2009

# Video Super Resolution

Michael Chukwu
*University of Windsor*

# VIDEO SUPER RESOLUTION

by

**Michael Chukwu**

**A Thesis**

**submitted to the Faculty of Graduate Studies through Electrical Engineering**

**in Partial Fulfillment of the Requirements for the Degree of Master of Applied Science at the University of Windsor**

**Windsor, Ontario, Canada**

**2009**

**VIDEO SUPER RESOLUTION**

**By**

**MICHAEL CHUKWU**

**APPROVED BY:**

---

Dr. Bubaker Boufama

School of Computer Science

---

Dr. Majid Ahmadi

Department of Electrical and Computer Engineering

---

Dr. Maher Sid-Ahmed, Advisor

Department of Electrical and Computer Engineering

---

Dr. M. Mirhassani, Chair of the Defense

Department of Electrical and Computer Engineering

September 18, 2009

## AUTHORS' DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

## ABSTRACT

Advances in digital signal processing technology have created a wide variety of video rendering devices from mobile phones and portable digital assistants to desktop computers and high definition television. This has resulted in wide diversity of video content with spatial and temporal properties fitting into their intended rendering devices. However the sheer ubiquity of video content creation and distribution mechanisms has effectively blurred the classification line resulting in the need for interchangeable rendering of video content across devices of varying spatio-temporal properties. This results in a need for efficient and effective conversion techniques; mostly to increase the resolution (referred to as super resolution) in-order to enhance quality of perception, user satisfaction and overall the utility of the video content.

## ACKNOWLEDGEMENTS

Special thanks and acknowledgements for the unparalleled advisory role and invaluable support of the research supervisor, Dr. Maher Sid-Ahmed, for piloting the research and providing the guidance for the work.

More so, special appreciation for the insights and inputs of the committee members, Dr. Boufama and Ahmadi, that provided the suitable appraisals and critic.

The funding support from Graduate Studies, University of Windsor in form in Graduate Assistantships and Scholarships is highly appreciated; the additional support from NSERC Research funds of Dr. Maher Sid-Ahmed is also appreciated.

Special gratitude to the graduate secretary Andria Ballo (nee Turner) for her unflinching support without which this work would not have been a success.

This is also to acknowledge the support and friendship of the department secretary Ms. Shelby Marchand, for responsive and great assistance.

# Table of Contents

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF ABBREVIATIONS

**SR**: Super Resolution

**HR**: Higher Resolution

**LR**: Lower Resolution

**PSNR**: Peak Signal to Noise Ratio

**RDWT**: Redundant Discrete Wavelet Transform

**DWT**: Discrete Wavelet Transform

**MRA**: Multi-Resolution Analysis

**Pixel**: picture elements

**MSE**: Mean Square Error

**MAD**: Mean Absolute Deviation

**MRME**: Multi-Resolution Motion Estimation

# CHAPTER 1: Introduction and Research Objective

## 1.0 Introduction

The capture and conversion of continuous time signal to digital samples (or discrete-time signals), is a core requirement for the current body of processing techniques available for digital electronics. The proximity and resemblance of the discrete-time signal to its analog counterpart, is measured by the sampling resolution. This determines the level of (and/or possibility for perfect) reconstruction of the original signal. However in the design of digital signal processing systems, the application requirement largely determines this feature as several constraints, limitations and design specifications constitute the central objective of such systems. This scenario is evident in the general classification of digital image and video processing devices according to the capture and rendering capabilities.

Meanwhile the deep penetration of this technology and the sheer ubiquity of its availability have created scenarios of interchangeable and multi-functional usage where for example images captured on humble mobile phones can end up being primary source material for gait recognition, this results in digital signal processing problems that require the upward conversion of the resolution of digitally sampled signal.

Moreover, digital video capture, distribution and utilization has become increasingly commonplace due to significant advances in camera technology, digital circuit integration and storage systems among a host of other factors. The pervasive deployment of the technology on diverse platforms with wide application requirements and varied use case scenarios has created a multi-platform creation and consumption chain where digital video from a broad spectrum of capture devices can be utilized on an equally wide variety of rendering devices.

The digital image processing devices generate images with specific spatial properties, lower resolution images are coarsely sampled version of the higher resolution ones, and are intended for devices with compatible display resolution.

However, usage scenarios and application requirement in the ubiquitous multimedia applications environment ultimately results in the need for increasing the spatial dimensions of an image. This need is further compounded by the fact that the lower resolution version of the image could be the only available media source, this inevitably results in an overarching constrain that requires the extraction of unavailable details, as the level of information available in images are statistically defined by their resolutions because the process of sampling is irreversible.

This presents a scenario where the only course of action for increasing the resolution of an image is the direct replication of the available information to the desired levels of spatial dimensions. This approach results in images with jagged edges and sharp discontinuities. The resulting images show visually disturbing artifacts and perceptually unnatural image edges lines that considerably distort the images. However the generated higher resolution output can be enhanced to remove these distortions by synchronizing the sharp edge gradients into a continuous and smooth curves exhibiting natural rhythm. The artifacts can also be eliminated by performing a similar action on the non edge portions of the image. This generally results in perceptually acceptable version of the image but confronts the problem of blurring. The post pixel replication actions intended for the removal of distortions also eliminates the variations of the pixels across the images, hence the generated higher spatial resolution image is blurred in comparison to the original lower resolution input. Several methods for increasing the resolutions of an image offer solutions aimed at managing the twin problems of sharp discontinuities and blurring, with an objective / expectation of finding the right balance between these two competing interests. Within the context of this problem definition, the blurriness of the spatially higher resolution output is a permanent feature of methods that adopt this model, hence are subject to some severe limitation on performance due to this conflict.

Alternative formulations of the problem of increasing the resolution of an image has sought to escape this balancing act by attempting an actual recovery of the lost information in an image due to coarse (under) sampling in the lower resolution input. These methods leverage on the similarity of the samples on the digital image to predict the missing samples, such methods adopt complex algorithms and employ possible background information to generate higher resolution versions of an image from lower resolution input. However, the prediction of unavailable information carries the twin outcomes of success or failure, where the failure could result in serious distortions of the image to the level of diminished utility. In addition, most of the prediction methods provides no feedback mechanism and /or prediction measurement metric to ascertain the level of deviations of these predictions from the norm. This is partly because the unavailability of the actual data set prevents the implementation of such mechanisms; this invariably consigns the usefulness of these algorithms to the accuracy of their models and ultimately providing no guarantees of success.

More so the video temporal resolution follows the similar principle as the spatial resolution, the digital video frames are sampled at specific rate, referred to as frame rate, this process is also irreversible and information lost due to low sampling rates cannot be recovered. The generation of higher temporal resolution from video sequences with low temporal resolutions can be achieved by estimating the inter-frame relationship of the objects in the video frames. The displacement of these objects across frames can

be estimated, interpolated and applied to generate the intermediate frames lost due to under-sampling. However methods for estimation of displacements and other inter-frame relationships involves computationally prohibitive calculations for accurate determination, this stems from the fact that the displacements and changes from real world scenes are difficult to model in terms of low level blocks of pixels. This leads to the adoption of high level techniques that certainly and successfully retrieve advanced displacements and complex inter-frame relations at a very high computational cost that completely erodes the usefulness of such techniques in real-time applications.

The challenge of providing an optimal combination of estimation accuracy and processing cost is very central to the task of increasing the temporal resolution of the video sequences.

Up-conversion techniques termed super resolution has been the subject of several research efforts in signal/image processing and presents enormous challenge because of the unavailability of the original signal hence focus on the up-sampling of the already digitally sampled signal.

Video super resolution is broken into two sets of techniques for spatial and temporal super resolution. The spatial super resolution as the name implies provides techniques for increasing the resolution of images/individual video frames. This consists of two dimensional digital signal processing techniques for up-sampling, re-sampling, enhancing and/or increasing the resolution of two dimensional digital samples.

The temporal super resolution provides a set of techniques for increasing the frame rate and the associated inter-frame temporal relationships.

The research presented in this thesis is structured according to two blocks of functional techniques for video spatial and temporal super resolution. The final optimal combination of the two sets of techniques is provided in the application example of video super resolution.

## 1.1 Objective and Purpose of Research

The objective of this research effort is the development of state of the art video super resolution, that improves on the performance of existing techniques, this goal is accomplished in the following two steps:
1. Research and development of video spatial super resolution that attempts to actually increase the resolution an under-sampled image or provide methods for improving the performance of current algorithms.

2. The research and development of temporal super resolution, that relies solely on two input frames to create interpolated frames between the frames in accurate inter-frame relations and objective precision to enhance the resolution.

## 1.2 Layout of thesis

The thesis is presented in the following structural layout, chapter one is this introduction, chapter two provides background information review, the third chapter provides detailed information of the work done; chapter four is the implementation of the research proposal. Chapter five is the test and comparison of the proposed techniques with pre-defined objective criteria, it provides performance analysis and chapter six is conclusion with additional insights on future work; and the references are provided after the last chapter.

## 1.3 Summary

The problem of increasing the spatial dimension (or resolution) of images has evolved from the exploration of techniques for enhancement of the image for visually / perceptually acceptable quality levels through the elimination of spikes, discontinuities and uncharacteristic variation in the statistical distribution of image pixel coloration to the attempting of an actual recreation of the original content. The temporal super resolution methods, is essentially the determination of the contiguous / coherent changes across frames and the replication of the changes in an intermediate frame in half measure.

# Chapter 2: Background Information Review

## 2.0 Introduction

Super resolution refers to the techniques used for increasing or enhancing the resolution of an image or video. The task of increasing the number of individual elements in a sparse data set has a long history, with deep origins in the interpolation theory. Interpolation as defined by Thiele [1, 2] in 1909 is referred to as the art of reading between the lines in a table. This functional definition has survived the various extensions, modifications, transformations and applications of the concept. Reading between the lines with an aim of increasing the sample size of the data set is a core application tool in signal, image and digital video processing. The overview of methods for super resolution is presented for spatial (image) and temporal resolutions respectively.

## 2.1 Spatial (or Image) Super Resolution

The earliest methods for increasing the vector dimension (or spatial resolution) of an image is the basic interpolation function known as the nearest neighbor [3], in this method the value of the new (or interpolated) point is a direct replication of the previous point. This method is equivalent to convolving an image with a rectangle function (Figure 2.1), which can be interpreted as multiplication with a sinc function in the frequency domain (Figure 2.2).

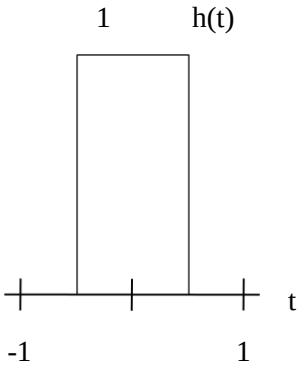



Figure 2.1: Nearest Neighbour time domain

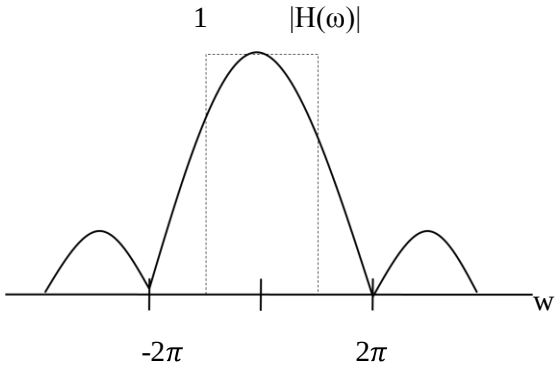



Figure 2.2: Frequency domain, Nearest Neighbour

The resulting image from this method has sharp discontinuities and is shifted with regard to the original co-ordinate points of the image. Therefore pixel relations cannot be maintained using this approach, in

addition, the frequency attributes of the output image and the original are exactly the same with a simple linear phase shift.

The linear interpolation (bi-linear for 2D), is an improvement of the nearest neighbour algorithm in which the new (interpolated) point is derived from the interpolation of two adjacent neighbours. The linear interpolation can be implemented as the convolution of the image with the triangle function. The process results in a low pass filtering in the frequency domain (Figure 2.3 and Figure 2.4) with strong smoothing in the cut-off frequency.

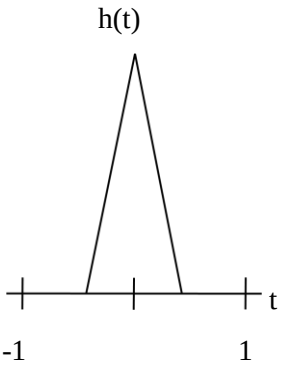



Figure 2.3: Linear Interpolation time domain

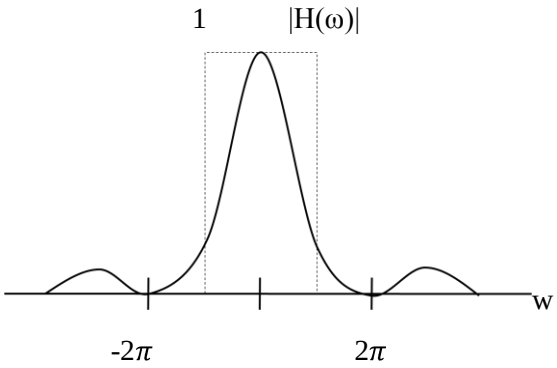



Figure 2.4: Frequency domain Linear Interpolation

The cubic (bi-cubic for 2D) interpolation [4, 5, 6] also improves on the linear interpolation by using three points instead of two, as a result of this extension, the interpolated image is smoother. The cubic interpolation is implemented using cubic convolution in the spatial domain. This results (see Figure 2.5 and Figure 2.6) in a low pass filter with smoother cut-off frequency response in the frequency domain.

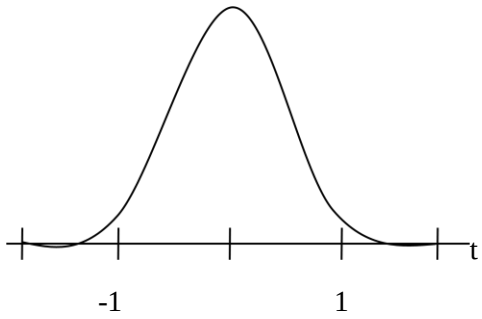



Figure 2.5: Cubic Interpolation time domain

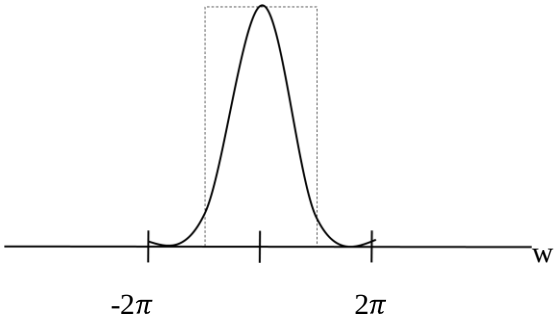



Figure 2.6: Frequency domain Cubic Interpolation

The cubic interpolation can also be implemented using the Lagrange polynomials and cubic B-splines.

However, additional methods of image super resolution exists, these include the many variations of splines. The concept of piecewise polynomials with smoothly separated units' points has been in existence in various shapes and forms and was formalized by Schoenberg [7] in 1946 as splines.

Splines can be described in terms B-splines expansion where B stands for basic:

$$s(x) = \sum_{k\in Z}^{n} c(k)\beta^n (x-k) \qquad (2.1)$$

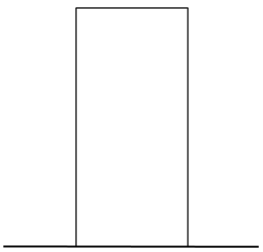
Equation 2.1 represents the integer shifts of the B-spline of degree n, the c(k) are the B-spline coefficients. B-splines are constructed from rectangular function ß defined as $ß^n(x)$.

$$ß^0 = \begin{cases} 1, & -\frac{1}{2} < x < \frac{1}{2} \\ \frac{1}{2}, & |x| = \frac{1}{2} \\ 0, & otherwise \end{cases} \qquad (2.2)$$

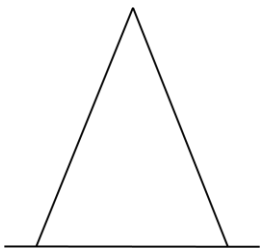
$$\beta^n(x) = \beta^0 * \beta^0 * \ ... \ * \beta^0(x) \qquad (2.3)$$

(n+1 times)

B-Splines of degree 0, 1 and 2 are piecewise constant, linear and cubic respectively (see Figure 2.7); higher degrees can be easily constructed from the basic splines known as B-splines.



Degree 0



Degree 1

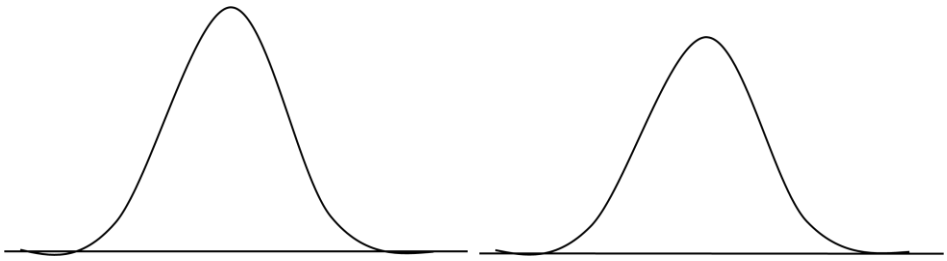



Figure 2.7: The B-splines functions for degrees 0…3

The cubic B-spline is very popular in image processing due to its curvature and can be derived from the B-spline as presented below in Equation 2.4:

$$\beta^3 = \begin{cases} \frac{2}{3} - |x|^2 + \frac{|x|^2}{2}, & 0 \le |x| \le 1 \\ \frac{(2-|x|)^3}{6}, & 1 \le |x| < 2 \\ 0, & 2 \le |x| \end{cases} \qquad (2.4)$$

However, direct low pass filters has been designed for increasing spatial resolution, several methods with varying low pass response(s) (see Figure 2.8) exists, these are designed to achieve a smooth pixel inter-relationships across interpolated points. The two dimensional (2D) recursive (IIR) filters [8] is a typical example, however other versions with FIR implementations also provides comparative performance at additional high computational cost. In addition separable two dimensional filtering is also used for lower computational cost with the obvious downside of performance penalties.

Figure 2.8: Typical design response(s) for low pass filters used for resolution enhancement.

Fractal zooming is another method of increasing the resolution of an image, it is derived from fractal coding theory[9, 10, 11] that treats images as a collage of smaller image structures that can be expanded

or compressed using geometric transformation functions. In fractal compression, the compressed image is resolution independent, this stems from the premise that images are characterized as fractals which is defined as a loosely connected conflagration of self-similar objects. This definition contends that fractals possesses no physical characteristic sizes, rather strictly self-similar units with metric properties defined on the atomic level, hence the overall arrangement is scale independent. Self similarity in fractal parlance is defined by the expression below in Equation 2.5:

$$M(r * x) = r^{f(D)} M(x) \qquad (2.5)$$

M(x) represents any metric property of the fractal (e.g. area or length), and *x* denotes the scale of measurement of the metric property, *r* is a scaling factor, such that $0 \leq r \leq 1$ and *f(D)* is a function of the fractal dimension D for the given metric property.

An example of the fractal coding method as originally introduced by Von Koch known as Von Koch Curve or Snowflakes is presented in Figure 2.9.

