Wavelets as a De-Noising Approach of Cartilage Displacement Field Determined by MRI

David Francisco Toribio Carvajal
Utah State University

Follow this and additional works at: http://digitalcommons.usu.edu/honors

Part of the Electrical and Electronics Commons

Recommended Citation

This Thesis is brought to you for free and open access by the Honors Program at DigitalCommons@USU. It has been accepted for inclusion in Undergraduate Honors Theses by an authorized administrator of DigitalCommons@USU. For more information, please contact beckythoms@usu.edu.
WAVELETS AS A DENOISING APPROACH OF CARTILAGE DISPLACEMENT FIELDS DETERMINED BY MIRI

by

David Francisco Toribio Carvajal

Thesis submitted in partial fulfillment of the requirements for the degree of

DEPARTMENTAL HONORS in

Electrical Engineering
in the Department of Electrical and Computer Engineering

Approved:

Thesis/Project Advisor
Dr. Donald Cripps

Departmental Honors Advisor
Dr. Wynn Walker

Director of Honors Program
Dr. Christie Fox

UTAH STATE UNIVERSITY
Logan, UT

Spring 2010
Wavelets as a de-noising approach of cartilage displacement field determined by MRI.

Monday, May 3, 2010

David Toribio
Table of Figures

Project Overview.................................................................5
Monte Carlo Simulation........................................................8
Linear Displacement Field.....................................................9
Non Linear Displacement Field.............................................10
Strain Field........................................................................11
Noisy Displacement Field....................................................11
Wavelets............................................................................13
Wavelets Tree........................................................................16
Detail Coefficient Treshold..................................................17
Sample Size Experiment.....................................................21
Linear Field Tests...............................................................24
Non Linear Field Tests.......................................................28
Gantt Chart........................................................................28
1.0 Introduction

Tissue engineering was once categorized as a subfield of biomaterials, but having grown in scope and importance, it can be considered as a field in its own right. Tissue engineering studies the mechanical properties of tissues and the applications that repair or replace portions of or whole tissues. The analyses of the mechanical properties of tissues can help in the diagnosis of tissue diseases and in the monitoring of the progress of tissue treatments and replacements.

In order to study the mechanical properties of tissues, it is often required to repeatedly compress and decompress the tissue. This deformation process helps simulating the body forces exerted on the tissue. The analyses of the displacement of particles of the tissues during deformation can be useful for the study of the mechanical properties of the tissues [1]. An image that portrays the displacement of the particles during deformation is called displacement field. The phase data of MRI images of the cartilage during deformation are used to create displacement fields [2]. In the process of creating the displacement fields, noise is produced disturbing the data. Consequently, the analyses of the mechanical properties of the tissues are not as accurate as desired.

The purpose of this project is to analyze the effectiveness of wavelet-based filters on eliminating the noise that is produced in the process of generating the displacement fields.
Wavelet filters decompose signals into wavelet components, suppress the components representing high-frequency using detail coefficient threshold algorithms, and reconstruct a denoised signal from the components left. There are different variables involved in the designing of wavelet-based filters. This project focuses on analyzing the effect of the variation of two design variables on the denoising of displacement fields. These design variables are level of wavelet decomposition and detail coefficient threshold algorithms. This project aims to find an optimal combination of these two design variables. It also aims to compare wavelet-based filters to Fourier-based filters, which already have been tested for the denoising of displacement fields. To test the effectiveness of the wavelet-based filters, Monte Carlo simulations are used on MATLAB.

Two variables are used on measuring the effectiveness of the wavelet filters: bias and precision error. Bias represents the average difference of pixel intensities between de-noised images and original images. Precision error represents the standard deviation of pixel intensities between de-noised images [3].

A diagram that shows an overview of the process that this project is intended to enhance is shown below:
2.0 Review of Preliminary Design

The problem that this project aims to solve is to find an optimal combination of wavelet-based filter design variables that could lead to optimal noise elimination from displacement fields. The problem requires designing wavelet-based filters and testing them on displacement fields. It also requires that one determine and apply a procedure that can lead to establish comparisons between different wavelet-based filters design.

Some of the constraints to solve this problem are the capabilities of the computers available. In this project, we focus on 100 pixels to establish comparisons between de-noised images and original images. However, the displacement field images that are being used for the tests have 65536 pixels. Even though the use of 100 pixels allows to establish fairly good comparisons, the results will not be as accurate as using all points of the image.

2.1 Objectives

The objective of the research project is to find an optimal combination of wavelet-based filter design parameters for the de-noising of displacement fields. Monte Carlo simulations are used to test the filters using MATLAB. All the analyses that are done to study the performance of the filters will use the data obtained from the Monte Carlo simulations. Therefore, this project
only needs software implementation of the filters and of the Monte Carlo Simulations to achieve its objective.

The deliverables of the project are:

- Software implementation of wavelet-based filters and Monte Carlo Simulation.
- Data obtained by testing the filters using Linear Displacement fields.
- Data obtained by testing the filters using Non-Linear Displacement fields.
- Document showing the analyses of data.

