```
function handles = mcaview_makeprofile(handles)
% function mcaview_makeprofile(handles)
% Generates profiles for profile plots.  Pays attention only to whether
% hold is on or off....
%
% Note that the center of mass is just the middle of the depth range for
% energy plots, but is the peak com (the peak energy) for depth plots
%
% ToDo: Figure out how to handle area plots...
%
% Notes on Gaussian fitting:
%    Tried a bunch of variations on Gaussian fitting to speed up the
%    results of background subtraction.  Fastest results (by about a factor
%    2) were obtained using lsqcurvefit.  A disadvantage of this technique
%    is that there is no built in facility for using weights.  I tried
%    emplying weights as an additional parameter to lsqcurvefit (additional
%    parameters beyond options are passed directly to the objective
%    function), but this slowed the function by almost a factor of 10.
%    (from 0.065 to 0.4 seconds for a particular test).  Note that using
%    user-supplied jacobians made very little difference in the time
%    (perhaps 5-10%)
%
%    Addendum 1: The factor of 10 turned out to be due to an increase in the
%    number of iterations for the fit to converge. This, in turn, was due
%    to the fact that parameter estimation doesn't work very well when the
%    model most return the model * the weights...
%
%    The best alternative to lsqcurvefit is fit, which does have a weights
%    option. However this was about a factor two slower (0.11-0.14 sec)
%    than the average time for lsqcurvefit.  On the other hand, for the
%    second part of background subtraction routine -- for which the center
%    and fwhm are fixed, we can use 'fit' to create a linear model in the
%    exponential term.  This is much faster than nonlinear fitting, and has
%    been implemented (with weights) below.
%
%    I am now using the curve fitting toolbox for both fits, as the best
%    compromise, since it is fairly fast and handles weights
%    properly and simply.
%
%    Next step would be to implement background subtraction for the area
%    plots.
%
%    Addendum 2: compiled version failed when using gaussbk and gauss
%    within the fittype definition for fitting (depth profiles).  They
%    could most likely be used if these two functions were compiled
%    separately, for example into a library.  As the quicker solution I
%    have simply written out the two functions explicitly.
%


type = handles.current_profile_type;

if ~isfield(handles.scandata.profiles, type)
    handles.scandata.profiles.(type) = [];
end

if handles.scandata.spec.dims > 1
    page = (get(handles.var3page, 'Value')-1) * handles.scandata.spec.size(2) + ...
        get(handles.var2page, 'Value');
else
    page = 1;
end

hold_on = (get(handles.profile_sethold, 'Value') == get(handles.profile_sethold, 'Max'));
```

```
% length(handles.d_roi) < 1 when importing new data.
% profiles = [] when switching from one type of profile to another.  In
% both cases, the existence of previously defined profiles should be
% checked...
if length(handles.d_roi) < 1
%       if isfield(handles.scandata, 'profiles') && isfield(handles.scandata.profiles, type) && ...
%           ~isempty(handles.scandata.profiles.(type))
%           if hold_on
%               saved_profiles = length(handles.scandata.profiles.(type));
%               handles.profiles = [handles.profiles handles.scandata.profiles.(type)];
%           else
%               handles.profiles = handles.scandata.profiles.(type);
%           end
%       else
%           handles.profiles = [];
%       end

    return
end

d_roi = handles.d_roi;
e_roi = handles.e_roi;
profile.type = type;

% handles.boxroi can be either 1 or 2 (not zero!).
box = get(handles.boxroi, 'Value') == 1;
bksub = get(handles.backsub, 'Value') == get(handles.backsub, 'Max');
dtcorr = get(handles.profile_dtcorr, 'Value') == get(handles.profile_dtcorr, 'Max');

image=handles.scandata.mcadata(:,:,page);
switch type
    case 'energy'
        % Calculate intensity vs. energy for a certain depth range
        if ~box
            e_roi = 1:size(image,1);
        end

        if length(d_roi)>1
            %               i_vs_d = sum(image(e_roi, d_roi));
            %               peak_data = find_peak(row(handles.scandata.depth(d_roi)), i_vs_d);
            %               profile.e_com = peak_data.com;
            y = sum(image(e_roi, d_roi)');
        else
            %               profile.com = handles.scandata.depth(d_roi);
            y = image(e_roi, d_roi)';
        end

        if dtcorr
            deadcorr = mean(handles.scandata.dtcorr(d_roi, page));
            if length(handles.scandata.dtdel) > 1
                dead_delta =  mean(handles.scandata.dtdel(d_roi, page));
            else
                dead_delta = handles.scandata.dtdel;
            end
        else
            deadcorr = 1;
            dead_delta = 1;
        end

        profile.y = y .* deadcorr;
        profile.delta = sqrt(y) .* dead_delta;
        profile.x = handles.scandata.energy(e_roi)';

        peak_data = find_peak(profile.x,profile.y);
        %profile.e_com = peak_data.com;
        profile.e_com = peak_data.com;
        profile.ch_com = handles.scandata.channels(e_roi(1))  - 1 + peak_data.ch_com;
        profile.fwhm = peak_data.fwhm;

        %profile.counts = peak_data.counts;
        %profile.delta = peak_data.delta;
    case 'depth'
        % Calculate intensity vs. depth for a certain energy range First,
        % get the center energy. find_peak just averages the first and last
        % points in the region to calculate background.  I use channel
        % number for the center for the purpose of calculating energy
        % calibration (in which case we want a channel center of mass to
        % associate with a particular energy. here might be a good place to
        % perform dead time correction...
