NTNU

Norwegian University of
Science and Technology

Empirical Mode Decomposition for Improved Noise Filtering and Classification of Two-Dimensional Data

Joachim Blaafjell Holwech

Master of Science in Cybernetics and Robotics
Submission date: June 2018
Supervisor: Marta Maria Cabrera Molinas, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

PROBLEM DESCRIPTION

This thesis will examine the use of Empirical Mode Decomposition (EMD) as a basis for filtering noise from two dimensional data. The main goal is to improve the visual quality of noisy images attained from professional and non-professional imaging devices. Additionally, EMD will be used to improve the information quality contained in images with the objective of improving object recognition and classification.

ABSTRACT

This thesis studies the use of Empirical Mode Decomposition (EMD) applied to two dimensional data and the application of this method to filter noise from images. Mode Mixing Separation (MMS) is presented and implemented as a technique for reducing mode mixing, which is a common problem that occurs with the usage of EMD. A new filter is also introduced, which combines theory from both adaptive filters and EMD in an attempt to improve the noise filtering even further. In total, three types of EMD based filters are presented in this thesis:

- A filter based on EMD.
- A filter based on EMD with MMS.
- An adaptive filter based on EMD with MMS.

The filters are tested on three specific use cases where noise filtering could be valuable. The first test is based on a visual assessment, where the filtered images are compared to the filtered images of conventional filters. This gives a better general understanding of how the filters perform, and also indicates of how they measure up against established filtering methods. The second test attempts to filter noisy satellite images of boats and icebergs, with the purpose of improving the classification accuracy of the two categories in the dataset. By using a neural network for classification, the accuracy values can be used as an objective measurement of how well the filters work. The final test looks at the use of EMD for filtering medical ultrasound images to improve the image quality. The purpose of this test is to see how the filters work on a real-world problem.

The results for the first test, showed that the EMD based filters perform significantly worse than the conventional filters. The EMD filters added artifacts that left the filtered images in a worse condition than the input. The classification of the filtered satellite images resulted in an accuracy of 52.87%. When combining the filtered set with the unfiltered set, an accuracy of 89.28% was achieved, which is a 0.75 percentage point improvement over training only on the unfiltered set. Applying the EMD based filter on medical ultrasound images had a minimal effect, but left the quality of the images in a slightly worse condition.

SAMMENDRAG

Denne oppgaven undersøker bruk av Empirical Mode Decomposition (EMD) i to dimensjoner og tar for seg hvordan denne methoden kan brukes til å fjerne støy fra bilder. En implementering av Mode Mixing Separation (MMS) presenteres som en løsning på mode miksing-problemet, et fenomen som vanligvis oppstår ved bruk av EMD. I et forsøk på å forbedre støyfjerningen ytterligere, introduseres et nytt filter som kombinerer teori fra både adaptive filtre og EMD. Totalt presenteres tre typer filtre i denne oppgaven som alle er basert på EMD:

- Et filter basert på EMD.
- Et filter basert på EMD med MMS.
- Et adaptivt filter basert på EMD med MMS.

Filtrene er testet på tre forskjellige bruksområder hvor støyfjerning kan være verdifullt. Den første testen er basert på en visuell evaluering hvor de filtrerte bildene sammenlignes med filtrerte bilder fra konvensjonelle filtre. Denne testen gir en bedre forståelse av hvordan filtrene fungerer og er en god indikator på hvordan de presterer i forhold til mer etablerte metoder. Den andre testen gjør et forsøk på å fjerne støy fra satelittbilder av isfjell og båter med den hensikten av å forbedre kvaliteten på bildene. Ved å bruke et nevralt nettverk til å klassifisere med, kan nøyaktighetsverdiene brukes som et mål for en objektiv evaluering av hvor bra filtrene fungerer. Den siste testen ser på bruk av EMD til filtrering av medisinske ultralydbilder hvor målet er å forbedre bildekvaliteten. Hensikten med denne testen er å undersøke hvordan filtrene fungerer på reelle bruksområder.

For den første testen, viser resultatene at de EMD-baserte filtrene presterer betydelig dårligere enn de konvensjonelle filtrene. EMD-filtrene introduserer artefakter som fører til at de filtrerte bildene er i en dårligere tilstand enn før de ble filtrert. Klassifiseringen av de filtrerte satelittbildene resulterte i en klassifiseringsnøyaktighet på 52.87%. Ved å kombinere det filtrerte datasettet med det ufiltrerte settet ble det oppnådd en nøyaktighet på 89.28%. Dette er 0.75 prosentpoeng høyere enn hvis man bare trente på det ufiltrerte settet. Anvendelse av det EMD-baserte filteret på medisinske ultralydbilder hadde en minimal effekt, men reduserte kvaliteten på bildene noe.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Professor Marta Molinas at the Department of Engineering Cybernetics at NTNU. She has been a helpful and responsive advisor during my work and development of my Master's thesis. She introduced me to the interesting field of adaptive data analysis, that has been the basis for all my work this last year here at NTNU. A big thanks to Maximiliano Bueno Lopez for giving me feedback and inspiration for new and interesting ways to solve the challenges presented in this thesis. I would also like to give my acknowledgements to Henriette Fanebust, MD, at Vestfold Hospital Trust for helping me record the ultrasound images used in this thesis, and thanks to Simen Trollvik for letting himself be recorded on.

There are so many people this last year who have had a direct or indirect impact on this thesis, that they cannot all be named and thanked by name. Thanks to my family, friends, the people at the office and everyone else for the help and support. It has been five challenging, inspiring and fun years here at NTNU that I will always remember.

ACRONYMS

AI Artificial Intelligence.

BEMD Bidimensional Empirical Mode Decomposition.

CNN Convolutional Neural Network.

DEMD Directional Empirical Mode Decomposition.

EMD Empirical Mode Decomposition.

HH Horizontal/Horizontal.

HV Horizontal/Vertical.

IEMD Image Empirical Mode Decomposition.

IMF Instantaneous Mode Function.

MMS Mode Mixing Separation.

PDF Probabilistic Density Function.

RBF Radial Basis Function.

SNR Signal-to-noise ratio.

CONTENTS

PREFACE

I first learned about Empirical Mode Decomposition (EMD) in autumn 2017 through my supervisor Professor Marta Molinas. At the time I was writing my project thesis, where I was working on analyzing EEG-signals using the methods that I learned from Marta's course in adaptive data analysis. During the autumn semester of 2017, I joined the Statoil/C-CORE Iceberg Classifier Challenge, and it brought me on to the idea of applying the new methods I learned on the iceberg dataset. Based on my experience working with EMD in one dimension, I knew that the noise filtering capabilities of this method was very powerful. Getting to test the use of EMD on images, both in the competition and also in Marta's course, gave me the confidence to take it a step further and research the topic of noise filtering in two dimensions more thoroughly through my Master's thesis. I believe this thesis is one of the most comprehensive reviews available on the use of EMD for image noise filtering and I truly hope someone will find the theory presented in this thesis useful.

All code used to produce the results presented in this thesis, are implemented solely by the author. The neural network implementation as the only exception, is based on a submission sent to the Statoil/C-CORE Iceberg Classifier Challenge [1]. The algorithms are written in the Python programming language and run on a Dell desktop machine supplied by the university. Some code libraries and functions are used to perform tasks that are outside the scope of the thesis, namely:

ALGLIB - Performing 2D RBF interpolations.

imextendedmax / imextendedmin - MatLab functions for finding extrema points.

Numpy / Scipy - Python scientific libraries for performing mathematical operations.

Matplotlib - Python library used for plotting and visualization of the results.

Tensorflow / Keras - Library for modelling and training of neural networks.

The full source code can be found at `github.com/holwech/image_emd` under the MIT license. The two MatLab based functions `imextendedmax / imextendedmin` are run using the "MatLab Engine API for Python". All tests and visualizations were executed using Jupyter running on the Windows 10 operating system. All external libraries and functions were found and implemented into the code base solely by the author.

The Statoil/C-CORE Iceberg Classifier Challenge dataset was provided under a open research

license and distributed through the Kaggle competition page. The medical ultrasound images were recorded at Vestfold Hospital Trust using the GE Logiq P9.

Professor Marta Molinas at the Department of Engineering Cybernetics at NTNU held the role as supervisor through the duration of the thesis work. She provided advice during the thesis work and gave suggestions to articles and papers that could be of relevance. Associate Professor Maxiliano Bueno Lopez held a minor advisory role and helped discuss alternative approaches to the problem at hand. Henriette Fanebust, MD, provided support with the recording of the medical ultrasound images.

Section 2.2 through 2.2.1 is reused from the author's project thesis from autumn 2017.

This thesis looks at the use of EMD for filtering noise from two dimensional data, with the intent to improve quality and classification accuracy of noisy images.

CHAPTER 1

INTRODUCTION

ALL photos captured with a digital camera or a smartphone contain some degree of noise. The quality of the cameras and their ability to remove said noise, have become so good that humans do not really notice the presence of noise anymore. This can mostly be attributed to the fact that companies like Google, Apple and Canon for the last decade have put significant amount of resources into creating better and more robust filtering methods that remove the noise. Today, these small hand held devices apply advance state-of-the-art noise reduction algorithms, to give us noise-free images that are crisp and clean. While digital photography is where the effect of the noise reduction algorithms are most visible, they also affect our everyday lives in other ways. Tools that are in widespread use like radios, 4G networks, medical ultrasound images and radar systems heavily rely on being able to efficiently remove noise. In general, one can assume that if there is some form of analog transmission involved, there is also noise present, and therefore also some form of noise reduction method used to improve the quality of the received signals. Furthermore, efficient noise reduction can also be useful in Artificial Intelligence (AI) applications like object recognition and classification. Performing accurate classification of objects in photos, is currently a very popular research topic in the AI community. The accuracy of the predictions rely on the quality of the image features and efficient noise filtering can aid in highlighting them. This is essential in helping the classifier learn the features that matter and making it converge faster towards a solution. While this field in the last decade has taken huge leaps in solving the noise problem, there is still much to be discovered.

This thesis will present some new methods from removing noise in 2D data that are based on Empirical Mode Decomposition (EMD) [2] and Mode Mixing Separation (MMS) [3]. Applying EMD in 2D is not unique and there have been attempts at using it both for classification and noise reduction [4] [5], [6], [7]. On the other hand, using Mode Mixing Separation to improve filtering of image noise has not been thoroughly tested and is an unexplored area where there could be something to gain. This thesis will examine the use of these two techniques and assess how well they remove noise in 2D data. We will formulate two research questions, one that is of especial interest for the AI research community and one that looks more generally at how EMD and MMS can improve image quality:

> **RQ1: Visual Enhancement** - Can EMD and MMS be used on medium to large photos to filter noise to a degree where the visual quality of the image has been significantly improved?

RQ2: Classification - Can EMD and MMS be used to improve the feature extraction and classification of noisy images?

It can be argued that **RQ1** and **RQ2**, essentially are the same. Improving the image quality would probably improve the classification, and similarly, improving the classification would probably mean that the image quality would improve, right? In most situations, that is probably the case. But while we for **RQ1** mostly care about the visual quality, in **RQ2** we want to make the useful information more visible and highlight important features. This does not always mean that the images are visually appealing and correct in terms of actual visual representation.

The filter methods will be assessed in such a way that the results can answer the questions for us. For **RQ1**, these is a two step approach. First, a more comprehensive test based on among other things, the filter performance, memory usage and processing time is done, which will give an overall impression of how useful and viable the filters are for consumer devices like smartphones and dedicated cameras. The second step looks at how the filters perform in medical imagery, and more specifically how well it is able to remove noise from ultrasound image. **RQ2** will be answered by applying the developed filters on a real dataset of satellite images of icebergs and boats, where a classifier will try to distinguish the classes based on the unfiltered and unfiltered images. The accuracy will then be used as a measure to assess how well the filters are able to highlight the important information in the images.

CHAPTER 2

BACKGROUND THEORY AND LITERATURE

THIS chapter contains a literature study and presentation of relevant theory needed to answer the research questions presented in the introduction. Conventional image filtering techniques will also be presented as they will be used as a platform of comparison. The literature study of relevant papers on 2D EMD describes the currently existing methods. Additionally, a description of MMS is provided, including how this applies to images.

2.1 Image Noise and Filters

Noise is defined as an unwanted disturbance or error and is often randomly, or at least semi-randomly, distributed. There exists multiple sources to the noise that normally is visible in images. With regards to cameras, the source of noise will often be the electrical components of the camera itself. But noise can also come from physical phenomenons, like air pollution or uneven surfaces that create a random backscatter which the sensors will pick up. While physical noise does not necessarily fit under the official definition of ”noise”, it can nevertheless be an unwanted disturbance one would want to remove. Noise can appear in multiple different forms, and so this chapter will limit itself to a few noise types that can appear in modern digital imaging technology or that have some other relevant properties that are useful for assessing the filters.

2.1.1 Types of noise

Noise is most commonly introduced to some data by simply adding it to the measured data. This category of noise is normally called additive noise and can be modelled by the following equation:

$$g(x, y) = f(x, y) + n(x, y) \tag{2.1}$$

where $g(x, y)$ is the image with noise, $f(x, y)$ is the image without noise and $n(x, y)$ is the measurement noise. The distribution and value of the noise n, depends on what type of noise

it represents. In the following sections, 4 noise types will be presented. Some of these types of noise can appear in recorded data, and thus are interesting cases for new filtering techniques. The rest have some interesting properties that are relevant when performing the test.

Impulse (Salt-and-pepper) Noise

Impulse noise will appear in an image as evenly distributed black or white dots. The Probabilistic Density Function (PDF) is given by:

$$p(z) = \begin{cases} P_a & \text{for } z = a, \\ P_b & \text{for } z = b, \\ 0 & \text{otherwise} \end{cases}$$

where z is the intensity, P_a and P_b are the probabilities of getting intensities a and b respectively [8]. The values a and b specify the brightness of the two noise speckle types, where if $b > a$, then b will be a bright spot and a will be darker. For non-zero probabilities P_a and P_b, the noise appears as black and white speckles, hence the name "salt-and-pepper". Impulse noise is normally assumed to be saturated, which for an 8-bit image means that the noise values are either 0 or 255.

Impulse noise values are normally very large compared to the average brightness of the image and appear as a result of fast transients during recording. These transient disturbances can be caused by faulty switching, and result in saturated speckles in the data.

Uniform Noise

Uniform noise has a uniform probability to appear in the data within a certain intensity range. The PDF of uniform noise can be written as:

$$p(z) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq z \geq b, \\ 0 & \text{otherwise} \end{cases}$$

The mean and variance of the PDF is given by:

$$\overline{z} = \frac{1}{b-a} \quad \text{and} \quad \sigma^2 = \frac{(b-a)^2}{12} \tag{2.2}$$

Uniform noise distribution does not specifically appear in any real-life scenarios, but is an interesting case for applying EMD and MMS on, because of its uniform characteristics.

Gaussian Noise

The Gaussian noise PDF of a random Gaussian variable z, is given by:

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(z-\overline{z})^2}{2\sigma^2}} \tag{2.3}$$

where z is the intensity, $\overline{z}$ is the mean of z and σ is the standard deviation. Gaussian noise is highly prominent in digital imaging technology. The source is often electronic components causing disturbances to neighbouring components because of electromagnetic interference. Thus, Gaussian noise will always appear in measurements performed by devices running on electricity. Gaussian noise can also occur in the image sensor when there is poor lighting conditions or high temperatures.

Speckle Noise

Speckle noise is a multiplicative noise, that is, some uniformly distributed random noise is multiplied with the pixel value of the image. Speckle noise can be modelled by the following equation:

$$f = s + s \cdot n \tag{2.4}$$

where f is the image with noise, s is the input image and n is the uniformly distributed noise.

Speckle noise is commonly observed in active radars, medical ultrasound and other active sensing devices.

2.1.2 Filter Types

There exists a variety of different filters that remove noise. Some filters perform better on some types of noise than others. The filters can be divided into two categories, namely linear and nonlinear, where a linear filter applies a linear operation to the image and a nonlinear filter applies a nonlinear operation. A filter can also be applied spatially or in the frequency domain. A spatial filter works by applying a predefined rectangular window (kernel) to the image. This process of applying a window to a sub-field of an image is called convolution and is used in a wide array of technical fields from computer vision to machine learning. A frequency domain filter on the other hand, applies filters directly to the frequency spectrum of the image using Fourier transform.

Arithmetic Mean Filter

The arithmetic mean filter simply calculates the average mean value of all the pixels in the window and sets the center pixel to this value. Say S_{xy} is the coordinates for the $m \times n$ subimage covered by the window, then the restored center pixel value at a position (x, y) is given by:

$$\hat{f}(x, y) = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} g(s, t) \tag{2.5}$$

where $\hat{f}(x, y)$ is the replaced pixel value in position (x, y), S_{xy} is the area of the window and $g(s, t)$ is the noisy input.

Median Filter

The median filter is an order-statistic filter, which means that the filter has a response based on the ordering of the pixel values in the window. The median filter is the most well known order-statistic filter, and works by taking the median of the pixel values included in the window. This can be presented mathematically as:

$$\hat{f}(x, y) = \underset{(s,t) \in S_{xy}}{\text{median}} \big\{ g(s, t) \big\} \tag{2.6}$$

where $\hat{f}(x, y)$ is the replaced pixel value in position (x, y), S_{xy} is the area of the window and $g(s, t)$ is the noisy input. The median filter is a popular filtering method since it creates little blurring compared to other linear filters. It works especially good on S&P noise, where the extreme noise values mostly appear in the end spectrum of the median distribution and therefore will often be filtered from the data.

Max and Min Filter

The max filter works similarly to the median filter, but instead of taking the median value as the new pixel value, it instead uses the max value. This results in an excellent filter against pepper noise, which will have a low brightness value. The max filter is simply given by:

$$\hat{f}(x, y) = \max_{(s,t) \in S_{xy}} \big\{ g(s, t) \big\} \tag{2.7}$$

where $\hat{f}(x, y)$ is the replaced pixel value in position (x, y), S_{xy} is the area of the window and $g(s, t)$ is the noisy input. Similarly, the min filter will find the dark spots in the image, thus removing salt noise. The min filter is defined by:

$$\hat{f}(x,y) = \min_{(s,t)\in S_{xy}} \{g(s,t)\} \tag{2.8}$$

Midpoint filter

The midpoint filter calculates the midpoint value between the max and the min value in a given window. We write this as:

$$\hat{f}(x,y) = \frac{1}{2}\left[\max_{(s,t)\in S_{xy}} \{g(s,t)\} + \min_{(s,t)\in S_{xy}} \{g(s,t)\}\right] \tag{2.9}$$

Adaptive Filter

While the linear filters presented above work excellent for removing noise, they also have a tendency to add significant blurring to the image. Adaptive filters try to solve this by taking into consideration the pixel values in the window when filtering. Adaptive filters have the potential to perform better than non-adaptive filters, but this comes at the cost of increased complexity. The filter response for any given window position is based on the following properties:

1. Value of a given pixel (x, y) in the image $g(x, y)$.
2. Variance σ_η^2 of the noise.
3. Local mean m_L of the pixels values in a given window S_{xy}.
4. Local variance σ_L^2 of the pixel values in a given window S_{xy}.

From this an adaptive behavior can be defined that takes into regard the available properties. Based on these properties the adaptive filter can be defined by the following:

1. If the variance of the noise σ_η^2 is zero, the value $\hat{f}(x, y)$ should simply be set to $g(x, y)$ since in this case there is zero noise.
2. A large local variance σ_L^2 compared to σ_η^2, indicates that the window area S_{xy} contains an edge or a corner. $\hat{f}(x, y)$ should in this case be set to a value close to $g(x, y)$ so that the details of the image is preserved.
3. If the two variances σ_η^2 and σ_L^2 are similar in value, the properties of the window is similar to the properties of the image. The noise can then be reduced by doing the arithmetic mean of the pixels in the area S_{xy}.

This definition for $\hat{f}(x, y)$ can be written as:

$$\hat{f}(x,y) = g(x,y) - \frac{\sigma_\eta^2}{\sigma_L^2}\left[g(x,y) - m_L\right] \tag{2.10}$$

2.1.3 Convolutional Neural Networks

Classification of images is currently one of the most popular research topics in the AI field. It is a highly complex problem, that requires advance nonlinear algorithms to solve. The algorithm that has risen as the unchallenged leader in image recognition and classification, is the Convolutional Neural Network (CNN). CNNs are supervised learning algorithms that train on a dataset where the labels are known. After a completed training of the network, it can look at new, unknown data and be able to perform the specific tasks that it trained for.

Neural Networks

CNNs are based on neural networks, which again are inspired from the biological processes that occur in the brain. Neural networks are highly nonlinear and belong to a class of algorithms called supervised learning algorithms. The analogy to supervised learning is the concept of the ”student” that tries to answer a question, and the ”teacher” that tells the ”student” whether it was right or wrong. In a similar fashion, the neural network will train on a dataset where the answers are already known. For example for object recognition, the network will try to guess what object is contained in the image. This guess will then be compared to the correct answer, and the network will then correct its own internal parameters to improve future guesses of similar objects. By doing this to a large amount of images, the network will reach a generalized understanding of what the different types of classes in the dataset look like.

The basic building blocks of the neural network is the perceptron, which by itself, is a binary linear classifier [9]. The perceptron is shown in Figure 2.1, with three inputs x_1, x_2, x_3 and a corresponding weight for each input. While there are only three inputs in this example, there is no limit to how many inputs a perceptron can have.

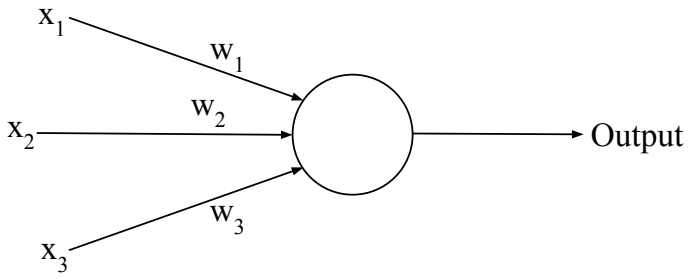

Figure 2.1: Illustration of a single perceptron.

A perceptron can be modelled by the equation:

$$output = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq t, \\ 1 & \text{if } \sum_j w_j x_j > t \end{cases} \tag{2.11}$$

where t is a preset threshold value, x_j is a given input value, and w_j is the corresponding weight. We see from Equation (2.11), that each input x_j gets multiplied by its corresponding weight before they all get summed together. By adjusting the weights, what input values that result in a 1 or a 0 can be finely tuned. If the sum of the input and weight values go above the threshold t, the perceptron will output 1. Similarly, if the output is less than the threshold, the output will be 0.

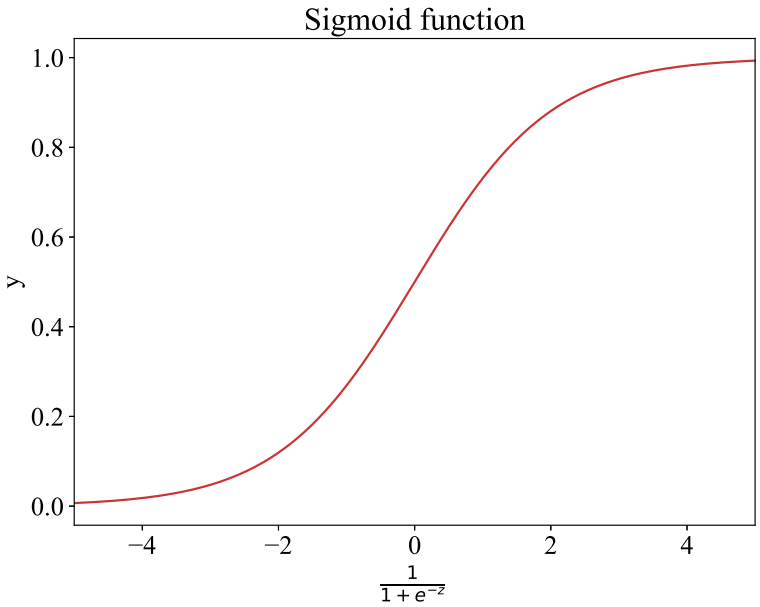

Figure 2.2: The output value of the sigmoid function.

While the perceptron works great as a logical operator, in its current form, when combining multiple of them into a neural network, will not work. This is because a small adjustment to a weight can make the output of the perceptron flip completely from 0 to 1 and vice versa. This makes it hard to do small incremental adjustments to the weights in a neural network, without making multiple parts of the network change in unpredictable ways. The way around this, is to use a activation function instead of a threshold to determine the output. If we also add a bias b for linear shifts, we get the following output equation:

$$output = a\left(\sum_j w_j x_j + b\right) \tag{2.12}$$

where a($\cdot$) is the activation function. A popular activation function is the *sigmoid*-function, which would give us the following output function:

$$output = \frac{1}{1 + exp(-\sum_j w_j x_j - b)} \tag{2.13}$$

The sigmoid neuron activation function is shown in Figure 2.2. We see that it has a smooth output that does not result in a big change in output when there is a small change in input.

