Use of Data Compression Techniques For Optimizing queries in an RFID Network

Shekhar Babanrao Teke

A thesis submitted to

Auckland University of Technology
in partial fulfilment of the requirements for the degree of
Master of Computer and Information Sciences (MCIS)

18th April 2012

Primary Supervisor: Dr. Russel Pears

School of Computing & Mathematical Sciences

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a University or other institution of higher learning, except where due acknowledgement is made in the acknowledgements.

Printed name: SHEKHAR BABANRAO TEKE

Signature:

Date:

Abstract

This Research investigates the effectiveness of Radio Frequency Identification (RFID) technology in navigating through a network of RFID tags which helps to direct a given person to a desired destination. Due to storage capacity constraints, it is not feasible to store exact information in the form of a map on every tag in the network. As such, each tag contains only a partial map which contains the greatest level of detail on the locations of its immediate neighbors and a lower level of detail on locations of tags that are outside of its neighborhood. This storage scheme enables a person equipped with an RFID reader to decide which neighboring tag should be chosen next on the path towards the eventual destination.

The research applies different compression schemes based on the Haar Wavelet techniques and investigates the effect of these on the length of the path to the desired destination.

Acknowledgment

I wish to express my gratitude towards my supervisor, Dr Russel Pears for his patience, kind support, guidance and advice given throughout this research, without whom I could not able to complete the study.

I am also sincerely thankful to my friends, colleagues to encourage me and support me in completion of my study. I am also thankful to staff of the School of Computing and Mathematical Science for their kindness and encouragement during my study. Lastly, I am thankful to AUT library that provides me all literatures in terms of books, e-journals which are helpful in my study.

Table of Contents

List of Tables

List of Figures

Chapter 1 : Introduction

1.1 Research Background

The volume of data processed by computers continues to grow in size. As data grows in size, processing time increases as a result. In general, storage devices have increased in capacity to cope with increased demand, some applications exist where the storage devices have very limited capacity as they were designed for applications that do not require mass storage devices. One such storage device is an RFID tag which is commonly used to track products and stores summary information such as product code and price. In this research we envisage using RFID tags to store location information. A collection of such RFID tags, with the application of a suitable data compression mechanism will then encode information that will enable a user to navigate a path from a given starting point to a desired destination without the use of a conventional terrestrial map.

This research thus addresses the problem of how best to store sparse maps on tags so as enable a user equipped with an RFID reader to select the shortest possible path from source to destination. Each tag is encoded with only a partial map which contains the greatest level of detail on the locations of tags within a given radial distance from any given tag (henceforth referred to in this thesis as the neighborhood) and a lower level of detail on locations of tags that are outside of its neighborhood. Thus each node will have the information to efficiently assess the most promising paths to the target node from any given node without the need to exhaustively explore all possible paths. Once the compression algorithm is in place, an experimental approach will be used to evaluate navigation efficiency. A number of experiments will be run and the sensitivity of key parameters on performance such as the number of neighbors and the wavelet thresholding level will be investigated. The performance metric that will be utilized is the number of hops necessary to traverse the network from a given source node to a given destination node.

The navigation problem stems from the need to design an algorithm that will be able to cope with a sparse, rather than a detailed terrestrial map. This research investigate the use of various forms of the well-known Haar wavelet and use an experimental approach to assess the effectiveness of the storage schemes in facilitating navigation in the network.

1.2 Motivation

Global Positioning System (GPS) has proved to be a very useful and effective tool to locate positions in an outdoor environment. It has been used with a great success by travellers who wish to navigate to destinations whose location is unknown. Many such applications exist in an indoor environment, the most common of which is product location in a physical warehouse. Another application is a pathfinder for disabled individuals who wish to move to a given unknown location from a given starting point. The reliability of GPS technology in indoor and underground environments is poor as it is a satellite-based technology that has a very low signal in such environments. As such, wireless technology has been implemented in such environments and provides more accurate and cost effective solutions than GPS based solutions.

Recently, Radio Frequency Identification (RFID) technology has achieved great success in the supply chain management system and tracking systems (Domdouzis, K., et. al. 1999, and Ward, M., et. al., 2006). It can also be effectively used in Indoor Location – Aware systems (Lionel, M. N., et. al., 2004 and Tesoriero, R., et. al., 2008) The main reason behind the emergence of RFID technology is its ability to recognize and track the movement of objects. RFID technology does not need any human intervention and has good tolerance to environmental factors such as temperature (Want, R., 2006). RFID technology is very cost effective (Potdar, M., et. al., 2006, Want, R., 2006 and Ward, M., et. al., 2006).

Though RFID has numerous advantages, its main limitation is its storage capacity. An RFID tag has a very limited storage capacity and thus a full indoor exact map cannot be stored in a single tag. As such, compression needs to be introduced to enable an RFID tag to encode sufficient information to support the navigation process in an efficient manner (Fazzinga, B., et. al., 2009). There are number of different compression techniques to available that can potentially be used. Our preference in this research is for the Haar wavelet coefficient compression technique as it has been shown to provide high compression ratios and is also easy to implement (Raviraj, P., et. al., 2007).

Thus the main objective of the study can be phrased as the research question - ***How can the Haar Wavelet technique be applied to compress and store a sparse version of a terrestrial map in an RFID tag in order to support efficient navigation***. In order to answer this question the research will involve two major activities: Firstly designing a compression scheme using the Haar wavelet that will use a multi resolution encoding scheme to store information on neighbors and more distant nodes while taking into account the severe storage constraints of an RFID tag; and secondly, a navigation algorithm that is capable of using inexact location information to plot the shortest possible path from source to destination.

1.3 Organization of the Thesis

This thesis is organized into following chapters. Chapter 1 has presented an introduction to the research and provided a motivation for undertaking the research.

Chapter 2 provides a literature review on RFID and different data compression techniques. In first half, it will discuss about RFID technology and its applications. Then, it will discuss different data compression techniques available like, Huffman technique, Arithmetic Coding, Wavelet transform, MPEG and JPEG.

Chapter 3 elaborates Haar Wavelets. One dimensional and two dimensional Haar transformation methods will be discussed thoroughly with some examples. It will also discuss about storing sparse data using a wavelet transformation.

Chapter 4 discusses about research methodologies used in the research study. It contains concept and selection of research methodologies like constructive research and experimental testing. It also discusses four different scenarios used in the research which helps to define the exact nature of the experiments that will be conducted.

Chapter 5 presents different experimental results obtained from the research study. Each scenario gives rise to different compression outcomes and uses a navigation algorithm to traverse the network. These experimental results are then discussed in the Chapter 6.

The last chapter presents concluding remarks, strengths and limitations of the study. It also discusses the application of the algorithms developed in a real world scenario and future research directions.

Chapter 2 : Literature Review

2.1 Introduction

This chapter presents an overview of RFID technology. The different types of tags and their properties from the viewpoint of storage capacity and communication capability are discussed. This is followed by a discussion of the major applications of RFID technology. Previous research in the area of data compression covering techniques such as Huffman Coding, Arithmetic coding and Wavelets are then discussed.

2.2 RFID Technology Overview

Radio Frequency Identification (RFID) technology is a wireless technology which detects electromagnetic signals (Domdouzis, K., et. al., 2007). It consists of three main components viz., RFID tag, Antenna and an RFID Reader. A RFID tag works as a transponder which transmits radio signals while the RFID Reader acts as a transceiver which reads radio signals sent by a RFID tag. There are two types of tags, Active and Passive. Active tags have in-built batteries for power while Passive tags do not have an in-built battery. This type of tag gets power from electromagnetic fields generated by an RFID reader. Active tags have an in-built battery and do not depend on the electromagnetic field generated by the reader. This type of tag has a wide range of activation while passive tags have a relatively low range as it depends on an RFID reader to generate an electromagnetic field.

Ward, M., et. al. (2006) describes five classes of RFID tags. They are :

Class	*Class Layer Name*	*Functionality*
1	Identity Tags	Purely passive, identification tags
2	Higher Functionality Tags	Purely passive, identification + some additional functionality (e.g. read/write memory)
3	Semi-Passive Tags	Addition of on-board battery power
4	Active *'ad hoc'* Tags	Communication with other active tags
5	Reader Tags	Able to provide power for and communicate with other tags i.e. can act as a reader, transmitting and receiving radio waves

Source: Ward, M., Van Kranenburg, R., (May 2006). "RFID: Frequency, standards, adoption and innovation." JISC Technology and Standards Watch.

Storage of data is also a point of difference between an Active tag and a Passive tag. On-chip storage capacity is different in these tags. Active tags have more storage capacity than that of Passive tags. Read-only tags can store only unique identification numbers permanently. This memory is called as a WORM – Write-Once-Read-Many memory. Read-Write tags are capable

of storing tag Id and user's data in which the user can change additional data stored in the memory.

Passive tags normally have a non-volatile memory ranging from 64 bits to 1 kilobyte and Active tags can store data up to 128 kilobytes. (Ward, M., et. al., 2006). A modern UHF Passive tag can store up to 32 KB of data (Pais, S. and Symonds, J.,2011).

2.3 Applications of RFID Technology

RFID technology has successfully been implemented in various scientific and technical fields like medicine and engineering. In the medical field, RFID technology has been used in blood transfusion and analysis. The RFID tag can be used to store the medical history of the patient and then this tag may be attached to the patient's wristband. RFID technology can be used in civil engineering such as on-site inspection support systems and for tracking of building materials (Domdouzis, K., et. al., 2007).

Tesoriero, R., et. al., (2008) describes the use of RFID technology to support Indoor-Location Awareness system and demonstrates how both active and passive tag environments help to find the location of mobile devices in closed spaces. This system has been deployed to assist visitors in finding information on artifacts in a museum. Each visitor is provided with a PDA at the entrance of the museum. A mobile application is installed on the PDA. Museum visitors use the PDA to communicate with the environment that has RFID tags installed close to exhibits throughout the museum. The visitor can select either between auto-navigation and manual modes of opeartion. Manual-navigation mode makes use of cursor keys on the mobile device (ie. PDA) to navigate in the museum building while auto-navigation mode automatically detects the visitor position and gets correct information about the museum as per the PDA location.

Lionel, M. N., et. al., (2004) describes the LANDMARC, a location sensing system using RFID technology for use in inside buildings. The main advantage of this system is to improve the accuracy of finding a location by deploying reference tags. The reference tags store information about all nearest neighbor tags and each reference tag stores information about four nearest neighbour tags. The experiment results in favour of RFID technology and found that active tags are viable and cost effective in indoor navigation systems.

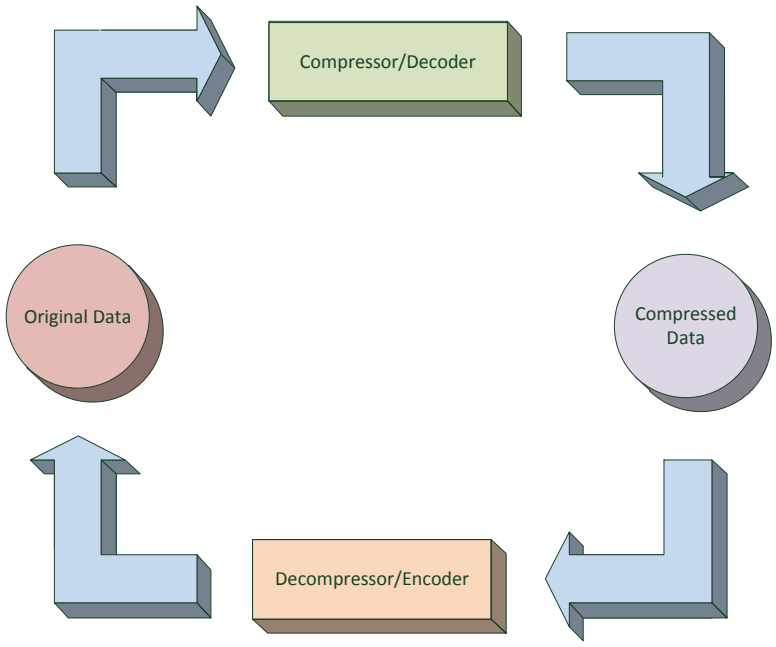
Fazzinga, B., et. al., (2009) addresses the data compression issue in the RFID to enable devices with less memory to execute queries on RFID data warehouse. They developed a lossy algorithm for collapsing tuples carrying items' information which are to be delivered at various locations in the supply chain system. Without connecting to main data warehouse, the user with the device like PDA can do data analysis by supplying them compressed data and allow them to

do data analysis with some acceptable approximation. The algorithm reads the data request from RFID reader and then on request, it compresses the requested data and sends it to the user's PDA. After receiving the required information on PDA, the user then processes it locally on a PDA without connecting to the main server.

RFID tags can be successfully implemented in inventory and warehouse management system (Potdar, M., et. al., 2006). Tracking entry and exit of the material is the biggest and challenging tasks which can be handled by RFID technology. Material tracking and smart shelving are two main requirements of any inventory management system where RFID can be useful. When the raw material is received and sent to warehouse, then RFID tags can be used to store material information like product name, vendor, received date etc. This facilitates the manager to trace the material and certify that the material is sent by the approved vendor. RFID tags can also be fixed on the finished goods to maintain the track of all goods dispatching the warehouse and also useful in maintaining the correct stock of finished goods. Smart shelf is – *they know what they are carrying*. RFID tags are useful to locate the products from long distance in a large warehouse.

2.4 Data Compression Techniques

Data compression is the process of reducing the data size by removing redundant data from the input data. Original data is replaced by the encoded data by using an encoder and a decoder uses the same model to reproduce the original data from the encoded string. There exists different data compression techniques that are used in practice. These techniques are used on text data, audio, video data and finally on hybrid data types. As mentioned are two vital components to any data compression technique – the compressor/encoder and the decompressor/decoder.



Balevic, A., et. al., (2008) classified data compression techniques into two categories: lossless and lossy. Lossless data compression technique gives exact reconstruction of original data whereas lossy techniques give acceptable distorted or perceptively lossless representation of the original data. Talukdar, K.H. and Harada, K. (2007) used another classification – *Lossless vs Lossy compression* and *Predictive vs. Transform coding*. The reconstructed image is identical after reconstruction from lossless compression while the reconstructed image may have some degradation after reconstructing it from the lossy compression technique because lossy compression entirely removes redundant information. Lossy compression always has better compression ratio as compared to loss-less compression techniques. In predictive coding, the current information is used to predict future data and the difference between the predicted and original values are recorded. This technique is easy to implement. Transform coding, alternatively, converts the image to a different form of representation using some conversion algorithm and then records the transformed coefficients. Transform coding has better compression ratio than predictive techniques but is more expensive in terms of computation. We now discuss some specific data compression techniques.

2.4.1 Huffman Compression

The Huffman technique is commonly used in data compression due to its simplicity. All it requires is statistical information for the data which needs to be encoded. Huffman coding algorithm is named after its inventor, D.A. Huffman. The steps in Huffman technique are :

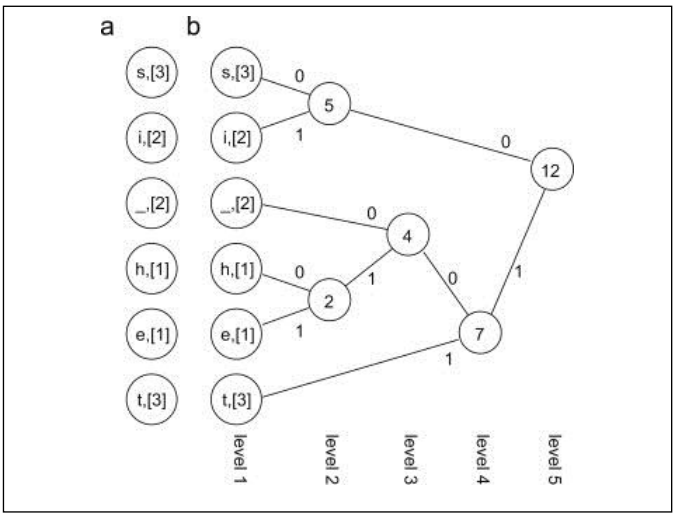
Step 1: Construct a Huffman tree by sorting the histogram and then join two bins of the smallest values till. Repeat this till just one bin remains.

Step 2: Convert the Huffman tree into coded form and save it along with the coded values.

Step 3: Encode the remaining image.

The performance and efficiency of the Huffman coding technique depends on the data distribution. If the data has a large range then the Huffman technique requires a greater computational cost. Huffman coding is one of the key concepts used in JPEG compression, which is currently the most commonly used compression method for images. Differential Pulse Code Modulation (DPCM) is used in lossless JPEG compression. The lossless JPEG uses two different coding schemes: arithmetic coding and Huffman coding. Arithmetic coding has been found to offer better compression results than that of Huffman coding but Huffman coding has less computational cost than arithmetic coding. (Hu, Y., and Chang C., 2000).

