Module 4

Sebastian Koranda

October 2025

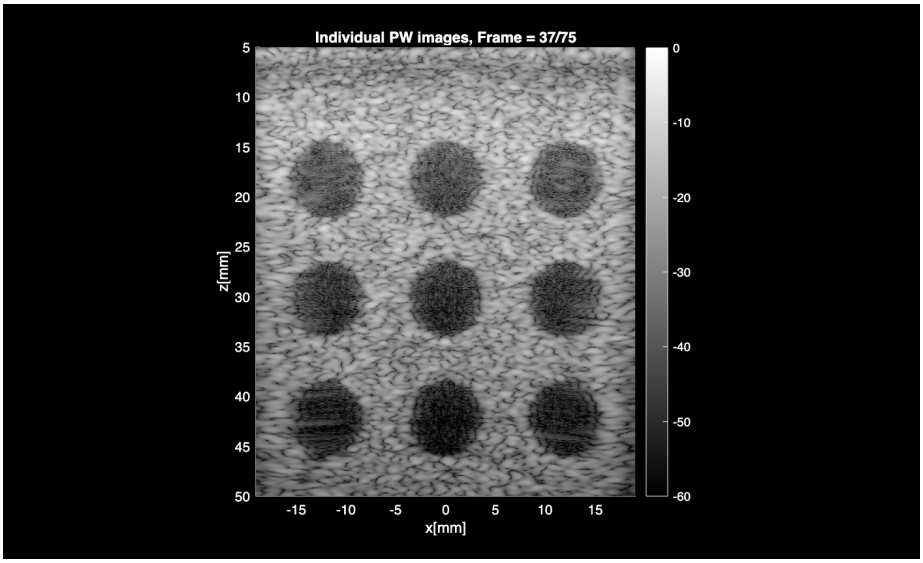


Figure 1: Single plane wave transmit image

1 Implementing Coherent and Incoherent Compounding

1.1 Setup

```
%% Getting the data
% We define the local path and the url where the data is stored

% data location
url='http://ustb.no/datasets/';  % if not found data will be downloaded from here
filename='PICMUS_experiment_resolution_distortion.uff';
filename='PICMUS_simulation_contrast_speckle.uff';

% checks if the data is in your data path, and downloads it otherwise.
% The defaults data path is under USTB's folder, but you can change this
% by setting an environment variable with setenv(DATA_PATH,'
    the_path_you_want_to_use');
tools.download(filename, url, data_path);

%% Loading channel data
channel_data=uff.read_object([data_path filesep filename],'/channel_data');

%% Defining an appropriate scan
scan=uff.linear_scan();
scan.x_axis = linspace(channel_data.probe.x(1),channel_data.probe.x(end),512)';
scan.z_axis = linspace(5e-3,50e-3,512)';

%% Set up the delay-and-sum beamforming using the midprocess
das = midprocess.das();
das.scan = scan;
das.channel_data = channel_data;
das.dimension = dimension.receive();
das.receive_apodization.window = uff.window.tukey50;
das.receive_apodization.f_number = 1.7;
das.transmit_apodization.window = uff.window.none;
b_data_all = das.go();

%% Finally, we will plot the individual plane wave images using the built in
% plot function in USTB
b_data_all.plot([],['Individual PW images'],[],[],[],[],'dark')
```

Listing 1: Loading data and beamforming setup

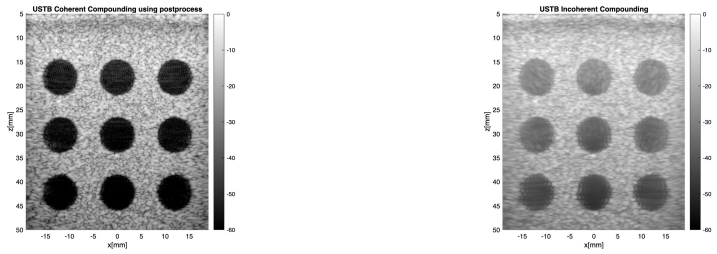
1.2 USTB implementation

```
% Using changing the dimension to "both" to get coherent compounding
das.dimension = dimension.both();
b_data_coherent = das.go();
b_data_coherent.plot([],['USTB Coherent Compounding using midprocess'])

% Alternatively one can use the coherent compounding postprocess object
cc = postprocess.coherent_compounding();
cc.input = b_data_all;
b_data_coherent = cc.go();
b_data_coherent.plot([],['USTB Coherent Compounding using postprocess'])
coherent_compounding_USTB = b_data_coherent.get_image();

% Using the postprocess incoherent_compunding
ic = postprocess.incoherent_compounding();
ic.input = b_data_all;
b_data_incoherent = ic.go();
b_data_incoherent.plot([],['USTB Incoherent Compounding'])
incoherent_compounding_USTB = b_data_incoherent.get_image();
```