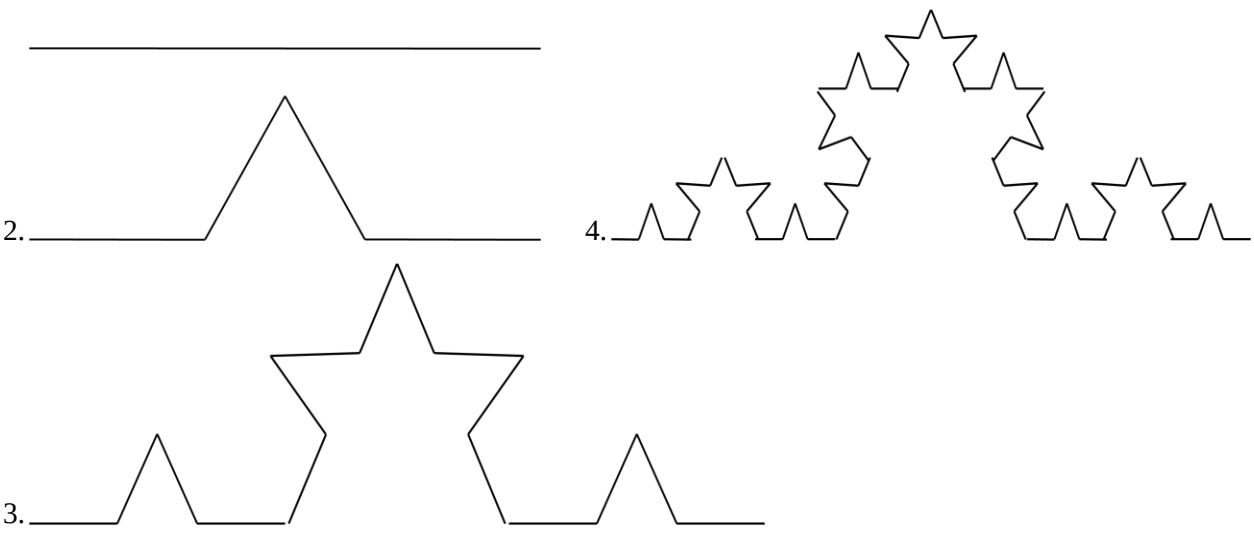



Figure 2.9: Van Koch Curve or Snow flakes, the rule is repeated at every stage

The Von Koch curve example in Figure 2.9 shows the different transformation of the original line, this simple illustration is at the heart of fractal coding theory, in which every image is decomposed into constituent atomic units and the decoding process involves the iterative application of the transformation

rules on these units to recover the original image. However since the atomic units is the sole content of the entire image, the decoding process can recover the image to any desired scale independent of the original. However, despite the promising nature of the theory and the utter usefulness of its application, Fractal coding is yet to gain traction due to the prohibitive computational cost of the encoding process. Several attempts at simplifying the process with wavelet-based decomposition have been proposed [12] and remain a strong focus of research efforts. Examples of fractal generated images are presented Figure 2.10.

Figure 2.10a: Example of fractal generated Image

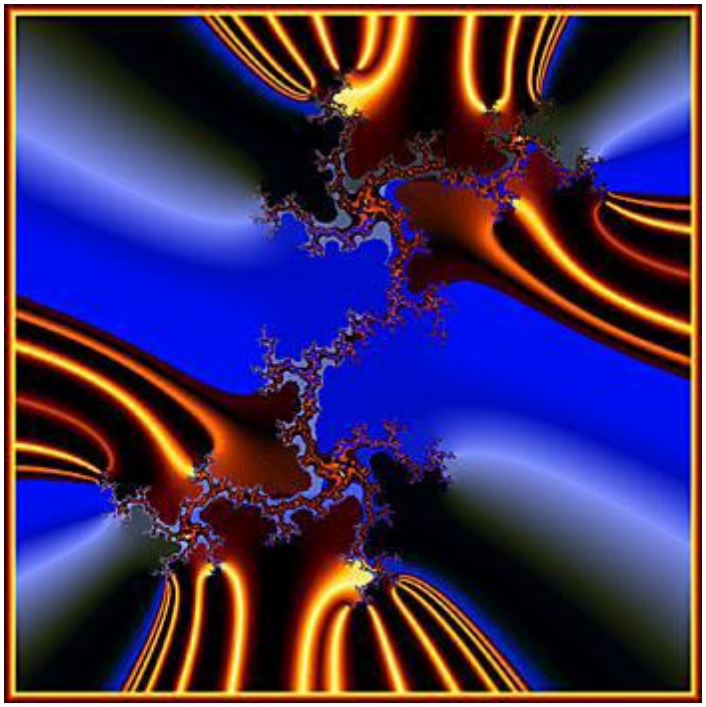


Figure 2.10b: Example of fractal generated Image

The wavelet based methods are the latest additions to the litany of super resolution algorithms. The unique contribution of this group, of super resolution techniques is the focus on the possibility of increasing the resolution (the spectral content) of a signal rather than enhancement. Common and central to all wavelet based techniques is the multi-resolution analysis of the input signal into several resolutions or scales and reliance on the structural self similarity evident across the scales to predict the next higher or finer scale of detail co-efficients (see Figure 2.11). However the divergence in approach occurs on the method of prediction, the Hidden Markov tree [13], have been used to predict the next set of co-efficients using training data from a large pool of images. The wavelet co-efficients at the finest scales is predicted based on the state transition probabilities across the scales and the determination of an expected value of the state of the co-efficient at the finest scale. The method also features a sign prediction algorithm where the predicted magnitude of the finest scale co-efficients is assigned a positive or negative sign based on the sign transitions of its ancestry.

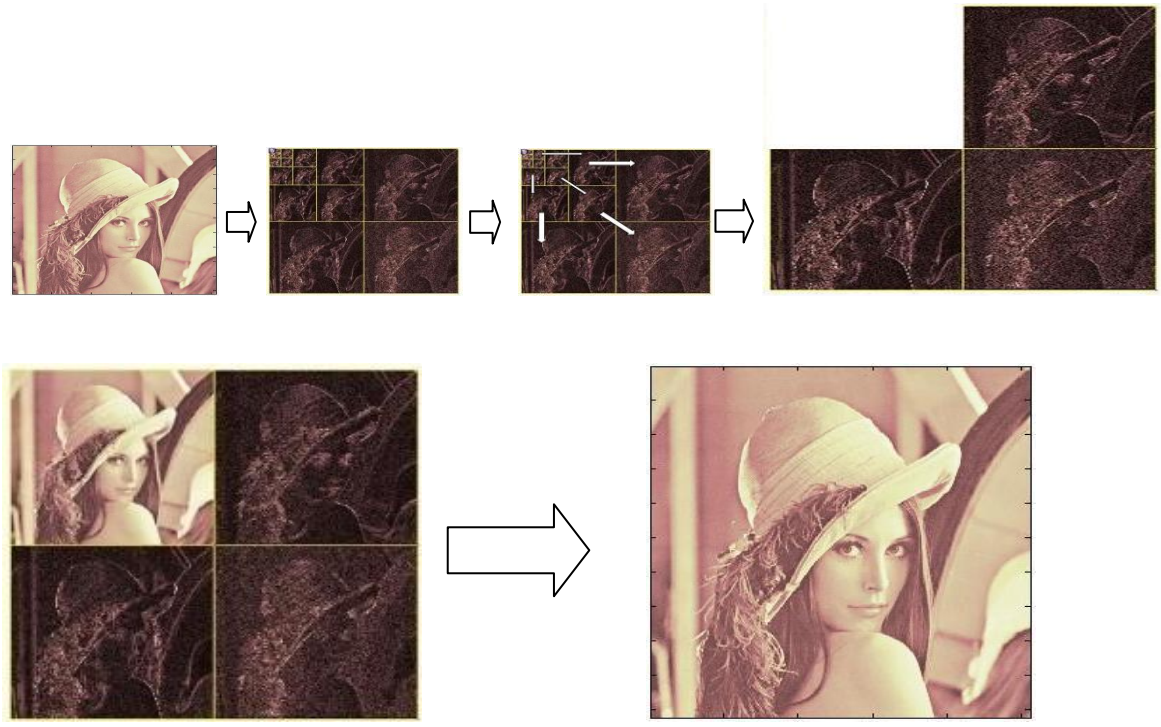


Figure 2.11: General Algorithmic Structure for Wavelet-based Super Resolution

More so additional modification [14] has been made to this method where the requirement for training is eliminated and the co-efficient sign prediction method simplified, resulting in improved comparative performance. However, the two methods follow a tightly coupled quad-tree inter-scale dependency where the magnitude of the parents propagates to child co-efficients, however the quad-tree structure provides a potential pitfall for this approach for higher levels of iterations as low magnitude parents often yield very high magnitude in the child co-efficients due to shifts. More so the simplified co-efficient sign prediction directly extends the sign of the parent to the children, this approach though computationally efficient is fraught with potential errors. Directional wavelet filtering [15] has been used to improve the performance of increasing resolution, it attempts to create multiple and flexible directional filtering radically different from the traditional horizontal and vertical separable two dimensional filters in order to closely model the characteristics of the image edges, the performance of the method lags standard wavelet methods in complex images though it provides marginal gains in less complex ones. Neural network method has been proposed [16], and an edge adaptive method using Markov chain Monte Carlo is also proposed [17], in addition several statistical and algebraic methods exists, but these methods lacks a definitive model that constrains the feasible solutions space for the predicted co-efficients, hence rely heavily on the validity and accuracy of the model to achieve desired results.

## 2.2 Temporal Super Resolution

Methods for increasing or enhancing the temporal resolution of video sequences include the simplistic replication of frames, in which the video frames are simply repeated in order to increase the frame rate. Beyond this method, other approaches are deeply rooted in, and practically synonymous with the motion estimation techniques for eliminating redundancy across video frames. However the application of this technique to super resolution requires the interpolation of the motion vectors for motion compensated prediction of the interpolated frame; this is commonly referred to as motion compensated interpolation for video format conversion (MC-VFC).

Frames with temporal proximity in video sequences generally contain little variations in content; a greater percentage of the variations can be grouped as motion of objects (group of pixels) in the frames. Motion estimation of pixels of a video frame is the focus of several research efforts, past and ongoing, several techniques has been proposed, implemented, deployed and/or modified. Majority of the existing algorithms provide a model of the motion estimation problem as the determination displacement vectors of a fixed and variable block of pixels in a frame, while this is uncharacteristic of motion in video sequences, the methods holds sway due to the computational efficiency. An alternative high level object

based motion estimation methods that leverages on object segmentation provides greater precision and accuracy in this regard, but this approach is computational prohibitive, within the context of the state of current implementations, for any practical usage in digital video processing applications requiring low processing latency. Prominent block based motion estimation algorithm includes the Full Search (also known as Exhaustive Search or Global Search), this method provides the best results among all matching algorithms due to its exhaustive search of the target frame for the best match for the candidate block of an image sequence. However, the approach is very computationally expensive and is rarely used. Other methods like the three step search, attempts to achieve the quality performance of the full search with far lower computational cost, this method employs a three step hierarchical search method (see Figure 2.12) where the search area is constantly refined according to a cost function and the new search area is defined that represents the best target for matching the candidate block. Additional modifications to this method include the New Three Step Search [18], Simple/Efficient Search [19] and the Four Step Search [20].

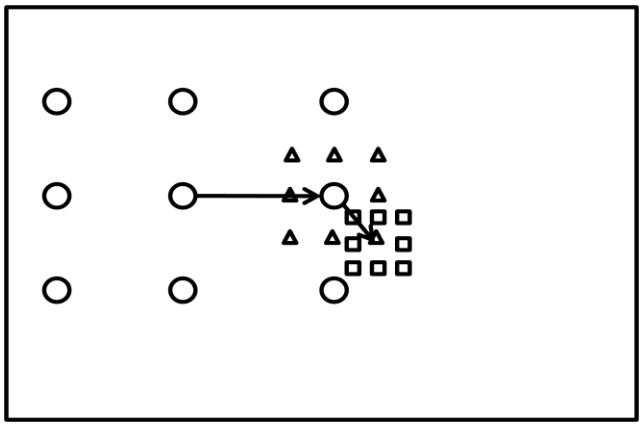


Figure 2.12: The Three Step Search, circles, triangles and squares represent the first, second and third step respectively

Several hybrid and modifications of these search patterns have been proposed and implemented, these include the Diamond Search [21], and this is an extension of the four step search but differentiates from it by adopting a diamond search point pattern instead of a square. The search method uses two fixed size diamond shapes, called the large diamond search pattern and the small diamond search pattern (see Figure 2.13); the search process is initiated with the large diamond and the search area is refined based on the

best match, the hierarchical refinement is implemented with large diamonds until the last step, when the small diamond is actually used.

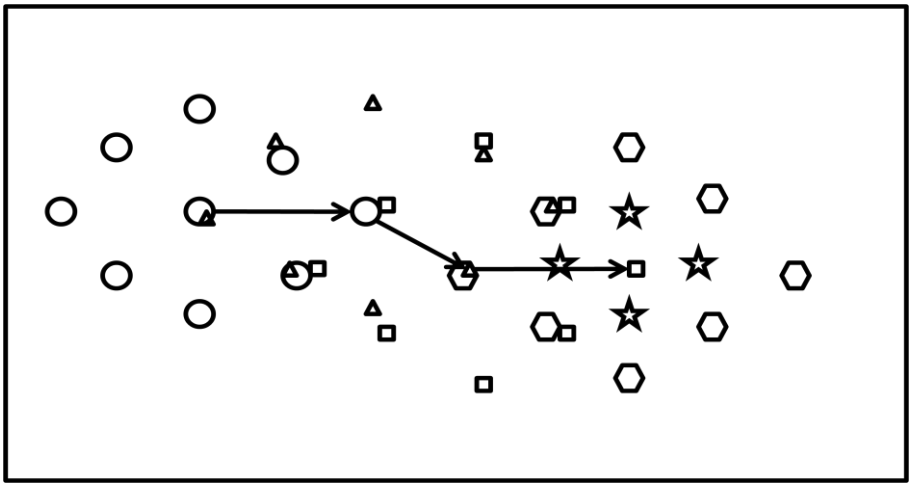


Figure 2.13: The Diamond Search, circles, triangles, squares, hexagons and stars represent the first, second, third, fourth and last step respectively

There are related methods derived from the methods presented, these derivations include extensions like the adoption of the complex shaped polygon patterns and variable shaped patterns. The diamond search method provides comparable quality performance to the Full Search and provides the basic structure for most existing extensions in literature.

However the quest for efficient and accurate motion estimation across video sequences is not limited to spatial domain techniques, the fix all wavelet phenomena has not spared this area of digital video processing, there are currently two evolutions in this area, the first is the multi-resolution motion estimation (MRME) methods that tend to create efficient method for motion vector determination at different levels of resolution without repetitive processing. The second line of approach is the wavelet-based motion estimation that focuses on traditional motion estimation but with a new wavelet based technique. Multi-resolution motion estimation (MRME) typically involves the estimation/determination of motion vectors at either coarsest or finest scale, followed by subsequent refinements of the vectors across the hierarchical structure in either directions, implementations of MRME include a premier work and widely referenced material [22], other research work in this field includes [23], more so an analysis [24] of the performance of the transversal methods (coarsest to finest OR finest to coarsest), suggests that

the finest to coarsest vector estimation and refinement approach provides superior performance. This result completely undercuts the super resolution potential of the coarsest to finest approach, as it would have provided a perfect fit to the problem formulation of wavelet based video super resolution.

Wavelet based motion estimation adopts multi-scale (resolution) sub-band based matching, in this scenario, the target area/blocks in each wavelet reference sub-band is matched to the corresponding current sub-band, additional approaches includes the use of redundant wavelet transform for sub-band matching. More so the wavelet based motion estimation techniques has recorded a giant leap in comparison to the established spatial domain methods, the best results presented [25] in this area as at the time of writing this thesis is competitive to the spatial domain approach in terms of quality though lags it severely in resource requirement.

## 2.3 Summary

The video spatial (image) super resolution methods with the exception of wavelet and fractal methods, provides an enhancement of the lower resolution image, this approach though practical and pragmatic in the face of the sheer impossibility of recovering the under-sampled information, provides a very strong limitation, such that the higher resolution (HR) image, is essentially as good (if not worse) as the lower resolution version, thereby paving the way for the exploration of techniques for the recovery of lost spectral content. The motion estimation methods for temporal super resolution, provides search methods for handling translational motion of blocks of pixels. Several existing propositions and algorithms, derived from the presented methods; all conforms to this general principle.

# Chapter 3: Proposed Video Super Resolution

## 3.0 Introduction

Video super resolution is accomplished in two steps, spatial and temporal; each of these steps requires distinct set of techniques for its implementation. The proposed methods are presented in that order, in the following three sections, the first section provides details of the proposed spatial (or image, it is used interchangeably throughout this text) super resolution, the second section deals with the temporal techniques and the third and final section provides an optimal combination of the two bodies of techniques to implement video super resolution.

## 3.1 Video Spatial (Image) Super Resolution

The main purpose of this work is to generate an output that is greater in spatial resolution than the original input and conforms to the statistical/spectral dynamics of the image at this new spatial resolution. However achieving the latter remains a daunting task owing to the fact that the Super Resolution (SR) algorithms are technically constrained to up-sampling of already digitally sampled signal. Several SR methods as described in chapter two provides an enhancement of the resolution at higher spatial dimensions owing to this challenging constraint. However wavelet based groups of techniques provide the possibility of increasing rather than enhancing the resolution of an image. This approach provides a platform that supports the probable recovery of lost image details due to under-sampling. The proposed video spatial SR method leverages on this possibility to attempt a recovery of high resolution image from the lower resolution version, using it as the only input.

The problem model and the detailed description of wavelet theory strictly within the context of spatial (image) super resolution is presented in the following sections succeeded by the proposed technique based on the model.

The problem definition: The proposed super resolution model is defined for a single constrain, it applies to band limited signals. For a continuous time one–dimensional signal *f (t)*

$$F(jw) = \int f(t) e^{jwt}\, dt \qquad (2)$$

there exist a certain frequency q for which |w| > q, F(jw) = 0; where (2) is the Fourier transform of the signal. Hence for the discrete-time spectrum

$$F_d\left(e^{jWdt}\right) = \frac{1}{T}\sum F[\mathrm{j}(\mathrm{w_d} + \mathrm{m}\Omega\mathrm{s})] \qquad (3.0a)$$

for $2q \geq \Omega_s \; and \; \frac{-\pi}{T} \leq w_d \leq \frac{\pi}{T}$, m = 0

$$F_d\left(e^{jWdt}\right) = \frac{1}{T}F_j(w_d) \qquad (3.0b)$$

This signal can be reconstructed perfectly from the discrete time version. Hence the super resolution problem can be characterized as the increase of the sampling frequency from a level (< 2q) to any desired level, with a maximum limit at the Nyquist limit (> 2q). To achieve this objective, a frequency domain solution that attempts to recover the higher frequencies beyond the range the signal are sampled is proposed. In this approach, the spectral distribution of the input signal is used to predict the lost/unavailable frequencies. To explore the frequency properties of the signal, wavelet multi-resolution analysis is applied; this provides the advantage of time-frequency [26] characterization of the signal unavailable in Fourier transform.