An example of a Huffman tree is :



Source: Stabno, M., Wrembel, R. (2009). "RLH: Bitmap compression technique based on run-length and Huffman encoding." Information Systems **34**(4-5): 400-414.

Huffman Coding generates the code tree in a bottom up fashion and builds the codes from right to left. (Rao, O.S., et. a., 2011). First it sorts all alphabets in the given input string in descending order of their probabilities and stores them in a list and then builds a binary tree with an alphabet at each leaf where, at each step, alphabets having the two smallest probabilities are selected. Belevic, A., et. al., (2008) stated that Huffman Coding is *"a statistical lossless data compression algorithm"*. It gives a diminution in the average code length by assigning smaller code values to more predominant alphabets. But in the real world, it is very hard to know probabilities in advance, so we have to either compromise on lower compression ratio or use adaptive Huffman coding which gives a one-pass encoding while accommodating changing statistics in the input data. Though Huffman Coding is conceptually simple, the main disadvantage of adaptive Huffman Coding is the high cost of tree maintenance.

2.4.2 Arithmetic Coding

The concept of Arithmetic Coding was first introduced by Elias in early 1960s. Arithmetic Coding converts an input string into an interval of real numbers between 0 and 1 (Witten, I. H., et. al., 1987). As the input string become larger and larger, the number interval becomes smaller but the number of bits to identify the interval increases. Successive characters or symbols of the input string decrease the interval size in accordance with their probabilities. Singla, V., et. al., (2008) describes the compression and decompression processes using Arithmetic Coding. The output from the Arithmetic Coding encoding process is a single number

between 0 and 1. In order to construct the resultant number, a set of probabilities needs to be calculated for each symbol of the string.

Following example illustrates the process of Arithmetic coding (Singla, V., et.al. 2008), which uses the string: “BILL GATES”. The first step in the process is to generate a probability and a range value for each character.

The above string contains 10 characters, so characters which occur once have 0.10 probability (A,B,E,G,I,S,T,space) and characters occurs twice have 0.20 probability (letter L). First step is to write all letters in the string in ascending order. Then consider the range will start from 0.00, so “Space” has range from 0.00 to 0.10 as it has 0.10 probability. The range of letter “A” will be 0.10 to 0.20 and so on.

Character	Probability	Range
Space	1/10	0.00 >= r > 0.10
A	1/10	0.10 >= r > 0.20
B	1/10	0.20 >= r > 0.30
E	1/10	0.30 >= r > 0.40
G	1/10	0.40 >= r > 0.50
I	1/10	0.50 >= r > 0.60
L	2/10	0.60 >= r > 0.80
S	1/10	0.80 >= r > 0.90
T	1/10	0.90 >= r > 1.00

The next step is to calculate range values, (Low and High) as :

Range = High – Low
Low = Low + Range * Low Range(c)
High = Low + Range * High Range(c)

To encode letter “B” using above formulae,
Range = 1 – 0 = 1 (Here we are considering the range between 0 and 1)
Low Value = 0 + 1 * 0.20 = 0.20
High Value = 0 + 1 * 0.30 = 0.30

Now Low and High Value to be used are 0.20 and 0.30 respectively.

To encode letter “I”,
Range = 0.30 – 0.20 = 0.10 (Considering low and high values of “B”)
Low Value = 0.20 + 0.10 * 0.50 = 0.25
High Value = 0.20 + 0.10 * 0.60 = 0.26

To encode letter “L” ,
Range = 0.26 – 0.25 = 0.01
Low Value = 0.25 + 0.01 * 0.60 = 0.256
High Value = 0.25 + 0.01 * 0.80 = 0.258

To encode second letter “L”
Range = 0.258 – 0.256 = 0.002
Low Value = 0.256 + 0.002 * 0.60 = 0.2572
High Value = 0.256 + 0.002 * 0.80 = 0.2576

Following is the table of Low values and High values of each letter after encoding them using above formulae –

Character	Low Value	High Value
	0.0	1.0
B	0.2	0.3
I	0.25	0.26
L	0.256	0.258
L	0.2557	0.2576
SPACE	0.25720	0.25724
G	0.257216	0.257220
A	0.2572164	0.2572168
T	0.25721676	0.257168
E	0.257216772	0.2572167776
S	0.2572167752	0.2572167756

So, we get 0.2572167752 as the resultant number for the original input string “BILL GATES”. In the decoding algorithm, we just reverse the above steps to achieve the original string. As we get 0.2572167752 which comes between 0.2 and 0.3, we easily get first character of the string i.e. *“B”*. To get all characters of the string, following calculations are done :

Character = Find Character (Number)
Range = High Range (character) – Low Range (Character)
Number = (Number – Low Range (Character)) / Range

Using these calculations, we get Range (0.3 – 0.2) as 0.1 and Number is ((0.2572167752 – 0.2)/0.1) = 0.572167752 which falls between the range 0.5 to 0.6 i.e. character *“I”*. After repeating this process, we get the original string “BILL GATES”.

Number	Low	High	Range	Character
0.2572167752	0.2	0.3	0.1	B
0.572167752	0.5	0.6	0.1	I
0.72167752	0.6	0.8	0.2	L
0.6083876	0.6	0.8	0.2	L
0.041938	0.0	0.1	0.1	SPACE
0.41938	0.4	0.5	0.1	G
0.1938	0.1	0.2	0.1	A
0.938	0.9	1.0	0.1	T
0.38	0.3	0.4	0.1	E
0.8	0.8	0.9	0.1	S
0.0				

Though Arithmetic Coding is useful in text data compression, Howard, P.G. and Vitter, J.S. (1994) mentioned one disadvantage of arithmetic coding is that the execution of Arithmetic Coding Algorithm is very slow as it has numerous multiplications and divisions. Suppose low and high values are too close that the scaling operation maps some different symbols of the model onto the same integer in the low (Low, High) interval. In this case, the encoding process may be possible to continue (Witten, I. H., et. al., 1987).

Witten, I. H., et. al., (1987) also discussed two overheads on the performance of the compression efficiency algorithm. The algorithm is added 2 extra bits at the end of the data which leads to the termination overhead and due to fixed-length arithmetic, the algorithm truncates the reminders while doing division arithmetic.

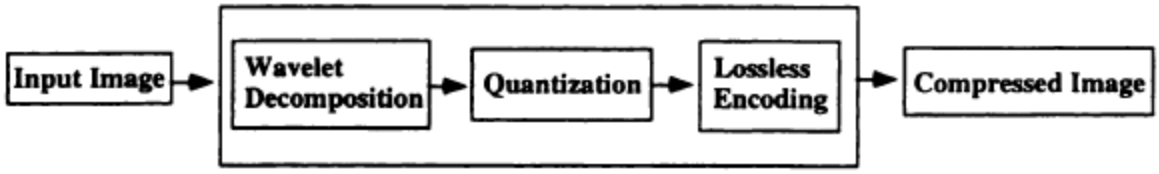
2.4.3 Wavelet Compression

The wavelet is described as a "*small wave*" which provides an analytical tool for transient, moving or time varying circumstances. It has the capability to allow concurrent time and frequency analysis. The wavelet transform is a powerful mathematical tool for data compression. Wavelets are adjustable and adaptable and can be designed for adaptive systems that adjust themselves to suit the signal. Wavelets can be made to tend to zero as fast as possible. It is this property that makes wavelets so effective in signal and audio compression. (Khalifa, O. O., et. al., 2008).

There are various types of wavelet transformation techniques. Some of the common and widely used methods are – Discrete Wavelet Transform (DWT), Continuous Wavelet Transform (CWT), Fast Wavelet Transform (FWT) and so on. Khorrami, H., Moavenian (2010) compared CWT and DWT in ECG arryhythmias classification for improving the pattern classifiers.

Wavelet transform technology is an efficient tool in video compression (Fakeh, R., et. al., 2009). Discrete Wavelet Transform (DWT) is a useful tool in visual applications, particularly in video sequencing because DWT is very flexible to use.

According to Schomer, D. F., et. al., (1998), there are three basic components of wavelet-based compression algorithms – Image transformation, Quantization, and encoding.



Source: Schomer, D. F., Elekes, A.A., Hazle, J.D., Huffman, J.C., Thompson, S.K., Chui, C.K., Murphy, W.A. (1998). "Introduction to wavelet-based compression of medical images." RadioGraphics - The Journal of continuing medical education in radiology **18**: 469-481.

The forward wavelet transform decomposes the image which results into a list of wavelet coefficients which represents the input image. The quantizer removes the redundant data and then the encoder presents the compressed image efficiently. The intention of quantization is to restrict the values of transformed coefficients to decreased numbers. The lossless encoding step implements the compression by substituting the original signal with quantized wavelet coefficients.

According to Fakeh, R., et. al., (2009), video compression algorithms are not as effective as their counterparts in other data type domains such as text or still image domains, and thus to get bigger compression ratios both amongst both spatial and temporal dimensions should be attempted in video compression. Spatial compression technique removes redundancy from video frames without compromising the video quality. The 2-D DWT technique is then used on the frames and creates decomposition levels into the LL, LH, HL and HH sub-bands. This leads to approximation coefficients which are defined over different levels. This technique is very successful in compressing video frames without compromising quality.

Garofalakis, M., and Gibbons, P.B. (2004) observed that wavelet compression techniques have achieved great success in image and signal processing applications. DWT is used in JPEG2000 compression. Coefficients thresholding is also useful in data reduction. The "*absolute normalized value*" is proposed as a threshold to reduce the overall *root-mean squared error* in data compression. In this process, all wavelet coefficients whose values are less than this threshold are replaced with the 0 value.

Kambli, M., and Bhatia, S. (2010) compares DCT (Discrete Cosine Transform) based JPEG, wavelet based SPIHT (Set Partitioning in Hierarchical Tree) using fingerprint data. The SPIHT wavelet base technique achieved a better compression ratio than the JPEG technique. It speeds the image transmission and minimizes the storage requirements.

Talukdar, K.H. and Harada, K. (2007) mentioned that the wavelet transformation separates the image information into approximation and detail coefficients. If these detail coefficients are too smaller than a given threshold value then they can be replaced with zero. The larger the number of zeros, the better is the compression ratio. Lossy compression should be implemented to obtain more compression. A positive threshold value (Tv) when set will yield a bigger compression ratio as it will cause all coefficients having value less than or equal to Tv to be replaced with the zero value. Thus, when T=0 then it results in lossless compression and when Tv>0, then we have lossy compression. The setting of the threshold value is important as it determines the trade-off among between compression ratio and quality of the reconstructed data. There are three types of thresholding – Hard, Soft and Universal thresholding.

Hard Thresholding :

$$T(Tv,x) = \begin{cases} O, & \text{if } |x|=Tv \\ x, & \text{Otherwise} \end{cases}$$

Soft Thresholding :

$$T(Tv,x) = \begin{cases} O, & \text{if } |x|=Tv \\ \text{Sign}(x)\,(|x|-Tv), & \text{Otherwise} \end{cases}$$

Universal Thresholding :

$$T(Tv,x) = \begin{cases} O, & \text{if } x < \delta(2\log_2 N)^{1/2} \\ x, & \text{Otherwise} \end{cases}$$

where, δ is standard deviation of wavelet coefficients and N is number of wavelet coefficients

We can calculate the compression ratio as :

$$\text{Compression Ratio} = \frac{\text{Number of nonzero coefficients in original matrix}}{\text{Number of nonzero coefficients in updated matrix}}$$

Following two sub-sections discusses about image and audio-video data compression. Image, Audio and Video data needs more memory to store. They are storage intensive. But all of them are good candidates for data compression as they contain a great deal of redundancy. This has made them a natural target for compression and methods such as JPEG and MPEG.

2.4.4 JPEG Compression

Image compression involves reducing the number of bits needed to store the image. To bring down the transmission bandwidth over Internet and storage space, JPEG is the preferred technique for image data compression over other image formats like BMP. JPEG stands for "*Joint Photographic Experts Group*" and established by ISO (*International Standards Organization*) and IEC (*International Electro-Technical Commission*).

Jain, A., et. al., (2007) stated that the JPEG compression is executed autonomously on sets of 8x8 pixels in an image. JPEG compression gives greater compression ratio as compared to other image compression techniques and also maintains the quality of the image. Another important property of the JPEG compression standard is that the brightness and chrominance data of an image that is divided into 8 rows of 8 pixels each. The JPEG compression standard encodes these 64 pixels at one time. Hence no 8x8 pixel sets are dependent on each other for compression.

JPEG supports DCT (*Discrete Cosine Transform*) scheme (Kambli, M., and Bhatia, S., 2010). It uses a lossy technique. Images have spatial representation and every pixel is identified by location coordinates and the colour. DCT converts this spatial representation into frequency-based array characterization. The DCT encoder uses 8x8 blocks of image in compression process. The JPEG technique uses both arithmetic and Huffman coding. Arithmetic coding gets 5-10% better compression performance than Huffman coding on the average but the complexity is greater than with arithmetic coding. Singh, S., Sharma, R.K., and Sharma, M.K. (2009) observed that the new JPEG2000 standard gives a better compression rate than normal JPEG. It is implemented in image compression, mobile transmission, PDAs and desktop computers. But when an image has graphics data like logos, the compression performance is reduced in JPEG2000 because such images are based on low color depth and have a limited number of colours.

Talukdar, K.H. and Harada, K. (2007) stated that an image contains identical information from a certain viewpoint so it is feasible to eliminate some redundant and identical information from the image using compression techniques. Discrete Cosine Transform (DCT) is an effective tool in image compression. It gives a better approximation of an image with fewer coefficients. JPEG method is commonly used to store images. JPEG is based on DCT and identifies three modes (sequential, progressive and hierarchical) for lossy compression and one for lossless encoding. There is degradation in performance at low bit-rates in JPEG due to the use of DCT.

2.4.5 MPEG Compression

MPEG is an acronym for "*Moving Picture Expert Group*" which is used in video compression and broadly used in different multimedia applications. There are different MPEG compression standards for video and the most popularly known standards are MPEG-2, MPEG4, MPEG-7 and so on. Xia, J., et. al., (2003) mentioned that due to limited bandwidth presently available for leading applications and exposure to the human visual system, investigation in the digital video compression is controlled by lossy compression, where a definite level of deformation is presented in order to accomplish the best achievable compression efficiency. It is also vital to introduce a certain distortion level while applying lossy compression technique so that the best possible compression can be achieved. The applications where the distortion is not tolerated, lossless compression plays an important role to save the storage place as well as the quality of the image for future use.

Koumaras, H., Kourtis, A., Lin, C.H., and Shieh, C.K. (2008) describes that MPEG compression uses lossy techniques as it gives a partial loss while video compression. MPEG is using spatial, frequency and temporal domain in the sequence of video frames. It compresses the video data by removing redundancy from these domains by loosing certain amount of data which is not possible to retrieve back. This problem leads the researchers to introduce certain level of distortion while using lossy compression so greater efficiency compression can be gained. Koumaras, H., et. al., (2008) proposes a framework for video quality prediction of encoded video signal and linking transmission loss ratio to video quality deterioration.

Apart from the data compression techniques covered in this review, there are a number of other techniques used, such as LZW, ZIP, and so on. While Huffman Coding and Arithmetic Coding are effective in text data compression they have also been used in image compression. On the other hand, techniques such as MPEG and JPEG techniques were specifically designed to work with image, audio and video data.

Many researchers found that Arithmetic Coding was superior to Huffman Coding. Witten, I.H., et. al., (1987) applied fixed probability model for symbols on the input string and then apply Arithmetic Coding algorithm. They found that Arithmetic Coding needed extra bits at end of the string and tell the algorithm about end of the input string. This increases the data termination overhead. Though Huffman Coding is one of the popular data compression techniques, Witten, I.H., et. al., (1987) found that the size of the compressed data is much less in Arithmetic Coding than Huffman. They also found that the processing time is reduced to almost

half when compared to Huffman coding. Hu, Y.C., et. al., (2000) found that the computation cost is much higher in Huffman coding if the input data has a large dynamic range. Singla, V., et. al., (2008) and Rao, O.S., et. al., (2011) compared Huffman Coding and Arithmetic Coding techniques in their research. Singla, V., et. al., did not used fixed probability model as Witten (1987) but the probability model was generated on the given input string. Both Singla (2008) and Rao (2011) agreed upon the efficacy of Arithmetic Coding over Huffman Coding. Rao (2011) did testing of both Arithmetic Coding and Huffman Coding and found that Arithmetic Coding gave better results and obtained compression ratios that were to 3-4 times better than that of Huffman coding. However they concluded that Arithmetic Coding is very slow to execute due to greater computational overhead.