Listing 2: USTB coherent and incoherent compounding



(a) USTB coherent compounding (b) USTB incoherent compounding

Figure 2: USTB coherent and incoherent compounding comparison

1.3 Own implementation

```
% First, let us get all the individual plane wave images in one matrix. The
% "none" argument means that we get the beamformed but complex data. Thus
% the data before envelope detection and log compression.
image_matrix = b_data_all.get_image('none');

% Extract single image
single_image = db(abs( ...
    image_matrix(:,:,37) ./ ...
    max(max(image_matrix(:,:,37))) ...
));

% Create coherent compounded image
coherent_compounding = db(abs( ...
    sum(image_matrix, 3) ./ ...
    max(max(sum(image_matrix, 3))) ...
));

% Create incoherent compounded image
incoherent_compounding = db(abs( ...
    sum(abs(image_matrix), 3) ./ ...
    max(max(sum(abs(image_matrix), 3))) ...
));
```

Listing 3: Own coherent and incoherent compounding implementation

2 Results Comparison

```
%% Compare your implementation of coherent compounding to the USTB
figure;
subplot(121)
imagesc(db(abs(coherent_compounding_USTB-coherent_compounding)));
colorbar
ax(1) = gca();
clim([-100 0])
title('Coherent compounding difference');

%% Compare your implementation of incoherent compounding to the USTB
subplot(122)
imagesc(db(abs(incoherent_compounding_USTB-incoherent_compounding)));
colorbar
ax(2) = gca();
clim([-100 0])
title('Incoherent compounding difference');
linkaxes(ax);
colormap jet
```

Listing 4: Plotting the difference between USTB and own implementation

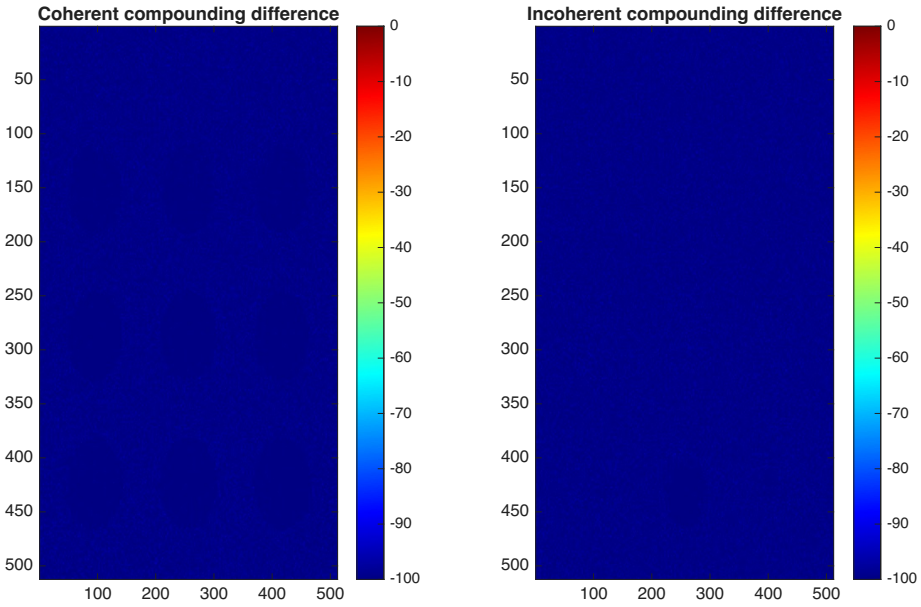


Figure 3: It is clear that the results from USTB and own implementation are identical to the given precision.

