Callose_Value)
230 Else() // a dark or clear image
231     ThresholdXX(TuneTH=1.25, Image=Callose_Value)
232     Mask(Threshold=Threshold * 2.15, Image=Callose_Value)
233
234 End()
235 Mask2Stencil()
236 Stencil2Objects()
237
238 // Use watershed method - this step is not necessary in case the input callose images are clear
239 Maximums(2, Mask=objects.body.mask, Image=Callose_Value)
240 Set(Points=Stencil2Objects(Stencil=Maximums).Objects)
241 CalcWatershed(body, Objects=Points, Basins=objects.body, Image=Callose_Value, IntensityConstraintsMode=-1, MinBLC=0.05)
242 Stencil2Objects(Objects.watershed)
243 CalcArea()
244 object_contrast_general(reference = Callose_Value, ContrastDef = "WithoutCyto")
245 // get rid of small callose signals - there is no deposition
246 CalcIntensity(Image=Callose_Value, Total= yes)
247 CalcAttr(AVG_Intensity, (intensity * 255)/area)
```

Additional File 2

```
246 Set(Intensity_Threshold = objects.AVG_Intensity.mean)
247 ObjectFilter(AVG_Intensity > Intensity_Threshold * 0.625 && contrast > 0.1 && area > 5)
248 ObjectFilter(contrast > 0.25 || AVG_Intensity > Intensity_Threshold * 1.25)
249 // Reassemble the area
250 Mask2Stencil(objects.body.mask.image)
251 Stencil2Objects()
252 set(ROI_Initial_Watershed=Objects)
253
254 //*** Step 1.3: Start to assemble the refined ROI ***//
255 //*** Step 1.3.1: Assemble the ROI objects
256 And(ROI_Initial_Watershed.body.mask.image, image = ROI_Initial_Con.body.mask.image)
257 Or(ROI_Initial_BM.body.mask.image, image = image)
258 // boolean operation "and" with initial the ROI mask
259 Mask2Stencil(image)
260 Stencil2Objects()
261 FillObjects()
262 CalcArea()
263 object_contrast_general(reference = Callose_Value, ContrastDef = "WithoutCyto")
264 ObjectFilter(contrast > 0.1 && area >= 5) // get rid of very low intensity and small objects
265 if(RemoveBorderObj)
266     RemoveBorderObjects(5)
267 End()
268 set(ROI_Refined_1 =Objects)
269 // Finish finding the ROI objects - bright pixels are retained if they can be detected by both
270 // the edge detection and the watershed
271
272 //*** Step 1.3.2: Remove very big vessel/noise/hair signals
273 ThresholdXX(TuneTH=1.25, Image=Callose_Value)
274 Mask(Threshold=Threshold * 4.5, Image=Callose_Value)
275 // Using thresholding to refine the edge detection of ROI objects
276 Mask2Stencil()
277 Stencil2Objects()
278 FillObjects()
279 CalcArea()
280 CalcArea(border, AutoRecalc=yes)
281 ObjectFilter(area > 325)
282 CalcWidthLength(upto100 = no)
283 object_contrast_general(reference = Callose_Value, ContrastDef = "WithoutCyto")
284 ObjectFilter(full_length > 325 || ((full_length > 25 && full_length < 125) && (half_width > 5.25 && half_width < 10) && contrast > 0.625))
285
286 // Filter based on shape and intensity
287 CalcIntensity(Image=Callose_Value, Total=no, objects=objects)
288 CalcAttr(Roundness, 4 * pi * (area / (border_area ^ 2)) * 100) // the Circularity field
289 // Small hair or objects
290 ObjectFilter((area < 925 && roundness > 30 && roundness < 65 && (intensity > Callose_Value.mean * 6.5 && contrast > 0.725)) || (area >= 925 && roundness < 30 && roundness
291 > 9.5 && (intensity > Callose_Value.mean * 7.25 || contrast > 0.725)))
292
293 Set(stencil = objects.body)
294 CalcErosion(-3, objects = objects, Stencil = stencil) // increasing 3 pixels
```

Additional File 2

```
295 RenameAttr(body = stencil_eroded)
296 Mask2Stencil(objects.body.mask.image)
297 Stencil2Objects()
298 Set(ROI_Refined_2 = objects)
299
300 // Assemble the ROI objects list
301 Xor(ROI_Refined_2.body.mask.image, image = ROI_Refined_1.body.mask.image)
302 Mask2Stencil(image)
303 Stencil2Objects()
304 CalcArea()
305 CalcWidthLength(Upto100=yes)
306 Set(ROI_Refined_3 = objects)
307
308 /*** Step 1.4: Start to remove high leaf vessel, mesophyll cells, and unsuitable objects
309 If(ROI_Refined_3.area.max > 625 || ROI_Refined_3.full_length.max > 35) // find bright vessel or big noisy signals
310     If(ROI_Refined_3.area.max > 2750 || ROI_Refined_3.full_length.max > 35)
311         // find very big/bright/long vessel or big noisy signals
312         // Divide the ROI objects according to their size/shape
313         ObjectFilter(area > 125 || full_length > 32.5, objects = ROI_Refined_3)
314         Set(ROI_Big_INI = objects)
315         ObjectFilter(area <= 125 && full_length <= 32.5, objects = ROI_Refined_3)
316         Set(ROI_Small_INI = objects)
317
318     /*** Step 1.4.1: Find refined small ROI ***/
319     CalcIntensity(Image=Callose_Value, Total=no, objects=ROI_Small_INI)
320     ObjectFilter(area > 5 && intensity > Callose_Value.mean * 1.5)
321     Set(ROI_Small_REF = objects)
322
323     /*** Step 1.4.2: Find refined big ROI ***/
324     ThresholdXX(TuneTH=1.25, Image=Callose_Value)
325     Mask(Threshold=Threshold * 2.15, Image=Callose_Value)
326     Mask2Stencil()
327     Stencil2Objects()
328     And(ROI_Big_INI.body.mask.image, image = objects.body.mask.image)
329     Mask2Stencil(image)
330     Stencil2Objects()
331     CalcArea()
332     CalcWidthLength(Upto100=yes)
333
334     If(!LargeCallose) // Large Callose deposits are not required
335         ObjectFilter(full_length > 0 && full_length < 25) // get rid of very big callose deposits
336     Else()
337         // Large callose deposits are rerquired
338         set(ROI_Big_INI_1 = objects)
339         // Divide the ROI objects list to handle very big objects
340         ObjectFilter(!(full_length <= 21.5 && area <125), objects = ROI_Big_INI_1)
341         set(ROI_Big_INI_1_Big = objects)
342         ObjectFilter(full_length <= 21.5 && area <125, objects = ROI_Big_INI_1)
343         set(ROI_Big_INI_1_Small = objects)
```

Additional File 2

