Standard Operating Procedure for Extracting Cell Morphology Using MATLAB

The guideline below outlines the steps for extracting the intrinsic properties of plant vascular cells using microscope images and MATLAB tools. The example is provided for wild type plants and involves a manual selection step. In the case of simpler or more colour-defined objects, the process of selection could be automated.

I. Cell Selection

In this section, we outline the manual selection process used to prepare the data for analysis.

1. Obtain image of known size/calibration
2. Load it in GIMP or another similar software
3. Identify the 'Fuzzy Select Tool' or an equivalent from the programme Toolbox (Fig. 1).

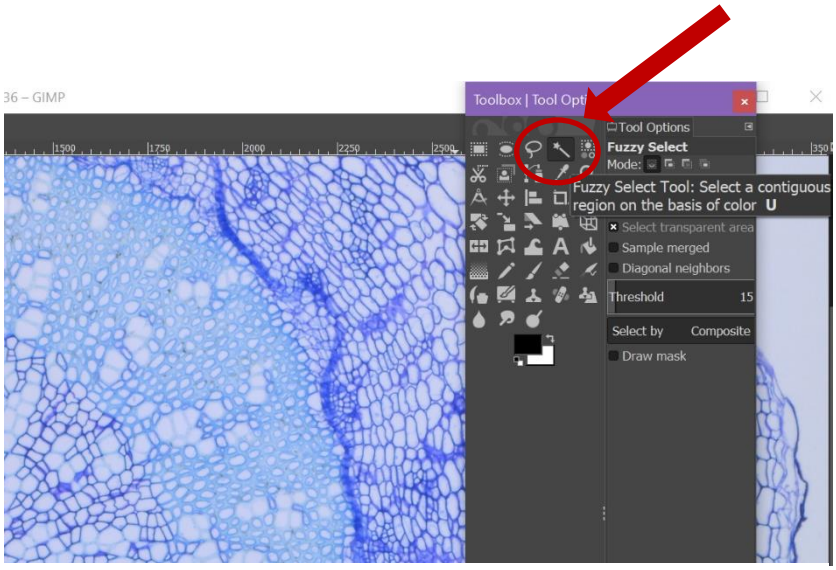

Fig. 1

4. Identify the object (in this case cell) of interest, adjusting the Fuzzy Select Tool threshold as required (Fig. 3).

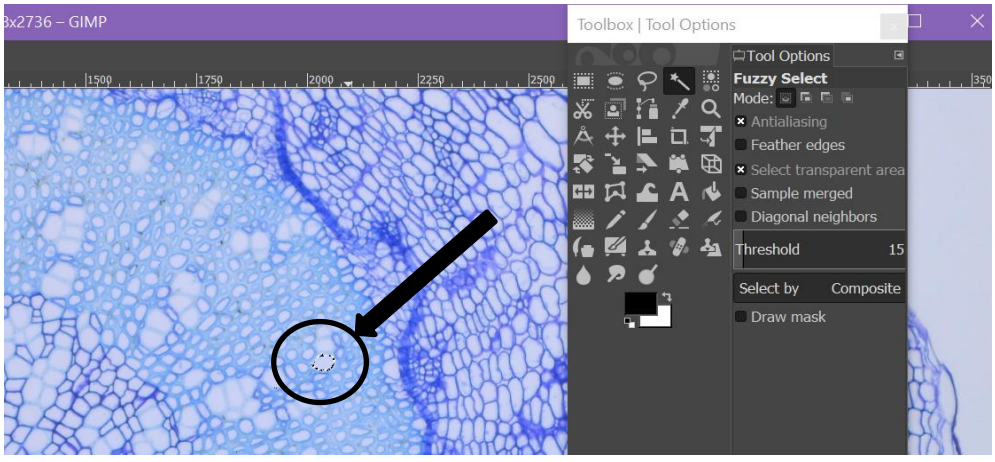

Fig. 2

5. Next, assign your object a colour, with a simple RGB code that can be tracked using MATLAB tools (e.g. ‘red’ (255,0,0)) (Fig. 3).
6. Use Paintbrush or Bucket Fill to colour object (Fig. 3).