3.0 Project Implementation

3.1 Monte Carlo Simulation

The Monte Carlo simulations that are used to test the filters consist of generating clean displacement fields, then adding Gaussian noise to the displacement fields, then filtering the displacement fields. Then, comparing the filtered displacement fields to the clean displacement fields by measuring the bias. Finally, comparing the filtered displacement fields by computing the precision. The MATLAB function created for implementing the Monte Carlo Simulation can be found in appendix 8.1. Explanations about how the bias and the precision are computed and their significance for evaluating filter effectiveness can be found on section 3.3 “Filter Assessment”.

A flow diagram that shows the process followed by this project to test and compare the wavelet based filters when applied to noisy displacement fields is the following:
The number "100" in the above diagram means that 100 noisy displacement fields are used to test each wavelet based filter. In order to determine this number of displacement fields to be used for testing each filter, first, 10 noisy MRI images of a cartilage were taken. Then, the average standard deviation across these 10 noisy images was measured. Then, noisy displacement fields were generated in MATLAB and it was found that 100 noisy displacement fields generated in MATLAB had very similar average standard deviation to the 10 noisy MRI displacement fields. Therefore, if this amount of MATLAB-generated displacement fields is used, the results obtained from the Monte Carlo Simulations will be a good representation of the results obtained if the displacement fields had been generated using the MRI device.

The types of displacement fields that were tested on the wavelet based filters are linear displacement fields and non-linear displacement fields.
Particles (pixels) in healthy tissues displace linearly during the deformation process. Therefore, in this research, linear fields are used to estimate the performance of wavelet based filters on healthy tissues.

A linear displacement field generated in MATLAB looks like the following picture:

![Linear Displacement Field](image_url)

The colored scale in this picture is used to determine the normalized amount of displacement achieved by each pixel of the image during the deformation process. It is a linear displacement field because the displacement is uniform in wide areas of the tissue and the variation in intensity of displacement between the different areas happens smoothly.

Particles in diseased tissues displace non-linearly. One reason of the non linear displacement of diseased tissues is that the region where the disease is present displaces differently to the healthy regions of the tissue. Also, during the deformation process of diseased
tissues, the particles displace in a curved shape. Therefore, in this research, non-linear fields are used to estimate the performance of wavelet-based filters on diseased tissues.

A non-linear displacement field generated in MATLAB looks like the following picture:

Biomedical engineers and doctors study the strain fields of tissues in order to analyze its mechanical properties, to diagnose tissue diseases and monitor treatments to tissue's diseases[2]. Strain fields are a geometrical measure of deformation representing the relative displacement between particles in a material body, i.e. a measure of how much a given displacement differs locally from a rigid-body displacement. Strain fields have a differential relationship to displacement fields.
An image of a strain field generated from a linear displacement field in MATLAB is shown in the next page.

Strain Field

After clean displacement fields are generated in the Monte Carlo simulation, random noise is added.

A noisy displacement field is shown below.

Noisy Displacement Field
After 100 noisy displacement fields are generated, wavelet based filters are used to de-noise the displacement fields.

3.2 Wavelets
3.2.1 Introduction

Just as a signal can be represented as a linear combination of sinusoids, a signal can as well be represented as a linear combination of wavelets components. There exist many different types of wavelets available for representing signals. The reason that wavelets can be used for representing signals is because shifted and scaled wavelets can form a basis for a signal space.

Images of the wavelet types used in this project are shown in the next page.

1st Order Daubechie

1st Order Coiflet
The steps for creating a wavelet transform can be summarized as follows:

1. Take a wavelet and compare it to a section at the start of the original signal.

2. Calculate a number, $C$, that represents how closely correlated the wavelet is with this section of the signal. The higher $C$ is, the more the similarity. More precisely, if the signal energy and the wavelet energy are equal to one, $C$ may be interpreted as a correlation coefficient.

3. Shift the wavelet to the right and repeat steps 1 and 2 until you've covered the whole signal.
4. Scale (stretch) the wavelet and repeat steps 1 through 3.

5. Repeat steps 1 through 4 for all scales.

### 3.2.2 Discrete Wavelet Transform

The process of computing a discrete wavelet transform is analogous to passing a signal through a high and low pass filter. The discrete wavelet transform produces approximation and details coefficients. The approximation coefficients are the amplitude of the higher scaled wavelets. Therefore, the approximation coefficients contain the information of the general characteristics of the signal. In the other hand, the details coefficients are the amplitude of the lower scaled wavelets and therefore, the details coefficients contain the information of localized characteristics of a signal (e.g. fast transients).
The wavelet decomposition of a signal can have different levels of wavelet decomposition. The levels of wavelet decomposition that a signal can have depend on the wavelet type used. The first level of wavelet decomposition is the initial decomposition of the signal into approximation and detail coefficients. Then, a second level of wavelet decomposition is the decomposition of the approximation coefficients produced by the initial level into approximation coefficients and detail coefficients. Then, the subsequent levels of wavelet decomposition represent the decomposition of the approximation coefficients produced by the previous level into approximation and detail coefficients. Below is an image that shows the wavelet decomposition of a signal over different levels.

