```
% LFUtilProcessWhiteImages - process a folder/tree of white images by fitting a grid model to each
%
% Usage:
%
%     LFUtilProcessWhiteImages
%     LFUtilProcessWhiteImages( WhiteImagesPath )
%     LFUtilProcessWhiteImages( WhiteImagesPath, FileOptions, GridModelOptions )
%     LFUtilProcessWhiteImages( WhiteImagesPath, [], GridModelOptions )
%
% All parameters are optional and take on default values as set in the "Defaults" section at the top
% of the implementation. As such, this can be called as a function or run directly by editing the
% code. When calling as a function, pass an empty array "[]" to omit a parameter.
%
% As released, the default values are set up to match the naming conventions associated with Lytro
% files extracted using LFP Reader v2.0.0.
%
% Lytro cameras come loaded with a database of white images. These are taken through a diffuser,
% and at different zoom and focus settings. They are useful for removing vignetting (darkening near
% edges of images) and for locating lenslet centers in raw light field images.
%
% This function prepares a set of white images for use by the LF Toolbox. To do this, it fits a grid
% model to each white image, using the function LFBuildLensletGridModel. It also builds a database
% for use in selecting an appropriate white image for decoding a light field, as in the function
% LFLytroDecodeImage / LFSelectFromDatabase. The default parameters are set up to deal with the
% white images extracted from a Lytro camera.
%
% For each grid model fit, a display is generated allowing visual confirmation that the grid fit is
% a good one. This displays each estimated grid center as a red dot over top the white image. If
% the function was successful, each red dot should appear at the center of a lenslet -- i.e. near
% the brightest spot on each white bump. It does not matter if there are a few rows of lenslets on
% the edges of the image with no red markers.
%
%
% Inputs -- all are optional, see code below for default values :
%
%     WhiteImagesPath : Path to folder containing white images -- note the function operates
%                       recursively, i.e. it will search sub-folders. The white image database will
%                       be created at the top level of this path. A typical configuration is to
%                       create a "Cameras" folder with a separate subfolder of white images for each
%                       camera in use. The appropriate path to pass to this function is the top
%                       level of the cameras folder.
%
%     FileOptions : struct controlling file naming and saving
%                       .SaveResult : Set to false to perform a "dry run"
%                        .ForceRedo : By default, already-processed white images are skipped; set
%                                     this to true to force reprocessing of already-processed files
%           .WhiteImageDatabasePath : Name of file to which white image database is saved
%     .WhiteMetadataFilenamePattern : File search pattern for finding white image metadata files
%       .WhiteRawDataFnameExtension : File extension of raw white image files
%   .ProcessedWhiteImagenamePattern : Pattern defining the names of grid model files; must include
%                                     a %s which gets replaced by the white image base filename
%       .WhiteImageMinMeanIntensity : Images with mean intensities below this threshold are
%                                     discarded; this is necessary because some of the Lytro white
%                                     images include are very dark and not useful for grid modelling
%
%     GridModelOptions : struct controlling grid construction by LFBuildLensletGridModel
%             .FilterDiskRadiusMult : Filter disk radius for prefiltering white image for locating
%                                     lenslets; expressed relative to lenslet spacing; e.g. a
%                                     value of 1/3 means a disk filte with a radius of 1/3 the
%                                     lenslet spacing
%                          .CropAmt : Edge pixels to ignore when finding the grid
%                         .SkipStep : As a speed optimization, not all lenslet centers contribute
%                                     to the grid estimate; <SkipStep> pixels are skipped between
%                                     the lenslet centers that get used; a value of 1 means use all
%
% Output takes the form of saved grid model files and a white image database.