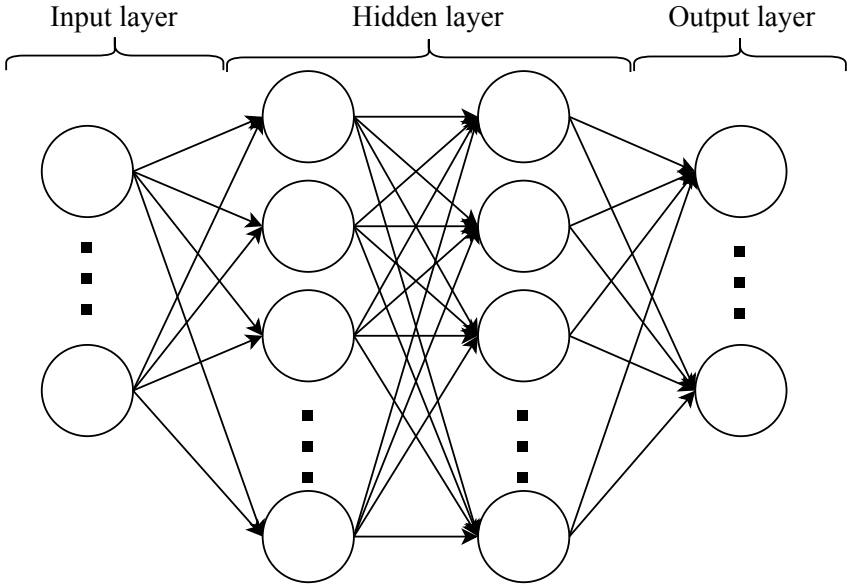

Figure 2.3: A diagram of a neural network.

These neurons can be stacked to create layers, and the layers can be connected to each other to create neural networks. This results in a structure that can learn highly nonlinear patterns in very complex data. Basically, we now have a very advance optimization algorithm that we can use to solve hard problems. Figure 2.3 shows an example of what a neural network can look like. The network is divided into three parts: input layer, hidden layer and output layer. While the input and output layer only have one row of neurons, the hidden layer can have multiple rows of neurons. The number of neurons in each layer, and also the number of the layers, can vary from architecture to architecture. Typically, a network is wider than it is long.

A neural network works by applying some input to the input layer, then in each neuron, some calculations will be performed, before the output is propagated forward to the next layer. This process is called forward propagation and is applied all the way from the input layer to the output layer. The output prediction is then compared to the true label, and an error is calculated based on the difference between the two answers. This error is then used to calculate updated weight parameters for every weight in the network. This is called gradient descent, and the process of moving backwards in the network updating weights is called backpropagation.

Convolutional Neural Networks

A CNN is a neural network that is modified to take images as input and analyze them in an efficient manner. A neural network can become extremely complex with many thousands of

parameters, thus processing large input data like images could potentially make the complexity of the network increase to extreme proportions. Because of this, some steps have to be taken to optimize how the data is handled, so that the necessary parameters can be reduced.

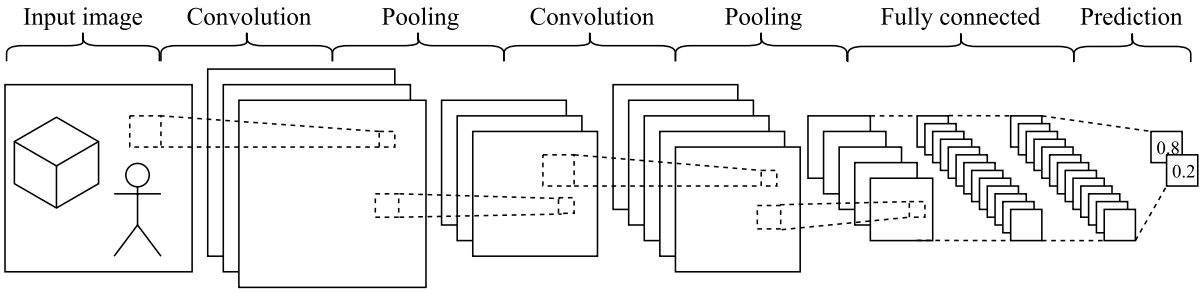

Figure 2.4: Example structure of a CNN.

Figure 2.4 illustrates an example setup of a CNN. We see that there are three main parts in this system: convolution layers, pooling layers and fully connected layers. Each part plays a role in simplifying and generalizing the input data. If we start with the first step, we see that a convolution is performed on the input image. This involves multiplying a square filter across the whole image to create a new feature map. This filter, also called a kernel, can be designed to extract specific features like corners or edges. The output of each convolutional step is sent through the ReLU activation function before being stored as a feature map. The ReLU activation function is shown in Figure 2.5 and is a piece wise linear function that is described by $output = \max(0, input)$. The ReLU neuron has the major advantage that it does not suffer from a phenomenon called vanishing gradients. Vanishing gradients means that the learning rate will slow down and often result in the optimal solution not being found. This is a typical problem for sigmoid neurons, when they reach saturation towards ± 1 the gradient will go towards zero. The feature map has multiple different parameters that are listed underneath:

Depth - Sets the number of feature maps.

Stride - Defines how many pixels the window skips for each convolutional step.

Zero-padding - Defines how many pixels wide the padding around the images should be.

After performing the convolution, a process called pooling is performed. The purpose of pooling is to downsample the feature map while keeping the most important features. Pooling is normally done by selecting a 2×2 window and performing some selection based on the values in this matrix. This could either involve taking the mean of the value, selecting the max value or even taking the total sum. The most widespread method is illustrated in Figure 2.6, where we see that max pooling is performed on a 4×4 matrix reducing it down to just 2×2 in size. Max pooling has many advantages; it makes the data more manageable and smaller, it reduces the number of parameters in the system which also reduces the chance of overfitting. It also makes the network scale invariant and also partially invariant to transformations, distortions and translations.

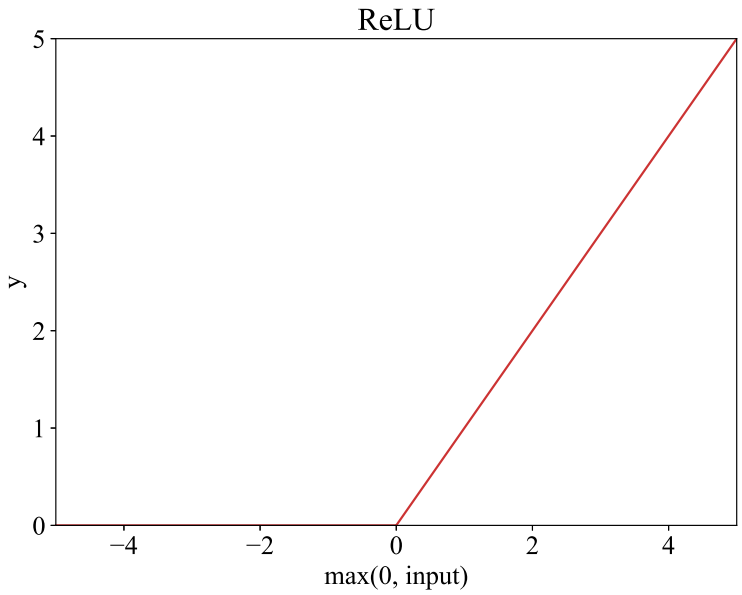

Figure 2.5: The output of the ReLU activation function.

The convolution and pooling step can be repeated multiple times depending the architecture. Following this, is one or more fully connected layers of neurons. The purpose of the fully connected layers are to learn nonlinear combinations of features in the feature maps, thus representing higher level features in the data. The last layer is connected to a fully connected output layer that uses the *softmax* activation function. Softmax ensures that the sum of the outputs are always equal to 1, so that they represent a probability distribution.

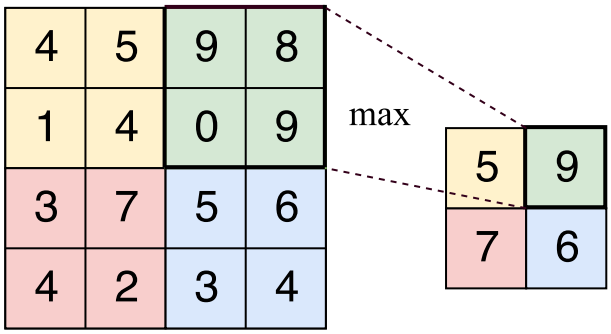

Figure 2.6: Example max pooling step for a 4×4 window.

The process of training a CNN is very similar to that of a neural network, and the training step can be summarized in the following steps:

1. Initialize all weights with small random values.
2. Input a training image and perform forward propagation.

3. Calculate the error between the output and the label.
4. Perform backpropagation based on the error gradients.

2.2 Empirical Mode Decomposition

This section (through Section 2.2.1) is taken from the authors project thesis "Implementation of a Brain-Computer-Interface Using the OpenBCI Ultracortex" [10]

Empirical Mode Decomposition assumes that the data consists of multiple intrinsic modes of oscillations. Each of these intrinsic modes added together make up the original signal. These modes are called Instantaneous Mode Function (IMF). An IMF is similar to a simple harmonic function, in that it can represent a separate part of a more complex signal. An IMF is more general than a harmonic function, as it does not necessarily need to have a constant amplitude or frequency and can vary depending on time. For a mode to be an IMF it has to satisfy two criteria:

1. The number of extrema and zero-crossings have to be equal or differ by at most one.
2. The mean of the envelope defined by the local maxima and the envelope defined by the local minima is equal to zero at any point in the data.

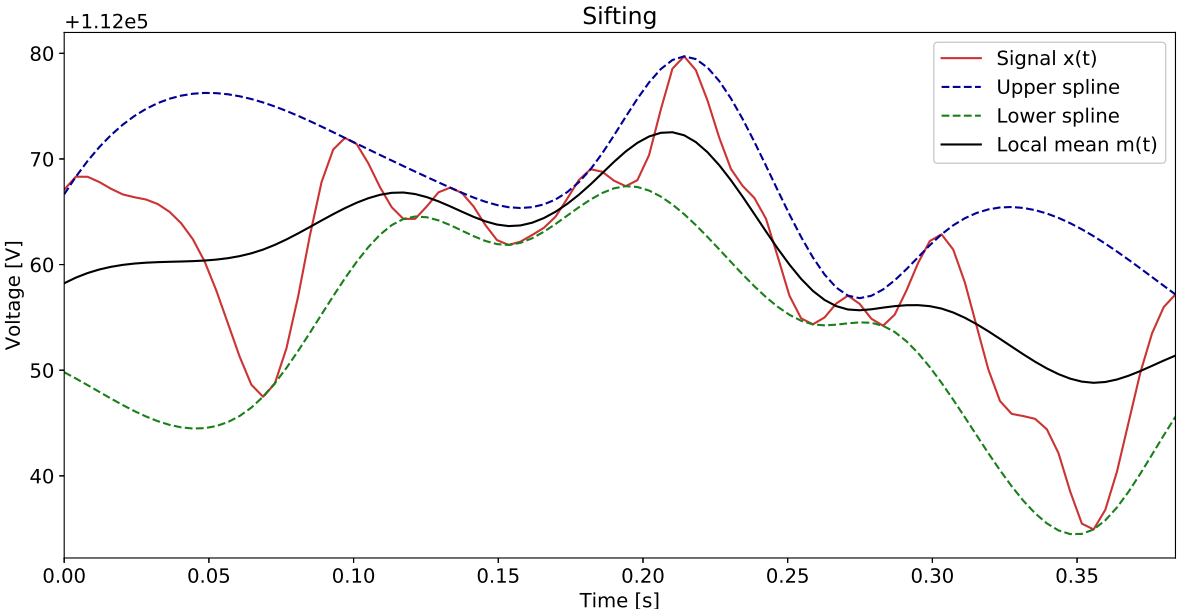

Figure 2.7: Illustration of a single sifting with the envelope around the signal shown as the dotted lines.

With this definition, EMD can be used to decompose any signal into its respective IMFs. The first step of EMD is to locate all the local extrema in the signal. The maxima are then connected

with a cubic spline. Similarly, this is also done with the minima. This creates an upper and lower envelope around the signal that can be used to calculate the mean of the envelope. This mean is then subtracted from the original signal. This represents the first step of the EMD process and is called sifting. Equation (2.14) shows the first step, where $x_{input}(t)$ is the input signal, h_1 is the first mode and m_1 is the envelope mean.

$$h_{1,1} = x_{input}(t) - m_{1,1} \tag{2.14}$$

If the mode does not satisfy the criteria for an IMF, the process is repeated with the extracted mode $h_{1,1}$:

$$h_{1,2} = h_{1,1} - m_{1,2} \tag{2.15}$$

This sifting process is repeated for k steps until the criteria for an IMF is satisfied:

$$h_{1,k} = h_{1,k-1} - m_{1,k} \tag{2.16}$$

This last step is then defined as the first IMF:

$$c_1 = h_{1,k} \tag{2.17}$$

The first IMF is then subtracted from the input $x(t)$, removing one mode from the original signal:

$$x_1(t) = x_{input} - c_1 \tag{2.18}$$

The sifting process is then repeated but now with the new input with one mode removed:

$$h_{2,1} = x_1(t) - m_{2,1} \tag{2.19}$$

This sifting process is continued until the resulting mode $h_{l,k}$ is monotonic, that is, a function that is only non-decreasing or non-increasing. More simply said in this situation; a function without any extrema. This last monotonic mode is normally called the residue or trend, as it describes the trend in the signal. The residue is the original signal with all the IMFs subtracted:

$$r_1 = x(t) - \sum_{n=1}^{K} c_n \tag{2.20}$$

From this we also see that the original signal can be reconstructed by adding all the components:

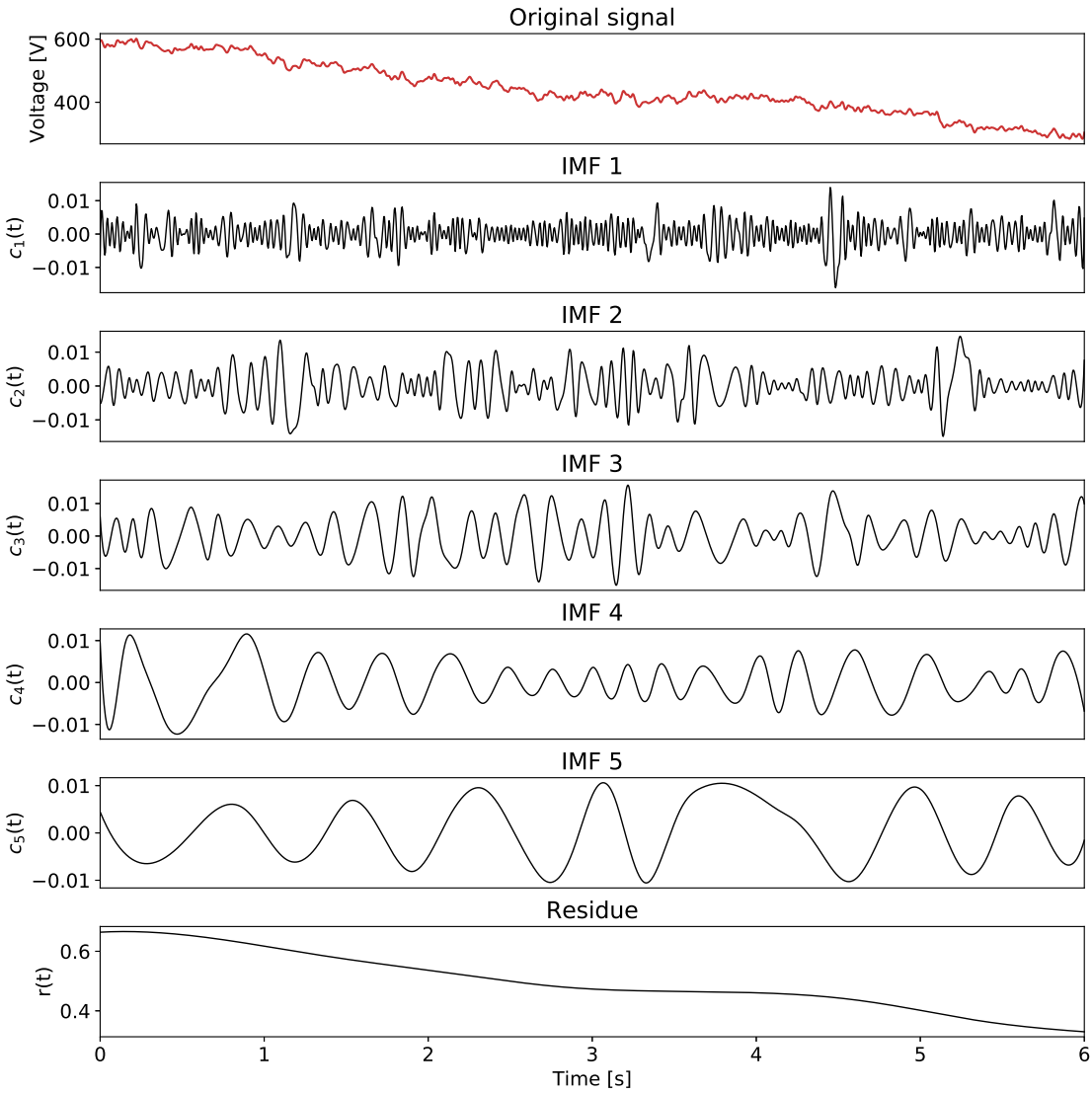

Figure 2.8: A 1D-signal with its corresponding IMFs

$$x(t) = \sum_{n=1}^{K} c_n + r_1$$

The number of IMFs that are extracted from the signal depends on a preset stopping criterion. The original stopping criterion proposed by Huang [2] checks whether two successive sifting operations have a normalized squared difference that is smaller than a give threshold:

$$SD_k = \frac{\sum_{t=0}^{T} |h_{k-1}(t) - h_k(t)|^2}{\sum_{t=0}^{T} h_{k-1}^2} < \epsilon$$

This criterion has two fundamental weaknesses, the first is the task of deciding an appropriate ϵ. Secondly, this criterion does not take into regard the IMFs and whether they have the same number of zero-crossings and extrema. To tackle this issue, an alternative criterion was proposed based on a S-number. This criterion will stop the process if the sifted signal satisfies the criteria for an IMF and there has not been any changes to number of zero-crossings or extrema for S consecutive siftings.

While this theory applies to 1D signals, much of the same theory can be applied to 2D signals like images with a few modifications. In the following sections the methods currently available that apply EMD in 2D, will be presented.

2.2.1 Bidirectional Empirical Mode Decomposition

The paper on Bidimensional Empirical Mode Decomposition (BEMD) was published by J.C. Nunes in 2003 [11] and presented as an efficient way of applying EMD in 2D. BEMD laid the foundation for use of EMD in 2D, as one of the first implementations to handle 2D signals. The implementation is similar to the conventional 1D method and fairly straight forward. Given an input image $x(x, y)$, where x and y are the pixel locations along the x- and y-axis, the sifting process is defined by the following steps:

1. Find all extrema in the image. These are selected through the use of morphological reconstruction based on geodesic operations.
2. Create an upper envelope $e^+(x, y)$ and a lower envelope $e^-(x, y)$ based on the found extrema. The envelopes are constructed using the Radial Basis Function (RBF) for interpolation.
3. Find the local mean of the envelopes by calculating the average at each position (x, y)

$$m_l(x, y) = \frac{e^+(x, y) + e^-(x, y)}{2} \tag{2.21}$$

4. Subtract the mean from the input

$$h_l = x - m_l \tag{2.22}$$

5. Stop sifting if the stopping criterion

$$SD = \sum_{x,y} \left[\frac{|h_{l+1}(x, y) - h_l(x, y)|^2}{h_l^2(x, y)} \right] < \epsilon$$

is satisfied. An IMF is found and so we can set

$$c_l = h_l \tag{2.23}$$

6. Repeat from step 1, now with the new input

$$x_{l+1} = x_l - c_l \tag{2.24}$$

Finally, the splines are created using RBF interpolation, which interpolates better than the cubic spline, when near the edges. The RBF is given on the form:

$$s(x) = p_m(x) + \sum_{i=1}^{N} \lambda_i \Phi(||x - x_i||) \tag{2.25}$$

where p_m is a low degree polynomial, λ_i are the RBF coefficients, Φ is the basis function and x_i are the RBF centres.

2.2.2 Directional Empirical Mode Decomposition

Directional Empirical Mode Decomposition (DEMD) was introduced by Z. Liu [12] in 2004 and takes a fundamentally different approach in its attempt at expanding EMD into 2D. Instead of processing the entire image in one go, DEMD applies EMD to individual rows or columns in the image. The original algorithm presented in the paper allows for a variable rotation θ, which makes it possible to perform DEMD in any direction. In this thesis, a simplified version will be presented where it is only possible to perform the EMD along the rows or columns of the image. The rotational property is mostly interesting for texture analysis or other patterns that do not necessarily follow the horizontal or vertical axis.

Firstly, we define the 2D IMF and also the DEMD of an image:

Definition 1 - A signal $u(x, y)$ is defined as a 2D IMF, where we define:

$$\begin{aligned} v_c(x) &= u(x, c) \\ v_r(x) &= u(c, x) \end{aligned} \tag{2.26}$$

if for any value c, $v_c(x)$ and $v_r(x)$ satisfy the requirements for a 1D IMF. $v_c(x)$ and $v_r(x)$ are named 1D samplings of the IMF.

Definition 2 - The DEMD of an image $f(x, y)$ is defined by:

$$f(x, y) = \sum_{i=1}^{N} \text{IMF}_i(x, y) + r_N(x, y) \tag{2.27}$$

where IMF_i are the 2D IMFs from the decomposition and r_N is the residue. The residue r_N will have at least one monotonic 1D sampling for $v_c(x)$ or $v_r(x)$.

With these definitions, the sifting process with an input $h_l(x, y)$ can be described by the following steps:

1. For each row of the input $h_l(x, y)$, find the local extrema.
2. For each row, create an upper envelope $e^+(x, y)$ and a lower envelope $e^-(x, y)$ based on the found extrema.
3. Calculate the mean of the envelopes:

$$m(x, y) = \frac{e^+(x, y) + e^-(x, y)}{2} \tag{2.28}$$

4. Then, find the extrema along the columns of the just found mean $m(x, y)$. Interpolate based on the extrema, again along the column, and then find the mean of the envelope:

$$m_l(x, y) = \frac{e_m^+(x, y) + e_m^-(x, y)}{2} \tag{2.29}$$

5. Set:

$$h_{l+1}(x, y) = h_l(x, y) - m_l(x, y) \tag{2.30}$$

6. Check whether the stopping criterion is satisfied:

$$SD = \sum_{x,y} \left[\frac{|h_{l+1}(x, y) - h_l(x, y)|^2}{h_l^2(x, y)} \right] < \epsilon$$

   if it is, set $\text{IMF}_i(x, y) = h_{l+1}(x, y)$ and update the residue $r_{i+1} = r_i(x, y) - \text{IMF}_i(x, y)$. If not, repeat steps 1-6 with new input $h_{l+1}(x, y)$ until the stopping criterion is satisfied.

That completes the sifting step for one IMF. This sifting process is repeated for every IMF until there exists a monotonic 1D sampling contained in one of the rows or columns of the residue $r_i(x, y)$. With a monotonic row or column found, the signal is fully decomposed and the DEMD algorithm can terminate.

2.2.3 Image Empirical Mode Decomposition

The Image Empirical Mode Decomposition (IEMD) was developed by Anna Linderhed and published in her PhD thesis in 2004 [13]. IEMD bares many similarities to the 1D method, with just a few minor modifications for make it suitable for 2D signals. BEMD and IEMD are similar in their approach.

Given an input image $x(x, y)$, where x and y are the pixel locations along the x- and y-axis correspondingly, then one sifting of the IEMD is comprised of the following steps:

1. Find the amplitude and location of the local extrema contained in the input 2D signal $h_{lk}(x, y)$.

2. Use thin-plate smoothing spline interpolation to create an upper envelope, $e^+(x, y)$, and a lower envelope, $e^-(x, y)$ of the extrema found in step 1.

3. For every position (x, y) calculate the mean of the upper and lower envelope, given by

$$m_k = \frac{e^+(x, y) + e^-(x, y)}{2} \tag{2.31}$$

4. Remove the mean from the input $g(x, y)$

$$h_{l(k+1)}(x, y) = h_{lk}(x, y) - m_{lk}(x, y) \tag{2.32}$$

   This concludes one sifting step.

5. Check with a predefined stopping criterion, and repeat from step 1 if it is not satisfied.

In the PhD thesis, Linderhed defined this stopping criterion as:

$$|m_{lk}(x, y)| < \epsilon \quad \forall\, (x, y) \tag{2.33}$$

where ϵ is set to a small value. This stopping criterion is satisfied when the mean of the envelope is close to zero. The goal of this stopping criterion is to ensure symmetry in the envelope so that the number of zero crossings are correct and satisfy the requirement of an IMF. If ϵ is set too large the sifting process will terminate too early, which results in the IMFs not getting extracted correctly. Setting ϵ too small, will on the other hand result in the sifting process continuing for a long time-period. Thus, the value of ϵ has to be tuned according to the input data. Given that the stopping criterion is satisfied, the IMF can be defined by:

$$c_l(x, y) = h_{lk}(x, y) \tag{2.34}$$

and the sifting process is repeated again now with the new input:

$$x_{l+1}(x, y) = x_l(x, y) - c_l(x, y) \tag{2.35}$$

The IEMD algorithm decides whether a pixel is an extrema based on either 4- or 8-connected neighbours. This works by comparing the specific pixel value with its 4 or 8 surrounding neighbouring pixel values. If the pixel value is greater (or equal) to its neighbours, the position is labelled as a minimum or maximum. Linderhed presents two ways to find these extrema; a simple method and a morphological method. The simple method does an *if-else* comparison between the center pixel and its neighbours to determine whether it is an extrema. The morphological method uses the MatLab function `imregionalmax`, where the only thing differentiating them is that the simple method must be greater than its neighbours, while the morphological method must be greater *or equal* to its neighbours. This small difference in implementation gives slightly different IMFs as output.