```
344
345     ThresholdXX(TuneTH=1.25, Image=Callose_Value)
346     Mask(Threshold=Threshold * 2.5, Image=Callose_Value)
347     // Using thresholding to refine the edge detection of ROI objects
348     Mask2Stencil()
349     Stencil2Objects()
350     And(ROI_Big_INI_1_Big.body.mask.image, image = objects.body.mask.image)
351     Set(ROI_Big_INI_IMG = image)
352     // Add relatvie small objects
353
354     Or(ROI_Big_INI_1_small.body.mask.image, image = ROI_Big_INI_IMG)
355     Mask2Stencil(image)
356     Stencil2Objects()
357     CalcArea()
358     CalcWidthLength(Upto100=yes)
359     ObjectFilter(area > 2.5 && !(full_length >375 && half_width > 12.5))
360     CalcIntensity(Image=Callose_Value, Total=no, objects=objects)
361     ObjectFilter(area > 5 && intensity > Callose_Value.mean * 1.5)
362 End()
363 Set(ROI_Big_REF = objects)
364
365 // Not checking objects overlapping
366 // Assemble small ROI and refined big ROI objects
367 Or(ROI_Small_REF.body.mask.image, image=ROI_Big_REF.body.mask.image)
368 Mask2Stencil(image)
369 Stencil2Objects()
370 CalcArea()
371 CalcWidthLength(Upto100=yes)
372 Set(ROI_Refined_INI = objects)
373
374 Else() // Cannot find very big ROI objects
375 // Divide objects list into two groups
376 ObjectFilter(area > 125, objects = ROI_Refined_3)
377 Set(ROI_Big_INI = objects)
378 ObjectFilter(area <= 125, objects = ROI_Refined_3)
379 Set(ROI_Small_INI = objects)
380
381 //*** Step 1.4.1: Find refined small ROI ***//
382 CalcIntensity(Image=Callose_Value, Total=no, objects=ROI_Small_INI)
383 ObjectFilter(area > 5 && intensity > Callose_Value.mean * 1.5)
384 Set(ROI_Small_REF = objects)
385
386 //*** Step 1.4.2: Find refined Big ROI Objects ***//
387 ThresholdXX(TuneTH=1.25, Image=Callose_Value)
388 Mask(Threshold=Threshold * 1.75, Image=Callose_Value)
389 Mask2Stencil()
390 Stencil2Objects()
391 And(ROI_Big_INI.body.mask.image, image = objects.body.mask.image)
392 Mask2Stencil(image)
```


Additional File 2

```
393         Stencil2Objects()
394         CalcArea()
395         CalcWidthLength(Upto100=yes)
396         ObjectFilter(area > 2.5 && !(full_length > 325 && half_width > 10))
397         CalcIntensity(Image=Callose_Value, Total=no, objects=objects)
398         ObjectFilter(area > 5 && intensity > Callose_Value.mean * 1.5)
399         Set(ROI_Big_REF = objects)
400
401         // Not checking objects overlapping
402         // Assemble small ROI and refined big ROI objects
403         Or(ROI_Small_REF.body.mask.image, image=ROI_Big_REF.body.mask.image)
404         Mask2Stencil(image)
405         Stencil2Objects()
406         CalcArea()
407         CalcWidthLength(Upto100=yes)
408         Set(ROI_Refined_INI = objects) // Use "ROI_refined" as the output objects list for this stage
409     End()
410 Else()
411     Set(ROI_Refined_INI = ROI_Refined_3)
412 End()
413
414 // Get rid of small objects if required - certain experiments only need to detect big callose deposition
415 If(!SmallCallose)
416     ObjectFilter(area > 10 && (full_length > 2.5 || half_width > 1.25), objects = ROI_Refined_INI)
417     set(ROI_Refined_INI = objects)
418 End()
419
420 //*** Step 1.4.3: categorise ROI Objects ***//
421 ObjectFilter(!(full_length > 10 && half_width < 1.5), objects = ROI_Refined_INI)
422 object_contrast_general(reference = Callose_Value, ContrastDef = "WithoutCyto")
423 CalcArea()
424 CalcArea(border)
425 CalcAttr(Roundness, 4 * pi * (area / (border_area ^ 2)) * 100) // the Circularity field
426 CalcIntensity(Image=Callose_Value, Total= no)
427 CalcWidthLength(Upto100= no)
428 set(TMP_ROI_Refined_INI = objects)
429
430 // Leave vessel or long fused callose
431 ObjectFilter((intensity < 0.8 * Callose_Value.max && intensity > 0.2 * Callose_Value.max) && ((full_length > 20 && half_width < 7.25 && contrast < 0.65) || (full_length > 50 &&
432 half_width < 10.25 && contrast < 0.75) || (full_length > 92.5 && half_width < 13.25 && contrast < 0.85)), objects = TMP_ROI_Refined_INI)
433 Set(ROI_Refined_Vessel = Objects)
434
435 // Remove Vessel
436 CalcStat("mean", Stencil = TMP_ROI_Refined_INI.body, Image = ROI_Refined_Vessel.body.mask.image, AttrName="overlapped", objects=TMP_ROI_Refined_INI)
437 ObjectFilter(overlapped == 0, objects = objects )
438 ObjectFilter(roundness > 30 || (contrast > 0.35 && intensity > Callose_Value.max * 0.5) || area > 1250)
439 ObjectFilter(!(intensity > Callose_Value.max * 0.75 && contrast > 0.825))
440 ObjectFilter(!(intensity > Callose_Value.max * 0.725 && contrast > 0.775 && half_width > 8.5 && (roundness < 70 && roundness > 35) && area > 375 && full_length > 25))
441 // Remove Hairs
```

Additional File 2

```
442         Set(ROI_Refined = objects)
443         // Include fused Callose and seperated callose
444     **** Finish Detecting ROI and generating ROI objects list****
445
446
447     **** STEP TWO: Start to Detect Callose Deposits Based on ROI ****
448     ** Step 2.1: Detect centres of callose candidates based on ROI
449     // Detect Zone image and centres of Callose Deposits
450     // Finds the maximum intensity pixels within ROI.
451     if(ROI_BIG_REF.area.mean > 150) // Contain very large ROI objects
452         Maximums(Distance= 2, Mask=ROI_Refined.Body, Image=Callose_IMG) // Scan 2-pixel regions
453     Else()
454         Maximums(Distance= 3, Mask=ROI_Refined.Body, Image=Callose_IMG) // Scan 3-pixel regions
455     End()
456     // do not use zone image as it generates imprecise centres
457
458     Set(stencil = maximums)
459     Stencil2Objects()
460     CalcIntensity(Image=Callose_Value, Total= no)
461     Set(Intensity_Threshold = Callose_Value.mean)
462     ObjectFilter(Intensity > Intensity_Threshold * 2.25) // Only retain points with very high intensity values
463     Set(Callose_Cadidates_Centre = objects)
464
465     // Start to combine some of the miss separated callose centres
466     Set(stencil = objects.body)
467     // pi * 3.5 * 3.5 = 38.5 - the size of small callose deposits is bigger than 40 based on median value
468         CalcErosion(-1, objects = objects, Stencil = stencil) // increasing 1 pixel
469     // Small callose objects are relatively small
470
471     RenameAttr(body = stencil_eroded)
472     Mask2Stencil(objects.body.mask.image)
473     Stencil2Objects()
474     CalcIntensity(Image=Callose_Value, Total= yes)
475     CalcArea()
476     CalcAttr(AVG_Intensity, intensity/area) // Apply local thresholding
477     Set(Intensity_Threshold = Callose_Value.mean)
478     ObjectFilter(AVG_Intensity > Intensity_Threshold * 2.5) // get as many centres as possible
479     CalcWidthLength(Upto100=yes) // get the centres of the body
480     Set(Callose_Cadidates_Centre_refined = objects) // Refined detection on callose centres!!!!
481
482     ** Step 2.1.1: Detect centres of very big callose candidates based on ROI
483     If(ROI_BIG_REF.area.mean > 150)
484         // image contains very big callose
485         // Apply global thresholding
486             ThresholdXX(TuneTH=1.25, Image=Callose_Value)
487             Mask(Threshold=Threshold * 5.525, Image=Callose_Value)
488     Else()
489             ThresholdXX(TuneTH=1.25, Image=Callose_Value)
490             Mask(Threshold=Threshold * 5.325, Image=Callose_Value)
```

Additional File 2

