Level-Weighed Wavelet Fusion:  
A Soft Decision Image Fusion Technique

A thesis submitted to the Graduate Division of the University of Hawai‘i in the partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

December 2007

By
Jeong Hwan Bang

Thesis Committee:

Todd R. Reed, Chairperson
Andy Stenger
N. Thomas Gaarder
We certify that we have read this thesis and that, in our opinion, it is satisfactory in scope and quality as a thesis for the degree of Master of Science in Electrical Engineering.

THESIS COMMITTEE

[Signatures]

Chairperson

[Signatures]

[Signatures]
Table of Contents

List of Tables ...................................................................................................................... v
List of Figures ..................................................................................................................... vi
Chapter 1: Introduction ....................................................................................................... 1
  1.1. Motivation ........................................................................................................... 1
  1.2. Outline ................................................................................................................. 2
Chapter 2: Background Information ................................................................................... 3
  2.1. Image Fusion ....................................................................................................... 3
  2.1.1 Evolution of Techniques ............................................................................. 4
  2.1.1.1 Spatial Domain Techniques ................................................................. 5
  2.1.1.2 Multiresolution Analysis Techniques ..................................................... 6
  2.2. Laplacian Pyramid .............................................................................................. 7
  2.3. Wavelet Transform ............................................................................................. 8
    2.3.1 1D Discrete Wavelet Transform ................................................................. 9
    2.3.2 2D Discrete Wavelet Transform ............................................................... 12
  2.4. Stationary Wavelet Transform .......................................................................... 14
  2.5. MRA Transform Fusion Techniques ................................................................ 17
    2.5.1 Classes of Algorithms ............................................................................... 17
    2.5.2 Comparison of Transforms ....................................................................... 19
    2.5.3 Max Fusion .................................................................................................. 19
  2.6. Image Denoising ............................................................................................... 22
    2.6.1 Basic Procedures ....................................................................................... 22
    2.6.2 Shift-Invariant Denoising .......................................................................... 26
Chapter 3: Level-Weighted Wavelet Fusion ..................................................................... 28
  3.1. Algorithm Classification ................................................................................... 28
  3.2. Level-Weighted Wavelet Fusion ...................................................................... 29
    3.2.1 Calculation of Weights ............................................................................ 30
  3.3. Denoising Application ...................................................................................... 36
  3.4. Algorithm Discussion ....................................................................................... 36
    3.4.1 Fusion of Image Edges .............................................................................. 37
    3.4.2 Utilizing the Interscale Information .......................................................... 39
    3.4.3 Fusing with Multiple Source Images ........................................................ 41
Chapter 4: Results ............................................................................................................. 42
  4.1. Test Images ....................................................................................................... 42
  4.2. LWWF Reconstructed Images .......................................................................... 44
  4.3. Visual Analysis .................................................................................................. 47
  4.4. Power Sum ........................................................................................................ 51
  4.5. Mutual Information ........................................................................................... 58
  4.6. Fusion in the Presence of Noise ........................................................................ 60
    4.6.1 Noise Model .............................................................................................. 60
    4.6.2 SNR Formulation ...................................................................................... 62
    4.6.3 Wavelet Denoising with the Universal Threshold ..................................... 64
    4.6.4 Wavelet Denoising with an Empirically Derived Optimal Threshold ...... 67
Chapter 5: Conclusion ....................................................................................................... 73
Appendix: MATLAB Codes ............................................................................................. 75
A.1. Preliminaries ............................................................................................................. 75
A.2. Image Fusion ............................................................................................................. 76
    A.2.1 Average Fusion ................................................................................................. 76
    A.2.2 Max Fusion ....................................................................................................... 79
A.2.3 Level-Weighted Wavelet Fusion ......................................................................... 82
    A.2.3.1 LWWF Method 1 ......................................................................................... 82
    A.2.3.2 LWWF Method 2 ......................................................................................... 85
A.3. Performance Evaluation ......................................................................................... 88
    A.3.1 Power Sums ....................................................................................................... 89
        A.3.1.1 Mask 1 ................................................................................................... 89
        A.3.1.2 Mask 2 ................................................................................................... 90
        A.3.1.3 Mask 3 ................................................................................................... 90
    A.3.2 Mutual Information ........................................................................................... 91
    A.3.3 Signal-to-Noise Ratio ....................................................................................... 92
References ...................................................................................................................... 93
List of Tables

<table>
<thead>
<tr>
<th>Tables</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Mutual information of fused images</td>
<td>60</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Venn diagram representation of the sources.</td>
<td>3</td>
</tr>
<tr>
<td>2. Block diagram of a weighted average for fusion.</td>
<td>6</td>
</tr>
<tr>
<td>3. Filter representation of the orthogonal mapping.</td>
<td>10</td>
</tr>
<tr>
<td>4. Block diagram of a 3 level DWT decomposition.</td>
<td>11</td>
</tr>
<tr>
<td>5. Subband block required for each level of 2D DWT decomposition.</td>
<td>14</td>
</tr>
<tr>
<td>6. Subband block required for each level of 2D SWT decomposition.</td>
<td>16</td>
</tr>
<tr>
<td>7. Diagram depicting max fusion using the DWT.</td>
<td>21</td>
</tr>
<tr>
<td>8. Hard and soft thresholding function.</td>
<td>25</td>
</tr>
<tr>
<td>9. High-Level block diagram of fusion process.</td>
<td>30</td>
</tr>
<tr>
<td>10. Illustration of the summation of coefficients across decomposition levels.</td>
<td>34</td>
</tr>
<tr>
<td>11. Two images to be processed for fusion.</td>
<td>37</td>
</tr>
<tr>
<td>12. Fused image using the max fusion algorithm.</td>
<td>38</td>
</tr>
<tr>
<td>13. Fused image using the LWWF method 1.</td>
<td>39</td>
</tr>
<tr>
<td>14. Fused image using the LWWF method 2.</td>
<td>39</td>
</tr>
<tr>
<td>15. Vertical coefficients across the different levels of a SWT.</td>
<td>40</td>
</tr>
<tr>
<td>16. Illustration of the receiver arrangement around the subject to be imaged.</td>
<td>42</td>
</tr>
<tr>
<td>17. Multicoil MRI images used for testing.</td>
<td>43</td>
</tr>
<tr>
<td>18. Fusion by taking the arithmetic mean between the source images.</td>
<td>44</td>
</tr>
<tr>
<td>19. Reconstruction using LWWF method 1.</td>
<td>45</td>
</tr>
<tr>
<td>20. Reconstruction using LWWF method 2.</td>
<td>46</td>
</tr>
</tbody>
</table>
21. Reconstruction by wavelet max fusion ................................................................. 46
22. Comparison of reconstruction methods ............................................................... 47
23. Zoomed plots of the left brain region ............................................................... 48
24. Zoomed plots of the right brain region ............................................................... 48
25. Zoomed plots of the upper brain region .............................................................. 49
26. Zoomed plots of the lower brain region .............................................................. 49
27. Magnitude plots of the negative valued pixels .................................................. 50
28. Log scaled Fourier spectra of the reconstructed images ..................................... 52
29. Spectra of the source images ............................................................................ 52
30. Progression of mask 1 ....................................................................................... 54
31. Normalized sums of mask 1 .............................................................................. 54
32. Progression of mask 2 ....................................................................................... 55
33. Normalized sums of mask 2 .............................................................................. 56
34. Progression of mask 3 ....................................................................................... 57
35. Normalized sums of mask 3 .............................................................................. 57
36. Mask used to calculate mutual information ...................................................... 59
37. Effect of additive noise on source 1 for various noise factor values ............... 62
38. SNR for various noise levels ............................................................................ 63
39. SNR for fixed wavelet denoising over various noise levels .............................. 66
40. SNR for cases with and without the applying wavelet denoising ................. 66
41. SNR values for varying thresholds for decomposition level 4 ....................... 69
42. SNR values for varying thresholds decomposition level 5 ............................ 70
43. SNR values for varying thresholds for decomposition level 6. .................... 71
44. SNR for adaptive wavelet denoising over different noise scales ................... 72
45. SNR for cases with and without the applying adaptive wavelet denoising. ....... 72
CHAPTER 1. Introduction

1.1. Motivation

The main objective of image fusion is to combine multiple images into one, complete image, maintaining all the salient features. This process can range extensively in complexity: from simple averages to transform-based methods. Since the incorporation of multiresolution analysis for image fusion, there has been a proliferation of new techniques. The class of the algorithm is usually categorized according to the following characteristics:

1) The space where fusion occurs (e.g., wavelet domain)
2) The region of operation (e.g., pixel-by-pixel or window based)

As for the actual fusion step, the max operation is very common among transform-based techniques. These categorizations will be further explored in Chapter 2.

Image fusion is a very practical and important area of image processing; current applications range from remote sensing to medical imaging [1],[2],[3]. In many of these cases, the actual choice of the algorithm is highly data dependent. Since there is no sole optimal method, there is an ongoing search for improvements. Aside from these application-specific algorithms, there is still a strong pursuit to obtain the most general, optimal method. Among other reasons, it could be used to test the compatibility of the solution to the problem. In this sense, besides good performance, this general method should be practical, intuitive, and easy to implement. As we will see, our proposed algorithm achieves all these criteria.

1
1.2. Outline

A brief historical introduction to the structured evolution of image fusion is presented in Chapter 2. Here, the progression of techniques, as well as the classification of the algorithms will be presented. This will be followed by a concise derivation of the relevant mathematical techniques, which includes the Laplacian pyramid transform and some variants of the wavelet transform. Chapter 3 will then introduce the proposed algorithm—the level-weighted wavelet fusion (LWWF). This technique is novel on multiple levels, the most noticeable being that we do not use a nonlinear, max operation for fusion. We instead opt for a weighted average among the source images. Hence, in our algorithm, the crucial part is the calculation of the weights, which is computed in the wavelet domain. As will be further elaborated, these weights utilize the intrascale (within level) and interscale (across level) dependencies of wavelet coefficients. This is novel in itself, since most wavelet-based fusion algorithms make no attempt to exploit the interscale coefficient dependencies. Additionally, we will explain and demonstrate other notable advantages of using a linear fusion operation. Experimental results on our main test images will be presented in Chapter 4. One unfortunate problem with image fusion methods is the lack of a universally accepted metric to objectively judge its performance. In spite of this drawback, we attempt to measure the performance of the LWWF with several known metrics and compare it with other well known algorithms. Combined with a subjective, visual analysis, the end results convincingly display the superior performance of the LWWF. Finally, Chapter 5 will conclude this thesis with final analysis and remarks.
CHAPTER 2. Background Information

2.1. Image Fusion

Image fusion is the process of combining multiple source images into one, with the purpose of retaining the most useful information from each image. Each of the source images can be thought up as a "view" of the observation scene. There will be regions of overlapping information and regions with exclusive information, referred to as the redundant information region and the complementary information region, respectively [4]. A Venn diagram representation of these regions for the case of two source images is depicted in Figure 1.

![Venn diagram](image)

Figure 1: Venn diagram representation of the information the sources provide over the entire scene.

Ultimately, image fusion should produce an image that integrates all the complementary information and suppress the redundancy, hence rendering it more suitable for human and computer analysis. Our criterion to achieve this is by means of contrast enhancement. Contrast is intrinsically linked to the human and computer ability to discern details of a scene of interest, as it delineated areas of discontinuity and similarity [5]. As we will see, the fusion processes that enable this can be as simple as
taking the arithmetic mean between the images, and as complex as using multiresolution analysis decompositions.

2.1.1 Evolution of Techniques

The general idea of combining images into one has an intriguing history [3]. Before the general framework of image fusion has been structured, primitive methods such as simple averages and, eventually, weighted averages were being practiced. These techniques were useful, but proved ineffective for more difficult tasks. Since all operations are done in the spatial domain, we categorize these techniques as spatial domain techniques.

Later, Burt and Adelson introduced the Laplacian pyramid transform which was originally tailored for image compression [6]. This technique seeded the idea of looking at images at various scale levels, emulating how the human visual system operates. As a consequence, it is able to extract different bands of high frequency signal, granting expert control over the fusion process. The general idea of decomposing an image to different scales is known as multiresolution analysis (MRA) [5]. This is the other category of techniques we will discuss.

More recently, a newer type of MRA – the wavelet transform – has supplanted the Laplacian pyramid transform of its role in image fusion. As we will elaborate later, it retains the strengths of its predecessor and offers additional advantages. In the following, we will further develop the two categories of algorithms – spatial domain and MRA techniques.
2.1.1.1 Spatial Domain Techniques

Early image fusion techniques were very simple. Let \( \{ X_i \}, i = 1, 2, \ldots, N \) denote the set of source images to be fused. An acceptable fusion method is to take its arithmetic mean:

\[
(X_1, X_2, \ldots, X_N) \mapsto \frac{1}{N} \sum_{n=1}^{N} X_n. \tag{1}
\]

The advantage of this is its intuitiveness and simplicity in implementation. It is also effective under certain conditions. Unfortunately, it has some serious drawbacks, one being the reduction of contrast [7]. This is expected since an averaging operation is essentially equivalent to a lowpass filter. An obvious generalization of the simple average fusion is the weighted average fusion:

\[
(X_1, X_2, \ldots, X_N) \mapsto \sum_{n=1}^{N} a_i X_i \tag{2}
\]

where each source image \( X_i \) is given a single, constant weight \( a_i \). This process is illustrated in Figure 2. This process, while being better than the uniform average, still suffers the same contrast degrading drawbacks. Overall, while there are some elaborate methods to optimize the weights, its inherent inability to separate frequency information limits the usefulness of this method. This is a general handicap of all spatial domain techniques.
Figure 2: Block diagram of a weighted average for fusion.

2.1.1.2 Multiresolution Analysis Techniques

By recent consensus, MRA techniques are recognized as the superior choice for fusion [7], a major improvement over spatial domain techniques. The premise of any MRA technique involves the decomposition of the source images into lower resolution copies of themselves; fusion is performed on these multiscale copies. The concept of MRA closely resembles the actual human visual system: identifying small details require high resolutions, while large objects are easily discernable with a coarse view [5]. One reason for its prominence is in its ability to spatially localize the high-frequency contents [7]. This is conducive to retaining desired edges and image contrasts.

The introduction of the Laplacian pyramid transform led to the advent of MRA fusion techniques. It has been used extensively for this purpose until the recent development of wavelet transforms. The wavelet transform is another MRA technique with unique, desirable properties, and has basically superseded the Laplacian pyramid for its role in fusion. Because of their importance for this paper, the next sections will
succinctly derive these prominent MRA transforms. To complete the discussion, we will conclude by mentioning some of the basic algorithms that utilize them for fusion.