The number of found extrema can be very high, and so a simple filtering of the extrema points to select the most significant extrema is performed. Extrema that have a small amplitude are filtered out as insignificant according to the following rule:

$$b(x,y) = \begin{cases} 0 & \text{if } |b(x,y)| \leq T, \\ b(x,y) & \text{otherwise} \end{cases}$$

where T is the threshold and $b(x,y)$ a given the extremum. This filtering of the extrema improves the time performance of the algorithm with a minimal deterioration to the quality of the output.

The spline used in the IEMD algorithm is the thin-plate smoothing spline (*tpaps*-function in MatLab), which is a 2D equivalent to the cubic spline normally used in the 1D-EMD. The thin-plate spine aims at selecting a function $f(x)$ that interpolates the provided extrema values and simultaneously minimizes the bending energy [14] according to the following equation:

$$E[f] = \int_{\mathbb{R}^n} |D^2 f|^2 dX \tag{2.36}$$

where $D^2 f$ is the second-order partial derivatives matrix of f and $|D^2 f|^2$ is the sum of squares for all matrix entries.

2.3 Mode Mixing Separation

While EMD is a very powerful tools for analysis of nonlinear and nonstationary signals, the output often suffers from a phenomenon called mode mixing. Mode mixing occurs when 2 or more of the modes in a signal have frequencies that lie close to each other. In cases like these, EMD will have issues distinguishing the modes, resulting in multiple frequencies appearing in the same IMF. In the regards to noise filtering, mode mixing is undesirable since the highest frequency IMF should only contain the noise and not other details that are unrelated to the image details.

Mode mixing occurs when the frequency and/or amplitude ratio (f_2/f_1 and a_2/a_1) between two components moves towards 1. The boundary map illustrating the ratios where a clean separation is attained using EMD, was initially presented in the paper *One or Two Frequencies? The Empirical Mode Decomposition Answers* [3]. A reconstruction of the boundary map for clean separation of modes is shown in Figure 2.9 and gives an overview of what ratios mode mixing will occur at. Not surprisingly, when the frequencies or amplitudes become more similar it will become harder to distinguish the components and at some point we will have mode mixing. For frequency, the boundary for mode mixing appears sharply at a ratio of $f_2/f_1 \approx 0.67$.

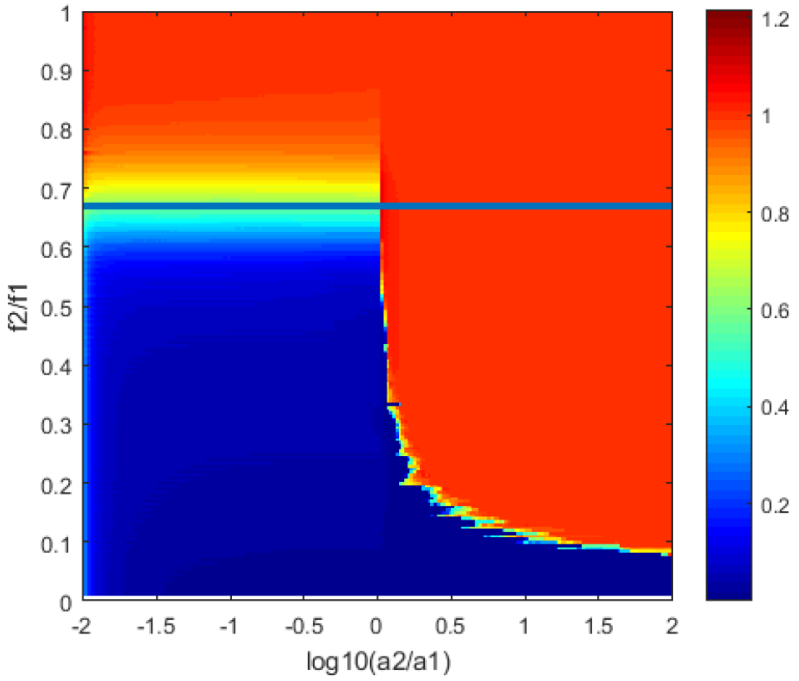

Figure 2.9: The mode mixing boundary condition map based on the frequency and amplitude ratios of two components [15]. A lower value indicates a better separation.

One solution to mode mixing, was published by Ryan Deering and James F. Kaiser [16], where they suggest applying a masking signal to the input signal. This is a simple yet efficient tool for separating frequencies into independent components. This method works by adding a masking signal with a frequency that is higher than the highest frequency component in the signal. The masking signal will then ”attract” the highest frequency component away from the lower frequency component it was mixing with. Afterwards, the masking signal can be removed, leaving only the clearly separated component. The mode mixing separation method using a masking signal, can be described with the following steps:

1. Create a masking signal $s(n)$ based on the frequencies of the input signal $x(n)$.
2. Create two signals, the first by summing the masking signal with the input signal:
$$x_+(n) = x(n) + s(n) \tag{2.37}$$
and the second by subtracting the masking signal from the input:
$$x_-(n) = x(n) - s(n) \tag{2.38}$$
3. Perform EMD on $x_+(n)$ and $x_-(n)$, obtaining the IMFs $c_+(n)$ and $c_-(n)$.
4. Cancel out the masking signal by calculating the average
$$c(n) = \frac{c_+(n) + c_-(n)}{2} \tag{2.39}$$

The signal $x(n)$ contains two components with frequencies f_a and f_b, where the frequency of the masking signal, f_m, is set higher than f_a. If f_m is chosen correctly the modes f_a and f_m will be contained in c_- and c_+. By calculating the average, the masking signals in c_- and c_+ will cancel each other out because of their opposite sign. This leaves us with an IMF, c, which only contains the highest frequency component.

2.4 Radial Basis Function

The Radial Basis Function (RBF) is widely used for interpolation in multidimensional space of irregularly placed data points. RBF interpolation in 2D can be compared to trying to bend a flexible metal sheet with the goal of making it pass through all the data points. The metal sheet has different properties that affect its flexibility, and that need to be carefully selected so that it can be formed accordingly. Similarly, when we want to perform RBF interpolation, we need to carefully tune the model according to the data. The RBF is an exact method, which means that the spline has to pass through every data point. This is in contrast to many other interpolation methods, where the spline does not necessarily pass directly through the data points, but rather do close approximations. While the RBF is a powerful method, it does not work optimally for rapidly changing values and so the properties of the data have to be taken into consideration when performing the interpolation.

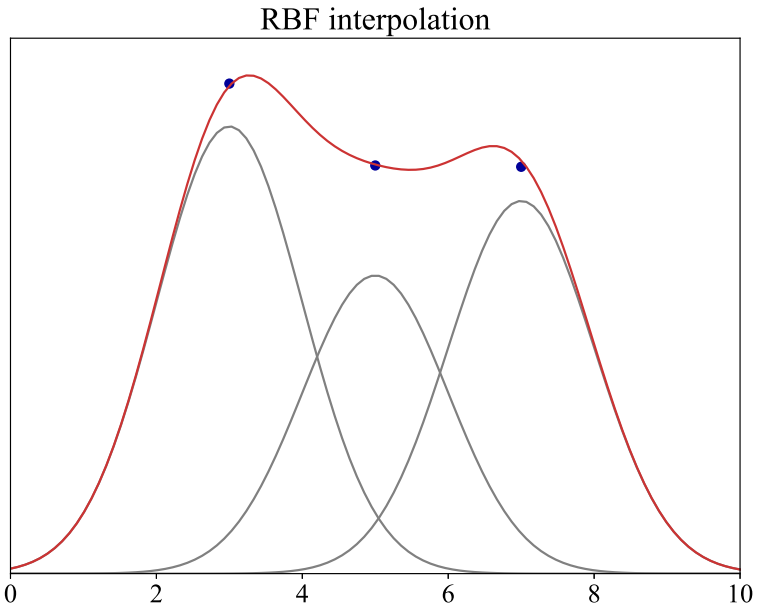

Figure 2.10: RBF interpolation of 3 points. The red line is the interpolation, while the gray lines are the RBF basis functions for each point.

RBF interpolation can most simply be described as multiple Gaussian distributions that summed together create a surface connecting a set of data points. If we define $(x_n, y_n) \in D$, where

(x_n, y_n) are data points in the dataset D, then the influence $h(x)$ is defined by $\|x - x_n\|$ [17]. From this, the RBF can be defined as:

$$h(x) = \sum_{n=1}^{N} w_n \varphi(-\gamma \|x - x_n\|^2) \tag{2.40}$$

where w_n is a tunable weight parameter and φ is a basis function. There exists different basis functions, for example the *multiquadratic basis function* or *polyharmonic basis function*. The most widespread basis function is the Gaussian basis function which is defined as:

$$h(x) = \sum_{n=1}^{N} w_n exp(-\gamma \|x - x_n\|^2) \tag{2.41}$$

When fitting the spline to our input point, we want to find weights $w_1, ..., w_n$ so that a given error is $E_{in} = 0$. To ensure that the error is zero, we must ensure that $h(x_n) = y_n$ for $n = 1, ..., N$, which for a given point x_n can be written as:

$$h(x_n) = \sum_{m=1}^{N} w_n exp(-\gamma \|x_n - x_m\|^2) = y_n \tag{2.42}$$

Equation (2.42) calculates the sum influence of all the neighbouring Gaussian basis functions on the point x_n. We see that the sum value of the function for each point x_n depends on the distance to its neighbouring points x_m and also the weight value w_n. The total contribution of all the points needs to be adjusted, so that the sum equals the target value y_n. Solving this for all N points results in a matrix with N equations and N unknowns that has to be solved. We set this up on the following form:

$$\begin{bmatrix} exp(-\gamma\|x_1 - x_1\|^2) & \cdots & exp(-\gamma\|x_1 - x_N\|^2) \\ \vdots & \ddots & \vdots \\ exp(-\gamma\|x_N - x_1\|^2) & \cdots & exp(-\gamma\|x_N - x_N\|^2) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \tag{2.43}$$

This can be rewritten on the form:

$$\Phi w = y \tag{2.44}$$

where if Φ is invertible, then:

$$w = \Phi^{-1} y \tag{2.45}$$

While this gives us an exact interpolation of our data points, the linear equation that needs to be solved in (2.43) can quickly grow in complexity. This is means that implementing and doing 2D EMD will often be limited by the run-time requirements needed to perform the RBF interpolation.

CHAPTER 3

METHOD AND IMPLEMENTATION

THIS chapter will present the implementation and solution for noise filtering using EMD. It introduces multiple new approaches to the process of noise reduction using EMD in 2D. Most of the methods presented have been tried in one shape or form in 1D, but are unique in terms of 2D implementation. Because the use case is new, the methods do not necessarily improve the images, but are presented nonetheless to show both the limitations and possibilities that lie within each respective method.

Applying EMD on images will in the remainder of this thesis be referred to as Image Empirical Mode Decomposition (IEMD), to better separate between EMD when it is performed in 1D or 2D.

3.1 Data and datasets

Two datasets will be used to more thoroughly study the filtering capabilities of the IEMD based filters. The first dataset is a set of labelled satellite images of icebergs and boats, while the second is a small dataset of medical ultrasound images. These two sets will be presented in the following subsections. Additionally, two images are used as "consumer" type images to give a better indication of how IEMD based filtering works for conventional images. These two images are show in their original format in Figure 3.1.

3.1.1 Iceberg dataset

One of the main goals of this thesis, is to use IEMD on images is to improve classification accuracy of noisy datasets. A particularly interesting dataset is the iceberg dataset published in the *Statoil/C-CORE Iceberg Classifier Challenge* [18]. This dataset contains 1604 labelled satellite images of either icebergs or boats. The resolution is 75×75 pixels which means that the level of detail is very low, making the classification extremely challenging.

Figure 3.2 shows an example of the quality of the images in the dataset. As one can see, the amount of noise in the images is significant. The images are attained by an active satellite radar system that operates in the C-Band. The C-Band ranges between 4 and 8 GHz, making it

Figure 3.1: Original images used in filter testing. The left image will in the following sections be referred to as the ”Lena” image and similarly the right image will be referred to as the ”Ellipses” image.

possible to do remote sensing through visual obstructions like fog, rain and clouds. The radar emits a signal and measures the backscatter from the objects that gets hit by the active source. From this, it is possible to build up an image of the sea surface. A higher backscatter will result in a brighter pixel value in the image, and similarly less backscatter will appear as darker pixels. Solid objects have a higher reflection, thus we see that the boat and the iceberg appear as bright objects in the image. More waves result in a brighter background because more of the energy source is scattered back when waves are present.

The images are gathered using two different methods for transmission and reception. The first, is Horizontal/Horizontal (HH) which is a horizontal transmission and reception. The second is Horizontal/Vertical (HV) which is horizontal transmission and vertical reception. These two bands will have slightly different characteristics because of how the signals bounces off surfaces. Additionally, there is a composite of the images which is simply the average of the two bands.

3.1.2 Medical ultrasound dataset

The ultrasound dataset was created using the GE Logiq 9, a general usage ultrasound device for medical imaging using a linear array transducer. The images were taken at Vestfold Hospital Trust and were recorded specifically for this thesis. None of the images were used for any medical purposes and are only used in this thesis for testing purposes. Most of the images are recordings of the upper throat region and jaw.

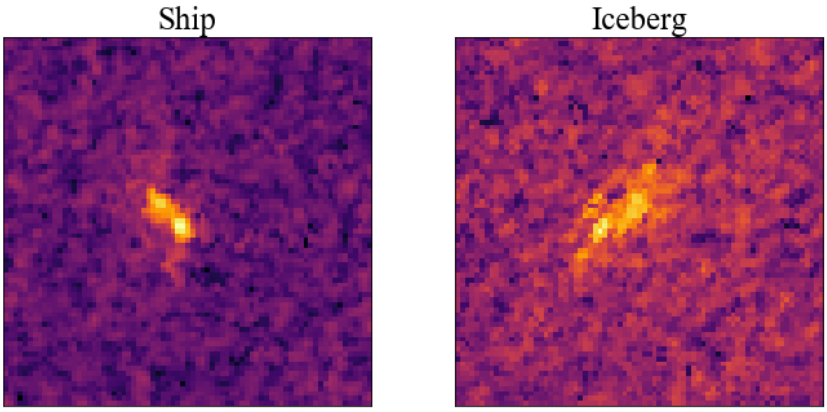

Figure 3.2: Two samples taken from the iceberg dataset of respectively a boat and an iceberg.

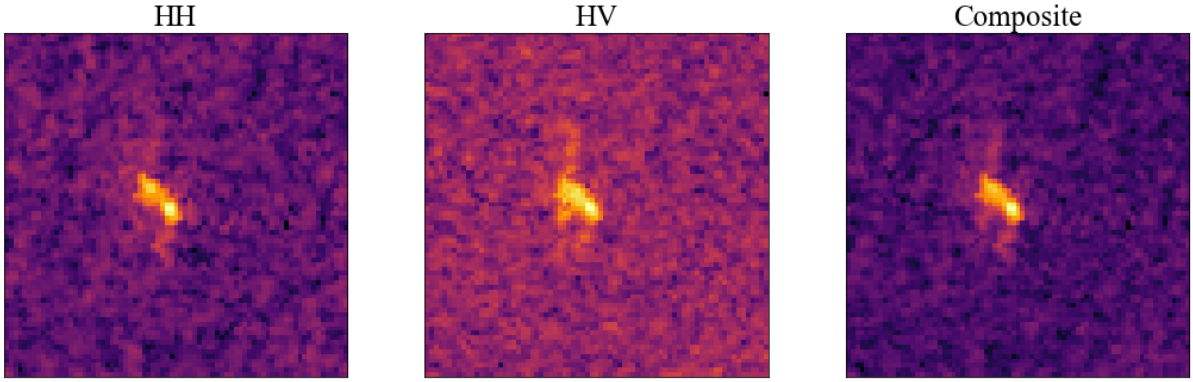

Figure 3.3: The same boat as in Figure 3.2, is here shown together with the two bands and their composite

3.2 Implementation of IEMD

The algorithmic steps of the IEMD method follows a procedure that is very similar to EMD. The code implementation is unique, and contains modifications that improve the performance over the BEMD and IEMD methods presented in Section 2.2.2 and 2.2.3 respectively. The following section will present a thorough description of the steps involved in the IEMD method.

The method presented in the following section is not to be mixed with the IEMD method presented in Section 2.2.3 which carry the same name, but is a different method with its own implementation.

3.2.1 Extrema detection

The extrema detection is based on morphological reconstruction. More specifically, the MatLab functions `imextendedmax` and `imextendedmin` were used to find the extrema. These function do not only find the extrema points, but actually also extrema regions. This means that if there is an extrema point with neighbouring points that have the same value, these also get classified as extrema.

$$\begin{matrix} 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 15 & 15 & 10 & 19 & 10 \\ 10 & 15 & 15 & 10 & 20 & 10 \\ 10 & 10 & 10 & 10 & 19 & 10 \\ 10 & 10 & 10 & 25 & 25 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 \end{matrix} \xrightarrow{imextendedmax} \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \tag{3.1}$$

A subset of the pixel values of an image is shown in (3.1), where we also can see the output from `imextendedmax`. We see here that whole regions will be marked as extrema. This gives us a binary map of all the extrema locations, that can be used to model the spline on. While this binary map contains the extrema locations, it is missing the amplitude value for a give extremum. An element-wise multiplication of the original image and the binary extrema map as shown in (3.2) can be performed to extract the extrema values with their corresponding positions. This leaves us with the point data needed to construct the splines required to perform IEMD.

$$\begin{bmatrix} 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 15 & 15 & 10 & 19 & 10 \\ 10 & 15 & 15 & 10 & 20 & 10 \\ 10 & 10 & 10 & 10 & 19 & 10 \\ 10 & 10 & 10 & 25 & 25 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 \end{bmatrix} \circ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 15 & 15 & 0 & 0 & 0 \\ 0 & 15 & 15 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 25 & 25 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.2}$$

If we apply this to the HH image in Figure 3.3, an image of the regional extrema can be constructed. Such visualization can be seen in Figure 3.4, where each individual white dot is an extremum point.

3.2.2 Interpolation using ALGLIB

After finding the extrema positions and values, we can now start creating the splines. Creating a good spline in 2D is the most computationally heavy and complex part of the IEMD process. It is essential that the interpolation process can be terminated within a reasonable time for IEMD

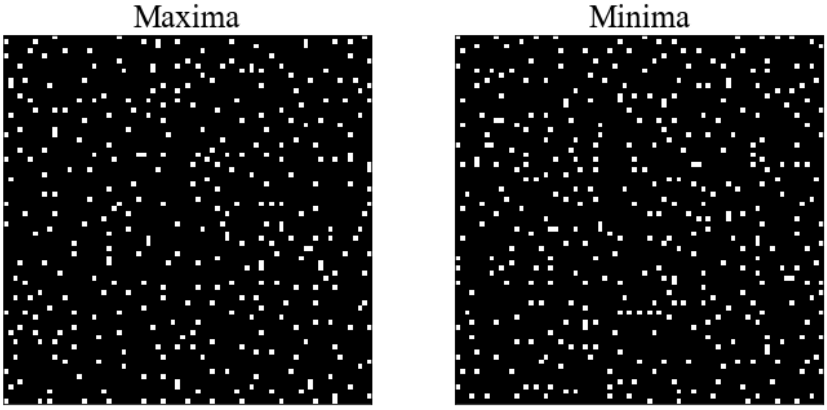

Figure 3.4: Binary map of the extrema in the HH band taken from Figure 3.3.

to be a viable filtering tool. A unique solution to this complexity issue is presented in the following section.

Performance has been a focus point while developing the algorithms used in the IEMD method. The conventional interpolation libraries are not able to process images over any significant size ($> 100 \times 100$ pixels) when run on a normal computer. The most popular library in Python for RBF interpolation is the `interpolate.Rbf` function in the SciPy library. This function is not designed for interpolation of large and dense sets of data points. The time complexity of this function is $O(N^3)$ and it has a space complexity of $O(N^2)$ [19]. This is extremely poor performance [20] and it could take down even the most impressive computer rigs. With the goal of improving the performance, ALGLIB was used to perform the interpolation. ALGLIB is a library for numerical calculations, and contained in it is a significantly faster implementation of the RBF interpolation method. In comparison to Scipy, ALGLIB has a time complexity of $O(N \cdot logN)$ and a space complexity of $O(N)$, an improvement by orders of magnitude. ALGLIB enables IEMD to be used on images of medium to large sizes without significant issues. Without ALGLIB, applying IEMD on images for filtering purposes would not be viable.

The ALGLIB interpolation function has some tunable parameters that directly affect the quality of the splines. These variables are listed below:

RBase - Defines the width, or spread, of the Gaussian basis function.

NLayers - Defines the number of iterations the algorithm performs and sets a limit to how small features are captured in the interpolation process.

LambdaV - Regularization coefficient which can reduce noise in the data.

Setting these variables correctly can be a challenge and depend strongly on the input data. Since `RBase` defines the spread of the basis function, this variable has to be set relative to the distance between the data points in the input. If it is too small, there will be an error introduced in the splines for the areas covered between the data points. Figure 3.5 illustrates how the interpolation fails to represent the data correctly when `RBase` is set too small. The result is that most of the

spline appears flat for every area far from a point, because the spread of the Gaussian basis function is too small. To avoid getting this error in the splines, there needs to be a way to ensure that `RBase` is always set large enough. Doubling the value of `RBase` will quadruple the run time of the interpolation, so it is important that this value is not set unnecessarily high. The ALGLIB documentation recommends setting `RBase` based on the average distance between the data points according to the following formula:

$$RBase = 4 \cdot d_{avg} \tag{3.3}$$

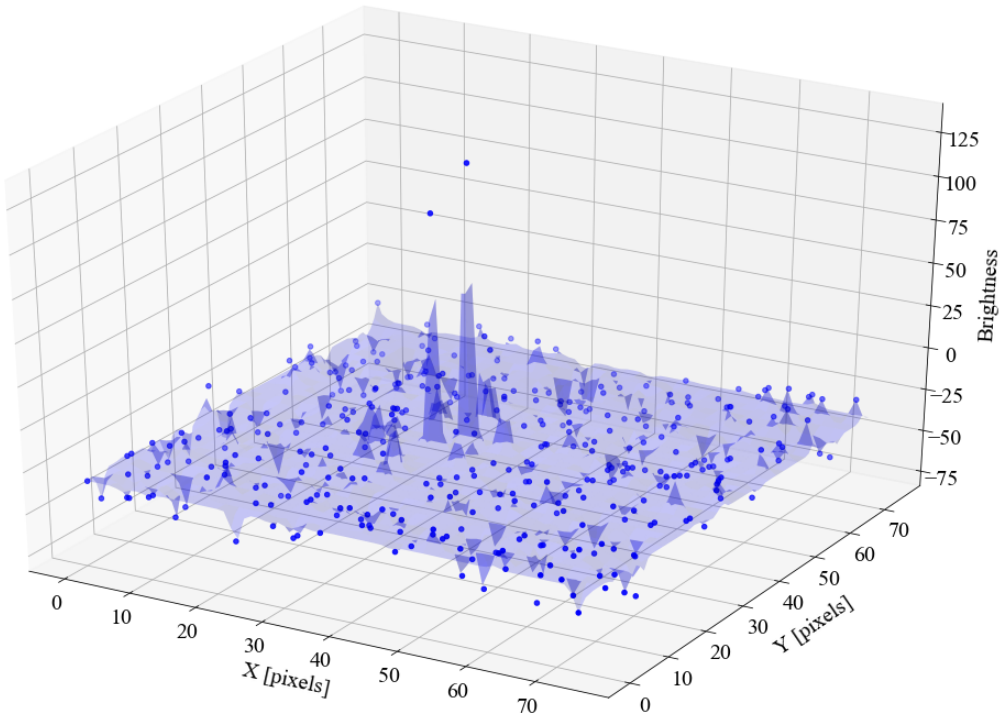

Figure 3.5: Interpolation with `RBase` $= 1$, a value that is too small for a correct interpolation.

where d_{avg} is the average distance between the data points. This formula by itself is not complex, but finding d_{avg} is. Therefore we need a simple way to estimate d_{avg} without performing calculations on every single point. To do this, we make an assumption that the extrema points are to some degree uniformly distributed. This is a fair assumption for images that contain noise, because the noise is often uniformly represented in the whole image, and thus the extrema will also be present in the whole image. Based on this we can set up an equation for estimating d_{avg}:

$$d_{avg} = \sqrt{\frac{A}{N_{peaks}}} \tag{3.4}$$

where A is the area in pixels and N_{peaks} is the number of maxima or minima in the data. This gives us the average distance between the points and is an efficient way of setting `RBase`.

