Denoising of Speckled Cardiac Ultrasound Images Using Wavelet

Safwan K Sami, Ahmed Khaleel, Muhamed Rojba, Ibrahim Rahman

December 6, 2024

Abstract

Cardiac Ultrasound Images (CUI) containing speckling can be detrimental to understanding and diagnosing cardiovascular patients' conditions. This paper will delve into the development of a tool designed to denoise CUI using MATLAB. The program designed by our team evaluates five wavelet mother functions across different thresholding types (Hard and Soft) and decomposition levels (2:4) to identify the most optimal combination for effective despeckling of CUI. To quantify the change in speckling between the original and denoised images we measured SNR, PSNR, and SSIM values across the various wavelet mother functions and thresholding types at different levels. Using these values we computed a score. The program keeps track of which mother function and level of decomposition has the highest composite score for soft and hard thresholding respectively. The program then identifies the two combinations with the highest score for both soft and hard thresholding and displays that as the most optimal wavelet for denoising the CUI and allows the user to choose which thresholding type they prefer. The wavelet mother function and parameters that most frequently provided the greatest denoising are bior6.8 at four levels of decomposition for soft thresholding and dmey at 4 levels of decomposition for hard thresholding. Thus our program proved to effectively utilize wavelets to produce results that denoised the speckled images.

Introduction

Ultrasound Images tend to contain small granular specks due to noise interference, called speckling. This speckling can obscure anatomy and make it difficult to analyze and diagnose patients based on their ultrasound results. There are various methods to denoise speckled images for example the usage of filters on ultrasound images can provide some despeckling but it results in the image becoming “overly smooth” **[1]**. As a result of our research, our group decided to explore an alternative method adjacent to filtering which led to our discovery of using wavelet transformations as a method of denoising.

Wavelets can be used for image smoothing, signal denoising, edge detection and more. For the purpose of this project, we will be using wavelets to denoise cardiac ultrasound images, while also preserving edges and overall image quality. This requires us to use different methods of thresholding. Pixel intensity gradients or frequency coefficients are zeroed out or coefficients at a set threshold are reduced. Small values typically correspond to smoother regions or less significant features in an image. We will allow the user to choose between hard and soft thresholding.

Methodology

Metrics

To compare the original ultrasound image and the final filtered image we used three different metrics. These metrics include PSNR, SNR, and SSIM. The PSNR (Peak Signal-to-Noise Ratio) is used to measure the quality of a reconstructed/compressed image or signal compared to the original. It quantifies the error introduced by compression, noise, or other distortions. The SNR (Signal-to–Noise-Ratio) quantifies how much of a signal's power is due to useful content versus noise. It is a measure of signal clarity and is used to evaluate the effectiveness of the filters used in removing noise from a noisy image **[2]**. The SSIM (Structural Similarity Index) is used to measure the structural similarity between two images **[3]**. Higher SNR and PSNR values indicate effective image denoising. Unlike PSNR or SNR, SSIM is designed to capture perceptual differences that align with human vision. This will help determine how much change has been made to the original image during the process of denoising.

Wavelet function

We chose 5 main mother functions to test. The five mother functions that are most commonly used for despeckling of images, and the ones we decided to use include: Daubechies Wavelet, Symlets Wavelet, Coiflet Wavelet, Biorthogonal Wavelet, and Discrete Meyer Wavelet. These wavelets are tested with different decomposition levels ranging from 2-4, and thresholds (soft and hard) to find the best possible PSNR, SNR, and SSIM values. With these metrics we are able to give each combination their respective scores and rank which is most suitable for denoising the selected image.

Thresholding (Soft and Hard)

Denoising the ultrasound image requires the use of thresholding. For an image, pixel intensity gradients or frequency coefficients are zeroed out or coefficients at a set threshold are reduced. Small values typically correspond to smoother regions or less significant features in an image. We will allow the user to choose between hard and soft thresholding.

