Main.m

```
% This script is used for data analysis of glial cells from .czi files.
% It processes the images, calculates various metrics, and saves the results
% along with the skeletal structure of the selected portions of the glial
silhouettes
% to a specified location. The script and included functions will bring up a
GUI interface
% and print quick controls for interaction in the console
% Tips:
% 1. Ensure that the destination folder for saving the results already exists.
% 2. The script will break if you move onto the next stack without saving a
ROI.
%    At least one glial structure must be identified before moving to the next
stack.
% 3. Extended idle times may cause the GUI to be interrupted and the code to
break.
% 4. Adjust Skeleton plots location in the rtrace.m function as needed.
% Define the folder number containing the data.
folder_num = 1039;
folder_num = num2str(folder_num);
doFolder = [folder_num];
% Define the file paths that specified .czi files are located in.
paths = getFilePaths(['RawData_Test\' doFolder],'.czi');
%%
% Define the properties to be extracted:
% Number of branch points \ Total volume \ Mean branch length \ Branch Depth \
Cell body size
allProperties = [{'Filename'} {'ROI_Number'} {'Number of branch points'}
{'Total Area'} {'Mean Branch Length'} ...
   {'Branch Depth'} {'Cell Body Size'}];
% Loop through each file path
for p = paths'
   data = bfopen(p{1});
   data = data{1};
   z = nan([size(data{1}) length(data)]);
   for i = 1:length(data)
       z(:,:,i) = double(data{i});
   end

   nz = znorm(z);
   tmp = rtrace(nz,z,p{1});
   allProperties = [allProperties; tmp];
end
% Define the destination folder and file name for saving the results
destinationFolderName = ['ROI_Save']; % Folder destination must already exist
customName = 'ROI_Stats_' ; % Uniform name of each excel file
saveTo = fullfile(destinationFolderName, [customName , doFolder]); % file path
of created excel file = 'destinationFolderName' named 'customName' + 'doFolder'
% Save the extracted properties to an Excel file
```

```
xlswrite(saveTo, allProperties);
```

bfCheckJavaMemory.m

```
function [] = bfCheckJavaMemory(varargin)
% bfCheckJavaMemory warn if too little memory is allocated to Java
%
% SYNOPSIS  bfCheckJavaMemory()
%
% Input
%
%   minMemory - (Optional) The minimum suggested memory setting in MB.
%   Default: 512
%
% Output
%
%    A warning message is printed if too little memory is allocated.
% OME Bio-Formats package for reading and converting biological file formats.
%
% Copyright (C) 2014 - 2021 Open Microscopy Environment:
%   - Board of Regents of the University of Wisconsin-Madison
%   - Glencoe Software, Inc.
%   - University of Dundee
%
% Redistribution and use in source and binary forms, with or without
% modification, are permitted provided that the following conditions are met:
%
% 1. Redistributions of source code must retain the above copyright
%    notice, this list of conditions and the following disclaimer.
%
% 2. Redistributions in binary form must reproduce the above copyright
%    notice, this list of conditions and the following disclaimer in
%    the documentation and/or other materials provided with the distribution.
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGE.
runtime = javaMethod('getRuntime', 'java.lang.Runtime');
maxMemory = runtime.maxMemory() / (1024 * 1024);
ip = inputParser;
ip.addOptional('minMemory', 512, @isscalar);
ip.parse(varargin{:});
minMemory = ip.Results.minMemory;
warningID = 'BF:lowJavaMemory';
```

```
if maxMemory < minMemory - 64
   warning_msg = [...
       '*** Insufficient memory detected. ***\n'...
       '*** %dm found ***\n'...
       '*** %dm or greater is recommended ***\n'...
       '*** See http://www.mathworks.com/matlabcentral/answers/92813 ***\n'...
       '*** for instructions on increasing memory allocation. ***\n'];
   warning(warningID, warning_msg, round(maxMemory), minMemory);
end
```