Selection of the second variable `NLayers` can be set based on d_{avg}, where the documentation recommends setting it based on the following formula:

$$NLayers = \text{round}\left(\frac{ln\left(2 \cdot \frac{RBase}{d_{avg}}\right)}{ln(2)}\right) + 2 \tag{3.5}$$

Since d_{avg} is already known, setting `NLayers` is fairly straight forward. This is the method used for the implementation presented in this thesis.

`LambdaV` can be set dynamically, but in this thesis it will be set as a constant value $1 \cdot 10^{-3}$, as recommended in the documentation.

3.2.3 Sifting

With the tools defined for finding the extrema points and constructing the spline, we are ready to perform the actual sifting process. This process follows similar steps to the EMD method with minor modifications to accommodate for the additional dimension.

As a first step for our sifting, we recover the minima and maxima that we need to model the splines on. A 3D visualization is shown in Figure 3.6 to give a more intuitive understanding of the steps involved. Using the extrema points, an upper and lower spline is constructed as shown in Figure 3.7. We see that using correct parameters for the splines gives us a smoother interpolation and a more correct representation of the image. Since the splines are only mathematical models representing the data and not actual images, it is necessary to reconstruct a representation of the splines that is easier to work with. A pixel grid can be inputted into the model function, giving us a 2D image reconstruction of the spline. This is done both for the upper and lower spline, resulting in two images. The mean can then be found by taking the average of each individual pixel pair in the upper and lower reconstruction image. Figure 3.8 show what these image reconstructions and their mean could look like. The mean image based on the reconstructions is then subtracted from the input and the sifting is repeated all over again until some stopping criterion is satisfied.

The siftings process can be summarized by the following steps:

1. Find all the maxima and minima points, m_p^+ and m_p^-, in the image by using morphological reconstruction.
2. Create an upper 2D-spline based on the maxima and a lower 2D-spline based on the minima.

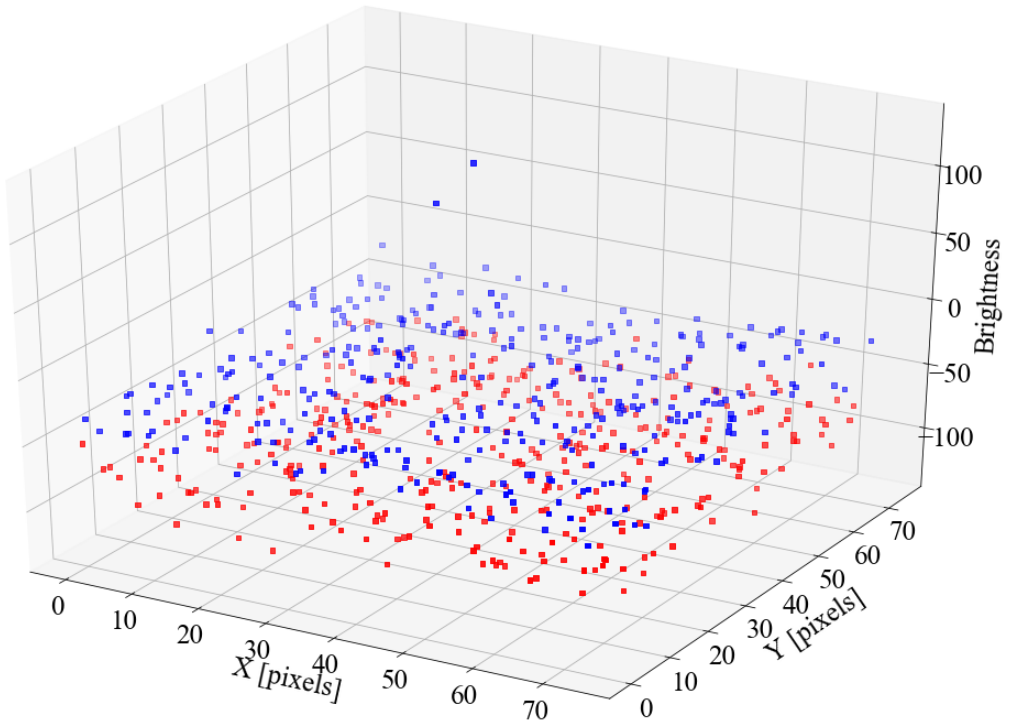

Figure 3.6: 3D map of the extrema in the HH band taken from Figure 3.3. Red points are the minima values and blue are the maxima.

3. Reconstruct an upper image e^+ based on the maxima spline and a lower image e^+ based on the minima spline

4. Calculate the average of the two images

$$m(x,y) = \frac{e^+(x,y) + e^-(x,y)}{2} \tag{3.6}$$

5. Subtract the mean from the input

$$h_l(x,y) = x(x,y) - m(x,y) \tag{3.7}$$

6. If the stopping criterion is satisfied, set the IMF

$$c_l = h_l \tag{3.8}$$

There exists different types of stopping criteria that are possible to use to terminate the sifting. Many of them have limitations that make them less useful. Additionally, they add unnecessary complexity to the process without improving the end result significantly. There are indications

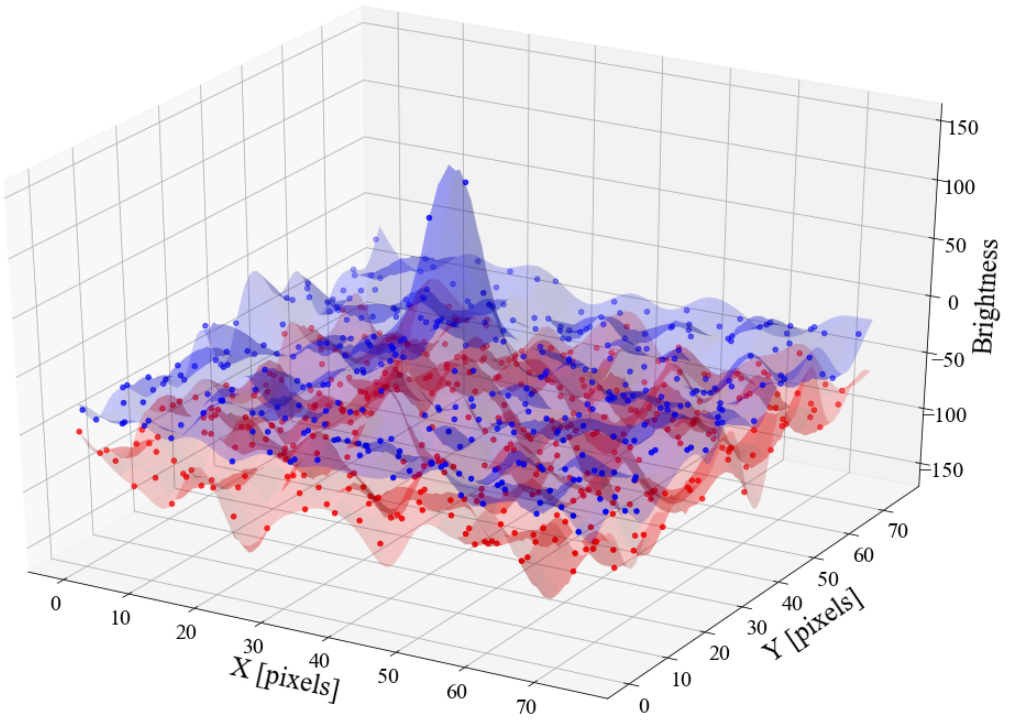

Figure 3.7: 3D map of the splines and extrema in the HH band from Figure 3.3. The red surface is the lower spline, while the blue surface is the upper spline.

that a fixed number of siftings is sufficient [21] enough as a stopping criterion. Thus, this is the stopping criterion used in this thesis because of its simple, robust and predictable qualities. The recommended number of siftings is suggested to be set between 4 and 10. A specific number is not set in this section, but is further investigated in Section 4.1.2.

3.2.4 Noise filtering

With the IEMD method defined and implemented, filtering away noise from a signal using IEMD is fairly simple. We know that the noise has a high frequency characteristic that should be contained in the first IMFs, since they contain the highest frequency modes in the data. To remove the noise from the input data we simply have to subtract the first, or even the second IMF from the input data. How many IMFs that should be subtracted from the input, depends on the input data and the characteristics of the noise. The filtering process can be defined by the following:

$$I_f = I - c_n \tag{3.9}$$

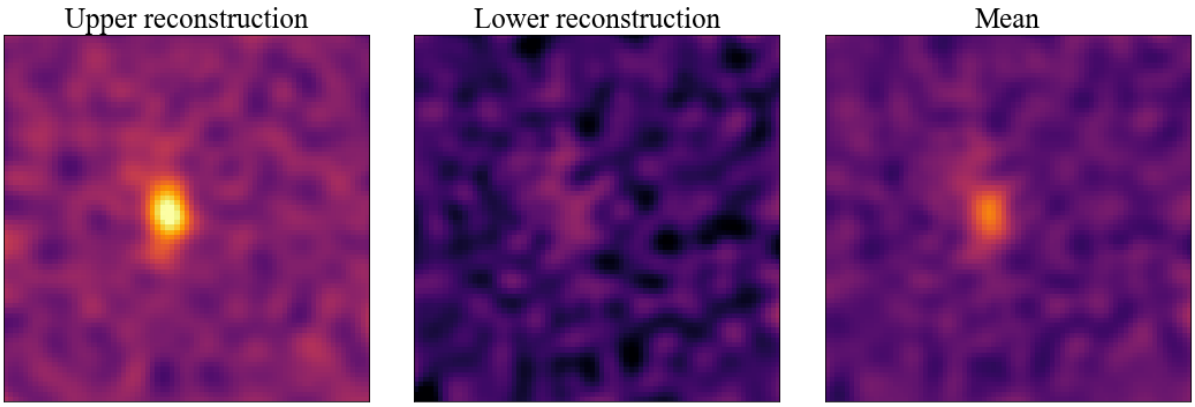

Figure 3.8: Left & middle: Reconstructed images based on the upper and lower spline models. Right: Mean of the upper and lower reconstructions.

where I_f is the filtered image, I is the input image and c_n is the n^{th} IMF obtained by IEMD.

3.3 Mode Mixing Separation for Images

Mode Mixing Separation (MMS) is a method for separating modes that lie close to each other in the frequency domain, that failed to be separated using conventional EMD. This method has been shown to work for 1D signals, and the following section will show that MMS also works in 2D.

Similarly to the 1D implementation presented in Section 2.3, we have to create a masking signal consisting of sine waves that can be included into the input image. We construct the masking signal by combining a vertical and a horizontal sine wave, which together construct a 2D oscillating surface. In a similar manner, test data is constructed that will be used to assess the performance of the MMS method in 2D. Figure 3.9 shows some test images with a given spatial frequency (cycles per pixel) that are so close that mode mixing will occur, if attempted separated with conventional EMD. The spatial frequency of mode 1 and mode 2 is $f_1 = 0.075$ and $f_2 = 0.055$, respectively. This gives us the ratio $f_2/f_1 = 0.73$ which is more that 0.67, thus we lie in the area shown in Figure 2.9, where mode mixing occurs.

The properties of the masking signal have to be selected so that $f_1/f_m > 0.67$ and $f_2/f_m < 0.67$. We see that a masking signal with $f_m = 0.1$ would give us a ratio of $f_1/f_m = 0.75$ and $f_2/f_m = 0.54$, which means that it would only attract one mode. This masking signal is shown in Figure 3.10, where we see visually that the spatial frequency is slightly higher than mode 1 and almost twice as high as mode 2.

The masking signal is then individually both added and subtracted from the image, so that we get two output images with opposite oscillating masking signals. IEMD is then applied to the images followed by an average calculation of the outputs, which cancels out the opposite mask-

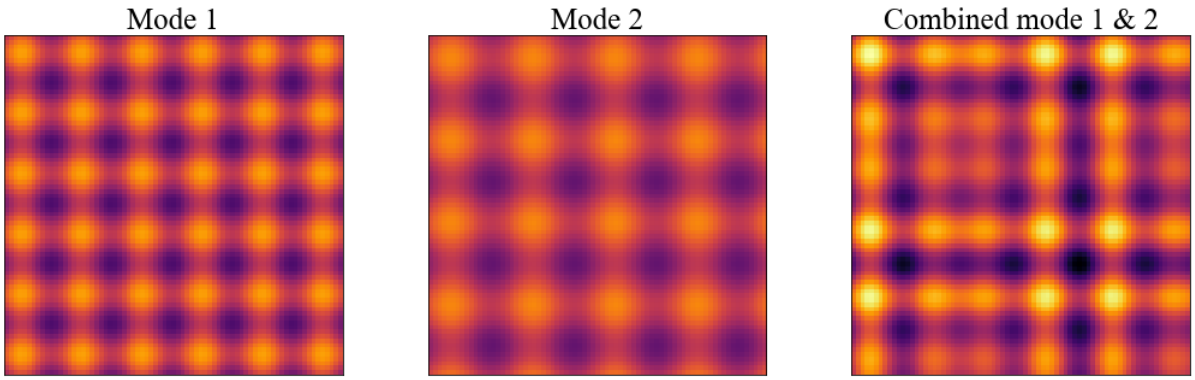

Figure 3.9: Left: Mode 1 with a spatial frequency of $f_1 = 0.075$ and the amplitude is $a_1 = 10$. Center: Mode 2 with a spatial frequency of $f_2 = 0.055$ and amplitude of $a_2 = 8$. Right: Mode 1 and 2 added together.

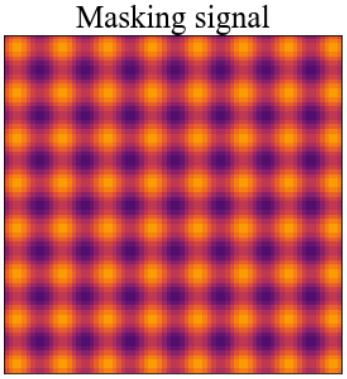

Figure 3.10: Masking signal with $f_m = 0.1$ and $a_m = 10$.

ing signals. This leaves us with an IMF containing the first mode without any mode mixing. In Figure 3.11 we see a comparison of the 1st and 2nd IMFs using IEMD with, and without, a masking signal. We see that using a masking signal significantly improves the separation of the synthetic test data so that that they closely resemble the original modes. Without a masking signal on the other hand, both modes are captured in the first IMF.

3.3.1 Setting masking signal properties

Setting the properties for the masking signal is trivial for the test data because the characteristics of the modes are already known. Optimally, this process should be automatic without the need for any manual configuration. Since this method will be used to remove noise, we can assume that we always want to remove the highest frequency component in the image. Thus, the frequency of the noise has to be found so that a masking signal with slightly higher frequency than the noise can be added to the image.

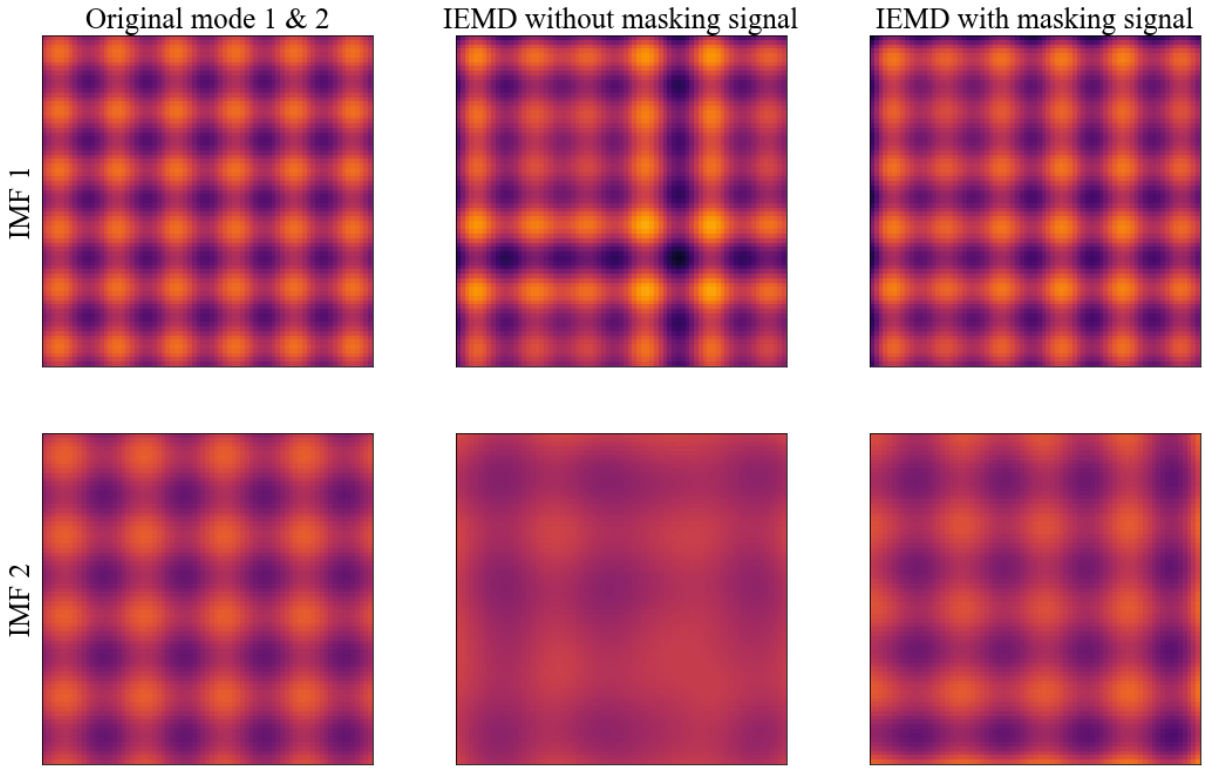

Figure 3.11: First column contains the original modes while the following two columns are attempts at separating the modes using IEMD without and with a masking signal, respectively.

Defining frequency in 2D for a spatial signal may not be very intuitive, but we are for simplicity's sake going to define it here as "peaks per pixels". To find the frequency, we initially find the number of peaks in the data by counting the number of maxima. The number of maxima is divided on the pixel area before taking the square root, giving us the frequency along an axis. The square root is required because the masking signal is created by combining a sine wave in x-direction and a sine wave in y-direction. The formula for finding the frequency can be expressed as

$$f_m = \sqrt{\frac{N_{peaks}}{A}} \tag{3.10}$$

where f is the frequency in x- or y-direction, N_{peaks} is the number of peaks in the data and A is the pixel area ($x \times y$).

The amplitude is found by simply finding the difference between the median maxima and minima values and dividing this difference by two. This can be expressed on the following form:

$$a_m = \frac{(\text{median}\{n_{0,\dots,i}\} - \text{median}\{m_{0,\dots,j}\})}{2} \tag{3.11}$$

where a is the masking signal amplitude, n is the maximum value for a given maximum i and m is the minimum value for a given minimum j.

Now, Equation (3.10) gives us the frequency of the high frequency component of the noise in the data. This is because the noise dominates the extrema count, resulting in a frequency estimate that is close to the noise. Meanwhile, the masking frequency needs to be set to a higher frequency than the highest frequency component, and so we add a scaling value that will ensure that the masking signal has this property:

$$f = F_m \sqrt{\frac{N_{peaks}}{A}}, \quad \text{where } K > 1 \tag{3.12}$$

While a scaling value is not as important for the amplitude value of the masking signal, we can still add it to our equation for future reference:

$$a_m = A_m \frac{(\text{median}\{n_{0,\dots,i}\} - \text{median}\{m_{0,\dots,j}\})}{2} \tag{3.13}$$

This approach is a simple way to automatically find the masking signal parameters. It is worth noting that it does have the downside of not taking into regard local variations in the data, which means that some information will be lost. This could result in poor separation in areas of the image where the parameters do not represent the image accurately. There are no specific rules for what F_m should be set to, but Section 4.1.2 further explores how tuning F_m affects the filtering.

3.4 Adaptive noise filtering with IEMD

The adaptive IEMD noise filter is inspired by the the adaptive filter presented in Section 2.1.2. The adaptive filter is better at preserving image details, because it adapts to local features in the image. Edges and corners can be preserved by removing less noise in these areas, while removing more noise where the image is uniform. Because of mode mixing that occurs when performing IEMD, some of the image details will be contained in the first IMF, or IMFs, which leads to loss of details when subtracting the them from the image. To reduce the amount of blurring, the IMFs can be subtracted from the image in an adaptive fashion, potentially mitigating the mode mixing problem.

The algorithm is very similar to the one described in Section 2.1.2, and can be described by the following steps:

1. Initialize a window size S.

2. Pad the image with a symmetric padding with half the pixel size of the window. Pad size P is given by

$$P = \text{floor}\left(\frac{S}{2}\right) \tag{3.14}$$

3. Run a window over the image, where for each step, the variance of the pixels covered by the window is calculated.

4. Compare the local variance to the noise variance and select one of the following actions:

   a. $Var(w) > Var(n_{noise})$ - The window contains an edge or a corner, keep the original value. Alternatively, set the pixel to the mean of a small window of neighbouring pixels.

   b. $Var(w) \leq Var(n_{noise})$ - The window contains a uniform area, subtract one or more IMFs from the given pixel.

In Figure 3.12 we see a comparison when filtering in the conventional way of removing the first IMF and in an adaptive manner. We see that the adaptive filter preserves the edges of the geometric figures better than if we only used IEMD.

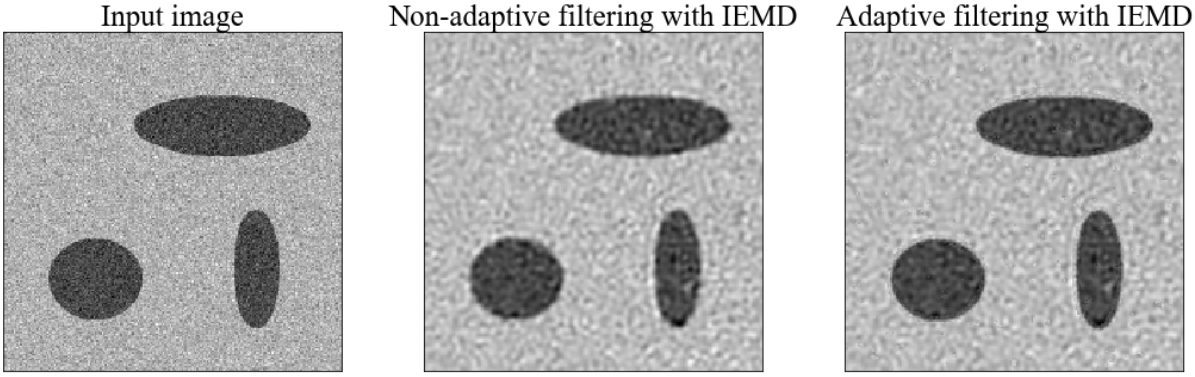

Figure 3.12: Left: Synthetic image with Gaussian noise added. Center: An attempted filtering of the input image by subtracting the first IMF from the image. Right: Image filtered with adaptive filter implementation using IEMD.

3.5 Implementation of Convolutional Neural Network

The CNN is used to differentiate between boats and icebergs in the iceberg dataset. The implementation of the network is based on an implementation published publicly by a contestant [1] in the *Statoil/C-CORE Iceberg Classifier Challenge*. It is based on Tensorflow and Keras, which are two popular deep learning libraries for neural network. Tensorflow is a library containing the fundamental building blocks for creating neural networks and is to date the most popular

deep learning library in use. Keras builds on top of Tensorflow and works as a simplified interface making it easier and faster to construct a network without getting into the lower level details.

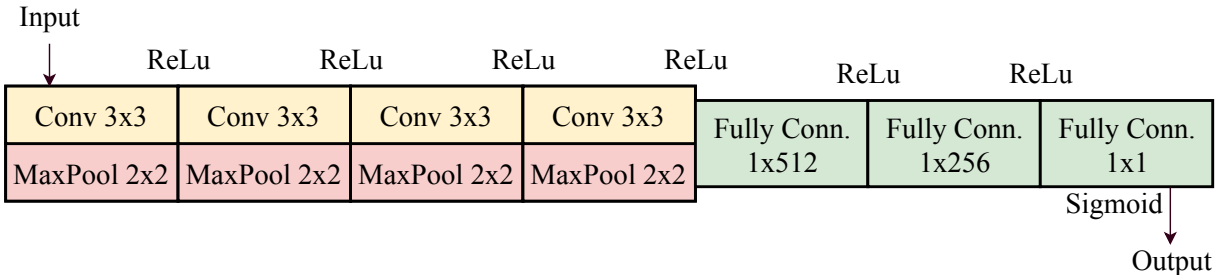

Figure 3.13: Setup of CNN used for classification of icebergs and ships.

Figure 3.13 illustrates the design of the CNN used in this thesis. This model has four convolutional and max pooling layers, with window size of 3×3 and 2×2 respectively. Following this, are three fully connected layers that culminate into a single neuron with a sigmoid activation function. Because of the sigmoid activation the output of this last neuron is always some value between 0 and 1. This property can be used to create a confidence estimate for a given class based on the output. This is useful because our output can then be represented as a probability value which tells us how sure the network is that a given input belongs to a specific class.

