Main Script for Digital Perfusion Phantom Using Cookie Cutter Method

By: Karthik Chellamuthu and Michael Mow

```
% Last updated: 2/27/2016

% This methods allows the user to define a set of regions of interest (ROI)
% and corresponding time attenuation curves for each ROI. The ROI is cut
% out of a head phantom image and the head phantom image and each ROI are
% forward projected. The projections are then added together while each ROI
% is scaled by a factor of its corresponding time attenuation curve.

% User can set parameters for ROIs and TACs or load existing .mat files

% Filesystem needed:
    % 'inData'   : holds input files
    % 'outData'  : holds output files

% Files needed:
    % 'head.mat' : matrix containing head phantom image

% Helper functions needed:
    % 'projections_generator.m'
    % 'ROI_generator.m'
    % 'transramp.m'
    % 'xyzfunc.m'
    % 'static_projector.m'
    % 'tac_generator.m'
    % 'alter_gvf.m'

clear all;

% 1 if creating new ROIs
% 0 if want to use existing ROIs stored in 'inData' folder
create_new_ROIs = 1;
filename = '2_23_16';

% 1 if creating new TACs
% 0 if want to use existing TACs stored in 'inData' folder
create_new_TACs = 1;


% directory to load files from
vnDir = 'inData';
% output directory
outDir = 'outData';

if create_new_ROIs == 1

    % PARAMETERS TO SET FOR ROI %
    % frame rate of scanner
    frame_rate = 1;

    % total scan time
    acquisition_time = 100;

    % radius of each ROI
    %ROI_radii = [30 30 30];
    ROI_radii = [30];

    % centers of corresponding ROIs
    % ROI_centers = [256 270 210; 256 270 150; 256 270 90];
    ROI_centers = [256 270 210];

    % call function to create projections
    % HEAD : struct containing projections of head phantom with ROIs cut out
    % ROI : struct containing projections of each ROI
    [HEAD,ROI]=projections_generator(frame_rate,acquisition_time,ROI_radii,ROI_centers,filename);

% else load exisiting files for ROIs
else
    vn = load([vnDir,'/head_projections_', filename, '.mat']);
    HEAD = vn.HEAD;
    vn = load([vnDir,'/ROI_projections_', filename, '.mat']);
    ROI = vn.ROI;
    clear vn;
    frame_rate = HEAD.frame_rate;
    acquisition_time = HEAD.acquisition_time;
end

%figure; imagesc(ROI(i).cutout(:,:,ROI(i).center)); colormap(gray); colorbar; axis image
```

```
if create_new_TACs == 1
    % PARAMETERS TO SET FOR TACs *

    % Set in vector format, 1 value for each ROI

    % max attenuation in HU
    max_atts = [30];

    % time delay before peak begins
    delays = [10];

    % time where max value occurs
    tmaxs = [25];

    % alpha param to specify shape
    alpha_params = [3];

    % call function to create time attenuation curves
    % TAC : struct containing all TACs
    TAC = tac_generator(frame_rate,acquisition_time,max_atts,delays,tmaxs,alpha_params,ROI,filename);

% else load exisiting files for ROIs
else
    vn = load([vnDir,'/TAC_', filename, '.mat']);
    TAC = vn.TAC;
    clear vn;
end

% number of iterations
nframes = frame_rate*acquisition_time;

projections = zeros(size(HEAD.projs));
max_values = zeros(nframes,1);


% Add together all projections with ROIs scaled by scale factor from TAC
for i = 1:nframes
    projections(:,:,i) = HEAD.projs(:,:,i);
    for j = 1:length(ROI)
        projections(:,:,i) = projections(:,:,i)+ROI(j).projs(:,:,i)*TAC(j).scale_factor(i);
    end
    y = projections(:,:,i);

    % calculate max value over each projection
    max_values(i) = max(y(:));
end

save([outDir,'/projections_' num2str(frame_rate) '_fps_' filename '.mat'],'projections');
```