bfCheckJavaPath.m

```
function [status, version] = bfCheckJavaPath(varargin)
% bfCheckJavaPath check Bio-Formats is included in the Java class path
%
% SYNOPSIS  bfCheckJavaPath()
%           status = bfCheckJavaPath(autoloadBioFormats)
%           [status, version] = bfCheckJavaPath()
%
% Input
%
%    autoloadBioFormats - Optional. A boolean specifying the action
%    to take if Bio-Formats is not in the javaclasspath.  If true,
%    tries to find a Bio-Formats jar file and adds it to the java
%    class path.
%    Default - true
%
% Output
%
%    status - Boolean. True if the Bio-Formats classes are in the Java
%    class path.
%
%
%    version - String specifying the current version of Bio-Formats if
%    Bio-Formats is in the Java class path. Empty string otherwise.
% OME Bio-Formats package for reading and converting biological file formats.
%
% Copyright (C) 2012 - 2021 Open Microscopy Environment:
%   - Board of Regents of the University of Wisconsin-Madison
%   - Glencoe Software, Inc.
%   - University of Dundee
%
% Redistribution and use in source and binary forms, with or without
% modification, are permitted provided that the following conditions are met:
%
% 1. Redistributions of source code must retain the above copyright
%    notice, this list of conditions and the following disclaimer.
%
% 2. Redistributions in binary form must reproduce the above copyright
%    notice, this list of conditions and the following disclaimer in
%    the documentation and/or other materials provided with the distribution.
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
```

```
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGE.
% Input check
ip = inputParser;
ip.addOptional('autoloadBioFormats', true, @isscalar);
ip.parse(varargin{:});
[status, version] = has_working_bioformats();
if ~status && ip.Results.autoloadBioFormats,
   jarPath = fullfile(fileparts(mfilename('fullpath')), ...
                      'bioformats_package.jar');
   if (exist(jarPath) == 2)
       javaaddpath(jarPath);
       [status, version] = has_working_bioformats();
       if (~status)
           javarmpath(jarPath);
       end
   end
end
% Return true if bioformats java interface is working, false otherwise.
% Not working will probably mean that the classes are not in
% the javaclasspath.
function [status, version] = has_working_bioformats()
status = true;
version = '';
try
   % If the following fails for any reason, then bioformats is not working.
   % Getting the version number and creating a reader is the bare minimum.
   reader = javaObject('loci.formats.in.FakeReader');
   if is_octave()
       version = char(java_get('loci.formats.FormatTools', 'VERSION'));
   else
       version = char(loci.formats.FormatTools.VERSION);
   end
catch
   status = false;
end
```

bfGetFileExtensions.m

```
function fileExt = bfGetFileExtensions
% bfGetFileExtensions list all extensions supported by Bio-Formats
%
% Synopsis: fileExt = bfGetExtensions()
%
% Input
%      none
%
% Output
%      fileExt:  a cell array of dimensions n x2 where the first colum
%      gives the extension and the second the name of the corresponding
%      format.
%      This cell array is formatted to be used with uigetfile funciton.
% OME Bio-Formats package for reading and converting biological file formats.
%
% Copyright (C) 2012 - 2021 Open Microscopy Environment:
%   - Board of Regents of the University of Wisconsin-Madison
%   - Glencoe Software, Inc.
%   - University of Dundee
%
% Redistribution and use in source and binary forms, with or without
% modification, are permitted provided that the following conditions are met:
%
% 1. Redistributions of source code must retain the above copyright
%    notice, this list of conditions and the following disclaimer.
%
% 2. Redistributions in binary form must reproduce the above copyright
%    notice, this list of conditions and the following disclaimer in
%    the documentation and/or other materials provided with the distribution.
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGE.
% Get all readers and create cell array with suffixes and names
readers = javaMethod('getReaders', javaObject('loci.formats.ImageReader'));
fileExt = cell(numel(readers), 2);
for i = 1:numel(readers)
   suffixes = readers(i).getSuffixes();
   if is_octave()
       %% FIXME when https://savannah.gnu.org/bugs/?42700 gets fixed
```

```
        ExtSuf = cell(numel(suffixes), 1);
        for j = 1:numel(suffixes)
            ExtSuf{j} = char(suffixes(j));
        end
        fileExt{i, 1} = ExtSuf;
    else
        fileExt{i, 1} = arrayfun(@char, suffixes, 'Unif', false);
    end
    fileExt{i, 2} = char(readers(i).getFormat());
end
% Concatenate all unique formats
allExt = unique(vertcat(fileExt{:, 1}));
allExt = allExt(~cellfun(@isempty, allExt));
fileExt = vertcat({allExt, 'All formats'}, fileExt);
% Format file extensions
for i = 1:size(fileExt, 1)
    fileExt{i, 1} = sprintf('*.%s;', fileExt{i, 1}{:});
    fileExt{i, 1}(end) = [];
end
```