The max pooling layers uses a stride of $(2, 2)$, which is a full window jump for each pooling step. Additionally, each layer has a dropout of 0.2, reducing the chance that the network will overfit to the data. Dropout works by removing a random selection of neurons from the model for every training pass. This forces the neurons to become less dependent on each other and also reduces the probability that a single neuron specializes in specific features, thus limiting how likely it is for a neuron to over fit to specific features. As for gradient decent method, a method called Adam (Adaptive Moment Estimation) is used to calculate the gradient.

Because of the high number of convolutional and max pooling layers, the image size right before the fully connected layer is actually only 2×2 pixels large. But because there are at a minimum of at least 64 filters, the number of parameters are still high. In total this model has $560\,193$ trainable parameters, which illustrates how complex even a simple network like this can become.

CHAPTER 4

RESULTS AND DISCUSSION

THIS chapter presents the results and related discussion in regards to noise filtering using IEMD. It contains three main parts that focus on different types of applications and data. The first section is an analysis of the performance of the filtering in regards to different parameters like speed, memory usage, but also visual performance and quality of output. Additionally, different parameter settings and how they affect the output is also assessed. The second section focuses on the classification of the iceberg dataset and how IEMD can be used to improve the classification accuracy. The last section looks at how the noise filters can be applied to medical ultrasound images as a means to improve eligibility and image quality.

4.1 Performance assessment and output analysis

Before applying IEMD based filters it is important to get a more general understanding of how the filter performs and also to what effect the parameters influence the output. This will help us make better choices when applying the filters on real world problems and contribute to getting better results.

4.1.1 Visualization of output and filtering

Firstly, we will look at the different IMFs we get when applying IEMD on an image. Ideally, all noise should be contained in the first IMF so that the noise can be completely removed from the image just by subtracting away the first IMF. This is not necessary always what happens though, and so it is important to look at the IMFs to better understand what output we get.

In Figure 4.1 we see the decomposition of a satellite image of a boat. Most of the details are retained in the first two IMFs, while the remaining IMFs are mostly just blur. In the first IMF we see a good example of mode mixing, the outline of the boat is clearly visible and most of the details that are important for classification are contained in this first IMF. The second IMF has some weak traces of the boat, but largely contain only the lower frequency spectrum of the surrounding noise. It is worth noting that all input images have a pixel intensity range set between -128 and 128. Since this specific input image has a mean of -54 the residue will converge towards this value, while the IMFs on the other hand will have zero mean.

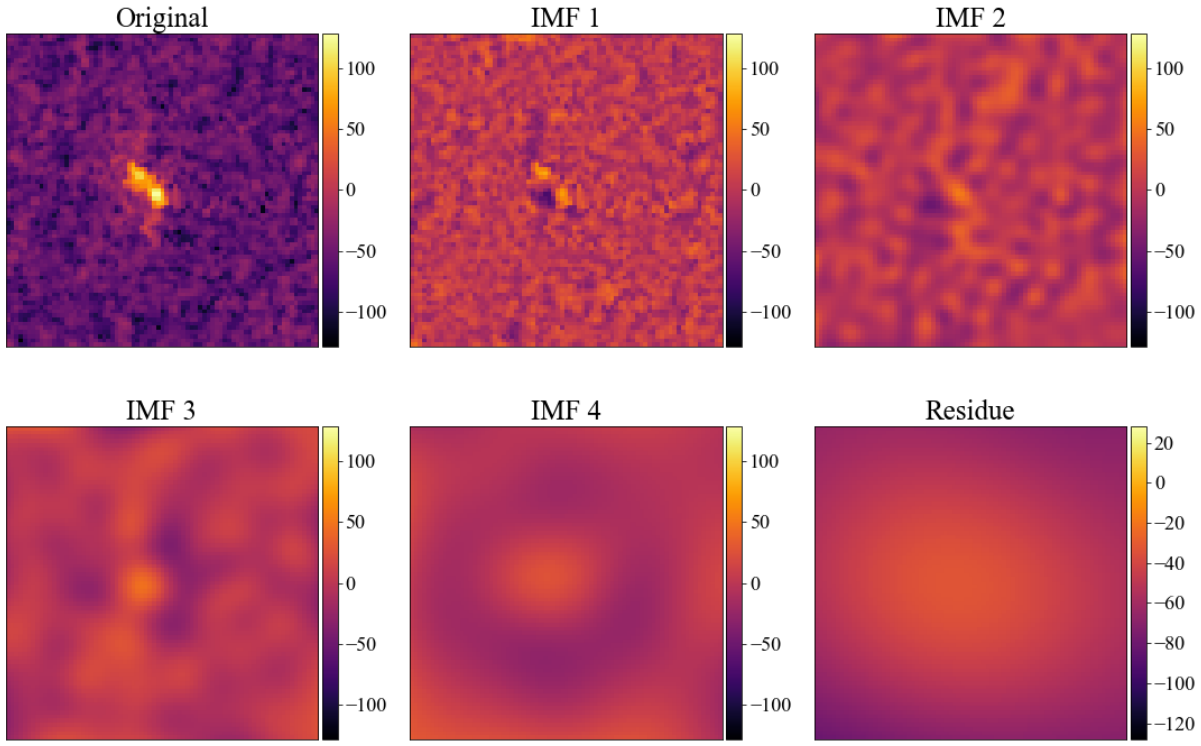

Figure 4.1: First 5 IMFs and residue of the boat image shown in Figure 3.2.

Similarly, we can perform IEMD on the well known Lena image which is a conventional photography without any noise. This will give us IMFs that have different characteristics than in the previous example. The decomposition of the Lena image can be seen in Figure 4.2, where the image has been decomposed into 5 IMFs. The first IMF contains the outlines of the sharpest image details, while the following IMFs have increasingly ”coarser” outlines containing the underlying details in the image. The sharpest details are contained in the first IMF because they have the most rapid changes in image intensity. Details that change rapidly can also be interpreted as having a high frequency, which explains why we do not see sharp details in the following IMFs. We see some degree of mode mixing in the first IMF at the right hat border, that appear as a form of bleeding into the surrounding areas. This is even more apparent in the second IMF, where there are bleeding effects at multiple places. It seems like these artifacts are introduced into the IMFs at areas where there is a rapid change in contrast. For most of the IMFs, this results in bleeding of the edges into the surrounding areas. Most likely, this is caused by errors in the interpolation because of very sudden changes in image intensity. Rapid changes in the data is mentioned in Section 2.10 as a cause of error and what we observe here fits well with this claim.

We can easily perform a filtering of the high frequency components in our input by subtracting the first IMF from the input image. In Figure A.1 we see an attempt of doing exactly that.

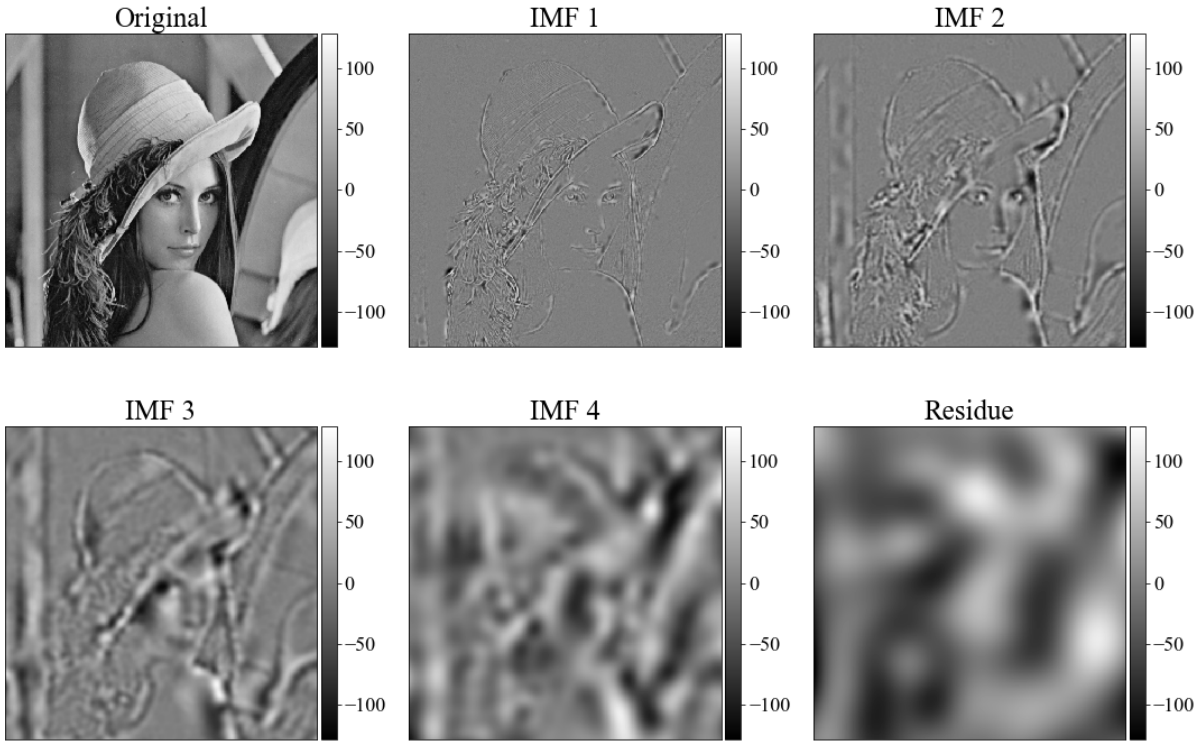

Figure 4.2: First 5 IMFs and residue of the boat image shown in Figure 3.2.

The high frequency noise is successfully captured in the first IMF and most of it removed in the filtered output, but because of the mode mixing we see significant blurring of the boat. Similarly, we can do this to the Lena image by first adding Gaussian noise to the image and then trying to filter away the noise. Also here, we see clear signs of mode mixing in the first IMF, resulting in a blurry output. We see that much of the noise is not actually filtered away from the image. Additionally, there is an introduction of artifacts in the output giving the filtered image a ”wave” or ”bubble” pattern. The source of this pattern can probably be attributed to lower frequency modes of the Gaussian noise that did not get included in the first IMF. Most likely, it is a combination of this and that errors are added into the output as a result of inaccuracies in the interpolation process. The lower frequency modes of the Gaussian noise are hard to remove, because compared to the highest frequency modes, they have a frequency that is much closer to the image details.

4.1.2 Parameter tuning

There is a large variety of parameters available that can, and should, be tuned for the filter to perform optimally. To be better equipped to tune the parameters correctly, we first have to get a better understanding of how they affect the output. Some of these tunable parameters are

related to the IEMD method itself, while others are parameters used in the interpolation library ALGLIB. The following list contains an explanation of available parameters and their meaning:

Depth - Related to the morphological operation used for finding the minimum and maximum positions and values. A higher depth will increase the sensitivity for extrema in that image.

Number of siftings - The number of sifting operations performed for each IMF.

RBase - ALGLIB variable mentioned in Section 3.2.2. Defines the radius of the Gaussian basis function.

NLayers - ALGLIB variable mentioned in Section 3.2.2. Defines the number of layers used to interpolate.

We will just look at the effect of the first two parameters, since the ALGLIB variables are automatically set based on the image properties.

Changing the depth value will change the sensitivity of the maxima extraction. This variable is more thoroughly described in Section 3.2. A larger depth value will make the extrema points larger, so that each extrema do no longer consist of a single pixel value, but rather extrema plateaus. Figure A.4 and A.3 shows the filtered output of the same Lena and boat image using various depth values. As we can see, the depth value has a larger impact on the boat image than on the Lena image. This is because the boat image is significantly smaller in size than the Lena image (512×512 vs. 75×75). For Figure A.3, there are some visible changes that occur when the depth value is changed. There are no obvious indications that the quality of the filtering improves with a higher depth value, but because of the nature of the boat image, it is not possible to conclude anything specifically. For the Lena image, it is obvious that the quality deteriorates quite visibly as the depth increases. This is an indication that what we see for the boat image, is also a deterioration of image quality.

It is a reasonable assumption that IEMD will perform worse on most images, as the depth value increases. With large depth values the extrema points will become plateaus that will constrain the interpolation process and add errors to the spline. These plateaus will act as clusters of points that the splines are forced to follow, thus reducing the splines' ability to estimate the upper and lower envelope. Furthermore, the interpolation performs best when there is a more even distribution of points, which would not be the case when the depth value is set very high. Figure A.4 shows that this error is minuscule as long as the depth is fairly small, but will quickly increase after a certain threshold.

A good stopping criteria limiting the number of siftings is still something left to be discovered by researchers, but there are indications that a fixed number of siftings is sufficient [21]. More specifically, between 4 and 10 siftings should be sufficient for enough for the IEMD method to converge towards an adequate solution. Figure A.5 contains samples of the boat image that has been filtered using different number of siftings. The difference between sifting once and sifting 10 times is not big. Some improvement in sharpness is visible between 1 and 5 siftings, but from 5 to 10 siftings there is little to no improvement. Either way, it is hard to assess whether

one filtered image is better than the other, because the image is quite blurry regardless of what the sifting number is set to.

For the Lena image it is more obvious how the sifting number affects the output. Figure A.6 contains filtered images of Lena for different sifting values and here we see that there is a very small improvement in sharpness as the sifting number increases. After the 5th sifting, increasing the number of siftings seems to have minimal effect on the output. Thus taking into regard the computational time requirements for this algorithm, increasing from 5 to 10 siftings is arguably not worth the minimal improvement in output.

We can subtract the filtered image created with 5 siftings, with the filtered image created with 10 siftings, to better see how small the difference is between these two images. This is visualized in Figure 4.3, for the boat and Lena image. For the boat image, the difference is minimal with just some small perturbations at various locations in the image. There is little to no difference between the two images in regards to the actual boat object at the center. The characteristics for the Lena image is similar, with small perturbations that are not directly related to the details in the image. The perturbations look smaller in the Lena image, because the image is significantly larger in pixel size than the boat image. This example does not give a definite answer, but is a strong indication that going from 5 to 10 siftings has a minimal return on the end result.

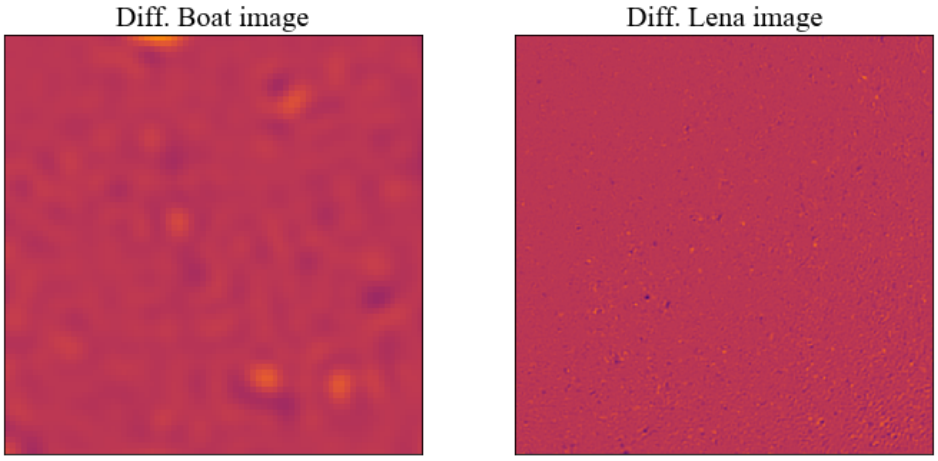

Figure 4.3: Difference of two filtered images with different number of siftings, where the image filtered using 5 siftings is subtracted from an image using 10 siftings.

There are two tunable parameters for the MMS masking signal that affect what frequency and amplitude it will have. These parameters are scaling values that set the frequency and amplitude of the masking signal relative to the average frequency and amplitude of the image. We defined these two scaling values as the following:

A_m - Amplitude modifier for the masking signal amplitude.

F_m - Frequency modifier for the masking signal frequency. Must be > 1.

The masking signal properties is then simply set as:

$$f_m = F_m \cdot f_{avg} \quad a_m = A_m \cdot a_{avg} \tag{4.1}$$

where f_m and a_m are the masking signal amplitude and frequency and f_{avg} and a_{avg} is the average frequency and amplitude of the input image.

We want the masking signal to have a frequency that is low enough to attract the noise, but high enough to not attract image details. Similarly, we want the amplitude to be in a range where we get the correct attraction of modes. Since we are attempting to filter out noise, it is important that F_m is larger than 1 so that the frequency of the masking signal is higher than the frequency of the noise in the image. The amplitude modifier A_m should be set to a correct value according to Figure 2.9. A comprehensive comparison chart showing the difference in parameter options can be seen in Figure A.7. Here we can see how changing the masking signal modifiers affect the filtered output. The differences are minimal, but there is a slight improvement in sharpness as the frequency modifier increases. Changing the amplitude modifier on the other hand does not seem to have any effect at all.

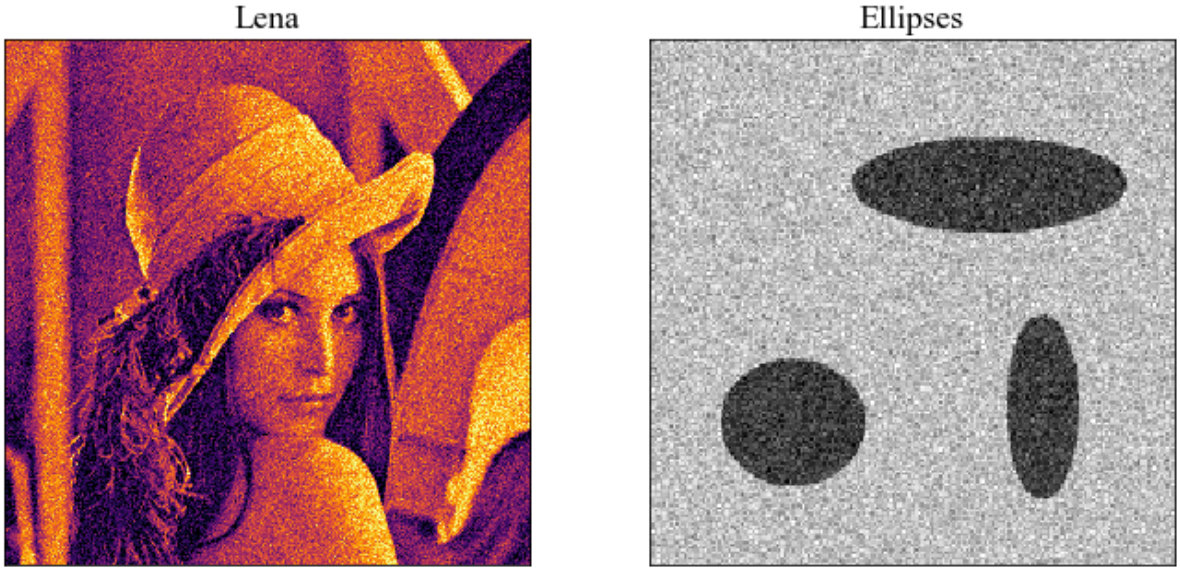

Figure 4.4: Left: Lena image with Gaussian noise. Right: Ellipses image with Gaussian noise.

4.1.3 Performance of individual IEMD solutions

This section will present the different IEMD based filters and investigate how they perform compared to each other. Three filters will be assessed in this section, namely using only IEMD,

IEMD with MMS and an adaptive filter using IEMD. The Lena and Ellipses image with Gaussian noise added to them will be used to test whether the filters can remove the noise. The test images are shown in Figure 4.4.

Filtering using only IEMD is briefly presented in Section 4.1.1, where one could see that the performance of the filter is indeed quite poor. Figure 4.5 shows a second attempt at filtering away Gaussian noise using only IEMD, where only the first IMF is subtracted from the image. As mentioned earlier, most of the high frequency noise is removed while the lower frequency noise is still visible, resulting in the ”wave” pattern all over the image. This is especially visible for the right most image, where for a optimal filtering, the the black and gray colors should be completely uniform. From a visual standpoint, the filtered output is almost in worse condition than the unfiltered input. This can be confirmed numerically by looking at the Signal-to-noise ratio (SNR) of the images, given in Table 4.1. Here we see that the SNR for the images that have been filtered using only IEMD, is just barely higher than for the unfiltered images.

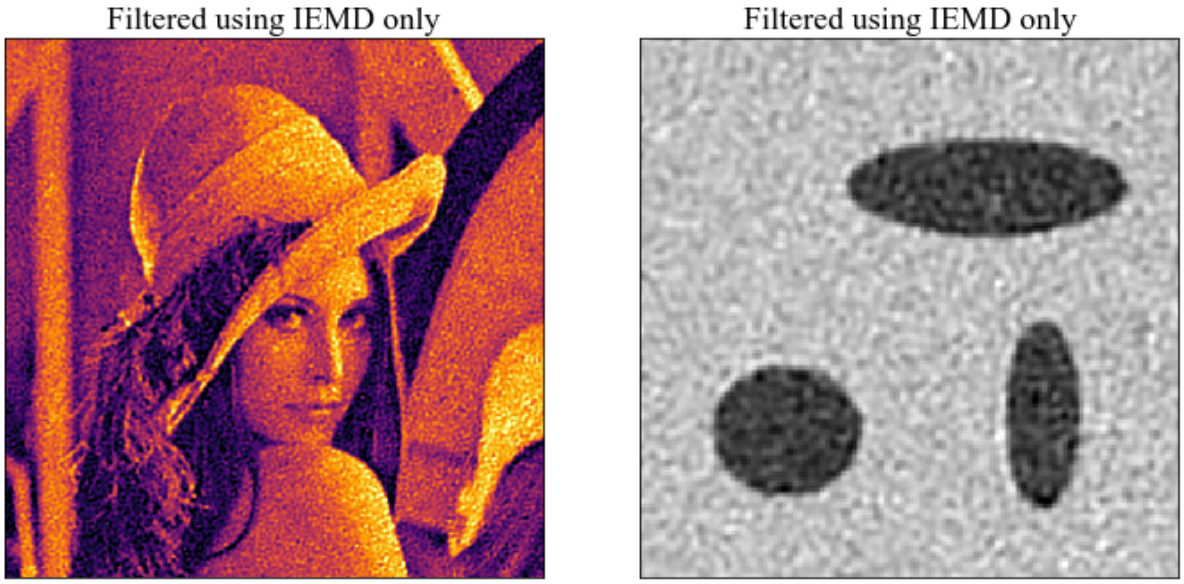

Figure 4.5: The two test images after being filtered using only IEMD.

We perform the same filtering now with MMS to improve the separation of the noise and the image details. The filtered output is shown in Figure 4.6 and we see a clear improvement over just using IEMD without MMS. The edges are sharper and more details are preserved in the output. We still have the residue of low frequency noise, but it is now more uniform and less prominent than in the previous example. While this filter does not yield great results, it is proof that using MMS can have a positive contribution to the overall results. The fact that the residue noise is more uniform, is a sign that the masking signal contributes to not only attracting specific modes, but also that it reduces local variations in the decomposition. The masking signal is able to attract the highest frequency components away from the modes containing image details, resulting an output that is of slightly better quality. While the output is more visually appealing, Table 4.1 indicates that the filtered output actually has a worse SNR when using MMS. This

is because the MMS actually makes the filter remove less noise than it would if MMS was not used. The filter removes a more narrow band of the upper part of the frequency spectrum, which means that more noise will remain in the image, but only as low frequency noise.

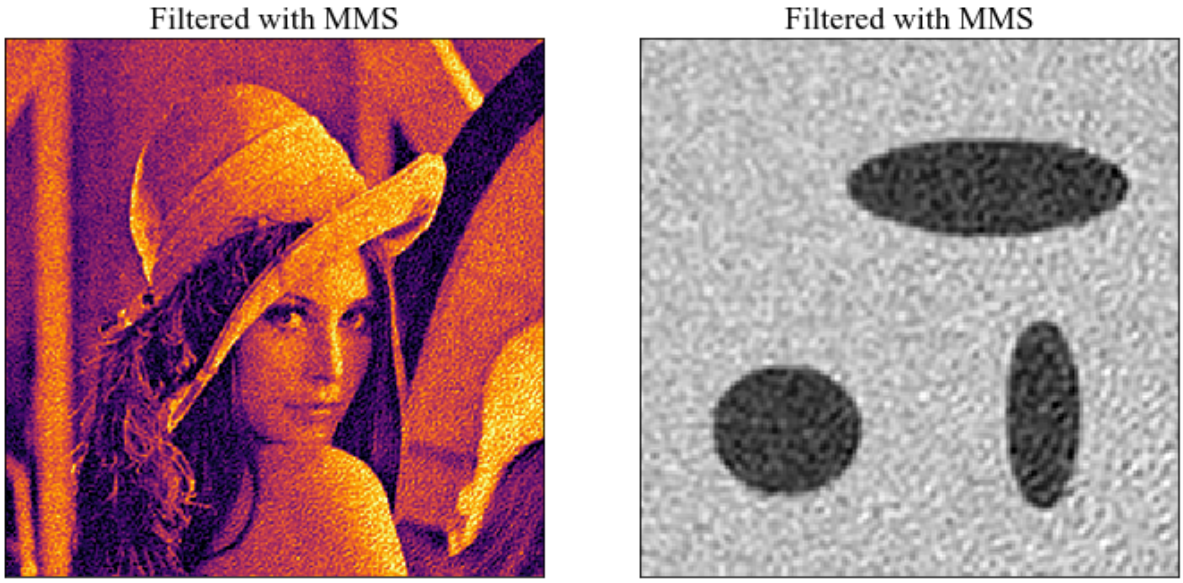

Figure 4.6: The two test images after being filtered using IEMD with MMS.