```
491         End()
492         Mask2Stencil()
493         Stencil2Objects()
494         CalcArea()
495         CalcWidthLength(Upto100=yes)
496         ObjectFilter(area > 75 && half_width > 3.75 && full_length > 9.25)
497         ObjectFilter(!(area > 1250 && half_width > 8.25 && full_length > 95))
498         set(Big_Callose_Area = objects)
499
500     If(Big_Callose_Area.@count > 0)
501         CalcStat("mean", Stencil = Callose_Cadidates_Centre_refined.body, Image = Big_Callose_Area.body.mask.image, AttrName="overlapped",
502 objects=Callose_Cadidates_Centre_refined)
503         ObjectFilter(overlapped == 0, objects = objects )
504         Set(Callose_Cadidates_Centre_refined_Small = objects) // Finish filtering the centres for big callose
505
506         Maximums(Distance= 4.5, Mask=Big_Callose_Area.Body, Image=Callose_IMG) // Scan 4.5-pixel regions
507         Set(stencil = maximums)
508         Stencil2Objects()
509         CalcIntensity(Image=Callose_Value, Total= no)
510         Set(Intensity_Threshold = Callose_Value.mean)
511         ObjectFilter(Intensity > Intensity_Threshold * 2.15) // Only retain points with very high intensity values
512         Set(Callose_Cadidates_Centre_refined_Big = objects)
513
514         Set(stencil = Callose_Cadidates_Centre_refined_Small.center_spot)
515         Stencil2Objects()
516         Set(Callose_Cadidates_Centre_refined_Small_centres = objects)
517         AddObjects(Callose_Cadidates_Centre_refined_BIG, objects=Callose_Cadidates_Centre_refined_Small_centres, CheckOverlap=no, DeleteGeometry=no)
518
519         Set(stencil = objects.body)
520         CalcErosion(-1, objects = objects, Stencil = stencil) // increasing 1 pixel
521         // Small callose objects are relatively small
522         RenameAttr(body = stencil_eroded)
523         Mask2Stencil(objects.body.mask.image)
524         Stencil2Objects()
525         CalcWidthLength(Upto100=yes) // get the centres of the body
526         Set(Callose_Cadidates_Centre_refined = objects) // Refined detection on callose centres!!!!
527     End()
528     // Finish refining the callose centres
529
530
531     /** Step 2.2: Finalise callose candidates based on refined the centres
532     Or(Callose_Cadidates_Centre_refined.body.mask.image, image = ROI_Refined.body.mask.image)
533     Mask2Stencil(image)
534     // Replace very small callose with eroded callose centres!!! important for rectifying the detected objects
535     ObjectsFromCenters(Centers=Callose_Cadidates_Centre_refined.center_spot, RestrictiveStencil=stencil, image=Callose_Value, MinimumArea= 5)
536     ControlBreakingLines_A(1) // Split as many objects as possible
537     FillObjects()
538     CalcArea(objects=objects, AutoRecalc=yes)
539     ObjectFilter(area > 5)
```

Additional File 2

```
540 object_contrast_general(reference = Callose_Value, ContrastDef = "WithoutCyto")
541 CalcIntensity(Image=Callose_Value, Total= yes)
542 CalcAttr(AVG_Intensity, (intensity * 255)/area) // Apply local thresholding
543 Set(Intensity_Threshold = Callose_Value.mean)
544 ObjectFilter(AVG_Intensity > Intensity_Threshold * 3.5 && contrast > 0.025) // get as many centres as possible
545 CalcWidthLength(Upto100=yes)
546 CalcAttr(WLRatio, half_width * 2 / full_length)
547 ObjectFilter(!(area > 25 && WLRatio < 0.165))
548 ObjectFilter(!(area > 225 && WLRatio < 0.5 && AVG_Intensity/255 > Callose_Value.max * 0.925 ))
549 Set(Callose_Candidates_final = objects)
550 // Finish finding the candidates of callose deposits
551
552
553 /** Step 2.3: Perform Edge Detection on Callose Candidates
554 // Divide Callose candidates according to their sizes
555 Set(Callose_Initial = Callose_Candidates_final)
556 If(!LargeCallose) // pi * 3.5 * 3.5 = 38.5
557     ObjectFilter(area > 125 && area <= 500, objects = Callose_Candidates_final)
558 Else()
559     ObjectFilter(area > 125 && area <= 1250, objects = Callose_Candidates_final)
560     // some experiments can generate very big callose
561 End()
562 Set(Callose_Initial_Big = objects)
563
564 If(!SmallCallose)
565     ObjectFilter(area <= 125 && area >= 10, objects = Callose_Candidates_final)
566 Else()
567     ObjectFilter(area <= 125 && area >= 5, objects = Callose_Candidates_final)
568 End()
569 Set(Callose_Initial_Small = objects)
570 //// Finish categorising callose candidates according to their size
571
572 /** Step 2.3.1: Perform Edge-based Detection on Big Callose Deposition and Remove Experimental Errors
573 EdgeDetection_BIG(Callose_Value, Callose_Candidates_Centre_refined, Callose_Initial_Big, ROI_Refined)
574 /**** Finish refining big callose deposits
575
576 /**** Step 2.3.2: Start to process small callose
577 // Start to refine the outline of small callose (big)
578 EdgeDetection_SMALL(Callose_Value, Callose_Candidates_Centre_refined, Callose_Initial_Small, ROI_Refined)
579 /**** Finish refining small callose deposits
580
581
582 /**** Step 2.3.3: Start to process small callose on plant vessel
583 // Start to refine the outline of small callose (big)
584 // In the future, another class can be added to detect callose with size below 50 pixel2
585 If(ROI_Refined_Vessel.intensity.mean > 0.25)
586 // image contains big intensity difference
587 // Apply global thresholding
588     ThresholdXX(TuneTH=1.25, Image=Callose_Value)
```

Additional File 2

```
589         Mask(Threshold=Threshold * 3.85, Image=Callose_Value)
590     Else()
591         ThresholdXX(TuneTH=1.25, Image=Callose_Value)
592         Mask(Threshold=Threshold * 3.75, Image=Callose_Value)
593     End()
594     Mask2Stencil()
595     Stencil2Objects()
596     set(Brightest_Area_Ini_Vessel = objects)
597
598     // Refine the ROI on plant vessel
599     And(Brightest_Area_Ini_Vessel.body.mask.image, image = ROI_Refined_Vessel.body.mask.image)
600     Mask2Stencil(image)
601     Stencil2Objects()
602     CalcArea()
603     CalcWidthLength(Upto100= no)
604     CalcAttr(WL_Ratio, full_length/(half_width * 2))
605     CalcIntensity(Image=Callose_Value, Total= no)
606     object_contrast_general(reference = Callose_Value, ContrastDef = "WithoutCyto")
607     ObjectFilter((WL_Ratio <= 3.75 && half_width > 4.5 && full_length > 27.5) || area < 325 || WL_Ratio > 7.5 || (WL_Ratio > 4.5 && area > 975) || (WL_Ratio > 3.75 && half_width
608 < 7.5 && full_length > 27.5 ))
609     ObjectFilter(!(area > 425 && intensity > Callose_Value.max * 0.775 && contrast > 0.725))
610     set(Callose_Vessel_INI = objects)
611
612     if(Callose_Vessel_INI.area.mean > 32.5) // Contain large ROI objects
613         Maximums(Distance= 4, Mask=Callose_Vessel_INI.Body, Image=Callose_IMG) // Scan 3-pixel regions
614     Else()
615         Maximums(Distance= 3, Mask=Callose_Vessel_INI.Body, Image=Callose_IMG) // Scan 2-pixel regions
616     End()
617     // do not use zone image as it generates imprecise centres
618
619     Set(stencil = maximums)
620     Stencil2Objects()
621     CalcIntensity(Image=Callose_Value, Total= no)
622     Set(Intensity_Threshold = ROI_Refined_Vessel.intensity.mean)
623     ObjectFilter(Intensity > Intensity_Threshold * 1.25) // Only retain points with very high intensity values
624     Set(Callose_Vessel_Centres = objects)
625
626     // Start to detect callose on vessel
627     ObjectsFromCenters(Centers=Callose_Vessel_Centres.body, RestrictiveStencil=Callose_Vessel_INI.body, image=Callose_Value, MinimumArea= 5)
628     ControlBreakingLines_A(1)
629     Set(stencil = objects.body)
630     Stencil2Objects()
631     CalcArea()
632     CalcIntensity(Image=Callose_Value, Total= no)
633     ObjectFilter(Intensity > Intensity_Threshold && area > 2.5)
634     object_contrast_general(reference = Callose_Value, ContrastDef = "WithoutCyto")
635     ObjectFilter(contrast > 0.325 || Intensity > Intensity_Threshold * 1.525 )
636     ObjectFilter(!(intensity > Callose_Value.max * 0.9 && contrast > 0.65 && area > 125))
637     ObjectFilter(!(intensity > Callose_Value.max * 0.825 && contrast > 0.625 && area > 275))
```

Additional File 2