For wavelet transform of an input signal x (t) the output wavelet co-efficients $C_{s,\tau}$.

$$C_{s,\tau} = \int x(t)\,\varphi_{s,\tau}(t)dt \qquad (4)$$

$$\varphi_{s,\tau} = \frac{1}{\sqrt{s}}\frac{\varphi(t-\tau)}{s} \qquad (5)$$

$\varphi$ is the wavelet function, s is scale (1/frequency), and $\tau$ is time shift. The inverse is the linear combination of the wavelet at all scales and translations.

$$x(t) = \sum_{s=-\infty}^{\infty}\sum_{\tau=-\infty}^{\infty} C_{s,\tau}\varphi_{s,\tau}\,(t) \qquad (6)$$

However, the signal can be approximated to a desired resolution (scale) s=A, in which case the approximated version of the original signal is $x_A(t)$.

$$x_A(t) = \sum_{s=-\infty}^{A}\sum_{\tau=-\infty}^{\infty} C_{s,\tau}\,\varphi_{s,\tau}(t) \qquad (7)$$

The difference between the approximated version at s=A and the original signal

$$x_D(t) = x(t) - x_A(t) \qquad (8)$$

For discrete time signals the approximation is represented by equation (9) where h (nT) is the impulse response of the wavelet function.

$$x(nT) * h(nT) = \sum_{\tau=-\infty}^{\infty} x(\tau)h(nT-\tau) \qquad (9)$$

The difference is the convolution of the input signal with a high pass filter g(nT) which is related to the approximation (low pass) filter as shown in equation 10

$$g(L-1-nT) = (-1)^n h(nT) \qquad (10)$$

where L is the filter length

Multiple approximations of the original signal at different scales (resolutions) with the resulting differences at each level constitute the much-famed wavelet multi-resolution analysis (MRA). This is typically implemented in a filter bank as shown in Figure 3.1, derived from sub-band coding.

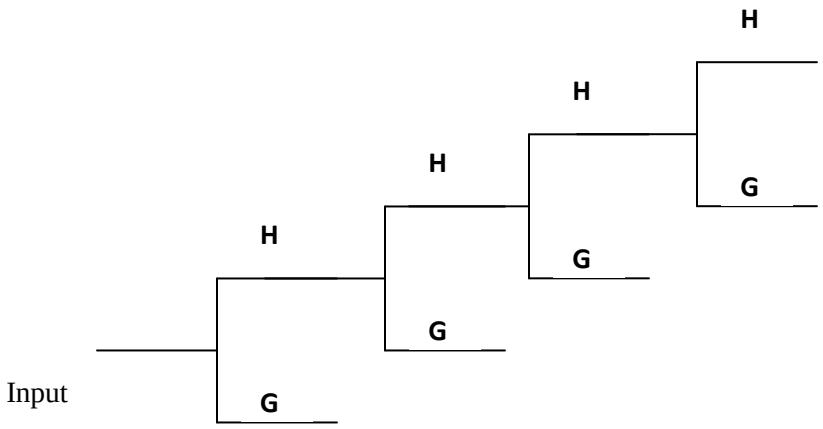



Figure 3.1: Filter bank for Multi-resolution Analysis

Multi-resolution analysis can be defined [27] as a sequence of spaces $\{V_j\}_{j\in z}$ of $L^2(R)$ with respect to a function if the following constraints are true.

$$\forall (j,k) \in Z^2,\ f(t) \in V_j \leftrightarrow f(t-2^j k) \in V_j \qquad (11.1)$$

$$\forall_j \in Z,\ V_{j+1} \in V_j \qquad (11.2)$$

$$\forall_j \in Z,\quad f(t) \in V_j \leftrightarrow f\left(\frac{t}{2}\right) \in V_{j+1} \qquad (11.3)$$

$$\lim_{j\to+\infty} V_i = \bigcap_{j=-\infty}^{+\infty} V_i = \{0\} \qquad (11.4)$$

$$\lim_{j\to-\infty} V_i = \left(\bigcup_{j=-\infty}^{+\infty} V_i\right) = \mathrm{L}^2(\mathrm{R}) \qquad (11.5)$$

$V_j$ is translation invariant proportional to scale $2^j$, equation (11.1), while equation (11.2) states the causality property, equation (11.3) specifies an approximation of the original signal at coarser resolution, more over all details of the signal is lost when the resolution goes to zero as shown in equation (11.4), and inversely (11.5) the original is recovered as resolution tends to infinity. In the wavelet based SR process, the LR input constitutes the approximation, while the difference is predicted. Wavelet multi-resolution analysis provides a tool for estimation of the relative similarity of the differences across scales, useful in predicting the next higher set of unavailable differences for which the input signal is the approximation.

The wavelet decomposition of signals into approximation and difference components, results in output wavelets co-efficients with same sample size as the original signal for each component. For two dimensional signals like images, the wavelet transform co-efficients will be four times the original image size, however for perfect reconstruction of the original input, only half of the wavelet co-efficients (in both directions) is required as this can be accomplished using either the set of even or odd co-efficients, this leads to the subsampling of the wavelet co-efficients used in discrete wavelet transform (DWT), as the original output is considered redundant or over-complete for reconstruction. However the output sub-sampled version (DWT), is altogether complete for perfect reconstruction but individually incomplete, as each component suffers from shift variance, due to subsampling, more over within the context of multi-resolution analysis, iterative subsampling greatly reduces the usefulness of co-efficients for inter-scale frequency analysis as the increasing reduction of samples sizes results in increasingly low frequency resolutions completely unsuitable for such analysis. This leads to the adoption of redundant discrete wavelet transform for inter-scale frequency analysis.

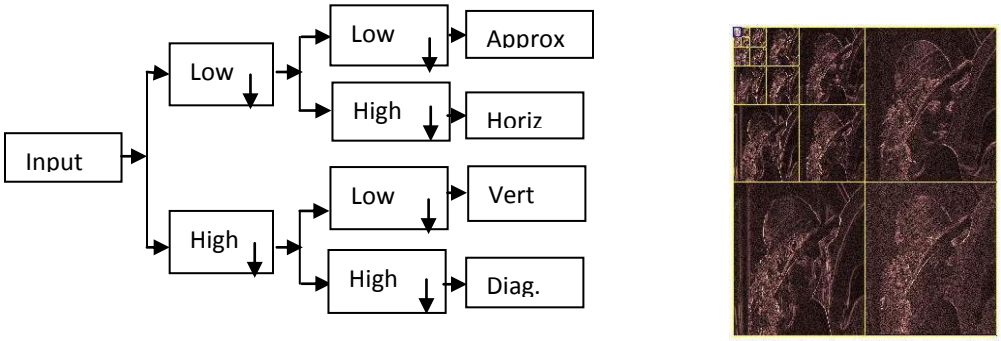



Figure 3.2: (a) Wavelet Decomposition (b) DWT Multi-resolution Analysis of Sample Image Lena

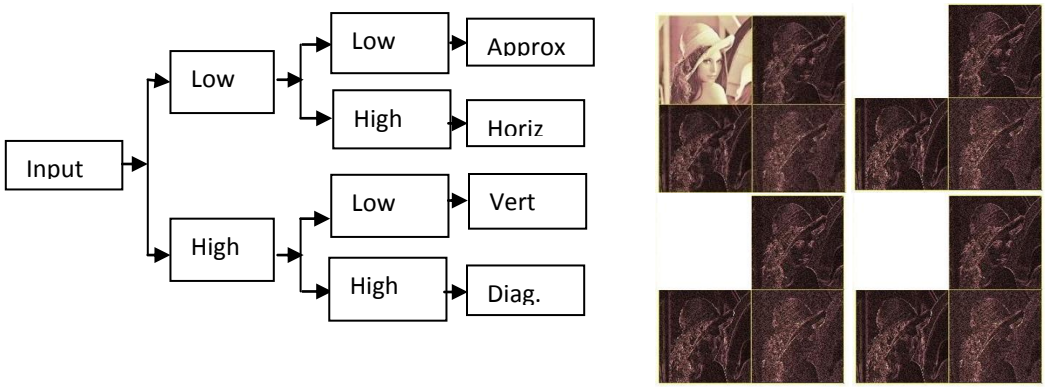



Figure 3.3: (a) Redundant Wavelet Decomposition (b) RDWT Multi-resolution Analysis of Sample Image Lena

The inter-scale analysis of the high frequency components (Horiz., Vert., and Diag.) of the resulting wavelet co-efficients accomplishes two goals: (1) The determination of the existence of sufficient high frequency information in the image, which is pre-requisite for the proposed image super resolution algorithm (2) It provides a useful resource for the detection of the resolution invariant features of the image, detection of these features provides the unique frequency characterization of the image. The details of the two design objectives are presented below:

**Spectral content pre-requisite:** The proposed algorithm relies on the higher ranges of frequency content, therefore the prior establishment of the availability of this crucial information is necessary to guarantee any performance. This is established by the estimating the regularity of the wavelet co-efficients across the scales in fine to coarse order as shown in Figure 3.4, 3.5, 3.6.

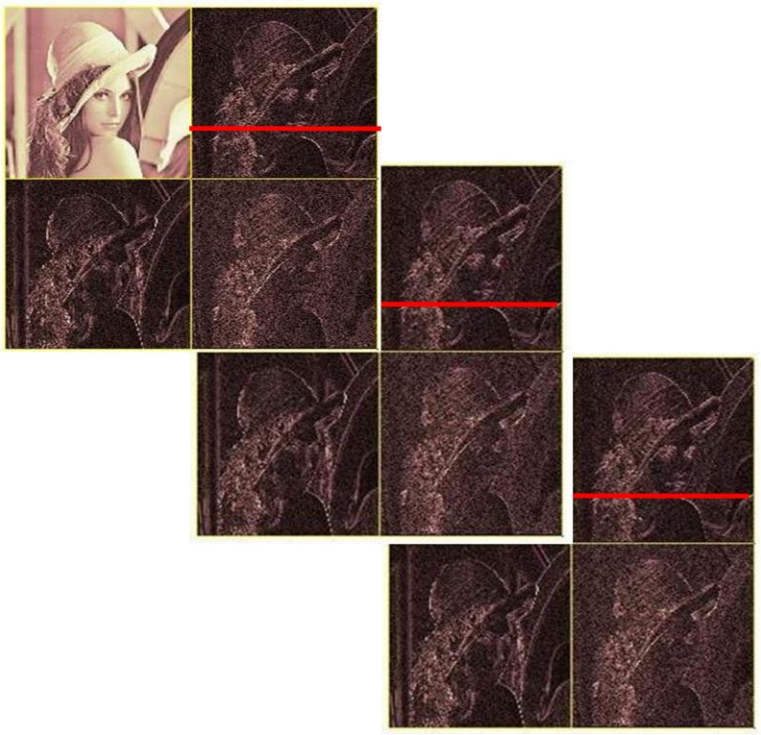


Figure 3.4: Inter-scale co-efficients regularity estimation for prediction

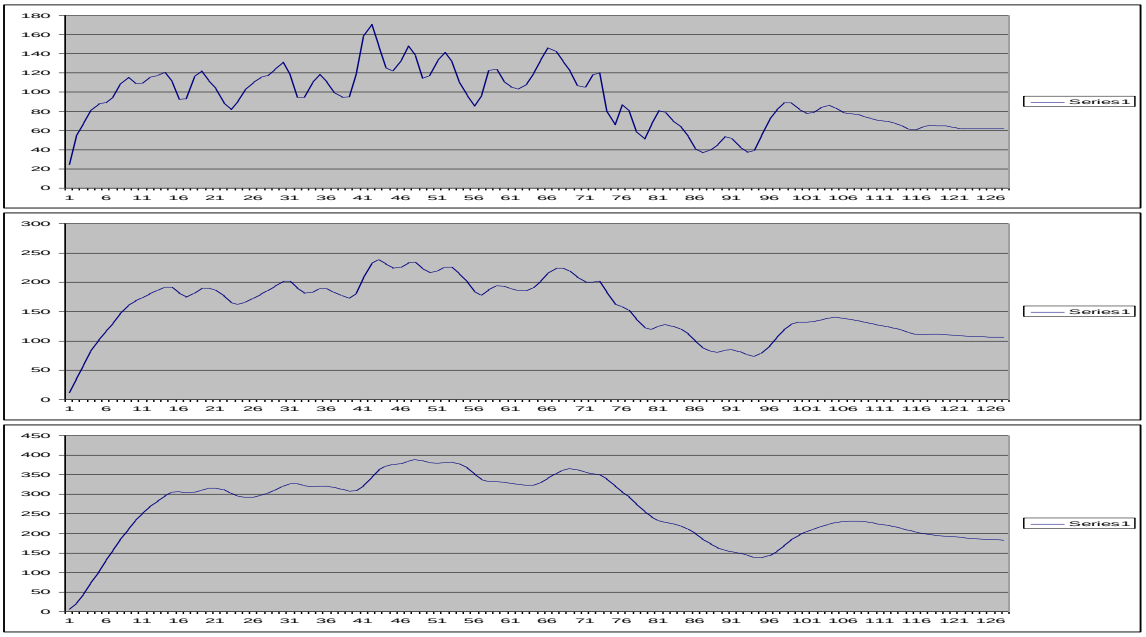



Figure 3.5: The Plot of the red lines in Figure 3.4 in ascending order, shown here in descending order.

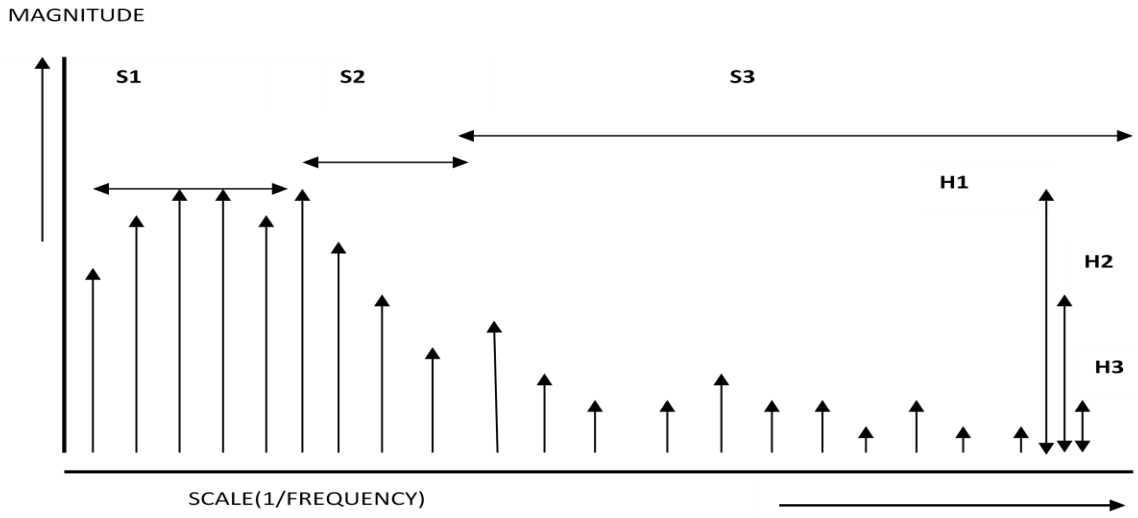



Figure 3.6: The delineation of the frequencies based on the inter-scale regularity

In the Figure 3.6, the S3 cluster shows a distinct regularity with increasing magnitudes from finest scale to the coarser scales. The S3 cluster must exist in an image for any guarantee of performance, as the method relies on the enforcement of the equality constraint on the image features grossly dependent on the spectrum range. Images (see Figure 3.7) adjudged by this process as having insufficient higher range spectrum content typically lack feature detail that can be reliably enforced on the estimated higher resolution version.

**Resolution Invariant Feature Detection**: The inter-scale analysis also provides a tool for detection and estimation of resolution invariant features of the image. The correlation between the co-efficients across scales within the S3 cluster (Figure 3.6) provides the synchronization points for automatic extraction of the exhaustive and salient features of the image that transcends resolutions.

However, this is a mathematically involved and computationally expensive process, an alternative proposition is the use of standard features of an image that are resolution invariant, these include the edge strength, edge direction and edge continuity. This is adopted in this research.

Succeeding the analysis is the prediction of the unavailable wavelet co-efficients using the Lipschitz regularity; this provides an estimation of the growth rate across scales. The predicted values are constrained to conform to the standard features of the lower resolution image, in comparison to the generated higher resolution version. The refinement process of the predicted co-efficients based on the

constraints is a bounded non-linear optimization, where the solution space is confined to the magnitude of the coarser co-efficients.

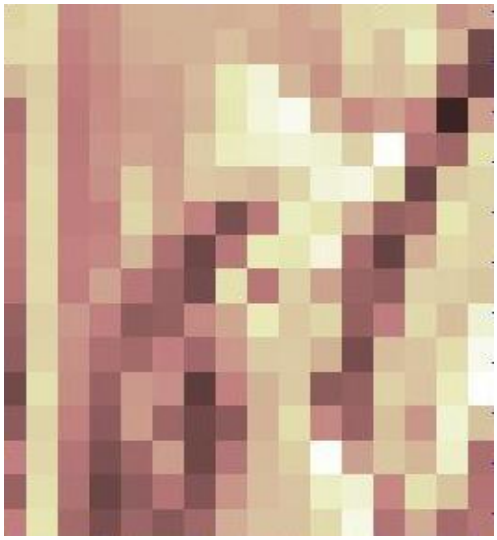


Figure 3.7: S1 cluster of Image Lena (or Lenna)

## 3.2 Temporal Super Resolution

The performance of motion estimation algorithms largely depends on the search and matching methods, the objective matching criteria has very limited tunable parameters and hence a move from Mean Square Error (MSE) to the Mean Absolute Deviation (MAD) has been the significant progression in this area.

The central core of the performance for motion estimation algorithms reside in the search methods and all existing algorithms are defined and differentiated by their search methods. Several algorithms for motion estimation from the exhaustive search (or Full Search) to the Four Step Search employ diverse approaches to implementing an optimal search algorithms, which ultimately aim at improving the computational performance of these techniques, in furtherance to this objective, a new method is proposed that uses redundant discrete wavelet transform and temporal filtering to extract the changes across video frames and constrain the search area to the defined regions of the frames temporally associated. The proposed technique has no close implementation in literature or equivalence for fair comparison, but provides an improvement in performance over existing methods.

The temporal aspect of the video super resolution is modeled as the increase of the frame rate by generating and inserting intermediate frames between two adjacent frames. This model requires two input frames temporally related in chronological order and returns an output with a number of intermediate frames.

The insertion or generation of an intermediate frame is solely based on the accurate depiction of the temporal relationship across the adjacent input frames. This process derives from techniques and algorithms that estimate the relative redundancies, differences and displacement across frames. These methods largely referred to as motion estimation algorithms (as presented in chapter two), employ mainly objective matching of the areas of the input frames. The proposed algorithm improves on these methods by restraining/confining the matching target to areas of the input with differences or displacement. This provides an improved performance and applicability to real-time video processing. The estimated displacement vectors across the two input frames are interpolated to produce the predicted intermediate frame(s).

To estimate the difference(s) or displacement across two input frames, redundant two dimensional (2D) discrete wavelet transform is applied to the frames of the input video, the resulting co-efficients of the adjacent frames (the approximation and three detail orientations or sub-bands) of the input video are filtered using high pass reversible integer Haar wavelet filter. The resulting frames from the temporal filtering contains low and/or near zero magnitude values for co-ordinates with little or no changes across frames and high values for areas of relative motion. The result of the temporal filtering hereafter referred to as motion profile is segmented using an adaptive threshold that classifies the values into motion and non – motion pixels.

In the threshold-based classification process the sorted co-efficients values of the motion profile is analyzed for sharp rise/relative discontinuity in magnitude, the resulting value from this analysis become the magnitude threshold. This process can also be accomplished using a simple hard coded threshold resulting in very minimal outliers. The consequence of obtaining sub-optimal threshold classification due to the hard coded baseline is the minimal or fringe noise spikes in the motion profile, this is far too low penalty in comparison the compulsory increase in processing footprint added by the adaptive threshold method.

The classification scheme is used for predicting the areas of the adjacent video frames with significant changes between the frames. The performance of this motion estimation algorithm hinges squarely on the

accuracy of this prediction. The co-efficients below the threshold is made equal to zero. A two step 2D inverse discrete wavelet transform is applied to the magnitude classified wavelet co-efficients detailed below:

1. The co-efficients are used in two dimensional inverse discrete wavelet transform.
2. The approximation co-efficients are completely replaced with zero and an inverse 2D discrete wavelet transform is applied.