2.2. Laplacian Pyramid

The Laplacian pyramid, first introduced by Burt and Adelson [6], is a very powerful tool for image processing, originally tailored for image coding. The main problem it addresses is in the riddance of the spatial correlation in images. It does so by creating multiple, decorrelated copies of the original image of decreasing dimensions. Hence, these copies enjoy lower entropy values, allowing for a more efficient coding scheme.

Denote $X$ as the original image. As a first stage, the original image is lowpass filtered with a Gaussian-like kernel, i.e.:

$$X_1 = X * w_G,$$  \hspace{1cm} (3)

where $*$ denotes a convolution operation (2D convolution in this case) and $w_G$ is the Gaussian-like weighting kernel. Some constraints are enforced on the kernel. Namely, it is normalized to sum to unity, it is chosen to be symmetric, and it maintains the equal contribution property [6]. Now, the residue between the original image and the lowpass filtered image is calculated:

$$L_0 = X - X_1,$$  \hspace{1cm} (4)
which is essentially the high frequency information. This residue image is largely
decorrelated and has a low entropy value. We keep the lowpass filtered image \( X_1 \) and the
residue image instead of the original image \( X \). Note that \( X_1 \) has a reduced sampling
rate. It is trivial to see how we recover the original image from \( X_1 \) and \( L_0 \).

This process is reiterated using the lowpass filtered image as the input at each
step. This gives us a sequence of residue images \( L_0, L_1, \ldots, L_N \) and a final lowpass
filtered image \( X_{N+1} \). Because of its reduced sampling rate, at the end of each iteration,
the lowpass filtered image can be downsampled (conventionally by a factor of 2) with no
loss of information. Hence, the sequence of residue images progressively decreases in
dimension. If this sequence were to be stacked by increasing iteration, it visually
resembles a pyramid structure.

Essentially, at each stage in the decomposition, the low frequency and the high
frequency signals are decoupled. This means that the sequence \( L_0, L_1, \ldots, L_N \) is a set of
high-frequency bands and the final image \( X_{N+1} \) is the remaining low frequency signal.
As opposed to working directly with the original image \( X \), it's easy to see that we have
much better control over the fusion process by operating with these sequences.

2.3. Wavelet Transform

The wavelet transform is a relatively new signal processing tool for time and frequency
analysis. Unlike the ubiquitous Fourier transform, the wavelet transform has the ability to
localize in frequency and time. It has an interesting history in that it has been
independently discovered by multiple disciplines. In the end, it was Mallat who was credited for assembling a unified theory of wavelets [8].

In the following, a very intuitive subband filter definition of the wavelet transform will be presented. We start by deriving the 1D discrete wavelet transform (DWT), which is a discretized version of the wavelet transform. We then generalize it to the 2D case which is required for image fusion. Furthermore, we introduce a necessary variant of the wavelet transform – the stationary wavelet transform.

2.3.1 1D Discrete Wavelet Transform

In the heart of the wavelet transform, there lies a pair of orthogonal filters that define it. This is a pair of lowpass and highpass filters, h and g defined by the sequences \( \{ h_n \} \) and \( \{ g_n \} \), respectively, that satisfy strict constraints [9]. Both filters are orthogonal to its even shifts:

\[
\sum_n h_{n-2k} h_{n-2k} = \delta_{jk} \quad \text{and} \quad \sum_n g_{n-2k} g_{n-2k} = \delta_{jk},
\]

and are mutually orthogonal:

\[
\sum_n h_{n-2j} g_{n-2k} = \delta_{jk}.
\]

If the filters are related by:
then by definition, they are called quadrature mirror filters. Assume that the original 1D signal $X$ is of length $2^J$, i.e., $X \in \mathbb{R}^{2^J \times 1}$. Define the decimation operator $D_0$ that takes the even indexed values of the sequence. Then we have the following orthogonal mapping:

$$ g_n = (-1)^n h_{2n} ,$$

(7)

which are two sequences of length $2^{J-1}$. This is illustrated in Figure 3. By construction of the filters, no information is lost in the decimation process. The sequences $e_{J-1}$ and $d_{J-1}$ are called the approximation and detail coefficients, respectively, of level $J-1$. In relation to the conventional definition of wavelet transforms, the lowpass filter $h$ and the highpass filter $g$ are equivalent to the discretized, time-reversed scaling function $\varphi(-x)$ and wavelet function $\psi(-x)$, respectively.

![Figure 3: Filter representation of the orthogonal mapping.](image)

Figure 3: Filter representation of the orthogonal mapping.
Now that the orthogonal mapping has been defined, we can define the DWT. The DWT is an iterative application of this mapping that starts with the original signal, and uses the resulting approximation coefficients as the progressive inputs, i.e.:

$$e_j \mapsto \left(D_0(e_j * h), D_0(e_j * g)\right), \quad j = J, J-1, \ldots, 0, \quad (9)$$

where, for notational convenience, we define $e_J = X$. Reminiscent of the Laplacian pyramid transform, we keep the set of detail sequences and the remaining approximation sequence $d_{J-1}, d_{J-2}, \ldots, d_0, c_0$. In practice, we can curtail this process to some arbitrary level $K \in \{k : J-1 < k \leq 0\}$ so that we are left with the set $d_{J-1}, d_{J-2}, \ldots, d_K, c_K$. We refer to level $K$ as the curtail level; in most cases, it is unnecessary to decompose to level 0, as the poor resolution of the sequence makes it useless for processing. To clarify this process, a block diagram representation of the DWT curtailed to level $J-3$ is shown in Figure 4.

![Figure 4: Block diagram of a 3 level DWT decomposition.](image)
By construction, the resulting sequences are all mutually orthogonal. Furthermore, the detail coefficients obtained at each iteration provide different bands of high frequency information. For example, \( d_{j=1} \) provides the finest level of detail since it was obtained from the original signal, while \( d_{k}, k < J - 1 \) provides coarser detail information since the high frequency information is extracted from a lowpass filtered signal. This is how frequency localization is obtained with the discrete wavelet transform. Furthermore, since the defined filters have compact support, we also obtain time localization. Because of the orthogonality of the transforms, the reconstruction process follows easily.

2.3.2 2D Discrete Wavelet Transform

For this experiment, we are interested in applying the wavelet transform on 2D image data. Therefore, we now extend the definition of the 1D discrete wavelet transform to the 2D discrete wavelet transform. For convenience, assume that the input image is of square dimensions \( 2^' \times 2^' \), i.e., \( X \in \mathbb{R}^{2^' \times 2^'} \). The extension to 2D is simple since it is essentially a repeated application of the 1D DWT. Specifically, we perform the 1D DWT along all the columns of \( X \), independently of each other. Concatenating the results in the order it was processed, we obtain two coefficient matrices \( C_{j=1}, D_{j=1} \in \mathbb{R}^{2^'-1 \times 2^'} \), corresponding to the lowpass filter output and highpass filter output, respectively. Mathematically, it is the following mapping:

\[
\mathbb{R}^{2^' \times 2^'} \rightarrow \left( \mathbb{R}^{2^'-1 \times 2^'}, \mathbb{R}^{2^'-1 \times 2^'} \right)
\]  

(10)
We now reapply the 1D DWT, but this time independently along the rows of \( C_{J-1} \) and \( D_{J-1} \). The outputs from \( C_{J-1} \) are the coefficient matrices \( C_{J-1}^L \) and \( D_{J-1}^L \), corresponding to the lowpass filter and the highpass filter output, respectively, of dimensions \( 2^{J-1} \times 2^{J-1} \), i.e.,:

\[
R^{2^{J-1} \times 2^{J-1}} \rightarrow \left( R^{2^{J-1} \times 2^{J-1}}, R^{2^{J-1} \times 2^{J-1}} \right)
\]

\( C_{J-1} \mapsto \left( C_{J-1}^L, D_{J-1}^L \right) \)

These are referred to as the approximation coefficients and the vertical detail coefficients. The approximation coefficients correspond to a 2D lowpass filtered \( X \). The outputs from \( D_{J-1} \) are the coefficient matrices \( D_{J-1}^H \) and \( D_{J-1}^P \), corresponding the lowpass filter and the highpass filter output, respectively:

\[
R^{2^{J-1} \times 2^{J-1}} \rightarrow \left( R^{2^{J-1} \times 2^{J-1}}, R^{2^{J-1} \times 2^{J-1}} \right)
\]

\( D_{J-1} \mapsto \left( D_{J-1}^H, D_{J-1}^P \right) \)

These are called the horizontal detail coefficients and the diagonal detail coefficients. In conclusion, the resultant detail coefficients \( D_{J-1}^H, D_{J-1}^P, D_{J-1}^D \) corresponds to 2D directional highpass filtered \( X \). Figure 5 shows this basic subband filter block.
Figure 5: Subband block required for each level of 2D DWT decomposition.

Following the methodology of the 1D case, we iteratively apply these functions on the approximation outputs. Again, let $K \in \{k : J - 1 < k \leq 0\}$ be the curtail level.

Then, the final set of matrices are:

$$\begin{align*}
\mathbf{D}_{j-1}^H, \mathbf{D}_{j-1}^v, \mathbf{D}_{j-2}^H, \mathbf{D}_{j-2}^v, \mathbf{D}_{j-2}^d, \ldots, \mathbf{D}_K^H, \mathbf{D}_K^v, \mathbf{D}_K^d, \mathbf{C}_K^d.
\end{align*}$$

As in the 1D case, these coefficient matrices are mutually orthogonal and the levels provide localized frequency information. Furthermore, it provides directional high frequency information.

2.4. Stationary Wavelet Transform

Decimation is performed at each level in the discrete wavelet transform. But by
construction of the pair of filters, this results in no loss of information. When the 1D DWT was defined, the even indexes were arbitrarily kept during decimation. If we had kept the odd indexes, the reconstructed image would be identical. In fact, at each level, we can arbitrarily choose whether the odd or even indexes are kept. Denote a sequence of 0’s and 1’s by $e = (e_{j-1}, e_{j-2}, \ldots, e_k)$, where:

$$
\begin{align*}
    e_j &= 0 \quad \Rightarrow \quad \text{keep the even indexes during the } j^{th} \text{ level of decimation} \\
    e_j &= 1 \quad \Rightarrow \quad \text{keep the odd indexes during the } j^{th} \text{ level of decimation}
\end{align*}
$$

Given a sequence $e$, an $e$-decimated DWT is defined as the DWT resulting from the decimation rule as defined above. It follows that any arbitrary $e$ will result in identical reconstructed images.

However, within the wavelet domain, the choice of the decimation sequence is not so arbitrary, i.e., the output coefficients are not the same. This is a minor issue if we are strictly using the wavelet transform for visual analysis, but operations in the wavelet domain can be significantly dependent on the choice of $e$. This issue is most commonly raised in the context of signal denoising. It turns out that the standard wavelet transform does not have an important shift-invariant property.

The solution to this problem is conceptually simple: we remove the decimation step as depicted in Figure 6. Consequently, to maintain the frequency localization, we must upsample the quadrature filters at each level. Only then will we have varying scales at each decomposition level. The drawbacks are that it requires more computations, more storage, and we are left with some redundant information. Because of the redundancy, the
reconstruction is not as straightforward as in the standard wavelet transform. But as a reward, we obtain the desirable shift-invariance property. This method of computing the wavelet transform is called the stationary wavelet transform (SWT). Since there is no need to downsample, the coefficients of the SWT includes all coefficients for any choice of \( \varepsilon \) in the \( \varepsilon \)-decimated DWT.

![Diagram](image)

**Figure 6:** Subband block required for each level of 2D SWT decomposition.

As mentioned, the reconstruction of the SWT is not so straightforward and a common approach is to take an average between the even and odd decimated coefficients (the SWT contains both). The shift-invariance property will be utilized during the denoising experiments shown in the results of Chapter 4. Aside from this useful property, the main reason the SWT is employed for the LWWF is because the dimensions of the coefficients remain the same at each level. In the proposed algorithm, the interscale coefficients need to be compared, i.e., the coefficients between the levels of decomposition. This kind of comparison is much more difficult with decimation where
the dimensions of the coefficients differ.

2.5. MRA Transform Fusion Techniques

Having established the relevant MRA transforms, this section will now introduce common image fusion techniques that utilize them. Among the plethora of new fusion techniques, many of them are just variants of existing ones. In the following, we develop categories based on the view around the pixel of interest the algorithm operates on.

2.5.1 Classes of Algorithms

In a very broad sense, we can classify the fusion algorithms into one of three classes: pixel-based fusion, region-based fusion, and feature-based fusion. This classification is solely based upon how much of the neighboring pixel information is incorporated.

Pixel-based is the most straightforward method of fusion; it only looks at the coefficient being fused and neglects its neighbors. Since a pixel in an image is typically highly correlated with its neighbors, this approach would not be expected to perform well. Additionally, it executes poorly in the presence of noise. For instance, when a detail map of the DWT is being fused, this view has no way to distinguish between a high frequency edge and high frequency noise. But, overall, it is the simplest approach.

In contrast to the pixel-based fusion, region-based fusion includes pixels within a defined neighborhood around the pixel of interest. The area of the neighborhood can be adaptive or fixed, such as a commonly used, fixed-size square window. By this, it is possible to utilize the spatial dependent information of the image. Ultimately, it is able to discern edges and noise much more effectively than with a pixel-based view. However, a
common problem arises when the neighborhood contain edge and non-edge coefficients; based on the window size, it could overemphasize the significance of a coefficient because of its proximity to the salient feature. In such cases, an adaptive window might be a suitable choice. Overall, although it is slightly more complex than pixel-based fusion, a region-based view offers much more power and flexibility over the fusion process.

Feature-based fusion works at a different level of fusion than the previously mentioned classes; it requires an initial processing to identify the distinct regions of interest, such as edges and flat areas. An example is if white and gray matter of a brain image were to be classified prior to fusion. This could be followed by a case based fusion rule. This approach adds a whole new layer of complexity since it usually requires a nontrivial segmentation or edge detection algorithm. Because of this, the performance capability of a feature-based fusion algorithm is largely dependent on the preprocessing step.

These classes of algorithms describe spatial domain techniques as well as MRA techniques. However, we are only interested in using MRA techniques since it gives much better control over the fusion process. There is an inherent tradeoff between the complexity of the algorithm and its performance. The feature based fusion sounds appealing, but the preprocessing step can be very complex – finding the right image segmentation or edge detection algorithm for the problem can be as difficult as finding the right fusion technique.
2.5.2 Comparison of Transforms

A crucial part of MRA fusion is in the proper choice of the MRA transform. The shift in preference away from the Laplacian pyramid transform is justified by some unique properties of the wavelet transform [1],[7]:

- Wavelets have compact support; they can localize spatially and in frequency.
- The wavelet functions are chosen to be orthogonal, and hence, information at each level of decomposition is unique.
- Wavelet decompositions provide directional highpass information.