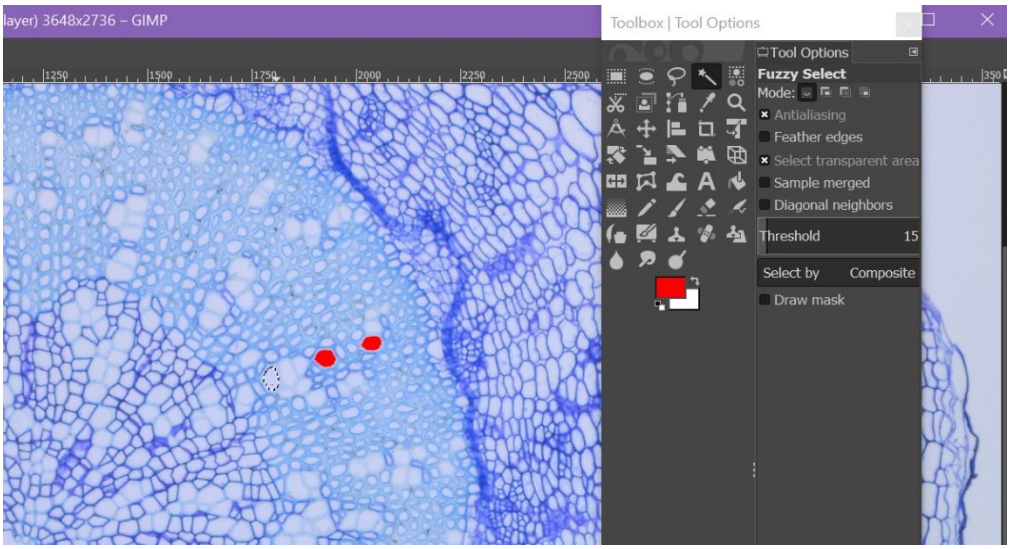

Fig. 3

7. Assign objects of the same type the same colour (Fig. 3).
8. Double-check for defects in colouring/missing areas and correct for those. Note that errors depend on image quality, bias in two-dimensional imaging and accuracy of software should be accounted for. Consistency in choice and method of selection/colouring is imperative for optimal results.

Note: Object colour code can be checked in MATLAB by reading an image X and applying the command ‘impixel(X)’ which allows the user to directly select a pixel and receive as an output its RGB code.

II. MATLAB Procedure

In this section, the MATLAB procedure used to analyse the image data prepared as per the section above is described. The code can be optimised depending on the user requirements. In the section “Main function”, we discuss the part of the programme that is run to obtain results, while the ‘Other functions’ section contains the subfunctions that are responsible for performing operations within the programme.

Main function

The function breaks up a coloured image into binary images corresponding to the different cell types / object types that had been colour-coded. There are different ways to organise and export the data, depending on the analysis requirements. Here, we describe one possible way of organisation.

1. Once you have colour-coded your images, deposit them in a folder (in this case, the folder is called ‘WT’ and the variable ‘sourceWT’ is assigned this folder’s address.)
2. The main function uses the function ‘extractCellProperties’ which acquires the properties of the objects. The function will be described later in this document, in the ‘Other functions’ section.

```
sourceWT = 'C:\Users\NBuser\Documents\MATLAB\Images Paper\Coloured Images\WT';
[propertiesXylem_WT,propertiesFiber_WT,propertiesPhloem_WT,propertiesParynchema_WT]=extractCellProperties(sourceWT);
```

3. The next part of the main function organises your output. It creates a vector of column names, then use the function 'AssignSpecies' to add 'WT' as the corresponding name to all rows that contain data for cells from 'WT' images. Use the function 'assignCellType' to give cell type names to the corresponding data from those cell types (assign row name 'xylem' for a data row with data for an object of type 'xylem').

```
CellProperties={'Species' 'CellType' 'CellArea' 'Ratio' 'Perimeter'};

%name species for each matrix
SpeciesWT='WT';
[Species_WT_Xy,Species_WT_Fi,Species_WT_Ph,Species_WT_Pa]=assignSpecies(propertiesXylem_WT,propertiesFiber_WT,propertiesPhloem_WT,propertiesParynchema_WT,SpeciesWT);

%assign names to the cell types in WT
[cellType_WT_Xy,cellType_WT_Fi,cellType_WT_Ph,cellType_WT_Pa]=assignCellTypes(propertiesXylem_WT,propertiesFiber_WT,propertiesPhloem_WT,propertiesParynchema_WT);
```