```
638 CalcWidthLength(Upto100=yes)
639 ObjectFilter(half_width >= 1 && full_length >= 2)
640 CalcAttr(WL_Ratio, full_length/(half_width * 2))
641 ObjectFilter(WL_Ratio <= 3.5)
642 ObjectFilter(WL_Ratio <= 2.75 || (Intensity_Threshold * 1.65 && contrast > 0.35)) // exceptional cases
643 // get as many callose deposits as possible
644 Set(Callose_Vessel_Refined = objects)
645
646
647 // Refine the centre detection with vessel centers
648 Set(stencil = Callose_Vessel_Refined.center_spot)
649 Stencil2Objects()
650 CalcErosion(-2, objects = objects, Stencil = stencil) // increasing 2 pixels
651 RenameAttr(body = stencil_eroded)
652
653 // Add vessel centres to the callose center objects list
654 Or(objects.body.mask.image, image = Callose_Candidates_Centre_refined.body.mask.image)
655 Mask2Stencil(image)
656 Stencil2Objects()
657 CalcWidthLength(Upto100=yes)
658 Set(Callose_Candidates_Centre_final = objects)
659 /*** Finish refining callose centers with centres on vessel
660
661 /*** Step 2.4: Start to assemble the detected big/small callose objects list
662 Or(Callose_Small_refined.body.mask.image, image = Callose_Big_refined.body.mask.image)
663 Or(image, image = Callose_Vessel_Refined.body.mask.image)
664 Mask2Stencil(image)
665 Stencil2Objects()
666 Set(Callose_Candidates_refined = objects)
667
668 // Refine object centres for detected callose objects
669 ObjectsFromCenters(Callose_Candidates_Centre_final.center_spot, Callose_Candidates_refined.body, Image=Callose_Value, MinimumArea= 4) // still use 4 pixel square as the
670 Minimum Area
671 ControlBreakingLines_A(1) // Split objects as many as possible
672
673 Set(stencil = objects.body)
674 Stencil2Objects()
675 CalcBorder(AutoRecalc=yes, objects=objects)
676 CalcArea(AutoRecalc=yes, objects=objects)
677 CalcArea(border, AutoRecalc=yes, objects=objects)
678
679 // Get rid of small callose if required
680 If(!SmallCallose)
681     ObjectFilter(area >= 25 && border_area >= 8)
682 Else()
683     ObjectFilter(area >= 5 && border_area >= 4)
684 End()
685
686 if(RemoveBorderObj)
```

Additional File 2

```
687         RemoveBorderObjects(25)
688     End()
689
690     // Start to screen out fake callose objects
691     CalcWidthLength(Upto100=yes)
692     ObjectFilter(half_width > 1 && full_length >= 2 && full_length < 42.5)
693     CalcIntensity(Image=Callose_Value, Total=No, CalcStdDev=no)
694     CalcIntensity(border, Callose_Value, Total=No, CalcStdDev=no)
695
696     CalcAttr(Intensity_Diff, (intensity - border_intensity)/(intensity + border_intensity))
697     set(Intensity_Threshold = objects.intensity.mean)
698     ObjectFilter(intensity_Diff > 0 || intensity > Callose_Value.mean * 3.25)
699     ObjectFilter((area >= 32.5 && (Intensity_Diff > 0.03 || intensity > Intensity_Threshold * 1.125)) || (area < 32.5 && (Intensity_Diff > 0.02 || intensity > Intensity_Threshold)))
700     // Use global intensity threshold
701     object_contrast_general(reference = Callose_Value, ContrastDef = "WithoutCyto")
702     ObjectFilter(contrast > 0.275 || (area < 50 && contrast > 0.25) || (intensity > Callose_Value.mean * 3.5 && Intensity_Diff > 0.025 && contrast > 0.185))
703     // Filter based on contrast, intensity, and the intensity of callose centres
704
705     If(Callose_Value.mean > 0.125)
706         ObjectFilter(Intensity_Diff > 0.025 || intensity > Callose_Value.mean * 3.05 || contrast > 0.25)
707     Else()
708         ObjectFilter(Intensity_Diff > 0.025 || intensity > Callose_Value.mean * 2.75 || contrast > 0.225)
709     End()
710     Set(Intensity_Threshold = objects.Intensity.mean) // Use local intensity threshold
711
712     // Filter based on contrast and intensity
713     If(LargeCallose)
714         ObjectFilter(Intensity > Intensity_Threshold * 0.725 || contrast > 0.175)
715     Else()
716         ObjectFilter(Intensity > Intensity_Threshold * 0.625 || contrast > 0.155)
717         ObjectFilter(!(area > 50 && (contrast < 0.525 || Intensity > Intensity_Threshold * 1.35)))
718     End()
719
720     If(!SmallCallose)
721         ObjectFilter(area > 20 || contrast > 0.275 || Intensity > Intensity_Threshold * 0.95)
722     Else()
723         ObjectFilter(area > 10 || contrast > 0.225 || Intensity > Intensity_Threshold * 0.75)
724     End()
725
726     CalcAttr(Roundness, 4 * pi * (area / (border_area ^ 2)) * 100) // the Circularity field
727     CalcAttr(WL_Ratio, full_length/(half_width * 2))
728     ObjectFilter(WL_Ratio < 4) // get rid of flat objects
729
730     // Filter based on roundness and width and length
731     If(LargeCallose)
732         ObjectFilter((WL_Ratio < 3.85 && Roundness > 47.5) || area < 40)
733         // some strange shape can be detected based on objects split due to centres cannot be separated
734         ObjectFilter((WL_Ratio < 2.5 && Roundness > 110) || contrast >= 0.225 || area < 40 || intensity > Callose_Value.mean * 3.75)
735     Else()
```

Additional File 2