Furthermore, using the Laplacian pyramid for fusion has been shown to display blocking artifacts when the fusing regions differ significantly [2]. These properties established the definitive role of the wavelet transform for image fusion.

2.5.3 Max Fusion

We now present a very well known image MRA fusion algorithm. It is important to introduce this technique since it will be one of the comparison cases for the proposed algorithm.

MRA transforms allow control over localized, high frequency information at different scales. The decoupling of the low frequency and high frequency contents of the image can be exploited with the max fusion algorithm. Since we are interested in wavelet-based fusion, we will explain it in terms of a wavelet transform. Suppose that we are given a set
of $N$ images $\{X_n\}, \ n = 1, 2, \ldots, N$ to be fused. The first step is to apply the 2D DWT on each of the images which results in $N$ sets of approximation and detail coefficients. In general, the detail coefficients correspond to the salient features of the image (e.g., edge, contrast) – the greater the magnitude of a particular coefficient, the more significant it is. This should be intuitively clear since these coefficients represent the high frequency information. Due to the orthogonality of the transform, white noise maps into white noise in the wavelet space, i.e., energy is spread over a large number of wavelet coefficients.

Following this, the logical fusion operation is to take the maximum magnitude valued detail coefficients among the images. The result should retain all the dominant features of the input images [2],[7]. As for the approximation coefficients, we take an arithmetic mean. This follows from the fact that the approximation coefficients are lowpass filtered version of the original image, which are essentially average values. Now that the $N$ sets of wavelet coefficients have been combined into one, we finish with the application of an inverse 2D DWT. The result is the reconstructed, fused image. A diagram depicting this scheme for a 1-level DWT is shown in Figure 7. For convenience, we summarize it into steps:

1) Apply the 2D DWT to each of the $N$ images.

2) Combine the detail coefficients by taking the maximum magnitude valued coefficients among them.

3) Combine the approximation coefficients by taking the arithmetic mean.

4) Reconstruct, via inverse 2D DWT.
In the literature, there are many variants of this max fusion such as looking about an intrascale window of a coefficient (i.e., adjacent coefficients of the same scale level) and judging its significance by its variance. But ultimately, they perform similarly due to the common max operation.

While the max fusion is an easy and intuitive approach, there are some disadvantages. One persistent problem is that it creates undesirable artifacts. Due to the discontinuous nature of the max operation, it produces artificial, high-frequency transitions, manifested as ringing artifacts in the image domain. Also, as expected, the max operation is a poor choice to take in the presence of noise. But due to its simplicity, acceptable performance, and ubiquitousness, it is thought of as the benchmark for wavelet-based image fusion techniques.
2.6. Image Denoising

Image denoising is an important and practical area of image processing; it increases the signal-to-noise ratio and enhances visual analysis. Since the pioneering works of Donoho and Johnstone [10], the incorporation of wavelet transforms for denoising has been a widely accepted choice. By this method, operations are performed on the wavelet coefficients, much like the fusion calculations. This implies that we can employ both techniques with only one pass of the wavelet transform. In most cases, the only computationally significant step in wavelet denoising is the wavelet decomposition; applying denoising during LWWF is thus a natural step to take that will increase SNR with almost no additional computations. With this in mind, this section will introduce the fundamental concepts of wavelet denoising. Although there are very complex denoising schemes in the literature, this strays too far from the main point of the paper; we choose to only present basic denoising schemes that will be adequate for our experiments.

2.6.1 Basic Procedures

For most wavelet denoising, or any type of transform-based denoising in general, the basic procedures are usually defined by a similar set of protocols. The additive noise model is described by:

\[ X = S + \eta, \]  

(15)
where $S$ is the underlying, noiseless image, $\eta$ is the additive noise (usually assumed to be Gaussian distributed), and $X$ is the observed signal. Given $X$, we want to obtain an estimate of the underlying image $\tilde{S}$ that is optimal in some defined sense. For wavelet denoising, this is typically done according to the following steps:

1) Given $X$, perform the DWT down to a curtailed level $k$.

2) For each of the detail coefficient sequences, perform a thresholding operation to eliminate coefficients (i.e., set to 0) less than some threshold $T$.

3) Perform reconstruction, via inverse DWT, on the new coefficients.

The approximation coefficients are left unaltered since they correspond to the lowpass filtered image; noise is usually associated as a high frequency signal.

Actually, if any orthogonal transform were used in place of the DWT, we would still attain acceptable results. Some transforms used for this purpose include the discrete Fourier transform and the discrete cosine transform. But the DWT has several advantages that make it a more desirable choice:

1) **Locality** – the ability to describe local features well in the time and frequency domains.

2) **Multiresolution** – the ability to analyze signals in different scales. It allows efficient analysis of various frequencies.

3) **Compression** – the phenomenon that causes significant signals to manifest as sparse, large-valued coefficients in the wavelet domain.
The compression property is largely due to the compact support property of wavelets. Another important property is that, because the wavelet transform is an orthogonal transform, white noise is transformed into white noise. This implies that the noise energy will be spread to a large number of small-valued coefficients in the transform domain. In total, these are the properties that generated a wide interest in wavelet denoising. In most cases, the thresholding rule is the defining step of the denoising method.

In general, there are two types of thresholding operations commonly seen in the literature: hard thresholding and soft thresholding. Hard thresholding simply sets all detail coefficients below the defined threshold \( T \) to zero:

\[
\{ x : x < T \} \mapsto 0. \tag{16}
\]

Soft thresholding takes it one step further by "shrinking" the remaining coefficients towards zero:

\[
x \mapsto \begin{cases} 
  x - T, & x \geq T \\
  x + T, & x \leq -T \\
  0, & |x| < T
\end{cases} \tag{17}
\]

Not surprisingly, hard thresholding creates discontinuities in the continuum of detail coefficient values. Soft thresholding, however, alleviates this problem by incorporating the scaling operation. To illustrate this, general hard thresholding and soft thresholding
functions are plotted in Figure 8. For the hard thresholding function, there is a visible gap in the output detail coefficient values for $|x| < T$. This discontinuity in values contributes to a higher incidence of artifacts upon reconstruction.

In contrast to hard thresholding, the smoother nature of soft thresholding gives it additional, desirable features [10],[11],[12]. It maintains smooth and adaptive properties, and also achieves near-optimal minimax rate over a large range of Besov spaces. All these benefits justify the use of soft thresholding for this experiment.

The remaining issue is the choice of the threshold value $T$. There are many elaborate methods to calculate this but we choose to use a simple method for reasons discussed above. We opt to use a universal threshold, which, under certain assumptions, yields asymptotic minimax performance in the mean-square error sense [10]. The threshold is given by:

$$T = \sigma \sqrt{2 \log N} ,$$

(18)
where $\sigma^2$ is the noise variance of the signal and $N$ is the number of pixels in the image.

The noise variance is estimated by a median estimator:

$$\hat{\sigma} = \frac{\text{median}\left(\|D_{j=1}^{D}\|\right)}{0.675}, \quad (19)$$

where $D_{j=1}^{D}$ is the sequence of diagonal detail coefficients on the finest level of wavelet decomposition. Due to its dependence on the size of the image, it is shown that the universal threshold yields overly smooth denoised images. In Chapter 4, we will test the performance of this threshold. Additionally, due to its suboptimal performance, we will also find and compare it with an empirically calculated, optimal threshold value.

2.6.2 Shift-Invariant Denoising

The normal DWT lacks the shift-invariance property which is essential for denoising. Consequentially, it produces local ringing artifacts in the neighborhood of discontinuities. These are sometimes referred to as pseudo-Gibbs phenomena since they are reminiscent of the global-ringing Gibbs phenomena resulting from singularities in the Fourier transform. These DWT artifacts are caused by misalignments between the salient features and the wavelet basis functions [13]. These misalignments are unfortunate consequence of decimation.

The SWT does not suffer the same offset problem as the DWT because it omits the decimation step. This is the shift-invariance property that is invaluable for denoising; it
alleviates the pseudo-Gibbs ringing errors. Since the LWWF incorporates the SWT, the denoising step is enhanced.
CHAPTER 3. Level-Weighted Wavelet Fusion

Now that all the necessary background information has been supplied, our proposed method – the level-weighted wavelet fusion (LWWF) algorithm – will be presented. Following this, Chapter 4 will assess its performance results in comparison to other well known techniques.

3.1. Algorithm Classification

Unlike most wavelet fusion techniques, e.g., max fusion, the LWWF does not create a hard decision map to fuse its detail coefficients, i.e., it does not choose to use just one prominent coefficient over the rest. It instead calculates a soft decision map. Where as a hard decision map creates indices to indicate which source image the coefficient is taken from, a soft decision map assigns weights to each coefficient of each source image to be used for a weighted average, therefore alleviating the problem of creating artificial high-frequency transitions.

The calculation of weights is done on a pixel-level and does not directly use its intrascale neighborhood information (i.e., adjacent coefficients of the same scale level). However, it does so indirectly. The main problem with pixel-level fusion with hard decision maps is the resulting unnatural discontinuities caused by its highly nonlinear operation. Although the LWWF is a pixel-level fusion method, the soft decisions allow for smooth neighborhood transitions. During the calculation of the weights, it indirectly uses the continuity of the original images to create natural transitions; in this sense, it does use neighboring information. Furthermore, as will be explained later, it also utilizes
the interscale information, the information between coefficients across scales (decomposition levels).

The LWWF employs the SWT. As explained earlier, unlike the DWT, SWT does not downsample the image at each of its decomposition levels. As a result, the coefficients at each decomposition level are of the same dimensions which is crucial to the calculation of weights. In addition, the SWT also has an inherent shift-invariance property which will be exploited in the denoising experiments of the next chapter.

3.2. Level-Weighted Wavelet Fusion

A high-level block diagram of the LWWF process is shown in Figure 9. As shown in the diagram, the LWWF uses weight maps for fusion. The calculation of the weights is the most crucial part of the algorithm and is what essentially differentiates it. Once they have been calculated, all that is left is a simple weighted average calculation followed by a standard reconstruction operation. In the following, the procedure for the weight calculation will be outlined. Steps for the reconstruction will follow.
3.2.1 Calculation of Weights

The crucial part of this fusion scheme is the calculation of weights. Assume the input images $X_1, X_2, \ldots, X_N$ are registered, i.e., the images are properly aligned, and are of dimensions $2^J \times 2^J$. Let $K \in \{k : J - 1 < k \leq 0\}$ be the curtail level. Then, each of the $N$ source images are decomposed with the SWT to level $K$:

$$
X_1 \rightarrow \{D_{J-1}^h, D_{J-1}^v, D_{J-1}^p, \ldots, D_K^h, D_K^v, D_K^p, C_K^h\}_1
$$

$$
X_2 \rightarrow \{D_{J-1}^h, D_{J-1}^v, D_{J-1}^p, \ldots, D_K^h, D_K^v, D_K^p, C_K^h\}_2
$$

$$
\vdots
$$

$$
X_N \rightarrow \{D_{J-1}^h, D_{J-1}^v, D_{J-1}^p, \ldots, D_K^h, D_K^v, D_K^p, C_K^h\}_N
$$

(20)

where level $J - 1$ represents the finest resolution and level $K$ is the coarsest. For a specific decomposition level $k$ of source image $n$, denote the horizontal detail, vertical
detail, and diagonal detail, and approximation coefficient maps as
\[
\begin{pmatrix}
D_{k,n}^H, D_{k,n}^V, D_{k,n}^D, C_{k,n}^A
\end{pmatrix}
\]. We now summarize the weight calculation steps:

1) For decomposition levels \( k = J - 1, J - 2, \ldots, K \) of source images \( n = 1, 2, \ldots, N \), initialize temporary weight maps:

\[
\begin{align*}
W_{k,n}^H &= |D_{k,n}^H| \\
W_{k,n}^V &= |D_{k,n}^V| \\
W_{k,n}^D &= |D_{k,n}^D| \\
W_{k,n}^A &= |C_{k,n}^A|
\end{align*}
\] (21)

where the absolute value is taken so there are no negative weights. Initially, each wavelet coefficient map is assigned its own weight map, totaling the number of weight maps to \( 4(J - K) \).

2) For all \( n, k \), normalize \( W_{k,n}^H, W_{k,n}^V, W_{k,n}^D, W_{k,n}^A \) with their respective elementwise sums:

\[
\begin{align*}
W_{k,n}^H &\leftarrow \frac{W_{k,n}^H}{\sum_{i,j} W_{k,n}^H[i,j]} \\
W_{k,n}^V &\leftarrow \frac{W_{k,n}^V}{\sum_{i,j} W_{k,n}^V[i,j]} \\
W_{k,n}^D &\leftarrow \frac{W_{k,n}^D}{\sum_{i,j} W_{k,n}^D[i,j]}
\end{align*}
\] (22)
\[ W_{k,n}^d \leftarrow \frac{W_{k,n}^t}{\sum_{i,j} W_{k,n}^t[i,j]} \]

where \([i,j]\) denotes the \((i,j)^{th}\) element of the matrix. This normalization insures that there are no biases towards specific levels in the weighted sum.

3) Now we combine the level specific weight maps into one. For, for each

\[ n = 1, 2, \ldots, N \]

we create four level-sum weight maps \(W_n^H, W_n^V, W_n^D, W_n^A\) with the temporary weight maps \(W_{k,n}, W_{k,n}^H, W_{k,n}^V, W_{k,n}^D, W_{k,n}^A, k = J - 1, J - 2, \ldots, K:\)

\[
\begin{align*}
W_n^H &= \sum_{k=K}^{J-1} W_{k,n}^H \\
W_n^V &= \sum_{k=K}^{J-1} W_{k,n}^V \\
W_n^D &= \sum_{k=K}^{J-1} W_{k,n}^D \\
W_n^A &= \sum_{k=K}^{J-1} W_{k,n}^A
\end{align*}
\]

(23)

This step reduces the number of weight maps for each source image from \(4(J - K)\) to 4. As we will see later, this level-sum utilizes the spatial dependences of the source images, hence, using the interscale information of the wavelet decompositions.

4) Normalize the level-sum weight maps across the source images, i.e., for

\[ n = 1, 2, \ldots, N : \]
where the division is performed elementwise. The resulting level-sum maps satisfy:

\[
\begin{align*}
W_n^H & \leftarrow \frac{W_n^H}{\sum_{n=1}^{N} W_n^H} \\
W_n^V & \leftarrow \frac{W_n^V}{\sum_{n=1}^{N} W_n^V} \\
W_n^D & \leftarrow \frac{W_n^D}{\sum_{n=1}^{N} W_n^D} \\
W_n^A & \leftarrow \frac{W_n^A}{\sum_{n=1}^{N} W_n^A}
\end{align*}
\]  

(24)

where 1 is a matrix of ones. This normalization insures that there are no biases towards specific source images.