bfGetPlane.m

```
function I = bfGetPlane(r, varargin)
% BFGETPLANE Retrieve the plane data from a reader using Bio-Formats
%
%   I = bfGetPlane(r, iPlane) returns a specified plane from the input
%   format reader. The index specifying the plane to retrieve should be
%   contained between 1 and the number of planes for the series. Given a
%   set of (z, c, t) plane coordinates, the plane index (0-based) can be
%   retrieved using r.getIndex(z, c, t).
%
%   I = bfGetPlane(r, iPlane, x, y, width, height) only returns the tile
%   which origin is specified by (x, y) and dimensions are specified by
%   (width, height).
%
% Examples
%
%    I = bfGetPlane(r, 1) % First plane of the series
%    I = bfGetPlane(r, r.getImageCount()) % Last plane of the series
%    I = bfGetPlane(r, r.getIndex(0, 0, 0) + 1) % First plane of the series
%    I = bfGetPlane(r, 1, 1, 1, 20, 20) % 20x20 tile originated at (0, 0)
%
% See also: BFGETREADER
% OME Bio-Formats package for reading and converting biological file formats.
%
% Copyright (C) 2012 - 2021 Open Microscopy Environment:
%   - Board of Regents of the University of Wisconsin-Madison
%   - Glencoe Software, Inc.
%   - University of Dundee
%
% Redistribution and use in source and binary forms, with or without
% modification, are permitted provided that the following conditions are met:
%
% 1. Redistributions of source code must retain the above copyright
%    notice, this list of conditions and the following disclaimer.
%
% 2. Redistributions in binary form must reproduce the above copyright
%    notice, this list of conditions and the following disclaimer in
%    the documentation and/or other materials provided with the distribution.
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
```

```
% POSSIBILITY OF SUCH DAMAGE.
% Input check
ip = inputParser;
isValidReader = @(x) isa(x, 'loci.formats.IFormatReader') && ...
   ~isempty(x.getCurrentFile());
ip.addRequired('r', isValidReader);
ip.parse(r);
% Plane check
isValidPlane = @(p) bfTestInRange(p,'iPlane',r.getImageCount());
% Optional tile arguments check
isValidX = @(x) bfTestInRange(x,'x',r.getSizeX());
isValidY = @(y) bfTestInRange(y,'y',r.getSizeY());
isValidWidth = @(w) bfTestInRange(w,'width',r.getSizeX()-varargin{2}+1);
isValidHeight = @(h) bfTestInRange(h,'height',r.getSizeY()-varargin{3}+1);
ip.addRequired('iPlane', isValidPlane);
ip.addOptional('x', 1, isValidX);
ip.addOptional('y', 1, isValidY);
ip.addOptional('width', r.getSizeX(), isValidWidth);
ip.addOptional('height', r.getSizeY(), isValidHeight);
ip.parse(r, varargin{:});
% Get pixel type
pixelType = r.getPixelType();
bpp = javaMethod('getBytesPerPixel', 'loci.formats.FormatTools', pixelType);
fp = javaMethod('isFloatingPoint', 'loci.formats.FormatTools', pixelType);
sgn = javaMethod('isSigned', 'loci.formats.FormatTools', pixelType);
little = r.isLittleEndian();
plane = r.openBytes(...
   ip.Results.iPlane - 1, ip.Results.x - 1, ip.Results.y - 1, ...
   ip.Results.width, ip.Results.height);
% Convert byte array to MATLAB image
I = javaMethod('makeDataArray2D', 'loci.common.DataTools', plane, ...
   bpp, fp, little, ip.Results.height);
if ~sgn
   % Java does not have explicitly unsigned data types;
   % hence, we must inform MATLAB when the data is unsigned
   I = I(:);        % Need vector for typecast
   switch class(I)
       case 'int8'
           I = typecast(I, 'uint8');
       case 'int16'
           I = typecast(I, 'uint16');
       case 'int32'
           I = typecast(I, 'uint32');
       case 'int64'
           I = typecast(I, 'uint64');
   end
   I = reshape(I, [ip.Results.height ip.Results.width]); % Convert back to
matrix
end
```