```
736         ObjectFilter(WL_Ratio < 3.75 && Roundness > 50 || area < 47.5)
737         ObjectFilter((WL_Ratio < 2.5 && Roundness > 110) || contrast >= 0.25 || area < 40 || intensity > Callose_Value.mean * 4.25)
738     End()
739
740     // Exceptional case
741     If(!LargeCallose)
742         ObjectFilter(!(full_length > 13.75 || half_width > 5 || (WL_Ratio > 2 && area > 125)))
743     End()
744     ObjectFilter(!(WL_Ratio > 2.25 && intensity < Callose_Value.mean))
745     ObjectFilter(!(WL_Ratio > 2.25 && area > 50 && intensity < Callose_Value.mean * 4.25) || !(WL_Ratio > 2.25 && area > 50 && contrast < 0.275))
746     ObjectFilter(!(area > 75 && (Intensity_Diff < 0.05 && intensity < Callose_Value.mean * 4.25)))
747     Set(Callose_refined_final = objects)
748
749
750
751     **** Optional Functions ****
752     *** START TO DETECT SPREADING CALLOSE NETWORKS ***
753     If(Callose_Network_Trigger && Callose_refined_final.@count > 1)
754     **** Step Three: Start to calculate the callose network ****
755         Set(DrawLine_Callose_IMG = Callose_Value)
756         set(IM_Height_Expanded = Callose_Value.height)
757         set(IM_Width_Expanded = Callose_Value.width)
758         Blank(IM_Width_Expanded, IM_Height_Expanded, 0)
759         set(Callose_Network_IMG = image) // Create a blank image
760
761         If(Network_Threshold == 25) // the default value, the distance threshold is calculated below
762             If(Callose_refined_final.full_length.mean > Callose_refined_final.full_length.median)
763                 set(Network_Threshold = (Callose_refined_final.full_length.mean + Callose_Big_refined.full_length.mean +
764 Callose_Small_refined.full_length.mean))
765             If(Network_Threshold < 20 || Network_Threshold > 27.5)
766                 set(Network_Threshold = 22.5)
767             End()
768         Else()
769             set(Network_Threshold = (Callose_refined_final.full_length.median + Callose_Big_refined.full_length.median +
770 Callose_Small_refined.full_length.median))
771             If(Network_Threshold < 20 || Network_Threshold > 27.5)
772                 set(Network_Threshold = 22.5)
773             End()
774         End()
775     End()
776
777     *** Step 3.1: Detect the callose network
778     Foreach(0..(Callose_refined_final.@count - 1), Obj_Counter)
779         Set(Current_Centre_Spot = Obj_Counter)
780         set(Obj_Distance_Counter = 0)
781         Set(current_X_Axis = Callose_refined_final.center_spot.x.target[Current_Centre_Spot])
782         Set(current_Y_Axis = Callose_refined_final.center_spot.y.target[Current_Centre_Spot])
783
784         ///Determine the area of the detection
```


Additional File 2

```
785         Mask2Stencil(Callose_refined_final[Current_Centre_Spot].body.image)
786         Stencil2Objects()
787         CalcErosion(Network_Threshold * -1, objects = objects, Stencil = stencil) // increasing 1 pixel
788         RenameAttr(body = stencil_eroded)
789         Set(Temp_Callose_Scan_Obj = Callose_refined_final)
790
791         CalcStat("mean", Stencil = Temp_Callose_Scan_Obj.body, Image = objects.body.mask.image, AttrName="overlapped", objects=Temp_Callose_Scan_Obj)
792         ObjectFilter(overlapped != 0, objects = objects )
793         Set(Temp_Callose_Scan_Obj = objects)
794
795         Foreach(0..(Temp_Callose_Scan_Obj.@count - 1), Obj_Distance_Counter)
796             If(Temp_Callose_Scan_Obj.center_spot.x.target[Obj_Distance_Counter] != current_X_Axis &&
797 Temp_Callose_Scan_Obj.center_spot.y.target[Obj_Distance_Counter] != current_Y_Axis)
798                 // Calculate the distance
799                 // Set(Callose_Distance = sqrt((current_X_Axis - Temp_Callose_Scan_Obj.center_spot.x.target[Obj_Distance_Counter])^2 +
800 (Temp_Callose_Scan_Obj.center_spot.y.target[Obj_Distance_Counter] - current_Y_Axis)^2))
801
802                 DrawLine(point(current_X_Axis, current_Y_Axis), point(Temp_Callose_Scan_Obj.center_spot.x.target[Obj_Distance_Counter],
803 Temp_Callose_Scan_Obj.center_spot.y.target[Obj_Distance_Counter]), color="white", image=DrawLine_Callose_IMG)
804                 Set(DrawLine_Callose_IMG = image)
805
806                 DrawLine(point(current_X_Axis, current_Y_Axis), point(Temp_Callose_Scan_Obj.center_spot.x.target[Obj_Distance_Counter],
807 Temp_Callose_Scan_Obj.center_spot.y.target[Obj_Distance_Counter]), "intensity", 215, "yellow", image=Callose_Network_IMG)
808                 // generate LINKS with DETERMINED coordinates
809                 OR(image, image = Callose_Network_IMG)
810                 set(Callose_Network_IMG = image)
811             End()
812         End()
813     End()
814
815     /** Step 3.2: Calculate the area of the callose network
816     Mask2Stencil(Callose_Network_IMG)
817     Stencil2Objects()
818     CalcErosion(-1, objects = objects, Stencil = stencil) // increasing 1 pixel
819     RenameAttr(body = stencil_eroded)
820     Mask2Stencil(objects.body.mask.image)
821     Stencil2Objects()
822     Set(Callose_Network = objects)
823
824     Mask2Stencil(Callose_Network.body.mask.image)
825     Stencil2Objects()
826     FillSmallHoles(150)
827     CalcWidthLength()
828     CalcArea()
829     CalcArea(border, objects = objects)
830     ObjectFilter(full_length > Network_Threshold * 1.75 && (half_width > 2.75 || full_length > Network_Threshold * 2.25) && area > 125)
831     Set(Callose_Network_refined = objects)
832
833     /** Step 3.3: Detect the skeleton of the network based on intensity of objects
```

Additional File 2

```
834         CalcSkeletonByIntensity(SkeletonType="cornerconnected", Image=Callose_Value, IntensityEvalParam=-oo, Objects=Callose_Network_refined)
835         set(stencil = objects.skeleton)
836         Stencil2Objects()
837         CalcArea(AutoRecalc=yes)
838         CalcIntensity(Image=Callose_Value, CalcStdDev=yes, Total=yes)
839
840         Mask2Stencil(objects.body.mask.image)
841         Stencil2Objects()
842         CalcErosion(-2, objects = objects, Stencil = stencil) // increasing 2 pixels
843         RenameAttr(body = stencil_eroded)
844         Mask2Stencil(objects.body.mask.image)
845         Stencil2Objects()
846         CalcArea()
847         CalcArea(border, objects = objects)
848         CalcWidthLength()
849         Set(Callose_Network_Skeleton = objects)
850     End()
851     /*** Callose Network Construction Finishes ***/
852
853
854     /*** Step Four: Start to assemble output fields ***/
855     // output generated images
856     set( namelength = length(Source_Path) )
857     set( corename = substr( Source_Path, 1, namelength - 4))
858
859     /** Step 4.1:Preview images if required
860     If(ShowIllustrations)
861         imageview(Callose_refined_final.body, "Detected_Callose", image=Callose_IMG, middlecolor=rgb(0x00,0xff,0x00))
862         If(Callose_Network_Trigger)
863             imageview(Callose_Network.body, "Callose_Network", image=Callose_IMG, middlecolor=rgb(0x00,0xff,0x00))
864             imageview(Callose_Network_Skeleton.body, "Callose_Network", image=Callose_IMG, middlecolor=rgb(0x00,0xff,0x00))
865         End()
866     End()
867
868     /** Step 4.2: Export result images
869     Gamma(1.0, image=Callose_IMG)
870     WriteImage(imagefile=CreateDir & "/" & Experiment_Name & "_1_Original_IMG.png", image=Image, imageformat="png")
871
872     Gamma(1.0, image=Callose_IMG)
873     CarryObjects(Callose_refined_final.body, image.max)
874     CarryObjects(Callose_refined_final.body, "rainbow")
875     WriteImage(imagefile=CreateDir & "/" & Experiment_Name & "_2_Detected_Callose_Obj.png", image=Image, imageformat="png")
876
877     set(stencil = Callose_refined_final.body)
878     Stencil2Objects()
879     // enlarge border stencil by 1 pixel for better visualisation
880     Gamma(1.0, image=Callose_IMG)
881     CarryObjects(objects.border, image.max)
882     CarryObjects(objects.border, "green")
```

Additional File 2