Latu, G. (2010) compared both sparse and dense data structure in his research. A sparse data structure generally consists of data where most of the data coefficients are almost equal to zero so that we can get benefit in both space and time as only non-zero data coefficients need to be considered. On the other hand, dense data structure stores all data values. Sparse data often results in reduction in memory overheads and provides a fast approach to storing data. Wavelet representation provides such sparsity.

Latu, G. (2010) compared three different data structures in wavelet representation – dense tree, hash table, and binary tree, by way of processing time and storage space required. After thorough experimentation, he found that binary tree gave better results in both processing time and storage space. Binary tree required relatively little memory to store coefficients compared to the dense tree which required large storage space. The computational time is also much less when compared to other binary tree methods of data compression. The Haar wavelet transform defines a binary tree structure and is found to be the simplest and most efficient amongst all wavelet transforms. To compress the data, we can also use wavelet thresholding which involves two steps. First transform original data into wavelet coefficients and then discard all wavelet coefficients which are very small in value. The wavelet thresholding results in a very small error during the reconstruction process.

Chapter 3 : Haar Wavelet

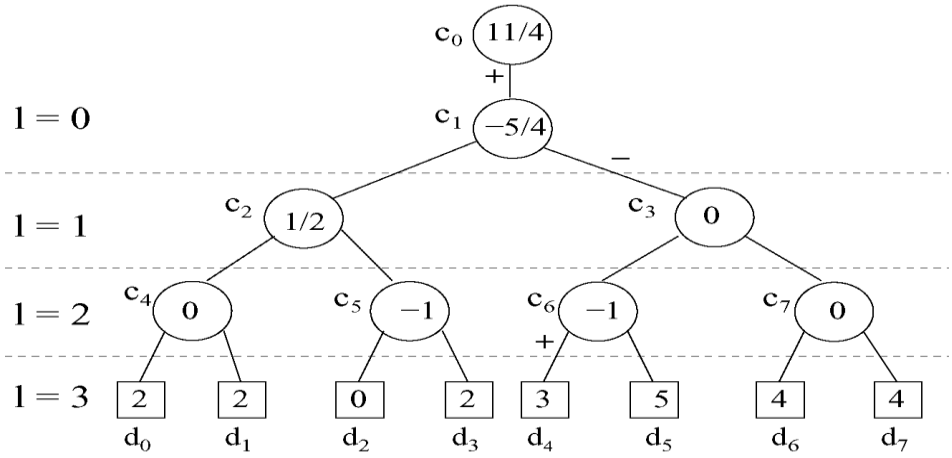
Deligiannakis, A., Garofalakis, M., Roussopoluos, N. (2007) describes *Haar wavelet* as a mathematical tool for the hierarchical decomposition of functions. It is successfully implemented in several applications like signal and image processing. It is also an efficient tool for data reduction in large databases and query processing on voluminous tables. The main idea of Haar wavelet is to deploy the decomposition process on large datasets with threshold values to get compact datasets comprising Haar wavelet coefficients.

3.1 One-Dimensional Haar Wavelet

We illustrate the decomposition process with the help of an illustrative example. Suppose that we have a one-dimensional array A = [2, 2, 0, 2, 3, 5, 4, 4] with N = 8 values, then the wavelet transform of this array will be calculated as follows.

First, pairwise averages are taken to get "*lower resolution*" of the data which yields [2,1,4,4]. Naturally, while calculating these averages, some information is lost in the process. So, to restore the original values, we need some "*detailed coefficients*". These detailed coefficients are the differences between average value and the first element of the pair values. So, it comes to [0, -1, -1, 0]. This process is repeated till we get overall average. Following table describes how we get averages and detail coefficients at each stage :

Resolution	Averages	Detailed Coefficients
3	[2,2,0,2,3,5,4,4]	--
2	[2,1,4,4]	[0,-1,-1,0]
1	[3/2,4]	[½,0]
0	[11/4]	[-5/4]



So after applying wavelet decomposition on this example, we get one-dimensional haar wavelet transform coefficients is WA = [11/4, -5/4, ½, 0, 0, -1, -1. 0].

3.2 Multi-Dimensional Haar Wavelet

Stollnitz, E. J., et. al., (1995) describes two-dimensional Haar wavelet transform. Standard decomposition and non-standard decomposition methods are used in multi-dimensional Haar wavelet decomposition process.

To get standard decomposition, one-dimensional wavelet transform has been applied to each row of multi-dimensional array. This gives average and detail coefficients for each row in the array. Then, these transformed rows are treated as data values, and then again one-dimensional wavelet transform has been applied to each column. The resulting array will have all detail coefficients with one overall average coefficient. The algorithm for standard decomposition method is :

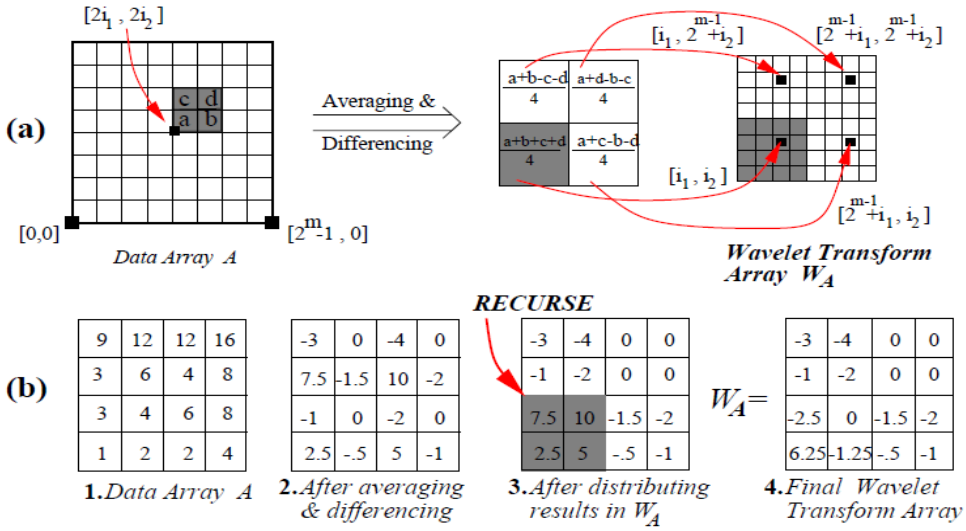
```
Proc StandardDecomposition(Array[1,j, 1.w) of reals)
for row = 1 to h
{
        Decomposition(Array[row, 1..w]
}
for col = 1 to w
{
        Decomposition(Array[1..h, col]
}
```

Non-standard decomposition alternates between operations on rows and columns. In the first step, horizontal pair-wise average and differences are calculated on each row values. Then same process is done vertically on column i.e. averaging and differencing pair-wise elements vertically on each column. The process is repeated only on the quadrant having averages. The algorithm for this non-standard decomposition is :

proc *NonstandardDecomposition* (C: array[l..h, l..h] of reals)

```
C = C/h                    --normalize input coefficients
while h > 1 do
        for row = 1 to h do
                DecompositionStep(C[row, 1 ..h])
        end for
        for col = l to h do
                DecompositionStep (C[1. .h, col])
        end for
        h = h/2
end while
```

Chakrabarti, K., et. al., (2001) employed a non-standard decomposition process in their paper.



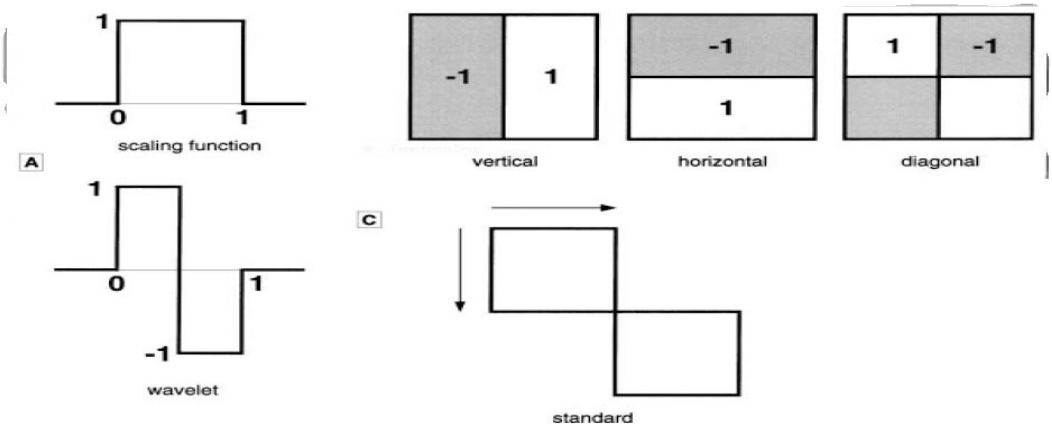
Source : Chakrabarti, K., Garofalakis, M., Rastogi, R., Shim, K. (2001). "Approximate query processing using wavelets." The VLDB Journal **10**(2-3): 199-233.

Chahrabarti et. al. (2001) showed that multi-dimensional wavelet decomposition can play an important role in improving query performance in modern applications having a large volume of data. They transformed raw source data into the wavelet domain to get approximate, yet accurate results of queries. Since their query processing algorithms operated in sparse wavelet space they were able to achieve very significant speedups in query processing. They also proposed an I/O efficient wavelet decomposition process for generating relational data.

Raviraj, P., and Sanavullah, M.Y. (2007) investigated the success of 2-D Haar wavelet transformation in image compression. They focused on compressing the image using 2-D Haar wavelet transformation to reduce the computational needs by using different threshold values on calculated wavelet coefficients to obtain the best trade-off between compression ratio and image reconstruction quality. They concluded that the Haar wavelet transformation was very effective on account of its high computational speed and high compression ration compared to other

methods of data compression.

Papageorgiou, C., Poggiom, T. (2000) implemented 2-D Haar wavelet decomposition in face detection, people detection and car detection tasks. They transferred their images from pixels to wavelet coefficients decomposed by applying non-standard 2-D DWT (Discrete Wavelet Transform) method.



Source : Papageorgiou, C., Poggiom, T. (2000). "A Trainable System for Object Detection." International Journal of Computer Vision **38**(1): 15-33.

Three types of wavelet functions in non-standard decomposition are used in this process – vertical, horizontal and diagonal coefficients. The vertical coefficient is difference in average intensity in vertical order, horizontal coefficient is difference in average intensity in horizontal order while diagonal coefficient is calculated by taking into consideration the diagonal order. (See section B of above figure).

The 2-D data is transformation a 2-D simplification of 1-D wavelet transformation (Talukdar, K.H. and Harada, K., 2007). So, in 2-D image compression, 1-D wavelet transformation is applied on each row which gives an average value of row elements and detail coefficients. Then this converted matrix is treated as an original and 1-D transformation is applied on each column. The all resultant values are detail coefficients except one which is overall average of all coefficients. Now, this method is repeated continuously on the section having all average values. Suppose that we have following 8x8 data matrix:

$$\begin{pmatrix} 64 & 2 & 3 & 61 & 60 & 6 & 7 & 57 \\ 9 & 55 & 54 & 12 & 13 & 51 & 60 & 16 \\ 17 & 47 & 46 & 20 & 21 & 43 & 42 & 24 \\ 40 & 26 & 27 & 37 & 36 & 30 & 31 & 33 \\ 32 & 34 & 35 & 29 & 28 & 38 & 39 & 25 \\ 41 & 23 & 22 & 44 & 45 & 19 & 18 & 48 \\ 49 & 15 & 14 & 52 & 53 & 11 & 10 & 56 \\ 8 & 58 & 59 & 5 & 4 & 62 & 63 & 1 \end{pmatrix}$$

Now, 1-D transformation can be applied on each row by averaging and differencing on pairs of elements. For example, considering first row then Averaging is (64+2)/2=33,

(3+61)/2=32, (60+6)/2=33, (7+57)/2=32 and Differencing is 64-33=31, 3-32=-29, 60-33-27 and 7-32=-25. So, we will get transformed row as (33,32,33,32,31,-29,27,-25). Now, next step is to do same thing again only on averages i.e. (33,32,33,32) which comes to (32.5,0,0.5,0.5,31,-29,27,-25). After doing this transformation on all rows, the following matrix will be generated:

$$\begin{pmatrix} 32.5 & 0 & 0.5 & 0.5 & 31 & -29 & 27 & -25 \\ 32.5 & 0 & -0.5 & -0.5 & -23 & 21 & -19 & 17 \\ 32.5 & 0 & -0.5 & -0.5 & -15 & 13 & -11 & 9 \\ 32.5 & 0 & 0.5 & 0.5 & 7 & -5 & 3 & -1 \\ 32.5 & 0 & 0.5 & 0.5 & -1 & 3 & -5 & 7 \\ 32.5 & 0 & -0.5 & -0.5 & 9 & -11 & 13 & -15 \\ 32.5 & 0 & -0.5 & -0.5 & 17 & -19 & 21 & -23 \\ 32.5 & 0 & 0.5 & 0.5 & -25 & 27 & -29 & 31 \end{pmatrix}$$

Now, after applying 1-D transformation on above matrix, we get final transformed matrix as:

$$\begin{pmatrix} 32.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & -4 & 4 & -4 \\ 0 & 0 & 0 & 0 & 4 & -4 & 4 & -4 \\ 0 & 0 & 0.5 & 0.5 & 27 & -25 & 23 & -21 \\ 0 & 0 & -0.5 & -0.5 & -11 & 7 & -7 & 5 \\ 0 & 0 & 0.5 & -0.5 & -5 & 7 & -9 & 11 \\ 0 & 0 & -0.5 & -0.5 & 21 & -23 & 25 & -27 \end{pmatrix}$$

Here, we can see that in the final transformed matrix, number of zeros is much more than in the original matrix. It is very easy to reconstruct the Original matrix from this transformed matrix by applying reverse operation of averaging and differencing. This is lossless non-standard technique of wavelet decomposition on 2-D data array.

3.3 Why Haar Wavelet ?

The rationale behind wavelet compression lies in selectively decomposing and reconstructing just those spatial data that are mainly significant to our eyes. Due to this reason, wavelet compression has been the preeminent compression method for two-dimensional images or textual data.

Stollinz, E.J., et. al., (1995) mentioned that an image in the form of wavelet coefficients rather than an actual image has lots of advantages. One of the benefits of storing wavelet coefficients is that the large number of the detailed coefficients frequently is very small in size. So, we can either truncate or remove these small wavelet coefficients from the representation, thus resulting in very small errors in reconstructing the original image. This gives us a lossy compression with minimal distortion in the image quality.

There are various types of wavelets. However, researchers prefer Haar wavelet due to the fact that they are conceptually simple, have high computational speed and are a most efficient method of compression.

Chapter 4: Research Methodology

In this chapter, research methodologies which are suitable for this study are discussed. Then, a number of research scenarios and terms used in this study such as distance and angle of RFID tags will also be discussed. Each scenario presents a different scheme for storing wavelet coefficients. The calculations for estimating distance and angles between tags are also presented in this Chapter.

4.1 Introduction

Every research has two main paradigms and a suite of research methods underlying these paradigms. They are *Positivist* and *Interpretivist* paradigms (de Villiers, M. R., 2005). The *positivist* paradigms states that the knowledge is implicit and neutral and it is observed by verifiable ways, e.g. experiments. Positivist research is supposed to create an exact picture of real scenarios. On the other hand, *Interpretivist* paradigms intend to discover new interpretations or inherent meanings and sticks firmly to the ontological premise of numerous realities, depending on time and context.

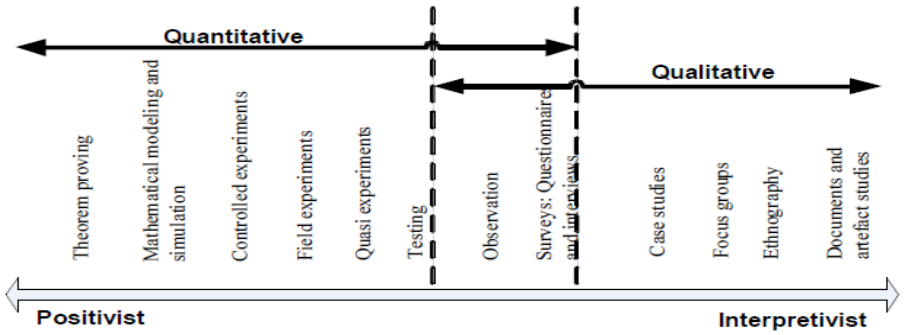


Figure 4.1: Basic Research Paradigms

Source: deVilliers, M. R. (2005). Three approaches as pillars for interpretive information systems research: development research, action research and grounded theory. Proceedings of the 2005 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries, South Africa.

Positivist research consists of quantitative methods such as experiments, testing, etc in which numbers and measurements data is used using statistical methods. While positivist paradigm examines hypotheses, *Interpretivist* paradigm analyses research questions and uses qualitative data. Qualitative methods such as case studies and grounded theory are used with the *Interpretivist* paradigm as shown in Figure 4.1.