bfGetReader.m

```
function r = bfGetReader(varargin)
% BFGETREADER return a reader for a microscopy image using Bio-Formats
%
%   r = bfGetReader() creates an empty Bio-Formats reader extending
%   loci.formats.ReaderWrapper.
%
%   r = bfGetReader(id) where id is a path to an existing file creates and
%   initializes a reader for the input file.
%
% Examples
%
%    r = bfGetReader()
%    I = bfGetReader(path_to_file)
%
%
% See also: BFGETPLANE
% OME Bio-Formats package for reading and converting biological file formats.
%
% Copyright (C) 2012 - 2021 Open Microscopy Environment:
%   - Board of Regents of the University of Wisconsin-Madison
%   - Glencoe Software, Inc.
%   - University of Dundee
%
% Redistribution and use in source and binary forms, with or without
% modification, are permitted provided that the following conditions are met:
%
% 1. Redistributions of source code must retain the above copyright
%    notice, this list of conditions and the following disclaimer.
%
% 2. Redistributions in binary form must reproduce the above copyright
%    notice, this list of conditions and the following disclaimer in
%    the documentation and/or other materials provided with the distribution.
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGE.
% Input check
ip = inputParser;
ip.addOptional('id', '', @ischar);
ip.addOptional('stitchFiles', false, @isscalar);
```

```
ip.parse(varargin{:});
id = ip.Results.id;
% verify that enough memory is allocated
bfCheckJavaMemory();
% load the Bio-Formats library into the MATLAB environment
status = bfCheckJavaPath();
assert(status, ['Missing Bio-Formats library. Either add bioformats_package.jar
'...
   'to the static Java path or add it to the Matlab path.']);
% Check if input is a fake string
isFake = strcmp(id(max(1, end - 4):end), '.fake');
if ~isempty(id) && ~isFake
   % Check file existence using fileattrib
   [status, f] = fileattrib(id);
   assert(status && f.directory == 0, 'bfGetReader:FileNotFound',...
       'No such file: %s', id);
   id = f.Name;
end
% Create a loci.formats.ReaderWrapper object
r = javaObject('loci.formats.ChannelSeparator', ...
              javaObject('loci.formats.ChannelFiller'));
if ip.Results.stitchFiles
   r = javaObject('loci.formats.FileStitcher', r);
end
% Initialize the metadata store
OMEXMLService = javaObject('loci.formats.services.OMEXMLServiceImpl');
r.setMetadataStore(OMEXMLService.createOMEXMLMetadata());
% Initialize the reader
if ~isempty(id), r.setId(id); end
```

bfInitLogging.m

```
function bfInitLogging(varargin)
% BFINITLOGGING initializes the Bio-Formats logging system
%
%   bfInitLogging() initializes the logging system at WARN level by default.
%
%   bfInitLogging(level) sets the logging level to use when initializing
%   the logging system
%
% Examples
%
%    bfInitLogging();
%    bfInitLogging('DEBUG');
% OME Bio-Formats package for reading and converting biological file formats.
%
% Copyright (C) 2016 - 2021 Open Microscopy Environment:
%   - Board of Regents of the University of Wisconsin-Madison
%   - Glencoe Software, Inc.
%   - University of Dundee
%
% Redistribution and use in source and binary forms, with or without
% modification, are permitted provided that the following conditions are met:
%
% 1. Redistributions of source code must retain the above copyright
%    notice, this list of conditions and the following disclaimer.
%
% 2. Redistributions in binary form must reproduce the above copyright
%    notice, this list of conditions and the following disclaimer in
%    the documentation and/or other materials provided with the distribution.
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGE.
% Check Bio-Formats is set in the Java class path
bfCheckJavaPath();
% Input check
levels = {'ALL', 'DEBUG', 'ERROR', 'FATAL', 'INFO', 'OFF', 'TRACE', 'WARN'};
ip = inputParser;
ip.addOptional('level', 'WARN', @(x) ismember(x, levels));
ip.parse(varargin{:});
% Set logging level
```

```
javaMethod('enableLogging', 'loci.common.DebugTools', ip.Results.level);
```