```
883 WriteImage(imagefile=CreateDir & "/" & Experiment_Name & "_3_Detected_Callose_Outline.png", image=Image, imageformat="png")
884
885 ObjectNumberOnImage(Callose_refined_final, "yellow", 14, image=Callose_IMG)
886 WriteImage(imagefile=CreateDir & "/" & Experiment_Name & "_4_Labelled_Callose.png", image=Image, imageformat="png")
887
888 If(Callose_Network_Trigger)
889     Gamma(1.0, image=Callose_IMG)
890     CarryObjects(Callose_Network.body, image.max)
891     CarryObjects(Callose_Network.body, "red")
892     WriteImage(imagefile=CreateDir & "/" & Experiment_Name & "_5_Callose_Network.png", image=Image, imageformat="png")
893
894     Gamma(1.0, image=Callose_IMG)
895     CarryObjects(Callose_Network_Skeleton.body, image.max)
896     CarryObjects(Callose_Network_Skeleton.body, "red")
897     WriteImage(imagefile=CreateDir & "/" & Experiment_Name & "_6_Spreading_Skeleton.png", image=Image, imageformat="png")
898 End()
899
900 /** Step 4.3: Export overall result CSV file
901 If(Callose_refined_final.@count > 1)
902     printfopen(CreateDir & "/" & Output_File & ".csv", yes)
903     Printf(Experiment_Name & "#")
904     Printf(Callose_refined_final.@count & "#")
905     Printf(Callose_refined_final.area.mean & "#")
906     Printf(Callose_refined_final.Intensity.mean*255 & "#")
907     // Network is required
908     If(Callose_Network_Trigger) // change as not ticking callose network will cause error
909         If(Callose_Network_refined.@count > 0)
910             Delete(Network_Length)
911             Set(Network_Length = ((Callose_Network_Skeleton.border_area.sum - Callose_Network_Skeleton.half_width.sum * 2)/2))
912             Printf(Network_Length & "#") // the calculated network length
913         Else()
914             Printf(nan & "#")
915         End()
916     End()
917     Printf(Callose_refined_final.area.sum & "#\n") // the measured area of callose deposits in pixels square
918     printfopen()
919 Else()
920     printfopen(CreateDir & "/" & Output_File & ".csv", yes)
921     Printf(Experiment_Name & "#")
922     Printf(0 & "#")
923     Printf(nan & "#")
924     Printf(nan & "#")
925     // Network is required
926     If(Callose_Network_Trigger)
927         Printf(nan & "#")
928     End()
929     Printf(nan & "#\n")
930     printfopen()
931 End()
```

Additional File 2

```
932         // Close the csv writing
933
934     /** Step 4.4: Start to export individual callose
935     set(Callose_Counter = 0)
936     If(Callose_refined_final.@count > 1)
937         Foreach(0..(Callose_refined_final.@count-1), Callose_Counter)
938             // Start to build up the output table
939             printfopen(CreateDir & "/" & Experiment_Name & "_Callose_Results"& ".csv", yes)
940             Printf(Experiment_Name & "#")
941             Printf((Callose_Counter + 1) & "#")
942             Printf(Callose_refined_final.area[Callose_Counter] & "#")
943             Printf(Callose_refined_final.Roundness[Callose_Counter]/100 & "#")
944             Printf(Callose_refined_final.Intensity[Callose_Counter] * 255 & "#\n")
945             printfopen()
946         End()
947     Else()
948         // Start to build up the output table
949         printfopen(CreateDir & "/" & Experiment_Name & "_Callose_Results"& ".csv", yes)
950         Printf(Experiment_Name & "#")
951         Printf(0 & "#")
952         Printf(nan & "#")
953         Printf(nan & "#")
954         Printf(nan & "#\n")
955         printfopen()
956     End()
957
958     // finish preparing the output csv file
959     Printf("Finish analysing image %s \n", files[file_counter])
960 end()
961 /*** Finish analysing one image ***/
962 End()
963 End()
964 /*** Finish analysing the whole folder ***/
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
```

Additional File 2

```
981  /** Internal Functions **/
982  proc FillSmallHoles(
983  int minimumholearea=20 in "Minimum area for holes. Holes with area less than the limit are filled",
984  objectlist objects inout "Input-output object list. Attributes in the input list are lost."
985  ) "Fills small holes in objects. Holes with area less than the limit MinimumHoleArea are filled."
986  {
987      set(obj_in=objects)
988      fillobjects()
989      set(obj01=objects)
990      carrypixels(image=obj01.body.image, mask=obj_in.body.mask, data=0)
991      stencil2mask(image.vector)
992      mask2stencil()
993      stencil2objects()
994      calcarea()
995      objectfilter(minimumholearea<=area)
996
997      carrypixels(image=obj01.body.image,mask=objects.body.mask,data=0)
998      stencil2objects(image.vector)
999      xor(image=objects.body.image,mask=obj_in.body.mask.image)
1000     setattr(filled, image.vector)
1001 }
1002
1003
1004 proc EdgeDetection_BIG(
1005 image Callose_Image in "the input callose image",
1006 objectlist Callose_Centres in "Input objects list for Callose centers.",
1007 objectlist Callose_objects in "Input objects list for Callose candidates.",
1008 objectlist ROI_Refined in "Input objects list which defines the ROI.",
1009 objectlist Callose_Big_refined out "Output objects list with refined edge detection."
1010 ) //Robatzek Procedures "Perform the edge detection using edge tracing and adaptive thresholding."
1011 {
1012     If(Callose_Image.stddev > 0.125)
1013         // image contains big intensity difference
1014         // Apply global thresholding
1015         ThresholdXX(TuneTH=1.25, Image=Callose_Image)
1016         Mask(Threshold=Threshold * 4.15, Image=Callose_Image)
1017     Else()
1018         ThresholdXX(TuneTH=1.25, Image=Callose_Image)
1019         Mask(Threshold=Threshold * 3.95, Image=Callose_Image)
1020     End()
1021
1022     Mask2Stencil()
1023     Stencil2Objects()
1024     set(Brightest_Area_Ini_Big = objects)
1025
1026     And(Brightest_Area_Ini_Big.body.mask.image, image = ROI_Refined.body.mask.image)
1027     And(Callose_objects.body.mask.image, image = image)
1028     Set(Callose_ROI_BIG = image)
1029     Mask2Stencil(Callose_ROI_BIG)
```

Additional File 2