![Wavelet Decomposition Tree](image)

### 3.2.3 Wavelet De-Noising

#### 3.2.3.1 De-Noising Scheme

This project follows a wavelet de-noising scheme proposed by Donoho and Johnstone in [4]. The scheme consists basically in three steps:

1) Transform the noisy image into an orthogonal domain by the 2D discrete wavelet transform.

2) Apply soft or hard thresholding to the detail coefficients.
3) Perform inverse wavelet transform to obtain the de-noised image.

The MATLAB function that implements this de-noising scheme can be found in appendix 8.2.

### 3.2.3.2 Detail Coefficient Tresholding

It is assumed that the noise of a signal is represented by the detail coefficients of the wavelet decomposition of a signal. Detail coefficient thresholding means using an algorithm to suppress specific detail coefficients. Detail coefficient thresholding methods generally classify into: soft and hard thresholding methods [5]. Hard thresholding methods set the value of any coefficient which absolute value is less than or equal to the threshold to zero. In soft thresholding, the threshold is subtracted from any coefficient which absolute value is larger than the threshold. In equation form, the soft and hard detail coefficient thresholding method are represented as follows [4]. Let $D$ be a detail coefficient thresholding function and having the threshold $\lambda$ and the coefficient $Y$ as argument. Then, the hard thresholding method is represented as:

$$D(Y, \lambda) = \begin{cases} 
Y & \text{if } ||Y|| > \lambda \\
0 & \text{otherwise}
\end{cases}$$

**Hard Detail Coefficient Treshold**

Also, using the same function and the same arguments, the soft detail coefficient thresholding methods can be represented as follows:
The detail coefficient thresholding methods used in this project are the methods provided by the MATLAB wavelet toolbox. The soft and the hard version of each method was used. The methods used in this project also classify into level dependent threshold methods and global threshold methods. Level dependent threshold methods measure the threshold to be used for the suppression of detail coefficients at each level of wavelet decomposition. Global threshold methods measure the threshold to be used for the suppression of detail coefficients from data of the original signal and use the same threshold at each level of wavelet decomposition.

The level dependent threshold methods used in this project are:

1. **Penalized High, Medium and Low.**

These strategies are based on a de-noising result by Birge and Massart in [6].

The threshold "T" applied to the detail coefficients is:

\[
T = \left| c(t^*) \right|
\]

with

\[
t^* = \arg \min \left[ -\sum c^2(k), k < t \right] + 2\nu t \left( \alpha + \log \left( \frac{n}{t} \right) \right); t = 1, ..., n
\]
where

- The sparsity parameter “a” should be greater than 1.
- The detail coefficients “c(k)” are sorted in decreasing order of their absolute value.
- $\nu$ is the noise variance

Three different intervals of choices for the sparsity parameter “a” are proposed in [6]:

- Penalized high, $2.5 \leq a < 10$
- Penalized medium, $1.5 < a < 2.5$
- Penalized low, $1 < a < 2$

2. Wavelet Shrinkage.

This method is described by Donoho and Johnstone in [4]. It is based on the following equation where, “t” is the threshold, “s” is the standard deviation of the noise of the level of wavelet decomposition and “[n,m]” are the dimensions of the signal:

$$t = s \sqrt{2 \log(n \cdot m)}$$

Wavelet Shrinkage

From the equation, it can be observed that this method is level dependent because the threshold used at each level of wavelet decomposition is dependent on the standard deviation of the noise at that specific level.
The global detail coefficient methods used in this project are:

1. **Wavelet Shrinkage**

   This method is described by Donoho and Johnstone in [4]. The equation used for determining the threshold value at each level is the same as shown in the previous page for the level dependent version of this method. However, in this method, the threshold value depends only in the standard deviation of the noise from the first level of coefficients.

2. **Penalized Birge Massart**

   This method is described by Birge and Massart in [6]. It is based on the same equations shown in the level dependent version of this method. However, in this method, only the variance of the noise in the first level of wavelet decomposition and the first level detail coefficients are used to compute the global threshold value.

### 3.3 Filter Assessment

The filters effectiveness is measured by using two variables: bias and precision. The bias is the average difference of pixel intensities between the 100 displacement filtered fields and the clean displacement field. The bias gives a sense of the potential of the filter to eliminate the noise from the displacement field generation process. The precision is the average standard deviation of the pixel intensities across the 100 filtered displacement fields used to test each
filter. The precision helps to give a sense of the consistency of the filter. In order to measure the bias and the precision, a region of interest of 100 pixels is used.