The image pixels resulting from the two steps process above is added to produce the motion areas of two adjacent video frames. The generated image is used for block matching where only block within the motion areas are selected for matching thereby optimizing the performance.

## 3.3 Summary

The presented video spatial (image) super resolution method, attempts the recovery of the lost image information due to under-sampling by re-creating it following an established rule (feature constraints), The extent of the recovery is dependent on the availability of these rules, the detection/creation of the feature constraints is not considered, additionally the detection of non-standard feature constraints would extensively enhance the potential robustness and precision of the proposed algorithm. The video temporal super resolution provided a stable search method for extraction of motion across frames.

# Chapter Four: Implementation

## 4.0 Introduction

The video super resolution algorithm was implemented in a two step design process of image and temporal super resolution. The individual merits and trade-off of the methods are analyzed independently, there after an optimal combination of the two methods for real-time video super resolution is presented. The presentation in this chapter follows that order.

## 4.1 Image Super Resolution

The implementation is a five step process listed below:

(1.) Redundant Discrete Wavelet Transform
(2.) Estimation of the growth rate
(3.) Grouping/ Delineation of the Image Frequencies
(4.) Prediction of the coefficients
(5.) Refinement of the predicted coefficients

The details of the steps are presented in the following sections:

### 4.1.1 – Redundant Discrete Wavelet Transform

Two dimensional separable discrete wavelet transform is applied on the input low resolution (LR) image, using the Cohen Daubechies Feauveau 9/7 bi-orthogonal filter as shown in Figure 4.1, the filter kernel is presented in Table 4.1.

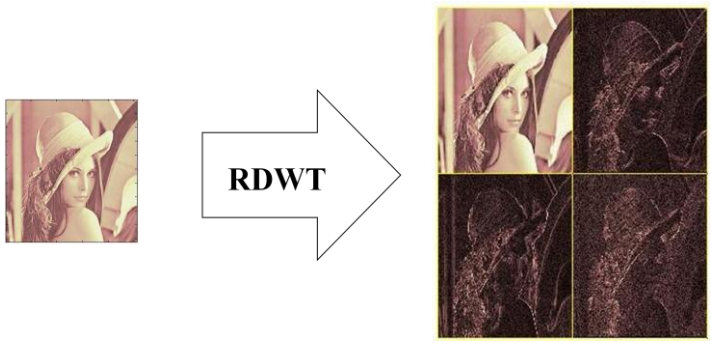



Figure 4.1: First step of the proposed image super resolution algorithm

Table 4.1: The cdf-9/7 filters kernel used in this implementation is the FBI version

| | |
|---|---|
| Analysis low-pass filter: | 0.0267487574108098,-0.0168641184428750,-0.0782232665289879, 0.2668641184428723, 0.6029490182363579, 0.2668641184428723, -0.0782232665289879, -0.0168641184428750, 0.0267487574108098; |
| Analysis high-pass filter: | 0.0456358815571247, -0.0287717631142498, -0.2956358815571235, 0.5575435262284970, -0.2956358815571235, -0.0287717631142498, 0.0456358815571247; |
| Synthesis low-pass filter: | 0.0534975148216208, -0.0912717631142514, -0.1564465330579798, 0.5912717631142532, 1.2058980364727310, 0.5912717631142532, -0.1564465330579798, -0.0912717631142514, 0.0534975148216208; |
| Synthesis high-pass filter: | 0.0337282368857512, -0.0575435262285022,-0.53372823688575, 1.1150870524570070, -0.5337282368857500, -0.0575435262285022, 0.0337282368857512; |

The output of the transform is four times the size of the input. The RDWT is used for Multi-resolution analysis applied to the next step.

### 4.1.2 – Estimation of regularity and growth rate

The multi-resolution analysis of the input image is used for estimating the regularity and growth rate of the wavelet coefficients across scales as shown in Figure 4.2 below.

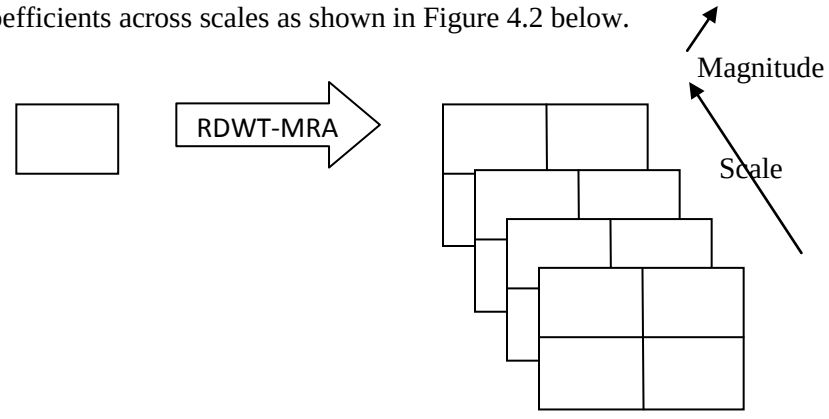



Figure 4.2: Multi-resolution analysis using RDWT, the co-efficients across scales is analyzed.

### 4.1.3 – Wavelet coefficients grouping and delineation

The frequency distribution of the image from the multi-resolution analysis is used for the estimation and delineation of the wavelet bands into groups based on regularity. The existence of S3 cluster in the image is a core requirement for the proposed method, the delineation is shown in Figure 4.3. The inter-scale relationship across the finer scales is estimated using the Lipschitz regularity [27], this defines the calculation of the growth factor termed Lipschitz exponent. To calculate the Lipschitz exponent, the absolute maximum value for the coefficients on the finest (first set of coefficients from the MRA) scale is found, and the slope of the log2 of both the coefficient and that of the absolute value of the coefficient in the same position in the next higher scale is the Lipschitz exponent for these two adjacent scales.

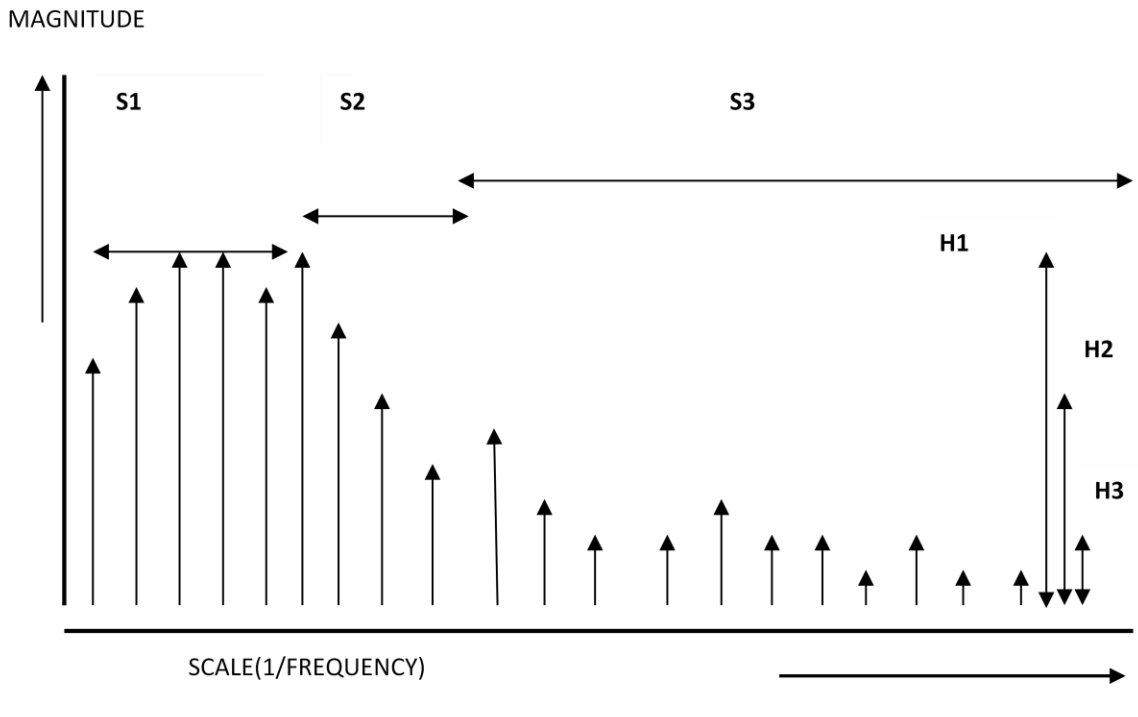



Figure 4.3: Frequency delineation based on areas of strong regularity

### 4.1.4 – Prediction of coefficients

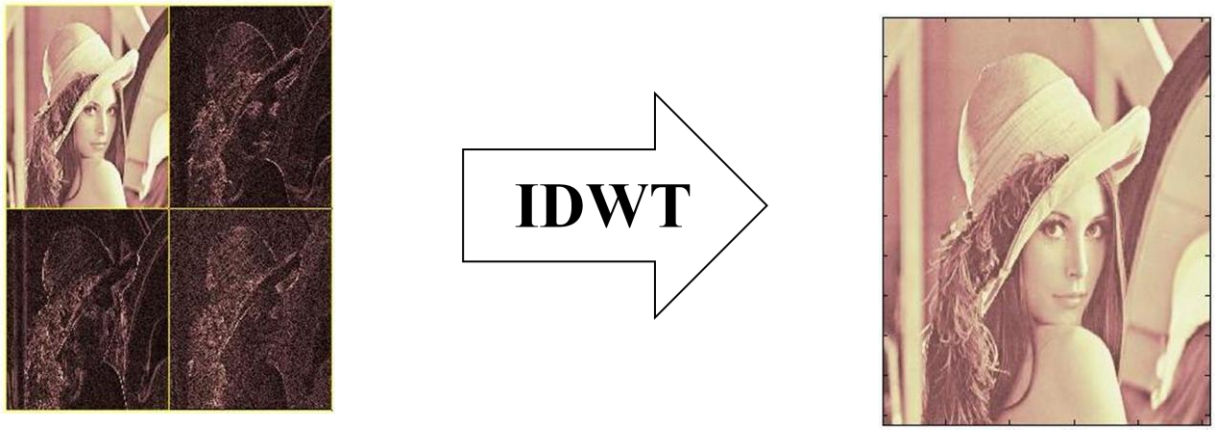
The next higher set of coefficients of the wavelet sub-bands is predicted using the regularity relations of the coefficients across scales within the S3 cluster, the steps of the prediction is presented in Figure 4.4 below.

(1.) The original LR input image:

(2.) The input LR image becomes the approximation coefficients for generating the HR image.

(3.) The detail coefficients are predicted using the regularity of the coefficients from S3 cluster.

(4.) The predicted coefficients and the approximation are used to generate HR image.



(5.) The predicted coefficients are refined using the comparison of the standard, resolution invariant features in the generated HR images and the original LR input.

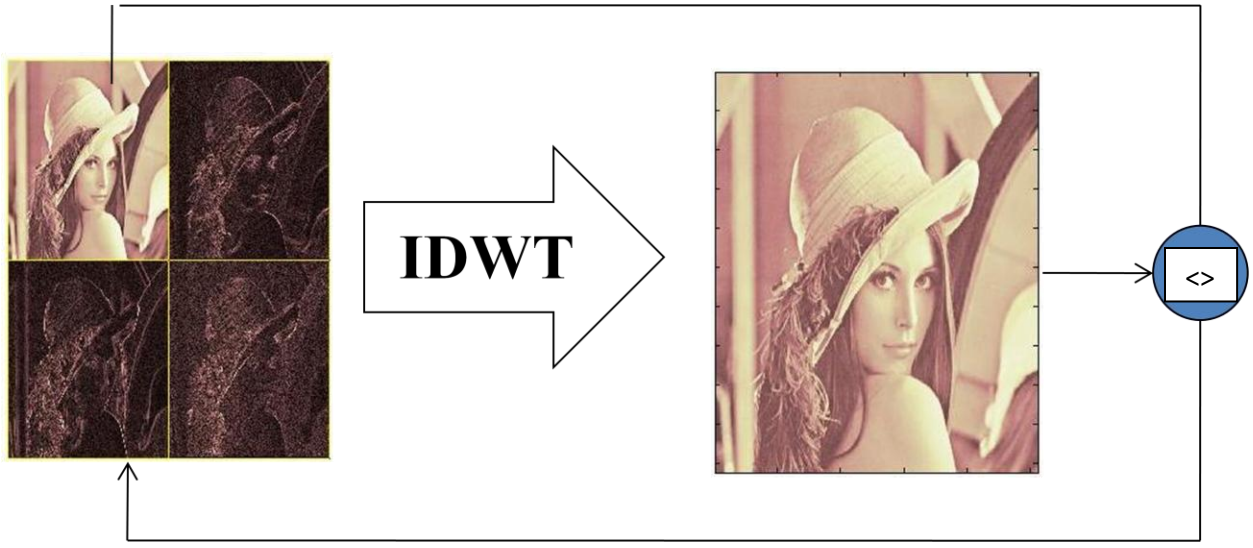



Figure 4.4: The steps for prediction and refinement of the unavailable detail coefficients.

The edge strength is used in the implementation; the refinement process is restricted to selection of the best comparative result for the standard feature within the range of the magnitude of the (preceding) coarser coefficients in the same position.

## 4.2 Temporal Super Resolution

The implementation of the Video Temporal Super resolution is also a five step process as presented below:

(1.) Redundant Discrete Wavelet Transform of input frames
(2.) Inter-frame (temporal) filtering of RDWT´ed frames
(3.) Application of a value threshold on the results of (2.)
(4.) Inverse RDWT of the results of (3.) and the Inverse RDWT of the results of (3.) with the approximation coefficients made equal to zero. The two results of (4.) is added together to produce the motion profile of the adjacent frames.

(5.) The motion profile is used deterministic search and matching across the input frames for estimation of the differences and displacement vectors.

The individual steps of the process is presented in the next sections

### 4.2.1 – Redundant Discrete Wavelet Transform of input frames

This is the single decomposition of the input frames into wavelet sub-bands, similar to the first step in image super resolution technique. However temporal super resolution receives two input frames.

### 4.2.2 – Temporal (inter-frame) Filtering of the wavelet sub-bands

The output of the first step for the input frames is temporally filtered using the Haar integer reversible high pass wavelet filter, as shown in Figure 4.5 below:

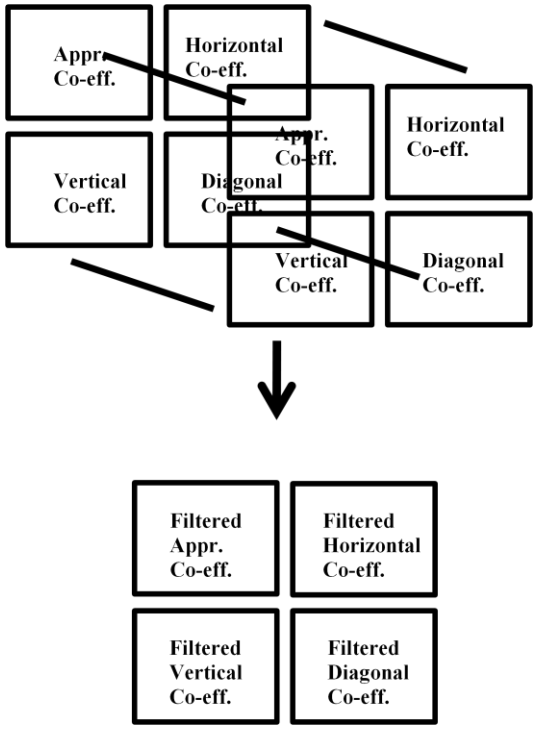



Figure 4.5: Temporal filtering of the wavelet sub bands, derived from the input frames.

### 4.2.3 – Threshold classification

The application of threshold classification to the temporally filtered coefficients, is used to eliminate the near zero values. The hard magnitude threshold is applied to the output.

### 4.2.4 – Inverse DWT and Inverse DWT with zero values in approximation coefficients

Inverse discrete wavelet transform of the output of step 3 above, and the output an inverse discrete wavelet with the filtered approximation coefficients replaced with zero values is summed up to produce the motion profile of the two frames, all non zero values in the motion profile are areas of relative motion across the frames, this is shown in Figure 4.6.

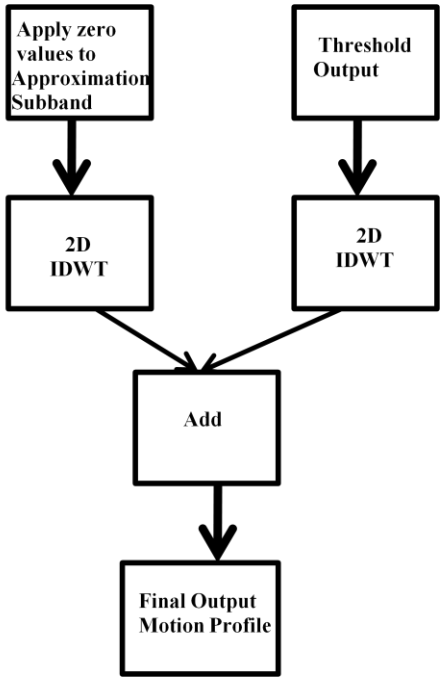



Figure 4.6: Generation of the changes across two input frames

**4.2.5 – The motion profile is used for deterministic search and matching across the input frames for estimation of the displacement vectors.**

The estimation of differences and displacement vectors across the input frames is restricted to the areas of changes across the frames, as defined by the motion profile. The objective criteria and the shape and size of the matching units (regions or blocks) used are user defined, any method can be implemented. An example is shown in Figure 4.7 below:

Figure 4.7: An example of estimation of areas of changes across two input frames

## 4.3 Video Super Resolution

The key modifications for optimal combination of the two techniques are presented in the following sub sections.

### 4.3.1 Optimal Combination of the Spatial and Temporal techniques for Video Super Resolution

The optimal combination of the techniques for efficient and possible real-time video super resolution is implemented by adding the following modifications to the steps in the spatial and temporal super resolutions. The following modifications are implemented:

1. The image (spatial domain) feature constraints is restricted to standard features (Edge Strength, Direction and Continuity), therefore eliminating automatic detection process, however only the edge strength feature constrain is implemented, additional optimizations can be achieved by streamlining the constrain implementation.
2. The estimation of the regularity (or growth) is eliminated to optimize for speed, requiring only single wavelet decomposition (RDWT).

3. The predicted coefficients are therefore directly derived from the last available scale.
4. The refinement process is implemented using the binary search method.