```
1030 set(restrict_stencil = stencil)
1031
1032 Stencil2Objects(restrict_stencil)
1033 CalcArea(AutoRecalc=yes, objects=objects)
1034 object_contrast_general(reference = Callose_Image, ContrastDef = "WithoutCyto")
1035 CalcIntensity(Image=Callose_Image, Total= no)
1036 Set(Intensity_Threshold = objects.intensity.mean) // Use local thresholding
1037 ObjectFilter(contrast > 0.25 || intensity > Intensity_Threshold * 0.5)
1038 Set(Callose_Initial_Big_processed = objects)
1039
1040 // Divide the big callose into two groups
1041 ObjectFilter(area > 50, objects = Callose_Initial_Big_processed)
1042 Set(Callose_Initial_Big_processed_Big = objects)
1043 ObjectFilter(area <= 50 && area >= 5, objects = Callose_Initial_Big_processed)
1044 Set(Callose_Initial_Big_processed_Small = objects)
1045
1046 // Start to process small objects in big callose deposits
1047 // Get rid of furry edges
1048 Set(stencil = Callose_Initial_Big_processed_Small.body)
1049 CalcErosion(-1, objects = Callose_Initial_Big_processed_Small, Stencil = stencil) // increasing 1 pixel
1050 RenameAttr(body = stencil_eroded)
1051 Set(stencil = objects.body)
1052 CalcErosion(1, objects = objects, Stencil = stencil) // decreasing 1 pixel
1053 RenameAttr(body = stencil_eroded)
1054 Mask2Stencil(objects.body.mask.image)
1055 Stencil2Objects()
1056 CalcArea()
1057 CalcIntensity(Image=Callose_Image, Total= yes)
1058 CalcAttr(AVG_Intensity, (intensity/area)) // Apply local thresholding
1059 Set(Intensity_Threshold = Callose_Image.mean)
1060 ObjectFilter(AVG_Intensity > Intensity_Threshold * 2.75 && area > 5) // get as many centres as possible
1061 Set(Callose_Initial_Big_processed_Small = objects)
1062
1063 // Start to process big objects in the big callose deposits
1064 // Get rid of furry edges
1065 If(Callose_Initial_Big_processed_Big.intensity.mean < Callose_Image.max * 0.75)
1066     If(Callose_Image.stddev > 0.125) // image contains big intensity difference
1067         ThresholdXX(TuneTH=1.25, Image=Callose_Image)
1068         Mask(Threshold=Threshold * 4.525, Image=Callose_Image)
1069     Else()
1070         ThresholdXX(TuneTH=1.25, Image=Callose_Image)
1071         Mask(Threshold=Threshold * 4.325, Image=Callose_Image)
1072     End()
1073
1074     Mask2Stencil()
1075     Stencil2Objects()
1076     // Refine the edge detection of relatively big callose
1077     And(Objects.body.mask.image, image = Callose_Initial_Big_processed_Big.body.mask.image)
1078 Else()
```

Additional File 2

```
1079         Set(image = Callose_Initial_Big_processed_Big.body.mask.image)
1080     End()
1081
1082     Mask2Stencil(image)
1083     Stencil2Objects()
1084     CalcErosion(-1, objects = objects, Stencil = stencil) // increasing 1 pixel
1085     RenameAttr(body = stencil_eroded)
1086     Set(stencil = objects.body)
1087     CalcErosion(1, objects = objects, Stencil = stencil) // decreasing 1 pixel
1088     RenameAttr(body = stencil_eroded)
1089
1090     Mask2Stencil(objects.body.mask.image)
1091     Stencil2Objects()
1092     CalcArea()
1093     CalcIntensity(Image=Callose_Image, Total= yes)
1094     CalcAttr(AVG_Intensity, (intensity/area)) // Apply local thresholding
1095     Set(Intensity_Threshold = Callose_Image.mean)
1096     ObjectFilter(AVG_Intensity > Intensity_Threshold * 2.75) // get as many centres as possible
1097     Set(Callose_Initial_Big_processed_Big = objects)
1098
1099     // Rebuild the callose objects list based on the refined callose centres
1100     Or(Callose_Initial_Big_processed_Small.body.mask.image, image = Callose_Initial_Big_processed_Big.body.mask.image)
1101     Set(Callose_ROI_BIG_refined = image)
1102     Mask2Stencil(Callose_ROI_BIG_refined)
1103     set(restrict_stencil = stencil)
1104     Stencil2Objects(restrict_stencil)
1105     // Find object centres for the big callose
1106     ObjectsFromCenters(Centers=Callose_Centres.center_spot, RestrictiveStencil=restrict_stencil, image=Callose_Image, MinimumArea= 5)
1107     ControlBreakingLines_A(1)
1108     Set(stencil = objects.body)
1109
1110     //Refine big callose based on intial big callose objects list
1111     Stencil2Objects()
1112     FillObjects()
1113     CalcArea(AutoRecalc=yes, objects=objects)
1114     ObjectFilter(area > 5)
1115     CalcIntensity(Image=Callose_Image, Total=no, objects=objects)
1116     Set(Intensity_Threshold = Callose_Image.mean)
1117     ObjectFilter(Intensity > Intensity_Threshold * 2.5) // Local value from callose_value objects list
1118     object_contrast_general(reference = Callose_Image, ContrastDef = "WithoutCyto")
1119     ObjectFilter(contrast > 0.215 || intensity > Intensity_Threshold * 4.25)
1120     ObjectFilter(contrast > 0.325 || intensity > Intensity_Threshold * 2.95) // either very bright or have dim surrounding areas
1121     CalcWidthLength(Upto100=yes)
1122     Set(Callose_Big_refined = objects)
1123 }
1124
1125 proc EdgeDetection_Small(
1126 image Callose_Image in "the input callose image",
1127 objectlist Callose_Centres in "Input objects list for Callose centers.",
```

Additional File 2

```
1128 objectlist Callose_objects in "Input objects list for Callose candidates.",
1129 objectlist ROI_Refined in "Input objects list which defines the ROI.",
1130 objectlist Callose_Small_refined out "Output objects list with refined edge detection."
1131 ) //Robatzek Procedures "Perform the edge detection using edge tracing and adaptive thresholding."
1132 {
1133     /*** Finish refining small callose deposits
1134     If(Callose_Image.stddev > 0.125) // image contains big intensity difference
1135         ThresholdXX(TuneTH=1.25, Image=Callose_Image)
1136         Mask(Threshold=Threshold * 1.625, Image=Callose_Image)
1137     Else()
1138         ThresholdXX(TuneTH=1.25, Image=Callose_Image)
1139         Mask(Threshold=Threshold * 1.525, Image=Callose_Image)
1140     End()
1141     Mask2Stencil()
1142     Stencil2Objects()
1143     set(Brightest_Area_Ini_Small = objects)
1144
1145     And(Brightest_Area_Ini_Small.body.mask.image, image = ROI_Refined.body.mask.image)
1146     // Find small callose ROI
1147     And(Callose_objects.body.mask.image, image = image)
1148     Set(Callose_ROI_Small = image)
1149     Mask2Stencil(Callose_ROI_Small)
1150     set(restrict_stencil = stencil)
1151
1152     Stencil2Objects(restrict_stencil)
1153     CalcArea(AutoRecalc=yes, objects=objects)
1154     object_contrast_general(reference = Callose_Image, ContrastDef = "WithoutCyto")
1155     CalcIntensity(Image=Callose_Image, Total= no)
1156     Set(Intensity_Threshold = objects.intensity.mean) // Use local thresholding
1157     ObjectFilter(contrast > 0.125 || intensity > Intensity_Threshold * 0.25)
1158     Set(Callose_Initial_Small_processed = objects)
1159
1160     ObjectFilter(area > 125, objects = Callose_Initial_Small_processed)
1161     Set(Callose_Initial_Small_processed_Big_INI = objects)
1162     ObjectFilter(area >= 25 && area <= 125, objects = Callose_Initial_Small_processed)
1163     Set(Callose_Initial_Small_processed_Middle_INI = objects)
1164     ObjectFilter(area < 25, objects = Callose_Initial_Small_processed)
1165     Set(Callose_Initial_Small_processed_Small_INI = objects)
1166
1167     // Start to process small objects in small callose deposits
1168     // Get rid of furry edges
1169     Set(stencil = Callose_Initial_Small_processed_Small_INI.body)
1170     CalcErosion(-1, objects = Callose_Initial_Small_processed_Small_INI, Stencil = stencil) // increasing 1 pixel
1171     RenameAttr(body = stencil_eroded)
1172     Set(stencil = objects.body)
1173     CalcErosion(1, objects = objects, Stencil = stencil) // decreasing 1 pixel
1174     RenameAttr(body = stencil_eroded)
1175     Mask2Stencil(objects.body.mask.image)
1176     Stencil2Objects()
```


Additional File 2