        %         if ~box
        %             d_roi = 1:size(image,2);
        %         end
        if length(e_roi) == 1
            %               profile.ch_com = handles.scandata.channels(e_roi);
            %               profile.com = channel2energy(profile.ch_com, handles.scandata.ecal);
            profile.y = image(e_roi, d_roi); % This is not sufficient -- e.g. bksub will fail...
            % else
        end

        if length(d_roi) > 1
            i_vs_e = sum(image(e_roi, d_roi)')';
        else
            errdlg('For depth profile, ROI must contain more than one depth point');
            exit;
            %i_vs_e = image(e_roi, d_roi);
        end

        chan = handles.scandata.channels(e_roi);
        peak_data = find_peak(chan', i_vs_e');

        if ~box
            d_roi = 1:size(image,2);
            i_vs_e = sum(image(e_roi, d_roi)')';
        end

        del = sqrt(i_vs_e); del(find(del==0)) = 1;
        wts = (1./del).^2;

        if bksub

            dfe = length(chan) - 4;
            ftype = fittype('bk + area*2.35482/(fwhm*sqrt(2*pi))*exp(-0.5*((xdata-cen)*2.35482/fwhm).^2)' ,...
                'ind', 'xdata', 'coeff', {'area', 'bk', 'cen', 'fwhm'});
            %ftype = fittype('gaussbk([area bk cen fwhm], x)', 'ind', 'x', 'coeff', {'area', 'bk', 'cen', 'fwhm'});
            fitopts = fitoptions('Method', 'NonLinearLeastSquares', 'Display', 'off', ...
                'StartPoint', [peak_data.counts peak_data.bkgd peak_data.com  peak_data.fwhm],   ...
                'Weights', wts);

            [gaussfit, goodness, output] = fit(chan, i_vs_e,ftype, fitopts);

            %fval = sum((output.residuals.*wts).^2)/goodness.dfe

            fval = goodness.sse/goodness.dfe;
            %fval = sum((output.residuals).^2)/goodness.dfe

            profile.ch_com = gaussfit.cen;
            profile.e_com = channel2energy(profile.ch_com, handles.scandata.ecal);

            cen = gaussfit.cen;
            fwhm = gaussfit.fwhm;
            area = gaussfit.area;
            bk = gaussfit.bk;


            model = fittype({'2.35482/(fwhm*sqrt(2*pi))*exp(-0.5*((x-cen)*2.35482/fwhm).^2)' , '1'},...
                'problem', {'cen', 'fwhm'},'coeff', {'area', 'bk'});
            %model = fittype({'gauss([cen fwhm], x)', '1'}, 'problem', {'cen', 'fwhm'},'coeff', {'area', 'bk'});
            cf_opts = fitoptions(model);
            % tic;
            % iter = 0;
            progress = waitbar(0, 'Background Subtraction...Please Wait' );
            for k = 1:length(d_roi)
                i_vs_e = image(e_roi, d_roi(k));
                % dfe = length(chan)-2;
                %del = sqrt(i_vs_e); del(find(del==0)) = 1;
                del = i_vs_e; del(find(del==0)) = 1;
                wts = 1./del;
                set(cf_opts,  'Weights', wts, 'Lower', [0 0]);
                [foo, good,out] = fit(chan, i_vs_e, model, cf_opts,  'problem', {cen , fwhm}); %,cf_opts,'StartPoint',[area bk],'Lower', [0 0]

                y(k) = foo.area;
                chi(k) = good.sse/good.dfe;
                waitbar(k/length(d_roi), progress);
            end
            close(progress);
            %h = figure;
            %plot(chi);

        else % if bksub
            profile.ch_com = peak_data.com;
            profile.e_com = channel2energy(profile.ch_com, handles.scandata.ecal);
            y = sum(image(e_roi, d_roi));
        end % if bksub

        if dtcorr
            deadcorr = handles.scandata.dtcorr(d_roi, page)';
            if length(handles.scandata.dtdel) > 1
                dead_delta = handles.scandata.dtdel(d_roi, page)';
            else
                dead_delta = handles.scandata.dtdel;
            end
        else
            deadcorr = 1;
            dead_delta = 1;
        end

        profile.y = y.*deadcorr;
        profile.delta = sqrt(y) .* dead_delta;
        profile.x = row(handles.scandata.depth(d_roi));

        peak_data = find_peak(profile.x,profile.y);
        %profile.com = peak_data.com;
        profile.fwhm = peak_data.fwhm;
        %profile.fwhm = 1;
        %end  % if length(e_roi) == 1
    case 'area'
        % Code to generate an area profile.  But I need to make a decision
        % here regarding the format of profiles, and the interface with
        % cxfit.