The resulting optimally combined spatial (image) and temporal super resolution is a three step process as shown in Figure 4.8 and 4.9:

Step 1. Single decomposition

Step 2. Refinement and Temporal Filtering

Step 3. Apply the Refinement factor and perform Inverse DWT

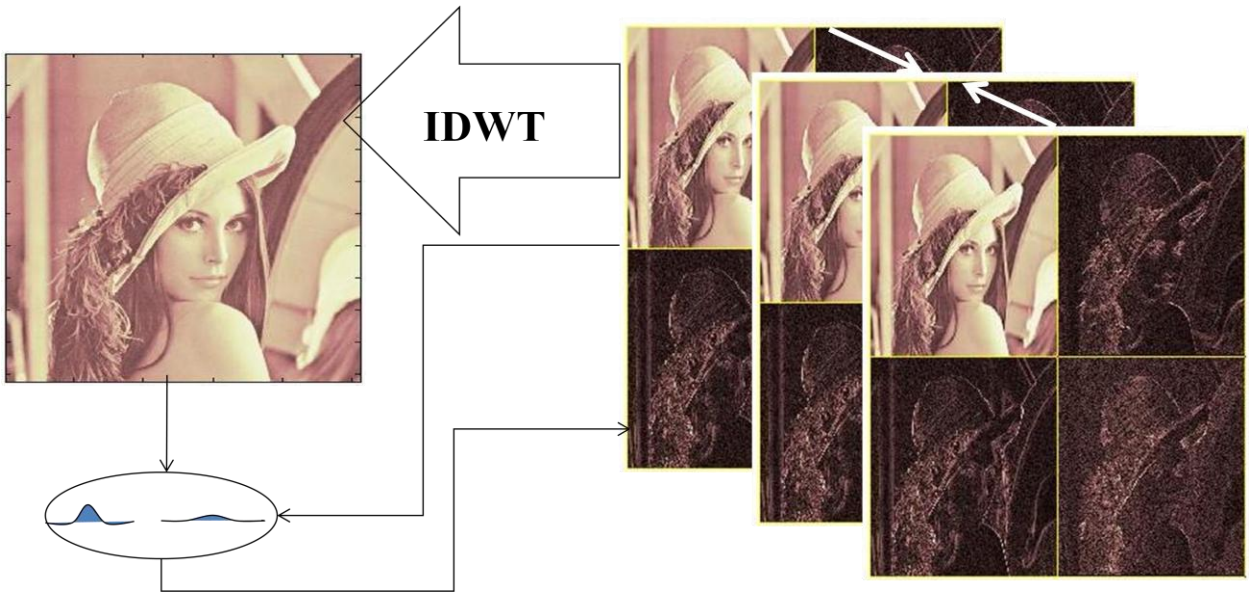



Figure 4.8: The implementation of the three step process for video super resolution


Figure 4.9: The final step of the optimal implementation of video super resolution

## 4.4 Summary

The implementation steps of the image super resolution algorithm support resource efficient offline or online applications, with several tunable parameters. However the detection of resolution invariant features was not implemented, this can be achieved for offline applications using dual frequency approach, where correlation of the frequency of the wavelet coefficients across resolutions can be extracted. The temporal resolution method provided an efficient extraction of motion across frames; however the choice of wavelet filters (CDF9/7), is solely based on the need to conform to the image super resolution step in order to support optimal combination of the two algorithms.

# Chapter Five: Experimental Analysis and Results

## 5.0 Introduction

The test and analysis of the performance of the proposed video super resolution is presented in two sections following the order established throughout this thesis, the first sections provides details of the test environment and benchmarks for image super resolution while the second section provides the comparative analysis of motion estimation functions of the temporal super resolution against other similar algorithms. The tests are presented in the following sections:

## 5.1 Test and Performance Analysis of Image Super Resolution

The establishment of baseline or benchmark for performance analysis of image super resolution propositions follows a general rule that tends to adopt bi-linear and/or bi-cubic interpolation. However this is inadequate for wavelet based methods [28], as all wavelet based methods will easily outperform bi-linear and bi-cubic interpolation, therefore a more accurate model for comparison is the objective measurement of the peak signal to noise ratio of generated HR image based on predicted coefficients against the zero valued coefficients.

The zero-valued coefficients provide the maximal proximity of the image at that resolution to the target HR version; this can be easily proved using any other spectral methods. The resulting HR estimate from the zero-valued detail coefficients version (shown in Figure 5.1) provides the equivalence of increasing the spatial domain resolution (more accurately dimensions) without the attendant increase in the spectral content.

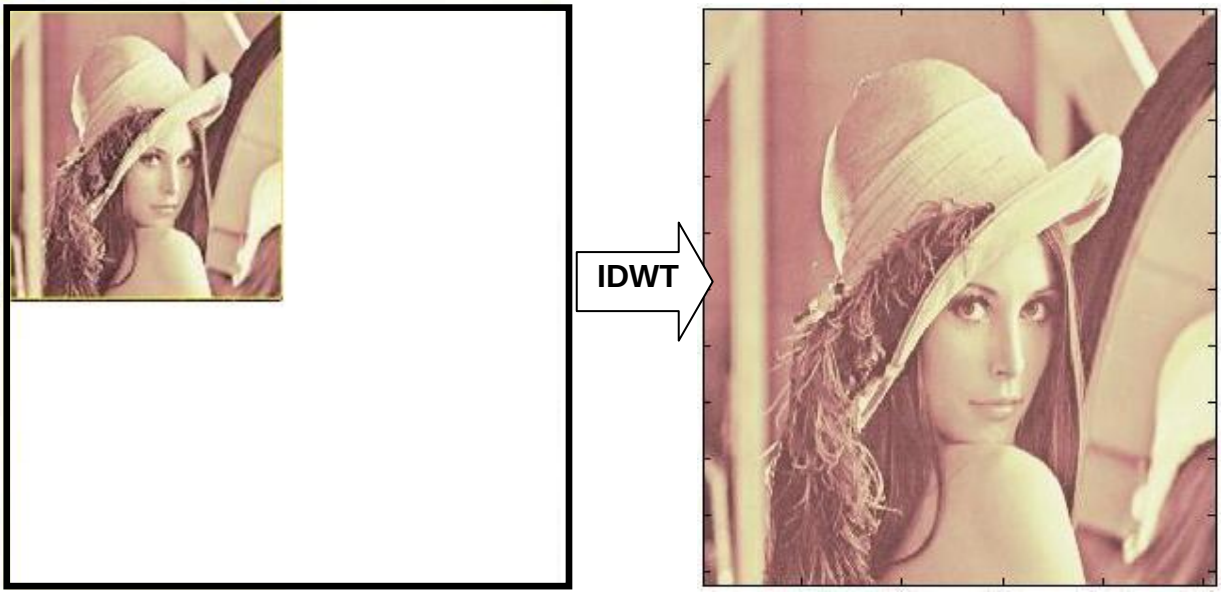



Figure 5.1: The generated High Resolution image based on the zero-valued detail coefficients

This provides the required benchmark for algorithms that attempt to introduce additional spectral information through prediction and other methods. These propositions or algorithms MUST outperform the zero-valued detail coefficient HR image in order to lay claim to any useful contribution and/or additional insight in the quest for improving/enhancing image super resolution. However outperforming zero-valued HR version is a very difficult and daunting task, this stems from the fact that predicting the unavailable information lost through under-sampling with any degree of certainty is almost technically untenable. For example (Figure 5.2) using the LR Lenna image of resolution 256x256 and generating a higher resolution version (512x512), will require the prediction of 3x256x256 detail coefficients, ALL the predicted values MUST be within the range of the actual values in order to outperform the zero-based version of the HR image, or else the prediction of the coefficients will result in a deterioration. The fluidity of the process makes the odds for accurate prediction very low as small errors, perturbations and outliers will result in degradation and an objective performance that lags the zero-valued baseline.

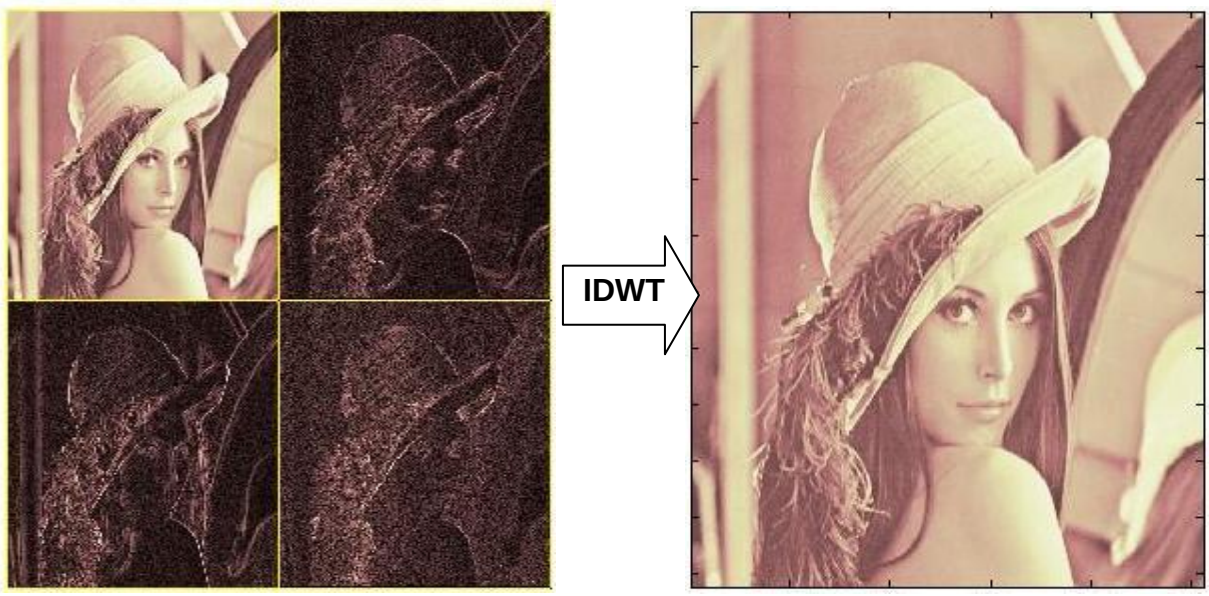



Figure 5.2: The prediction of coefficients for wavelet based super resolution

Three categories of tests are used in the performance analysis. (1.) The comparison of the generated high resolution image against a reference image. The reference and the low resolution image are obtained by direct capture. (2.) In the second case, the low resolution image is obtained by down-sampling a high resolution image to a low resolution one using bi-cubic interpolation with anti-aliasing filter. (3.) In the last category the results of other algorithms [29, 30, 31] are compared using the similar evaluation metrics (images and resolutions) as contained in the documentation.

For the first category of tests, a snap shot of a computer desktop screen was captured at three resolutions using 24bit color. The second and third resolutions being half and quarter of the first respectively, these were up-scaled and compared with the first resolution in three colour components. The results are presented in table 5.1.

In the second category of tests, a 24 bit colour image of Lena (Lenna) was down-sampled, and the algorithm was similarly evaluated, the result is presented in table 5.2.

In comparison with other algorithms, two 8 bit grayscale images, boat and bridge [32] was used in the evaluation and the result is presented in table 3 for the third category. The overall performance of the algorithm in the results are marginal in comparison to ‘zero-valued wavelet’ but substantial, compared to other methods (wavelet and non-wavelet), the adoption of zero-valued wavelet as benchmark is meant to prove an actual (and definite) increase in the resolution. The proposed super resolution technique consistently outperformed the zero-valued counterpart which validates the approach taken in this research

and provides a guarantee of performance in the reconstruction of lost signal details. Several other constraints like edge continuity and spatial similarity (or inter-pixel relationships) within groups of pixels is the focus of current research for the extension of the proposed method. The technique can be applied to any under-sampled signal for the recovery of lost detail if there exists a set of constraints that can be used to delimit the solution space.

Table 5.1: Results of first category of tests

| Image:Desktop | 320 x 240 → 640x480 | |
|---|---|---|
| Colour Index | Zero-Valued | Wavelet Proposed |
| 1. | 34.91dB | 35.26dB |
| 2. | 35.43dB | 35.86dB |
| 3. | 34.24dB | 34.72dB |
| | 160 x 120 → 640x480 | |
| 1. | 32.50dB | 32.73dB |
| 2. | 32.74dB | 32.96dB |
| 3. | 32.41dB | 32.78dB |

Table 5.2: Results of second category of tests

| Image: Lena (Figure 3) | 256 x 256 → 512x512 | |
|---|---|---|
| Colour Index | Zero-Valued | Wavelet Proposed |
| 1. | 35.23dB | 35.61dB |
| 2. | 35.22dB | 35.46dB |
| 3. | 35.73dB | 36.07dB |
| | 128 x 128 → 512x512 | |
| 1. | 32.77dB | 33.04dB |
| 2. | 32.56dB | 32.75dB |
| 3. | 33.58dB | 33.67dB |

Table 5.3: Results of third category of tests

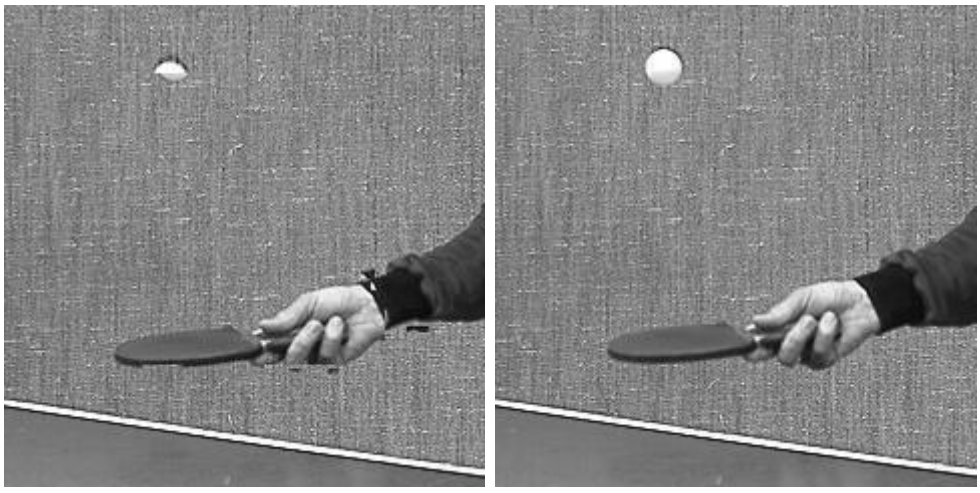
| Method | 256 x 256 → 512x512 | |
|---|---|---|
| | Boat | Bridge |
| Proposed | 33.85dB | 31.56dB |
| Zero-Valued | 33.63dB | 31.49dB |
| Vandewalle[31] | 25.97dB | 23.21dB |
| Tian / Ma [30] | 26.89dB | 25.18dB |
| Tian/Ma [29] | 27.40dB | 25.68dB |
| Kinebuchi[33] | 29.12dB | - |
| Dong [34] | 31.05dB | - |
| | 128 x 128 → 512x512 | |
| Proposed | 31.62dB | 29.95dB |
| Zero-Valued | 31.43dB | 29.87dB |
| Vandewalle[31] | 23.57dB | 22.10dB |
| Tian / Ma [30] | 24.78dB | 22.93dB |
| Tian/Ma [29] | 25.27dB | 23.52dB |

## 5.2 Video Temporal Super Resolution

The central point of the temporal super resolution proposition is the timely, efficient and accurate determination of the moving/changing areas across two input frames, the subsequent motion vector interpolation for motion compensated prediction of an intermediate frame is somewhat inconsequential to

the performance of the technique. Therefore the algorithm is evaluated in terms of objective quality of the predicted frames. In this scenario, a set of temporally related video frames are sub-sampled, and the missing frame is predicted using the proposed interpolation method. The quality of this prediction is evaluated against the sub-sampled frame. A group of eight (8) temporally related images from [35] shown in Appendix A (A1.1) is used for the evaluation. The table of quality and output frame is presented in Figures 5.3.

Additionally, an interpolation example using the images from [32] and the proposed algorithm is presented in Figure 5.4. In this case, two input frames from the sequence without subsampling are used for the prediction. The results are visually (or perceptually) close to the expected higher frame rate (resolution) version. The proposed algorithm provides close interpolated prediction of the frames of the sub-sampled



| Frame Number | PSNR Value |
|---|---|
| 2 | +34.95dB |
| 4 | +31.76dB |
| 6 | +32.39dB |

Figure 5.3: The predicted interpolated frame and the actual frame (frame 6), the PSNR qualities of frames

frames; however, the method is not resilient to occlusions as the quality of the prediction dips significantly in occluded portion of the frames. The proposed method provides a computationally efficient temporal interpolation, but the existence of visual artifacts and noise reduces the quality; however the performance can be improved based on the proposed algorithmic structure.

Figure 5:4: Interpolated frames from the temporally related sequence, showing noise and occlusion effect

The distortion from mild noise is due to sub-optimal implementation of the algorithm and lack of extensive/rigorous code validation. However the implementation serves the purposes of academic research and can be extended into commercial package for real-time temporal resolution up-conversion.

## 5.3 Summary

The implementation validated the proposed algorithms for video spatial and temporal super resolutions; the edge strength feature constraint can be extended to include edge continuity and direction. The temporal super resolution provided pseudo-object segmentation approach then tends avails of all the

benefits of the object centered motion detection but eliminates the prohibitive computational cost associated with object segmentation.

# Chapter Six: Conclusion

## 6.0 Introduction

The super resolution algorithms provided a technique to recover with certainty a degree of the lost high frequency information in under-sampled images, however further work on preprocessing the images for aliasing problems and the development of additional constraints based on the characteristics of the signals will improve the recovery results. The following conclusions drawn the experimentation is presented in the following sections.

## 6.1 Image Super Resolution

The proposed technique for image super resolution provided a confirmation that lost information due to under-sampling can be recovered with deterministic but severely limited accuracy from under studying the available information and their inter-resolution relationships. However the scheme can be improved by developing robust resolution invariant feature detection algorithms, in addition the model can be extended to support optional high level information, in this scheme, several portions of the image can be automatically identified as real world image detail like texts or objects and hence used as an input in the application of the constraints. More so automatic detection of resolution invariant features of an image could be explored for offline line applications like forensics and astronomy.

## 6.2 Video Temporal Super Resolution

The temporal super resolution algorithm provides a threshold in temporal resolution up conversion and additional programming effort could translate the algorithm into standard library for video processing as none currently exists. The method could be optimized for robust matching of object motion resilient to occlusions and complex motion.

## 6.3 Summary

The proposed algorithms for video super resolution provided a set of techniques that could be tuned for several application of image and video super resolution; however additional work on pre-processing of images for aliasing problems and post processing of frames for noise is required.

## References

[1] Meijering, E. “A chronology of interpolation: from ancient astronomy to modern signal and image processing”, Proceedings of the IEEE, March 2002, Vol. 90, Issue 3, pp. 319–342.

[2] Thiele, T. N. Interpolationsrechnung, B. G. Teubner, Leipzig, 1909. (Available at: http://www.archive.org/details/Interpolationsrechnung)

[3] Parker, J. Anthony Kenyon, Robert V. Troxel, Donald E. “Comparison of Interpolating Methods for Image Re-sampling” IEEE Transactions on Medical Imaging, March 1983, Vol. 2, Issue: 1, page(s): 31-39, Davis, CA, USA.

[4] Thevenaz, P. Blu, T. and Unser, M. “Interpolation Revisited,” IEEE Transactions on medical imaging, July 2000, Vol. 19, Issue 7, pp. 739-758.

[5] Keys, R. “Cubic convolution interpolation for digital image processing”. IEEE Transactions on Signal Processing, Acoustics, Speech, and Signal Processing, (1981)

[6] Unser, M. “Splines, A perfect fit for Signal and Image Processing”, IEEE Signal Processing Magazine, November 1999

[7] Schoenberg, I.J. “Contribution to the problem of approximation of equidistant data by analytic functions”, Quarterly Applied Mathematics, Vol. 4, pp.45-99, 112-141, 1946 (Available at I. J. Schoenberg: Selected Papers By I. J. Schoenberg, Carl De Boor, Google Books)

[8] Sid-Ahmed, M.A. “Real-time television image pixel multiplication methods and apparatus”, United States Patent, September 16, 1997, Patent Number: 5668602

[9] Barnsley, M. F., Jacquin, A. E., “Application of recurrent iterated function systems to images”, Proc. SPIE Visual Comm. and Image Processing (1988)

[10] Jacquin, A. E., “Fractal image coding: a review”, Proc. IEEE, Vol. 81, No. 10, pp. 1451-1465 (1993)

[11] Wohlberg, B., de Jager, G., “A review of fractal image coding literature”, IEEE Trans. Image Processing, Vol. 8, No. 12, pp. 1716-1729 (1999)

[12] Davis, G. “Self-quantized wavelet sub trees: a wavelet-based theory for fractal image compression,” DCC, pp.232, IEEE Data Compression Conference (DCC '95), 1995

[13] Kinebuchi, K. Muresan, D.D. Parks, T.W., “Image interpolation using wavelet based hidden Markov trees”, IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001. Vol. 3, page(s): 1957-1960, Salt Lake City, UT, USA

[14] Dong Him Woo, Il Kyn Eom, Yoo Shin Kim, “Image Interpolation based on inter-scale dependency in wavelet domain” International Conference on Image Processing, 2004. Vol. 3, page(s): 1687- 1690

[15] Vladan Velisavljevic, “Edge-preservation resolution enhancement with oriented wavelets”, Proceedings of the International Conference on Image Processing, ICIP 2008, October 12-15, 2008, San Diego, California, USA. IEEE 2008

[16] Yu-Len Huang, Ruey-Feng Chang, “MLP interpolation for digital image processing using wavelet transform” IEEE International Conference on Acoustics, Speech, and Signal Processing,15-19 Mar 1999, Vol. 6, page(s): 3217-3220, Phoenix, AZ, USA

[17] Jing Tian and Kai-Kuang Ma, “Edge-Adaptive Super-Resolution Image Reconstruction Using A Markov Chain Monte Carlo Approach” 6th International Conference on Information, Communications & Signal Processing, 10-13 Dec. 2007, page(s): 1-5, Singapore

[18] Renxiang Li, Bing Zeng, and Ming L. Liou, “A New Three-Step Search Algorithm for Block Motion Estimation”, IEEE Trans. Circuits And Systems For Video Technology, Vol. 4., no. 4, pp. 438-442, August 1994.