3 Implementing Mixed Compounding

```
% We can also do something inbetween full coherent and incoherent
% compounding. We can for example split the low quality images into two parts,
% and sum the different halfs coherently, before summing those two images
% incoherently. If you for example split the transmit angles into two as:

angles_first_sum = 1:channel_data.N_waves/2;
angles_second_sum = round(channel_data.N_waves/2)+1:channel_data.N_waves;

% And then sum the plane wave images resulting from the angles_first_sum
% coherently but separately, and then angles_second_sum coherently but separately
    .
% Before they both are combined incoherently. Thus you have done mix
% compounding. You can put the results in the mix_compounding variable:

first_half_coherent = sum(image_matrix(:, :, angles_first_sum), 3);
second_half_coherent = sum(image_matrix(:, :, angles_second_sum), 3);

mix_compounding = db(abs( ...
    (abs(first_half_coherent) + abs(second_half_coherent)) ./ ...
    max(max(abs(first_half_coherent) + abs(second_half_coherent))) ...
));
```

Listing 5: Mixed compounding implementation

3.1 Comparing all methods

```
figure()
subplot(221)
imagesc(scan.x_axis*1000, scan.z_axis*1000, single_image)
colormap gray; axis image;colorbar; caxis([-60 0])
title('Single transmit');
subplot(222)
imagesc(scan.x_axis*1000, scan.z_axis*1000, coherent_compounding)
colormap gray; axis image;colorbar; caxis([-60 0])
title('Coherent Compounding')
subplot(223)
imagesc(scan.x_axis*1000, scan.z_axis*1000, incoherent_compounding)
colormap gray; axis image;colorbar;  caxis([-60 0])
title('Incoherent Compounding')
subplot(224)
imagesc(scan.x_axis*1000, scan.z_axis*1000, mix_compounding)
colormap gray; axis image;colorbar;  caxis([-60 0])
title('Mix Compounding')
```