4. Next, the main function creates a table with the extracted properties.

```
%create tables with the properties
%add variable names for each
%For WT
WT_Table_Xy = table(Species_WT_Xy, cellType_WT_Xy,
propertiesXylem_WT(:,2),propertiesXylem_WT(:,3),propertiesXylem_WT(:,4)); %the name of the rows come after 'RowNames'
WT_Table_Xy.Properties.VariableNames = CellProperties;

WT_Table_Fi = table(Species_WT_Fi, cellType_WT_Fi,
propertiesFiber_WT(:,2),propertiesFiber_WT(:,3),propertiesFiber_WT(:,4)); %the name of the rows come after 'RowNames'
WT_Table_Fi.Properties.VariableNames = CellProperties;

WT_Table_Ph = table(Species_WT_Ph, cellType_WT_Ph,
propertiesPhloem_WT(:,2),propertiesPhloem_WT(:,3),propertiesPhloem_WT(:,4)); %the name of the rows come after 'RowNames'
WT_Table_Ph.Properties.VariableNames = CellProperties;

WT_Table_Pa = table(Species_WT_Pa, cellType_WT_Pa,
propertiesParynchema_WT(:,2),propertiesParynchema_WT(:,3),propertiesParynchema_WT(:,4)); %the name of the rows come after 'RowNames'
WT_Table_Pa.Properties.VariableNames = CellProperties;
```

5. Finally, spreadsheets with data for each cell type are created.

```
filename_WT_Xy='WT_Table_Xy_tidy.xlsx';
writetable(WT_Table_Xy, filename_WT_Xy);
```

```
filename_WT_Fi='WT_Table_Fi_tidy.xlsx';
writetable(WT_Table_Fi, filename_WT_Fi);

filename_WT_Ph='WT_Table_Ph_tidy.xlsx';
writetable(WT_Table_Ph, filename_WT_Ph);

filename_WT_Pa='WT_Table_Pa_tidy.xlsx';
writetable(WT_Table_Pa, filename_WT_Pa);
```

Other functions

1. The function 'extractCellProperties' take the address of the source folder, loops over the images and outputs the properties of each object type (here, cell types). The sub-functions are described below.

```
function[propertiesXylem,propertiesFiber,propertiesPhloem,propertiesParynchema]=extract
CellProperties(sourceFolder)
images =dir(fullfile(sourceFolder,'*.png'));
numFiles = length(images); %how many images in the folder

%start with a row of zeros to build tables
[propertiesXylem, propertiesFiber, propertiesPhloem, propertiesParynchema] =
deal(zeros(4,1)');

%loop over images
%splitExport splits the images into binary images - white objects on black background
%these can be viewed as connected components of pixels whose properties can
%be extracted (area, perimeter, ellipticity)
%splitExport returns, one image at a time, all the properties for all the
%different cell types for that image.
%the below loops over the images and stacks in a table the properties for
%all images for a specific genotype
for k = length(images):-1:1
[Xylem_Table_Num, Fiber_Table_Num,Phloem_Table_Num,
Parynchema_Table_Num]=splitExport(images(k).name,sourceFolder);
   propertiesXylem=vertcat(propertiesXylem, Xylem_Table_Num);
   propertiesFiber=vertcat(propertiesFiber, Fiber_Table_Num);
   propertiesPhloem=vertcat(propertiesPhloem, Phloem_Table_Num);
   propertiesParynchema=vertcat(propertiesParynchema, Parynchema_Table_Num);
end

%remove zeros and convert pixels to microns
[propertiesXylem, propertiesFiber, propertiesPhloem,
propertiesParynchema]=convert2Microns(propertiesXylem, propertiesFiber,
propertiesPhloem, propertiesParynchema);
```