[19] Jianhua Lu, and Ming L. Liou, “A Simple and Efficient Search Algorithm for Block-Matching Motion Estimation”, IEEE Trans. Circuits And Systems For Video Technology, Vol. 7, no. 2, pp. 429-433, April 1997

[20] Lai-Man Po, and Wing-Chung Ma, “A Novel Four-Step Search Algorithm for Fast Block Motion Estimation”, IEEE Trans. Circuits And Systems For Video Technology, Vol. 6, no. 3, pp. 313-317, June 1996.

[21] Shan Zhu, and Kai-Kuang Ma, “A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation”, IEEE Trans. Image Processing, Vol. 9, no. 2, pp. 287-290, February 2000.

[22] Zhang, Y.-Q. and Zafar, S. “Motion-compensated wavelet transform coding for color video compression,” IEEE Trans. Circuits Syst. Video Technol., Vol. 2, pp. 285-296, September 1992.

[23] Secker, A. and Taubman, D. “Motion-compensated highly scalable video compression using an adaptive 3d wavelet transform based on lifting,” Proceedings of ICIP 2001, Vol. 2, pp. 1029-1032, Thessaloniki, Greece, Oct. 2001.

[24] Conklin, G. and Hemami, S. "Multi-Resolution Motion Estimation," ICASSP, vol. 4, pp.2873, 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'97) – Vol. 4, 1997

[25] Park, H. W., and Kim H.S., "Motion Estimation Using Low-Band-Shift Method for Wavelet-Based Moving-Picture Coding", IEEE Transactions on Image Processing, Vol. 9, No. 4, April 2000

[26] Daubechies, I. et al, The wavelet transform, time-frequency localization and signal analysis, IEEE Transactions on Information Theory, Vol. 36, Issue 5, Pages 961-1005, September 1990.

[27] Mallat, S. “A wavelet tour of signal processing”, Elsevier Science & Technology Books, September 1999

[28] Xin Li, “Image Resolution Enhancement via Data-Driven Parametric Models in the Wavelet Space”, EURASIP Journal on Image and Video Processing, Volume 2007, Article ID 41516

[29] Jing Tian and Kai-Kuang Ma, "Edge-Adaptive Super-Resolution Image Reconstruction Using A Markov Chain Monte Carlo Approach" 6th International Conference on Information, Communications & Signal Processing, 10-13 Dec. 2007, page(s): 1-5, Singapore

[30] Tian, J. and Ma, K.-K. “A MCMC approach for Bayesian super resolution image reconstruction,” in Proc. IEEE Int. Conf. on Image Processing, Genoa, Italy, Sept. 2005, pp. 45–48.

[31] Vandewalle, P., Susstrunk, S. and M. Vetterli, “A frequency domain approach to registration of aliased images with application to super resolution,” EURASIP Journal on Applied Signal Processing, vol. 2006, Article ID 71459, 2006.

[32] Signal & Image Proc. Institute, Electrical Eng., University of Southern California (http://sipi.usc.edu/database/)

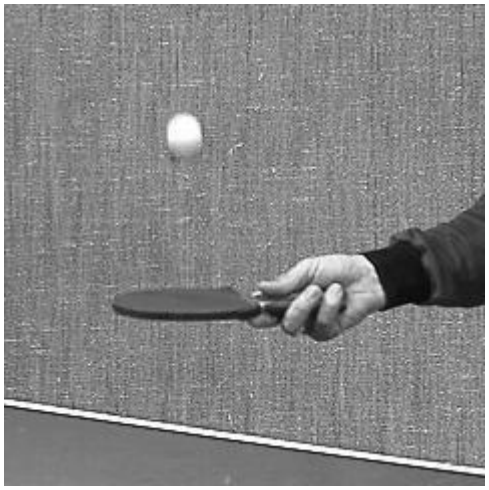
[33] Kinebuchi, K. Muresan, D.D. and Parks, T.W., “Image interpolation using wavelet based hidden Markov trees,” IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001. Vol. 3, page(s): 1957-1960, Salt Lake City, UT, USA

[34] Dong Him Woo, Il Kyn Eom and Yoo Shin Kim, "Image Interpolation based on inter-scale dependency in wavelet domain," International Conference on Image Processing, 2004. Vol. 3, page(s): 1687- 1690

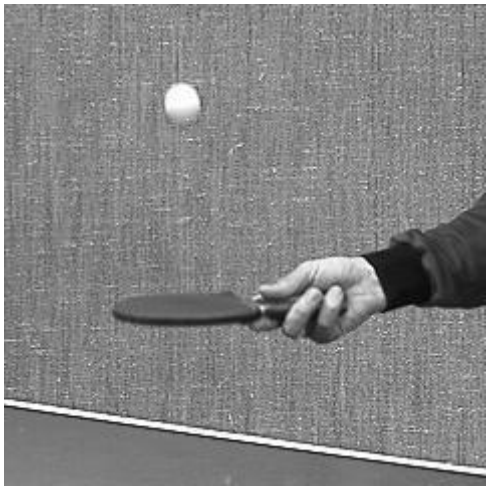
[35] Xiph.org Test Media (which is available at: http://media.xiph.org/video/derf/)

# Appendix A

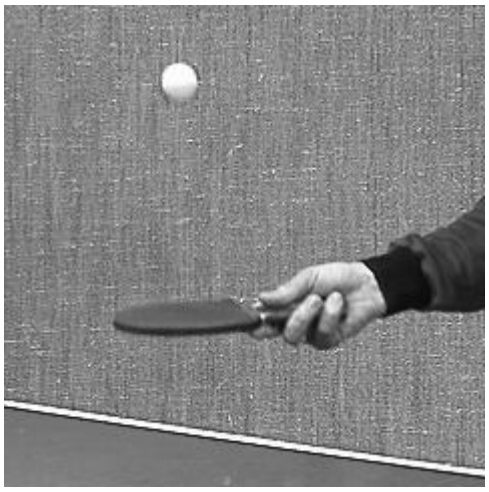
A.1: Temporally related images for inter-frame change evaluation



(1)



(2)



(3)

(4)

(5)

(6)

(7)

(8)

# Appendix B

## B.1: Clarifications of terms used in the thesis

**φ(t)** – is the wavelet function, it is generic representation of continuous time wavelet function used as standard notation in wavelet literature for theoretical analysis, the proposed design is implemented with discrete wavelet transform.

**Low, High** – the low, high in Figure 3.2 refers to the low pass and high pass filters represented in the figure, the Figure is also a standard wavelet notation for discrete wavelet transform.

**Lipschitz regularity** – the Lipschitz regularity refers the continuous smoothness of the high frequency co-efficients of the wavelet transform; it is measured as a gradient, of the co-efficient distribution across scales, details of Lipschitz regularity is also available in the referenced material [27]

**Haar** – Haar wavelet filter: low pass [0.7071 0.7071] and high pass [0.7071 -0.7071]

**CDF9/7** – The CDF9/7 is defined in section 4.1.1 and it is a popular and widely acknowledged wavelet filter selected for JPEG 2000 image compression format and also used for FBI finger print compression.

**S3** – the S1, S2 and S3 clusters are defined in Figure 3.6 and Figure 4.3

**Integer Haar** – Integer Haar wavelet filter: low pass [1 1] and high pass [1 -1]

**Coarse** – Higher levels of co-efficients in multiple wavelet decomposition

**Fine** – Lower levels (or initially generated co-efficients) in multiple wavelet decomposition

## B.2: Source Code for the Implementation

Three codes are attached: The essential implementation sections of Image Super Resolution using C++, Scaled down version (grayscale version) of Image Super Resolution using Matlab. Temporal Super Resolution using C++.

### B.2.1: The Matlab Implementation of Image Super Resolution

```
% Function for increasing the resolution of an image using redundant
% discrete wavelet transform, the version is scaled down to support only
% grayscale images for simplicity.
function imgwaveletsupres(grayscaleimgfile)
img_lr = imread(grayscaleimgfile);
color_cnt = numel(size(img_lr));

if (color_cnt < 2)
```

```
    disp('Error: Only images or 2D signals are the allowed input');
    return;
elseif (color_cnt == 2)
    [i,j] = size(img_lr);
    imgs(:,:,1) = img_lr;
    nr = 1;
else
    disp('Sorry: Only grayscale images are supported in this version, use the c++ code instead');
    return;  % DO NOT REMOVE THIS LINE!!!
    imgs = img_lr;
    nr = 3;
    [i,j,k] = size(img_lr);
end

rdwtout(:,:,:) = zeros(2*i,2*j,nr);
for clrcnt = 1:nr
   rdwtout(:,:,clrcnt) = RDWTwavecdf97 (imgs(:,:,clrcnt),2);
end

% Find the largest value
maxvalues(:,:,:) = zeros(1,2,nr);
h_v = zeros([1 nr]);
for clrcnt = 1:nr
    %vertical edges
    [vy,vz] = max(abs(rdwtout(i+4:(2*i)-4,1+4:j-4,clrcnt))); % +4 and -4 to search within for maximum,
and eliminate border edge point
    [vy,vzk] = max(max(abs(rdwtout(i+4:(2*i)-4,1+4:j-4,clrcnt))));
    % horizontal edges
    [hy,hz] = max(abs(rdwtout(1+4:i-4,j+4:(2*j)-4,clrcnt)));
    [hy,hzk] = max(max(abs(rdwtout(1+4:i-4,j+4:(2*j)-4,clrcnt))));

    h_v(clrcnt) = 1; offsetx = 0; offsety = j;
    maxvalues(:,:,clrcnt) = [hz(hzk) hzk];
    if (vy > hy)
```

```
        maxvalues(:,:,clrcnt) = [vz(vzk) vzk];
        h_v(clrcnt) = 0;
        offsetx = i; offsety = 0;
    end

end


% Test for regularity using one color component
cntr = 1;
orig_img = imgs(:,:,cntr);
Noofdecomp = 3; % Decompose three times, as shown in thesis Figure 3.4 and 3.5
decompoutput(:,:,:) = zeros(2*i,2*j,nr*Noofdecomp);

for numberofdecomposition = 1:Noofdecomp
     decompoutput(:,:,numberofdecomposition) = RDWTwavecdf97 (orig_img,2);
     orig_img = decompoutput(1:i,1:j,numberofdecomposition);
end

if ( log2( abs(decompoutput(maxvalues(1,1,1)+offsetx,maxvalues(1,2,1)+offsety,2)) -
abs(decompoutput(maxvalues(1,1,1)+offsetx,maxvalues(1,2,1)+offsety,1)) ) < log2(
abs(decompoutput(maxvalues(1,1,1)+offsetx,maxvalues(1,2,1)+offsety,3)) -
abs(decompoutput(maxvalues(1,1,1)+offsetx,maxvalues(1,2,1)+offsety,2)) ))
    verycoarseestimate =
decompoutput(maxvalues(1,1,1)+offsetx,maxvalues(1,2,1)+offsety,2)/decompoutput(maxvalues(1,1,1)+o
ffsetx,maxvalues(1,2,1)+offsety,1);
else
    disp('Pre-requirement was not met, sorry, try another image');
    return;
end
% Calculate the variance-to-mean of the original
vvar(:,:,:) = zeros(1,9,nr);
mvar(:,:,:) = zeros(1,9,nr);
```

```
for clrcnt = 1:nr
   if (h_v(clrcnt) == 0)
      %vvar = var(double(imgs(maxvalues(1,1,clrcnt)-4:maxvalues(1,1,clrcnt)+4,
maxvalues(1,2,clrcnt),clrcnt)));
      %mvar = mean(imgs(maxvalues(1,1,clrcnt)-4:maxvalues(1,1,clrcnt)+4,
maxvalues(1,2,clrcnt),clrcnt));
      vvar(:,:,clrcnt) = imgs(maxvalues(1,1,clrcnt)-4:maxvalues(1,1,clrcnt)+4,
maxvalues(1,2,clrcnt),clrcnt);
   else
      %vvar = var(double(imgs(maxvalues(1,1,clrcnt), maxvalues(1,2,clrcnt)-
4:maxvalues(1,2,clrcnt)+4,clrcnt)));
      %mvar = mean(imgs(maxvalues(1,1,clrcnt), maxvalues(1,2,clrcnt)-
4:maxvalues(1,2,clrcnt)+4,clrcnt));
      vvar(:,:,clrcnt) = imgs(maxvalues(1,1,clrcnt), maxvalues(1,2,clrcnt)-
4:maxvalues(1,2,clrcnt)+4,clrcnt);
   end

end

for clrcnt = 1:nr
   for itr=1:8
      mvar(1,itr,clrcnt)= (var(double(vvar(1,itr:itr+1,clrcnt))))/mean(vvar(1,itr:itr+1,clrcnt));
   end
end

% zero valued
zimgs(:,:,:) = zeros(2*i,2*j,nr);
zrdwtout(:,:,:) = zeros(2*i,2*j,nr);

% predicted values
pimgs(:,:,:) = zeros(2*i,2*j,nr);
prdwtout(:,:,:) = zeros(2*i,2*j,nr);

for clrcnt = 1:nr
```

```
  % zero valued
  zimgs(1:i,1:j,clrcnt) = imgs(1:i,1:j,clrcnt);
  zrdwtout(:,:,clrcnt) = RDWTwavecdf97 (zimgs(:,:,clrcnt),-1);
end

% predict and refine till the best is obtained
searchquest = 0;
pvvar(:,:,:) = zeros(1,9,nr);
pmvar(:,:,:) = zeros(1,9,nr);

minval = 255*ones([1 nr]); ratefactor = verycoarseestimate*ones([1 nr]); finalfactor =
verycoarseestimate*ones([1 nr]);
allcolorsmatched = zeros([1 nr]);
direction_change = ones([1 nr]);

while (searchquest == 0)

   for clrcnt = 1:nr
      % predicted values
      pimgs(:,:,clrcnt) = ratefactor(clrcnt)*rdwtout(:,:,clrcnt);
      pimgs(1:i,1:j,clrcnt) = imgs(1:i,1:j,clrcnt);
      prdwtout(:,:,clrcnt) = RDWTwavecdf97 (pimgs(:,:,clrcnt),-1);

   end

   % Calculate the variance to mean ratio of the predicted

   for clrcnt = 1:nr
      if (h_v(clrcnt) == 0)
         %pvvar = var(double(prdwtout((maxvalues(1,1,clrcnt)*2)-4:(maxvalues(1,1,clrcnt)*2)+4,
maxvalues(1,2,clrcnt)*2,clrcnt)));
         %pmvar = mean(prdwtout((maxvalues(1,1,clrcnt)*2)-4:(maxvalues(1,1,clrcnt)*2)+4,
maxvalues(1,2,clrcnt)*2,clrcnt));
```

```
        pvvar(:,:,clrcnt) = prdwtout((maxvalues(1,1,clrcnt)*2)-4:(maxvalues(1,1,clrcnt)*2)+4,
maxvalues(1,2,clrcnt)*2,clrcnt);
      else
        %pvvar = var(double(prdwtout(maxvalues(1,1,clrcnt)*2, (maxvalues(1,2,clrcnt)*2)-
4:(maxvalues(1,2,clrcnt)*2)+4,clrcnt)));
        %pmvar = mean(prdwtout(maxvalues(1,1,clrcnt)*2, (maxvalues(1,2,clrcnt)*2)-
4:(maxvalues(1,2,clrcnt)*2)+4,clrcnt));
        pvvar(:,:,clrcnt) = prdwtout(maxvalues(1,1,clrcnt)*2, (maxvalues(1,2,clrcnt)*2)-
4:(maxvalues(1,2,clrcnt)*2)+4,clrcnt);
      end
   end

   for clrcnt = 1:nr
      for itr=1:8
        pmvar(1,itr,clrcnt)= (var(double(pvvar(1,itr:itr+1,clrcnt))))/mean(pvvar(1,itr:itr+1,clrcnt));
      end
   end

   for clrcnt = 1:nr
      if ( (abs(pmvar(1,6,clrcnt)-pmvar(1,7,clrcnt))) < minval(clrcnt) && mvar(1,5,clrcnt) <
sum(pmvar(:,:,clrcnt)))
          minval(clrcnt) = abs(pmvar(1,6,clrcnt)-pmvar(1,7,clrcnt));
          finalfactor(clrcnt) = ratefactor(clrcnt);
      else
        if (direction_change(clrcnt) == 1)
          direction_change(clrcnt) = -1;
          ratefactor(clrcnt) = finalfactor(clrcnt);
        else
          allcolorsmatched(clrcnt) = 1;
        end
      end
      ratefactor(clrcnt) = ratefactor(clrcnt) + (0.1*direction_change(clrcnt));
   end
    if ( (sum(allcolorsmatched)/nr) == 1)
```

```
          for clrcnt = 1:nr
            % predicted values
              pimgs(:,:,clrcnt) = finalfactor(clrcnt)*rdwtout(:,:,clrcnt);
              pimgs(1:i,1:j,clrcnt) = imgs(1:i,1:j,clrcnt);
              prdwtout(:,:,clrcnt) = RDWTwavecdf97 (pimgs(:,:,clrcnt),-1);
           end
          break;

    end
end
% zero-valued
imwrite(mat2gray(ceil(zrdwtout)),'zerovalued.bmp', 'BMP');
% predicted
imwrite(mat2gray(ceil(prdwtout)), 'predicted.bmp', 'BMP');

disp('Done! Two files created zerovalued.bmp and predicted.bmp for the zero and predicted co-efficients
images');

end
```

