Contents

```
function [im,params] = mtimport(varargin)
```

```
% MTIMPORT import and block-process Zeiss MultiTime LSM image sets
%
%   IM = MTIMPORT(PARAM1,VAL1,PARAM2,VAL2,...) imports each
%   LSM image in a given folder and processes them in blocks of a given
%   size, applying a given function to each block. If no function is
%   supplied, the concatenated, raw image data will be returned. The
%   results are then concatenated into the output matrix IM according to
%   the MultiTime parameters indicated in the LSM file names (as in a tile
%   scan).
%
%   Use optional name-value pairs to specify the following parameters:
%
%   'Path'      The folder from which to process all LSM images. If none is
%               specified, the user will be asked to select a folder
%
%   'BinSize'   The size, in microns, of the 2D bins. Note that bins are
%               square. The default is 50 microns.
%
%   'Ytiles'    The number of MultiTime tiles in the vertical direction. If
%               no value is given, all tiles are concatenated vertically.
%
%   'BlockFun'  The function to apply to each block of each image. If no
%               function is supplied, then IM will just be the raw,
%               concatenated image data.
%
%   IM is a multidimensional array, where the dimensions correspond to:
%
%   1 - Y, or image vertical
%   2 - X, or image horizontal
%   3 - Z, or image pages
%   4 - Color channel
%   5 - MultiTime repetition
%   6 - MultiTime group
%   7 - MultiTime block
%
%   [IM,PARAMS] = MTIMPORT(...) also returns the processing parameters into
%   the structure PARAMS.
%
%   Example
%       [im,params] = mtimport('Path','Example images','YTiles',2) ;
%       implay(im)
%
%   See also BLOCKPROC
```

```
%
%   Notes:
%   - Created and tested in MATLAB 2014b
%   - Uses parallel processing toolbox
%   - Requires BFMATLAB, the Bio-Formats matlab package available from
%     http://www.openmicroscopy.org/ (developed using Bio-Formats 5.2.1)
%   - Image pages (z steps) and color channels are processed individually
%     (i.e. separately)
%
%   Lena Bartell
%   Last edited: June 2015
```

Setup

```
% Parse inputs
params = parseInputs(varargin) ;
d = params.Path ;
binSize_um = params.BinSize ;
nY = params.Ytiles ;
blockFun = params.BlockFun ;
if isempty(blockFun), applyFun = false ;
else applyFun = true; end

% Gather information on LSM image files in the directory
files = dir([d '*.lsm']) ;
N = length(files) ;

% Parse file names to get tile size and row/column placement of each tile
R = cell2mat( cellfun( @(x) ...
    textscan(x,'%*s%u%*[^\n]','Delimiter',{'_R'}) , {files.name}' ) ) ;
L = cell2mat( cellfun( @(x) ...
    textscan(x,'%*s%u%*[^\n]','Delimiter',{'_L'}) , {files.name}' ) ) ;
G = cell2mat( cellfun( @(x) ...
    textscan(x,'%*s%u%*[^\n]','Delimiter',{'_GR'}) , {files.name}' ) ) ;
B = cell2mat( cellfun( @(x) ...
    textscan(x,'%*s%u%*[^\n]','Delimiter',{'_B'}) , {files.name}' ) ) ;
[nR,nL,nG,nB] = lena_deal( max([R,L,G,B]) ) ;
% if no y size supplied, assume all locations are aligned vertically
if isempty(nY), nY = nL ; end
nX = nL/nY ;
[col,row] = ind2sub( [nX nY], L ) ;

% Infer image information from example image:
evalc( 'r = bfGetReader( [d files(1).name ] );' ) ;
metadata = r.getSeriesMetadata();
nZ = metadata.get('DimensionZ') ;           % 1. number of pages / z-steps
nC = metadata.get('DimensionChannels') ;    % 2. number of color channels
umppx = metadata.get('VoxelSizeX') ;        % 3. pixel size (microns per pixel)
imWidth = metadata.get('DimensionX');       % 4. image width, in pixels
imHeight = metadata.get('DimensionY');      % 5. image height, in pixels
params.umppx = umppx ;

% Calculate block (2D bin) size
% binSize_px = 2^nextpow2( binSize_um / umppx ) ;
binSize_px = round( binSize_um / umppx ) ;
params.BinSize_actual = binSize_px * umppx ;
```

```
blockSize = [1 1] * binSize_px ;

% Infer more image information from example image:
tmp_im = imread( [d files(1).name], 1 ) ;   % example image
                                            % 6. data type of output and
if applyFun                                 % 7. X,Y size of output for each image file
    datatype_ex = blockproc( tmp_im, blockSize, blockFun, 'UseParallel', true );
    outSizeW = size(datatype_ex,2) ;        %
    outSizeH = size(datatype_ex,1) ;        %
else                                        %
    datatype_ex = tmp_im(1) ;               %
    outSizeW = imWidth ;                    %
    outSizeH = imHeight ;                   %
end                                         %
```

Process images in blocks, using parallel processing

```
% Initialize the output image
out = cell( N, 1 ) ;

parfor ii = 1:N % for each file

    % get the info of the current file
    info = imfinfo([d files(ii).name] ) ;

    % loop through the pages in the current file and process each page in
    % blocks, conserving the class & size of the output produced by blockFun
    blockIm = cast( zeros( outSizeH, outSizeW, nZ, nC ), ...
        'like', datatype_ex ) ;

    for jj = 1:2:length(info) % for each page (z-step)
        im = imread( info(1).Filename, jj, 'Info', info ) ;
        if applyFun
            for kk = 1:nC ; % for each color channel
                blockIm( :, :, ceil(jj/2), kk ) = ...
                    blockproc( im(:,:,kk), blockSize, blockFun,...
                    'UseParallel', true ) ;
            end
        else
            blockIm( :, :, ceil(jj/2), : ) = im ;
        end
    end

    % store the processed blocks and pages
    out{ii} = blockIm ;

end
```

Post process image according to tiles and MultiTime parameters

```
% initialize final image
im = cell( nY, nX, 1, 1, nR, nG, nB ) ;

% store parallel output appropriately
```

```
for ii = 1:N
    im( row(ii), col(ii), 1, 1, R(ii), G(ii), B(ii) ) = out(ii) ;
end

im = cell2mat(im) ;
```

```
params = 

           BinSize: 50
          BlockFun: []
              Path: 'Example images\'
            Ytiles: 2
             umppx: 1.3120
    BinSize_actual: 49.8567
```

Subfunction - Parse Inputs

```
function params = parseInputs(inputs)

% Setup input parser and possible name-value pairs
defaultBinSize = 50 ; % microns
p = inputParser ;
addParameter(p,'Path', [], @isdir )
addParameter(p,'BinSize', defaultBinSize, @isnumeric )
addParameter(p,'Ytiles', [], @(x)~rem(x,1) )
addParameter(p,'BlockFun', [], @(x)isa(x,'function_handle') )

% Parse inputs and store in a struct
parse(p,inputs{:}) ;
params = p.Results ;

% If no directory is supplied, ask the user to pick one and
% make sure it ends with a backslash (for ease of use later)
if isempty( params.Path )
    params.Path = uigetdir ;
end
if ~strcmp( params.Path(end), '\' )
    params.Path = [ params.Path '\' ] ;
end
```

Published with MATLAB® R2014b