Hard thresholding is a more aggressive method when compared to soft thresholding. It simply sets any value that is below a certain threshold to zero, and any values that are above the threshold remain unchanged. For smoother denoising, the user also has an option to choose soft thresholding. Soft thresholding is a gradual shrinkage process, setting values below the threshold to zero, but also shrinking the remaining values by the threshold amount. Hard thresholding is known as a “keep or kill” rule, while soft thresholding is known as a “shrink or kill” rule **[4]**. For this reason, soft thresholding is preferred over hard thresholding for smoother image denoising, especially in the case of wavelet-based denoising.

Ranking System

To determine which mother function and decomposition level was best for each thresholding method and for denoising the selected ultrasound image, we created a ranking system, ranking the best possible combination of the three parameters. There are 30 different possible combinations that can be used for denoising the image. The code will run 15 variations of each soft and hard thresholding (5 mother functions and 3 decomposition levels) to determine which combination is optimal for each threshold. To determine which combination was best, we used the three aforementioned metrics (SNR, PSNR, SSIM).

One of the issues with the ranking system was regarding the metrics themselves. The PSNR and SNR used the dB scale, while the SSIM ranged from values 0 to 1. This conflicted with the final rankings because the scales were different. To fix this issue, we linearized the PSNR and SNR values, and did the following calculations:

```
% Composite metric
metricScore = (10^(snrValue / 10) / 1000) / 3 + (10^(psnrValue / 10) / 1000) / 3 + (ssimValue / 3);
```

GUI/ User Input

To allow for user input during the despeckling process, we created a GUI that allows the user to choose between soft and hard thresholding. Therefore, the code will process 15 different variations of mother functions and decomposition levels for each threshold. After the code processes the ultrasound image with 30 different variations, and determines which combination is the most optimal, two filtered images will be displayed in a figure, each processed using their respective thresholds (soft and hard). The user will then be prompted to select which thresholding method they would like to use.

After the user selects their desired threshold, a secondary figure displays the original speckled ultrasound image and the final filtered image using the optimal mother function and decomposition level. At the bottom of the figure, the PSNR, SNR and SSIM values are also displayed. A tertiary figure also displays the optimal wavelet mother function used, as well as the decomposition levels.

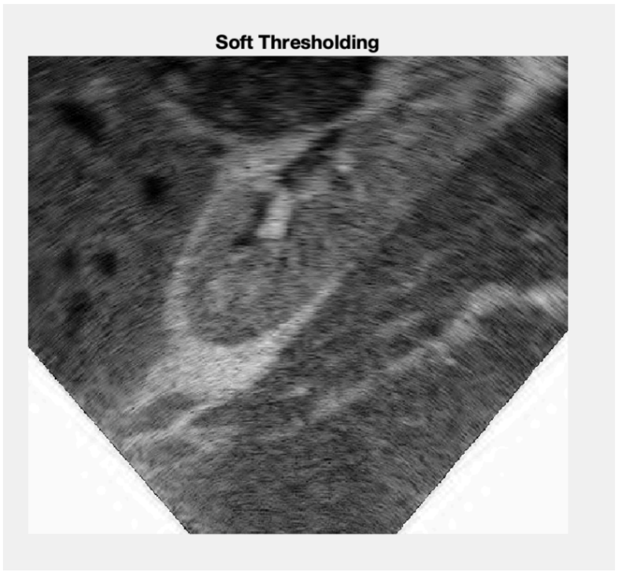
Results & Discussion

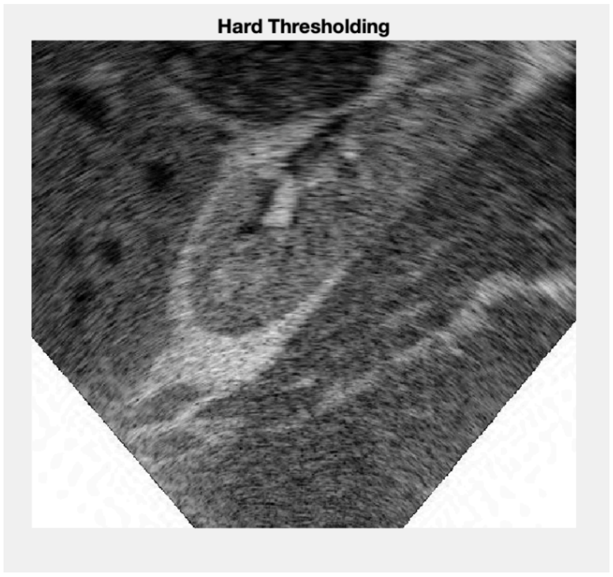
Soft Thresholding #1	**Hard Thresholding #1**
SNR: 23.52 dB | PSNR: 29.78 dB | SSIM: 0.8973 Wavelet: bior6.8 Decomposition Level: 4	SNR: 29.57 dB | PSNR: 35.84 dB | SSIM: 0.9690 Wavelet: dmey Decomposition Level: 4