2. 'splitExport' uses functions 'imagesNew', 'ConnectedComponents' and 'NumberProperties'. It takes the images in the source file and outputs the properties of the cells.

```
function[Xylem_Table_Num_WT, Fiber_Table_Num_WT,Phloem_Table_Num_WT,
Parynchema_Table_Num_WT]=splitExport(baseFileName,source)
```

```
%this function takes the 'name' for a source and the base name for an image
%reads the image and returns table of properties for that image
   fullFileName = fullfile(source, baseFileName); %combine for every filename
   imageRead= imread(fullFileName); %read each image
   [Ix,Ip,Iph,Ipa]=imagesNew(imageRead); %separate the cell images for each
   %for the table of properties
   [Ix_cc,Ip_cc,Iph_cc,Ipa_cc]=ConnectedComponents(Ix,Ip,Iph,Ipa); %connected
components for each cell type
   Xylem_Table_Num_WT=NumberProperties(Ix_cc);
   Fiber_Table_Num_WT=NumberProperties(Ip_cc);
   Phloem_Table_Num_WT=NumberProperties(Iph_cc);
   Parynchema_Table_Num_WT=NumberProperties(Ipa_cc);
```

3. The function 'imagesNew' takes an image and splits it into binary images according to the specified colour-scheme. Note that in 'imagesNew', the RGB codes used must be specific to the data, i.e. the colours chosen in the "Cell Selection" section. The command 'bwareaopen', known as 'area opening', is used to clear noise and removes all connected components with fewer than the specified number of pixels, thus producing another binary image. The specified number of pixels depends on the objects under investigation, and will likely be experiment-specific.

```
function [Ix,Ip,Iph,Ipa]=imagesNew(I)

clc;
[x,y,numberofcolours]=size(I);

%create black images corresponding to each cell type
[Ix,Ip,Iph,Ipa] = deal(0*I);

for i=1:x
   for j=1:y
      if ((I(i,j,1)==254)&&(I(i,j,2)==0)&&(I(i,j,3)==0)) %image for xylem
         Ix(i,j,1)=255;
         Ix(i,j,2)=255;
         Ix(i,j,3)=255;
      elseif ((I(i,j,1)==0)&&(I(i,j,2)==0)&&(I(i,j,3)==254)) %image for fibers
         Ip(i,j,1)=255;
         Ip(i,j,2)=255;
         Ip(i,j,3)=255;
      elseif ((I(i,j,1)==0)&&(I(i,j,2)==255)&&(I(i,j,3)==3))%image for phloem
         Iph(i,j,1)=255;
         Iph(i,j,2)=255;
         Iph(i,j,3)=255;
      elseif ((I(i,j,1)==255)&&(I(i,j,2)==255)&&(I(i,j,3)==0)) %image for parynchema
         Ipa(i,j,1)=255;
         Ipa(i,j,2)=255;
         Ipa(i,j,3)=255;
      end
   end
end
```

```
Ix=im2bw(Ix); %transform into binary image
Ix=bwareaopen(Ix,50);%clear noise


Ip=im2bw(Ip);
Ip=bwareaopen(Ip,20);

Iph=im2bw(Iph);
Iph=bwareaopen(Iph,10);

Ipa=im2bw(Ipa);
Ipa=bwareaopen(Ipa,50);
```

4. The function 'NumerProperties' extracts the area ellipticity and perimeter of the objects.

```
function [NumberPropertiesTable]=NumberProperties(cc) %input connected components,
receive their properties

celldata = regionprops(cc, 'Area','MajorAxisLength','MinorAxisLength', 'Eccentricity',
'Perimeter'); %properties of connected components
cell_number=cc.NumObjects;
cell_areas=[celldata.Area]; %each element is a scalar=actual number of pixels in the region

lambda1=[celldata.MajorAxisLength];
lambda2=[celldata.MinorAxisLength];
ratio=lambda2./lambda1; %create ratio column
cell_perimeter=[celldata.Perimeter];

typeofcellarray = zeros(cell_number, 1);

NumberPropertiesTable(:, 1)= typeofcellarray';
NumberPropertiesTable(:, 2) = cell_areas';
NumberPropertiesTable(:, 3) = ratio';
NumberPropertiesTable(:, 4) = cell_perimeter';
NumberPropertiesTable;
```