%
% See also: LFBuildLensletGridModel, LFUtilDecodeLytroFolder, LFUtilProcessCalibrations

% Part of LF Toolbox v0.4 released 12-Feb-2015
% Copyright (c) 2013-2015 Donald G. Dansereau

function LFUtilProcessWhiteImages( WhiteImagesPath, FileOptions, GridModelOptions )

%---Defaults---
WhiteImagesPath = LFDefaultVal( 'WhiteImagesPath', 'Cameras' );

FileOptions = LFDefaultField( 'FileOptions', 'SaveResult', true );
FileOptions = LFDefaultField( 'FileOptions', 'ForceRedo', false );
FileOptions = LFDefaultField( 'FileOptions', 'WhiteImageDatabasePath', 'WhiteImageDatabase.mat' );
FileOptions = LFDefaultField( 'FileOptions', 'WhiteMetadataFilenamePattern', '*MOD_*.TXT' );
FileOptions = LFDefaultField( 'FileOptions', 'WhiteRawDataFnameExtension', '.RAW' );
FileOptions = LFDefaultField( 'FileOptions', 'ProcessedWhiteImagenamePattern', '%s.grid.json' );
FileOptions = LFDefaultField( 'FileOptions', 'WhiteImageMinMeanIntensity', 500 );

GridModelOptions = LFDefaultField( 'GridModelOptions', 'Orientation', 'horz' );
GridModelOptions = LFDefaultField( 'GridModelOptions', 'FilterDiskRadiusMult', 1/3 );
GridModelOptions = LFDefaultField( 'GridModelOptions', 'CropAmt', 25 );
GridModelOptions = LFDefaultField( 'GridModelOptions', 'SkipStep', 250 );

DispSize_pix = 160; % size of windows for visual confirmation display
DebugBuildGridModel = false; % additional debug display

%---Load white image info---
fprintf('Building database of white files...¥n');
WhiteImageInfo = LFGatherCamInfo( WhiteImagesPath, FileOptions.WhiteMetadataFilenamePattern );

% The Lytro F01 database has two exposures per zoom/focus setting -- this eliminates the darker ones
F01Images = find(strcmp({WhiteImageInfo.CamModel}, 'F01'));
IsF01Image = false(size(WhiteImageInfo));
IsF01Image(F01Images) = true;
ExposureDuration = [WhiteImageInfo.ExposureDuration];
MeanDuration = mean(ExposureDuration(F01Images));
DarkF01Images = find(IsF01Image & (ExposureDuration < MeanDuration));
CamInfoValid = true(size(WhiteImageInfo));
CamInfoValid(DarkF01Images) = false;

%---Tagged onto all saved files---
TimeStamp = datestr(now,'ddmmmyyyy_HHMMSS');
GeneratedByInfo = struct('mfilename', mfilename, 'time', TimeStamp, 'VersionStr', LFToolboxVersion);

%---Iterate through all white images---
fprintf('Processing each white file...¥n');
fprintf('Visually confirm the estimated grid centers (red) are a good match to the lenslet centers...¥n');

for( iFile = 1:length(WhiteImageInfo) )
    [CurFnamePath, CurFnameBase] = fileparts( WhiteImageInfo(iFile).Fname );
    fprintf('%s [File %d / %d]:¥n', fullfile(CurFnamePath,CurFnameBase), iFile, length(WhiteImageInfo));

    ProcessedWhiteFname = sprintf( FileOptions.ProcessedWhiteImagenamePattern, CurFnameBase );
    ProcessedWhiteFname = fullfile(WhiteImagesPath, CurFnamePath, ProcessedWhiteFname);

    if( ~FileOptions.ForceRedo && exist(ProcessedWhiteFname, 'file') )
        fprintf('Output file %s already exists, skipping.¥n', ProcessedWhiteFname);
        % note that the file can still be added to the database, after the if/else/end,
        % unless it's a dark image as detected below
    else
        if( ~CamInfoValid(iFile) )
            fprintf('Dark F01 image, skipping and not adding to database¥n');
            continue;
        end

        %---Load white image and white image metadata---
        CurMetadataFname = fullfile(WhiteImagesPath, WhiteImageInfo(iFile).Fname);
        CurRawImageFname = LFFindLytroPartnerFile( CurMetadataFname, FileOptions.WhiteRawDataFnameExtension );

        WhiteImageMetadata = LFReadMetadata( CurMetadataFname );
        WhiteImageMetadata = WhiteImageMetadata.master.picture.frameArray.frame.metadata;
        ImgSize = [WhiteImageMetadata.image.width, WhiteImageMetadata.image.height];
        switch( WhiteImageInfo(iFile).CamModel )
            case 'F01'
                WhiteImage = LFReadRaw( CurRawImageFname, '12bit' );

                %---Detect very dark images in F01 camera---
                if( mean(WhiteImage(:)) < FileOptions.WhiteImageMinMeanIntensity )
                    fprintf('Detected dark image, skipping and not adding to database¥n');
                    CamInfoValid(iFile) = false;
                    continue
                end

            case 'B01'
                WhiteImage = LFReadRaw( CurRawImageFname, '10bit' );

            otherwise
                error('Unrecognized camera model');
        end

        %---Initialize grid finding parameters based on metadata---
        GridModelOptions.ApproxLensletSpacing = ...