The crucial part of the weight calculation is Step 3. To further clarify, the main summation operation for a single source image is illustrated in Figure 10. The normalization step is not depicted. By summing up the levels, different resolution information is combined – the fine levels contribute to the edge and sharp details, and the coarse levels contribute to the smoothness of the operation. Further implications will be further explored in a later section.
3.2.2 Two methods of Reconstruction

With the calculated weights, we can now complete the image fusion process. We will discuss two variations of the LWWF fusion scheme, both being used for the experiments of Chapter 4. The algorithm is summarized as follows:

1) Each of the \( N \) source images are decomposed with the SWT curtailed to the \( K^{th} \) level resulting in \( D_{k,n}^H, D_{k,n}^V, D_{k,n}^D, C_{k,n}^A, n = 1, 2, \ldots, N, \ k = J-1, J-2, \ldots, K, \)

where level \( J-1 \) represents the finest resolution and level \( K \) is the coarsest.
2) For each source image \( n \), four weight maps \( W_n^H, W_n^V, W_n^D, W_n^A \) are calculated as explained in Section 3.2.1.

3) Each decomposition level is fused independently. For each decomposition level \( k \), the detail coefficient maps are fused:

\[
\begin{align*}
F_k^H &= \sum_{n=1}^{N} W_n^H \otimes D_{k,n}^H \\
F_k^V &= \sum_{n=1}^{N} W_n^V \otimes D_{k,n}^V \\
F_k^D &= \sum_{n=1}^{N} W_n^D \otimes D_{k,n}^D
\end{align*}
\]  

(26)

where \( \otimes \) denotes an element-by-element (Hadamard) product.

4) There are two methods to fuse the approximation coefficients together. The first method takes their weighted sum:

\[
F_k^A = \sum_{n=1}^{N} W_n^A \otimes C_{k,n}^A
\]  

(27)

The second method takes the average of all the coefficients:

\[
F_k^A = \frac{1}{N} \sum_{n=1}^{N} C_{k,n}^A
\]  

(28)

Both methods produce good results, and are used in the following discussions.
With the fused coefficient maps, the inverse SWT is applied to obtain the final reconstructed image.

The first method is justified in the same way the detail coefficient fusion is – the weights indicates the importance of the source images. It makes sense to accordingly adjust their contributions to the fusion. The second method follows the common approach for approximation coefficient fusion – taking averages of lowpass filtered images is much like taking the average of averages.

3.3. Denoising Application

As mentioned earlier, denoising is a very important area of image processing, with wavelet denoising being a prominent choice. The steps of the LWWF also require most of the processing to be done in the wavelet domain. It follows that executing both the denoising and fusion with one wavelet decomposition is very efficient; it requires very little extra computations. The optimum time to perform denoising would be before any fusion or weight calculation is taken place. This way, we operate on images with higher SNR. The denoising application is also enhanced with the use of SWT instead of the standard DWT. The introduction of shift-invariance reduces the incidence of unwanted artifacts.

3.4. Algorithm Discussion

This section aims to qualitatively show the advantages of the LWWF over the max fusion algorithm.
3.4.1 Fusion of Image Edges

We mentioned earlier that the discontinuities caused by hard decision maps are manifested as ringing artifacts in the spatial domain. We also claimed that the smooth nature of the soft decision maps alleviates this problem. Now we will qualitatively show the difference between the two by comparing the LWWF, utilizing a soft decision map, over the max fusion that uses a hard decision map.

For this demonstration, an image was created by layering a set of different intensity valued circles. The two images to be fused were created by dividing the original exactly in half as shown in Figure 11. The fusion algorithm should ideally reconstruct the original, whole circles. The problem is that because of the artificial edges (from the division), fusing in the wavelet domain could result in severe artifacts.

![Figure 11: Two images to be processed for fusion.](image)

The fused image using the max fusion algorithm is shown in Figure 12. As expected, the hard decision map created severe pseudo-Gibb phenomenon artifacts. In contrast, the fused images using the LWWF method 1 and LWWF method 2 are shown in
Figure 13 and Figure 14, respectively. Although there are traces of artifacts, they are not as visually profound as with the max fusion.

The hard decision map can be blamed for the disappointing performance of the max fusion. The max operation essentially removes wavelet coefficients (from the unused image) resulting in the abrupt subtraction of signal. When big coefficients are removed, such as the coefficients corresponding to the synthesized edge in Figure 11, the subtraction is much more noticeable. In contrast, the linear nature of the soft decision, i.e., the weighted average, better handles cases where both of the source images contain significant information.

![Fused Image](image)

**Figure 12:** Fused image using the max fusion algorithm.
Figure 13: Fused image using the LWWF method 1.

Figure 14: Fused image using the LWWF method 2.

3.4.2 Utilizing the Interscale Information

Due to the orthogonality of the DWT, information at each level is decorrelated. However, this does not imply that there are no dependencies between them. In fact, high frequency
information at each decomposition level corresponds to disjoint frequency bands of a single image, and thus is inherently dependent to each other for any practical image.

In the LWWF, while the decomposition levels are fused independently of each other, the calculated weights utilize the intrinsic interscale information. If a region of an image displays significant coefficients across the decomposition levels, then it will be emphasized by the resultant weight maps. To illustrate, Figure 15 shows the vertical detail coefficients of a 3-level decomposition using an example image. We can see that the significant detail information is passed on across the levels; the level-sum during weight calculation exploits this phenomenon.

![Figure 15: Vertical coefficients across the different levels of a SWT.](image)

The interscale information is very useful for identifying salient features of an image since it consists of a wide range of high frequency information. On the other hand, noise usually has a limited frequency range, usually residing at the high end of the spectrum. As a consequence, when decomposed with the DWT, it mainly contributes to the finest decomposition level. Since there are only trace amounts of power in the other decomposition levels, it easily follows that the LWWF weights will place less emphasis
on these regions. Hence, overall, the weights of the LWFF extract the signal
discriminating interscale information. This is a distinguished feature in contrast to other
algorithms.

3.4.3 Fusing with Multiple Source Images

A less commonly explored issue with hard decision rules is the case when more than two
images are to be fused together. In the case of only two images, having to discard the less
significant information is not as consequential since one could argue that the salient
feature will typically reside in one or the other. Fusing with more source images greatly
increases the chance of discarding important information, i.e., the salient features will
most likely be distributed among multiple sources. In the extreme case where each image
contains an equivalent amount of information, this elimination will be devastating.

This justifies the use of soft decision maps in this general case of multiple images.

Using weights, each image can contribute to the fusion according to the degree of their
importance. It can efficiently handle the extreme cases: zero-weights are assigned to
unused images, and a uniform average is performed when each image contains equivalent
amount of information. Considering multiple source fusion (i.e., more than two) is
relevant to the experiments in Chapter 4 where eight images are to be fused together.
CHAPTER 4. Results

This chapter presents the experimental results using the proposed LWWF algorithm. Our main test images are a set of multicoil magnetic resonance imaging (MRI) images. We aim to qualitatively and quantitatively show the superiority of the LWWF over the typical methods that are currently used.

4.1. Test Images

Multicoil MRI images are the base images used to test the performance of the LWWF algorithm. These images are multiple MRI snapshots of a subject, simultaneously taken by strategically placed receivers. Figure 16 roughly illustrates the spatial arrangement of the sensors around the subject for imaging. From this arrangement, the received images are shown in Figure 17. Each image is 128x128 pixels in dimension.

![Figure 16: Illustration of the receiver arrangement around the subject to be imaged.](image-url)
In Figure 17, we can identify the regions of complementary and redundant information for fusion (recall Figure 1). For example, between coils 1 and 3, the left-most and right-most regions correspond to the complementary information, and the middle-region corresponds to the redundant information. The goal of image fusion is to combine these eight images into one complete image. It should be noted that these test images have very little noise, i.e., they have high signal-to-noise ratio (SNR). To emulate cases of low SNR, a later subsection will evaluate the algorithms under simulated noise.

![Figure 17: Multicoil MRI images used for testing.](image)

The most common method to combine multicoil images is to take the arithmetic mean between them, rendered in Figure 18. While taking the average may be an acceptable operation in the redundant regions, averaging over complementary regions will result in unwanted blurring, i.e., reduction in contrast. This implies the loss of high frequency content. For example, including coil 1 to resolve the right-most region will
degrade the result as versus not using it at all; coil 1 contains very little information pertaining to that region, but an arithmetic mean will nevertheless weigh all coils uniformly. The desire to optimize the contrast over all the regions is the primary motivation for using a different approach. The LWWF accomplishes this by performing a pixel-level weighted average over the wavelet space, where each coefficient is given a specific weight. This differs from some other weighted average schemes where entire source images are given a single weight value. The LWWF gives the flexibility to emphasize or deemphasize specific regions.

![Average](image.png)

**Figure 18:** Fusion by taking the arithmetic mean between the source images.

### 4.2. LWWF Reconstructed Images

The reconstructed images of LWWF method 1 and method 2 are shown in Figure 19 and Figure 20, respectively. As mentioned before, the arithmetic mean, shown in Figure 18, is the most commonly used method for multicoil fusion, and hence will be used for
comparison. To assess how the LWWF compares with other wavelet fusion techniques, the pixel-by-pixel wavelet max fusion of Section 2.5.3 will also be compared. The wavelet max fusion is the most commonly known wavelet-based fusion method – the reconstructed image using this fusion is shown in Figure 21. There is a plethora of variants to this max fusion scheme such as looking over a window of values and employing consistency checks on the decision maps. Experimentally, it was found that these variants do not significantly change the reconstructed image. Hence, the pixel-by-pixel wavelet max fusion will be used to represent this entire class of algorithms.

![LWWF Method 1](image)

**Figure 19**: Reconstruction using LWWF method 1
Figure 20: Reconstruction using LWWF method 2

Figure 21: Reconstruction by wavelet max fusion
4.3. **Visual Analysis**

For visual comparison, the four reconstructed images are plotted again in Figure 22 with a fixed grayscale range.

![Images of reconstruction methods](image)

**Figure 22:** Comparison of reconstruction methods

It’s very apparent that the average fusion lacks the dynamic range of grayscale values the other methods display. As for the max fused image, there are many strong artifacts around the edges. For easier comparison, zoomed plots of different brain regions are shown in Figure 23, Figure 24, Figure 25, and Figure 26.

Figure 23 and Figure 24 show similar differences: the average fusion displays poor contrast and the max fusion exhibits severe artifacts. The artifacts of max fusion are the dark areas near the edges of the image. While the LWWF method 2 shows similar artifacts, they are relatively mild. Overall, LWWF method 1 looks the best: there are no
visible artifacts and it displays better contrast between the different regions. The
differences between the reconstructed images in Figure 25 and Figure 26 are more subtle.
The contrast difference between the images is not as distinguishable, but the wavelet max
fusion still displays undesirable artifacts evidenced by grains near the edges.

Figure 23: Zoomed plots of the left brain region.

Figure 24: Zoomed plots of the right brain region.
Figure 25: Zoomed plots of the upper brain region.

Figure 26: Zoomed plots of the lower brain region.
It should be noted that, unlike average fusion, the reconstructed images using the wavelet transform contain negative pixel values due to approximation errors and artifacts. Since negative values are erroneous, they have all been set to zero. As expected, due to the strong contrast, the negative values are most prolific near the border of the brain image. This is due to the elimination or attenuation of significant coefficients.

We can assess the severity of these artifacts by looking at the magnitude of the negative values. The magnitudes of the negative pixels for the wavelet-based fused images are shown in Figure 27. A good fusion scheme should aim to minimize these values. From Figure 27, it is clear that the wavelet max fusion (left image) creates the strongest artifacts. The LWWF method 2 (right image) also contains similar artifacts at a lower scale, and LWWF method 1 (middle image) is cured of this problem. This analysis strongly supports the use of a smooth weighted average scheme over a discontinuous maximum scheme.

![Figure 27: Magnitude plots of the negative valued pixels.](image)

Overall, visual analysis of the reconstructed images is very subjective and, hence, a quantitative measure of the fusion performances is highly desirable. Unfortunately,
there is no generally accepted performance measure for image fusion. In light of this, different measurements will be calculated in an attempt to quantitatively show some of the desirable characteristics of the fusion algorithms.

4.4. Power Sum

One of the main goals of image fusion is to improve the overall contrast in the image. Contrast corresponds to the high frequency content of the signal. For easy frequency analysis, the discrete Fourier transform is employed. Specifically, the power spectra of the fused images are calculated with the 2-dimensional discrete Fourier transform and the high frequency power is calculated by summing coefficients within specific bandwidths. Unfortunately, both high contrast and high frequency noise can contribute to this sum. Thus, this alone can not be used as a final judgment of performance.

The log-scaled Fourier spectra of the reconstructed images are shown in Figure 28. These spectra have been shifted so the center point corresponds to the DC component. We can immediately see the lack of high-frequency power in the average spectrum. This is not surprising since some of the individual source images have directionally oriented edges as illustrated in Figure 29. Since the Fourier transform is a linear transformation, the average operation in the image domain amounts to an average operation in the Fourier domain.
Figure 28: Log scaled Fourier spectra of the reconstructed images.

Figure 29: Spectra of the source images.
To measure the high frequency power, several progressive masks were created to encapsulate the power within specific bandwidths. Each mask is designed to reveal a unique aspect of the power distribution along the frequencies. For mask 1, a circle mask is iteratively enlarged and coefficients outside the mask are summed. This is followed by normalization by the total sum of the spectrum, giving a fractional contribution value. To elucidate, Figure 30 plots the masks at several stages of progression. The starting circle radius is 0 pixels (i.e., the entire spectrum is summed), and progresses until it is at a radius of 75 pixels. Recall that each of the reconstructed spectra is $128 \times 128$ pixels in dimension. A plot of the normalized sum versus the radius of the circle mask is shown in Figure 31.

As expected, the average spectrum has the least high frequency power: Figure 31 quantitatively shows the reduction in contrast resulting from the average operation. It reinforces the conjecture that it is a suboptimal fusion method. The other noticeable feature of Figure 31 is that the max fusion retains the greatest high frequency power. But this is largely due to the copious amounts of noise and artifacts, consequential to the crude max operation. Hence, while the plot indicates that the max fusion has the most high-frequency power, it is still inconclusive about how visually acceptable the actual image is. The plots of Figure 22 support this argument. In fact, we will later show that a significant portion of the high-frequency power is contributed by unwanted artifacts.
Figure 30: Progression of mask 1.