While MMS does reduce the blurring somewhat, important details are still lost in the filtering process. The third filter method tries to solve this problem by taking an adaptive approach to the filtering task. The adaptive IEMD filter tries to retain more of the details, by only subtracting the IMF from the image in areas where there are no edges or corners. The filtered output is shown in Figure 4.7, where an attempt to filter out the noise is performed using the adaptive filter. The difference between this example and the MMS based filter is not immediately obvious, but when taking a closer look, we see that the output images from the adaptive filter have slightly sharper details. This is especially visible for the ellipse image where the outlines of the geometric objects are visibly sharper. Table 4.1 shows that for the Lena image, the adaptive approach gives the highest SNR values of all three filters. It is the opposite case for the Ellipses image, where the adaptive filter has the lowest SNR of all the filters.

Table 4.1: Signal-to-noise ratio

Name	**Unfiltered**	**IEMD**	**MMS**	**Adaptive**
Lena	13.97 dB	16.95 dB	16.43 dB	17.10 dB
Ellipses	16.82 dB	20.35 dB	19.78 dB	17.99 dB

Signal-to-noise ratio of the two test images using the original images without noise as a reference. The ”Unfiltered” column contains the SNR of the original images with Gaussian noise added. Similarly, IEMD column contains SNR after IEMD filtering, MMS is SNR after filtering with MMS and ”Adaptive” contains SNR after filtering with adaptive IEMD.

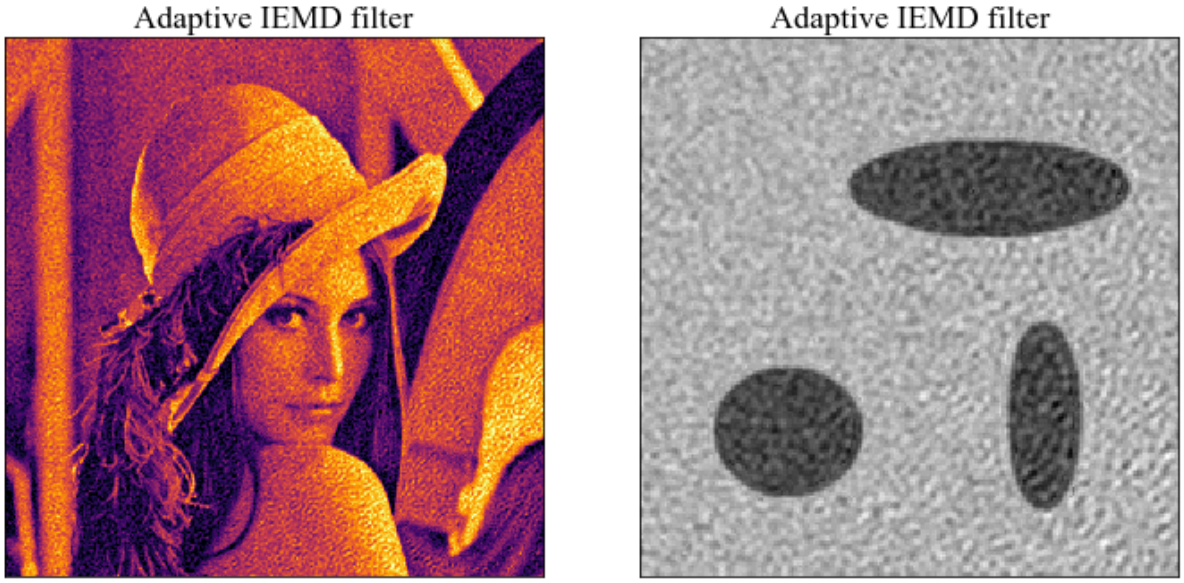

Figure 4.7: The two test images after being filtered with a adaptive IEMD filter.

Overall, these results already strongly indicate that the IEMD based filters performs inadequately when used to remove Gaussian noise. Looking at Table 4.1, we see that the improvement in SNR is minimal, especially when taking into regard the number of computations required for these filters to work.

4.1.4 Comparison to conventional methods

This section will compare conventional filters with the IEMD based filters presented in the previous section. The filters will be applied on images with different noise types added to them, so that the performance can be assessed for different conditions. The following noise types will be used to test the filters on:

- Gaussian noise
- Salt and Pepper noise (S&P)
- Speckle noise
- Uniform noise

These noise types appear in different situations depending on the technology and use case. Thus, taking a closer look at how these filters remove the different noise types is important if we want to determine the robustness of the filters. Gaussian noise is present in most analog processes

and it is the litmus test for any noise filter. S&P is less common in imaging application, but differs strongly from the other noise types and is therefore useful for testing the robustness of the filters. Speckle noise occurs in ultrasound imaging, and it is of high interest in the medical field to be able to filter this noise in a efficient manner. Uniform noise is less common, but given IEMD's susceptibility to variations in frequency and amplitude, this is an interesting case to study.

Five filters will be tested on the four noise types and the filtered output will be compared with each other. The five filters that are to be tested, are listed below:

- Uniform filter
- Median filter
- Conventional adaptive filter
- IEMD filter using MMS
- Adaptive filter using IEMD

The first two are simple filters that look at local properties in the image to remove noise. The conventional adaptive filter is based on the theory presented in Section 2.1.2, and is very similar to the IEMD based adaptive filter. The last two filters are the IEMD based filters that were tested in the previous section.

An extensive comparison chart is shown for the Ellipses image in Figure A.8. All images filtered with the different filters are compared to each other, giving an overview of the filter performance. The first column contains the input reference images with their corresponding noise types. Going right, column for column, the first filter we see in the chart is the uniform filter. It is probably the filter that is able to remove the most noise, but this comes at the price of a significant blurring of the details. It handles the S&P noise adequately without adding any strongly visible artifacts. The median filter arguably performs better though, and it mostly removes all the noise while keeping the image details sharp. It removes the S&P noise almost perfectly. The adaptive filter also performs well, and is able to conserve even more of the sharpness than the median filter. That being said, it is not able to remove all of the Gaussian noise and also fails to adequately remove the S&P noise. The first IEMD filter in the chart does not filter the noise in any way as well that the conventional filters. The edges are sharp but the introduction of the "wave" pattern (or failure to remove the low frequency part of the noise) completely diminishes the quality of the image. The filter works best on the uniform and speckle noise, while completely failing to remove the S&P noise. The adaptive IEMD filter behaves similarly, but with a marginally better output. There is slightly less wave pattern and the edges are sharper. It is also more capable at removing the S&P noise than the MMS filter.

A similar comparison chart for the Lena image is shown in Figure A.9. The characteristics of the output images are mostly similar to that of the filtered Ellipses image, with a rather poor performance for the IEMD based filters. Both the adaptive IEMD filter and the MMS filter perform best on the image with speckle noise. On all the other noise types, the output of the

IEMD based filters, are of low quality with a prominent wave pattern that diminishes the image. Also here, we see that they fail to remove the S&P noise and instead end up introducing artifacts that in large part destroy the data.

Table 4.2: Signal-to-noise ratio for Ellipses image

Name	Gaussian	S&P	Speckle	Uniform
Input	16.82 dB	15.01 dB	19.98 dB	18.06 dB
Uniform	25.60 dB	24.39 dB	26.35 dB	25.99 dB
Median	27.23 dB	36.85 dB	30.74 dB	26.52 dB
Adap. conv.	24.87 dB	20.40 dB	27.36 dB	26.77 dB
IEMD with MMS	19.78 dB	15.34 dB	22.42 dB	20.88 dB
Adaptive IEMD	19.83 dB	17.75 dB	22.07 dB	20.67 dB

The Tables 4.2 and 4.3 contain the SNR for both the Ellipses image and the Lena image respectively, and do to a certain extent confirm the visual observations we have done. The median filter has the highest SNR both for the Ellipses image and the Lena image. Mostly, the SNR for the two IEMD based filters are significantly lower than any of the other filters. For the Ellipses image, the IEMD based filters have a rather substandard SNR that is much lower than even the uniform filter. This difference is not as big for the Lena image, where the difference is no more than about 5 dB. In comparison, the Ellipse image have a difference that in many cases is more than 7 dB

Table 4.3: Signal-to-noise ratio for Lena image

Name	Gaussian	S&P	Speckle	Uniform
Input	13.97 dB	12.28 dB	20.00 dB	15.32 dB
Uniform	21.29 dB	20.44 dB	22.04 dB	21.45 dB
Median	21.38 dB	23.85 dB	23.09 dB	21.06 dB
Adap. conv.	20.09 dB	17.47 dB	21.76 dB	21.12 dB
IEMD with MMS	16.43 dB	13.46 dB	21.20 dB	17.13 dB
Adaptive IEMD	17.10 dB	16.48 dB	20.78 dB	17.36 dB

4.1.5 Speed and resource requirements

The IEMD method is extremely resource heavy, and so looking at the performance of the methods involved in the filtering, is very relevant for assessing their feasibility in practice. Even for smaller images of sizes less that 100×100 pixels, the processing time can be significant. This means that the filter cannot, with the current implementation, be used for filtering in real-time.

To get a better understanding of its limitations, this section will do a more thorough assessment of the performance of the IEMD filter.

The processing time requirement for extracting 1 IMF, from the Ellipses image with added Gaussian noise, for different pixel dimensions is shown in Figure 4.8. The processing time has a quadratic growth in correspondence with the quadratic growth of pixel area. This is a problem, since the time requirement will have such a steep increase, that it will render the filter useless after a relatively small increase in image size. As one can see in the plot, already at 250×250 pixels, the processing time has surpassed 1 minute. At 500×500 pixels the processing time is over 4 minutes. It is worth noting that the plot tests fairly small image sizes. Even at 500×500 pixels the images can be categorized as very small, and taking into regard that photos from smartphones can have dimensions of 3000×3000, there is no doubt that this is an issue.

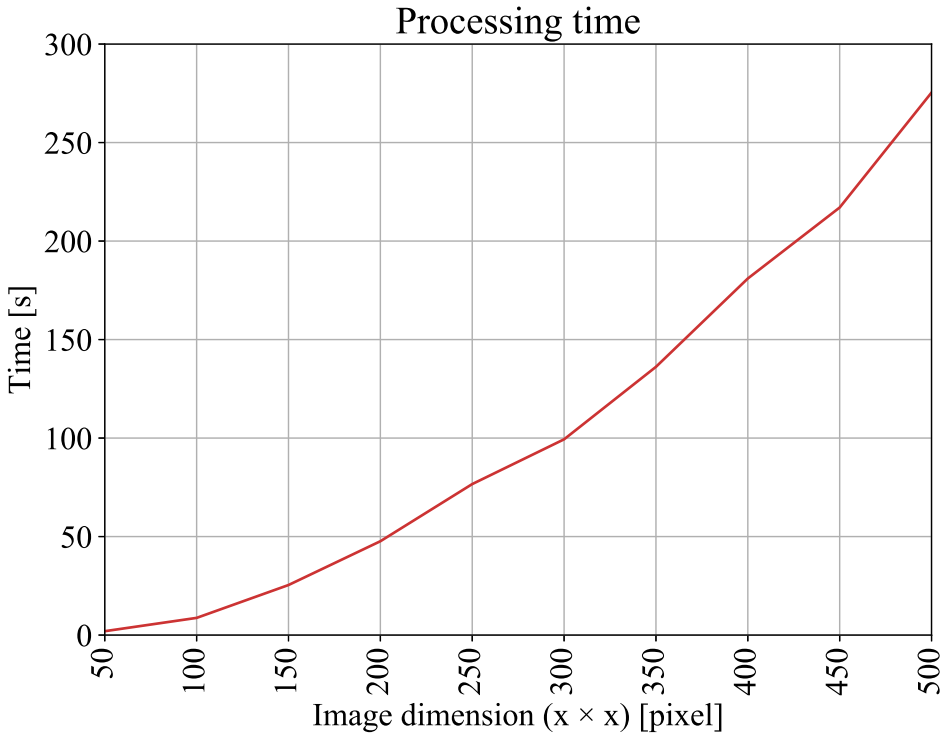

Figure 4.8: Processing time of different image sizes using IEMD to create 1 IMF, using 5 siftings.

The processing time relative to the number of IMFs extracted from the image, is plotted in Figure 4.9. This plot shows a weak semi-linear increase in processing time from 1 to 8 IMFs. This trend flattens out as more of the image is decomposed. As the blue line indicates, the processing time for each individual IMF decreases substantially after the first IMF, and for each following IMF this decreases further. This happens because there is a decrease in extrema points in the residue, as the higher frequency modes are removed from the image. Fewer extrema points

means that the linear equation for the spline construction becomes simpler and faster to solve. Based on this plot, we know that at least the cost of extracting additional IMFs after the first or second one, is not very high. Even though the IEMD algorithm is very slow, there is without doubt, a potential for further improvements in performance. Although the implementation of the ALGLIB library for spline creation is an optimization that greatly improves performance, building a dedicated solution would probably shave of some additional processing time.

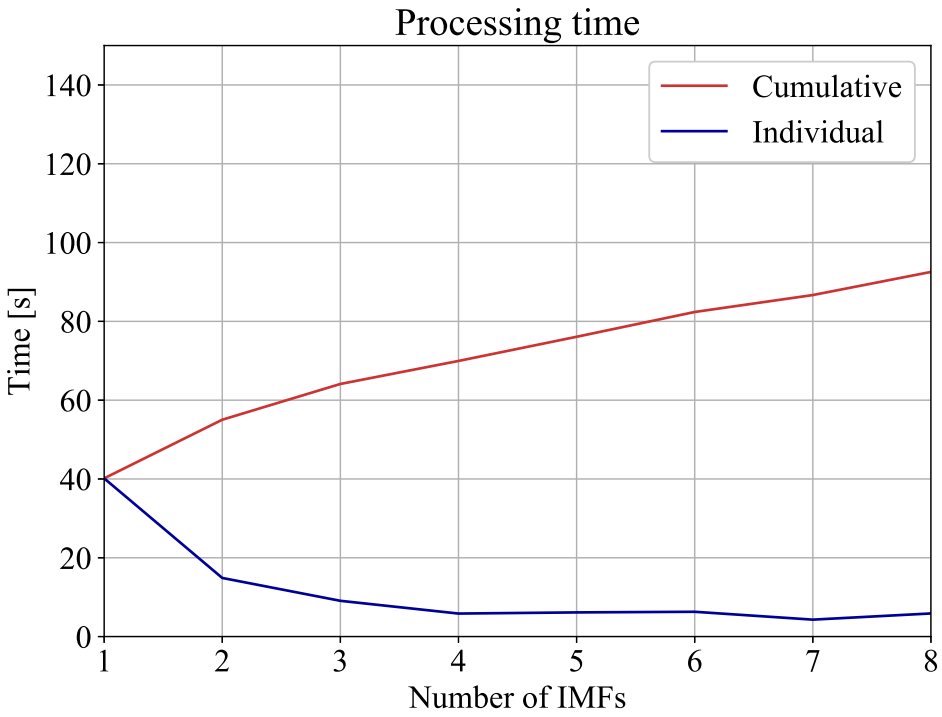

Figure 4.9: Processing time for additional IMFs extracted.

4.2 Iceberg classification

Most of the analysis done so far in this thesis has mostly been based on visual inspection of the filtered output. While this is undoubtedly an useful and important way to assess the performance of the filtering methods, it leaves some room for subjective interpretation in regards to whether the filters truly work or not. The SNR does give us an objective number on how much signal-to-noise there is in the image, but it does not say anything about how valuable the information contained in the image really is. The iceberg dataset classification problem is a great way to more objectively assess whether the IEMD based filters actually work or not. By using a neural network for classification, human bias is left out of the process as much as possible. If the filters manage to better highlight the useful data, then the classification accuracy will also improve.

That way, the classification accuracy can to some extent be used as a measure of performance of the filter.

The classification results are shown in Table 4.4. The quality of the predictions are shown by the test accuracy and test loss. The accuracy gives us the percentage of the dataset that the classifier guessed correct. Since the test set is almost evenly split between the two classes (753 icebergs vs. 851 boats), accuracy is a suitable measure of performance in this specific case. Additionally, the test loss is calculated, this value gives the total error of the predictions and can give additional insight into how the classifier performs.

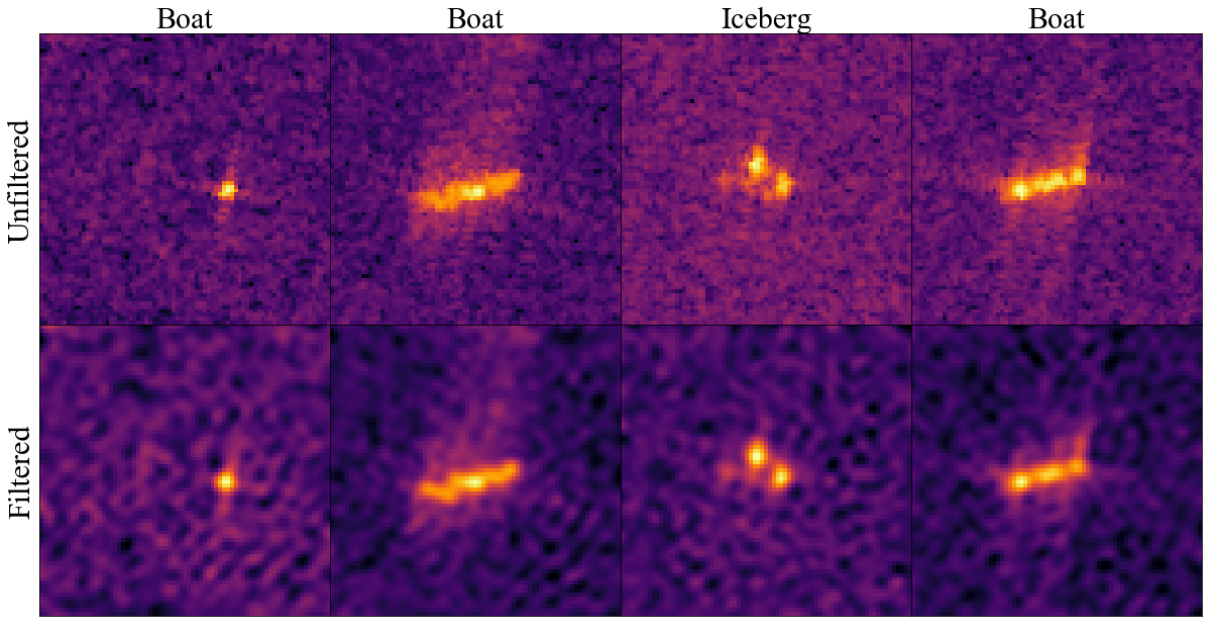

Figure 4.10: A comparison of a subset of the unfiltered and filtered dataset.

Looking at the table, we see that the classification accuracy of the filtered dataset is just barely above 50%, which means that the network is not able to converge towards a solution. Combining the unfiltered and the filtered set and training the network on both, does improve the accuracy slightly. This could indicate that the filter is able to highlight some useful information that is not clearly visible when training only with the original dataset. The difference in classification accuracy with and without the filtered dataset, is so small that one cannot decisively conclude one way or another when the dataset contains so few images. That being said, this improvement in accuracy is interesting and is a sign that there could be more to it.

Table 4.4: Classification accuracy of iceberg dataset

Dataset	Test accuracy	Test loss
Unfiltered	88.53%	0.28
IEMD & MMS	52.87%	7.60
Unfiltered + IEMD & MMS	89.28%	0.26

It is not hard to understand why the network is having issues learning if one takes a look at some of the filtered images from the data set. Figure 4.10 contains a small subset comparison of the filtered images and their original unfiltered counterpart. As mentioned earlier, we see that high frequency noise is removed, but also many important details disappear that are essential for accurate classification. An example is the light rays reflected off the reflective surfaces of the boats. These can be seen at the dotted lines radiating from the objects, and are important details that indicate that the image contains a boat. These features are not visible in the filtered images and this will have an impact on the classification accuracy. Furthermore, there is a significant blurring which leads to a loss of edge details, making boats and icebergs look more similar. By inspecting the filtered images visually, it is hard to distinguish the class of even the most obvious examples. All filtering will to some degree remove useful information that otherwise could be used by the classifier, and so, a filter is most useful as a complement to the original dataset, highlighting features that would not be visible in its original format.

4.3 Medical ultrasound images

Ultrasound images are known to contain speckle noise, which can make it hard to distinguish important features. In this section an attempt is made to remove speckle noise using IEMD with MMS, with the goal to improve the quality and details of medical ultrasound images.

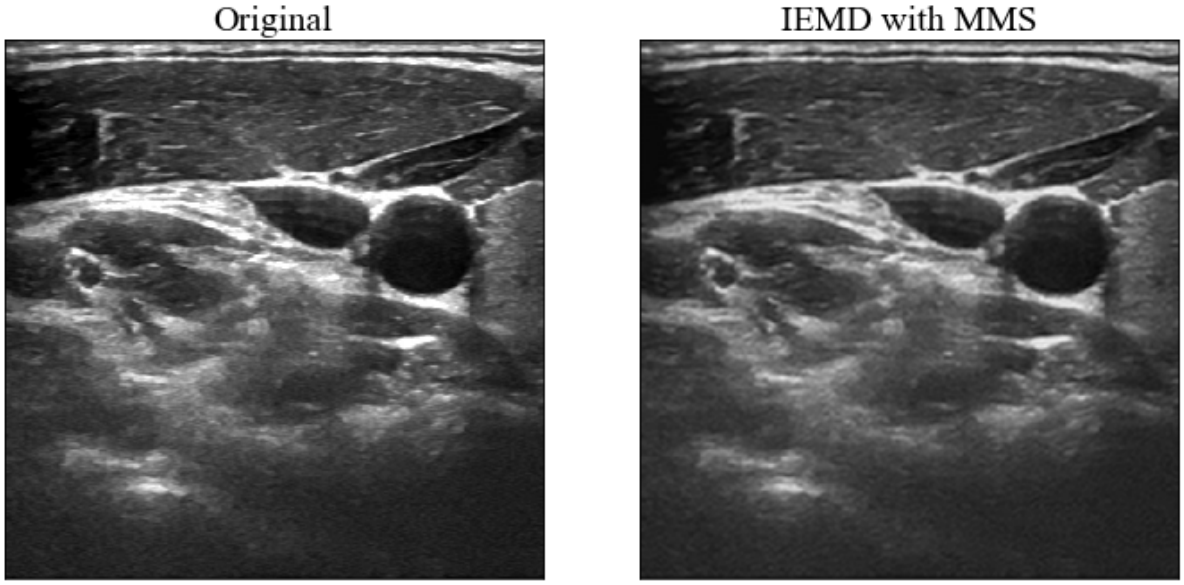

Figure 4.11: An attempted filtering of a medical ultrasound image using IEMD with MMS.

A decomposition of two sample ultrasound images are shown in Figure A.10 and A.11, where

we can see their corresponding IMFs and residue. Again, we see that the sharpest outlines are included in the first IMFs, while the more underlying structures appear in the following IMFs. Interestingly enough, the most important details seem to be contained in the second IMF. This is especially the case for Figure A.10, where we see that the main outline of the blood vessel (the dark circular area) is most clearly visible in the second IMF. This means that some high frequency components are included in the first IMF that are not image details. By the looks of it, these high frequency components are a combination of speckle noise and some of the sharpest details in the image.

An attempt at filtering the images with IEMD and MMS is shown in Figure 4.11. We see that there is very little difference between the original image and the filtered image. Looking at the image, it is hard to say whether there is any real gain from the filter. The image is not that blurry to start with and the little noise that is removed, comes at the cost of slightly diminished image details. This can more specifically be observed in the region of muscular tissue, which can be seen as a larger darker area in the upper part of the image. These areas contain details of white ”dots” or ”stripes” in the muscles which are not speckle noise. After the image is filtered, these details are either completely removed or partially blurred out, something that is not ideal.

Similar procedure is performed in Figure 4.12 with similar results. We see some reduction in noise, but not to any significant degree. Another side effect of the IEMD based filters, and it is even more visible in this example, is that there is a loss of contrast. The light colors are darker and the dark colors are lighter. This most likely contributes to diminishing the visibility of the details and also the amount of information that is retained in the output.

