Module 3

Sebastian Koranda

October 2025

Beambased Beamformer

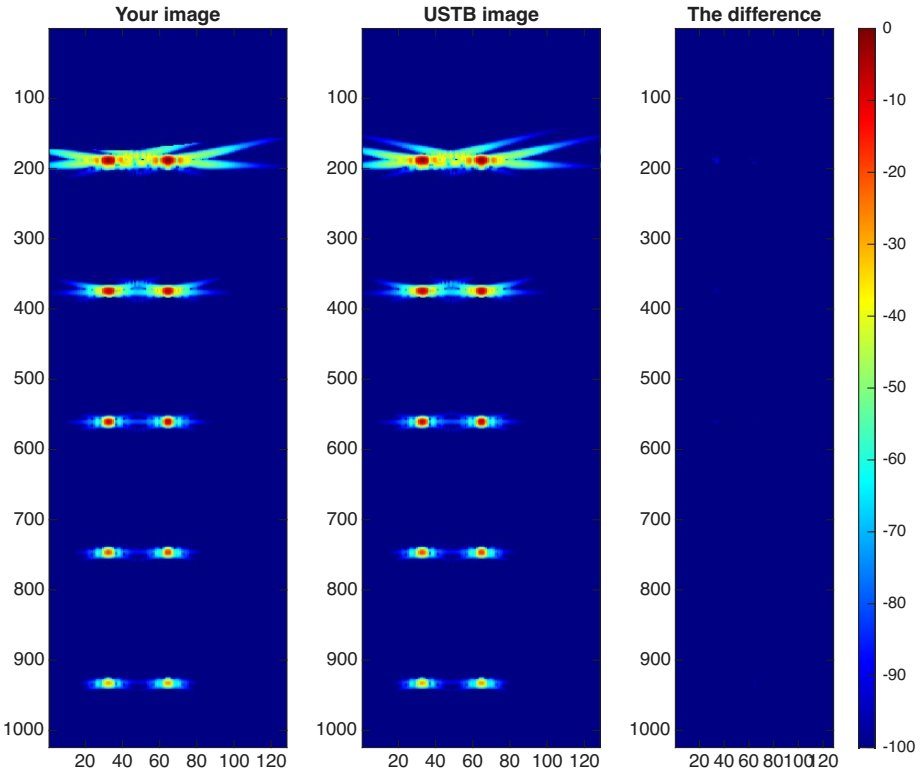

Figure 1: Comparison of beamformer implementations on point scatter data. USTB reference implementation in the middle. The code for the left image is shown below. The right image shows the difference between the two implementations.

```
N_depth = scan.N_depth_axis;                    % Number of depth pixels
N_transmits = channel_data.N_waves;             % Number of transmits
N_elements = channel_data.N_elements;           % Number of elements
x_element_position = channel_data.probe.x;      % The x-position of the elements
z_element_position = channel_data.probe.z;      % The z-position of the elements
c = channel_data.sound_speed;                   % Speed of sound
fs = channel_data.sampling_frequency;           % The sampling frequency
rfData = hilbert(channel_data.data(:,:,:,1)); % The analytical signal
sample_time = channel_data.time;                % The time in seconds per sample

% Preallocation of angle and offset vectors
angles = zeros(1, N_transmits);
offset = zeros(1, N_transmits);

for t = 1:N_transmits
    % Transmit angles (in radians)
    angles(t) = channel_data.sequence(t).source.azimuth;
    % Extract the offset for each transmit event
    offset(t) = channel_data.sequence(t).delay;
end

radial_distance = linspace(0,110e-3,N_depth);    % Depth (or radial distance)

delays = zeros(N_transmits, N_depth, N_elements);% Buffer for the delays
img = zeros(N_depth,N_transmits);                % Buffer for the final image

% Beambased phased array beamformer
for tx = 1:N_transmits

    % Calculate cartesian coordinates
    x_points = radial_distance .* sin(angles(tx));
    z_points = radial_distance .* cos(angles(tx));

    for rx = 1:N_elements
        % Distance from each point to the receive element
        receive = sqrt((x_points - x_element_position(rx)).^2 + ...
                    (z_points - z_element_position(rx)).^2);

        % Total delay with offset correction
        delays(tx,:,rx) = (radial_distance + receive) ./ c - offset(tx);

        % Interpolate (accumulating)
        img(:,tx) = img(:,tx) + interp1(sample_time, rfData(:,rx,tx)', delays(tx
    ,:,rx))';
    end
end

%Normalize the image to maximum value = 1
if max(img(:)) ~= 0
    img = img./max(img(:));
end
```