Figure 31: Normalized sums of mask 1.

Much of the power resides in the low frequency region of the spectrum, including the direct-current (DC) component. To see the power distribution away from this band,
an annular mask, referred to as mask 2, was created. The inner circle, fixed at a 5 pixel radius, was empirically chosen to circumscribe the low-frequency band. The initial radius of the outer circle is 10 pixels and progresses to a 75 pixel radius. The coefficients within the annular region for each reconstructed image are summed and normalized by their respective total power sums. To illustrate, Figure 32 plots the masks at several stages of the progression. A plot of the normalized sum versus the radius of the outer circle in the mask is shown in Figure 33.

It's interesting to see the points where the curves intersect. For example, the average fusion, max fusion, and LWWF method 2 intersects when the outer circle radius is approximately 32 pixels. After this point, the curve of the average fusion falls below the rest, indicating the lack of power outside this radius. Conversely, the max fusion starts below the rest of the curves prior to the radius 32 pixels mark and ends up above. The final data point of each of the curves in Figure 33 shows the total contribution of the high-frequency power for each reconstruct method.

![Mask 2 Diagram]

Figure 32: Progression of mask 2.
We now want to approximately see how much each frequency contributes to the overall power of the spectrum. To do this, mask 3 was created. Mask 3 is an annular mask that maintains a constant bandwidth of 10 pixels. Initially, the inner circle is 5 pixels in radius. To maintain a constant bandwidth, the outer circle is correspondingly initiated at 15 pixels in radius. The inner circle is enlarged to an ending radius of 65 pixels, and the outer circle is adjusted each iteration to maintain a constant 10 pixel bandwidth. At each step, the coefficients within the annular region are summed and normalized by their respective total power sums. To illustrate, Figure 34 plots the masks at several stages of the progression. A plot of the normalized sum versus the radius of the inner circle in the mask is shown in Figure 35.

Aside from the low frequency power (power within 5 pixel radius circle) Figure 35 shows that the biggest contribution of power occurs when the inner circle of the
annular region is 11 pixels in radius. Figure 35 also shows that, relatively, the LWWF method 1 only has slightly more high frequency power than the average fusion, while the LWWF method 2 and the max fusion contains significantly more.

Figure 34: Progression of mask 3.

Figure 35: Normalized sums of mask 3.
Overall, the masks prove the inability of the average fusion to retain high frequency components if the signal. We also saw that the max fusion retains the most high frequency content. But as we will show later, much of this is contributed by noise and unwanted artifacts.

4.5. Mutual Information

If we model each pixel value in the reconstructed image as a random variable, then we can statistically measure the similarity between two images by measuring the similarity, or the “distance,” between their probability distributions. Conveniently, the mutual information between two random variables does just that. The mutual information is a measure of the statistical independence between two distributions – a higher value indicates a higher dependence and the mutual information of zero implies statistical independence. Intuitively then, an optimally fused image should maintain a high mutual information between each of the source images that produced it [14]. It then follows that we can use this as a measure of fusion performance.

The formula for the mutual information between two random variables $X$ and $Y$ is:

$$I(X,Y) = \sum_{x,y} P_{XY}(x,y) \log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)} ,$$

(29)

where $P_{XY}(x,y)$ is the joint probability density between $X$ and $Y$, and $P_X(x)$, $P_Y(y)$ are their respective marginal probability densities. Treating images as a collection of random
samples, we quantize the fusion performance by calculating the average mutual information between the reconstructed image and each of the source images:

\[
M = \frac{1}{N} \sum_{n=1}^{N} I(X_n, R)
\]  

(30)

where \(X_n, n = 1, 2, \ldots, N\) are the source images used for fusion and \(R\) is the fused image. We obtain sample joint and marginal probability densities by calculating normalized histograms using pixels within a defined mask shown in Figure 36. The white region represents the region of pixels to be used for calculation. The black region corresponds to the background of the image, which essentially contains no useful information. Results using this measure are shown in Table 1.

![Figure 36: Mask used to calculate mutual information. Pixels in the white region are used.](mask.png)
Looking at Table 1, LWWF method 1 maintains the highest average mutual information among all the fusion methods, with the max fusion trailing close to it. On an individual coil basis, the two methods are approximately equal. In contrast, the average fusion gives the smallest average mutual information by a significant margin. While this measure gives us some aspect of the fusion algorithm, due to the initial assumptions, it can not be used as a sole measure of quality. However, it gives us a statistical perspective of the fusion algorithms.

4.6. Fusion in the Presence of Noise

It was mentioned earlier that the source images used for fusion contain relatively little noise. For practical reasons, it is desirable to test the fusion algorithms in the presence of different levels of noise. In the following, noise will be synthetically injected onto the MRI images according to a defined noise model. The signal-to-noise ratio (SNR) will then be formulated. We will show that wavelet denoising is a very effective technique to increase SNR with minimal extra computations.

4.6.1 Noise Model
The acquired MRI signal $X_c$ is complex. Upon signal acquisition, a widely accepted model depicts the real and imaginary components being corrupted by additive white Gaussian noise [15]. If we denote $D$ as the noiseless complex signal, then the acquired signal is represented in the following equation:

$$X_c = [\text{Re}(D) + \eta] + i[\text{Im}(D) + \eta], \quad (31)$$

where $\eta$ is a zero-mean Gaussian random variable, with variance $\sigma^2_n$. We operate on the magnitude signal, defined by:

$$X = \left(\left[\text{Re}(X_c)\right]^2 + \left[\text{Im}(X_c)\right]^2\right)^{1/2}, \quad (32)$$

and thus the new noise is represented by a Rayleigh distribution.

Let $m$ be the maximum pixel value among all the magnitude source images:

$$m = \max_n \left\{ \max_{i,j} X_n[i,j] \right\}, \quad (33)$$

where $X_n[i,j]$ denotes the $(i,j)^{th}$ element of the $n^{th}$ source image $X_n$. To simulate different levels of noise, zero-mean Gaussian random variables with standard deviation of $am$, with varying values of $a$, are independently added to the real and imaginary components of the source images. Simulations show that a representative set of values for $a$ is $\{0.015, 0.030, 0.045, \ldots, 0.150\}$; any more noise make the tests unpractical, and
the chosen increments are sufficient to clearly delineate the SNR plots. The constant \( a \) will be referred to as the noise factor. To illustrate, Figure 37 shows the effect of the noise added on the first source image for the given set of noise factor values.

![Figure 37: Effect of additive noise on source 1 for various noise factor values.](image)

### 4.6.2 SNR Formulation

To assess the performance of the fusion algorithms, the signal-to-noise ratio (SNR) is calculated as follows. Denote \( R \) as the reconstructed image without the synthesized noise and denote \( R_a \) as the reconstructed image with synthesized noise using scaling parameter \( a \). If we make the assumption that \( R \) is the noiseless image, then, the remaining noise is calculated as:

\[
N = R_a - R.
\]  

(34)
We define the SNR of the image as the logarithm of the variance of the signal over the variance of the noise, i.e.,:

$$SNR = \log \left( \frac{\text{var}\{R\}}{\text{var}\{N\}} \right),$$  \hspace{1cm} (35)

where, again, only the pixels within the mask shown in Figure 36 are used to calculate variances. It should be noted that the noise map $N$ captures the unwanted artifacts from wavelet fusion as well as the remnants of the additive noise. This means that there is a tradeoff between the reduction of additive noise and the creation of unwanted artifacts. As seen in the next subsection, this is an important issue when dealing with wavelet denoising. The plots of the SNR for the various fusion algorithms for varying values of $a$ are shown in Figure 38.

![SNR comparison graph](image)

**Figure 38:** SNR for various noise levels.
Figure 38 clearly shows that the LWWF method 1 achieves the best SNR. LWWF method 2 and average fusion are similar, and max fusion gives the worst results by a significant margin. The poor SNR performance of the max fusion algorithm can be attributed to two factors. First of all, the maximum operation has a preference for high-frequency noise, and second, the false edges manifest in the reconstructed image as ringing artifacts. This plot further validates the point as to why the frequency power plots of Figure 31, Figure 33, and Figure 35 are inconclusive about the fusion performance: high frequency power can be attributed to noise. While the LWWF algorithms create artifacts as well, they are not as severe due to the smooth nature of the fusion.

4.6.3 Wavelet Denoising with the Universal Threshold

To mitigate signal loss due to noise, image denoising is commonly performed on the source images before fusion. Wavelet denoising is known as an effective technique for this purpose. As explained earlier, a well-known and widely used method is to apply a judicious soft threshold to the coefficients in the wavelet domain. This will be used for the purpose of this experiment.

Since denoising and image fusion are both executed in the wavelet domain, we can conveniently do both with only one pass of the wavelet transform. By doing this, we save a significant amount of additional computations. Since denoising is optional, it can be thought of as an extended application to using the LWWF methods. To demonstrate its performance, wavelet denoising is performed on the previously synthesized noisy images.
of Figure 37. As an initial test, soft thresholding (see Figure 8) will be applied using the universal threshold. Recall from Section 2.6 that the universal threshold is defined as:

\[ T = \sigma_n \sqrt{2 \log N}, \]  

(36)

where \( N \) is the number of pixels in the image, and \( \sigma_n \) is the standard deviation of the noise. The standard deviation of the noise is estimated as:

\[ \hat{\sigma}_n = \frac{\text{median}(\| D_{j-1}^0 \|)}{0.675} \]  

(37)

where \( D_{j-1}^0 \) is the sequence of diagonal detail coefficients on the finest level of wavelet decomposition. A different value of \( T \) is calculated for each source image.

Figure 39 shows the results of applying wavelet denoising. For comparison, the SNR plot for average fusion without denoising is superimposed on the figure. By observation, the average images perform better when the noise factor is relatively small. When the noise level increases, the denoising methods—LWWF in particular—demonstrate superior performance. As before, the max fusion achieves the worst SNR in all cases. For an easy comparison, Figure 40 combines the plots for all fusion methods with and without applying wavelet denoising. The dashed lines represent the SNR curves without denoising. It clearly shows at what noise factors denoising via the universal threshold can be effectively applied. Namely, the SNR is higher without denoising when the noise factor is less than 0.030.
Figure 39: SNR for fixed wavelet denoising over various noise levels.

Figure 40: Plot depicting SNR for cases with and without the applying wavelet denoising.

Even though denoising, in general, improves SNR, the denoising via universal thresholds actually degraded the images in some cases. This implies that the thresholds...
were poor estimates, i.e., they are not general enough for the range of noise scales experimented.

4.6.4 Wavelet Denoising with an Empirically Derived Optimal Threshold

In response to the underperformance of the universal threshold for noise factors \( a : 0 \leq a \leq 0.030 \), an optimal, adaptive threshold was calculated for each noise level. These thresholds were optimized empirically. We should emphasize the main tradeoff in choosing a threshold value: while a large enough threshold value will essentially eliminate all the noise, it will also eliminate significant coefficients, creating worse ringing artifacts. Since the SNR accounts for both these characteristics, there should be a unique balancing point.

To find the optimal threshold, we start with a slightly modified universal threshold equation using the standard deviation estimate:

\[
T^k_{\text{univ}} = \frac{\text{median}(|D^P_k|)\sqrt{2\log N}}{0.675},
\]

where the threshold has been generalize to include all levels of decomposition, not just the finest level \( k = J - 1 \). Hence, unlike the original universal threshold, all the detail coefficients will be operated on. This implies that we must find multiple optimal thresholds. We will do this empirically where each threshold is calculated independently of each other. First, the equation for the universal threshold is generalized:
where we have introduced the tuning parameter $c_k$. By varying the tuning parameter $c_k$ and calculating the resultant SNR values, we can find the ad hoc optimal thresholds.

Recall that for the experiments, we decompose the MRI images down to 3 levels. Using the established SWT notations, this means we start with a level 6 decomposition and curtail it down to level 4, since each image is of dimensions $128 \times 128$ pixels. In the following, we will attempt to sequentially optimize the coarsest level to the finest, i.e., we will optimize level 4, level 5, and level 6, in that order. The tuning parameter $c_k$ is ranged from 0.1 to 10, with increments of 0.1. For each value of $c_k$, the SNR using LLWF method 1 and method 2 is calculated. Additionally, the SNR using the average fusion, with and without thresholding, is calculated – if the SNR obtained with thresholding can not surpass the case without, then this implies that thresholding should not be performed.

Experimentally, we found that thresholding any decomposition level other than the finest level results in degradation. For illustration, we focus on optimizing the thresholds for the case when the noise factor $\alpha$ is 0.075. Figure 41 shows a plot of the SNR versus the tuning parameter for level 4 wavelet coefficients. The curves for the LLWF methods are monotonically increasing and converge to horizontal asymptotes. Comparing with Figure 40, we see that the asymptote values are exactly the SNR values for the case when no denoising is performed. The average fusion plots provide further evidence of this phenomenon: in Figure 41, the SNR curve of the denoised average

$$T_k = \frac{\text{median}(|D_k^0|)}{c_k \sqrt{2\log N}}. \quad (39)$$
fusion clearly converges to the constant, unthresholded average fusion. Notice that the peak SNR value occurs at the limit $c_k \to \infty$. The corresponding threshold converges to $T_k \to 0$, implying that no coefficients are thresholded. In conclusion, the optimal thresholding option for level 4 coefficients is to not threshold at all.

![SNR comparison](image)

**Figure 41:** SNR values for varying thresholds for decomposition level 4 and noise factor 0.075.

Results for level 5 coefficients are very similar to level 4 coefficients. Figure 42 shows the plot of the SNR versus the tuning parameter. Although the curves for the LWWF methods are not monotonic, the peak SNR occurs at the horizontal asymptotes. As before, the asymptote values are exactly the SNR values for the case when no denoising is performed. As for the average fusion, although the SNR for the denoised average fusion converges to the case where no denoising is performed, the peak SNR value occurs with a tuning parameter of approximately 2. However, our concern is in the

69
optimization of the SNR values for the LWWF. By this, we conclude that the optimal thresholding option for level 5 coefficients is to not threshold at all.

![SNR Comparison Graph]

Figure 42: SNR values for varying thresholds decomposition level 5 and noise factor 0.075.