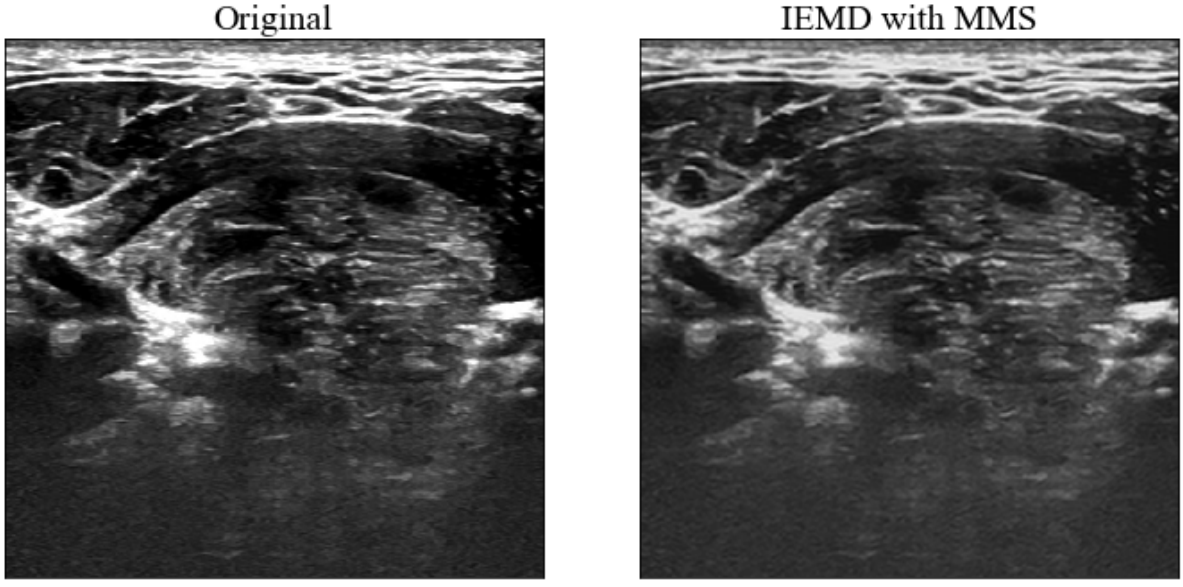

Figure 4.12: An attempted filtering of a medical ultrasound image using IEMD with MMS.

All in all, applying IEMD with MMS on the ultrasound images did not really make that big of a difference. Compared to the examples presented in earlier sections, the output for the

ultrasound image mostly did not change, pre- and post-filtering. It is hard to pinpoint exactly why that is, there seems to be some forms of high frequency modes in the images that are placed further away in the frequency spectrum than the overall image details. This could be seen in the decompositions, where the first IMF contained surprisingly few image details, even when not using MMS to separate the noise. Despite this, removing the high frequency components did not really improve the quality of the images.

CHAPTER 5

CONCLUSION

REVISITING the research questions presented in the introduction, we will briefly discuss to what degree we were able to answer them or not. If we take a look at **RQ2** first, the classification of the iceberg dataset was the selected way of assessing whether IEMD and MMS can be used to improve the feature extraction and classification of noisy images. Trying to train the neural network on only the filtered dataset, resulted in the network not being able to learn the features needed to separate the classes. Combining the filtered and the unfiltered set improved the accuracy slightly, indicating that the filter is able to highlight information that is not clearly visible in the unfiltered images. Visual inspection of the filtered images showed that many of the important details get filtered out by the IEMD process, resulting in a rather illegible output. It is arguable, whether the small improvement in accuracy when combining the sets, is substantial enough to say that the filter actually works. This does not necessarily mean that the filter is useless for this task, but it is hard to make a confident claim about its performance when the improvement in accuracy is so minuscule.

Research question **RQ1** focused on whether IEMD can be used to remove noise from images. IEMD by it self, proved to not work well enough to be useful as a noise filter. Implementing MMS together with IEMD, did prove to have a positive impact on the IEMD filtering capabilities. Similarly, the results showed that the adaptive filter also was a step up from only using IEMD. The adaptive filter worked especially well for preserving the edge details, while removing much of the high frequency noise. But while these methods improve the IEMD filter significantly, it is not enough to make them good. The biggest weakness of the IEMD based filters is that they either introduce, or fail to remove, the ”wave” pattern from the images, which leaves the output in very poor condition. Finding a way to remove this ”wave” pattern without diminishing the original image, would without a doubt, make the IEMD based filters much more powerful. This should be a prioritized research topic for anyone who would want to continue this work, as this is where one can gain most in performance. This could be a challenge though, since the residue ”wave” pattern seems to be an inherit problem of the EMD method and it could mean that this is a limitation which cannot be overcome.

The source of the issue can be boiled down to the fact that in general, images contain regions where the data is semi-monotonous. These areas of the image that contain little variance are handled very poorly by IEMD. This problem is not unique for IEMD, and also occurs in 1D. It is best demonstrated with an example, as shown in Figure 5.1. As we can see, the square wave has multiple individual monotonous parts that will, after being filtered, still have large

variations in these monotonous areas. While the residue noise in is not extremely prominent in 1D, perturbations like these are very visible in images. It is also important to remember that the errors introduced in the interpolation, probably amplifies this residue noise.

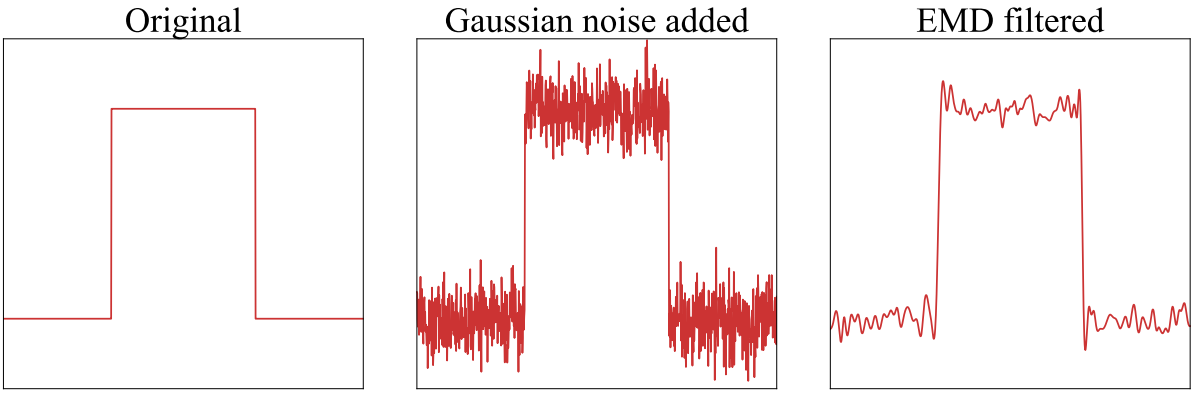

Figure 5.1: Left: Square wave as original signal. Middle: Square wave with Gaussian noise added. Right: Filtered noisy square wave by removing the first 3 IMFs.

Because of the computational resources needed to create the 2D splines, accuracy has to be sacrificed in exchange for performance for the filter to be usable. This loss of accuracy propagates into the following decompositions, resulting in an output where artifacts are introduced as a bi-product of the interpolation. The computational requirements also limit how large the images can be, for the images to be filtered using IEMD. Improving the interpolation process for 2D data, could thus potentially improve the performance of the IEMD based filters both in terms of speed and quality, and should be a focus area for future research.

IEMD as a filtering method is, with its current implementation, limited in its usefulness. It is very simply not able to filter out image noise in an efficient manner. Not only does the filter fail to remove noise, in most cases it also adds artifacts to the image that results in an output that is in worse condition than the original input. That being said, it does not mean that the research presented in this thesis does not have any value. This thesis contains one of the most thorough reviews of the application and usage of EMD in 2D. It explores methods like MMS and adaptive IEMD filters, and applies them in ways that have never been done in 2D before. This thesis gives a better understanding of the limits, strengths and weaknesses of EMD when applied to 2D data, and sets the foundation for those who wish to explore this topic further.

REFERENCES

[1] Devesh Maheshwari. *Keras Model for Beginners (0.210 on LB)+EDA+R&D*. URL: https://www.kaggle.com/devm2024/keras-model-for-beginners-0-210-on-lb-eda-r-d.

[2] Norden E. Huang et al. "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 454.1971 (1998), pp. 903–995. ISSN: 1364-5021. DOI: 10.1098/rspa.1998.0193. URL: http://rspa.royalsocietypublishing.org/content/454/1971/903.

[3] G. Rilling and P. Flandrin. "One or Two Frequencies? The Empirical Mode Decomposition Answers". In: *IEEE Transactions on Signal Processing* 56.1 (Jan. 2008), pp. 85–95. ISSN: 1053-587X. DOI: 10.1109/TSP.2007.906771.

[4] Anna Linderhed. "Image Empirical Mode Decomposition: A new tool for image processing". In: *Advances in Adaptive Data Analysis* 01.02 (2009), pp. 265–294. DOI: 10.1142/S1793536909000138. URL: http://www.worldscientific.com/doi/abs/10.1142/S1793536909000138.

[5] Srishti Sondele and Indu Saini. "Classification of Mammograms Using Bidimensional Empirical Mode Decomposition Based Features and Artificial Neural Network". In: *International Journal of Bio-Science and Bio-Technology* 5.6 (2013), pp. 171–180. DOI: 10.14257/ijbsbt.2013.5.6.18.

[6] Zhuofu Liu and Zhongming Luo. "Bidimensional empirical mode decomposition for noise reduction in sonar images". In: *International Forum on Strategic Technology 2010*. Oct. 2010, pp. 225–229. DOI: 10.1109/IFOST.2010.5668033.

[7] Y. Pei, Y. Wu, and D. Jia. "Image denoising based on Bidimensional Empirical Mode Decomposition". In: *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*. Aug. 2011, pp. 1122–1125. DOI: 10.1109/MEC.2011.6025664.

[8] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Dorling Kindersley, 2014.

[9] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[10] Joachim Blaafjell Holwech. "Implementation of a Brain-Computer-Interface Using the OpenBCI Ultracortex". In: (Dec. 2017).

[11] J.C Nunes et al. "Image analysis by bidimensional empirical mode decomposition". In: *Image and Vision Computing* 21.12 (2003), pp. 1019–1026. ISSN: 0262-8856. DOI: 10.

1016/S0262-8856(03)00094-5. URL: http://www.sciencedirect.com/science/article/pii/S0262885603000945.

[12] Zhongxuan Liu, Hongjian Wang, and Silong Peng. "Texture segmentation using directional empirical mode decomposition". In: *Image Processing, 2004. ICIP '04. 2004 International Conference on*. Vol. 1. Oct. 2004, 279–282 Vol. 1. DOI: 10.1109/ICIP.2004.1418744.

[13] Anna Linderhed. *Adaptive image compression with wavelet packets and empirical mode decomposition*. Department of Electrical Engineering, Linköing University, 2004.

[14] David Eberly. *Thin-Plate Splines*. Mar. 1996. URL: https://www.geometrictools.com/Documentation/ThinPlateSplines.pdf.

[15] O. B. Fosso and M. Molinas. "Method for Mode Mixing Separation in Empirical Mode Decomposition". In: *ArXiv e-prints* (Sept. 2017).

[16] Ryan Deering and James F. Kaiser. "The Use of a Masking Signal to Improve Empirical Mode Decomposition". In: *Proceedings. (ICASSP 05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.* (2005). DOI: 10.1109/icassp.2005.1416051.

[17] Yaser Abu-Mostafa. *Lecture 16 - Radial Basis Functions*. May 2012. URL: https://www.youtube.com/watch?v=O8CfrnOPtLc&t.

[18] *Statoil/C-CORE Iceberg Classifier Challenge*. Oct. 2017. URL: https://www.kaggle.com/c/statoil-iceberg-classifier-challenge.

[19] ALGLIB. *Fast RBF interpolation/fitting*. URL: http://www.alglib.net/interpolation/fastrbf.php#header3.

[20] Eric Drowell. *Know Thy Complexities!* URL: http://bigocheatsheet.com/.

[21] Gabriel Rilling, Patrick Flandrin, and Paulo Gonçalves. "On empirical mode decomposition and its algorithms". In: *Proceedings of IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing NSIP-03*. June 2003. URL: https://hal.inria.fr/inria-00570628.

APPENDIX A

IMAGES AND CHARTS

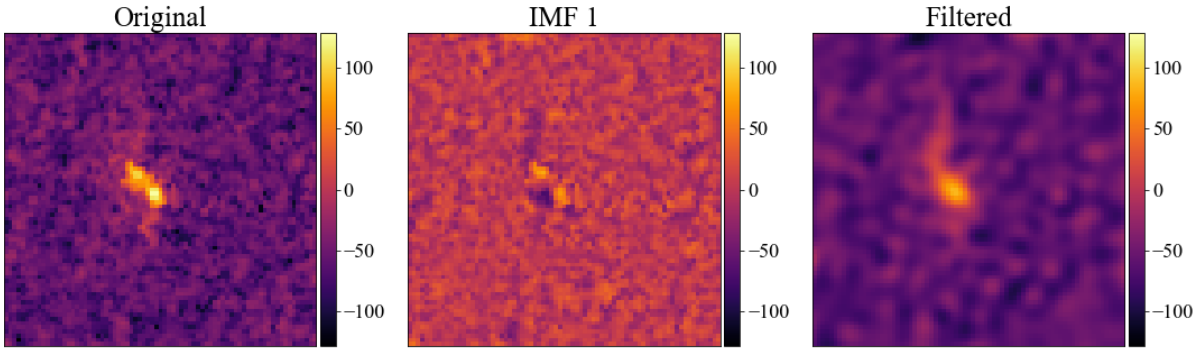

Figure A.1: Original input of a boat, its first IMF and the original with the first IMF subtracted from the image.

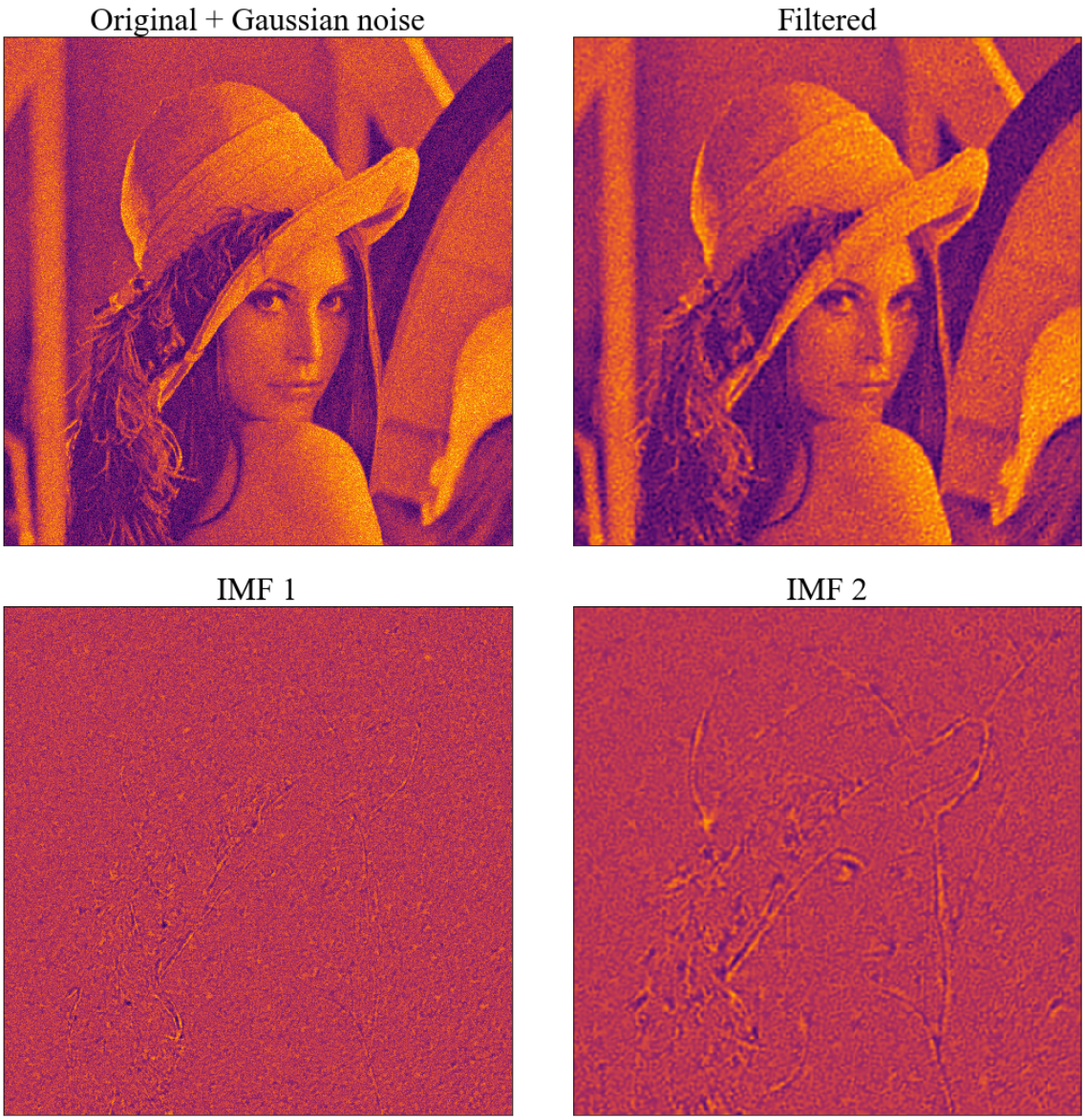

Figure A.2: The Lena image with Gaussian noise added to it, its first IMF and the original with the first IMF subtracted from the image. The images are all gray scale but presented with a different color map for improved visibility of details.

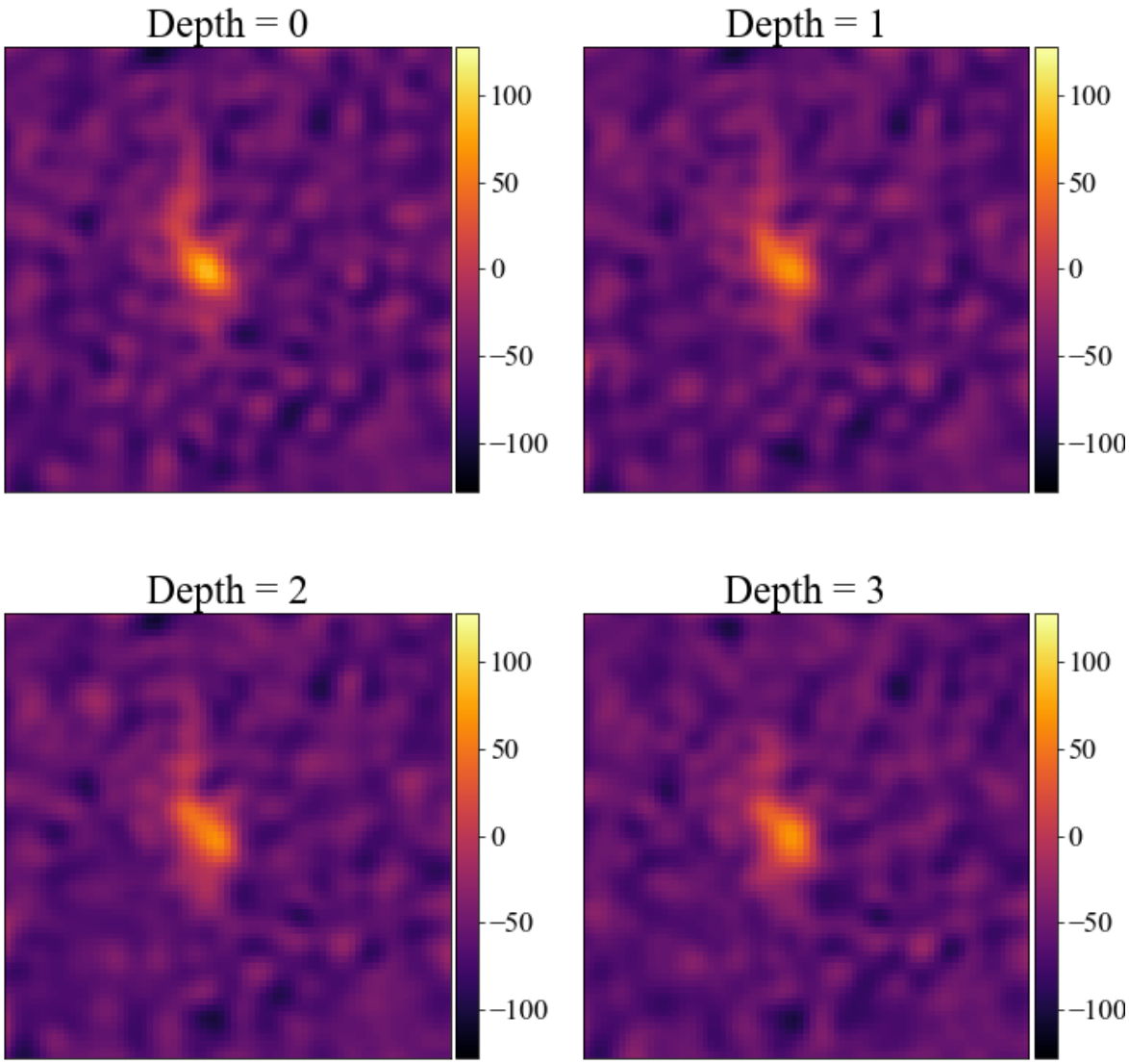

Figure A.3: Satellite image of a boat with IEMD filtering applied with different depth values.

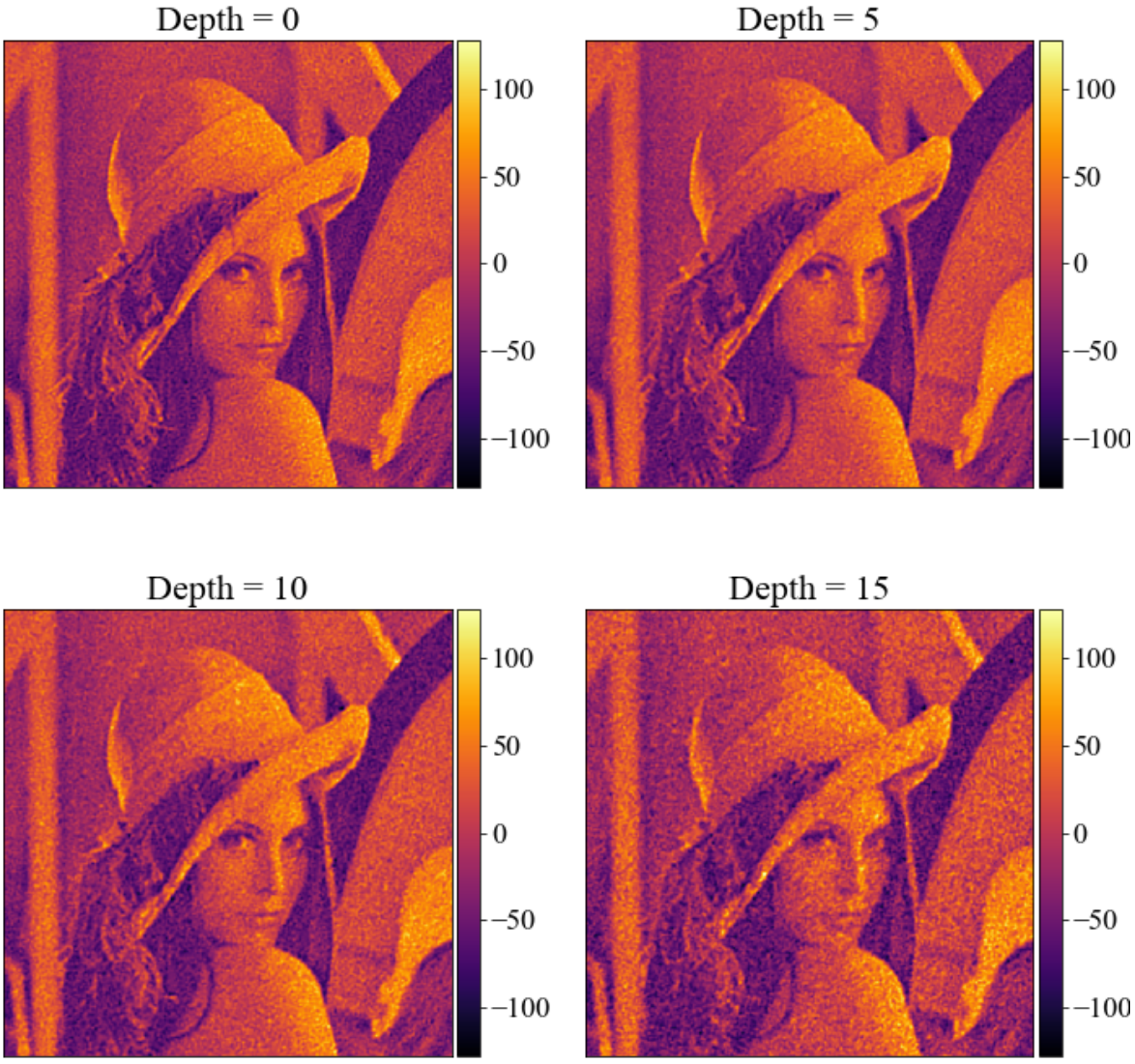

Figure A.4: Lena image with IEMD filtering applied with different depth values.