        if ~box
            d_roi = 1:size(image,2);
        end
        i_vs_e = sum(image(e_roi, d_roi),2);
        peak_data = find_peak(handles.scandata.channels(e_roi)', i_vs_e');
        profile.ch_com = peak_data.com;
```

```
profile.e_com = channel2energy(profile.ch_com, handles.scandata.ecal);

image = handles.scandata.mcadata;
deadcorr = handles.scandata.dtcorr;
dead_delta = handles.scandata.dtdel;

if handles.scandata.spec.dims == 2
    delta = sqrt(image(:, d_roi,:));
    image = image(:, d_roi,:);
    if dtcorr
        [image, delta] = dtcorrect(image, delta, deadcorr, dead_delta, d_roi, 1:size(image,3));
    end
    var1 = handles.scandata.spec.var1(d_roi,:);
    var2 = handles.scandata.spec.var2(d_roi,:);
    var2page = get(handles.var2page, 'Value');
else % dims = 3
    n1 = handles.scandata.spec.size(1);
    n2 = handles.scandata.spec.size(2);
    n3 = handles.scandata.spec.size(3);
    n_fast = size(handles.scandata.mcadata, 3);

    var2page = get(handles.var2page, 'Value');
    var3page = get(handles.var3page, 'Value');

    slice = get(handles.slice_select,  'Value');
    switch slice
        case 1
            first = (var3page-1)*n2+1;
            n2 = get(handles.var2page, 'Max'); % In case last page is incomplete...
            last = first-1 + n2;
            page = var2page;
            delta = sqrt(image(:, d_roi,first:last));
            image = image(:,d_roi, first:last);
            if dtcorr
                [image, delta] = dtcorrect(image, delta, deadcorr, dead_delta, d_roi, first:last);
            end
            var1 = handles.scandata.spec.var1(d_roi, first:last);
            var2 = handles.scandata.spec.var2(d_roi, first:last);
        case 2
            if (n3-1)*n2 + var2page > n_fast
                n3 = n3-1;
            end
            first = var2page;
            last = first + n2 * (n3-1);
            % current_page = var3page;
            page = var3page;
            delta = sqrt(image(:, d_roi,first:n2:last));
            image = image(:,d_roi, first:n2:last);
            if dtcorr
                [image, delta] = dtcorrect(image, delta, deadcorr, dead_delta, d_roi, first:n2:last);
            end
            var1 = handles.scandata.spec.var1(d_roi, first:n2:last);
            var2 = handles.scandata.spec.var3(d_roi, first:n2:last);
        case 3
            if (n3-1)*n2 + var2page > n_fast
                errordlg('Currently-viewed page cannot be included in aerial profile' );
                return
            elseif n2*n3 > n_fast
                n3 = n3-1;
            end
            last = n2*n3;
            % current_page = (n3-1)*n2 + var2page;
            page = (n3-1)*n2 + var2page;
            delta = sqrt(image(:, d_roi,1:last));
            image = image(:,d_roi,1:last);
```

```
                    if dtcorr
                        [image, delta] = dtcorrect(image, delta, deadcorr, dead_delta, d_roi, 1:last);
                    end
                    delta = sqrt(sum(delta.*delta,2));
                    image = sum(image, 2);

                    % Check dimensions of image -- dimenion 2 singleton
                    % should remain
                    var1 = reshape(squeeze(handles.scandata.spec.var2(d_roi(1), 1:last)), n2, n3);
                    var2 = reshape(squeeze(handles.scandata.spec.var3(d_roi(1), 1:last)), n2, n3);
                end
            end

            if length(e_roi)==1
                z = squeeze(image(e_roi, :,:));
            else
                i_vs_e = sum(image(e_roi, :, page), 2);

                del = sqrt(i_vs_e); del(find(del==0)) = 1;
                wts = (1./del).^2;
                chan = handles.scandata.channels(e_roi);
                peak_data = find_peak(chan', i_vs_e');

                if bksub
                    dfe = length(chan) - 4;
                    ftype = fittype('bk + area*2.35482/(fwhm*sqrt(2*pi))*exp(-0.5*((xdata-cen)*2.35482/fwhm).^2)',...