So far, we have seen that it is best to leave the coarser levels (i.e., any level aside from the finest) unthresholded. However, as Figure 43 shows, thresholding level 6 coefficients can significantly increase the SNR. As \( c_i \to \infty \), the SNR values for all denoising methods depicted converge to a much lower value than the initial ones. For this specific case of noise factor 0.075, a tuning parameter value of 0.65 achieves the peak LWWF SNR values. The significant dip in SNR is most likely resultant from the inclusion of noise, i.e., the threshold is too low to include the majority of noise coefficients. Overall, these results verify the effect noise has on the wavelet coefficients – most of its energy is distributed on the finest scale.
Figure 43: SNR values for varying thresholds for decomposition level 6 and noise factor 0.075.

The procedure for finding the optimal thresholds for the case when the noise factor was 0.075 was repeated for the entire set of experimental noise factor values. The resulting SNR values using these optimized adaptive thresholds are shown in Figure 44. For comparison, the SNR plot for average fusion without denoising is superimposed on the figure. Figure 45 combines the plots for all fusion methods with and without applying the adaptive wavelet denoising. This time, we can clearly see that denoising achieves higher SNR for all the experimental noise levels.

In general, we have shown that applying wavelet denoising is always a preferable option, and almost mandatory for cases with strong noise. With the LWWF, this operation is done with minimal extra computational cost.
Figure 44: SNR for adaptive wavelet denoising over different noise scales.

Figure 45: Plot depicting SNR for cases with and without the applying adaptive wavelet denoising.
CHAPTER 5. Conclusion

Wavelet-based fusion methods have proven to be very successful and are prominently employed in current works. The fact that it is a MRA technique makes it very suitable for this application. Furthermore, unique properties of the wavelet transform make it outperform other MRA techniques such as the Laplacian transform. We have followed this trend by basing the proposed algorithm on the stationary wavelet transform.

Despite the popularity of hard decision maps for fusion, we have opted to utilize a soft decision scheme. We have fully justified this course and proven its superiority in many practical cases. As a result of these experiments, we see that soft decision maps offer much more flexibility, and can be seen as a generalization of hard decision maps.

The main concern in the use of soft decision maps is the judicious selection of weights. For the LWWF, we have provided a simple and effective method that exploits the intrascale and interscale information of the wavelet coefficients. Combined with a linear weighted average, it expertly handles sensitive regions such as edges. As demonstrated in Chapter 4, this method outperforms both the average fusion and also the benchmarking, wavelet-based max fusion technique.

We earlier explained the necessity to perform denoising techniques to increase the SNR of the source images. Wavelet denoising is a prominent candidate for the job. The convenience of being able to perform both the fusion and denoising with one pass of the SWT is a substantial merit of the LWWF. We have demonstrated its effectiveness in Section 4.6.

Overall, the LWWF is a very effective and efficient technique for image fusion. The denoising application supports this case. In future work, we hope to build on this
algorithm by incorporating other statistical information such as regional variance and mutual information. The success of these early results indicates a promising future.
Appendix: MATLAB Codes

For reference, this chapter will show all the main MATLAB codes used during the experiments. This includes codes for the various fusion methods and their performance measurements.

A.1. Preliminaries

The steps common to all the fusion methods experimented (i.e., average fusion, max fusion, and LWWF) include the loading of images and definition of some parameters (e.g., denoising parameters) as shown:

```matlab
% Define parameters

% Additive noise options
nf = 0.075; % Noise factor
n=8; % Number of source images

% SWT options
filt = 'bior3.7'; % Wavelet filter to use
lev = 3; % Number of levels of decomposition

% Wavelet denoising options
flag_waveshrink = 1; % Perform denoising?
thrLvl = 1; % Lowest decomposition level to threshold
tvsc = [0.55]; % Define a threshold value for each decomposition level

% LOAD images
load c1_tl_im;
load c2_tl_im;
load c3_tl_im;
load c4_tl_im;
load c5_tl_im;
load c6_tl_im;
load c7_tl_im;
load c8_tl_im;

% Find max pixel value among all the input images
pmax = max(abs(c1_tl_im(:));abs(c2_tl_im(:)); ...
                       abs(c3_tl_im(:));abs(c4_tl_im(:)); ...
                       abs(c5_tl_im(:));abs(c6_tl_im(:)); ...
                       abs(c7_tl_im(:));abs(c8_tl_im(:)));

% Standard deviation of the additive noise
```

75
nvar = nf * pmax;

/* Add noise to source images */
c1_t1_im = c1_t1_im + nvar*randn(128,128) + nvar*i*randn(128,128);
c2_t1_im = c2_t1_im + nvar*randn(128,128) + nvar*i*randn(128,128);
c3_t1_im = c3_t1_im + nvar*randn(128,128) + nvar*i*randn(128,128);
c4_t1_im = c4_t1_im + nvar*randn(128,128) + nvar*i*randn(128,128);
c5_t1_im = c5_t1_im + nvar*randn(128,128) + nvar*i*randn(128,128);
c6_t1_im = c6_t1_im + nvar*randn(128,128) + nvar*i*randn(128,128);
c7_t1_im = c7_t1_im + nvar*randn(128,128) + nvar*i*randn(128,128);
c8_t1_im = c8_t1_im + nvar*randn(128,128) + nvar*i*randn(128,128);

/* Store all input data into a 3D matrix 'x' */
x(:,:,1) = abs(c1_t1_im);
x(:,:,2) = abs(c2_t1_im);
x(:,:,3) = abs(c3_t1_im);
x(:,:,4) = abs(c4_t1_im);
x(:,:,5) = abs(c5_t1_im);
x(:,:,6) = abs(c6_t1_im);
x(:,:,7) = abs(c7_t1_im);
x(:,:,8) = abs(c8_t1_im);

Most of the variables are self explanatory. There is a flag for wavelet denoising since it's an optional method. The variable pmax, which is the max pixel value among all the source images, is used with the user-defined noise factor for the additive noise. If no noise is desired, then the noise factor nf can be set to 0. Notice that all the source images are stored in the 3D matrix x.

A.2. Image Fusion

Codes for all the experimented fusion methods will be displayed in this section. This includes the average fusion, max fusion, and the 2 methods of LWWF.

A.2.1 Average Fusion

While the average fusion by itself is very basic (it is literally just the average of all the source images), it becomes slightly more complicated with the incorporation of wavelet denoising as seen below:
% ***** Perform 3-level SWT and store the coefficients *****
for k = 1:n
    % Perform SWT
    % [swa,swh,swv,swd] = mri_swtx(x(:,1,k),lev,filt);
    % Optional: Perform wavelet denoising, via soft thresholding
    if(flag_waveshrink)
        % Denoise up to level thrlvl
        for kk = 1:thrlvl
            % Get median values
            medh = median(abs(reshape(swh(:,1,kk),1,[1]));
            medv = median(abs(reshape(swv(:,1,kk),1,[1]));
            medd = median(abs(reshape(swd(:,1,kk),1,[1]));

            % Find level-dependent global threshold
            wthrh = sqrt(2*log(prod(size(swh(:,1,kk)))*medh/tvec(v));
            wthrv = sqrt(2*log(prod(size(swv(:,1,kk)))*medv/tvec(v));
            wthrd = sqrt(2*log(prod(size(swd(:,1,kk)))*medd/tvec(v));

            % Denoise
            tmp = (abs(swh(:,1,kk))-wthrh); tmp = (tmp+abs(tmp))/2;
            swh(:,1,kk) = sign(swh(:,1,kk)).*tmp;
            tmp = (abs(swv(:,1,kk))-wthrv); tmp = (tmp+abs(tmp))/2;
            swv(:,1,kk) = sign(swv(:,1,kk)).*tmp;
            tmp = (abs(swd(:,1,kk))-wthrd); tmp = (tmp+abs(tmp))/2;
            swd(:,1,kk) = sign(swd(:,1,kk)).*tmp;
        end
    end

    % Store LEVEL 1 coefficients
    WA1(k,1,:) = swa(:,1,1);
    WH1(k,1,:) = swh(:,1,1);
    WV1(k,1,:) = swv(:,1,1);
    WD1(k,1,:) = swd(:,1,1);

    % Store LEVEL 2 coefficients
    WA2(k,1,:) = swa(:,2,1);
    WH2(k,1,:) = swh(:,2,1);
    WV2(k,1,:) = swv(:,2,1);
    WD2(k,1,:) = swd(:,2,1);

    % Store LEVEL 3 coefficients
    WA3(k,1,:) = swa(:,3,1);
    WH3(k,1,:) = swh(:,3,1);
    WV3(k,1,:) = swv(:,3,1);
    WD3(k,1,:) = swd(:,3,1);
end

% ***** Perform average fusion *****
mapsize = size(x(:,1,1));
avgWA1 = mean(reshape(WA1,n,prod(mapsize)),1);
avgWA1 = reshape(avgWA1,mapsize);
avgWA2 = mean(reshape(WA2,n,prod(mapsize)),1);
avgWA2 = reshape(avgWA2,mapsize);
avgWA3 = mean(reshape(WA3,n,prod(mapsize)),1);
avgWA3 = reshape(avgWA3,mapsize);
avgWH1 = mean(reshape(WH1,n,prod(mapsize)),1);
avgWH1 = reshape(avgWH1,mapsize);
avgWH1 = reshape(avgWH1,mapszie);
avgWH2 = mean(reshape(WH2,n,prod(mapszie),1));
avgWH3 = reshape(avgWH3,mapszie);
avgWH1 = mean(reshape(WH1,n,prod(mapszie),1));
avgWH2 = reshape(avgWH1,mapszie);
avgWH2 = mean(reshape(WH2,n,prod(mapszie),1));
avgWH2 = reshape(avgWH2,mapszie);
avgWH3 = mean(reshape(WH3,n,prod(mapszie),1));
avgWH3 = reshape(avgWH3,mapszie);

avsWH2 = mean(reshape(WH2,n,prod(mapszie),1));
avsWH2 = reshape(avsWH2,mapszie);
avsWH3 = mean(reshape(WH3,n,prod(mapszie),1));
avsWH3 = reshape(avsWH3,mapszie);

t ***** Reconstruct *****

# Piece all levels together
sWa(:,:,1) = avgWA1;
sWa(:,:,2) = avgWA2;
sWa(:,:,3) = avgWA3;
sWh(:,:,1) = avgWH1;
sWh(:,:,2) = avgWH2;
sWh(:,:,3) = avgWH3;
sWV(:,:,1) = avgWV1;
sWV(:,:,2) = avgWV2;
sWV(:,:,3) = avgWV3;
sWd(:,:,1) = avgWD1;
sWd(:,:,2) = avgWD2;
sWd(:,:,3) = avgWD3;

# Reconstruction: Inverse SWT
Favg = mri_iswt2(swa,swh,sWv,sWd,filt1);

# Remove negative artifacts
Favg(Favg<0) = 0;

Note that the code contains the custom functions mri_swt2 and mri_iswt2. These are slightly modified versions of the functions swt2 and iswt2 of the MATLAB Wavelet Toolbox [16]. Specifically, it removes the unwanted shifting of the wavelet coefficients between levels. This shift is disastrous for the LWFF algorithms when coefficients between levels are compared.
A.2.2 Max Fusion

We now show the code for the max fusion in the wavelet domain. The focus of this code is the creation of a hard decision map for the source images.

```matlab
% ***** Perform SWT; Store coefficients *****
for k = 1:n
    [swa,swh,swv,swd] = mri_swt2(x(:,1,k),lev,filt);

    % Optional: Perform wavelet denoising, via soft thresholding
    if(flag_waveshrink)
        % Denoise up to level thrvl
        for kk = 1:thrvl
            % Get median values
            medh = median(abs(reshape(swh(:,1,kk),1,[])));
            medv = median(abs(reshape(swv(:,1,kk),1,[])));
            medd = median(abs(reshape(swd(:,1,kk),1,[])));

            % Find level-dependent global thresholds
            wthrh = sqrt(2*log(prod(size(swh(:,1,kk)))*medh/tvec(v));
            wthrv = sqrt(2*log(prod(size(swv(:,1,kk)))*medv/tvec(v));
            wthrd = sqrt(2*log(prod(size(swd(:,1,kk)))*medd/tvec(v));

            % Denoise
            tmp = (abs(swh(:,1,kk))-wthrh); tmp = (tmp+abs(tmp))/2;
            swh(:,1,kk) = sign(swh(:,1,kk).*tmp);
            tmp = (abs(swv(:,1,kk))-wthrv); tmp = (tmp+abs(tmp))/2;
            swv(:,1,kk) = sign(swv(:,1,kk).*tmp);
            tmp = (abs(swd(:,1,kk))-wthrd); tmp = (tmp+abs(tmp))/2;
            swd(:,1,kk) = sign(swd(:,1,kk).*tmp);
        end
    end

    % Store LEVEL 1 coefficients
    WA1(k,:) = swa(:,1,1);
    WH1(k,:) = swh(:,1,1);
    WV1(k,:) = swv(:,1,1);
    WD1(k,:) = swd(:,1,1);

    % Store LEVEL 2 coefficients
    WA2(k,:) = swa(:,1,2);
    WH2(k,:) = swh(:,1,2);
    WV2(k,:) = swv(:,1,2);
    WD2(k,:) = swd(:,1,2);

    % Store LEVEL 3 coefficients
    WA3(k,:) = swa(:,1,3);
    WH3(k,:) = swh(:,1,3);
    WV3(k,:) = swv(:,1,3);
    WD3(k,:) = swd(:,1,3);
end

% ***** Create hard decision decision map *****
mapsize = size(x(:,1,1));
```
\* Initialize decision maps

\begin{verbatim}
WAImap = zeros(mapsize);
WHlmap = zeros(mapsize);
WVImap = zeros(mapsize);
WDlmap = zeros(mapsize);
WA2map = zeros(mapsize);
WH2map = zeros(mapsize);
WV2map = zeros(mapsize);
WD2map = zeros(mapsize);
WA3map = zeros(mapsize);
WH3map = zeros(mapsize);
WV3map = zeros(mapsize);
WD3map = zeros(mapsize);
\end{verbatim}