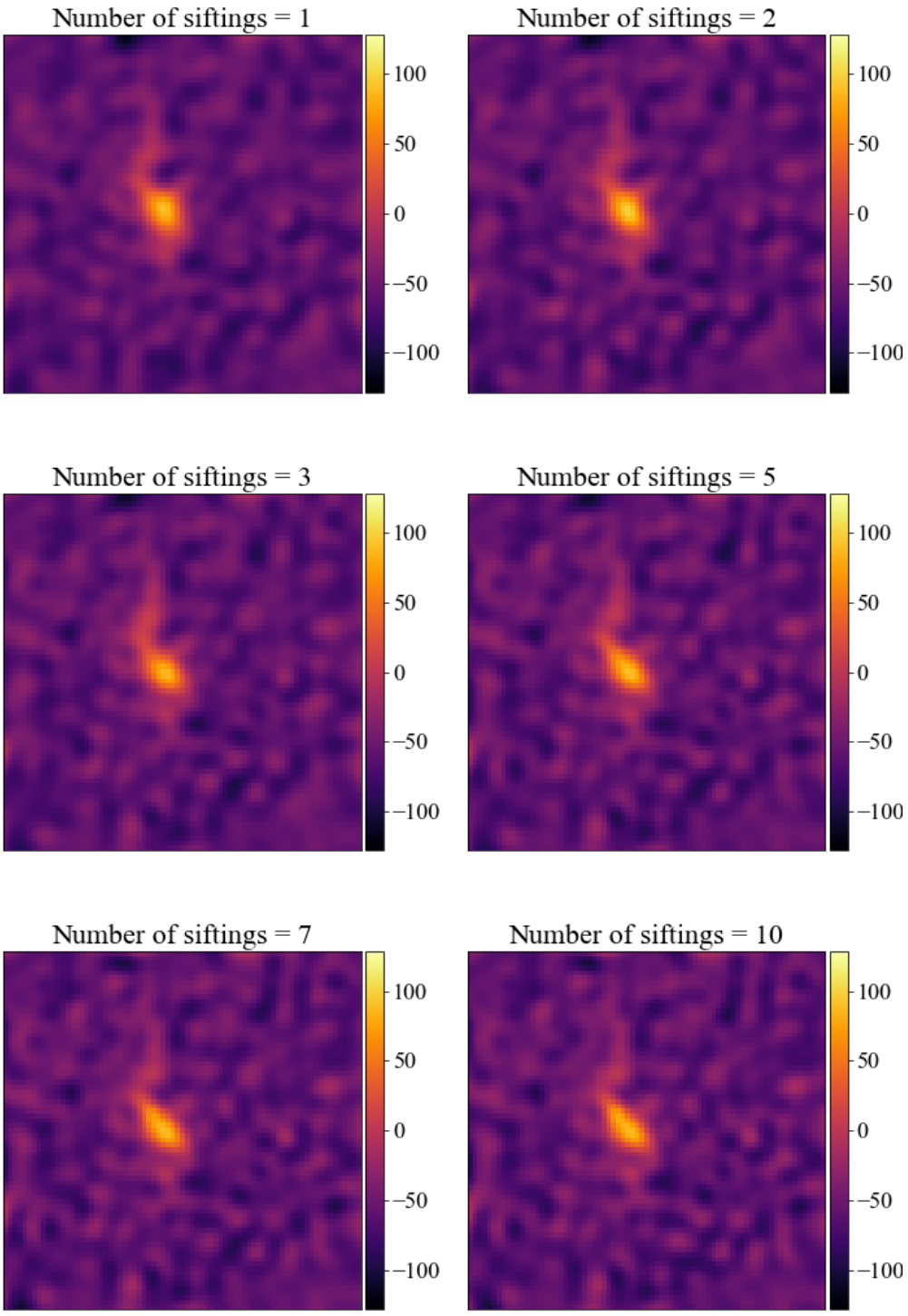

Figure A.5: Satellite image of a boat with IEMD filtering applied with different number of siftings.

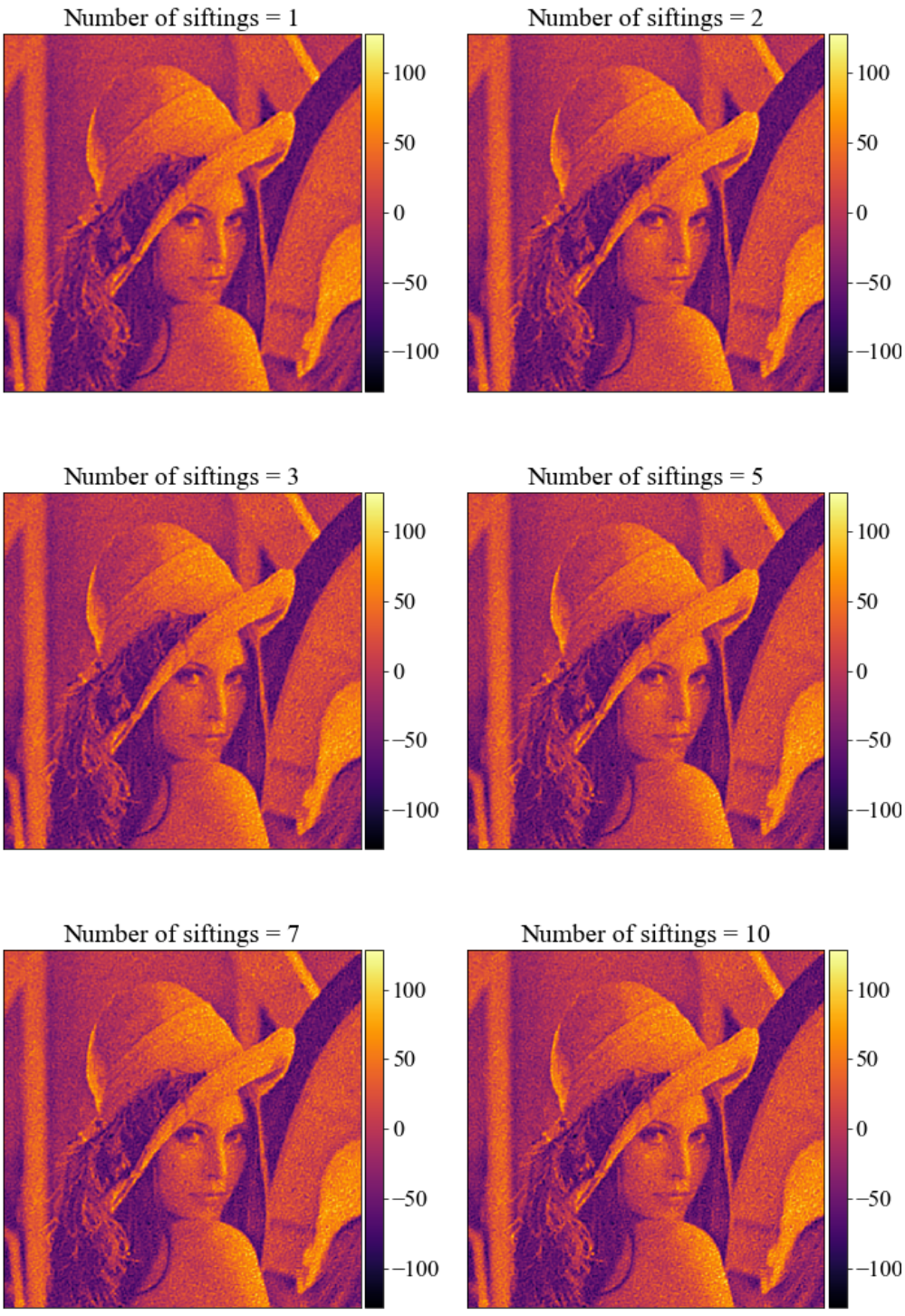

Figure A.6: Lena image with IEMD filtering applied with different number of siftings.

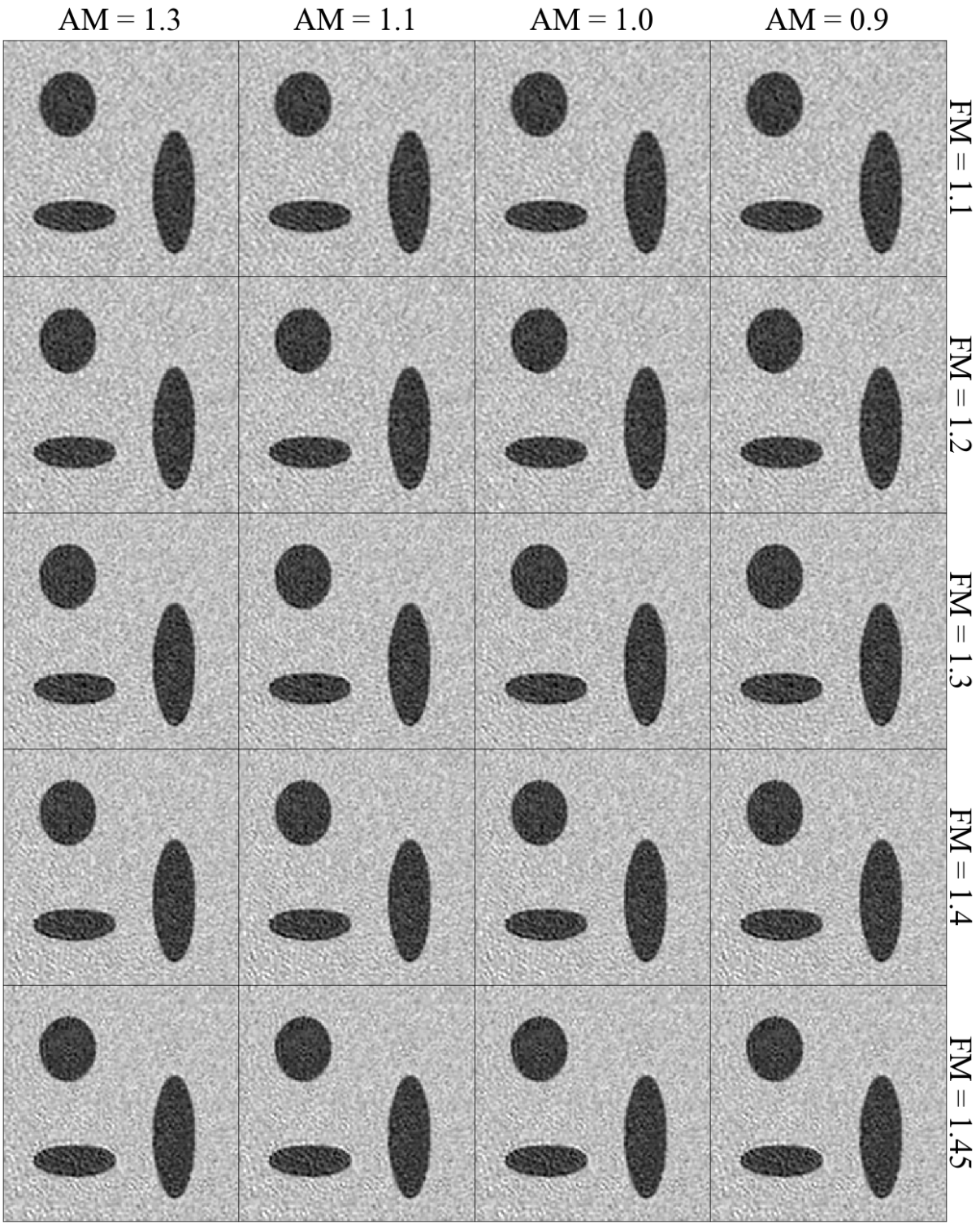

Figure A.7: Comparison chart of the filtered Ellipses image with different modifiers for frequency and amplitude of the masking signal.

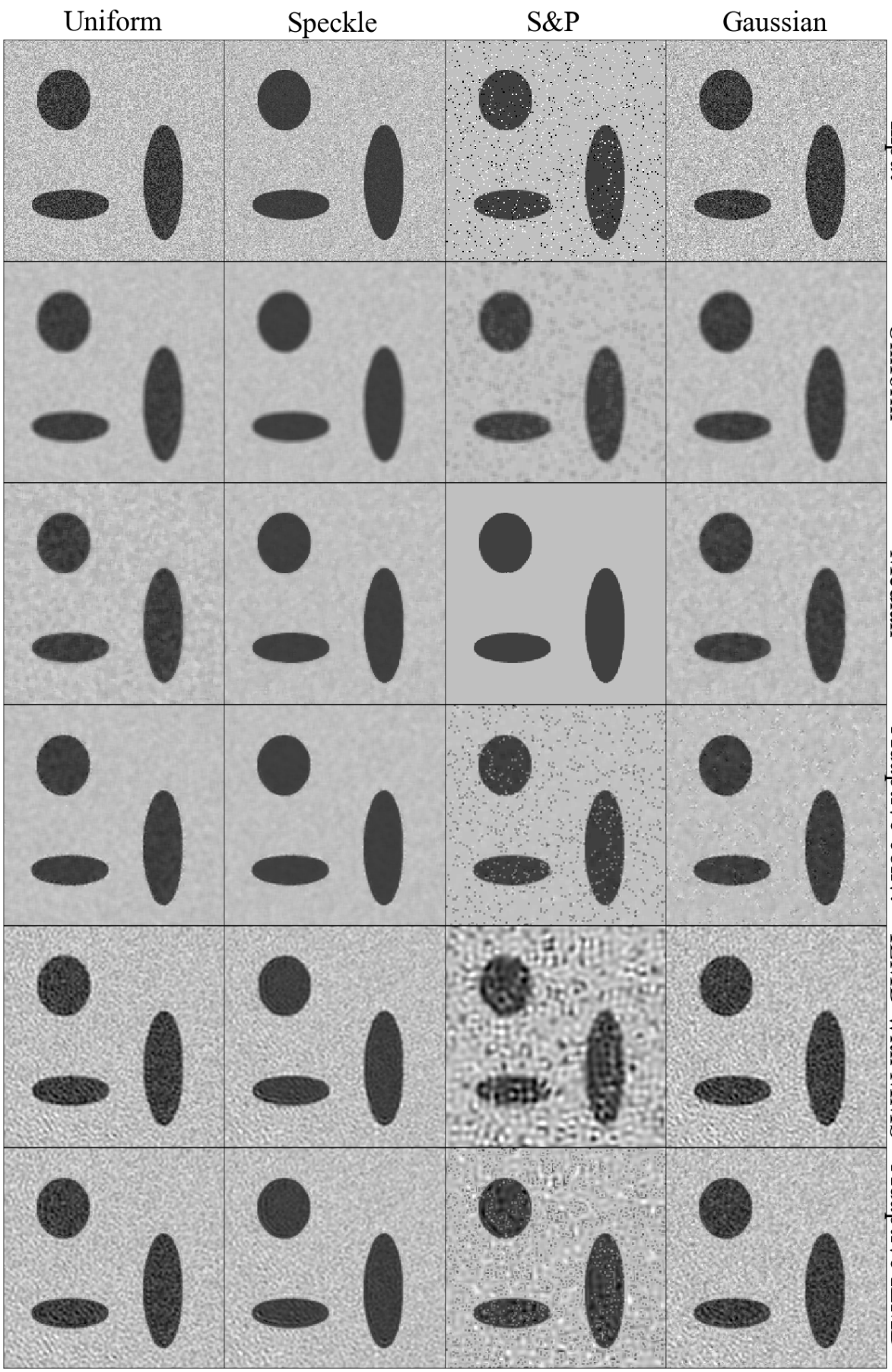

Figure A.8: Comparison of the different filter types for various noise types for the Ellipse image. The short side is noise type, while the long side is filter type.

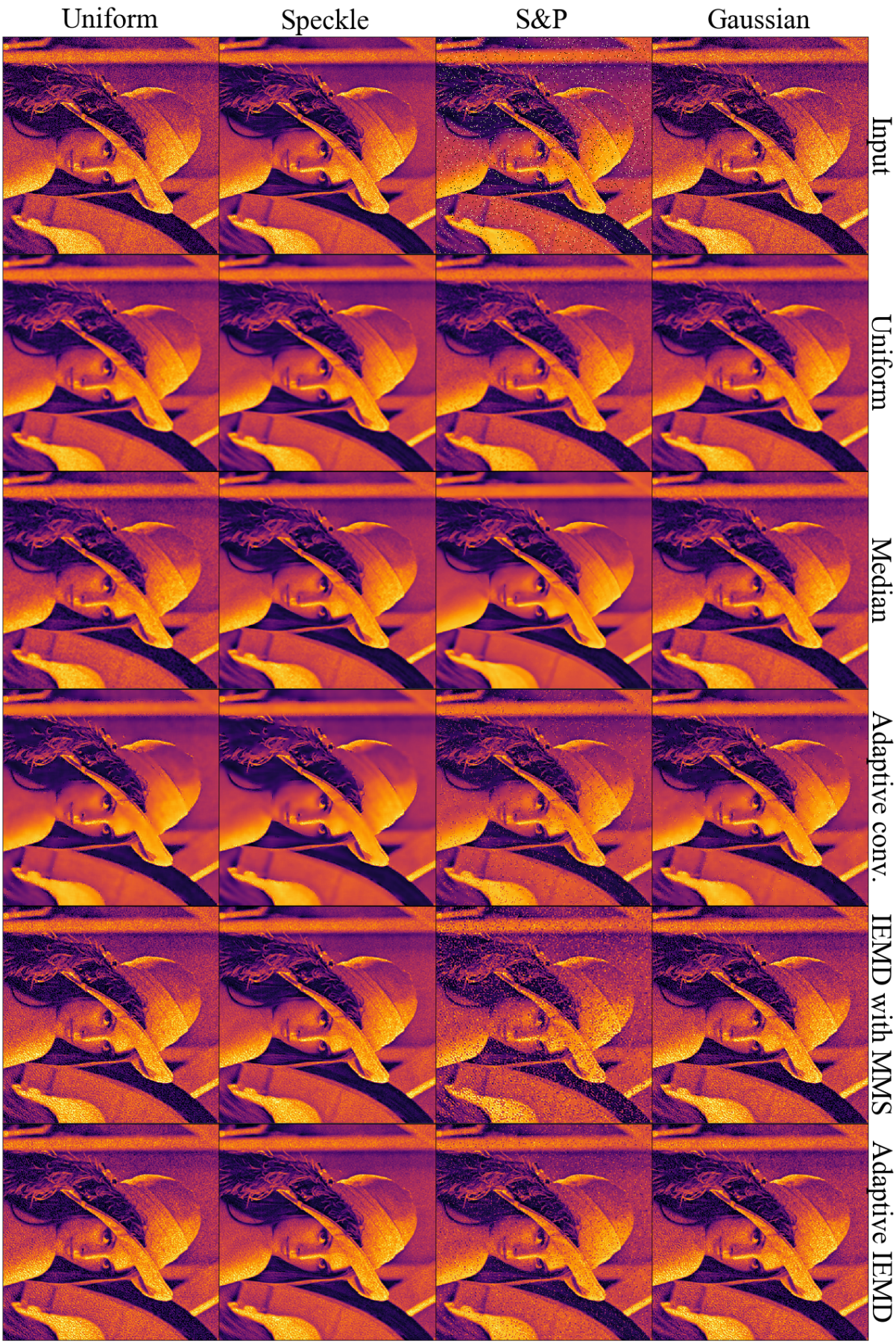

Figure A.9: Comparison of the different filter types for various noise types for the Lena image. The short side is noise type, while the long side is filter type.

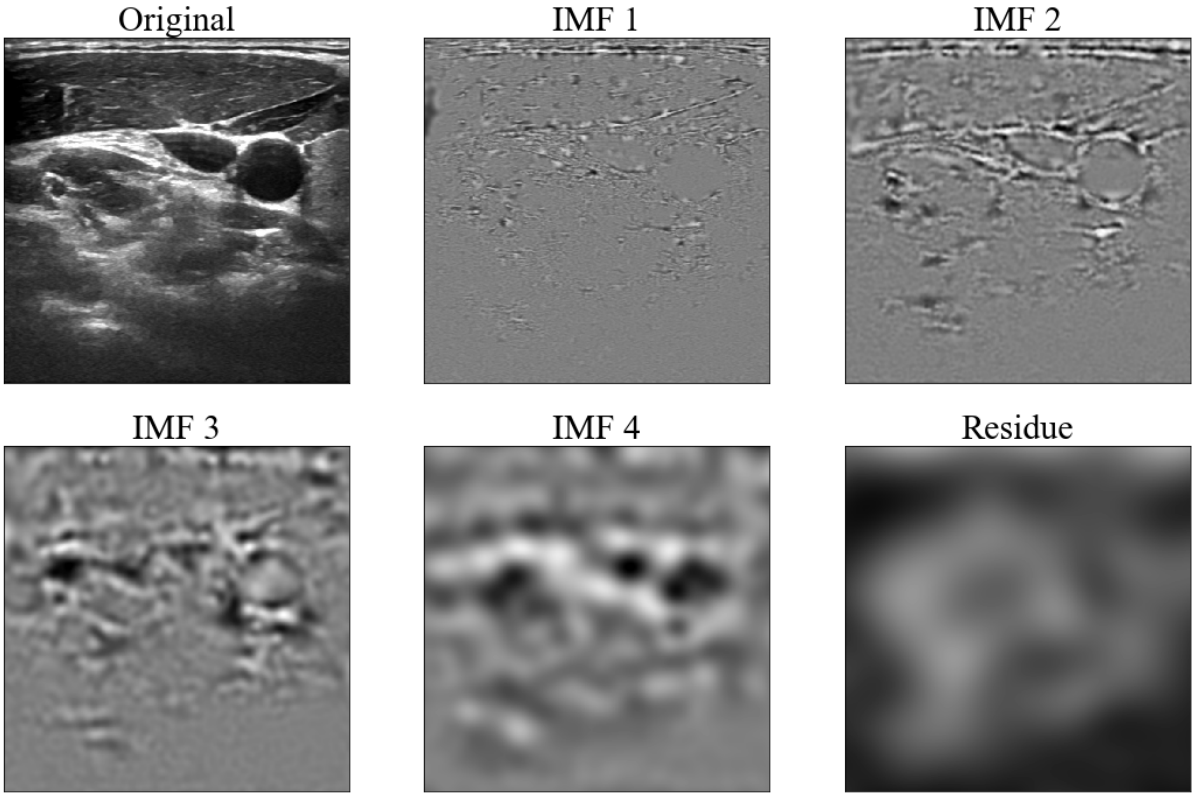

Figure A.10: An ultrasound image and its corresponding IMFs and residue

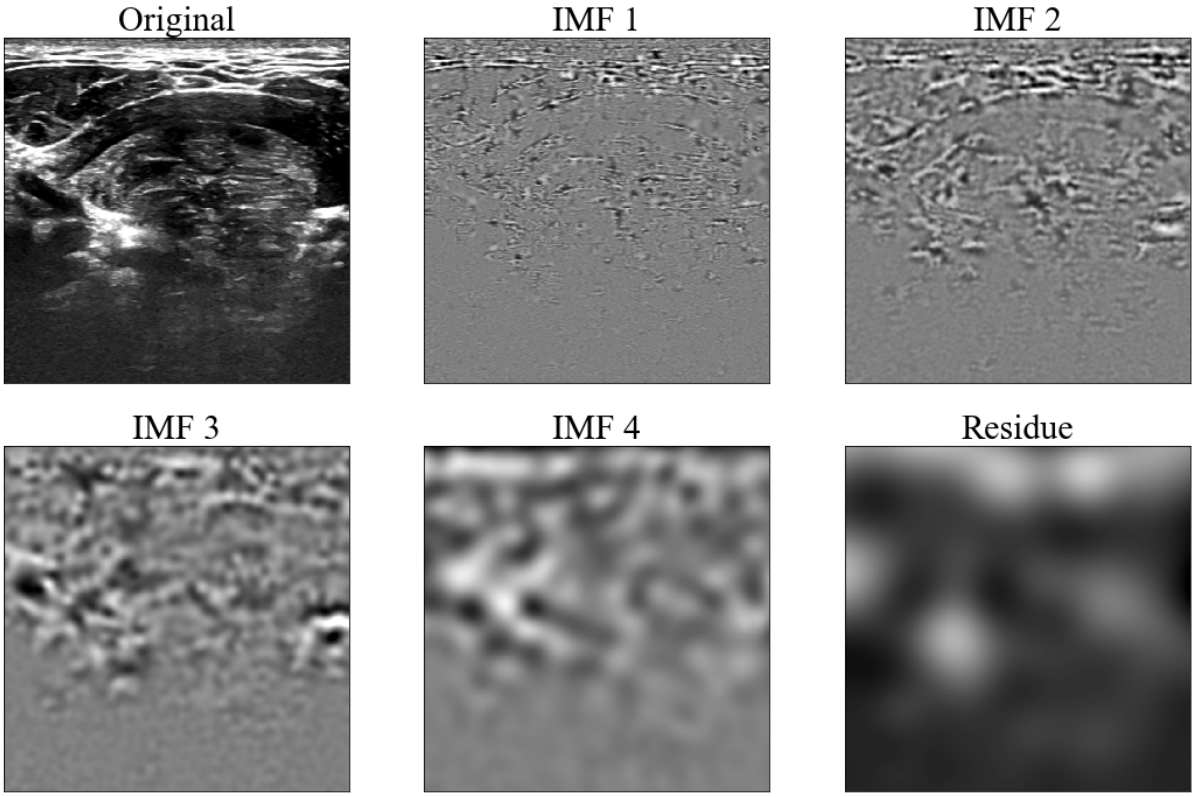

Figure A.11: An ultrasound image and its corresponding IMFs and residue