                        'ind', 'xdata', 'coeff', {'area', 'bk', 'cen', 'fwhm'});
                    %ftype = fittype('gaussbk([area bk cen fwhm], x)', 'ind', 'x', 'coeff', {'area', 'bk', 'cen', 'fwhm'});
                    fitopts = fitoptions('Method', 'NonLinearLeastSquares', 'Display', 'off', ...
                        'StartPoint', [peak_data.counts peak_data.bkgd peak_data.com  peak_data.fwhm],  ...
                        'Weights', wts);

                    [gaussfit, goodness, output] = fit(chan, i_vs_e,ftype, fitopts);

                    fval = goodness.sse/goodness.dfe;

                    profile.ch_com = gaussfit.cen;
                    profile.e_com = channel2energy(profile.ch_com, handles.scandata.ecal);

                    cen = gaussfit.cen;
                    fwhm = gaussfit.fwhm;
                    area = gaussfit.area;
                    bk = gaussfit.bk;

                    model = fittype({'2.35482/(fwhm*sqrt(2*pi))*exp(-0.5*((x-cen)*2.35482/fwhm).^2)', '1'},...
                        'problem', {'cen', 'fwhm'},'coeff', {'area', 'bk'});
                    cf_opts = fitoptions(model);
                    nspectra = size(image, 3)*length(d_roi);

                    tic;
                    progress = waitbar(0, 'Background Subtraction...Please Wait');
                    for p = 1: size(image,3)
                        for k = 1:length(d_roi)
                            i_vs_e = image(e_roi, k,p);
                            del = sqrt(i_vs_e); del(find(del==0)) = 1;
                            wts = (1./del).^2;

                            set(cf_opts, 'Weights', wts, 'Lower', [0 0]);
                            [foo, good,out] = fit(chan, i_vs_e, model, cf_opts,  'problem', {cen , fwhm}); %,cf_opts,'StartPoint',[area bk],'Lower',
0]);
                            z(k,p) = foo.area;
                            waitbar(((p-1)*length(d_roi)+k)/nspectra, progress);
                        end
                    end
                    close(progress);
                    fprintf('Just closed wait bar\n' );
                    toc
                else
                    z = squeeze(sum(image(e_roi, :, :)));
                    delta = sqrt(z);
                end
            end


            profile.z = z;
            profile.delta = delta;

            if handles.scandata.spec.dims == 3
                profile.z = reshape(profile.z, size(var1));
                profile.delta = reshape(profile.delta, size(var1));
            end
            profile.y = var2;
            profile.x = var1;
            if get(handles.depth_abs, 'Value') == get(handles.depth_abs, 'Min')
                for k = 1:size(profile.x,2);
                    profile.x(:,k) =  profile.x(:,k) - profile.x(1, k);
                end
            end
        case 'volume'
            % Code to generate an area profile.  But I need to make a decision
            % here regarding the format of profiles, and the interface with
            % cxfit.
            if ~box
                d_roi = 1:size(image,2);
            end
            i_vs_e = sum(image(e_roi, d_roi),2);
            peak_data = find_peak(handles.scandata.channels(e_roi)', i_vs_e);
            profile.ch_com = peak_data.com;
            profile.e_com = channel2energy(profile.ch_com, handles.scandata.ecal);

            n1 = length(d_roi);
            n2 = handles.scandata.spec.size(2);
            n3 = handles.scandata.spec.size(3);
            if ~handles.scandata.spec.complete
                if n3 < 3
                    warndlg('Need at least two complete 2D cycles to do volume scan' );
                    return
                end
                n3 = n3 - 1;
            end
            n_fast = n3*n2;
            image = handles.scandata.mcadata(:,:,1:n_fast);
            var1 = handles.scandata.spec.var1(d_roi,1:n_fast);
            var2 = handles.scandata.spec.var2(d_roi,1:n_fast);
            var3 = handles.scandata.spec.var3(d_roi,1:n_fast);

            if page > n_fast
                errordlg('This spectra cannot be included in a volume plot. Please adjust var3 and try again' );
                return
            end

            if length(e_roi)==1
                z = squeeze(image(e_roi, d_roi,:));
            else
                i_vs_e = sum(image(e_roi, d_roi, page)')';

                del = sqrt(i_vs_e); del(find(del==0)) = 1;
                wts = (1./del).^2;
                chan = handles.scandata.channels(e_roi);
                peak_data = find_peak(chan', i_vs_e');
                if bksub
                    dfe = length(chan) - 4;
                    ftype = fittype('bk + area*2.35482/(fwhm*sqrt(2*pi))*exp(-0.5*((xdata-cen)*2.35482/fwhm).^2)' ,...