Soft Thresholding #2 SNR: 47.54 dB | PSNR: 58.81 dB | SSIM: 0.9997 Wavelet: coif5 Decomposition Level: 4	**Hard Thresholding #2** SNR: 63.03 dB | PSNR: 74.30 dB | SSIM: 1.0000 Wavelet: coif5 Decomposition Level: 4
Soft Thresholding #3 SNR: 40.86 dB | PSNR: 50.06 dB | SSIM: 0.9952 Wavelet: bior6.8 Decomposition Level: 4	**Hard Thresholding #3** SNR: 49.89 dB | PSNR: 59.09 dB | SSIM: 0.9992 Wavelet: bior6.8 Decomposition Level: 4
Soft Thresholding #4 SNR: 36.29 dB | PSNR: 49.73 dB | SSIM: 0.9960 Wavelet: bior6.8 Decomposition Level: 4	**Hard Thresholding #4** SNR: 46.57 dB | PSNR: 60.01 dB | SSIM: 0.9994 Wavelet: bior6.8 Decomposition Level: 4

Soft Thresholding #5	**Hard Thresholding #5**
SNR: 31.03 dB | PSNR: 36.06 dB | SSIM: 0.9662 Wavelet: bior6.8 Decomposition Level: 4	SNR: 38.52 dB | PSNR: 43.55 dB | SSIM: 0.9892 Wavelet: bior6.8 Decomposition Level: 4

Cardiac Ultrasound Image 1 (Best Result)





Bior6.8 was scored as the best wavelet mother function to denoising the majority of the 5 cardiac ultrasound images chosen. The CUI in example one displayed the best visual denoising of the image, making the image smoother. SSIM values show that it was able to denoise the image significantly when compared to the other ultrasound images tested and although having a lower SSIM value of about 0.9, it was effective in denoising and preserving image structure.

The SSIM values did not accurately represent how well the image was denoised, mainly because the other 4 ultrasound images used contain predominantly low pixel intensity values (dark or black image). This can cause the SSIM to be relatively high for the darker images, when compared to the image which was used in example one (high range of pixel values).

The scoring method also determined the decomposition level of 4 to be the most optimal for denoising every CUI tested. Speckle noise can span multiple frequency bands, so a higher level can help suppress it more effectively. This is because the ultrasound images used contain noise with significant low frequency or large-scale noise.

Both soft and hard thresholds show that they were able to denoise the image effectively, while preserving structural detail and fidelity. Although the scoring method doesn't provide the user with the threshold, the user is able to choose between the two and compare PSNR, SNR and SSIM values to determine which method is the most optimal.

Conclusion

This paper demonstrated the denoising of cardiac ultrasound images using wavelet-based approaches, investigating five wavelet mother functions, two thresholding techniques, soft and hard, and decomposition levels from 2 to 4. Results indicated that Biorthogonal wavelet-Bior 6.8, decomposition level 4 with soft thresholding provided the best performance for effective denoising that maintained structural fidelity with significant reduction in speckle noise. The Discrete Meyer wavelet was selected at the same decomposition level as the one that worked best with hard thresholding. A scoring system of SNR, PSNR, and SSIM allowed a comparison among the best configurations; however, SSIM is observed underperforming when reflecting actual denoising quality for mainly dark images. This MATLAB tool allowed one to compare soft and hard thresholding results by providing quantitative metrics and visual clarity to decide on the most appropriate approach for despeckling, while retaining the quality and structural information of the image.