4.2 Selection of Methodology

Constructive Research Method will be used in this project. This research work is based on the constructive research method, which will be to build and test the algorithms designed for compression and navigation. For algorithm testing, as experimental study will be used to to analyze and compare the effectiveness of the proposed algorithms.

The compression algorithms will encode information about the location of each node as well as a certain number of its nearest neighbors. Each node will contain information to efficiently assess the most promising paths that originate from that node. Once the compression algorithm is in place, an experimental approach will be used to evaluate navigation efficiency. A number of experiments will be run and the sensitivity of key parameters on performance such as the number of nearest neighbors and the wavelet thresholding level will be investigated. The performance metric to be used is the number of hops necessary to traverse the network from a given source node to a pre-specified target node.

4.2.1 Constructive Research

The constructive research method is the most commonly used methodology in computer science research. The method is intended to solve problems that occur in real world systems. It is used to produce innovative constructions which make a contribution to the theory of the discipline in which the method is applied. (Lukka, 2003). According to Lukka, "*The constructive research approach is a research procedure for producing innovative constructions, intended to solve problems faced in the real world and, by that means, to make a contribution to the theory of the discipline in which it is applied.*" Caplinskas, A., Vasilecas, O. (2004) describes "Constructive Research" as an important research method in the Information Systems field. With this method an artifact is first constructed and subsequently the constructed artifact is evaluated to determine its efficacy. In most cases the evaluation is performed through the application of an experimental approach.

4.2.2 Experiment

According to Caplinskas, A., Vasilecas, O. (2004), Experimental Research evaluates tools algorithms and techniques that are oriented to Information Systems research. Experimental Research includes experimental simulation, adaptive experiments, laboratory experiments, and field experiments. Walker, W. (2005) observed that Experimental Research provides a model for establishing relationship between source and outcomes. The researcher exercises logical thinking to prove hypotheses which comprises of manipulating independent variables (causes or source)

and discovering the result on dependent variables (effects). The strength of an experimental research is the ability control key variables to investigate cause and affect relationships. Experiment methodology includes the use of standardised procedures to decrease systematic bias and eliminate or greatly reduce the chance of arriving at incorrect conclusions.

Experimental Research involves testing hypothetical anticipations against reality. Experiments are used when theory and reasoning analysis do not fully explain the phenomenon under consideration. Experiments investigate the power of assumptions, get rid of alternate explanations of phenomena, and find new phenomena in demand of explanation. In this manner, experiments help to initiate deriving theories from measures. Repeatability is the vital requirement of an experimental approach. It guarantees that outcomes can be tested severally and increases assurance in the outcomes which helps to remove mistakes, frauds and hoaxes (Tichy, W. F., 1998). The main benefits of experimental study are:

- Experiments help to develop an authentic base of cognition and increase certainty about adequacy of processes, methods, theories, etc.
- Experiments can speed up development by speedily removing unproductive approaches, wrong assumptions.
- Experimentation can guide to new, useful and unpredicted perceptions and open new areas of research.

Given the advantages of experimentation and the natural fit with the style of research in this study we thus employ the Experimental approach in the evaluation phase of our research.

4.3 Scenarios for experimentation

Four different scenarios are considered for wavelet compression and then the resulting compressed data is evaluated with the use of a navigation algorithm that operates under different experimental configurations. Each scenario has different algorithm for data compression. In scenario 1, it is assumed that each RFID tag is able to store all wavelet coefficients. In Scenario 2, standard wavelet decomposition method is applied on a multi-dimensional data array. The standard decomposition method is applied on an array of size (N-1, 2) which contains values for X and Y coordinates, and Tag Labels. In scenario 3, each tag contains location information about every other tag at a level of resolution that depends on its distance from the source tag. The tag stores exact information about the neighbor tags and progressively fuzzier information about tags which are far away. In Scenario 4, non-standard wavelet decomposition method was applied on the tag data.

In addition to different algorithms, thresholding is also used in all scenarios. Three threshold values are used in the experiments : 0, 0.50 and 25%. The block diagram of the experimentation plan is drawn in following figure (see Figure 4.2) :

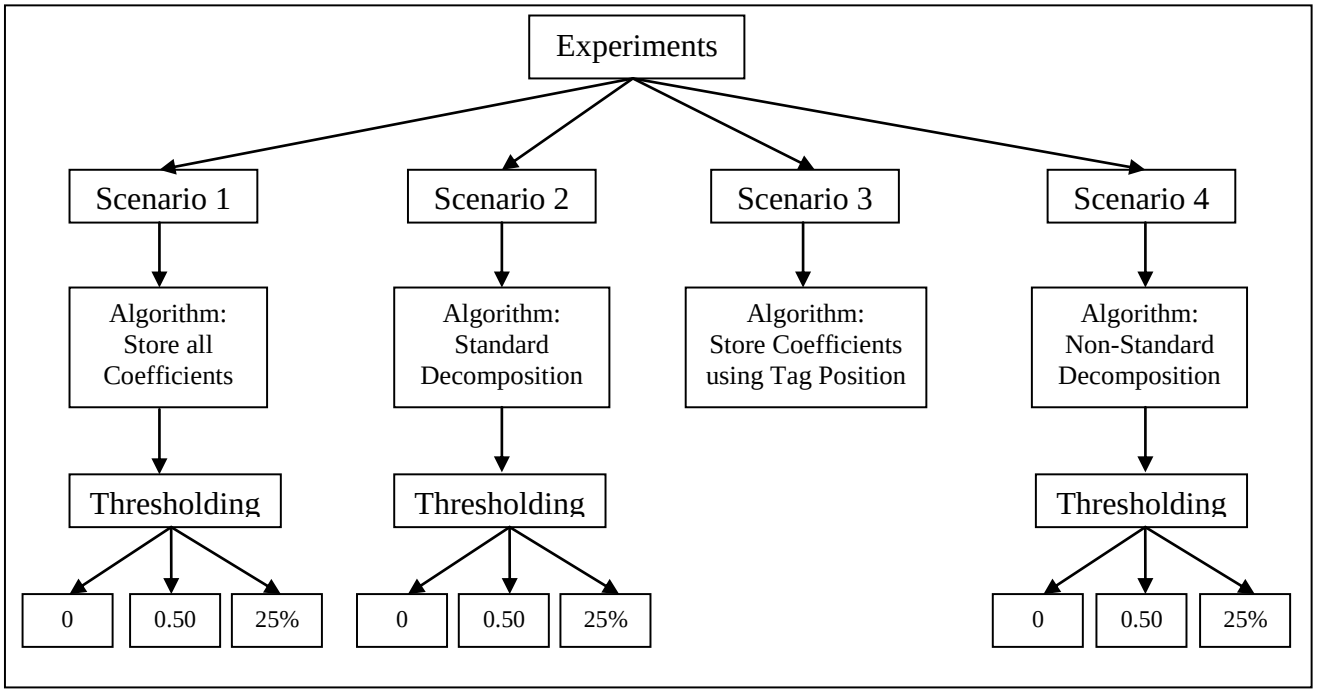


Figure 4.2: Experimentation Plan

4.4 Important factors in the Navigation Algorithm

To navigate from the source tag to the destination tag in the RFID environment, the user must know the distance between the tags, especially the distance to the next reachable tag in the navigation path. The angle between tags also plays a vital role as it provides the directional information which is necessary to identify the best candidate to move to from the current tag position.

4.4.1 Distance

In an indoor and underground navigation environment, RFID tags are installed with their specific positional information. Each and every tag in this environment has three data sets, i.e. tag label, X-Axis and Y-Axis to indicate the exact location of the tag. The X and Y coordinates are encoded in the form of wavelet coefficients that have been subjected to thresholding and are thus stored in approximate form. From each source tag, the algorithm can calculate the distance between itself and other remaining tags in order to move to the most promising neighboring tag in its search for a path to the destination tag.

In operation, the end-user will specify source and destination tags, and the system will then reconstruct X-Axis and Y-Axis data from wavelet coefficients stored on the source tag and

calculate the distance between the current tag and destination tag. The expression used to calculate distance is the Euclidean metric as illustrated below:

```
Distance = Math.Sqrt((DestX - CurrentX) ^ 2 + (DestY - CurrentY) ^ 2)
```

This distance metric is then used to identify the nearest neighbours to the current tag. For experimental purposes, the system calculates four nearest neighbours on each hop to the destination tag.

4.4.2 Angle

The angle is also an important aspect to consider in the navigation system. The angle between source tag and destination tag guides the navigation algorithm by providing the directional information that will result in which neighbor to move to from the pool of 4 that are available. A number of different mechanisms were tested for calculating the angle between any two given tags. After rigorous testing of different alternatives, the following formula proved to be best and is thus utilized in the experimentation:

```
DestAngle = Math.Atan2((SourceY - DestY), (SourceX - DestX))
DestAngle = Math.Round(DestAngle * (180 / Math.PI), 2)
 If (SourceY - DestY) > 0 Then
    If SourceX > DestX Then DestAngle = DestAngle + 270
    If SourceX = DestX Then DestAngle = DestAngle + 180
    If SourceX < DestX Then DestAngle = DestAngle + 90
 End If
```

4.5 Experiment Prototype

As this research study does not involve implementing algorithms in a real world environment at this time, it then becomes important to demonstrate efficiency of these algorithms in a laboratory environment. To study the efficiency of the algorithms, application software was developed in Microsoft Visual Basic .Net. The tag data is stored in the text files. The software coding for the research is attached in this report in the Annexures. Both Compression and Navigation algorithms are presented in Annexure 1 to Annexure 10.

4.6 Experiment Data

The experimental data is created using “Tag Creation Algorithm”. The algorithm details are discussed is the next section. The same data is used in all four scenarios. Table 1 presents the tag coordinates with N=16.

Tags	X Axis	Y Axis
14	1	3
5	3	5
9	4	4
2	4	7
12	7	8
7	9	9
3	7	11
16	10	11
10	13	10
11	14	12
4	17	10
6	19	12
15	20	16
8	21	14
1	22	16
13	24	20

Table 1: Tag coordinates used in all experiments

The graphical representation of these tags is displayed in Figure 4.3 :

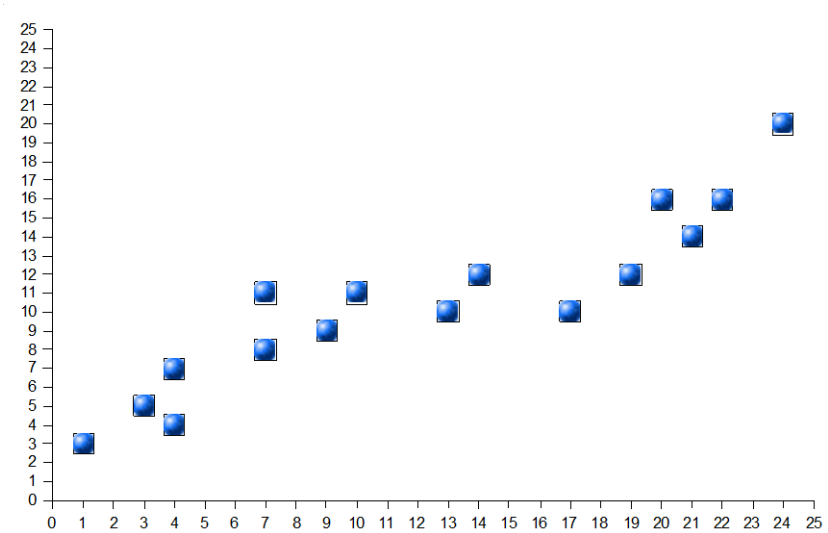


Figure 4.3: Tag distribution in 2 dimensional space.

This data is used in all scenarios to test the effectiveness and efficiency of each compression method. Two types of thresholding is used, an absolute and relative. With absolute thresholding, a number (for example 0.50) is specified and all wavelet coefficients whose numeric values are less than or equal to the threshold are replaced with 0. If relative thresholding is used, then the coefficients in the bottom region will be set to zero. For example, if the threshold value is 25% and the array contains 8 wavelet coefficients, then 25% wavelet coefficients the bottom quartile will be set to 0 and eliminated from storage. Thus only 6 wavelet coefficients will be stored in the storage array out of 8 coefficients when there are 8 tags and 12 coefficients will be stored when there are 16 tags in total.

4.7 Algorithms

For this research study, three different algorithms were designed, namely the tag creation algorithm, data compression algorithm and the navigation algorithm. In the data creation algorithm, tag data is created and stored in the text file. The data compression algorithm converts this tag data into wavelet coefficients using different scenarios and applies thresholding and then stores the resulting set of wavelet coefficients in text files as well. The navigation algorithm reads these wavelet coefficients files, reconstructs the tag data and finds the navigation path between the user specified source and destination tags. The block diagrams of the algorithms are given in Figure 4.4 and 4.5.

4.7.1. Tag Creation Algorithm

A collection of 64 tags and their X-Axis and Y-Axis coordinates have been created and stored in a text file. Tags are randomly generated with X-Axis and Y-Axis values. Then are they checked for duplicate values in X-Axis and Y-Axis. If duplicate values are found, then values for X-Axis and Y-Axis for that tag are calculated again. Then, all tags are sorted by ascending order of distance and then by angle (angle to the dummy point (0, 0)) so that tags nearer to each other are grouped together (Figure 4.4). The total number of tags used in each experiment was either 8 or 16.

4.7.2. Compression Algorithm

Using Haar Wavelet Data Compression technique, this algorithm converts tag coordinates into wavelet coefficients. The algorithm asks for number of tags in the network. Then it reads tag coordinates data from the text file which is created using *tag creation algorithm*. After calculating wavelet coefficients then depending on which scenario it is, stores the wavelet coefficients in the text file. Before storing wavelet coefficients, threshold value is also applied on these wavelet coefficients. Only one file of wavelet coefficients is created in all

scenarios except scenario 3. In scenario 3, N files are created where N is number of tags in the network, as different wavelet coefficients are stored for each tag depending on the tag position. So, if there are 16 tags then 16 text files have been created in scenario 3 (Figure 4.4).

4.7.3. Navigation Algorithm

The source and destination tag need to be entered into the navigation algorithm. Then the algorithm reads appropriate wavelet coefficients text file and reconstruct tag values. After getting reconstructed X Coordinates and Y Coordinates, the angle (DestAngle) and distance (DestDistance) are calculated from source to destination tag. In addition to this, distances and angles are calculated from source tag to other tags and stored in different arrays in ascending order. Now, the angle differences are calculated for first 4 NN tags which are not already visited before (i.e. DestAngle – TagAngle) and stored in the array. If no such tags are found then there is no available tag to move forward. Then the angle difference array are sorted in the ascending order of the difference. The first member of an array is the next tag to move forward. So, the value of the original source tag is replaced with the newly found tag. This process is recursively executed till the destination tag is found (Figure 4.5).