Listing 1: Beambased beamformer

Receive Apodization

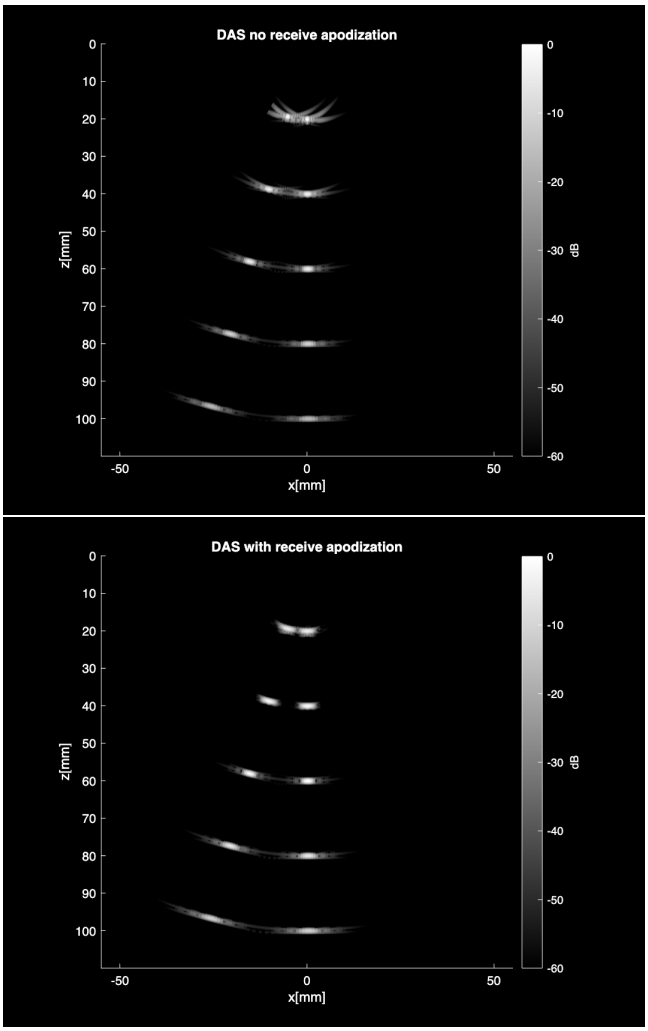

Figure 2: Point scatter image receive apodization. Top: Boxcar window. Bottom: Hamming window, f# = 2.

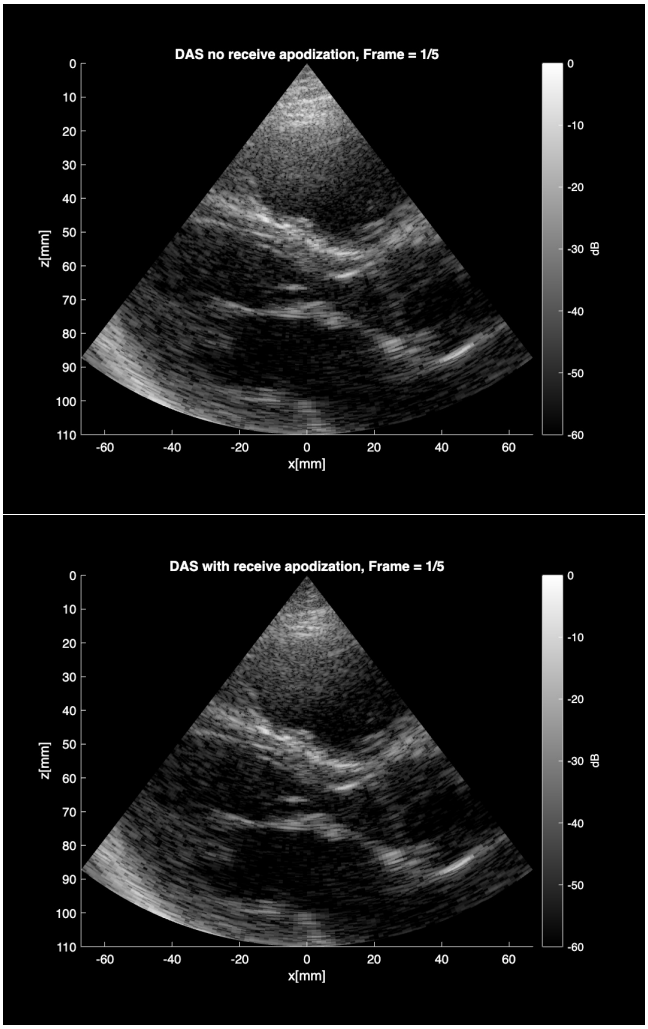

Figure 3: Cardiac image receive apodization. Top: Boxcar window. Bottom: Hamming window, f# = 2.

The receive apodization compenates for the varying resolution with depth. The f-number is defined as the ratio between the depth and the aperture size. A low f-number gives a better resolution, but also a higher side lobe level. The Hamming window reduces the side lobe level compared to the Boxcar window, but at the cost of a slightly worse main lobe resolution.

The effect is most apparent in the point scatter image, where the point spread function (PSF) can be observed. In the cardiac image, the effect is less pronounced, but still visible in the overall image quality. In both cases the effect is more pronounced at shallower depths, where the aperture size is smaller.