References

[1] Michailovich, Oleg V, and Allen Tannenbaum. "Despeckling of medical ultrasound images." *IEEE transactions on ultrasonics, ferroelectrics, and frequency control* vol. 53,1 (2006): 64-78. doi:10.1109/tuffc.2006.1588392

[2] Bouaynaya, Nidhal & Charif-Chefchaouni, Mohammed & Schonfeld, Dan. (2006). Spatially Variant Morphological Restoration and Skeleton Representation. IEEE transactions on image processing : a publication of the IEEE Signal Processing Society. 15. 3579-91. 10.1109/TIP.2006.877475.

[3] Ndajah, Peter & Kikuchi, Hisakazu & Yukawa, Masahiro & Watanabe, Hidenori & Muramatsu, Shogo. (2010). SSIM image quality metric for denoised images. International Conference on Visualization, Imaging and Simulation - Proceedings. 53-57.

[4] Khmag, Asem. (2013). Review of Image Denoising Algorithms Based on the Wavelet Transformation.

Appendix

Matlab Code:

```
% Main Wavelet Denoising Code with GUI Integration
% Define wavelet functions and decomposition levels
waveletFunctions = {'db1', 'sym6', 'coif5', 'bior6.8', 'dmey'};
decompLevels = 2:4;
thresholdTypes = {'soft', 'hard'}; % Thresholding types
% Initialize metrics for soft and hard thresholding
bestSoftMetrics = struct('wavelet', '', 'level', 0, 'snr', 0, 'psnr', 0,
'ssim', 0);
highestSoftMetricScore = -Inf;
bestHardMetrics = struct('wavelet', '', 'level', 0, 'snr', 0, 'psnr', 0,
'ssim', 0);
highestHardMetricScore = -Inf;
bestSoftImage = [];
bestHardImage = [];
% Load and preprocess original image
[fileName, filePath] = uigetfile('*.png', 'Select an Image for Denoising');
if isequal(fileName, 0)
   error('No file selected.');
end
originalImage = imread(fullfile(filePath, fileName));
if size(originalImage, 3) == 3
   originalImage = rgb2gray(originalImage); % Convert to grayscale if needed
end
originalImage = double(originalImage);
% Perform wavelet denoising with soft and hard thresholding
for i = 1:length(waveletFunctions)
   for level = decompLevels
        for t = 1:length(thresholdTypes)
            wavelet = waveletFunctions{i};
            thresholdType = thresholdTypes{t};
            % Decompose the image using wavelets
            [coeffs, sizes] = wavedec2(originalImage, level, wavelet);
            % Thresholding: Universal threshold
            noiseEstimate = median(abs(coeffs)) / 0.6745;
            threshold = noiseEstimate * sqrt(2 * log(numel(originalImage)));
            % Apply thresholding to detail coefficients
            coeffsThresh = coeffs;
            for j = 1:level
                idxH = sum(prod(sizes(1:j-1, :), 2)) + (1:prod(sizes(j, :)));
                idxV = idxH(end) + (1:prod(sizes(j, :)));
                idxD = idxV(end) + (1:prod(sizes(j, :)));
                H = coeffs(idxH);
                V = coeffs(idxV);
                D = coeffs(idxD);
                coeffsThresh(idxH) = wthresh(H, thresholdType(1), threshold);
                coeffsThresh(idxV) = wthresh(V, thresholdType(1), threshold);
                coeffsThresh(idxD) = wthresh(D, thresholdType(1), threshold);
            end
            % Reconstruct the denoised image
            denoisedImage = waverec2(coeffsThresh, sizes, wavelet);
            % Normalize images for metric computation
            normalizedOriginal = originalImage / 255;
            normalizedDenoised = denoisedImage / 255;
            % Compute metrics
            noise = originalImage - denoisedImage;
            snrValue = snr(originalImage, noise);
            psnrValue = psnr(normalizedDenoised, normalizedOriginal);
            ssimValue = ssim(normalizedDenoised, normalizedOriginal);
            % Composite metric