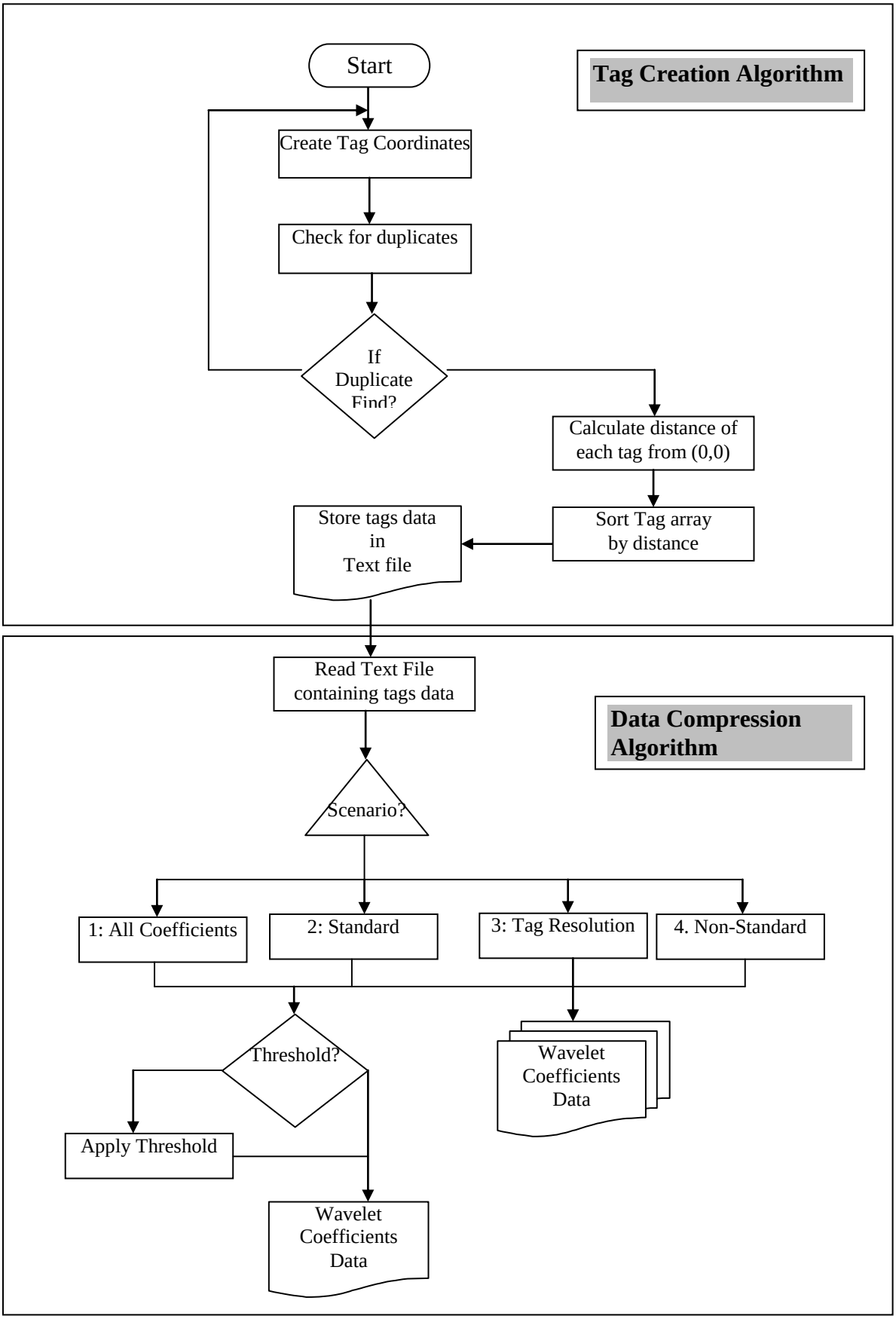


Figure 4.4: Overview of the tag and encoding process

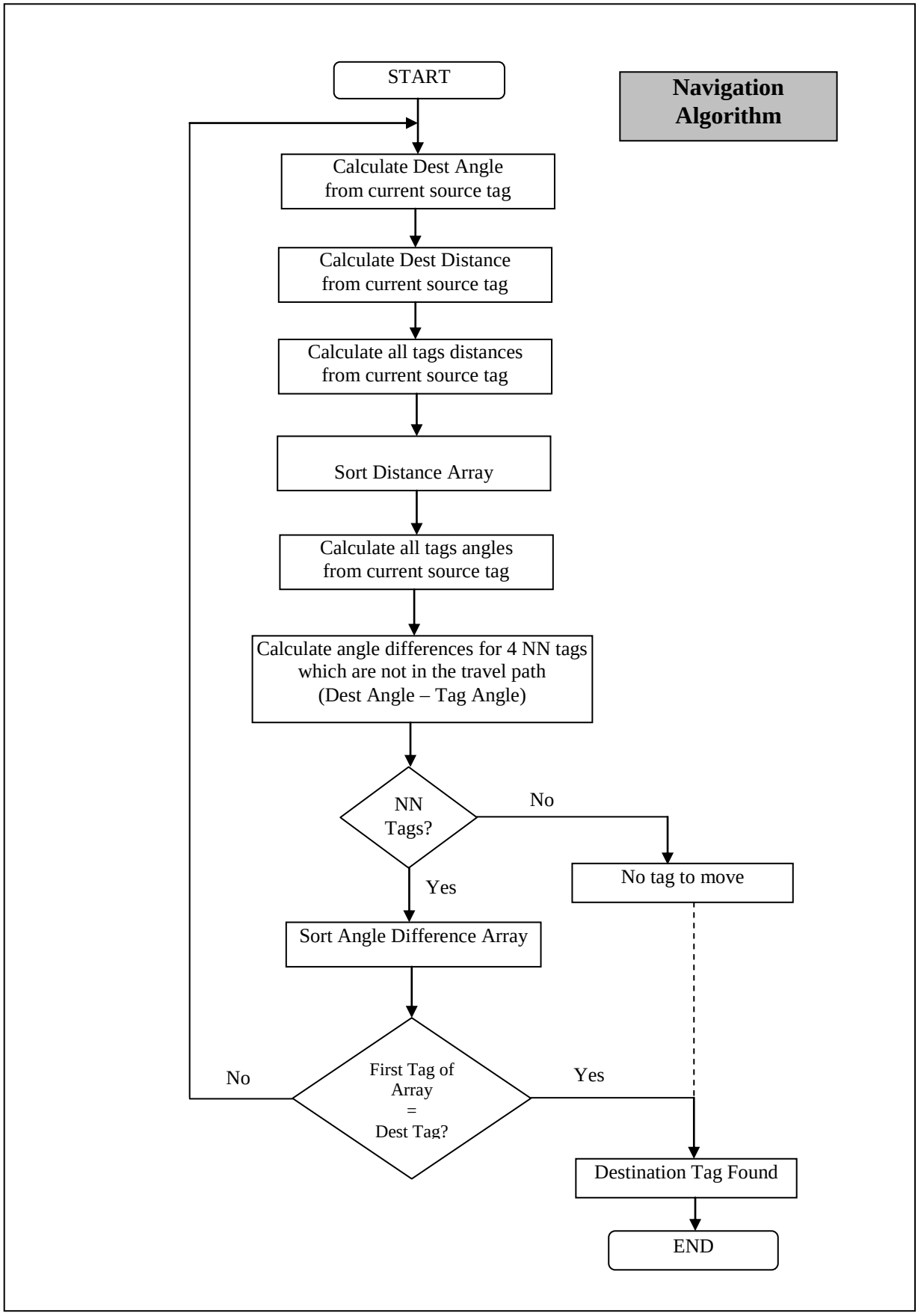


Figure 4.5: Overview of the navigation process

Chapter 5: Experimental Study and Results

5.1 Introduction

The experimental study is divided into two parts : Compression and navigation. Different algorithms (compression schemes) are used across the four scenarios. These algorithms are used to reduce storage space of 2-dimensional data for RFID tags, comprising the X-Axis and Y-Axis co-ordinates. All four scenarios are analysed with respect to the compression ratio obtained with different levels of thresholding. The navigation algorithm reads the text file having wavelet coefficients and reconstructs the 2-dimensional data for all tags in the environment. After reconstructing values for X-Axis and Y-Axis, the distance and angle will be calculated between source tag and other tags in the path in an attempt to find the next reachable tag on the navigation path.

5.2 Data Compression

The main research question investigated in this study is: ***How can Haar Wavelet techniques be used effectively to compress location information on RFID tags?*** To investigate the research question, four scenarios, as described in the previous Chapter, are tested and analyzed.

5.2.1 Scenario 1 : Tags contains all wavelet coefficients

In this scenario, it is assumed that each RFID tag is capable of storing all wavelet coefficients. Here, the original X-Axis and Y-Axis data values are stored in two separate one-dimensional arrays and after the application of wavelet decomposition on each array the resulting wavelet coefficients are stored in three different arrays, one each for Tag id, X-Axis and Y-Axis sets of coefficients.. The threshold values used here are 0.50, 1 and 25%.

Here, 8 tags and 16 tags are tested differently to check the compression ratio. After applying one-dimensional wavelet coefficient decomposition, the wavelet coefficients generated for 8 tags are displayed in Table 2:

Tags	X-Axis	Y-Axis
8	5.62,-2.62,-1,-0.25,-1,0,-1,-1.5	7.25,-2.5,-0.75,-1.25,-1,-1.5,-0.5,0
16	12.19,-6.56,-2.62,-3,-1,-0.25,-2.25,-1.25,-1,0,-1,-1.5,-0.5,-1,-0.5,-1	10.5,-3.25,-2.5,-2.75,-0.75,-1.25,0,-1.5,-1,-1.5,-0.5,0,-1,-1,1,-2

Table 2: List of wavelet coefficients for 8 tags and 16 tags in Scenario 1

After applying thresholding at 0.50, the resulting wavelet coefficients are given in Table 3 :

Tags	X-Axis	Y-Axis
8	5.62,-2.62,-1,0,-1,0,-1,-1.5	7.25,-2.5,-0.75,-1.25,-1,-1.5,0,0
16	12.19,-6.56,-2.62,-3,-1,0,-2.25,-1.25,-1,0,-1,-1.5,0,-1,0,-1	10.5,-3.25,-2.5,-2.75,-0.75,-1.25,0,-1.5,-1,-1.5,0,0,-1,-1,1,-2

Table 3: List of wavelet coefficients with threshold = 0.50 for 8 tags and 16 tags in Scenario 1

Compression Ratios for X-Axis and Y-Axis Co-ordinates for 16 tags after applying threshold = 0.50 are shown in Table 4 :

Axis	**Wavelet Coefficients before threshold**	**Wavelet Coefficients after threshold**	**Compression Ratio**
X	12.19,-6.56,-2.62,-3,-1,-0.25,-2.25,-1.25,-1,0,-1,-1.5,-0.5,-1,-0.5,-1	12.19,-6.56,-2.62,-3,-1,0,-2.25,-1.25,-1,0,-1,-1.5,0,-1,0,-1	1.25
Y	10.5,-3.25,-2.5,-2.75,-0.75,-1.25,0,-1.5,-1,-1.5,-0.5,0,-1,-1,1,-2	10.5,-3.25,-2.5,-2.75,-0.75,-1.25,0,-1.5,-1,-1.5,0,0,-1,-1,1,-2	1.08

Table 4: Compression Ratio for threshold = 0.50 in Scenario 1

After application of threshold = 1, wavelet coefficients are shown in Table 5 :

Tags	X-Axis	Y-Axis
8	5.62,-2.62,0,0,0,0,0,-1.5	7.25,-2.5,0,-1.25,0,-1.5,0,0
16	12.19,-6.56,-2.62,-3.00,0,0,-2.25,-1.25, 0,0,0,-1.5,0,0,0,0	10.5,-3.25,-2.5,-2.75,0,-1.25,0,-1.5,0,-1.5,0,0, 0,0,0,-2

Table 5: List of wavelet coefficients with threshold = 1 for 8 tags and 16 tags in Scenario 1

Compression Ratios for X-Axis and Y-Axis Co-ordinates for 16 tags after applying threshold = 1 are given in Table 6 :

Axis	**Wavelet Coefficients before threshold**	**Wavelet Coefficients after threshold**	**Compression Ratio**
X	12.19,-6.56,-2.62,-3,-1,-0.25,-2.25,-1.25,-1,0,-1,-1.5,-0.5,-1,-0.5,-1	12.19,-6.56,-2.62,-3,0,0,-2.25,-1.25, 0,0,0,-1.5,0,0,0,0	2.14
Y	10.5,-3.25,-2.5,-2.75,-0.75,-1.25,0,-1.5,-1,-1.5,-0.5,0,-1,-1,1,-2	10.5,-3.25,-2.5,-2.75,0,-1.25,0,-1.5,0,-1.5,0,0,-0,0,0,-2	1.75

Table 6: Compression Ratio for threshold = 1 in Scenario 1

After the application of threshold at the 25% level, the resulting wavelet coefficients are as shown in Table 7 :

Tags	X-Axis	Y-Axis
8	5.62,-2.62,-1,-0.25,-1,0,**0,0**	7.25,-2.5,-0.75,-1.25,-1,-1.5,**0,0**
16	12.19,-6.56,-2.62,-3,-1,-0.25,-2.25,-1.25,-1,0,-1,-1.5,**0,0,0,0**	10.5,-3.25,-2.5,-2.75,-0.75,-1.25,0,-1.5,-1,-1.5,-0.5,0,**0,0,0,0**

Table 7: List of wavelet coefficients with threshold = 25% for 8 tags and 16 tags in Scenario 1

The compression ratios obtained with 16 tags after applying threshold at 25% are given in Table 8 :

Axis	**Wavelet Coefficients before threshold**	**Wavelet Coefficients after threshold**	**Compression Ratio**
X	12.19,-6.56,-2.62,-3,-1,-0.25,-2.25,-1.25,-1,0,-1,-1.5,-0.5,-1,-0.5,-1	12.19,-6.56,-2.62,-3,-1,-0.25,-2.25,-1.25,-1,0,-1,-1.5,0,0,0,0	1.36
Y	10.5,-3.25,-2.5,-2.75,-0.75,-1.25,0,-1.5,-1,-1.5,-0.5,0,-1,-1,1,-2	10.5,-3.25,-2.5,-2.75,-0.75,-1.25,0,-1.5,-1,-1.5,-0.5,0,0,0,0,0	1.40

Table 8: Compression Ratio for threshold = 25% in Scenario 1

The error obtained after reconstruction is calculated from:

$$\%Err = \frac{|(\text{Original Value} - \text{Reconstructed Value})|}{\text{Original Value}} \times 100$$

With no thresholding applied the reconstructed values obtained were exactly the same as the original values, as shown in Tables 9 and 10 :

	Tags	14	5	9	2	12	7	3	16
Original Values	X Axis	1	3	4	4	7	9	7	10
	Y Axis	3	5	4	7	8	9	11	11
Reconstructed Values	X Axis	1	3	4	4	7	9	7	10
	Y Axis	3	5	4	7	8	9	11	11
%Err	X Axis	0%	0%	0%	0%	0%	0%	0%	0%
	Y Axis	0%	0%	0%	0%	0%	0%	0%	0%

Table 9: Reconstructed values for X-Axis and Y-Axis for 8 tags when all wavelet coefficients are stored and no threshold has been applied

Original Values	X	1	3	4	4	7	9	7	10	13	14	17	19	20	21	22	24
	Y	3	5	4	7	8	9	11	11	10	12	10	12	16	14	16	20
Construct-ed Values	X	1	3	4	4	7	9	7	10	13	14	17	19	20	21	22	24
	Y	3	5	4	7	8	9	11	11	10	12	10	12	16	14	16	20
%Err Value	X	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	Y	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

Table 10: Reconstructed values for X-Axis and Y-Axis for 16 tags when all wavelet coefficients are stored and no threshold has been applied

At the 0.50 thresholding level the coefficients obtained are displayed in Tables 11 and 12:

	Tags	14	5	9	2	12	7	3	16
Original Values	X Axis	1	3	4	4	7	9	7	10
	Y Axis	3	5	4	7	8	9	11	11
Reconstructed Values	X Axis	1	3	4	4	7	9	7	10
	Y Axis	3	5	4	7	8	8	11	11
%Err	X Axis	0%	0%	0%	0%	0%	0%	0%	0%
	Y Axis	0%	0%	0%	0%	0%	11%	0%	0%

Table 11: Reconstructed values for X-Axis and Y-Axis for 8 tags when all wavelet coefficients are stored and threshold = 0.50 has been applied

Original Values	X	1	3	4	4	7	9	7	10	13	14	17	19	20	21	22	24
	Y	3	5	4	7	8	9	11	11	10	12	10	12	16	14	16	20
Reconstructed Values	X	1	3	4	4	7	9	7	10	14	14	17	19	20	20	22	24
	Y	3	5	4	7	8	8	11	11	10	12	10	12	16	14	16	20
%Err Values	X	0%	0%	0%	0%	0%	0%	0%	0%	8%	0%	0%	0%	0%	5%	0%	0%
	Y	0%	0%	0%	0%	0%	11%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

Table 12: Reconstructed values for X-Axis and Y-Axis for 16 tags when all wavelet coefficients are stored and threshold = 0.50 has been applied

After reconstructing the data from the wavelet coefficients array with threshold set to 1, the resulting coefficients are shown in Tables 13 and 14 :

	Tags	14	5	9	2	12	7	3	16
Original Values	X Axis	1	3	4	4	7	9	7	10
	Y Axis	3	5	4	7	8	9	11	11
Reconstructed Values	X Axis	3	3	3	3	8	8	7	10
	Y Axis	5	5	3	6	8	8	11	11
%Err	X Axis	200%	0%	25%	25%	14%	11%	0%	0%
	Y Axis	67%	0%	25%	14%	0%	11%	0%	0%

Table 13: Reconstructed values for X-Axis and Y-Axis for 8 tags when all wavelet coefficients are stored and threshold = 1 has been applied

	Tags	14	5	9	2	12	7	3	16	10	11	4	6	15	8	1	13
Ori. Value	X	1	3	4	4	7	9	7	10	13	14	17	19	20	21	22	24
	Y	3	5	4	7	8	9	11	11	10	12	10	12	16	14	16	20
Constructed Value	X	3	3	3	3	8	8	7	10	14	14	18	18	20	20	23	23
	Y	5	5	3	6	8	8	11	11	11	11	11	11	15	15	16	20
%Err	X	200%	0%	25%	25%	14%	11%	0%	0%	8%	0%	6%	5%	0%	5%	5%	4%
	Y	67%	0%	25%	14%	0%	11%	0%	0%	10%	8%	10%	8%	6%	7%	0%	0%

Table 14: Reconstructed values for X-Axis and Y-Axis for 16 tags when all wavelet coefficients are stored and threshold = 1 has been applied

After reconstructing the data from the wavelet coefficients array with threshold set to 25%, the resulting coefficients are shown in Tables 15 and 16 :

	Tags	14	5	9	2	12	7	3	16
Original Values	X Axis	1	3	4	4	7	9	7	10
	Y Axis	3	5	4	7	8	9	11	11
Reconstructed Values	X Axis	1	3	4	4	8	8	8	8
	Y Axis	3	5	4	7	8	8	11	11
%Err	X Axis	0%	0%	0%	0%	14%	11%	14%	20%
	Y Axis	0%	0%	0%	0%	0%	11%	0%	0%

Table 15: Reconstructed values for X-Axis and Y-Axis for 8 tags when all wavelet coefficients are stored and threshold = 25% has been applied

	Tags	14	5	9	2	12	7	3	16	10	11	4	6	15	8	1	13
Original Values	X	1	3	4	4	7	9	7	10	13	14	17	19	20	21	22	24
	Y	3	5	4	7	8	9	11	11	10	12	10	12	16	14	16	20
Constructed Values	X	1	3	4	4	7	9	7	10	14	14	18	18	20	20	23	23
	Y	3	5	4	7	8	9	11	11	11	11	11	11	15	15	18	18
%Err Value	X	0%	0%	0%	0%	0%	0%	0%	0%	8%	0%	6%	5%	0%	5%	5%	4%
	Y	0%	0%	0%	0%	0%	0%	0%	0%	10%	8%	10%	8%	6%	7%	13%	10%

Table 16: Reconstructed values for X-Axis and Y-Axis for 16 tags when all wavelet coefficients are stored and threshold = 1 has been applied

5.2.2 Scenario 2 : Standard Decomposition Method

In this scenario, standard wavelet decomposition method has been applied on multi-dimensional array to generate wavelet coefficient matrix. Standard Decomposition method is applied on an array of size (N/2-1,3) i.e, for 8 tags, it is storing 16 wavelet coefficients in 4 x 4 matrix while for tag 16 it is storing 32 wavelet coefficients in 8 x 4 matrix.