Listing 6: Plotting all compounding methods

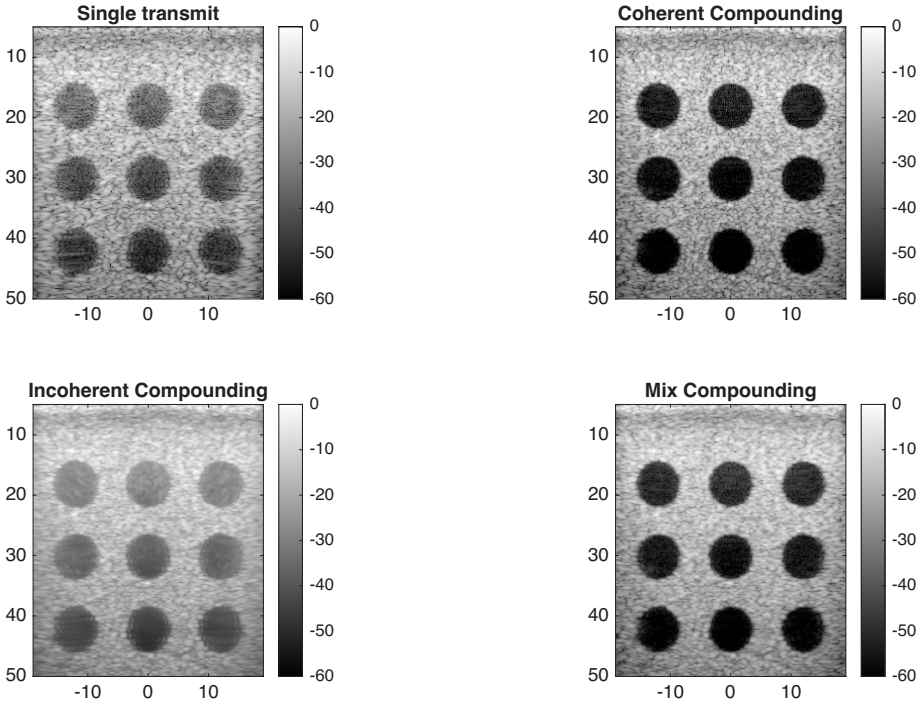


Figure 4: Resulting images

4 Resolution comparison

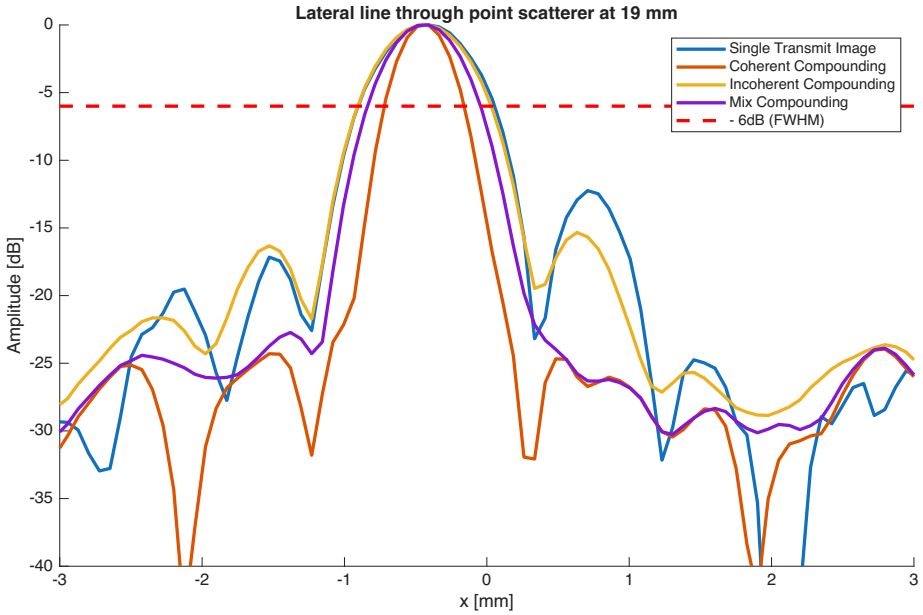


Figure 5: Resolution of different compounding methods

5 Contrast comparison

```
xc_ROI = -0;
zc_ROI = 30;
r_ROI = 3;
r_background_inner = 5;
r_background_outer = 7.5;


% Create masks to mask out the ROI of the cyst and the background.
for p = 1:length(scan.z_axis)
    positions(p,:,1) = scan.x_axis;
end

for p = 1:length(scan.x_axis)
    positions(:,p,2) = scan.z_axis;
end
points = ((positions(:,:,1)-xc_ROI*10^-3).^2) + (positions(:,:,2)-zc_ROI*10^-3)
    .^2;
idx_ROI = (points < (r_ROI*10^-3)^2);
idx_background_outer =  (((positions(:,:,1)-xc_ROI*10^-3).^2) + (positions(:,:,2)
    -zc_ROI*10^-3).^2 < (r_background_outer*10^-3)^2); %ROI speckle
idx_background_inner =  (((positions(:,:,1)-xc_ROI*10^-3).^2) + (positions(:,:,2)
    -zc_ROI*10^-3).^2 < (r_background_inner*10^-3)^2); %ROI speckle
idx_background_outer(idx_background_inner) = 0;
idx_background = idx_background_outer;

% Display the mask and the images with the mask applied of the background
% and ROI.
figure;
subplot(221)
imagesc(scan.x_axis*1000, scan.z_axis*1000, idx_background)
axis image; xlabel('x [mm]'); ylabel('z [mm]'); title('Background region')
subplot(222)
imagesc(scan.x_axis*1000, scan.z_axis*1000, idx_background.*single_image)
colormap gray; caxis([-60 0]); axis image; xlabel('x [mm]'); ylabel('z [mm]');
    title('Background image values')
subplot(223)
imagesc(scan.x_axis*1000, scan.z_axis*1000, idx_ROI)
axis image; axis image; xlabel('x [mm]'); ylabel('z [mm]'); title('ROI region')
subplot(224)
imagesc(scan.x_axis*1000, scan.z_axis*1000, idx_ROI.*single_image)
colormap gray; caxis([-60 0]); axis image; xlabel('x [mm]'); ylabel('z [mm]');
    title('ROI image values')
```