```
1177 CalcArea()
1178 CalcIntensity(Image=Callose_Image, Total = yes)
1179 CalcAttr(AVG_Intensity, (intensity/area)) // Apply local thresholding
1180 Set(Intensity_Threshold = Callose_Image.mean)
1181 ObjectFilter(AVG_Intensity > Intensity_Threshold * 1.5 && area > 2.5)
1182 // get as many callose deposits as possible
1183 Set(Callose_Initial_Small_processed_Small = objects)
1184
1185 // Start to process big objects in the big callose deposits
1186 // Get rid of furry edges
1187 If(Callose_Image.stddev > 0.125) // image contains big intensity difference
1188     ThresholdXX(TuneTH=1.25, Image=Callose_Image)
1189     Mask(Threshold=Threshold * 2.25, Image=Callose_Image)
1190 Else()
1191     ThresholdXX(TuneTH=1.25, Image=Callose_Image)
1192     Mask(Threshold=Threshold * 2.15, Image=Callose_Image)
1193 End()
1194
1195 Mask2Stencil()
1196 Stencil2Objects()
1197 // Refine the edge detection of relatively big callose
1198 And(Objects.body.mask.image, image = Callose_Initial_Small_processed_Middle_INI.body.mask.image)
1199
1200 Mask2Stencil(image)
1201 Stencil2Objects()
1202 CalcErosion(-1, objects = objects, Stencil = stencil) // increasing 1 pixel
1203 RenameAttr(body = stencil_eroded)
1204 Set(stencil = objects.body)
1205 CalcErosion(1, objects = objects, Stencil = stencil) // decreasing 1 pixel
1206 RenameAttr(body = stencil_eroded)
1207 Mask2Stencil(objects.body.mask.image)
1208 Stencil2Objects()
1209 CalcArea()
1210 CalcIntensity(Image=Callose_Image, Total= yes)
1211 CalcAttr(AVG_Intensity, (intensity/area)) // Apply local thresholding
1212 Set(Intensity_Threshold = Callose_Image.mean)
1213 ObjectFilter(AVG_Intensity > Intensity_Threshold * 1.5 && area > 2.5)
1214 // get as many callose deposits as possible
1215 Set(Callose_Initial_Small_processed_Middle = objects)
1216
1217 // Start to process big objects in the big callose deposits
1218 // Get rid of furry edges
1219 If(Callose_Image.stddev > 0.125) // image contains big intensity difference
1220     ThresholdXX(TuneTH=1.25, Image=Callose_Image)
1221     Mask(Threshold=Threshold * 3.05, Image=Callose_Image)
1222 Else()
1223     ThresholdXX(TuneTH=1.25, Image=Callose_Image)
1224     Mask(Threshold=Threshold * 2.95, Image=Callose_Image)
1225 End()
```

Additional File 2

```
1226
1227 Mask2Stencil()
1228 Stencil2Objects()
1229 // Refine the edge detection of relatively big callose
1230 And(Objects.body.mask.image, image = Callose_Initial_Small_processed_Big_INI.body.mask.image)
1231
1232 Mask2Stencil(image)
1233 Stencil2Objects()
1234 CalcErosion(-2, objects = objects, Stencil = stencil) // increasing 1 pixel
1235 RenameAttr(body = stencil_eroded)
1236 Mask2Stencil(objects.body.mask.image)
1237 Stencil2Objects()
1238 CalcErosion(2, objects = objects, Stencil = stencil) // decreasing 1 pixel
1239 RenameAttr(body = stencil_eroded)
1240 Mask2Stencil(objects.body.mask.image)
1241 Stencil2Objects()
1242 CalcArea()
1243 CalcIntensity(Image=Callose_Image, Total= yes)
1244 CalcAttr(AVG_Intensity, (intensity/area)) // Apply local thresholding
1245 Set(Intensity_Threshold = Callose_Image.mean)
1246 ObjectFilter(AVG_Intensity > Intensity_Threshold * 1.75)
1247 // get as many callose deposits as possible
1248 Set(Callose_Initial_Small_processed_Big = objects)
1249
1250 // Rebuild the callose objects list based on the refined callose centres
1251 Or(Callose_Initial_Small_processed_Small.body.mask.image, image = Callose_Initial_Small_processed_Middle.body.mask.image)
1252 Or(image, image = Callose_Initial_Small_processed_Big.body.mask.image)
1253 Set(Callose_ROI_SMALL = image)
1254 Mask2Stencil(Callose_ROI_SMALL)
1255 set(restrict_stencil = stencil)
1256 Stencil2Objects(restrict_stencil)
1257 CalcWidthLength(Upto100=yes)
1258
1259 // Find object centres for the small callose
1260 ObjectsFromCenters(Centers=Callose_Centres.center_spot, RestrictiveStencil=restrict_stencil, image=Callose_Image, MinimumArea= 5)
1261 ControlBreakingLines_A(1)
1262 Set(stencil = objects.body)
1263
1264 //Start to refine small callose
1265 Set(stencil = objects.body)
1266 Stencil2Objects()
1267 CalcIntensity(Image=Callose_Image, Total=no, objects=objects)
1268 Set(Intensity_Threshold = Callose_Image.mean)
1269 ObjectFilter(Intensity > Intensity_Threshold * 1.75) // Local value from callose_value objects list
1270 CalcArea()
1271 ObjectFilter(area >= 5)
1272
1273 calcArea(border, AutoRecalc=yes, objects=objects)
1274 object_contrast_general(reference = Callose_Image, ContrastDef = "WithoutCyto")
```

Additional File 2

```
1275 ObjectFilter((area < 32.5 && (contrast > 0.125 || intensity > Intensity_Threshold * 2.25)) || (area >= 32.5 && (contrast > 0.15 || intensity > Intensity_Threshold * 2.5)))
1276 // either very bright or have dim surrounding areas
1277
1278 // The small callose objects have been expanded
1279 CalcAttr(Roundness, 4 * pi * area / (border_area * border_area) * 100)
1280 ObjectFilter(Roundness > 90 || contrast > 0.195 || intensity > Intensity_Threshold * 2.25)
1281 ObjectFilter(Roundness > 115 || ((intensity > Intensity_Threshold * 2.25 || contrast > 0.325) && area >= 25) || area < 25)
1282 // normally roundness should be over 90
1283
1284 CalcWidthLength(Upto100=yes)
1285 ObjectFilter(half_width > 1 && full_length > 1.5)
1286 ObjectFilter((full_length > 3.5 && full_length < 13.75 && half_width > 1.5) || (contrast > 0.175 || intensity > Intensity_Threshold * 2.75))
1287
1288 CalcAttr(WL_Ratio, full_length/(half_width * 2))
1289 ObjectFilter(!(WL_Ratio >= 3 && contrast < 0.35 && area < 75)) // exceptional cases
1290 DeleteAttr("outerzone", objects = objects)
1291 DeleteAttr("zone", objects = objects)
1292 DeleteAttr("top_spot1", objects = objects)
1293 DeleteAttr("top_spot2", objects = objects)
1294 Set(Callose_Small_refined = objects)
1295 }/** Internal functions complete**//
1296
```