\* Create decision maps

\begin{verbatim}
for a = 1:n
    \* Initialize dummy 'compare' maps
    tempWA1 = logical(ones(mapsize));
    tempWH1 = logical(ones(mapsize));
    tempWVI = logical(ones(mapsize));
    tempWDl = logical(ones(mapsize));
    tempWA2 = logical(ones(mapsize));
    tempWH2 = logical(ones(mapsize));
    tempWV2 = logical(ones(mapsize));
    tempWD2 = logical(ones(mapsize));
    tempWA3 = logical(ones(mapsize));
    tempWH3 = logical(ones(mapsize));
    tempWV3 = logical(ones(mapsize));
    tempWD3 = logical(ones(mapsize));

    for b=1:n
        \* Compare which map has the greater value, iteratively
        cmpWA1 = abs(squeeze(WA1(a, :, :))) > abs(squeeze(WA1(b, :, :)));
        cmpWH1 = abs(squeeze(WH1(a, :, :))) > abs(squeeze(WH1(b, :, :)));
        cmpWVI = abs(squeeze(WVI(a, :, :))) > abs(squeeze(WVI(b, :, :)));
        cmpWDl = abs(squeeze(WDl(a, :, :))) > abs(squeeze(WDl(b, :, :)));
        cmpWA2 = abs(squeeze(WA2(a, :, :))) > abs(squeeze(WA2(b, :, :)));
        cmpWH2 = abs(squeeze(WH2(a, :, :))) > abs(squeeze(WH2(b, :, :)));
        cmpWV2 = abs(squeeze(WV2(a, :, :))) > abs(squeeze(WV2(b, :, :)));
        cmpWD2 = abs(squeeze(WD2(a, :, :))) > abs(squeeze(WD2(b, :, :)));
        cmpWA3 = abs(squeeze(WA3(a, :, :))) > abs(squeeze(WA3(b, :, :)));
        cmpWH3 = abs(squeeze(WH3(a, :, :))) > abs(squeeze(WH3(b, :, :)));
        cmpWV3 = abs(squeeze(WV3(a, :, :))) > abs(squeeze(WV3(b, :, :)));
        cmpWD3 = abs(squeeze(WD3(a, :, :))) > abs(squeeze(WD3(b, :, :)));

        tempWA1 = tempWA1 & cmpWA1;
        tempWH1 = tempWH1 & cmpWH1;
        tempWVI = tempWVI & cmpWVI;
        tempWDl = tempWDl & cmpWDl;
        tempWA2 = tempWA2 & cmpWA2;
        tempWH2 = tempWH2 & cmpWH2;
        tempWV2 = tempWV2 & cmpWV2;
        tempWD2 = tempWD2 & cmpWD2;
        tempWA3 = tempWA3 & cmpWA3;
        tempWH3 = tempWH3 & cmpWH3;
        tempWV3 = tempWV3 & cmpWV3;
        tempWD3 = tempWD3 & cmpWD3;
    end
\end{verbatim}

\* Map gets filled with an integer value, corresponding to the image index

80
WA1map = WA1map + a*(WA1map==0).*tempWA1;
WH1map = WH1map + a*(WH1map==0).*tempWH1;
WV1map = WV1map + a*(WV1map==0).*tempWV1;
WD1map = WD1map + a*(WD1map==0).*tempWD1;
WA2map = WA2map + a*(WA2map==0).*tempWA2;
WH2map = WH2map + a*(WH2map==0).*tempWH2;
WV2map = WV2map + a*(WV2map==0).*tempWV2;
WD2map = WD2map + a*(WD2map==0).*tempWD2;
WA3map = WA3map + a*(WA3map==0).*tempWA3;
WH3map = WH3map + a*(WH3map==0).*tempWH3;
WV3map = WV3map + a*(WV3map==0).*tempWV3;
WD3map = WD3map + a*(WD3map==0).*tempWD3);
end

# ***** Fuse detail coefficients *****

# Initialize fusion variables
fuseWH1 = zeros(mapsize);
fuseWV1 = zeros(mapsize);
fuseWD1 = zeros(mapsize);
fuseWH2 = zeros(mapsize);
fuseWV2 = zeros(mapsize);
fuseWD2 = zeros(mapsize);
fuseWH3 = zeros(mapsize);
fuseWV3 = zeros(mapsize);
fuseWD3 = zeros(mapsize);

# Fusion based on region map: Take MAX coefficient values
for k = 1:n
    fuseWH1(find(WH1map==k)) = WH1(k,find(WH1map==k));
    fuseWV1(find(WV1map==k)) = WV1(k,find(WV1map==k));
    fuseWD1(find(WD1map==k)) = WD1(k,find(WD1map==k));
    fuseWH2(find(WH2map==k)) = WH2(k,find(WH2map==k));
    fuseWV2(find(WV2map==k)) = WV2(k,find(WV2map==k));
    fuseWD2(find(WD2map==k)) = WD2(k,find(WD2map==k));
    fuseWH3(find(WH3map==k)) = WH3(k,find(WH3map==k));
    fuseWV3(find(WV3map==k)) = WV3(k,find(WV3map==k));
    fuseWD3(find(WD3map==k)) = WD3(k,find(WD3map==k));
end

# ***** Fuse approximation coefficients *****

# Take AVERAGE coefficient values
fuseWA1 = mean(reshape(WA1,n,prod(mapsize)),1);
fuseWA1 = reshape(fuseWA1,mapsize);
fuseWA2 = mean(reshape(WA2,n,prod(mapsize)),1);
fuseWA2 = reshape(fuseWA2,mapsize);
fuseWA3 = mean(reshape(WA3,n,prod(mapsize)),1);
fuseWA3 = reshape(fuseWA3,mapsize);

# ***** Reconstruct *****

# Place all levels together
swh(:,1) = fuseWA1;
swh(:,2) = fuseWA2;
swh(:,3) = fuseWA3;
swh(:,1) = fuseWH1;
swh(:,2) = fuseWH2;
swh(:,3) = fuseWH3;
A.2.3 Level-Weighted Wavelet Fusion

Lastly, the code for the LWWF is shown. The main focus of this algorithm is the creation of the soft decision map. Both methods of reconstruction are given. Notice that the only difference between method 1 and method 2 is how the approximation coefficients are fused.

A.2.3.1 LWWF Method 1

In method 1 of the LWWF, the approximation coefficients are fused, via weighted average:

```matlab
swv(:, :, 1) = fuseWV1;
swv(:, :, 2) = fuseWV2;
swv(:, :, 3) = fuseWV3;
swd(:, :, 1) = fuseWD1;
swd(:, :, 2) = fuseWD2;
swd(:, :, 3) = fuseWD3;

% Reconstruction: Inverse SWT
Fmax = mri_iswt2(swa, swh, swv, swd, filt);

% Remove negative artifacts
Fmax(Fmax<0) = 0;
```

```matlab
% ***** Perform 3-level SWT and store the coefficients *****
for k = 1:n
    % Perform SWT
    [swa, swh, swv, swd] = mri.swt2(x(:, :, k), lev, filt);

    % Optional: Perform wavelet denoising, via soft thresholding
    if(flag_waveshrink)
        % Denoise up to level thrlvl
        for kk = 1:thrlvl
            % Get median values
            medh = median(abs(reshape(swh(:, :, kk), 1, [])));
            medv = median(abs(reshape(swv(:, :, kk), 1, [])));
            medd = median(abs(reshape(swd(:, :, kk), 1, [])));

            % Find level-dependent global threshold
            wthrh = sqrt(2*log(prod(size(swh(:, :, kk)))))*medh/tvec(v);
            wthrv = sqrt(2*log(prod(size(swv(:, :, kk)))))*medv/tvec(v);
```
\[ wthrd = \sqrt{2 \log(\text{prod(size}(s\text{wd}(i,i,kk))))} \times \text{medd/tevec}(v); \]

* Denoise
\[
\text{tmp} = (\text{abs}(\text{swh}(i,i,kk) - wthrh); \text{tmp} = (\text{tmp}+\text{abs}(/text{tmp}))/2; \\
\text{swh}(i,i,kk) = \text{sign}(\text{swh}(i,i,kk)) \times \text{tmp}; \\
\text{tmp} = (\text{abs}(\text{swv}(i,i,kk) - wthrv); \text{tmp} = (\text{tmp}+\text{abs}(/text{tmp}))/2; \\
\text{swv}(i,i,kk) = \text{sign}(\text{swv}(i,i,kk)) \times \text{tmp}; \\
\text{tmp} = (\text{abs}(\text{swd}(i,i,kk) - wthrd); \text{tmp} = (\text{tmp}+\text{abs}(/text{tmp}))/2; \\
\text{swd}(i,i,kk) = \text{sign}(\text{swd}(i,i,kk)) \times \text{tmp}; \\
\]

end

* Store LEVEL 1 coefficients
\[
\text{WA1}(k,i,:) = \text{swa}(i,i,1); \\
\text{WH1}(k,i,:) = \text{swh}(i,i,1); \\
\text{WV1}(k,i,:) = \text{swv}(i,i,1); \\
\text{WD1}(k,i,:) = \text{swd}(i,i,1); \\
\]

* Store LEVEL 2 coefficients
\[
\text{WA2}(k,i,:) = \text{swa}(i,i,2); \\
\text{WH2}(k,i,:) = \text{swh}(i,i,2); \\
\text{WV2}(k,i,:) = \text{swv}(i,i,2); \\
\text{WD2}(k,i,:) = \text{swd}(i,i,2); \\
\]

* Store LEVEL 3 coefficients
\[
\text{WA3}(k,i,:) = \text{swa}(i,i,3); \\
\text{WH3}(k,i,:) = \text{swh}(i,i,3); \\
\text{WV3}(k,i,:) = \text{swv}(i,i,3); \\
\text{WD3}(k,i,:) = \text{swd}(i,i,3); \\
\]

* ***** Create soft decision map (weight map) *****
\[
\text{mapsize} = \text{size}(x(:,1)); \]

* Initialize decision maps
\[
\text{WAm} = \text{zeros}(); \text{mapsize}(); \text{n}); \\
\text{WHm} = \text{zeros}(); \text{mapsize}(); \text{n}); \\
\text{WVm} = \text{zeros}(); \text{mapsize}(); \text{n}); \\
\text{WDM} = \text{zeros}(); \text{mapsize}(); \text{n}); \\
\]

* Calculate weights
\[
\text{for a = 1:n} \; \% \text{Process each coil} \\
\text{for k = 1:level} \; \% \text{Sum up each level} \\
\text{\% Get wavelet coefficients} \\
\text{eval([‘tempA = squeeze(WA’ num2str(k) ’(a,i,:))]);’]); \\
\text{eval([‘tempH = squeeze(WH’ num2str(k) ’(a,i,:))]);’]); \\
\text{eval([‘tempV = squeeze(WV’ num2str(k) ’(a,i,:))]);’]); \\
\text{eval([‘tempD = squeeze(WD’ num2str(k) ’(a,i,:))]);’]); \\
\text{\% Take absolute values} \\
\text{tempA = abs(tempA);} \\
\text{tempH = abs(tempH);} \\
\text{tempV = abs(tempV);} \\
\text{tempD = abs(tempD);} \\
\text{\% Sum maps} \\
\text{sumA = sum(tempA(:));} \\
\text{sumH = sum(tempH(:));} \\
\text{sumV = sum(tempV(:));} \\
\text{sumD = sum(tempD(:));} \\
\]
Remove zero values

\[
\text{sum}(\text{sumA} == 0) = 1; \\
\text{sum}(\text{sumH} == 0) = 1; \\
\text{sum}(\text{sumV} == 0) = 1; \\
\text{sum}(\text{sumD} == 0) = 1;
\]

Normalize by level

\[
\text{tempA} = \text{tempA}/\text{sumA}/\text{lev}; \\
\text{tempH} = \text{tempH}/\text{sumH}/\text{lev}; \\
\text{tempV} = \text{tempV}/\text{sumV}/\text{lev}; \\
\text{tempD} = \text{tempD}/\text{sumD}/\text{lev};
\]

Sum of levels

\[
\text{WAmap}(i,j,a) = \text{squeeze}(\text{WAmap}(i,j,a)) + \text{tempA}; \\
\text{WHmap}(i,j,a) = \text{squeeze}(\text{WHmap}(i,j,a)) + \text{tempH}; \\
\text{WVmap}(i,j,a) = \text{squeeze}(\text{WVmap}(i,j,a)) + \text{tempV}; \\
\text{WDmap}(i,j,a) = \text{squeeze}(\text{WDmap}(i,j,a)) + \text{tempD};
\]

Sum along source images

\[
\text{sumWAmap} = \text{sum}(\text{WAmap},3); \\
\text{sumWHmap} = \text{sum}(\text{WHmap},3); \\
\text{sumWVmap} = \text{sum}(\text{WVmap},3); \\
\text{sumWDmap} = \text{sum}(\text{WDmap},3);
\]

Renormalize weights

\[
\text{for a = 1:n} \\
\text{WAmap}(i,j,a) = \text{squeeze}(\text{WAmap}(i,j,a)) \div \text{sumWAmap;}
\]

***** Fuse coefficients *****

Initialize fusion variables

\[
\text{fuseWA1} = \text{zeros(mapsize)}; \\
\text{fuseWA2} = \text{zeros(mapsize)}; \\
\text{fuseWA3} = \text{zeros(mapsize)};
\]

\[
\text{fuseWH1} = \text{zeros(mapsize)}; \\
\text{fuseWV1} = \text{zeros(mapsize)}; \\
\text{fuseWD1} = \text{zeros(mapsize)}; \\
\text{fuseWH2} = \text{zeros(mapsize)}; \\
\text{fuseWV2} = \text{zeros(mapsize)}; \\
\text{fuseWD2} = \text{zeros(mapsize)}; \\
\text{fuseWH3} = \text{zeros(mapsize)}; \\
\text{fuseWV3} = \text{zeros(mapsize)}; \\
\text{fuseWD3} = \text{zeros(mapsize)};
\]

Fuse all coefficients based on weighted average

\[
\text{for a = 1:n} \\
\text{fuseWA1} = \text{fuseWA1} + \text{squeeze}(\text{WAmap}(i,j,a)) \times \text{squeeze}(\text{WA1}(a,i,i)) ; \\
\text{fuseWA2} = \text{fuseWA2} + \text{squeeze}(\text{WAmap}(i,j,a)) \times \text{squeeze}(\text{WA2}(a,i,i)) ; \\
\text{fuseWA3} = \text{fuseWA3} + \text{squeeze}(\text{WAmap}(i,j,a)) \times \text{squeeze}(\text{WA3}(a,i,i)) ; \\
\text{fuseWH1} = \text{fuseWH1} + \text{squeeze}(\text{WHmap}(i,j,a)) \times \text{squeeze}(\text{WH1}(a,i,i)) ; \\
\text{fuseWV1} = \text{fuseWV1} + \text{squeeze}(\text{WVmap}(i,j,a)) \times \text{squeeze}(\text{WV1}(a,i,i)) ; \\
\text{fuseWD1} = \text{fuseWD1} + \text{squeeze}(\text{WDmap}(i,j,a)) \times \text{squeeze}(\text{WD1}(a,i,i)) ;
\]
A.2.3.2 LWWF Method 2