B.2.1.1: Required Function for the Matlab Code [Source: Matlab Exchange, Tianhui Wang]

```
function c = RDWTwavecdf97(x, nlevel)
%WAVECDF97: Multi-level discrete 2-D wavelet transform
%with the Cohen-Daubechies-Feauveau (CDF) 9/7 wavelet.
%
% c = wavecdf97(x, nlevel) does the follows according to the value of
%   nlevel:
%   nlevel > 0:   decomposes 2-dimension matrix x up to nlevel level;
%   nlevel < 0:   does the inverse transform to nlevel level;
%   nlevel = 0:   sets c equal to x;
%   omitted:      does the same as nlevel=5.
%
% The boundary handling method is symmetric extension.
```

```
%
% x may be of any size; it need not have size divisible by 2^L.
%   For example, if x has length 9, one stage of decomposition
%   produces a lowpass subband of length 5 and a highpass subband
%   of length 4.  Transforms of any length have perfect
%   reconstruction (exact inversion).
%   NOTE: the 5 lines above are quoted directly form [3].
%
% If nlevel is so large that the approximation coefficients become
%   a 1-D array, any further decomposition will be performed as for 1-D
%   decomposition until the approximation coefficients be a scale number.
%
% Lifting algorithm is not used here; we use subband filters directly.
%   Lifting algorithm and spline 5/3 wavelets and other jpeg2000 related
%   codes will be available soon.
%
% Example:
%   Y = wavecdf97(X, 5);    % Decompose image X up to 5 level
%   R = wavecdf97(Y, -5);   % Reconstruct from Y
%
% You can test wavecdf97.m with the following lines:
%   % get a 2-D uint8 image
%   x=imread('E:\study\jpeg2000\images\lena.tif');
%   % decompose
%   y=wavecdf97(x,2);
%   % show decomposed result
%   figure;imshow(mat2gray(y));
%   % reconstruct without change of anything
%   ix=wavecdf97(y,-2);
%   % show and compare the original and reconstructed images
%   figure;subplot(1,2,1);imshow(x);subplot(1,2,2);imshow(uint8(ix));
%   % look at the MSE difference
%   sum(sum((double(x)-ix).^2))/numel(x)
%
```

```
% Reference:
%   [1] D.S.Taubman et al., JPEC2000 Image Compression: F. S. & P.,
%       Chinese Edition, formula 10.6-10.9 in section 10.3.1
%       and formula 10.13 in section 10.4.1.
%   [2] R.C.Gonzalez et al., Digital Image Processing Using MATLAB,
%       Chinese Edition, function wavefast in section 7.2.2.
%   [3] Pascal Getreuer, waveletcdf97.m from Matlab file Exchange website
%   [4] Matlab files: biorwavf.m, wavdec2.m, wawrec2.m, etc.
%
% Contact information:
%   Email/MSN messenger:  wangthth@hotmail.com
%
% Tianhui Wang at Beijing, China,   July, 2006
%                 Last Revision:   Aug 5, 2006

%---------------------- input arguments checking  ----------------------%
error(nargchk(1,2,nargin));
if nargin == 1
   nlevel = 5; % default level
end
% check x
if ~isreal(x) || ~isnumeric(x) || (ndims(x) > 2)
   error('WAVELIFT:InArgErr', ['The first argument must' ...
       ' be a real, numeric 2-D or 1-D matrix.']);
end
if isinteger(x)
   x = double(x);
end
% check nlevel
if ~isreal(nlevel) || ~isnumeric(nlevel) || round(nlevel)~=nlevel
   error('WAVELIFT:InArgErr', ['The 2nd argument shall be ' ...
       'a real and numeric integer.']);
end
%---------------- forming low-pass and high-pass filters ---------------%
```

```
% CDF 9/7 filters: decomposition low-pass lp and high-pass hp
%                  reconstruction low-pass lpr and high-pass hpr
% The filter coefficients have several forms.
% What D.S.Taubman et al. suggest in [1] are used here:
lp = [.026748757411 -.016864118443 -.078223266529 .266864118443];
lp = [lp .602949018236 fliplr(lp)];
hp = [.045635881557 -.028771763114 -.295635881557];
hp = [hp .557543526229 fliplr(hp)];
lpr = hp .* [-1 1 -1 1 -1 1 -1] * 2;
hpr = lp .* [1 -1 1 -1 1 -1 1 -1 1] * 2;
% Matlab 'bior4.4' use the varied version (see Matlab's biorwavf.m):
%  lp=lp*sqrt(2);hp=hp*(-sqrt(2));lpr=lpr*(1/sqrt(2));hpr=hpr*(-1/sqrt(2));
% P.Getreuer's waveletcdf97.m [3] alters the Taubman's version as follows:
%  lp=lp*sqrt(2);hp=hp*sqrt(2);lpr=lpr*(1/sqrt(2));hpr=hpr*(1/sqrt(2));
% while R.C.Gonzalez et al in [2] alter the Taubman's version as follows:
%  lp=lp;hp=hp*(-2);lpr=lpr;hpr=hpr*(-1/2);
%----------------  remain unchanged when nlevel = 0  -------------------%
if nlevel == 0
   c = x;
%--------------------  decomposition,  if nlevel < 0  ------------------%
elseif nlevel > 0
   c = zeros(size(x));
   x = double(x);
   for i = 1:nlevel
      % [ll, hl; lh, hh]: 1-level FWT for x
      temp = symconv2(x, hp, 'col');    % high filtering
      % temp = temp(2:2:end, :);         % down sampling
      hh = symconv2(temp, hp, 'row');   % high filtering
      % hh = hh(:, 2:2:end);             % down sampling
      lh = symconv2(temp, lp, 'row');   % low filtering
      % lh = lh(:, 1:2:end);             % down sampling

      temp = symconv2(x, lp, 'col');    % low filtering
      % temp = temp(1:2:end, :);         % down sampling
```

```
        hl = symconv2(temp, hp, 'row');   % high filtering
       % hl = hl(:, 2:2:end);              % down sampling
        ll = symconv2(temp, lp, 'row');   % low filtering
       % ll = ll(:, 1:2:end);              % down sampling
        % update coefficient matrix
        c=nlevel*c;
        c(1:nlevel*size(x,1), 1:nlevel*size(x,2)) = [ll, hl; lh, hh];
        % replace x with ll for next level FWT
        x = ll;
        % give a warning if nlevel is too large
        if size(x,1)<=1 && size(x,2)<=1 && i~=nlevel
            warning('WAVECDF97:DegradeInput', ['Only decompose to ' ...
                num2str(i) '-level instead of ' num2str(nlevel) ...
                ', \nas the approximation coefficients at ' num2str(i) ...
                '-level has row or/and column of length 1.']);
            break
        end
    end
%--------------------  reconstruction,  if nlevel < 0  -----------------%
else
    sx = size(x);
    % find reconstruction level
    nl = -nlevel;
    while sx(1)/2^nl<=1/2 && sx(2)/2^nl<=1/2,  nl = nl-1;  end
    if nl ~= -nlevel
        warning('WAVECDF97:DegradeInput', ['Only reconstruct to ' ...
            num2str(nl) '-level instead of ' num2str(-nlevel) ...
            ', \nas the approximation coefficients at ' num2str(nl) ...
            '-level has row or/and column of length 1.']);
    end
    % nl-level reconstruction
    for i = 1:nl
        % find the target ll hl lh hh blocks
        sLL = ceil(sx/2^(nl-i+1));
```

```
    sConstructed = ceil(sx/2^(nl-i));
    sHH = sConstructed - sLL;
    lrow = sConstructed(1); lcol = sConstructed(2);

    ll = x(1:sLL(1), 1:sLL(2));
    hl = x(1:sLL(1), sLL(2)+1:sLL(2)+sHH(2));
    lh = x(sLL(1)+1:sLL(1)+sHH(1), 1:sLL(2));
    hh = x(sLL(1)+1:sLL(1)+sHH(1), sLL(2)+1:sLL(2)+sHH(2));

    % upsample rows and low filter
    temp = zeros(sLL(1), lcol); temp(:, 1:2:end) = ll;
    ll = symconv2(temp, lpr, 'row');
    % upsample rows and high filter
    temp = zeros(sLL(1), lcol); temp(:, 2:2:end) = hl;
    hl = symconv2(temp, hpr, 'row');
    % upsample columns and low filter
    temp = zeros(lrow, lcol); temp(1:2:end, :) = ll + hl;
    l = symconv2(temp, lpr, 'col');

    % upsample rows and high filter
    temp = zeros(sHH(1), lcol); temp(:, 1:2:end) = lh;
    lh = symconv2(temp, lpr, 'row');
    % upsample rows and high filter
    temp = zeros(sHH(1), lcol); temp(:, 2:2:end) = hh;
    hh = symconv2(temp, hpr, 'row');
    % upsample rows and high filter
    temp = zeros(lrow, lcol); temp(2:2:end, :) = lh + hh;
    h = symconv2(temp, hpr, 'col');

    % update x with the new ll, ie. l+h
    x(1:lrow, 1:lcol) = l + h;
end
% output
c = x;
```

```
end
%------------------------ internal function  --------------------------%
%      2-dimension convolution with edges symmetrically extended       %
%-------------------------------------------------------------------------%
function y = symconv2(x, h, direction)
% symmetrically extended convolution(see section 6.5.2 in [1]):
%    x[n], E<=n<=F-1, is extended to x~[n] = x[n], 0<=n<=N-1;
%                                x~[E-i] = x~[E+i], for all integer i
%                               x~[F-1-i] = x~[F-1+i], for all integer i
%    For odd-length h[n], to convolve x[n] and h[n], we just need extend x
%    by (length(h)-1)/2  for both left and right edges.
% The symmetric extension handled here is not the same as in Matlab
%  wavelets toolbox nor in [2]. The last two use the following method:
%    x[n], E<=n<=F-1, is extended to x~[n] = x[n], 0<=n<=N-1;
%                                x~[E-i] = x~[E+i-1], for all integer i
%                               x~[F-1-i] = x~[F+i], for all integer i

l = length(h); s = size(x);
lext = (l-1)/2; % length of h is odd
h = h(:)'; % make sure h is row vector
y = x;
if strcmp(direction, 'row') % convolving along rows
   if ~isempty(x) && s(2) > 1 % unit length array skip convolution stage
      for i = 1: lext
         x = [x(:, 2*i), x, x(:, s(2)-1)]; % symmetric extension
      end
      x = conv2(x, h);
      y = x(:, l:s(2)+l-1);
   end
elseif strcmp(direction, 'col') % convolving along columns
   if ~isempty(x) && s(1) > 1 % unit length array skip convolution stage
      for i = 1: lext
         x = [x(2*i, :); x; x(s(1)-1, :)]; % symmetric extension
      end
```

```
      x = conv2(x, h');
      y = x(l:s(1)+l-1, :);
   end
end
% EOF
```

**B.2.2 : The Essential sections of the program code for implementation of image super resolution using C++**

```
void Onwavelet_image(CString m_csSourceImage, int lx, int ly, int rx,
int ry)
{
      lx=0; ly=0; rx=height; ry=width;

      vff = 0.0;
      hff = 0.0;
      dff = 0.0;

      // Initialize the GDI+
    GdiplusStartupInput StartupInput;
    ULONG_PTR GdiplusToken = NULL;
    GdiplusStartup( &GdiplusToken,&StartupInput,0 );
      {

            UpdateData();
            int magfactor;
            int magnification[4] = {2, 4, 8, 16};

            int c_size = (ry-ly);
            if ((rx-lx) > (ry-ly)) c_size = rx-lx;

            int w_size = c_size;
            int h_size = c_size;
            // please remove this crap and make the necessary
corrections
            int tmp = ly;
            ly = lx;
            lx = tmp;

            if((ly+c_size) > width)
                  w_size = width-ly;
            if((lx+c_size) > height)
                  h_size = height-lx;
            //
            lx=0; ly=0; rx=height; ry=width;
            w_size = height;h_size = width;
            //
```

```
            if (mag_factor < 0)
            {     magfactor = 2;   }
            else
            {magfactor = magnification[mag_factor];}

            if (magfactor > 2)
            {
                  AfxMessageBox(_T("The magnifaction factor for wavelet
scaling function defaults to 2 for this update"));
                  magfactor = 2;
            }

            // Create the image.
            USES_CONVERSION;
            const wchar_t* wstr_file =
T2W(m_csSourceImage.GetBuffer());
            Bitmap ColorImage(wstr_file);

            int r,g,b;
            int rr_r, gg_g, bb_b;

            int *maxpos_x =  (int *)malloc(3*sizeof(int));
            int *maxpos_y =  (int *)malloc(3*sizeof(int));

            double *maxcoeff = (double *)malloc(3*sizeof(double));

            // Second data storage for calculated co-efficients
            double *ttemp_r = (double
*)malloc(magfactor*magfactor*h_size*w_size*sizeof(double));
            double *ttemp_g = (double
*)malloc(magfactor*magfactor*h_size*w_size*sizeof(double));
            double *ttemp_b = (double
*)malloc(magfactor*magfactor*h_size*w_size*sizeof(double));
            // memory locations
            int i = 0, j = 0, k = 0;

            // the three scales of the wavelet subbands and the
three(3) colour components and the two dimension image co-efficients
            double **** aval = (double ****) malloc(3*sizeof(double
***));
            double **** aprx = (double ****) malloc(3*sizeof(double
***));
            double **** hort = (double ****) malloc(3*sizeof(double
***));
            double **** vert = (double ****) malloc(3*sizeof(double
***));
            double **** diag = (double ****) malloc(3*sizeof(double
***));
            double *** upscale = (double ***) malloc(3*sizeof(double
**));

            for (i = 0; i < 3; i++)
```

```
            {
                  aval[i] = (double ***) malloc(3*sizeof(double **));
                  aprx[i] = (double ***) malloc(3*sizeof(double **));
                  hort[i] = (double ***) malloc(3*sizeof(double **));
                  vert[i] = (double ***) malloc(3*sizeof(double **));
                  diag[i] = (double ***) malloc(3*sizeof(double **));
            }
            for (i = 0; i < 3; i++)
            {
                  for (j = 0; j < 3; j++)
                        {
                              aval[i][j] = (double
**)malloc(h_size*sizeof(double *));
                              aprx[i][j] = (double
**)malloc(h_size*sizeof(double *));
                              hort[i][j] = (double
**)malloc(h_size*sizeof(double *));
                              vert[i][j] = (double
**)malloc(h_size*sizeof(double *));
                              diag[i][j] = (double
**)malloc(h_size*sizeof(double *));
                        }
            }

            for (i = 0; i < 3; i++)
            {
                  for (j = 0; j < 3; j++)
                        {
                              for (k = 0; k < h_size; k++)
                              {
                                    aval[i][j][k] = (double
*)malloc(w_size*sizeof(double));
                                    aprx[i][j][k] = (double
*)malloc(w_size*sizeof(double));
                                    hort[i][j][k] = (double
*)malloc(w_size*sizeof(double));
                                    vert[i][j][k] = (double
*)malloc(w_size*sizeof(double));
                                    diag[i][j][k] = (double
*)malloc(w_size*sizeof(double));
                              }
                        }
            }


            for (i = 0; i < 3; i++)
            {
                  upscale[i] = (double **)
malloc(magfactor*h_size*sizeof(double *));
            }
            i=0;
            for (i=0; i<3; i++)
```

```
            {
                  for (j = 0; j < (magfactor*h_size); j++)
                  {
                        upscale[i][j] = (double
*)malloc(magfactor*w_size*sizeof(double));
                  }
            }
            // memory allocation ended

            // read the image and store in the right positions,
            // Get the pixels
         Color PixelColor;
            int cnts = 0; int xindx,yindx,zindx,m=0,h=0;

            for (xindx = ly; xindx < (ly+h_size); xindx++ )
            {
                  for (yindx = lx; yindx < (lx+w_size); yindx++)
                  {
                        // Get the pixel at x,y
                  ColorImage.GetPixel( xindx, yindx, &PixelColor );
                        // Convert it to COLORREF
                  COLORREF RgbColor = PixelColor.ToCOLORREF();
                        aval[0][0][m][h] = GetRValue(RgbColor);
aval[0][1][m][h] = GetGValue(RgbColor); aval[0][2][m][h] =
GetBValue(RgbColor);
                        cnts++;h++;
                  }
                  m++;h=0;
            }
            cnts = 0;

            // Scale 1
            forward_waveletcdf97
(aval[0][0],aprx[0][0],vert[0][0],hort[0][0],diag[0][0],h_size,
w_size);
            forward_waveletcdf97
(aval[0][1],aprx[0][1],vert[0][1],hort[0][1],diag[0][1],h_size,
w_size);
            forward_waveletcdf97
(aval[0][2],aprx[0][2],vert[0][2],hort[0][2],diag[0][2],h_size,
w_size);
            // Scale 2
            forward_waveletcdf97
(aprx[0][0],aprx[1][0],vert[1][0],hort[1][0],diag[1][0],h_size,
w_size);
            forward_waveletcdf97
(aprx[0][1],aprx[1][1],vert[1][1],hort[1][1],diag[1][1],h_size,
w_size);
            forward_waveletcdf97
(aprx[0][2],aprx[1][2],vert[1][2],hort[1][2],diag[1][2],h_size,
w_size);
            // Scale 3
```

```
            forward_waveletcdf97
(aprx[1][0],aprx[2][0],vert[2][0],hort[2][0],diag[2][0],h_size,
w_size);
            forward_waveletcdf97
(aprx[1][1],aprx[2][1],vert[2][1],hort[2][1],diag[2][1],h_size,
w_size);
            forward_waveletcdf97
(aprx[1][2],aprx[2][2],vert[2][2],hort[2][2],diag[2][2],h_size,
w_size);
            //

            // Get the maximum value for the co-efficients in vertical
or horizontal on scale 1(0 for first decomposition) for the three(3)
colours
            maxcoeff[0] = 0.0; maxcoeff[1] = 0.0; maxcoeff[2] = 0.0;
            int h_v = 1,offsety, offsetx;
            for (xindx = 4; xindx < h_size-4; xindx++ ) // with -+ 4 of
the image borders is to eliminate border edges
            {
                  for (yindx = 4; yindx < w_size-4; yindx++) // with -+
4 of the image borders is to eliminate border edges
                  {
                        for (zindx = 0; zindx < 3; zindx++)
                        {
                              // three(3) colour components 0-2  vertical
subband co-efficients
                              if (abs(vert[0][zindx][xindx][yindx])  >
maxcoeff[zindx])
                              {
                                          maxcoeff[zindx] =
abs(vert[0][zindx][xindx][yindx]);
                                          maxpos_x[zindx] =  xindx;
                                          maxpos_y[zindx] =  yindx;
                                          h_v = 0; offsety=0; offsetx=1;
                              }

                              if (abs(hort[0][zindx][xindx][yindx])  >
maxcoeff[zindx] )
                              {
                                          maxcoeff[zindx] =
abs(hort[0][zindx][xindx][yindx]);
                                          maxpos_x[zindx] =  xindx;
                                          maxpos_y[zindx] =  yindx;
                                          h_v = 1; offsety=1; offsetx=0;
                              }


                        }
                  }
            }
            //
            double coarseprediction = 0.0;
```

```
            //Establish regularity with the assumption that it holds
for all the colours

                                    if (
log2(abs(hort[2][1][maxpos_x[1]][maxpos_y[1]])-
abs(hort[1][1][maxpos_x[1]][maxpos_y[1]])) >
log2(abs(hort[2][1][maxpos_x[1]][maxpos_y[1]])-
abs(hort[1][1][maxpos_x[1]][maxpos_y[1]])))
                                    {
                                      coarseprediction =
abs(hort[2][1][maxpos_x[1]][maxpos_y[1]])/abs(hort[1][1][maxpos_x[1]][
maxpos_y[1]]);
                                    }
                                    else
                                    {
                                          // requirement cannot be
satisfied
                                          return;
                                    }

            // Apply the coarse prediction
            for (xindx = 0; xindx < h_size; xindx++ )
            {
                  for (yindx = 0; yindx < w_size; yindx++)
                  {
                        for (zindx = 0; zindx < 3; zindx++)
                        {
                              hort[0][zindx][xindx][yindx] =
hort[0][zindx][xindx][yindx]*coarseprediction;
                              vert[0][zindx][xindx][yindx] =
vert[0][zindx][xindx][yindx]*coarseprediction;
                              diag[0][zindx][xindx][yindx] =
vert[0][zindx][xindx][yindx]*coarseprediction;
                        }
                  }
            }

            int directionofsearch = 1; // i.e. positive direction
            double minres = 255.0;
            double finalpredrate = 0.0, incrementrate = 0.1;

            while (1)
            {

                  //Perform the scaling of the image
                  // Generate one colour to optmize for speed
                              for (xindx = 0; xindx < h_size; xindx++ )
                                    {
                                          for (yindx = 0; yindx < w_size;
yindx++)
                                          {
```

```
      hort[0][0][xindx][yindx] =
hort[0][zindx][xindx][yindx]*coarseprediction;

      vert[0][0][xindx][yindx] =
vert[0][zindx][xindx][yindx]*coarseprediction;

      diag[0][0][xindx][yindx] =
vert[0][zindx][xindx][yindx]*coarseprediction;

                                        }
                                    }
                  inverse_waveletcdf97(upscale[0], aval[0][0],
vert[0][0], hort[0][0], diag[0][0], h_size, w_size);
                  //inverse_waveletcdf97 (upscale[1], aval[0][1],
vert[0][1], hort[0][1], diag[0][1], h_size, w_size);
                  //inverse_waveletcdf97 (upscale[2], aval[0][2],
vert[0][2], hort[0][2], diag[0][2], h_size, w_size);


                        if (
abs(vmr(upscale[0][(2*maxpos_x[1])+offsetx][(2*maxpos_y[1])+offsety],u
pscale[0][(2*maxpos_x[1])+(2*offsetx)][(2*maxpos_y[1])+(2*offsety)]) -
vmr(upscale[0][(2*maxpos_x[1])+(2*offsetx)][(2*maxpos_y[1])+(2*offsety
)],upscale[0][(2*maxpos_x[1])+(3*offsetx)][(2*maxpos_y[1])+(3*offsety)
])) < minres &&
vmr(aval[0][1][maxpos_x[1]][maxpos_y[1]],aval[0][1][maxpos_x[1]+offset
x][maxpos_y[1]+offsety]) >
(vmr(upscale[0][(2*maxpos_x[1])+offsetx][(2*maxpos_y[1])+offsety],upsc
ale[0][(2*maxpos_x[1])+(2*offsetx)][(2*maxpos_y[1])+(2*offsety)])/aval
[0][1][maxpos_x[1]+offsetx][maxpos_y[1]+offsety]) )
                        {
                              minres =
abs(vmr(upscale[0][(2*maxpos_x[1])][(2*maxpos_y[1])+1],upscale[0][(2*m
axpos_x[1])][(2*maxpos_y[1])+2]) -
vmr(upscale[0][(2*maxpos_x[1])][(2*maxpos_y[1])+2],upscale[0][(2*maxpo
s_x[1])][(2*maxpos_y[1])+3]));
                              finalpredrate = coarseprediction;
                        }
                        else if (directionofsearch == 1)
                        {
                                    directionofsearch = -1;
                                    coarseprediction = finalpredrate;

                        }
                        else
                        {
                                    // end of search
                                    // generate the final image
                                    for (xindx = 0; xindx < h_size;
xindx++ )
                                    {
```

```
                                        for (yindx = 0; yindx < w_size;
yindx++)
                                        {
                                              for (zindx = 0; zindx < 3;
zindx++)
                                              {

      hort[0][zindx][xindx][yindx] =
hort[0][zindx][xindx][yindx]*coarseprediction;

      vert[0][zindx][xindx][yindx] =
vert[0][zindx][xindx][yindx]*coarseprediction;

      diag[0][zindx][xindx][yindx] =
vert[0][zindx][xindx][yindx]*coarseprediction;
                                              }
                                        }
                                  }

                                  inverse_waveletcdf97(upscale[0],
aval[0][0], vert[0][0], hort[0][0], diag[0][0], h_size, w_size);
                                  inverse_waveletcdf97 (upscale[1],
aval[0][1], vert[0][1], hort[0][1], diag[0][1], h_size, w_size);
                                  inverse_waveletcdf97 (upscale[2],
aval[0][2], vert[0][2], hort[0][2], diag[0][2], h_size, w_size);
                                  break;
                            }
                            coarseprediction +=
(incrementrate*directionofsearch);
            }

            // create magnified image
            Bitmap ColorImageMag(magfactor*h_size,magfactor*w_size);
            Color MagPixelColor;
            COLORREF MagRgbColor = MagPixelColor.ToCOLORREF();

            for (xindx = 0; xindx < (magfactor*h_size); xindx++ )
            {
                  for (yindx = 0; yindx < (magfactor*w_size); yindx++)
                  {
                                  // set into image
                                  MagRgbColor =
RGB((BYTE)upscale[0][xindx][yindx],(BYTE)upscale[1][xindx][yindx],(BYT
E)upscale[2][xindx][yindx]);
                                  MagPixelColor.SetFromCOLORREF(
MagRgbColor );
                                  ColorImageMag.SetPixel( xindx, yindx,
MagPixelColor);

                  }

            }
```

```
            CLSID bmpClsid;
            GetEncoderClsid( L"image/bmp", &bmpClsid );
            ColorImageMag.Save( L"predicted.bmp", &bmpClsid, NULL);

            ImgDlg Disp_mag;
            Disp_mag.DoModal();

      }

}
```