            metricScore = (10^(snrValue / 10) / 1000) / 3 + ...
                          (10^(psnrValue / 10) / 1000) / 3 + ...
                          (ssimValue / 3);
            % Update best configurations
            if strcmp(thresholdType, 'soft') && metricScore >
highestSoftMetricScore
                highestSoftMetricScore = metricScore;
                bestSoftMetrics = struct('wavelet', wavelet, 'level', level, ...
                                         'snr', snrValue, 'psnr', psnrValue,
'ssim', ssimValue);
                bestSoftImage = denoisedImage;
            end
            if strcmp(thresholdType, 'hard') && metricScore >
highestHardMetricScore
                highestHardMetricScore = metricScore;
                bestHardMetrics = struct('wavelet', wavelet, 'level', level, ...
                                         'snr', snrValue, 'psnr', psnrValue,
'ssim', ssimValue);
                bestHardImage = denoisedImage;
            end
        end
    end
end
% Display GUI for selecting and comparing filters
launchGUI(originalImage, bestSoftImage, bestHardImage, bestSoftMetrics,
bestHardMetrics);
% Function to launch the GUI
function launchGUI(originalImage, softImage, hardImage, softMetrics,
hardMetrics)
   f = figure('Name', 'Select Preferred Filter', 'NumberTitle', 'off',
'Position', [100, 100, 1200, 600]);
   % Display soft thresholding image
   softAx = axes('Parent', f, 'Position', [0.1, 0.4, 0.35, 0.5]); % Adjusted to
the left
   imshow(uint8(softImage), 'Parent', softAx);
   title(softAx, 'Soft Thresholding', 'FontSize', 12);
   % Display hard thresholding image
   hardAx = axes('Parent', f, 'Position', [0.55, 0.4, 0.35, 0.5]); % Adjusted
to the right
   imshow(uint8(hardImage), 'Parent', hardAx);
   title(hardAx, 'Hard Thresholding', 'FontSize', 12);
   % Add buttons for filter selection
   uicontrol('Style', 'pushbutton', 'String', 'Select Soft', ...
             'Position', [150, 50, 200, 40], ... % Aligned under the soft image
             'Callback', @(~, ~) displayResults(originalImage, softImage,
softMetrics));
   uicontrol('Style', 'pushbutton', 'String', 'Select Hard', ...
             'Position', [750, 50, 200, 40], ... % Aligned under the hard image
             'Callback', @(~, ~) displayResults(originalImage, hardImage,
hardMetrics));
end
% Function to display results
function displayResults(originalImage, filteredImage, metrics)
   % Display the original and filtered images
   f1 = figure('Name', 'Filtered Image and Metrics', 'NumberTitle', 'off',
'Position', [100, 100, 1000, 600]);
   % Original image
   subplot(1, 2, 1);
   imshow(uint8(originalImage));
   title('Original Image', 'FontSize', 14);
   % Filtered image
   subplot(1, 2, 2);
   imshow(uint8(filteredImage));
   title('Filtered Image', 'FontSize', 14);
   % Metrics
   annotation(f1, 'textbox', [0.3, 0.05, 0.5, 0.1], ...
              'String', sprintf('SNR: %.2f dB | PSNR: %.2f dB | SSIM: %.4f',
...
                                metrics.snr, metrics.psnr, metrics.ssim), ...
              'FontSize', 12, 'HorizontalAlignment', 'center', ...
              'EdgeColor', 'none');
   % Display a second figure for wavelet and decomposition level
   f2 = figure('Name', 'Wavelet Information', 'NumberTitle', 'off', 'Position',
[1200, 100, 400, 200]);
   % Show the chosen wavelet and decomposition level
   annotation(f2, 'textbox', [0.1, 0.3, 0.8, 0.4], ...
              'String', sprintf('Wavelet: %s\nDecomposition Level: %d',
metrics.wavelet, metrics.level), ...
              'FontSize', 14, 'HorizontalAlignment', 'center', ...
              'EdgeColor', 'none');
end
```