In order to determine the amount of pixels needed to accurately test the bias and precision of the filtered displacement fields, a sample size experiment was performed. The sample size experiment consisted on measuring the bias and precision of the filtered displacement fields by focusing on 9, 25, 100 and 2500 pixels of the filtered displacement fields. The 1st order Daubechie wavelet was used in this experiment. Below are graphs showing the filtered displacement field bias over the different detail threshold methods, level of wavelet decomposition and sample sizes. The legend related to the detail coefficient threshold methods shown in the graphs below and other similar graphs shown later in this report is shown in appendix 8.3.
Below are graphs showing the filtered displacement field precision over the different threshold methods, level of wavelet decomposition and sample sizes.
It can be observed that the filtered displacement field bias and precision graphs smooth out as the region of interest size increases. It can also be observed that there is not a significant difference between the results obtained when using 2500 pixels and 100 pixels. Therefore, since, using 100 pixels for the computations of the filtered displacement field bias and precision is more time efficient that when using 2500 pixels, then, 100 pixels was selected for the computations.

### 4.0 Tests

#### 4.1 Linear Field Tests

In order to perform these tests, three different discrete wavelets were used: 1st Order Dubechies, 1st Order Coiflets and 1.3 Order Biorthogonals. Below are shown the graphs of the filtered displacement field precision over the different detail threshold methods, level of wavelet decomposition and wavelet families.
Below are shown the graphs of the filtered displacement field bias over the different detail coefficient threshold methods, level of wavelet decomposition and wavelet families.
From the graphs shown above, it can be observed that the different wavelet families produce very similar filtered displacement field precision values for the different threshold levels.
method used and level of wavelet decomposition. However, it can be observed that coiflets produce better displacement field bias over the different detail coefficient threshold method and level of wavelet decomposition than the biorthogonal and the daubechie wavelet. It is the aim of our future work to analyze the differences in the properties of the coiflets wavelets to the biorthogonal and daubechie wavelets in order to understand better why they are more suitable for this application.

From the graphs shown above, it can also be observed that the best combination of filtered displacement field bias and precision are among the middle range (3rd and 4th) level of wavelet decomposition. This is because although the bias is lower at lower level of wavelet decomposition, the precision is higher at these levels. Also, although the precision is lower at higher level of wavelet decomposition, the bias is higher at these levels.

From the graphs shown above and the data obtained from these linear field tests found, it is not clear yet what detail coefficient threshold method would suit best this application of wavelet based filters. This is because even though soft threshold methods provide better precision error, they provide worse bias at each level of wavelet decomposition than hard threshold method. The difference in precision error between using soft or hard threshold method is similar to their difference in bias. Therefore, from linear tests it is still difficult to prefer one type of threshold method over the other.

From the linear field tests, it was also observed that the filtered displacement fields presented a high amount of discontinuities at each level of wavelet decomposition when using any of the different wavelet types used. We believe that much of this discontinuity is due to the
high presence of the Gibbs phenomenon on the wavelet reconstruction process of noisy signals as explained in [7] and [8].

Since the strain fields depend on the derivative of the displacement fields, the high amount of discontinuity in the filtered displacement fields do not allow for a proper computation of the strain fields. Donoho and Coifman in [7] describe a translation invariant scheme that can help reducing the Gibbs phenomenon, and consequently allow for the proper computation of the strain fields.

Non linear field tests were performed, but not completed, since it was not possible to compute strain fields from the filtered non linear displacement fields either.

Below are shown graphs of a noiseless non linear displacement field, its correspondent strain field. Also, below are shown graphs of a filtered displacement field (resultant from adding noise and de-noising the noiseless non linear displacement field) and its correspondent strain field using 1st order daubechie and the penalized low soft detail coefficient threshold method.
5.0 Project Management

The personnel involved in this project are Dr. Corey Neu, his graduate assistant Deva Chan and me. Neu and Chan have worked on the project for one month, and I have worked for two months. The budget of the project was our salary for the amount of time that we have worked on the project. Neu's salary was $5,000, Chan's salary was $1,666 and my salary was $4,000. Therefore, the budget totaled $10,666.
Other resources needed to develop the project were a computer and the MATLAB software. Each of the people involved in the project already had access to these two resources. Therefore, no money was allocated to obtain these resources.

The tasks and the schedule that were followed to achieve the objective of the project are summarized in the following Gantt Chart:

---

**6.0 Future Work**

The next tasks for completing this project will be to implement the translation invariant scheme proposed by Coifman and Donoho in [7] in order to reduce the Gibbs Phenomena presence in the filtered displacement field. Another task will be to do the linear field tests again and observe the behavior of the hopefully improved bias and precision of the filtered
displacement field when varying the wavelet filter design parameters (detail coefficient threshold method, level of wavelet decomposition and wavelet type). We also expect to complete the non linear field tests after the translation invariant scheme is implemented.

In the future, we also expect to compare the effectiveness of the wavelet based filters to fourier based filters on de-noising displacement fields.