For 8 tags, the matrix structure is :

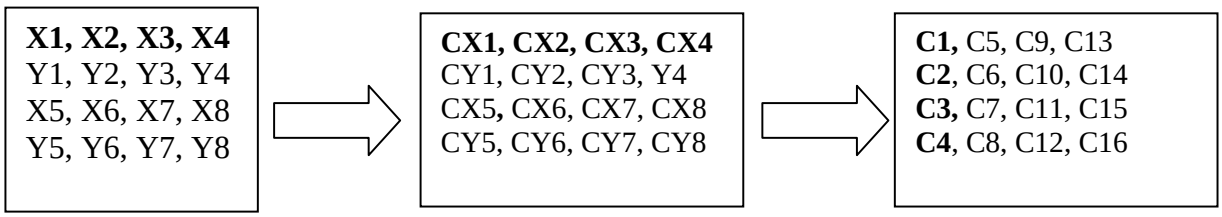
X1, X2, X3, X4
Y1, Y2, Y3, Y4
X5, X6, X7, X8
Y5, Y6, Y7, Y8

The structure for 16 tags matrix is :

X1, X2, X3, X4, X5, X6, X7, X8
Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8
X9, X10, X11, X12, X13, X14, X15, X16
Y9, Y10, Y11, Y12, Y13, Y14, Y15, Y16

To obtain standard decomposition, the one-dimensional wavelet transform is applied to each row. This operation gave an average value along with detail coefficients for each row. Next, these transformed rows are treated as if they were themselves an array and applied the one-dimensional transform to each column. The resulting values are all detail coefficients except for a single overall average coefficient.

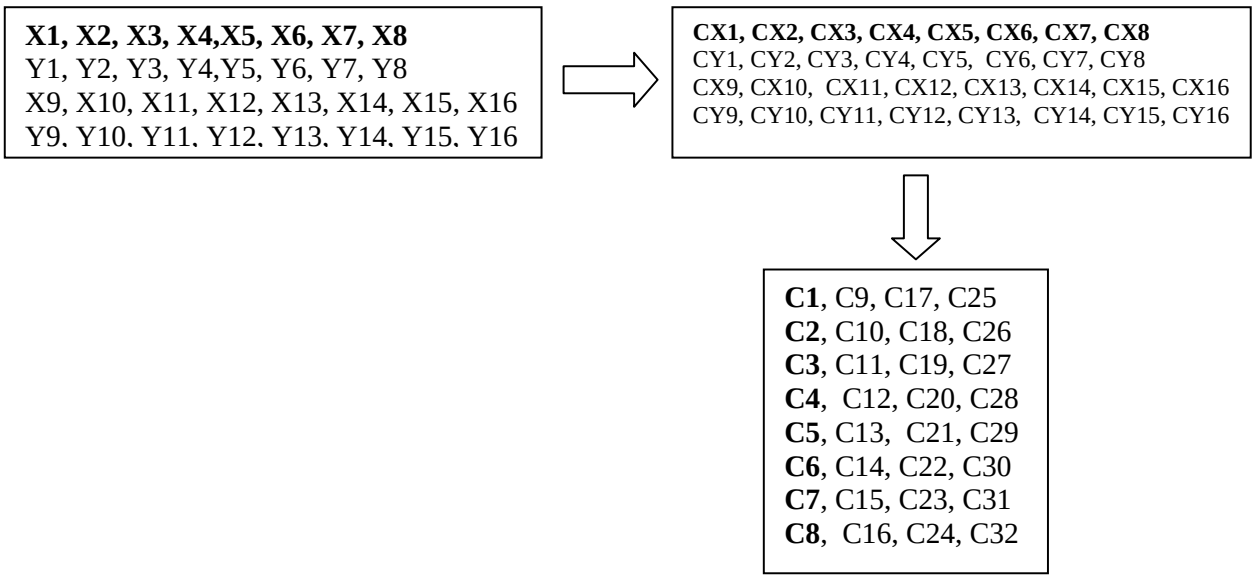
Standard decomposition method for 8 tags is :



After applying standard decomposition method on the original values matrix, we get following wavelet coefficient matrix for 8 tags :

6.5, -2.57, -0.88, -0.75
-0.75, -0.06, -0.13, 0.50
-1, -0.13, 0, -0.25
-1, 0, 0.75, -0.75

Standard decomposition method for 16 tags is :



After applying standard decomposition method on the original values matrix, we get following wavelet coefficient matrix for 16 tags –

$$\begin{pmatrix} 11.34, -4.91, -0.82, 2.5 \\ -2.72, 0.16, -0.06, -0.13 \\ -1, 0.13, -0.13, -1.13 \\ -1.06, 0.32, 0.5, 0.13 \\ -0.88, -0.13, 0, 0.25 \\ -0.88, 0.13, 0.75, 0 \\ -0.25, -0.5, -0.25, -0.75 \\ -1.12, 0.38, -0.75, 0.5 \end{pmatrix}$$

After reconstructing these wavelet coefficients matrices, we get the following coefficients, as shown in Tables 17 and 18.

	Tags	14	5	9	2	12	7	3	16
Original Values	X Axis	1	3	4	4	7	9	7	10
	Y Axis	3	5	4	7	8	9	11	11
Reconstructed Values	X Axis	1	3	4	4	7	9	7	10
	Y Axis	3	5	4	7	8	9	11	11

Table 17: Reconstructed values for X-Axis and Y-Axis for 8 tags when Standard Decomposition method is used and no threshold has been applied

	Tags	14	5	9	2	12	7	3	16	10	11	4	6	15	8	1	13
Ori. Value	X	1	3	4	4	7	9	7	10	13	14	17	19	20	21	22	24
	Y	3	5	4	7	8	9	11	11	10	12	10	12	16	14	16	20
Recon struct Value	X	1	3	4	4	7	9	7	10	13	14	17	19	20	21	22	24
	Y	3	5	4	7	8	9	11	11	10	12	10	12	16	14	16	20

Table 18: Reconstructed values for X-Axis and Y-Axis for 16 tags when Standard Decomposition method is used and no threshold has been applied

Now, after applying different threshold values on original wavelet coefficients matrix, we get the following results, as shown in Tables 19 and 20 :

a. Threshold = 0.50

Wavelet Coefficients Matrix for 8 tags

$$\begin{pmatrix} 6.5, -2.57, -0.88, -0.75 \\ -0.75, 0, 0, 0 \\ -1, -0.13, 0, 0 \\ -1, 0, 0.75, -0.75 \end{pmatrix}$$

	Tags	14	5	9	2	12	7	3	16
Original Values	X Axis	1	3	4	4	7	9	7	10
	Y Axis	3	5	4	7	8	9	11	11
Reconstructed Values	X Axis	1	3	4	4	7	8	7	11
	Y Axis	3	5	4	7	8	10	10	11
%Err	X Axis	0%	0%	0%	0%	0%	11%	0%	10%
	Y Axis	0%	0%	0%	0%	0%	11%	9%	0%

Table 19: Reconstructed values for X-Axis and Y-Axis for 8 tags when Standard Decomposition method is used and threshold =0.50 has been applied

Wavelet Coefficients Matrix for 16 tags

$$\begin{pmatrix} 11.34, -4.91, -0.82, 2.5 \\ -2.72, 0, 0, 0 \\ -1, 0, 0, -1.13 \\ -1.06, 0, 0, 0 \\ -0.88, 0, 0, 0 \\ -0.88, 0, 0.75, 0 \\ 0, 0, 0, 0.75 \\ -1.12, 0, -0.75, 0 \end{pmatrix}$$

	Tags	14	5	9	2	12	7	3	16	10	11	4	6	15	8	1	13
Ori. Value	X	1	3	4	4	7	9	7	10	13	14	17	19	20	21	22	24
	Y	3	5	4	7	8	9	11	11	10	12	10	12	16	14	16	20
Reconstruct Value	X	1	3	4	4	7	7	8	11	13	15	17	19	20	21	21	24
	Y	3	4	4	7	9	9	11	11	10	12	10	12	16	15	16	19
%Err	X	0%	0%	0%	0%	0%	22%	14%	10%	0%	7%	0%	0%	0%	0%	5%	0%
	Y	0%	20%	0%	0%	13%	0%	0%	0%	0%	0%	0%	0%	0%	7%	0%	-5%

Table 20: Reconstructed values for X-Axis and Y-Axis for 16 tags when Standard Decomposition method is used and threshold = 0.50 has been applied

The compression ratio at the 0.50 thresholding level with standard decomposition method is given in Table 21 :

Tags	**Wavelet Coefficients before threshold**	**Wavelet Coefficients after threshold**	**Compression Ratio**
8	6.5, -2.57, -0.88, -0.75 -0.75, -0.06, -0.13, 0.50 -1, -0.13, 0, -0.25 -1, 0, 0.75, -0.75	6.5, -2.57, -0.88, -0.75 -0.75, 0, 0, 0 -1, -0.13, 0, 0 -1, 0, 0.75, -0.75	1.40
16	11.34, -4.91, -0.82, 2.5 -2.72, 0.16, -0.06, -0.13 -1, 0.13, -0.13, -1.13 -1.06, 0.32, 0.5, 0.13 -0.88, -0.13, 0, 0.25 -0.88, 0.13, 0.75, 0 -0.25, -0.5, -0.25, -0.75 -1.12, 0.38, -0.75, 0.5	11.34, -4.91, -0.82, 2.5 -2.72, 0, 0, 0 -1, 0, 0, -1.13 -1.06, 0, 0, 0 -0.88, 0, 0, 0 -0.88, 0, 0.75, 0 0, 0, 0, 0.75 -1.12, 0, -0.75, 0	2.14

Table 21: Compression Ratio for threshold = 0.50 in Scenario 2

b. Threshold = 25%

Wavelet Coefficients Matrix for 8 tags (The fourth column i.e. 0 is not stored in the physical file to save the storage space but during reconstruction process is added to the array). Results are in Table 22.

$$\begin{pmatrix} 6.5, -2.57, -0.88, 0 \\ -0.75, -0.06, -0.13, 0 \\ -1, -0.13, 0, 0 \\ -1, 0, 0.75, 0 \end{pmatrix}$$

	Tags	579	506	376	40	62	790	329	768
Original Values	X Axis	1	3	4	4	7	9	7	10
	Y Axis	3	5	4	7	8	9	11	11
Reconstructed Values	X Axis	1	3	4	4	8	9	9	11
	Y Axis	3	5	4	7	8	9	9	11
%Err	X Axis	0%	0%	0%	0%	14%	0%	29%	10%
	Y Axis	0%	0%	0%	0%	0%	0%	18%	0%

Table 22: Reconstructed values for X-Axis and Y-Axis for 8 tags when Standard Decomposition method is used and threshold = 25% has been applied

Wavelet Coefficients Matrix for 16 tags (Here also the fourth column i.e. 0 is not stored in the physical file to save the storage space but during reconstruction process is added to the array). Results are in Table 23.

$$\begin{pmatrix} 11.34, -4.91, -0.82, 0 \\ -2.72, 0.16, -0.06, 0 \\ -1, 0.13, -0.13, 0 \\ -1.06, 0.32, 0.5, 0 \\ -0.88, -0.13, 0, 0 \\ -0.88, 0.13, 0.75, 0 \\ -0.25, -0.5, -0.25, 0 \\ -1.12, 0.38, -0.75, 0 \end{pmatrix}$$

	Tags	14	5	9	2	12	7	3	16	10	11	4	6	15	8	1	13
Ori. Value	X	1	3	4	4	7	9	7	10	13	14	17	19	20	21	22	24
	Y	3	5	4	7	8	9	11	11	10	12	10	12	16	14	16	20
Reconstruct Value	X	1	3	4	4	7	9	7	10	11	13	13	16	18	18	19	22
	Y	3	5	4	7	8	9	11	11	11	13	13	16	18	18	19	22
%Err	X	0%	0%	0%	0%	0%	0%	0%	0%	15%	7%	24%	16%	10%	14%	14%	8%
	Y	0%	0%	0%	0%	0%	0%	0%	0%	10%	8%	30%	33%	13%	29%	19%	10%.