bfopen.m

```
function [result] = bfopen(id, varargin)
% Open microscopy images using Bio-Formats.
%
% SYNOPSIS r = bfopen(id)
%          r = bfopen(id, x, y, w, h)
%
% Input
%    r - the reader object (e.g. the output bfGetReader)
%
%    x - (Optional) A scalar giving the x-origin of the tile.
%    Default: 1
%
%    y - (Optional) A scalar giving the y-origin of the tile.
%    Default: 1
%
%    w - (Optional) A scalar giving the width of the tile.
%    Set to the width of the plane by default.
%
%    h - (Optional) A scalar giving the height of the tile.
%    Set to the height of the plane by default.
%
% Output
%
%    result - a cell array of cell arrays of (matrix, label) pairs,
%    with each matrix representing a single image plane, and each inner
%    list of matrices representing an image series.
%
% Portions of this code were adapted from:
% http://www.mathworks.com/support/solutions/en/data/1-2WPAYR/
%
% This method is ~1.5x-2.5x slower than Bio-Formats's command line
% showinf tool (MATLAB 7.0.4.365 R14 SP2 vs. java 1.6.0_20),
% due to overhead from copying arrays.
%
% Thanks to all who offered suggestions and improvements:
%     * Ville Rantanen
%     * Brett Shoelson
%     * Martin Offterdinger
%     * Tony Collins
%     * Cris Luengo
%     * Arnon Lieber
%     * Jimmy Fong
%
% NB: Internet Explorer sometimes erroneously renames the Bio-Formats library
%     to bioformats_package.zip. If this happens, rename it back to
%     bioformats_package.jar.
%
% For many examples of how to use the bfopen function, please see:
```

```
%
https://docs.openmicroscopy.org/latest/bio-formats/developers/matlab-dev.html
% OME Bio-Formats package for reading and converting biological file formats.
%
% Copyright (C) 2007 - 2021 Open Microscopy Environment:
%   - Board of Regents of the University of Wisconsin-Madison
%   - Glencoe Software, Inc.
%   - University of Dundee
%
% Redistribution and use in source and binary forms, with or without
% modification, are permitted provided that the following conditions are met:
%
% 1. Redistributions of source code must retain the above copyright
%    notice, this list of conditions and the following disclaimer.
%
% 2. Redistributions in binary form must reproduce the above copyright
%    notice, this list of conditions and the following disclaimer in
%    the documentation and/or other materials provided with the distribution.
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGE.
% -- Configuration - customize this section to your liking --
% Toggle the autoloadBioFormats flag to control automatic loading
% of the Bio-Formats library using the javaaddpath command.
%
% For static loading, you can add the library to MATLAB's class path:
%     1. Type "edit classpath.txt" at the MATLAB prompt.
%     2. Go to the end of the file, and add the path to your JAR file
%        (e.g., C:/Program Files/MATLAB/work/bioformats_package.jar).
%     3. Save the file and restart MATLAB.
%
% There are advantages to using the static approach over javaaddpath:
%     1. If you use bfopen within a loop, it saves on overhead
%        to avoid calling the javaaddpath command repeatedly.
%     2. Calling 'javaaddpath' may erase certain global parameters.
autoloadBioFormats = 1;
% Toggle the stitchFiles flag to control grouping of similarly
% named files into a single dataset based on file numbering.
stitchFiles = 0;
% -- Main function - no need to edit anything past this point --
```

```
% load the Bio-Formats library into the MATLAB environment
status = bfCheckJavaPath(autoloadBioFormats);
assert(status, ['Missing Bio-Formats library. Either add bioformats_package.jar
'...
   'to the static Java path or add it to the Matlab path.']);
% Prompt for a file if not input
if nargin == 0 || exist(id, 'file') == 0
 [file, path] = uigetfile(bfGetFileExtensions, 'Choose a file to open');
 id = [path file];
 if isequal(path, 0) || isequal(file, 0), return; end
end
% Initialize logging
bfInitLogging();
% Get the channel filler
r = bfGetReader(id, stitchFiles);
% Test plane size
if nargin >=4
   planeSize = javaMethod('getPlaneSize', 'loci.formats.FormatTools', ...
                          r, varargin{3}, varargin{4});
else
   planeSize = javaMethod('getPlaneSize', 'loci.formats.FormatTools', r);
end
if planeSize/(1024)^3 >= 2,
   error(['Image plane too large. Only 2GB of data can be extracted '...
       'at one time. You can workaround the problem by opening '...
       'the plane in tiles.']);
end
numSeries = r.getSeriesCount();
result = cell(numSeries, 2);
globalMetadata = r.getGlobalMetadata();
for s = 1:numSeries
   fprintf('Reading series #%d', s);
   r.setSeries(s - 1);
   pixelType = r.getPixelType();
   bpp = javaMethod('getBytesPerPixel', 'loci.formats.FormatTools', ...
                    pixelType);
   bppMax = power(2, bpp * 8);
   numImages = r.getImageCount();
   imageList = cell(numImages, 2);
   colorMaps = cell(numImages);
   for i = 1:numImages
       if mod(i, 72) == 1
           fprintf('\n    ');
       end
       fprintf('.');
       arr = bfGetPlane(r, i, varargin{:});
       % retrieve color map data
       if bpp == 1
           colorMaps{s, i} = r.get8BitLookupTable()';
```

```
else
    colorMaps{s, i} = r.get16BitLookupTable()';
end
warning_state = warning ('off');
if ~isempty(colorMaps{s, i})
    newMap = single(colorMaps{s, i});
    newMap(newMap < 0) = newMap(newMap < 0) + bppMax;
    colorMaps{s, i} = newMap / (bppMax - 1);
end
warning (warning_state);
% build an informative title for our figure
label = id;
if numSeries > 1
    seriesName = char(r.getMetadataStore().getImageName(s - 1));
    if ~isempty(seriesName)
        label = [label, '; ', seriesName];
    else
        qs = int2str(s);
        label = [label, '; series ', qs, '/', int2str(numSeries)];
    end
end
if numImages > 1
    qi = int2str(i);
    label = [label, '; plane ', qi, '/', int2str(numImages)];
    if r.isOrderCertain()
        lz = 'Z';
        lc = 'C';
        lt = 'T';
    else
        lz = 'Z?';
        lc = 'C?';
        lt = 'T?';
    end
    zct = r.getZCTCoords(i - 1);
    sizeZ = r.getSizeZ();
    if sizeZ > 1
        qz = int2str(zct(1) + 1);
        label = [label, '; ', lz, '=', qz, '/', int2str(sizeZ)];
    end
    sizeC = r.getSizeC();
    if sizeC > 1
        qc = int2str(zct(2) + 1);
        label = [label, '; ', lc, '=', qc, '/', int2str(sizeC)];
    end
    sizeT = r.getSizeT();
    if sizeT > 1
        qt = int2str(zct(3) + 1);
        label = [label, '; ', lt, '=', qt, '/', int2str(sizeT)];
    end
```

```
        end
        % save image plane and label into the list
        imageList{i, 1} = arr;
        imageList{i, 2} = label;
    end
    % save images and metadata into our master series list
    result{s, 1} = imageList;
    % extract metadata table for this series
    seriesMetadata = r.getSeriesMetadata();
    javaMethod('merge', 'loci.formats.MetadataTools', ...
               globalMetadata, seriesMetadata, 'Global ');
    result{s, 2} = seriesMetadata;
    result{s, 3} = colorMaps;
    result{s, 4} = r.getMetadataStore();
    fprintf('\n');
end
r.close();
```