7.0 Conclusion

The outcome of this project can greatly help the medical and the bio-medical engineering fields. More accurate studies of the mechanical properties of tissues will be possible. The results of this project might also lead to a diagnosis of tissue diseases, such as osteoarthritis, at an earlier stage and to enhance the application of tissue disease treatments.

The objective of this research project is to find an optimal combination of wavelet-based filters designed parameters that could lead to the maximum noise elimination of displacement fields. Eliminating noise will allow bio-medical engineers and orthopedist to better analyze the mechanical properties of tissues.

The approach to test these wavelet-based filters will be to run MATLAB Monte Carlo Simulations on them. Displacement fields will be generated, and then noise will be added to them. Noisy displacement fields will get filtered through the wavelet-based filters, and filtered displacement fields will be compared to the original displacement fields that were initially generated. The bias and precision of the filtered images will be measured in order to study the
effect of the wavelet-based filter design parameter variations on the filtered images. From the analysis of the bias and precision of the filtered images, we hope to select optimal combinations of the design parameters analyzed.

8.0 Appendix

8.1 Monte Carlo Simulation MATLAB function

function MCPrecBias = MCSim_precbias(tag,filt,iter,wtype,level,tmethod)
t0 = clock;
MCPrecBias = nan(1,1,4); % Predefine precision and bias values
mfil = 9; % max acceptable index for 'filt'

% Variation of Resolution [tag = 1, 2, or 3]
ress = [0.1, 0.2, 0.3];
mats = [256, 128, 128];
nois = [0.022218810, 0.015379824, 0.009885561];

% Variation of Number of Averages (SNR) [tag = 1, 2, or 4]
% ress = [0.2 0.2 0 0.2];
% mats = [128 128 0 128];
% nois = [0.024627686, 0.020002047, 0, 0.015580764];

if filt > mfil || filt < 0 % default filter if inappropriate choice is made
    filt = 0;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%% Initial Definitions
matrix = [mats(tag) mats(tag)];
ImageRes = [ress(tag) ress(tag)];
nstd = nois(3);

DefObjSize = [18 18]; % Object Size [mm]
Strain = [0.05 -0.06]; % Strain [x (horizontal), y (vertical)]

MCIter = 1; % number of Monte Carlo simulations

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%% Create Simulated Objects and "True" Displacement based on Strains

31
% linear field
[dXsim, dYsim, IMsim] = CreateDSIM(matrix, ImageRes, DefObjSize, Strain);

% nonlinear, custom field
[dXsim, dYsim, IMsim] = CreateNLDSIM(matrix, ImageRes, DefObjSize, Strain);

discont = 0; % set to 1 to add discontinuity to displacement field
if discont
    discROI = zeros(matrix); discROI(41:70,56:71) = 1; % define shape of ROI at (41:70,56:71)
    dYsim(find(discROI)) = 0.1+dYsim(find(discROI)); % introduce an added discontinuity
end

% Select Points for Precision and Bias Computations
fullroi = 0; % set to 1 to analyze all points on interior of ROI;
if fullroi
    fprintf('Analyzing the Full ROI
');
    [nI, nJ] = find(bwmorph(IMsim, 'erosion', 2)); % works similar to FindObject w/ B-splines mod
else
    [nI, nJ] = FindPts(10, IMsim); % 10 points selected for strains
    % NOTE: FindPts only works well for rectangular ROI. Perhaps switch to a % function that picks X random points instead, since these points would % then be different for each Monte Carlo simulation
end

pI = nI; pJ = nJ; % choose same points for displacements as for strains

strainson = 1; % also compute precision and bias for strains
if strainson
    fprintf('Calculating Ideal Strains
');
    ssstrain = Disp2Strain(dXsim, dYsim, matrix, ImageRes, nI, nJ);
else
    fprintf('Strains NOT Calculated
');
end

% Plot Ideal Displacements
% figure, imagesc(dXsim), colormap('jet'), axis('square'); colorbar; title('Original dx')
% if ~fullroi, hold on; plot(pJ, pI, 'yx'); hold off; end
figure, imagesc(dYsim), colormap('jet'), axis('square'); colorbar; title('Original dy')
if ~fullroi, hold on; plot(pJ, pI, 'wx'); hold off; end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Monte-Carlo Simulation
noisy = 1; % 1 = noise, 0 = noiseless
for mc = 1:MCIter
    if mc == 1 || rem(mc, 10) == 0
        fprintf(1, '	Iteration %i of %i
', mc, MCIter);
    end

% Add on Random Error...
dX = dXsim + noisy * normrnd(0, nstd, matrix(1), matrix(2));
dY = dYsim + noise * normrnd(0, nstd, matrix(1), matrix(2));
    if mc == 1
        figure, imagesc(dX), colormap('jet'), axis('square'); colorbar; title('Noisy dX')
        if ~fullroi, hold on; plot(pJ, pI, 'yx'); hold off; end
        figure, imagesc(dY), colormap('jet'), axis('square'); colorbar; title('Noisy dY')
        if ~fullroi, hold on; plot(pJ, pI, 'wx'); hold off; end
    end;