Listing 7: Masking

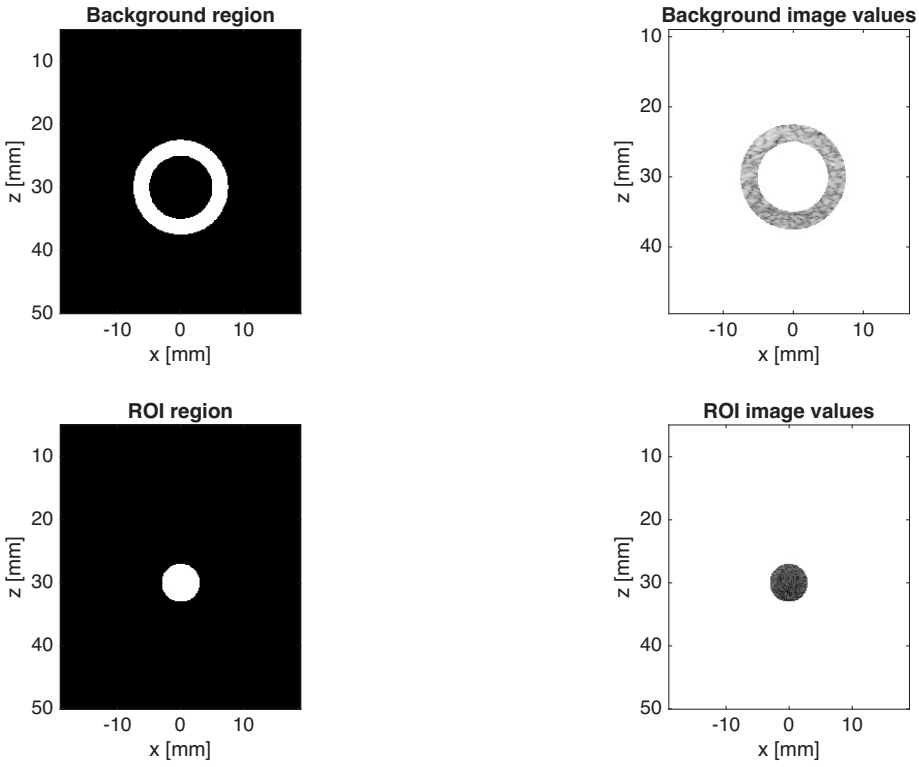


Figure 6: Masks for CR and CNR calculation

```
% First, let's get the images as one vector to easier "look up" the
% region of background and ROI
single_image_signal = image_matrix(:,:,37);
single_image_signal = single_image_signal(:);
coherent_compounding_signal = b_data_coherent.data;
incoherent_compounding_signal = b_data_incoherent.data;
mix_compounding_signal = abs(first_half_coherent) + abs(second_half_coherent);
mix_compounding_signal = mix_compounding_signal(:);

% Estimate the mean and the background of all images
mean_background_single = mean(abs(single_image_signal(idx_background(:))).^2);
mean_ROI_single = mean(abs(single_image_signal(idx_ROI(:))).^2);

mean_background_coherent = mean(abs(coherent_compounding_signal(idx_background(:)
    )).^2);
mean_ROI_coherent = mean(abs(coherent_compounding_signal(idx_ROI(:))).^2);

mean_background_incoherent = mean(abs(incoherent_compounding_signal(
    idx_background(:))).^2);
mean_ROI_incoherent = mean(abs(incoherent_compounding_signal(idx_ROI(:))).^2);

mean_background_mix = mean(abs(mix_compounding_signal(idx_background(:))).^2);
mean_ROI_mix = mean(abs(mix_compounding_signal(idx_ROI(:))).^2);

% Calculate Contrast Ratio
CR_single = 10*log10(mean_ROI_single/mean_background_single);
CR_coherent = 10*log10(mean_ROI_coherent/mean_background_coherent);
CR_incoherent = 10*log10(mean_ROI_incoherent/mean_background_incoherent);
CR_mix = 10*log10(mean_ROI_mix/mean_background_mix);

figure
bar([CR_single CR_coherent CR_incoherent CR_mix])
set(gca, 'YDir','reverse')
xticklabels({'Single Image','Coherent Compounded','Incoherent Compounding', 'Mix
    Compounding'})
ylabel('CR [dB]')
title('Contrast ratio comparison')
```