bfTestInRange.m

```
function test = bfTestInRange(x,name,maxValue)
%bfTestInRange A validation function that tests if the argument
% is scalar integer between 1 and maxValue
%
% This should be faster than ismember(x, 1:maxValue) while also producing
% more readable errors.
% OME Bio-Formats package for reading and converting biological file formats.
%
% Copyright (C) 2012 - 2021 Open Microscopy Environment:
%   - Board of Regents of the University of Wisconsin-Madison
%   - Glencoe Software, Inc.
%   - University of Dundee
%
% Redistribution and use in source and binary forms, with or without
% modification, are permitted provided that the following conditions are met:
%
% 1. Redistributions of source code must retain the above copyright
%    notice, this list of conditions and the following disclaimer.
%
% 2. Redistributions in binary form must reproduce the above copyright
%    notice, this list of conditions and the following disclaimer in
%    the documentation and/or other materials provided with the distribution.
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGE.
   % Check to see if x is a single value
   test = isscalar(x);
   if(~test)
       error('bfTestInRange:notScalar', ...
           [name ' value, [' num2str(x) '], is not scalar']);
   end
   % Check to see if x is a whole number
   test = mod(x,1) == 0;
   if(~test)
       error('bfTestInRange:notAnInteger', ...
           [name ' value, ' num2str(x) ', is not an integer']);
   end
   % Check to see if x is between 1 and maxValue
   test = x >= 1 && x <= maxValue;
```

```
    if(~test)
        error('bfTestInRange:notInSequence', ...
            [name ' value, ' num2str(x) ', is not between 1 and ', ...
            num2str(maxValue)]);
    end
end
```