% Smooth Displacement Fields
% Basic 2-D averaging filters
    if filt < 5 && iter > 0
        [dXs, dYs] = sDisp_filt2D(dX, dY, IMSim, pI, pJ, filt, iter, level);
    end

% B-spline bases for cubic or quadratic, open, periodic bases
% B-spline methods only smooth around points of interest
    if filt == 5 || filt == 6 && iter > 0
        ord = filt - 2;
        dXs = dX; dYs = dY;
        for ii = 1:iter
            [dXs] = sDisp_bSpline44(dXs, pI, pJ, ord);
            [dYs] = sDisp_bSpline44(dYs, pI, pJ, ord);
        end
    end

% Generic FIR filters
    if filt == 7 || filt == 8 && iter > 0
        [dXs, dYs] = sDisp_filt2D(dX, dY, IMSim, pI, pJ, filt, iter, level);
    end

% Wavelets
    if filt == 9 && iter > 0
        dXs = wavefilt(dX, wtype, level, tmethod);
dYs = wavefilt(dY,wtype,level,tmethod);
dXs = dXs.*IMsim; dYs = dYs.*IMsim; % trims disps to only ROIs
fprintf('Filtered Using Wavelets\r');
end

% No iterations
if iter == 0
    dXs = dX; dYs = dY;
    fprintf('Not Filtered\r');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clean up smoothed edges (re-isolates the ROI)
% [dXs]=dXs.*IMsim;
% [dYs]=dYs.*IMsim;
if mc == 1
    figure, imagesc(dXs), colormap('jet'), axis('square'); colorbar; title('Filtered dX')
    if ~fullroi, hold on; plot(pJ,pI,'yx'); hold off; end
    figure, imagesc(dYs), colormap('jet'), axis('square'); colorbar; title('Filtered dY')
    if ~fullroi, hold on; plot(pJ,pI,'wx'); hold off; end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute Displacement Error
errX = zeros(matrix); errY = zeros(matrix);
for pp = 1:length(pI)
    disp_difX(pp,:,mc) = [pI(pp) pJ(pp) (dXs(pI(pp),pJ(pp))-
    dXsim(pI(pp),pJ(pp))];
    disp_difY(pp,:,mc) = [pI(pp) pJ(pp) (dYs(pI(pp),pJ(pp))-
    dYsim(pI(pp),pJ(pp))];
    errX(pI(pp),pJ(pp)) = dXs(pI(pp),pJ(pp)) - dXsim(pI(pp),pJ(pp));
    errY(pI(pp),pJ(pp)) = dYs(pI(pp),pJ(pp)) - dYsim(pI(pp),pJ(pp));
end
[disp_sample,disp_loc] = DisplacementError(dXs,dYs,pI,pJ);
disp_out(:,:,mc)=[disp_loc disp_sample];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute Disp2Strain at selected points and Strain Error
if strainson
    ostrain(:,:,mc) = Disp2Strain(dXs,dYs,matrix,ImageRes,nI,nJ);
    strn_dif(:,:,mc) = ostrain(:,:,mc) - sstrain;
    if mc==1, % true data
        ostrain(:,:,mc)=Disp2Strain(dXs,dYs,matrix,ImageRes,pI,pJ);
if mc == 1 && fullroi == 1
    strain=zeros(matrix(1),matrix(2),4);
    for ss=[1,2,4];
        for ii=1:length(pI)
            strain(pJ(ii),pI(ii),ss)=ostrain(ii,ss,mc);
        end
        strain(:,ss)=strain(:,ss)';
    end
    Exx = strain(:,1); Eyy = strain(:,2); Exy =
    strain(:,4);

    figure,imagesc(Exx),colormap('jet'),axis('square');colorbar;title('Calculated E_x_x')%
    figure,imagesc(Eyy),colormap('jet'),axis('square');colorbar;title('Calculated E_y_y')%
    figure,imagesc(Exy),colormap('jet'),axis('square');colorbar;title('Calculated E_x_y')%
end

%% Evaluate Precision
avgDisp=mean(disp_out(:,3,:),3);
stdDisp=std(disp_out(:,3,:),0,3);
% Displacement Precision:
dprec = Precision(disp_out(:,3,:),0);
MCPrecBias(:,1) = dprec;
%MPrecbias(1) = dprec;
if strainson
    avgStrain=mean(ostrain,3);
    stdStrain=std(ostrain,0,3);
    % Strain Precision:
ebias = Precision(ostrain,1);
    MCPrecBias(:,2) = ebias;
    %MPCprecbias(2) = ebias;
end