            WhiteImageMetadata.devices.mla.lensPitch / WhiteImageMetadata.devices.sensor.pixelPitch;

        %---Find grid params---
        [LensletGridModel, GridCoords] = LFBuildLensletGridModel( WhiteImage, GridModelOptions, DebugBuildGridModel );
        GridCoordsX = GridCoords(:,:,1);
        GridCoordsY = GridCoords(:,:,2);

        %---Visual confirmation---
        if( strcmpi(GridModelOptions.Orientation, 'vert') )
            WhiteImage = WhiteImage';
        end
        ImgSize = size(WhiteImage);
        HPlotSamps = ceil(DispSize_pix/LensletGridModel.HSpacing);
        VPlotSamps = ceil(DispSize_pix/LensletGridModel.VSpacing);

        LFFigure(1);
        clf
        subplot(331);
        imagesc(WhiteImage(1:DispSize_pix,1:DispSize_pix));
        hold on
        colormap gray
        plot(GridCoordsX(1:VPlotSamps,1:HPlotSamps), GridCoordsY(1:VPlotSamps,1:HPlotSamps), 'r.')
        axis off

        subplot(333);
        imagesc(WhiteImage(1:DispSize_pix, ImgSize(2)-DispSize_pix:ImgSize(2)));
        hold on
        colormap gray
        plot(-(ImgSize(2)-DispSize_pix)+1 + GridCoordsX(1:VPlotSamps, end-HPlotSamps:end), GridCoordsY(1:VPlotSamps, end-HPlotSamps:end), 'r.')
        axis off

        CenterStart = (ImgSize-DispSize_pix)/2;
        HCenterStartSamps = floor(CenterStart(2) / LensletGridModel.HSpacing);
        VCenterStartSamps = floor(CenterStart(1) / LensletGridModel.VSpacing);
        subplot(335);
        imagesc(WhiteImage(CenterStart(1):CenterStart(1)+DispSize_pix, CenterStart(2):CenterStart(2)+DispSize_pix));
        hold on
        colormap gray
        plot(-CenterStart(2)+1 + GridCoordsX(VCenterStartSamps + (1:VPlotSamps), HCenterStartSamps + (1:HPlotSamps)), -CenterStart(1)+1 + GridCoordsY(
VCenterStartSamps + (1:VPlotSamps), HCenterStartSamps + (1:HPlotSamps)),'r.');
        axis off

        subplot(337);
        imagesc(WhiteImage(ImgSize(1)-DispSize_pix:ImgSize(1), 1:DispSize_pix));
        hold on
        colormap gray
        plot(GridCoordsX(end-VPlotSamps:end,1:HPlotSamps), -(ImgSize(1)-DispSize_pix)+1 + GridCoordsY(end-VPlotSamps:end,1:HPlotSamps), 'r.')
        axis off

        subplot(339);
        imagesc(WhiteImage(ImgSize(1)-DispSize_pix:ImgSize(1),ImgSize(2)-DispSize_pix:ImgSize(2)));
        hold on
        colormap gray
        plot(-(ImgSize(2)-DispSize_pix)+1 + GridCoordsX(end-VPlotSamps:end, end-HPlotSamps:end), -(ImgSize(1)-DispSize_pix)+1 + GridCoordsY(end-
VPlotSamps:end, end-HPlotSamps:end), 'r.')
        axis off

        truesize % bigger display
        drawnow

        %---Optionally save---
        if( FileOptions.SaveResult )
            fprintf('Saving to %s¥n', ProcessedWhiteFname);
            CamInfo = WhiteImageInfo(iFile);
            LFWriteMetadata(ProcessedWhiteFname, LFVar2Struct(GeneratedByInfo, GridModelOptions, CamInfo, LensletGridModel));
        end
    end
end

CamInfo = WhiteImageInfo(CamInfoValid);
%---Optionally save the white file database---
if( FileOptions.SaveResult )
    FileOptions.WhiteImageDatabasePath = fullfile(WhiteImagesPath, FileOptions.WhiteImageDatabasePath);
    fprintf('Saving to %s¥n', FileOptions.WhiteImageDatabasePath);
    save(FileOptions.WhiteImageDatabasePath, 'GeneratedByInfo', 'CamInfo');
end

fprintf('Done¥n');

end
```