5. The function 'ConnectedComponents' extracts the connected components of white pixels in the images. This is done by the command 'bwconncomp' which takes an image and a user-specified connectivity parameter. The connectivity parameter (i.e. how many pixel neighbours of the same colour are required to consider that pixel part of the connected component) are again specified for the particular objects (i.e. the specific cell types). Connectivity can be adjusted according to the requirements of the problem to be solved.

```
function [Ix_cc,Ip_cc,Iph_cc,Ipa_cc]=ConnectedComponents(Ix,Ip,Iph,Ipa)

Ix_cc = bwconncomp(Ix, 8);
Ip_cc = bwconncomp(Ip, 4);
Iph_cc = bwconncomp(Iph, 8);
Ipa_cc = bwconncomp(Ipa, 8);
```

6. The function ‘convert2Microns’, used in ‘extractCellProperties’ converts the data from pixels to microns using a calibration ratio which was measured interactively using a graticule image and the command ‘imtool’ in MATLAB.

```
function[propertiesXylem, propertiesFiber, propertiesPhloem,
propertiesParynchema]=convert2Microns(propertiesXylem, propertiesFiber,
propertiesPhloem, propertiesParynchema)
%the function takes the table of properties of different cell types
%removes first row of zeros
%then uses callibration factors for area and length to convert pixels to
%microns

%calculate spatial and length factors for conversion from pixels to microns
SpatialFactor_Area=[10/33]*[10/33];
SpatialFactor_Length=[10/33];

%remove first row of zeros
propertiesXylem(1,:)=[];
propertiesFiber(1,:)=[];
propertiesPhloem(1,:)=[];
propertiesParynchema(1,:)=[];
%callibrate pixels to microns
propertiesXylem(:,2)=propertiesXylem(:,2)*SpatialFactor_Area;
propertiesXylem(:,4)=propertiesXylem(:,4)*SpatialFactor_Length;

propertiesFiber(:,2)=propertiesFiber(:,2)*SpatialFactor_Area;
propertiesFiber(:,4)=propertiesFiber(:,4)*SpatialFactor_Length;

propertiesPhloem(:,2)=propertiesPhloem(:,2)*SpatialFactor_Area;
propertiesPhloem(:,4)=propertiesPhloem(:,4)*SpatialFactor_Length;

propertiesParynchema(:,2)=propertiesParynchema(:,2)*SpatialFactor_Area;
propertiesParynchema(:,4)=propertiesParynchema(:,4)*SpatialFactor_Length;
```

7. The functions ‘assignSpecies’ and ‘assignCellTypes’ are responsible for organising the data in such a way so as to have each row of data with corresponding genotype (e.g. wild type) and cell type (e.g. xylem, phloem, etc.) information.

```
function[Species_Xy,Species_Fi,Species_Ph,Species_Pa]=assignSpecies(propertiesXylem,
propertiesFiber,propertiesPhloem,propertiesParynchema,speciesName)

Species_Xy=repmat(speciesName,size(propertiesXylem(:,1)),1);
Species_Fi=repmat(speciesName,size(propertiesFiber(:,1)),1);
Species_Ph=repmat(speciesName,size(propertiesPhloem(:,1)),1);
Species_Pa=repmat(speciesName,size(propertiesParynchema(:,1)),1);

function[cellType_Xy,cellType_Fi,cellType_Ph,cellType_Pa]=assignCellTypes(propertiesXyl
em,propertiesFiber,propertiesPhloem,propertiesParynchema)
cellTypes={'Xylem' 'Fiber' 'Phloem' 'Parynchema'};
cellType_Xy=repmat(cellTypes(1),size(propertiesXylem(:,1)),1);
cellType_Fi=repmat(cellTypes(2),size(propertiesFiber(:,1)),1);
cellType_Ph=repmat(cellTypes(3),size(propertiesPhloem(:,1)),1);
cellType_Pa=repmat(cellTypes(4),size(propertiesParynchema(:,1)),1);
```