branchDepth.m

```
function depthMap = branchDepth(bw,cb)
   branchPoints = bwmorph(bw,'branchpoints');
   branchPoints(cb) = false;

   bw(cb) = true;
   tmp = regionprops(cb,'centroid');
   start = fliplr(round(tmp.Centroid));
   toCheck = [start 0];
   depthMap = nan(size(bw));
   haveNotChecked = bw;
   while length(toCheck(:,1))>0
       do = toCheck(1,:);
       toCheck(1,:) = [];

       haveNotChecked(do(1),do(2)) = false;
       currentDepth = do(3);
       if branchPoints(do(1),do(2))
           currentDepth = currentDepth+1;
       end
       depthMap(do(1),do(2)) = currentDepth;
       checkMatX = nanmax(do(1)-1,1):nanmin(do(1)+1,length(bw));
       checkMatY = nanmax(do(2)-1,1):nanmin(do(2)+1,length(bw));
       tc = haveNotChecked(checkMatX,checkMatY);% works for square images bc
len assumes
       [x y] = meshgrid(checkMatX,checkMatY);
       couldCheck = [x(tc') y(tc') repmat(currentDepth,[nansum(tc(:)) 1])];
       if any(tc(:))
           if length(toCheck(:,1))==0
               toCheck = couldCheck;
           else
               toCheck = [toCheck;
couldCheck(~ismember(couldCheck(:,1:2),toCheck(:,[1:2]),'rows'),:)];
           end
       end
   end
end
```

checkP.m

```
function checkP(p)
   for i = find(ismember(p,'\/'))
       if ~isdir(p(1:i-1))
           mkdir(p(1:i-1))
       end
   end
end
```

getFilePaths.m

```
function p = getFilePaths(root,type)
   p = [];
   stack = dir(root);
   stack = cat(1,{stack(:).name});
   stack(ismember(stack,[{'.'} {'..'}])) = [];
   for i = 1:length(stack)
       stack{i} = [root '/' stack{i}];
   end
   while ~isempty(stack)
       if isdir(stack{1})
           add = dir(stack{1});
           add = cat(1,{add(:).name});
           add(ismember(add,[{'.'} {'..'}])) = [];
           for i = 1:length(add)
               add{i} = [stack{1} '/' add{i}];
           end
           stack = [stack add];
       elseif ismember({stack{1}(end-(length(type)-1):end)},{type})
           p = [p; stack(1)];
       end
       stack(1) = [];
   end
   for i = 1:length(p)
       p{i} = macheck(p{i});
   end
end
```

getProp.m

```
function prop = getProp(r)
   smth = imfilter(double(r),fspecial3('gaussian',[10 10 10],1),'same');
   mps = nanmax(smth,[],3);

   bw = bwskel(mps>0.1,'minbranchlength',20);
   bodyThresh = 0.9;
   tmp = regionprops(mps>bodyThresh,'area');
   cell_body = bwareaopen(mps>bodyThresh,nanmax(cat(1,tmp.Area)));
   branchPoints = bwmorph(bw,'branchpoints');
   branchPoints(cell_body) = false;
   nBranchPoints = nansum(branchPoints(:));
   bwcb = bw;
   bwcb(cell_body) = false;
   bwbp = bw;
   bwbp(bwmorph(branchPoints,'dilate')) = 0;
   branchEnds = bwmorph(bw,'endpoints');
   branchEnds = bwmorph(branchEnds,'dilate');
   dialatedBP = bwmorph(branchPoints,'dilate');
   tmp = bwcb;
   bdm = branchDepth(bw,cell_body);

   % Number of branch points \ Total volume \ Mean branch length \ Branch Depth
\ Cell body size
   figure(2)
   set(gcf,'position',[50 50 800 800])
   tmp = bwcb;
   tmp(cell_body) = 2;
   imagesc(tmp)
   axis equal
   axis off
   colormap bone
   prop = [nBranchPoints nansum(bwcb(:)) nansum(bwcb(:))./nBranchPoints
nanmax(bdm(:)) nansum(cell_body(:))];
end
```