%% Evaluate Bias
avg_difX = mean(disp_difX,3); % mean of errors - bias (by point)
avg_difY = mean(disp_difY,3);
avg_difD = avg_difX(:,1:2);
% Displacement Bias
dbias = Bias([disp_difX(:,3,:) disp_difY(:,3,:)],0);
MCPrecBias(:,:,3) = dbias;
% MCPrecBias(3) = dbias;
difX = zeros(matrix); difY = zeros(matrix); difD = zeros(matrix);
for dd = 1:length(avg_difX)
    difX(avg_difX(dd,1),avg_difX(dd,2)) = avg_difX(dd,3);
    difY(avg_difY(dd,1),avg_difY(dd,2)) = avg_difY(dd,3);
end
% figure, imagesc(difX), colormap('gray'), axis('square'); colorbar; title('dX Bias')
% figure, imagesc(difY), colormap('jet'), axis('square'); colorbar; title('dY Bias')

% Strain Bias
if strainson
    ebias = Bias(strn_dif,1);
    MCPrecBias(:,:,4) = ebias;
    % MCPrecbias(4) = ebias;
end

timeout = etime(clock,t0);

8.2 Wavelet De-noising Scheme MATLAB function

function denoised = wavefilt(noisy,wtype,wlevel,tmethod);

% Default Thresholding
if(tmethod == 1)
    [thr,sorh,keepapp] = ddencmp('den','wv',noisy);  
    [denoised,a,b,c,d] = wdencmp('gbl',noisy,wtype,wlevel,thr,sorh,keepapp);
end

% Soft Level Dependent Thresholding
if(tmethod == 2)
    [c,s] = wavedec2(noisy,wlevel,wtype);
    sorh = 's';
    alpha_hi = 8;
    thr = wthrmngr('dw2ddenoLVL','penalhi',c,s,alpha_hi);
    [denoised,a,b,c,d] = wdencmp('lvd',noisy,wtype,wlevel,thr,sorh);
end

if(tmethod == 3)
    [c,s] = wavedec2(noisy,wlevel,wtype);
    sorh = 's';
    alpha_me = 2;
    thr = wthrmngr('dw2ddenoLVL','penalme',c,s,alpha_me);
    [denoised,a,b,c,d] = wdencmp('lvd',noisy,wtype,wlevel,thr,sorh);
end

if(tmethod == 4)
    [c,s] = wavedec2(noisy,wlevel,wtype);
    sorh = 's';
    alpha_lo = 1.5;
thr = wthrmngr('dw2ddenoLVL','penallo',c,s,alpha_lo);
[denoised,a,b,c,d] = wdencmp('lvd',noisy,wtype,wlevel,thr,sorh);
end

if(tmethod == 5)
    [c,s] = wavedec2(noisy,wlevel,wtype);
sorh = 's';
SCAL = 'mln';
thr = wthrmngr('dw2ddenoLVL','sqtwolog',c,s,SCAL);
[denoised,a,b,c,d] = wdencmp('lvd',noisy,wtype,wlevel,thr,sorh);
end

if(tmethod == 6)
    [c,s] = wavedec2(noisy,wlevel,wtype);
sorh = 's';
SCAL = 'sln';
thr = wthrmngr('dw2ddenoLVL','sqtwolog',c,s,SCAL);
[denoised,a,b,c,d] = wdencmp('lvd',noisy,wtype,wlevel,thr,sorh);
end

if(tmethod == 7)
    [c,s] = wavedec2(noisy,wlevel,wtype);
sorh = 's';
SCAL = 'one';
thr = wthrmngr('dw2ddenoLVL','sqtwolog',c,s,SCAL);
[denoised,a,b,c,d] = wdencmp('lvd',noisy,wtype,wlevel,thr,sorh);
end

if(tmethod == 8)
    [c,s] = wavedec2(noisy,wlevel,wtype);
sorh = 's';
thr = wthrmngr('dw2ddenoLVL','sqrtbal_sn',c,s);
[denoised,a,b,c,d] = wdencmp('lvd',noisy,wtype,wlevel,thr,sorh);
end

if(tmethod == 9)
    [c,s] = wavedec2(noisy,wlevel,wtype);
sorh = 's';
alpha_bm = 2;
thr = wdcbm2(c,s,alpha_bm);
[denoised,a,b,c,d] = wdencmp('lvd',noisy,wtype,wlevel,thr,sorh);
end

% Soft Global Thresholding
if(tmethod == 10)
    [c,s] = wavedec2(noisy,wlevel,wtype);
keepapp = 1;
sorh = 's';
alpha_pbm = 2;
det1 = detcoef2('compact',c,s,1);
sigma = median(abs(det1))/0.6745;
thr = wbmpen(c,s,sigma,alpha_pbm);
[denoised,a,b,c,d] = wdencmp('gbl',noisy,wtype,wlevel,thr,sorh,keepapp);
end

if(tmethod == 11)
[c,s] = wavedec2(noisy,wlevel,wtype);
keepapp = 0;
sorh = 's';
alpha_pbm = 2;
detl = detcoef2('compact',c,s,1);
sigma = median(abs(detl))/0.6745;
thr = wbmpen(c,s,sigma,alpha_pbm);
[denoised,a,b,c,d] = wdencmp('gbl',noisy,wtype,wlevel,thr,sorh,keepapp);
end