Table 23: Reconstructed values for X-Axis and Y-Axis for 16 tags when Standard Decomposition method is used and threshold = 25% has been applied

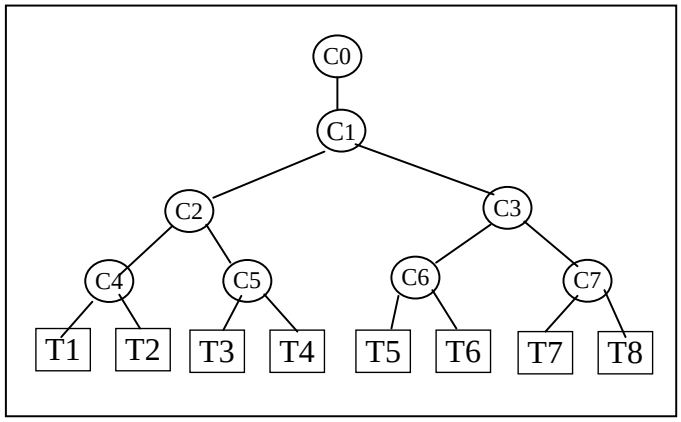
Compression ratios obtained after applying threshold at 25% with the Standard Decomposition method is given in Table 24 :

Tags	Wavelet Coefficients before threshold	Wavelet Coefficients after threshold	Compression Ratio
8	6.5, -2.57, -0.88, -0.75 -0.75, -0.06, -0.13, 0.50 -1, -0.13, 0, -0.25 -1, 0, 0.75, -0.75	6.5, -2.57, -0.88, 0 -0.75, -0.06, -0.13, 0 -1, -0.13, 0, 0 -1, 0, 0.75, 0	1.40
16	11.34, -4.91, -0.82, 2.5 -2.72, 0.16, -0.06, -0.13 -1, 0.13, -0.13, -1.13 -1.06, 0.32, 0.5, 0.13 -0.88, -0.13, 0, 0.25 -0.88, 0.13, 0.75, 0 -0.25, -0.5, -0.25, -0.75 -1.12, 0.38, -0.75, 0.5	11.34, -4.91, -0.82, 0 -2.72, 0.16, -0.06, 0 -1, 0.13, -0.13, 0 -1.06, 0.32, 0.5, 0 -0.88, -0.13, 0, 0 -0.88, 0.13, 0.75, 0 -0.25, -0.5, -0.25, 0 -1.12, 0.38, -0.75, 0	1.30

Table 24: Compression Ratio for threshold = 25% in Scenario 2

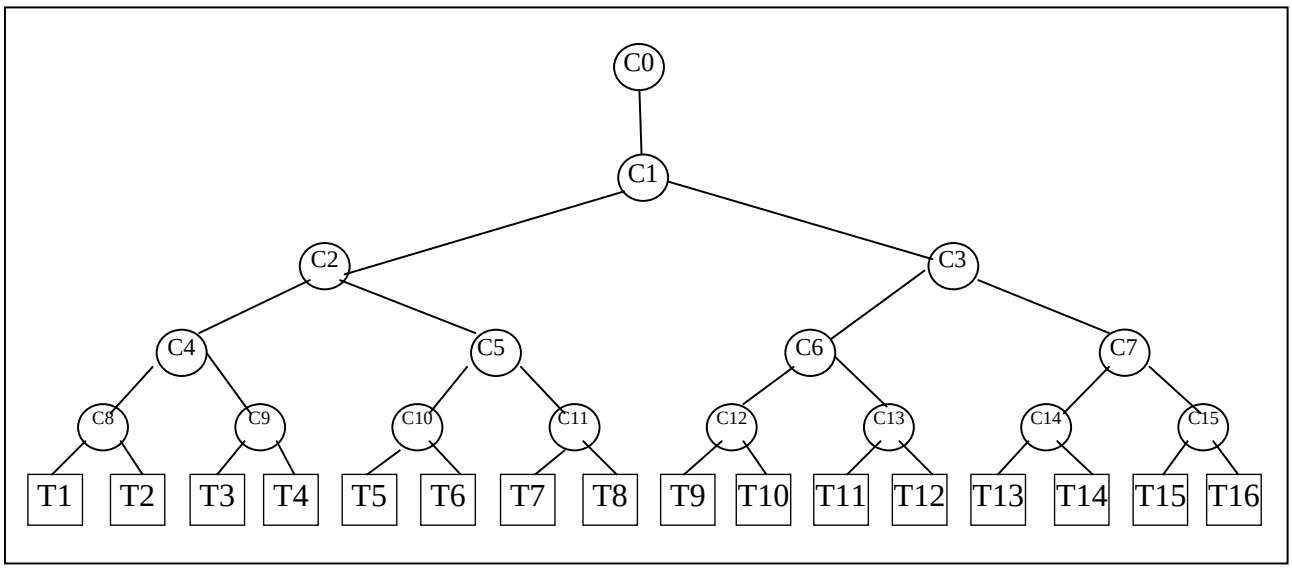
5.2.3 Scenario 3: Tags storing coefficients depending on tags' resolution

In this scenario, each tag contains location information about every other tag. Each tag stores detailed information about its neighboring tags and progressively less detailed information about tags that are further away from it.



Tags	Wavelet Coefficients
T1, T2, T3, T4	C0, C1, C2, C3, C4, C5
T5, T6, T7, T8	C0, C1, C2, C3, C6, C7

In the 8 tags network, there are two groups of tags, Group1 and Group2. Group1 consists of tags T1, T2, T3 and T4 while Group2 consists of tags T5, T6, T7 and T8. Tag T1 consists of exact information about its group members i.e. tags T2, T3 and T4 and fuzzier information about tags in the other group, i.e. Group2, by having just one coefficient which is C3.



Tag	Wavelet Coefficients
T1,T2,T3,T4	C0,C1,C2,C3,C4,C5,C8,C9
T5,T6,T7,T8	C0,C1,C2,C3,C4,C5,C10,C11
T9,T10,T11,T12	C0,C1,C2,C3,C6,C7,C12,C13
T13,T14,T15,T16	C0,C1,C2,C3,C6,C7,C14,C15

In the 16 tags network, tags are divided into three groups. Tags T1, T2, T3, T4 form Group1, while tags T5, T6, T7 and T8 are in Group2 which is the neighboring group to Group 1. Group 3 consists of tags T9 to T16 for which tags in Group 1 will have location information at a low level of resolution. So, tag T1 will have wavelet coefficients such as C0, C1, C2, C4, C8 and C9 which will give exact information about its group members, then C5 coefficient will give less precise information about neighboring group consists of tags T5, T6, T7 and T8 while coefficient C3 will give even less precise information about tags T9 to T16. Hence, tag T1 will end up having C0, C1, C2, C3, C4, C5, C8 and C9 wavelet coefficients.

The graphical representation of 8 tags appears in Figure 5.1 below :

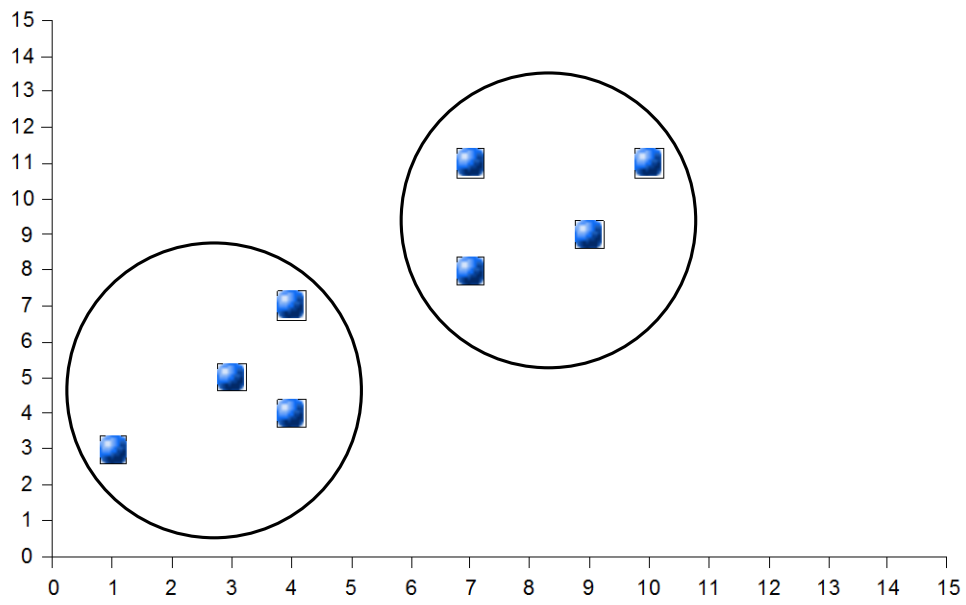


Figure 5.1: Graphical Representation of 8 Tags

The two groups of tags, easily identifiable from Figure 5.1 above stores exact information of group members and fuzzy information of Neighbouring group members. So, after reconstructing the actual data values from wavelet coefficients, we get exact values for X-Axis and Y-Axis of all group members of a given group and approximate or fuzzy values for X-Axis and Y-Axis for members of the other group.

If we consider Group 1, then we will have exact data for Tags 1 to 4 and approximate data for Tags 5 to 8. In case of Group 2, we will have exact data for Tags 5 to 8 and approximate data for Tags 1 to 4.

After data reconstruction from wavelet coefficients, we get the following tag networks. Considering the Group 1 as current group, we will have Tag 1, Tag 2, Tag 3 and Tag4 from Group 1. On the other hand, with respect to Group 2, Tags 5 and 6 coalesce into one unit and Tags 7 and 8 also coalesce, thus yielding two units in this group (See Figure 5.2 below) :

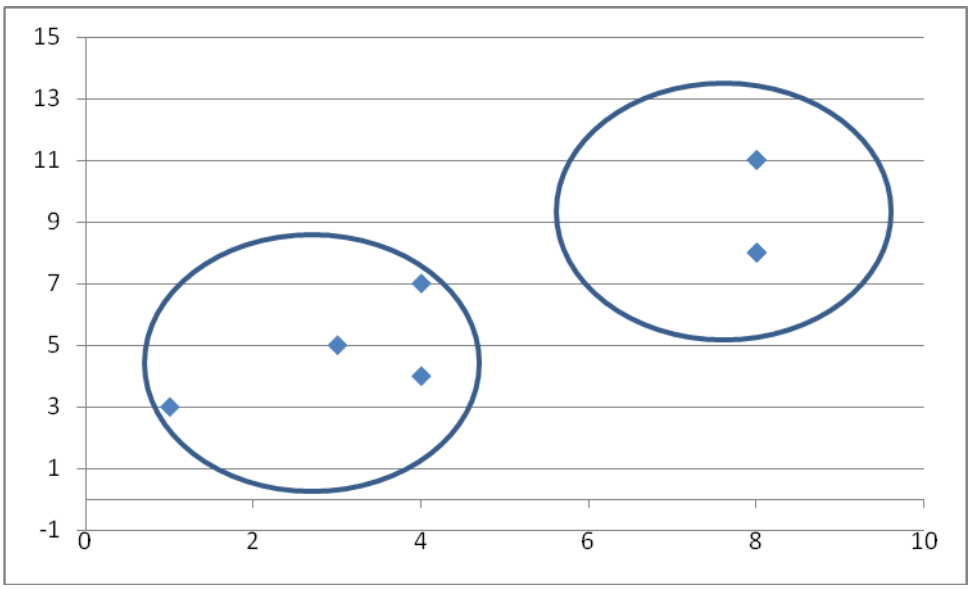


Figure 5.2: Graphical Representation of Group 2 Tags after Data construction in Scenario 3

When considering the Group 2 as current group, a similar situation arises, with pairs of tags coalescing in Group 1, while Group 2 tags remain separate from each other (See Figure 5.3 below) :

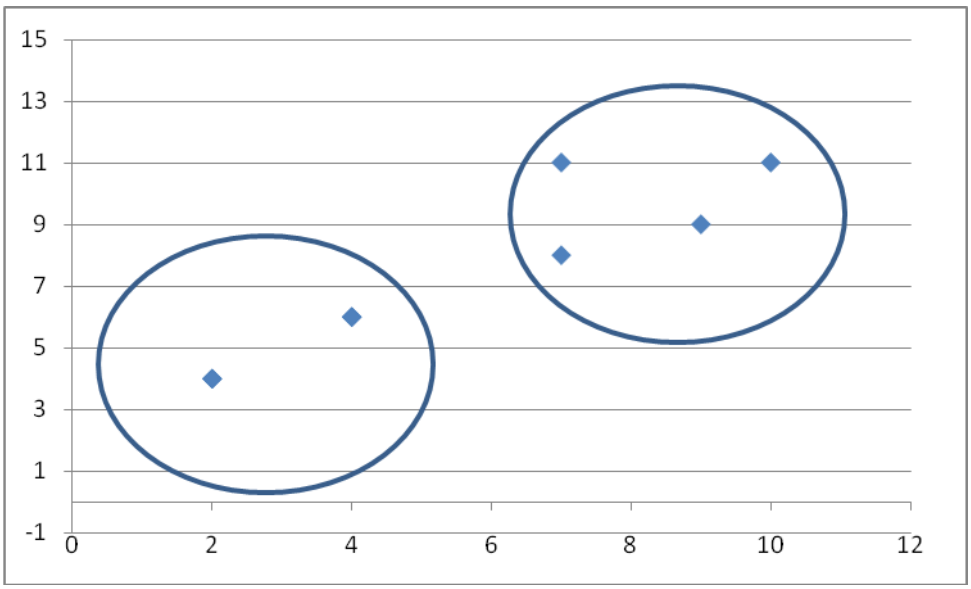


Figure 5.3: Graphical Representation of Group 1 Tags after Data construction in Scenario 3

So, while navigating from the current group to the neighbouring group, the next tag will be calculated from the nearest pair in the path of the neighbouring group. In most cases, the first tag in the sequence will be picked up as the next reachable tag in the path.

As in previous Scenarios, the wavelet compression results for 8 tags and 16 tags configurations for this scenario are analyzed and presented in following two tables (Table 25 and 26).

		Tags	14	5	9	2	12	7	3	16
Original Values		X Axis	1	3	4	4	7	9	7	10
		Y Axis	3	5	4	7	8	9	11	11
Compressed Values	Source Tags 14, 5, 9, 2	X Axis	1	3	4	4	8	8	8	8
		Y Axis	3	5	4	7	8	8	11	11
%Err		X Axis	0%	0%	0%	0%	14%	11%	14%	20%
		Y Axis	0%	0%	0%	0%	0	11%	0	0
Compressed Values	Source Tags 12, 7, 3, 16	X Axis	2	2	4	4	7	9	7	10
		Y Axis	4	4	6	6	8	9	11	11
%Err		X Axis	100%	33%	0%	0%	0%	0%	0%	0%
		Y Axis	33%	20%	50%	14%	0%	0%	0%	0%

Table 25: Reconstructed values for X-Axis and Y-Axis for 8 tags when Wavelet Coefficients as per the tag resolution are stored

For 16 tags :

	Tags	14	5	9	2	12	7	3	16	10	11	4	6	15	8	1	13
Ori. Value	X	1	3	4	4	7	9	7	10	13	14	17	19	20	21	22	24
	Y	3	5	4	7	8	9	11	11	10	12	10	12	16	14	16	20

Comp value	Src Tag 14,5,9,2	X	1	3	4	4	8	8	8	8	16	16	16	16	22	22	22	22
		Y	3	5	4	7	8	8	11	11	11	11	11	11	16	16	16	16
%Err		X	0%	0%	0%	0%	14%	11%	14%	20%	23%	14%	6%	16%	`0%	5%	0%	8%
		Y	0%	0%	0%	0%	0%	11%	0%	0%	10%	8%	10%	8%	0%	14%	0%	20%
Comp value	Src Tag 12, 7, 3, 16	X	2	2	4	4	7	9	7	10	16	16	16	16	22	22	22	22
		Y	4	4	6	6	8	9	11	11	11	11	11	11	16	16	16	16
%Err		X	100 %	33%	0%	0%	0%	0%	0%	0%	23%	14%	6%	16%	10%	5%	0%	8%
		Y	33%	20%	50%	14%	0%	0%	0%	0%	10%	8%	10%	8%	0%	14%	0%	20%

Comp value	Src Tag 10, 11, 4, 6	X	3	3	3	3	8	8	8	8	13	14	17	19	20	20	23	23
		Y	5	5	5	5	10	10	10	10	10	12	10	12	15	15	18	18
%Err		X	200 %	0%	25%	25%	14%	11%	14%	20%	0%	0%	0%	0%	0%	5%	5%	4%
		Y	67%	0%	25%	29%	25%	11%	9%	9%	0%	0%	0%	0%	6%	7%	13%	10%
Comp value	Src Tag 15, 8, 1, 13	X	3	3	3	3	8	8	8	8	14	14	18	18	20	21	22	24
		Y	5	5	5	5	10	10	10	10	11	11	11	11	16	14	16	20
%Err		X	200 %	0%	25%	25%	14%	11%	14%	20%	8%	0%	6%	5%	0%	0%	0%	0%
		Y	67%	0%	25%	29%	25%	11%	9%	9%	10%	8%	10%	8%	0%	0%	0%	0%

Table 26: Reconstructed values for X-Axis and Y-Axis for 16 tags when Wavelet Coefficients as per the tag resolution are stored

With this exercise, it is concluded that the error is 0% for group members, small for neighboring group tags and highest for the furthest group. Compression ratios obtained with the 8 and 16 tag networks are given in the Table 27 below:

Tags	Number of Wavelet Coefficients	Number of Wavelet Coefficients stored in each tag	Compression Ratio
8	8	6	1.33
16	16	8	2

Table 27: Compression Ratio in Scenario 3

5.2.4 Scenario 4: Non-Standard Decomposition Algorithm

In this scenario, Non-standard wavelet decomposition method has been applied on multi-dimensional array to generate wavelet coefficient matrix. Non-Standard Decomposition method is applied on a data array similar to the Scenario 2 i.e. Standard Decomposition Method.

For 8 tags, the matrix structure is:

$$\begin{pmatrix} X1, X2, X3, X4 \\ Y1, Y2, Y3, Y4 \\ X5, X6, X7, X8 \\ Y5, Y6, Y7, Y8 \end{pmatrix}$$

The structure for 16 tags matrix is :

$$\begin{pmatrix} X1, X2, X3, X4, X5, X6, X7, X8 \\ Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8 \\ X9, X10, X11, X12, X13, X14, X15, X16 \\ Y9, Y10, Y11, Y12, Y13, Y14, Y15, Y16 \end{pmatrix}$$

To obtain non-standard decomposition, the one-dimensional wavelet transform is applied by averaging and differencing pair-wise data elements to each row. This operation gave an average value along with detail coefficients for each row. Next, these transformed rows are treated as if they are themselves an array and applied the one-dimensional transform again to each column. All the resultant coefficients are detailed coefficients except one which is overall average. This process is repeated continuously on the section with all average values.