**B.2.2.1: The required functions for the program code for implementation of image super resolution using C++**

```
double log2(double x)
{
  static const double xlog = 1.0/log(2.0);
  return log(x)*xlog;
}

double vmr(double x, double y)
{
  double meanval = (x+y)/2.0;
  return ((x-meanval)*(x-meanval))/meanval;
}


int CInterpolateframesDlg::GetEncoderClsid(const WCHAR* format, CLSID*
pClsid)
{
    UINT  num = 0;          // number of image encoders
    UINT  size = 0;         // size of the image encoder array in
bytes

    ImageCodecInfo* pImageCodecInfo = NULL;

     GetImageEncodersSize(&num, &size);
    if(size == 0)
        return -1;  // Failure

    pImageCodecInfo = (ImageCodecInfo*)(malloc(size));
    if(pImageCodecInfo == NULL)
        return -1;  // Failure

    GetImageEncoders( num, size, pImageCodecInfo );

    for(UINT j = 0; j < num; ++j)
    {
        if( wcscmp(pImageCodecInfo[j].MimeType, format) == 0 )
        {
            *pClsid = pImageCodecInfo[j].Clsid;
```

```
            free(pImageCodecInfo);
            return j;  // Success
        }
    }

    free(pImageCodecInfo);
    return -1;  // Failure
}


void inverse_dilatedcdf97 (double ** outputval, double ** inapprox,
double ** invertical, double ** inhorizontal, double ** indiagonal,
int rnum, int cnum )
{
int i,j,k,cnt;

int sz = sizeof(lowpass_analy)/sizeof(double);
sz /= 2;

double **temp1 = (double **)malloc(rnum*sizeof(double *));

for (i=0; i<rnum; i++)
     temp1[i]=(double*)malloc(cnum*sizeof(double));

double **temp2 = (double **)malloc(rnum*sizeof(double *));

for (i=0; i<rnum; i++)
     temp2[i]=(double*)malloc(cnum*sizeof(double));

double **temp3 = (double **)malloc(rnum*sizeof(double *));

for (i=0; i<rnum; i++)
     temp3[i]=(double*)malloc(cnum*sizeof(double));

     // synthesis level 1
     for (j=0; j<cnum; j++)
     {
          for (i=0; i<rnum; i++)
          {


     temp1[i][j]=inapprox[i][j]*lowpass_synth[sz];

     temp2[i][j]=invertical[i][j]*lowpass_synth[sz];

     outputval[i][j]=inhorizontal[i][j]*highpas_synth[sz];

     temp3[i][j]=indiagonal[i][j]*highpas_synth[sz];

                         for (k=1; k<5; k++)
                         {
                                   if ((i-k) >= 0 && (i+k) < rnum)
```

```
                                        {
                                                temp1[i][j] +=
((inapprox[i-k][j]*lowpass_synth[sz-k]) +
(inapprox[i+k][j]*lowpass_synth[sz+k]));
                                                temp2[i][j] +=
((invertical[i-k][j]*lowpass_synth[sz-k]) +
(invertical[i+k][j]*lowpass_synth[sz+k]));
                                                outputval[i][j]
+= ((inhorizontal[i-k][j]*highpas_synth[sz-k]) +
(inhorizontal[i+k][j]*highpas_synth[sz+k]));
                                                temp3[i][j] +=
((indiagonal[i-k][j]*highpas_synth[sz-k]) +
(indiagonal[i+k][j]*highpas_synth[sz+k]));
                                        }
                                        else
                                             break;
                                }

            }
      }
      // data re-packing

            for (i=0; i<rnum; i++)
            {
                        for (j=0; j<cnum; j++)
                        {
                                    //
                                    inhorizontal[i][j] = temp1[i][j] +
outputval[i][j];
                                    indiagonal[i][j] = temp2[i][j] +
temp3[i][j];
                                    //

                        }
            }


      // synthesis level 2

      for (i=0; i<rnum; i++)
      {
                  for (j=0; j<cnum; j++)
                  {
                        temp1[i][j]=inhorizontal[i][j]*lowpass_synth[sz];
                        temp2[i][j]=indiagonal[i][j]*highpas_synth[sz];

                        for (k=1; k<5; k++)
                        {
                                    if ((j-k) >= 0 && (j+k) < cnum)
                                    {
```

```
                                        temp1[i][j] +=
((inhorizontal[i][j-k]*lowpass_synth[sz-k]) +
(inhorizontal[i][j+k]*lowpass_synth[sz+k]));
                                        temp2[i][j] +=
((indiagonal[i][j-k]*highpas_synth[sz-k]) +
(indiagonal[i][j+k]*highpas_synth[sz+k]));

                                  }
                                  else
                                        break;
                            }
                      }

       }

       // data synthesis ends and final output
                  for (i=0; i<rnum; i++)
                        for (j=0; j<cnum; j++)

      outputval[i][j]=temp1[i][j]+temp2[i][j];

}



void forward_waveletcdf97 (double ** inputval, double **
outputapproximation, double ** outputvertical, double **
outputhorizontal, double ** outputdiagonal, int row, int col )
{

      int i=0,j=0,k=0,cnt=0;
      int sz = sizeof(lowpass_analy)/sizeof(double);
      sz /= 2;
      double ** outputapprox = (double **) malloc(row*sizeof(double
*));
      for (i = 0; i < row; i++)
                              outputapprox[i] = (double
*)malloc(col*sizeof(double));

      double ** outputvert = (double **) malloc(row*sizeof(double *));
      for (i = 0; i < row; i++)
                              outputvert[i] = (double
*)malloc(col*sizeof(double));


      // approximation co-efficients  and vertical co-efficients --
subband decomposition - first step
            for (i=0; i<row; i++)
            {
                              for (j = 0; j < col; j++)
                              {
```

```
                                        outputapprox[i][j] =
inputval[i][j]*lowpass_analy[sz]; // remember to refine the code
                                        outputvert[i][j] =
inputval[i][j]*highpas_analy[sz];

                                        for (k=1; k<5; k++)
                                        {
                                              if ((j-k) >= 0 && (j+k) <
col)
                                              // approximation co-
efficients
                                              {     outputapprox[i][j] +=
((inputval[i][j-k]*lowpass_analy[sz-k]) +
(inputval[i][j+k]*lowpass_analy[sz+k]));
                                                    outputvert[i][j] +=
((inputval[i][j-k]*highpas_analy[sz-k]) +
(inputval[i][j+k]*highpas_analy[sz+k]));
                                              }
                                              else
                                                    break;
                                        }
                              }
                  }
      // subband decomposition  --- second step

                  for (j = 0; j < col; j++)
                  {
                                    for (i=0; i<row; i++)
                                    {
                                              outputapproximation[i][j] =
outputapprox[i][j]*lowpass_analy[sz];
                                              outputhorizontal[i][j] =
outputapprox[i][j]*highpas_analy[sz];
                                              outputvertical[i][j] =
outputvert[i][j]*lowpass_analy[sz];
                                              outputdiagonal[i][j] =
outputvert[i][j]*highpas_analy[sz];

                                              for (k=1; k<5; k++)
                                              {
                                                    if ((i-k) >= 0 &&
(i+k) < row)
                                                    {

      outputapproximation[i][j] += ((outputapprox[i-
k][j]*lowpass_analy[sz-k]) +
(outputapprox[i+k][j]*lowpass_analy[sz+k]));

      outputvertical[i][j] += ((outputvert[i-k][j]*lowpass_analy[sz-k])
+ (outputvert[i+k][j]*lowpass_analy[sz+k]));
                                                                  //
```

```
      outputhorizontal[i][j] += ((outputapprox[i-
k][j]*highpas_analy[sz-k]) +
(outputapprox[i+k][j]*highpas_analy[sz+k]));

      outputdiagonal[i][j] += ((outputvert[i-k][j]*highpas_analy[sz-k])
+ (outputvert[i+k][j]*highpas_analy[sz+k]));
                                                      }
                                                      else
                                                            break;

                                                }

                                          }
                  }


}
```

**B.2.3.1: The program code for implementation of motion area extraction using C++**

```
void MotionAreaExtract(CString first_file, CString second_file,
CString output_file)
{
      progressval.SetPos(5);

      clock_t start, finish;
      double  duration;
      CString outtime;

      // Initialize the GDI+
    GdiplusStartupInput StartupInput;
    ULONG_PTR GdiplusToken = NULL;
    GdiplusStartup( &GdiplusToken,&StartupInput,0 );

    // Code Block for Bitmap to get destroyed before
GdiplusShutdown().
    {
            int h,i,j, wi,hi, im, jn;
            int blcksize = 16;

            USES_CONVERSION;
            const wchar_t* first_frame = T2W(first_file.GetBuffer());
            const wchar_t* second_frame = T2W(second_file.GetBuffer());
            const wchar_t* output_frame = T2W(output_file.GetBuffer());

            // Create the first image.
```

```
            Bitmap ColorImage(first_frame);
            width = ColorImage.GetWidth();
            height = ColorImage.GetHeight();

            // create the second image
            Bitmap ColorImage2(second_frame);
            wi = ColorImage2.GetWidth();
            hi = ColorImage2.GetHeight();

            if (wi != width || hi != height)
            {     AfxMessageBox(_T("Error!: File size mismatch, the
files MUST be of the same dimensions"), MB_ICONERROR );     return;
      }

            // create memory
            double *** vidframes = (double ***) malloc(10*sizeof(double
**));

            for (h = 0; h < 10; h++)
                  vidframes[h] = (double **) malloc(height*sizeof(double
*));

            for (h = 0; h < 10; h++)
                  for (i = 0; i < height; i++)
                        vidframes[h][i] = (double *)
malloc(width*sizeof(double));
            // // Get the pixels
         Color PixelColor, PixelColor2;
            progressval.SetPos(10);

            for (i = 0; i < height; i++)
            {
                  for (j = 0; j < width; j++)
                  {
                        ColorImage.GetPixel(i,j, &PixelColor );
                        ColorImage2.GetPixel(i,j, &PixelColor2 );
                        COLORREF RgbColor = PixelColor.ToCOLORREF();
                        COLORREF RgbColor2 = PixelColor2.ToCOLORREF();
                        vidframes[0][i][j] = (GetRValue(RgbColor)*0.299)
+ (GetGValue(RgbColor)*0.587) + (GetBValue(RgbColor)*0.114);
                        vidframes[5][i][j] = (GetRValue(RgbColor2)*0.299)
+ (GetGValue(RgbColor2)*0.587) + (GetBValue(RgbColor2)*0.114);
                  }
            }
            progressval.SetPos(20);

            start = clock();
            forward_waveletcdf97 (vidframes[0], vidframes[1],
vidframes[2], vidframes[3], vidframes[4], height, width );
            forward_waveletcdf97 (vidframes[5], vidframes[6],
vidframes[7], vidframes[8], vidframes[9], height, width );
```

```
            progressval.SetPos(50);
            // Temporal filtering

            for (i = 0; i < height; i++)
            {
                  for (j = 0; j < width; j++)
                  {
                        vidframes[1][i][j] = vidframes[1][i][j] -
vidframes[6][i][j];
                        vidframes[1][i][j] = (abs(vidframes[1][i][j]) >
35 ) ? vidframes[1][i][j] : 0;
                        vidframes[2][i][j] = vidframes[2][i][j] -
vidframes[7][i][j];
                        vidframes[2][i][j] = (abs(vidframes[2][i][j]) >
35 ) ? vidframes[2][i][j] : 0;
                        vidframes[3][i][j] = vidframes[3][i][j] -
vidframes[8][i][j];
                        vidframes[3][i][j] = (abs(vidframes[3][i][j]) >
35 ) ? vidframes[3][i][j] : 0;
                        vidframes[4][i][j] = vidframes[4][i][j] -
vidframes[9][i][j];
                        vidframes[4][i][j] = (abs(vidframes[4][i][j]) >
35 ) ? vidframes[4][i][j] : 0;
                        //

                  }
            }


            progressval.SetPos(60);

            for (i = 0; i < height; i++)
            {     for (j = 0; j < width; j++)
                  {

                        vidframes[9][i][j]=0.0;
                        vidframes[8][i][j]=0.0;
                        //vidframes[1][i][j]=0.0;
                  }
            }

            //inverse_dilatedcdf97 (vidframes[9], vidframes[1],
vidframes[7], vidframes[8], vidframes[5], height, width );
            //--
            inverse_dilatedcdf97 (vidframes[6], vidframes[1],
vidframes[2], vidframes[3], vidframes[4], height, width );
            inverse_dilatedcdf97 (vidframes[9], vidframes[8],
vidframes[2], vidframes[3], vidframes[4], height, width );

            finish = clock();
            duration = (double)(finish - start) / CLOCKS_PER_SEC;
```

```
            // the frames for the prediction
                  for (i=0; i<height; i++)
                  {
                        for (j=0; j<width; j++)
                        {
                                    vidframes[7][i][j] =
vidframes[0][i][j];
                                    vidframes[8][i][j] =
vidframes[0][i][j];
                                    vidframes[6][i][j] =
abs(vidframes[6][i][j] + vidframes[9][i][j]);
                              }
                  }

                  int blkrow, blkcol, matchedi, matchedj; double madval,
lowestval;// making lowestval = infinity
                  //calculate the number of blocks and search per block
                  blkrow = height/blcksize;
                  blkcol = width/blcksize;
                  int st_h,en_h,st_w,en_w;


            Bitmap ColorImageMag(height, width);
            Color MagPixelColor;
            COLORREF MagRgbColor = MagPixelColor.ToCOLORREF();

            progressval.SetPos(80);

            for (i = 0; i < height; i++)
            {
                  for (j = 0; j < width; j++)
                  {

vidframes[6][i][j] = (vidframes[6][i][j] > 0 ) ? vidframes[5][i][j] :
0;
MagRgbColor = RGB((BYTE)vidframes[6][i][j], (BYTE)vidframes[6][i][j]
,(BYTE)vidframes[6][i][j]);


      MagPixelColor.SetFromCOLORREF( MagRgbColor );
      ColorImageMag.SetPixel(i,j,  MagPixelColor);
                  }
            }

            progressval.SetPos(85);

            CLSID bmpClsid;
            GetEncoderClsid( L"image/bmp", &bmpClsid );
            ColorImageMag.Save( output_frame, &bmpClsid, NULL);

            progressval.SetPos(90);
```

```
      }
    // Shutdown the Gdi+
    GdiplusShutdown( GdiplusToken );

      outtime.Format(_T("Processing Complete! in %f
seconds"),duration);

      //result_text = L"Processing Complete!";
      result_text = outtime;
      UpdateData(FALSE);
      progressval.SetPos(100);


}
```

# VITA AUCTORIS

Michael Chukwu was born in 1977 in Aba, Nigeria. He graduated from High School in 1994; and obtained Bachelors' Degree at Enugu State University of Technology, Nigeria from 1995 to 2000. Graduated with a Masters Degree (Magna cum laude) at University of Jyväskylä, Finland from August 2005 to December 2007 and in January 2008 enrolled in a second Masters Degree in the area of Digital Image/Video Processing at University of Windsor. Submitted his thesis for review in August 2009 and hopes to defend and graduate in September 2009.