Listing 8: Calculating CR

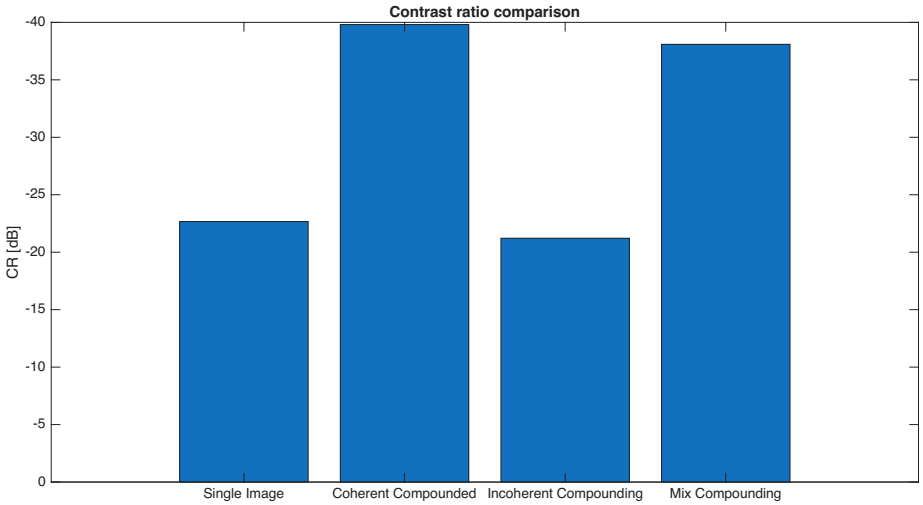


Figure 7: Contrast Ratio (CR) of different methods

```
% Calculate standard deviations
std_background_single = std(abs(single_image_signal(idx_background(:))).^2);
std_background_coherent = std(abs(coherent_compounding_signal(idx_background(:)))
    .^2);
std_background_incoherent = std(abs(incoherent_compounding_signal(idx_background
    (:))).^2);
std_background_mix = std(abs(mix_compounding_signal(idx_background(:))).^2);

% Calculate CNR
CNR_single = (mean_background_single - mean_ROI_single) / std_background_single;
CNR_coherent = (mean_background_coherent - mean_ROI_coherent) /
    std_background_coherent;
CNR_incoherent = (mean_background_incoherent - mean_ROI_incoherent) /
    std_background_incoherent;
CNR_mix = (mean_background_mix - mean_ROI_mix) / std_background_mix;

figure
bar([CNR_single CNR_coherent CNR_incoherent CNR_mix])
xticklabels({'Single Image','Coherent Compounded','Incoherent Compounding', 'Mix
    Compounding'})
ylabel('CNR')
title('Contrast-to-Noise Ratio Comparison')
```

Listing 9: Calculating CNR

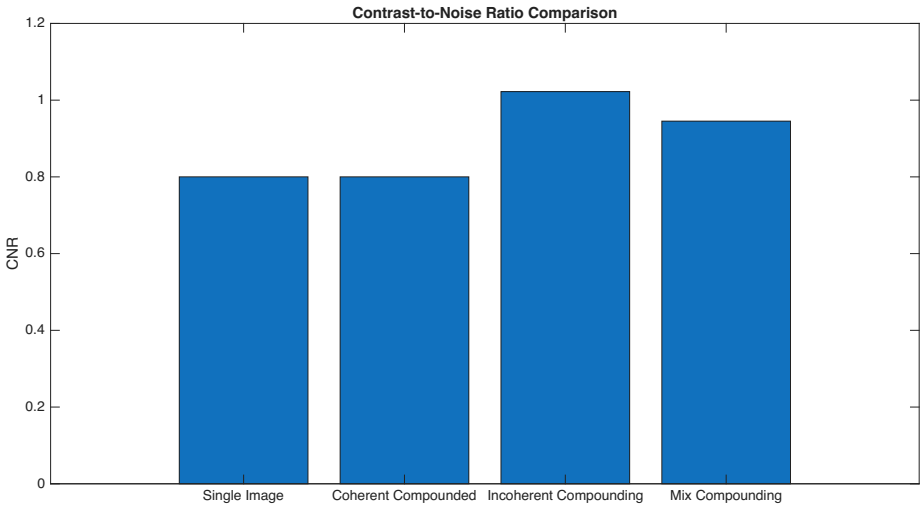


Figure 8: Contrast-to-Noise Ratio (CNR) of different methods