                        'ind', 'xdata', 'coeff', {'area', 'bk', 'cen', 'fwhm'});
                    %ftype = fittype('gaussbk([area bk cen fwhm], x)', 'ind', 'x', 'coeff', {'area', 'bk', 'cen', 'fwhm'});
                    fitopts = fitoptions('Method', 'NonLinearLeastSquares' , 'Display' , 'off', ...
                        'StartPoint', [peak_data.counts peak_data.bkgd peak_data.com  peak_data.fwhm],   ...
                        'Weights', wts);

                    [gaussfit, goodness, output] = fit(chan, i_vs_e,ftype, fitopts);

                    fval = goodness.sse/goodness.dfe;

                    profile.ch_com = gaussfit.cen;
                    profile.e_com = channel2energy(profile.ch_com, handles.scandata.ecal);

                    cen = gaussfit.cen;
                    fwhm = gaussfit.fwhm;
                    area = gaussfit.area;
                    bk = gaussfit.bk;


                    model = fittype({'2.35482/(fwhm*sqrt(2*pi))*exp(-0.5*((x-cen)*2.35482/fwhm).^2)' , '1'},...
                        'problem', {'cen', 'fwhm'},'coeff', {'area', 'bk'});
                    cf_opts = fitoptions(model);
                    for p = 1: size(image,3)
                        for k = 1:length(d_roi)
                            i_vs_e = image(e_roi, d_roi(k),p);
                            del = sqrt(i_vs_e); del(find(del==0)) = 1;
                            wts = (1./del).^2;

                            set(cf_opts,  'Weights', wts, 'Lower', [0 0]);
                            [foo, good,out] = fit(chan, i_vs_e, model, cf_opts,  'problem', {cen , fwhm}); %,cf_opts,'StartPoint',[area bk],'Lower',
    0]);
                            z(k,p) = foo.area;

                        end
                    end
                else
                    z = squeeze(sum(image(e_roi, d_roi, :)));
                end
            end

            if dtcorr
                deadcorr = handles.scandata.dtcorr(d_roi,:);
                if length(handles.scandata.dtdel) > 1
                    dead_delta = handles.scandata.dtdel(d_roi, :)';
                else
                    dead_delta = handles.scandata.dtdel;
                end
            else
                deadcorr = 1;
                dead_delta = 1;
            end
            profile.v = reshape(z.*deadcorr, n1, n2, n3);
            profile.delta = sqrt(profile.v) .* dead_delta;

            for k = 1:size(var1,2);
                var1(:,k) =  var1(:,k) - var1(1, k);
            end

            profile.x = reshape(var1,n1,n2,n3);
            profile.y = reshape(var2,n1,n2,n3);
            profile.z = reshape(var3,n1,n2,n3);
```

```
end


if hold_on && length(handles.scandata.profiles.(type))>0
    next = length(handles.scandata.profiles.(type))+1;
    handles.scandata.profiles.(type)(next) = profile;
else  % Hold is off...
    handles.scandata.profiles.(type) = profile;
end
handles.scandata_saved = 0;

function [image, delta] = dtcorrect(image, delta, deadcorr, dead_delta, d_roi, spectra)
% dtcorrect is a utility used by make_aplot, to modularize the computation
% of dead-time-corrected spectra.  image, and delta are already the correct
% dimensions, wherease deadcorr and dead_delta have the full dimensions of
% the data array. In other words, image has the dimensions deadcorr(d_roi,
% spectra).

deadcorr = deadcorr(d_roi, spectra);
dead_delta = dead_delta(d_roi, spectra);
for k = 1:d_roi*size(image, 3)
    image(:,k) = image(:,k) .* deadcorr(k);
    if length(dead_delta > 1)
        dead_delta = dead_delta(:, spectra);
        delta(:,k) = delta(:,k) .* dead_delta(k);
    end
end
```