%Hard Level Dependent Thresholding
if(tmethod == 12)
[c,s] = wavedec2(noisy,wlevel,wtype);
sorh = 'h';
alpha_hi = 8;
thr = wthrmngr('dw2ddenoLVL','penalhi',c,s,alpha_hi);
[denoised,a,b,c,d] = wdencmp('lvd',noisy,wtype,wlevel,thr,sorh);
end

if(tmethod == 13)
[c,s] = wavedec2(noisy,wlevel,wtype);
sorh = 'h';
alpha_me = 2;
thr = wthrmngr('dw2ddenoLVL','penalme',c,s,alpha_me);
[denoised,a,b,c,d] = wdencmp('lvd',noisy,wtype,wlevel,thr,sorh);
end

if(tmethod == 14)
[c,s] = wavedec2(noisy,wlevel,wtype);
sorh = 'h';
alpha_lo = 1.5;
thr = wthrmngr('dw2ddenoLVL','penallo',c,s,alpha_lo);
[denoised,a,b,c,d] = wdencmp('lvd',noisy,wtype,wlevel,thr,sorh);
end

if(tmethod == 15)
[c,s] = wavedec2(noisy,wlevel,wtype);
sorh = 'h';
SCAL = 'mln';
thr = wthrmngr('dw2ddenoLVL','sqtwolog',c,s,SCAL);
[denoised,a,b,c,d] = wdencmp('lvd',noisy,wtype,wlevel,thr,sorh);
end

if(tmethod == 16)
[c,s] = wavedec2(noisy,wlevel,wtype);
sorh = 'h';
SCAL = 'sln';
thr = wthrmngr('dw2ddenoLVL','sqtwolog',c,s,SCAL);
[denoised,a,b,c,d] = wdencmp('lvd',noisy,wtype,wlevel,thr,sorh);
end

if(tmethod == 17)
[c,s] = wavedec2(noisy,wlevel,wtype);
sorh = 'h';
SCAL = 'one';
thr = wthrmngr('dw2ddenoLVL','sqrtwolog',c,s,SCAL);
    [denoised,a,b,c,d] = wdencmp('lvd',noisy,wtype,wlevel,thr,sorh);
end

if(tmethod == 18)
    [c,s] = wavedec2(noisy,wlevel,wtype);
    sorh = 'h';
    thr = wthrmngr('dw2ddenoLVL','sqrtbal_sn',c,s);
    [denoised,a,b,c,d] = wdencmp('lvd',noisy,wtype,wlevel,thr,sorh);
end

if(tmethod == 19)
    [c,s] = wavedec2(noisy,wlevel,wtype);
    sorh = 'h';
    alpha_bm = 2;
    thr = wdcbm2(c,s,alpha_bm);
    [denoised,a,b,c,d] = wdencmp('lvd',noisy,wtype,wlevel,thr,sorh);
end

%Hard Global Thresholding
if(tmethod == 20)
    [c,s] = wavedec2(noisy,wlevel,wtype);
    keepapp = 1;
    sorh = 'h';
    alpha_pbm = 2;
    det1 = detcoef2('compact',c,s,1);
    sigma = median(abs(det1))/0.6745;
    thr = wbmpen(c,s,sigma,alpha_pbm);
    [denoised,a,b,c,d] = wdencmp('gbl',noisy,wtype,wlevel,thr,sorh,keepapp);
end

if(tmethod == 21)
    [c,s] = wavedec2(noisy,wlevel,wtype);
    keepapp = 0;
    sorh = 'h';
    alpha_pbm = 2;
    det1 = detcoef2('compact',c,s,1);
    sigma = median(abs(det1))/0.6745;
    thr = wbmpen(c,s,sigma,alpha_pbm);
    [denoised,a,b,c,d] = wdencmp('gbl',noisy,wtype,wlevel,thr,sorh,keepapp);
end
### 8.3 Detail Coefficient Threshold Legend

- Default
- Penalized Medium (Soft)
- Square Root of Logarithm with Level Noise Estimation (Soft)
- Square Root Logarithm with No Reescaling (Soft)
- Birge Massart (Soft)
- Penalized Birge Massart without keeping App. Coefficients (Soft)
- Penalized Medium (Hard)
- Square Root of Logarithm with Level Noise Estimation (Hard)
- Square Root Logarithm with No Reescaling (Hard)
- Birge Massart (Hard)
- Penalized Birge Massart without keeping App. Coefficients (Hard)
- Penalized High (Soft)
- Penalized Low (Soft)
- Square Root of Logarithm with First Level Noise Estimation (Soft)
- Birkel's Sn (Soft)
- Penalized Birge Massart with Approximation Coefficients Kept (Soft)
- Penalized High (Hard)
- Penalized Low (Hard)
- Square Root of Logarithm with First Level Noise Estimation (Hard)
- Birkel's Sn (Hard)
- Penalized Birge Massart with Approximation Coefficients Kept (Hard)

### 8.4 References