Is_octive.m

```
% OME Bio-Formats package for reading and converting biological file formats.
%
% Copyright (C) 2014 - 2021 Open Microscopy Environment:
%   - Board of Regents of the University of Wisconsin-Madison
%   - Glencoe Software, Inc.
%   - University of Dundee
%
% Redistribution and use in source and binary forms, with or without
% modification, are permitted provided that the following conditions are met:
%
% 1. Redistributions of source code must retain the above copyright
%    notice, this list of conditions and the following disclaimer.
%
% 2. Redistributions in binary form must reproduce the above copyright
%    notice, this list of conditions and the following disclaimer in
%    the documentation and/or other materials provided with the distribution.
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGE.
function is = is_octave ()
is = exist ('OCTAVE_VERSION', 'builtin') == 5;
end
```

macheck.m

```
function in = macheck(in,forceMac)
   if nargin < 2 || isempty(forceMac)
       forceMac = false;
   end
   if ~forceMac && ispc
       in(ismember(in,'/')) = '\';
   else
       in(ismember(in,'\')) = '/';
   end
end
```

rtrace.m

```
function allProps = rtrace(z,rz,root)
   % Adjust location of skele plots here
   % Number of branch points \ Total volume \ Mean branch length \ Branch Depth
\ Cell body size
   fprintf(['\n\n*************************\nw/s=raise/lower threshold; ' ...
       '\na/d=previous/next ROI\np=process
ROI\tq=quit\n*************************\n\n'])
   done = false;
   thresh = 0.96;
   doRegion = 1;
   allProps = [];
   didChange = true;
   while ~done
       if didChange
           minPixIm = bwareaopen(z>thresh,5000); % minimum pixels
           lm = labelmatrix(bwconncomp(minPixIm));
       end
       figure(1)
       set(gcf,'position',[200 200 1000 500])
       subplot(1,2,1)
       imagesc(nanmax(rz,[],3))
       axis equal
       axis off
       subplot(1,2,2)
       colormap('jet')
       imagesc(nanmax(lm==doRegion,[],3))
       alpha(double(nanmax(lm~=0,[],3)))
       axis equal
       axis off
       w = waitforbuttonpress;
       if w
          str = get(gcf, 'CurrentCharacter');
       end

       if  str == 'w'
           thresh = thresh + 0.001;
           didChange = true;
       elseif str == 's'
           thresh = thresh - 0.001;
           didChange = true;
       elseif str == 'a'
           doRegion = nanmax(doRegion-1,1);
           didChange = false;
       elseif str == 'd'
           doRegion = nanmin(doRegion+1,nanmax(lm(:)));
           didChange = false;
       elseif str == 'p'
           allProps = [allProps; getProp(lm==doRegion)];
```

```
            figure(2)

            % Adjust Location of Skeleton Plots Here
            skeletonLocation = 'ExperimenterName/Morph_Skeletons/';

            saveFig(gcf,[skeletonLocation, slind(root,[1 3],'/\.') '/ROI_'
num2str(length(allProps(:,1)))],'tiff');
            close(2)
            didChange = false;
        elseif str == 'q'
            done = true;
            continue
        end
    end
    allProps = [repmat({slind(root,[1 3],'\/.')},[length(allProps(:,1)) 1])
num2cell([[1:length(allProps(:,1))]' allProps])];
end
```

saveFig.m

```
function saveFig(h,p,form)

   if ~iscell(form)
       form = {form};
   end
   checkP(p);

   ax = findall(h,'type','axes');
   set(ax,'fontname','arial','fontweight','normal','fontsize',9);
   drawnow;

   h.Renderer= 'Painters';
   checkP(p);
   STYLE = hgexport('readstyle','default');

   for i = 1:length(form)
       hgexport(h, [p '.' form{i}],STYLE, 'Format',form{i});
   end
end
```

slind.m

```
function ns = slind(s,num,marker)
   if nargin < 3 || isempty(marker)
       marker = '/\';
   end

   tmp = find(ismember(s,marker));
   if num(1)==0
       ns = s(1:tmp(num(2))-1);
   elseif num(2)==0
       ns = s(tmp(num(1))+1:end);
   else
       ns = s(tmp(num(1))+1:tmp(num(2))-1);
   end
end
```

znorm.m

```
function nz = znorm(z)
   nz = z;
   [a b] = sort(z(:),'ascend');
   nz(b) = [1:numel(z)]./numel(z);
end
```