In contrast to method 1, method 2 averages the approximation coefficients with uniform weighting:

```matlab
% ***** Perform 3-level SWT and store the coefficients *****
for k = 1:n
    % Perform SWT
    [swa,swh,swv,swd] = mri_swt2(x(:,:,k),lev,filt);

    % Optional: Perform wavelet denoising, via soft thresholding
    if (flag_waveshrink)
        % Denoise up to level thrlvl
        for kk = 1:thrlvl
            % Get median values
            medh = median(abs(reshape(swh(:,:,kk),1,[])));
            medv = median(abs(reshape(swv(:,:,kk),1,[])));
            medd = median(abs(reshape(swd(:,:,kk),1,[])));

            % Find level-dependent global threshold
            wthr = sqrt(2*log(prod(size(swh(:,:,kk)))))*medh/tvec(v);
```

85
wthrv = sqrt(2*log(prod(size(swv(:, :, kk)))))*medv/tvec(v);
wthrd = sqrt(2*log(prod(size(swd(:, :, kk)))))*medd/tvec(v);

\% Denoise
tmp = (abs(swh(:, :, kk)) - wthr); tmp = (tmp + abs(tmp))/2;
swv(:, :, kk) = sign(swh(:, :, kk)) * tmp;
tmp = (abs(slwv(:, :, kk)) - wthrv); tmp = (tmp + abs(tmp))/2;
swd(:, :, kk) = sign(swd(:, :, kk)) * tmp;
tmp = (abs(sldw(:, :, kk)) - wthrd); tmp = (tmp + abs(tmp))/2;

\% Store LEVEL 1 coefficients
WA1(k, :, :) = swa(:, :, 1);
WH1(k, :, :) = swh(:, :, 1);
WV1(k, :, :) = swv(:, :, 1);
WD1(k, :, :) = swd(:, :, 1);

\% Store LEVEL 2 coefficients
WA2(k, :, :) = swa(:, :, 2);
WH2(k, :, :) = swh(:, :, 2);
WV2(k, :, :) = swv(:, :, 2);
WD2(k, :, :) = swd(:, :, 2);

\% Store LEVEL 3 coefficients
WA3(k, :, :) = swa(:, :, 3);
WH3(k, :, :) = swh(:, :, 3);
WV3(k, :, :) = swv(:, :, 3);
WD3(k, :, :) = swd(:, :, 3);
end

\% ***** Create soft decision map (weight map) *****
mapsize = size(x(:, :, 1));

\% Initialize decision maps
WAmmap = zeros(mapsize(1), mapsize(2), n);
WHmmmap = zeros(mapsize(1), mapsize(2), n);
WVmmmmap = zeros(mapsize(1), mapsize(2), n);
WDmmmap = zeros(mapsize(1), mapsize(2), n);

\% Calculate weights
for a = 1:n \% Process each coil
  for k = 1:lev \% Sum up each level
    \% Get wavelet coefficients
    eval(['tempA = squeeze(WA' num2str(k) ' (a, :, :))']);
    eval(['tempH = squeeze(WH' num2str(k) ' (a, :, :))']);
    eval(['tempV = squeeze(WV' num2str(k) ' (a, :, :))']);
    eval(['tempD = squeeze(WD' num2str(k) ' (a, :, :))']);

    \% Take absolute values
    tempA = abs(tempA);
    tempH = abs(tempH);
    tempV = abs(tempV);
    tempD = abs(tempD);

    \% Sum maps
    sumA = sum(tempA(:));
    sumH = sum(tempH(:));
    sumV = sum(tempV(:));
    sumD = sum(tempD(:));
  end
end
% Remove zero values
sumA(sumA==0) = 1;
sumH(sumH==0) = 1;
sumV(sumV==0) = 1;
sumD(sumD==0) = 1;

% Normalize by level
tempA = tempA/sumA/lev;
tempH = tempH/sumH/lev;
tempV = tempV/sumV/lev;
tempD = tempD/sumD/lev;

% Sum of levels
WAmap(:,:,a) = squeeze(WAmap(:,:,a)) + tempA;
WHmap(:,:,a) = squeeze(WHmap(:,:,a)) + tempH;
WVmap(:,:,a) = squeeze(WVmap(:,:,a)) + tempV;
WDmap(:,:,a) = squeeze(WDmap(:,:,a)) + tempD;

end

% Sum along source images
sumWAmap = sum(WAmap,3);
sumWHmap = sum(WHmap,3);
sumWVmap = sum(WVmap,3);
sumWDmap = sum(WDmap,3);

% Renormalize weights
for a = 1:n
    WAmap(:,:,a) = squeeze(WAmap(:,:,a)) ./ sumWAmap;
    WHmap(:,:,a) = squeeze(WHmap(:,:,a)) ./ sumWHmap;
    WVmap(:,:,a) = squeeze(WVmap(:,:,a)) ./ sumWVmap;
    WDmap(:,:,a) = squeeze(WDmap(:,:,a)) ./ sumWDmap;
end

% ***** Fuse coefficients *****

% Initialize fusion variables
fuseWA1 = zeros(mapsize);
fuseWA2 = zeros(mapsize);
fuseWA3 = zeros(mapsize);
fuseWH1 = zeros(mapsize);
fuseWH2 = zeros(mapsize);
fuseWH3 = zeros(mapsize);
fuseWD1 = zeros(mapsize);
fuseWD2 = zeros(mapsize);
fuseWD3 = zeros(mapsize);

% Fusion detail coefficients based on weighted average
for a = 1:n
    fuseWH1 = fuseWH1 + squeeze(WHmap(:,:,a)) .* squeeze(WH1(:,a,:));
    fuseWH2 = fuseWH2 + squeeze(WHmap(:,:,a)) .* squeeze(WH2(:,a,:));
    fuseWH3 = fuseWH3 + squeeze(WHmap(:,:,a)) .* squeeze(WH3(:,a,:));
    fuseWD1 = fuseWD1 + squeeze(WDmap(:,:,a)) .* squeeze(WD1(:,a,:));
    fuseWD2 = fuseWD2 + squeeze(WDmap(:,:,a)) .* squeeze(WD2(:,a,:));
    fuseWD3 = fuseWD3 + squeeze(WDmap(:,:,a)) .* squeeze(WD3(:,a,:));
end
```matlab
fuseWH3 = fuseWH3 + squeeze(WHmap(:,:,a)) .* squeeze(WH3(:,:,a));
fuseWV3 = fuseWV3 + squeeze(WVmap(:,:,a)) .* squeeze(WV3(:,:,a));
fuseWD3 = fuseWD3 + squeeze(WDmap(:,:,a)) .* squeeze(WD3(:,:,a));
end

%%% Fuse approximation coefficients via averaging
fuseWA1 = mean(reshape(WA1,n,prod(mapsize)),1);
fuseWA1 = reshape(fuseWA1,mapsize);
fuseWA2 = mean(reshape(WA2,n,prod(mapsize)),1);
fuseWA2 = reshape(fuseWA2,mapsize);
fuseWA3 = mean(reshape(WA3,n,prod(mapsize)),1);
fuseWA3 = reshape(fuseWA3,mapsize);

%%% ***** Reconstruction *****
%%% Piece all levels together
swa(:,:,1) = fuseWA1;
swa(:,:,2) = fuseWA2;
swa(:,:,3) = fuseWA3;
swh(:,:,1) = fuseWH1;
swh(:,:,2) = fuseWH2;
swh(:,:,3) = fuseWH3;
swv(:,:,1) = fuseWV1;
swv(:,:,2) = fuseWV2;
swv(:,:,3) = fuseWV3;
swd(:,:,1) = fuseWD1;
swd(:,:,2) = fuseWD2;
swd(:,:,3) = fuseWD3;

%%% Reconstruction: Inverse SWT
Fswt2 = mri_iswt2(swa,swh,swv,Bwd,filt);

%%% Remove negative artifacts
Fswt2(Fswt2<0) = 0;
```

### A.3. Performance Evaluation

This section lists the codes for the various performance measurements used in the experiments. This includes the codes for the power sum, mutual information, and SNR measurement.
A.3.1  Power Sums

In Section 4.4, the frequency power spectra of the fused images were summed for various masks. This section provides the codes to create and sum over the utilized masks. The codes follow the following notations:

- Fad: Fused image using average fusion.
- Fmax: Fused image using max fusion.
- Fw1: Fused image using LWWF method 1.
- Fw2: Fused image using LWWF method 2.

As a preliminary step, the fused images are transformed using the fast Fourier transform:

\[
\begin{align*}
\text{Gad} &= \text{fftsift} (\text{fft2}(\text{Fad})) ; \\
\text{Gmax} &= \text{fftsift} (\text{fft2}(\text{Fmax})) ; \\
\text{Gw1} &= \text{fftsift} (\text{fft2}(\text{Fw1})) ; \\
\text{Gw2} &= \text{fftsift} (\text{fft2}(\text{Fw2})) ;
\end{align*}
\]

With the frequency spectra, the power sum follows.

A.3.1.1  Mask 1

Mask 1 sums the power of signals with frequencies above \( r \). In the following, the custom function \( \text{circ}(n, r) \) creates an \( n \times n \) logical circle mask of radius \( r \).

```
% Sum power of signals with frequencies above r
Bad = sum(abs(Gad(:)));
Emax = sum(abs(Gmax(:)));
Ew1 = sum(abs(Gw1(:)));
Ew2 = sum(abs(Gw2(:)));
for r = 1:75
    temp = logical(1-circ(128,r));
    Bad(r+1) = sum(abs(Gad(temp))) / Bad(1);
    Emax(r+1) = sum(abs(Gmax(temp))) / Emax(1);
    Ew1(r+1) = sum(abs(Gw1(temp))) / Ew1(1);
    Ew2(r+1) = sum(abs(Gw2(temp))) / Ew2(1);
end

Emax(1) = 1; Bad(1) = 1; Ew1(1) = 1; Ew2(1) = 1;

% Plot results
```
A.3.1.2 Mask 2

Mask 2 sums the power of signals within an annular frequency range, where the outer circle is \( r \) and the inner circle is \( r_0 \).

\[
r_0 = 5; \quad \# \text{ Omit values within } r_0 \text{ radius}
\]
\[
r_1 = 10; \quad \# \text{ Initial radius}
\]
\[
\% \text{ Sum power of signal with frequencies above } r
\]
\[
E_{\text{max}} = [];\]
\[
E_{\text{ad}} = [];\]
\[
E_{\text{w1}} = [];\]
\[
E_{\text{w2}} = [];\]
\[
\text{for } r = r_1:75
\]
\[
\text{temp}r = \text{logical} (\text{circ}(128,r) - \text{circ}(128,r_0));
\]
\[
E_{\text{max}}(\text{end}+1) = \text{sum} (\text{abs} (G_{\text{max}}(\text{temp}r))) / \text{tot}G_{\text{max}};
\]
\[
E_{\text{ad}}(\text{end}+1) = \text{sum} (\text{abs} (G_{\text{ad}}(\text{temp}r))) / \text{tot}G_{\text{ad}};
\]
\[
E_{\text{w1}}(\text{end}+1) = \text{sum} (\text{abs} (G_{\text{w1}}(\text{temp}r))) / \text{tot}G_{\text{w1}};
\]
\[
E_{\text{w2}}(\text{end}+1) = \text{sum} (\text{abs} (G_{\text{w2}}(\text{temp}r))) / \text{tot}G_{\text{w2}};
\]
\[
\text{end}
\]
\[
\% \text{ plot results}
\]
\[
\text{figure},
\]
\[
\text{plot} (r_1:75,E_{\text{ad}},'r',r_1:75,E_{\text{max}},'m',r_1:75,E_{\text{w1}},'g',r_1:75,E_{\text{w2}},'b');
\]
\[
\text{title} ('\text{Mask 2}') ;
\]
\[
\text{xlabel} ('\text{Radius of Mask}'); \text{ylabel} ('\text{Summed Energy}');
\]
\[
\text{xlim} ([r_1 75]); \text{ylim} ([0 1.01*\text{max} (\text{max}E_{\text{max}},\text{max}E_{\text{ad}},\text{max}E_{\text{w1}},\text{max}E_{\text{w2}})]);
\]
\[
\text{legend} ('\text{AVG}', '\text{MAX}', '\text{LWWF1}', '\text{LWWF2}');
\]

A.3.1.3 Mask 3

Mask 3 sums the power of signals with frequencies within an annular region with a constant bandwidth.

\[
a_0 = 5; \quad \# \text{ Initial radius of the inner circle}
\]
\[
bw = 10; \quad \# \text{ Width of the annular region (bandwidth)}
\]
\[
\% \text{ Sum power of signal with frequencies above } r
\]
Emax = [];
Ead = [];
Ewl = [];
Ew2 = [];
for r = a0:(75-bw)
  temp = logical(circ(128,r+bw)-circ(128,r));
  Emax(end+1) = sum(abs(Gmax(temp))) / totGmax;
  Ead(end+1) = sum(abs(Gad(temp))) / totGad;
  Ewl(end+1) = sum(abs(Gwl(temp))) / totGwl;
  Ew2(end+1) = sum(abs(Gw2(temp))) / totGw2;
end

% plot results
figure,
plot(a0:(75-bw),Ead,'r',a0:(75-bw),Emax,'m', -a0:(75-bw),Ewl,'g',a0:(75-bw),Ew2,'b');
title('Mask 3');
xlabel('Radius of Mask'); ylabel('Summed Energy');
xlim([a0 (75-bw)]); ylim([0 1.01*max([max(Emax),max(Ead),max(Ewl),max(Ew2)]]));
legend('AVG', 'MAX', 'LWWF1','LWWF2');

A.3.2 Mutual Information

In order to calculate the mutual information, the probability density functions must be estimated. This is done via histograms. The following code is a general function that calculated the mutual information between two random variables. It uses the relationships between entropy and mutual information to provide the desired results.

function I = mri_mi(x,y,n)
    % I = MRI_MI(X,Y,N)
    % Given 2 registered images X and Y of equal size, MRI_MI will
    % calculate the mutual information I between them. To calculate
    % the mutual information, the probability density functions must be
    % estimated. This is done via normalized histograms. Hence, the
    % number of bins N (number of gray levels for quantization) needs
    % to be specified.
    %
    % The formula for the mutual information is:
    %
    % I(X,Y) = H(X) + H(Y) - H(X,Y) ,
    %
    % where H(.) is the entropy of the argument.
    %
    % Written by: Jeong Hwan Bang
    % Date: October 16, 2006

    % CONVERT to double data-format, column vectors
    x=double(x); x = x(:);
A.3.3 Signal-to-Noise Ratio

The SNR was defined in Section 4.6.2. The calculation of the SNR requires a noiseless image, which is approximated by fusing images without synthetic noise. This image is denoted by the variable F. The fused image with synthesized, additive noise is denoted by the variable Fhat. Given these two variables, the following is the code to calculate the SNR. Note that bmask is the logical mask of Figure 36.
REFERENCES