After applying standard decomposition method on the original values matrix, we get following wavelet coefficient matrix for 16 tags :

$$\begin{pmatrix} 11.34, -1.03, -0.56, -1 \\ -4.91, 0.22, -0.31, 0.25 \\ -2.56, -0.06, -0.12, 0 \\ -2.88, 0.12, -0.5, 0.25 \\ -0.88, -0.12, 0, 0.75 \\ -0.75, 0.5, -0.25, -0.75 \\ 2.38, -1.12, 0.25, 0 \\ 2.62, 0.12, -0.75, 0.5 \end{pmatrix}$$

After reconstructing these wavelet coefficients matrices, we get the following compressed values for 16 tags :

	Tags	14	5	9	2	12	7	3	16	10	11	4	6	15	8	1	13
Ori. Value	X	1	3	4	4	7	9	7	10	13	14	17	19	20	21	22	24
	Y	3	5	4	7	8	9	11	11	10	12	10	12	16	14	16	20
Recon struct Value	X	1	3	4	4	7	9	7	10	13	14	17	19	20	21	22	24
	Y	3	5	4	7	8	9	11	11	10	12	10	12	16	14	16	20

Table 28: Reconstructed values for X-Axis and Y-Axis for 16 tags when Non-standard Decomposition method is used and no threshold has been applied

Now, after applying different threshold values on original wavelet coefficients matrix, we get following results:

a. Threshold = 0.50

Wavelet Coefficients Matrix for 16 tags

$$\begin{pmatrix} 11.34, -1.03, -0.56, -1 \\ -4.91, 0, 0, 0 \\ -2.56, 0, 0, 0 \\ -2.88, 0, 0, 0 \\ -0.88, 0, 0, 0.75 \\ -0.75, 0, 0, -0.75 \\ 2.38, -1.12, 0 \\ 2.62, 0, -0.75, 0 \end{pmatrix}$$

	Tags	14	5	9	2	12	7	3	16	10	11	4	6	15	8	1	13
Ori. Value	X	1	3	4	4	7	9	7	10	13	14	17	19	20	21	22	24
	Y	3	5	4	7	8	9	11	11	10	12	10	12	16	14	16	20
Reconstruct Value	X	1	3	4	4	7	8	8	11	13	14	17	19	19	22	22	24
	Y	3	4	4	8	8	9	11	11	11	12	10	12	16	15	17	19
%Err	X	0%	0%	0%	0%	0%	11%	14%	10%	0%	0%	0%	0%	5%	5%	0%	0%
	Y	0%	20%	0%	14%	0%	0%	0%	0%	10%	0%	0%	0%	0%	7%	6%	5%

Table 29: Reconstructed values for X-Axis and Y-Axis for 16 tags when Non-standard Decomposition method is used and threshold = 0.50 has been applied

Compression Ratio after applying threshold = 0.50 with the standard decomposition method is:

Tags	**Wavelet Coefficients before threshold**	**Wavelet Coefficients after threshold**	**Compression Ratio**
16	11.34, -1.03, -0.56, -1 -4.91, 0.22, -0.31, 0.25 -2.56, -0.06, -0.12, 0 -2.88, 0.12, -0.5, 0.25 -0.88, -0.12, 0, 0.75 -0.75, 0.5, -0.25, -0.75 2.38, -1.12, 0.25, 0 2.62, 0.12, -0.75, 0.5	11.34, -1.03, -0.56, -1 -4.91, 0, 0, 0 -2.56, 0, 0, 0 -2.88, 0, 0, 0 -0.88, 0, 0, 0.75 -0.75, 0, 0, -0.75 2.38, -1.12, 0 2.62, 0, -0.75, 0	1.93

Table 30: Compression Ratio for threshold = 0.50 in Scenario 4

b. Threshold = 25%

Wavelet Coefficients Matrix for 16 tags (Here also the fourth column i.e. 0 is not stored in the physical file to save the storage space but during reconstruction process, that 0 is added in the array)

(11.34, -1.03, -0.56, 0
-4.91, 0.22, -0.31, 0
-2.56, -0.06, -0.12, 0
-2.88, 0.12, -0.5, 0
-0.88, -0.12, 0, 0
-0.75, 0.5, -0.25, 0
2.38, -1.12, 0.25, 0
2.62, 0.12, -0.75, 0)

So, after encoding these wavelet coefficients, we get following reconstructed values for X and Y co-ordinates in Table 31 below :

	Tags	14	5	9	2	12	7	3	16	10	11	4	6	15	8	1	13
Ori. Value	X	1	3	4	4	7	9	7	10	13	14	17	19	20	21	22	24
	Y	3	5	4	7	8	9	11	11	10	12	10	12	16	14	16	20
Reconstruct Value	X	1	3	4	4	7	9	8	8	13	14	18	18	20	21	23	23
	Y	3	5	6	6	8	9	11	11	10	12	11	11	16	14	18	18
%Err	X	0%	0%	0%	0%	0%	0%	14%	20%	0%	0%	6%	5%	0%	0%	5%	4%
	Y	0%	0%	50%	14%	0%	0%	0%	0%	0%	0%	10%	8%	0%	0%	13%	10%

Table 31: Reconstructed values for X-Axis and Y-Axis for 16 tags when Non-standard Decomposition method is used and threshold = 25% has been applied

So, Compression Ratio after applying threshold = 25% on Non-standard Decomposition method is :

Tags	**Wavelet Coefficients before threshold**	**Wavelet Coefficients after threshold**	**Compression Ratio**
16	11.34, -1.03, -0.56, -1 -4.91, 0.22, -0.31, 0.25 -2.56, -0.06, -0.12, 0 -2.88, 0.12, -0.5, 0.25 -0.88, -0.12, 0, 0.75 -0.75, 0.5, -0.25, -0.75 2.38, -1.12, 0.25, 0 2.62, 0.12, -0.75, 0.5	11.34, -1.03, -0.56, 0 -4.91, 0.22, -0.31, 0 -2.56, -0.06, -0.12, 0 -2.88, 0.12, -0.5, 0 -0.88, -0.12, 0, 0 -0.75, 0.5, -0.25, 0 2.38, -1.12, 0.25, 0 2.62, 0.12, -0.75, 0	1.26

Table 32: Compression Ratio for threshold = 25% in Scenario 4

Now, it is necessary to check the effect of compressing location information on navigation. The efficacy of these compression methods are dependent on how successful the navigation algorithm can find the next tag to move to. For the compression to be effective, the navigation path should be very similar if not identical to the path followed when the original coordinates for X-Axis and Y-Axis are stored in the RFID tag, instead of their compressed versions.

5.3 Navigation Experiments

The research question under investigation in this part of the research is:

Can the wavelet compression in RFID tags support efficient traversal between a given source and target node in an RFID network?

In this section, the navigation algorithm is discussed and tested. This algorithm identifies the next tag to be travelled in the path from source to destination tag. To test the algorithm, we use four different scenarios as used in compression experiments and each scenario will use two configurations, each consisting of 8 tags and 16 tags. The first scenario to be discussed and tested is that all wavelet coefficients can be stored fully in the RFID tag. Second scenario represents the situation when wavelet coefficients are stored using the standard decomposition method, while in the third scenario the compression is controlled by storing data at different levels of resolution which is determined by distance from a given reference point. The fourth scenario uses the non-standard wavelet decomposition method.

All the four scenarios use the same data. The files for experiments were generated as described in the previous section on compression using Haar wavelets. For testing the 8-tag configuration, the first eight tags are used. The actual data of 16 tags is:

Tags	X Axis	Y Axis
14	1	3
5	3	5
9	4	4
2	4	7
12	7	8
7	9	9
3	7	11
16	10	11
10	13	10
11	14	12
4	17	10
6	19	12
15	20	16
8	21	14
1	22	16
13	24	20

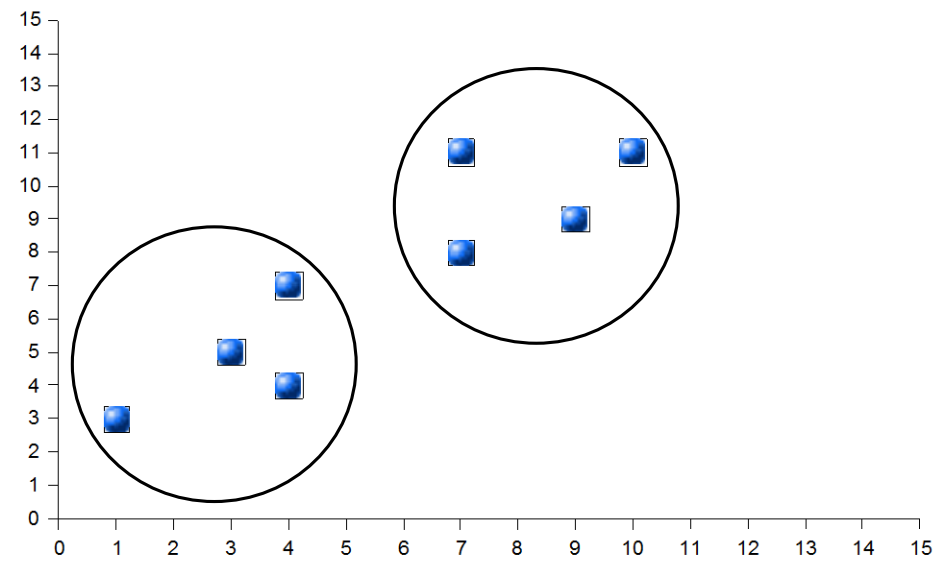
5.3.1 Scenario 1: All Coefficients are stored in the tag

In this scenario, as stated earlier it is assumed that all wavelet coefficients are stored in each RFID tag.

Experiment 1 : This experiment uses 8 tags as follows :

Tags	Label	X Axis	Y Axis
T1	14	1	3
T2	5	3	5
T3	9	4	4
T4	2	4	7
T5	12	7	8
T6	7	9	9
T7	3	7	11
T8	16	10	11

The graphical representation of these tags is :



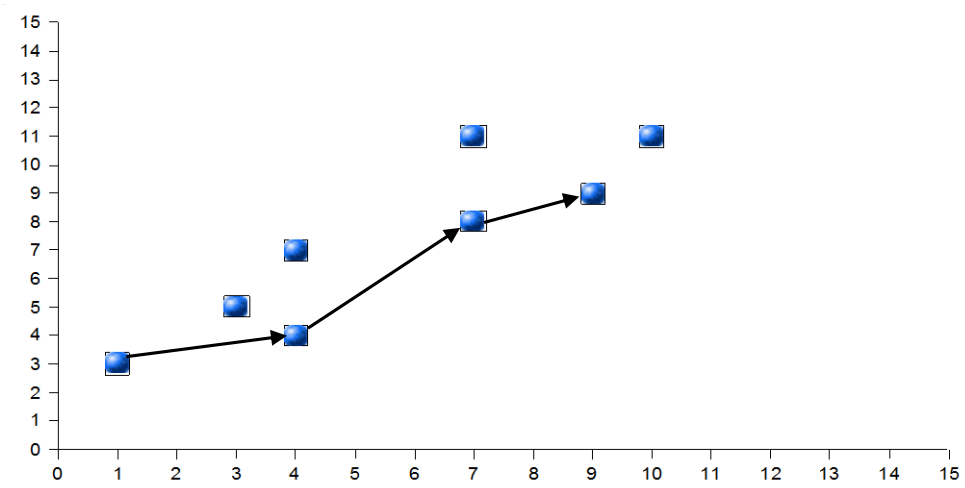
Each tag will store all wavelet coefficients in a lossless compression mode. For calculation of distance and angle, we use the X-Axis and Y-Axis wavelet coefficients. They are (5.62,-2.62,-1,-0.25,-1,0,-1,-1.5) and (7.25,-2.50,-0.75,-1.25,-1,-1.5,-0.5,0) respectively. The navigation algorithm first reconstructs all X-Axis and Y-Axis co-ordinates using these wavelet coefficients.

In the first test, the objective is to find a path between T1 (Source Tag) and T6 (Destination Tag). The distance between T1 and T6 is 10 and angle is 28.76. The algorithm

calculates the next tag to move towards T6 and it is tag T3. The distance between T3 and T1 is 3.16 and the angle between T1 and T3 is 10.32. Now, source tag is T3 and the destination tag is still T6. So next nearest tag from T3 towards T6 is tag T5 which is 5 units away from tag T3 at the angle of 45.02. The next run has T5 as the source tag. The algorithm then moves to tag T6 (destination tag) which is 2.24 units away from tag T5 with an angle of 18.45. So, to reach from tag T6 from tag T6, we have to travel total 10.40 units as compared to 10 units with the original uncompressed data (see table below).

Source Tag	Next Tag	Angle between source to Next	Angle between Source to Destination	Distance between Source to Destination	Distance to travel
T1	T3	10.32	28.76	10	3.16
T3	T5	45.02	36.89	7.07	5.00
T5	T6	18.45	18.45	2.24	2.24
				Total Distance Travelled	10.40
				Excess Distance Travelled	0.40
				Percentage Excess Travelled	4%

So, the shortest travel path from tag T1 to the destination tag T6 is:



This algorithm is also tested with finding the path from tag T2 to tag T7 and the path from tag T4 to tag T8. The following table describes the findings and percentage excess travel to reach destination path in all these three tests.

Experiment No	Source Tag	Destination Tag	Distance to Travel	Tags Travelled	Actual Distance Travelled	Difference	Excess Percentage
1	T1	T6	10	T1, T3, T5, T6	10.40	0.40	4%
2	T2	T7	9.22	T2, T4, T5, T6, T7	9.88	0.88	9.54%
3	T4	T8	7.21	T4, T5, T6, T8	7.64	0.43	5.96%

Table 33: Experiment Readings using 8 tags in Scenario 1

Now, Experiment 1 above is repeated using *Lossy data compression* with threshold values 1, 0.50 and 25%. With threshold value 1, all wavelet coefficients are replaced with 0 if absolute value of coefficient is less than or equal to 1. With threshold value 0.50, all wavelet coefficients are replaced with 0 if absolute value of coefficient is less than or equal to 0.50. With 25% threshold, each RFID tag assume to be stored only 75% of all wavelet coefficients i.e. for 8 tags, each tag will store 6 wavelet coefficients for the data instead of all.

Using threshold = 1, each tag stores (5.62,-2.62,0,0,0,0,-1.5) and (7.25,-2.50,0,-1.25,0,-1.5,0,0) for X-Axis and Y-Axis respectively which in turn after reconstruction is giving (3, 3, 3, 3, 8, 8, 7, 10) and (5, 5, 3, 6, 8, 8, 11, 11) coordinates for X-Axis and Y-Axis. The tags in the path to reach tag t6 from T1 in this test are T1, T5, T7, T6 and travelled 11.35 distance as compare to original 5.83 distance. For threshold = 0.50 each tag stores (5.62, -2.62, -1, 0, -1, 0, -1, -1.5) and (7.25, -2.50, -0.75, -1.25, -1, -1.5, 0, 0) for X-Axis and Y-Axis respectively. After reconstructing these wavelet coefficients, we get (1, 3, 4, 4, 7, 9, 7, 10) and (3, 5, 4, 7, 8, 8, 11) coordinates for X-Axis and Y-Axis respectively. The algorithm gives tags T1, T3, T5, T7, T6 in the path of travelling from source tag T1 to destination tag T6. The distance travelled in this path is 14.16 as compare to original distance of 9.43. Using threshold = 25%, each tag stores (5.62,-2.62,-1,-0.25,-1,0,0,0) and (7.25,-2.50,-0.75,-1.25,-1,-1.5,0,0) respectively.

Threshold	Source Tag	Destination Tag	Distance to Travel	Tags Travelled	Actual Distance Travelled	Difference	Excess Percentage
1	T1	T6	5.83	T1, T5, T7, T6	11.35	5.52	94.68%
0.50	T1	T6	9.43	T1, T3, T5, T7, T6	14.16	4.73	50.16%
25%	T1	T6	9.22	T1, T3, T5, T6	10.56	1.34	14.53%

Table 34: Experiment Readings using 8 tags using different threshold values in Scenario 1

Here, after analyzing data compression done in Scenario 1 and in navigation algorithm, it is observed that threshold value 1 gives worse performance in %Err (Max 200%) while reconstructing the data as well as excess percentage (94.58%) in navigation. Hence, this

threshold value 1 is not used in other experiments.

Experiment 2: In this experiment, 16 tags are being used as stated above.

Each tag will store all wavelet coefficients *Lossless data compression*. For calculation of distance and angle, we are using X-Axis coefficients and Y-Axis coefficients. They are (12.19,-6.56,-2.62,-3.00,-1,-0.25,-2.25,-1.25,-1,0,-1,-1.5,-0.5,-1,-0.5,-1) and (10.5,-3.25,-2.50,-2.75,-0.75,-1.25,0,-1.5,-1,-1.5,-0.5,0,-1,-1,1,-2) respectively. Threshold values used in this experiment are 0, 0.50 and 25%.

a. Threshold = 0

This represents the lossless method.

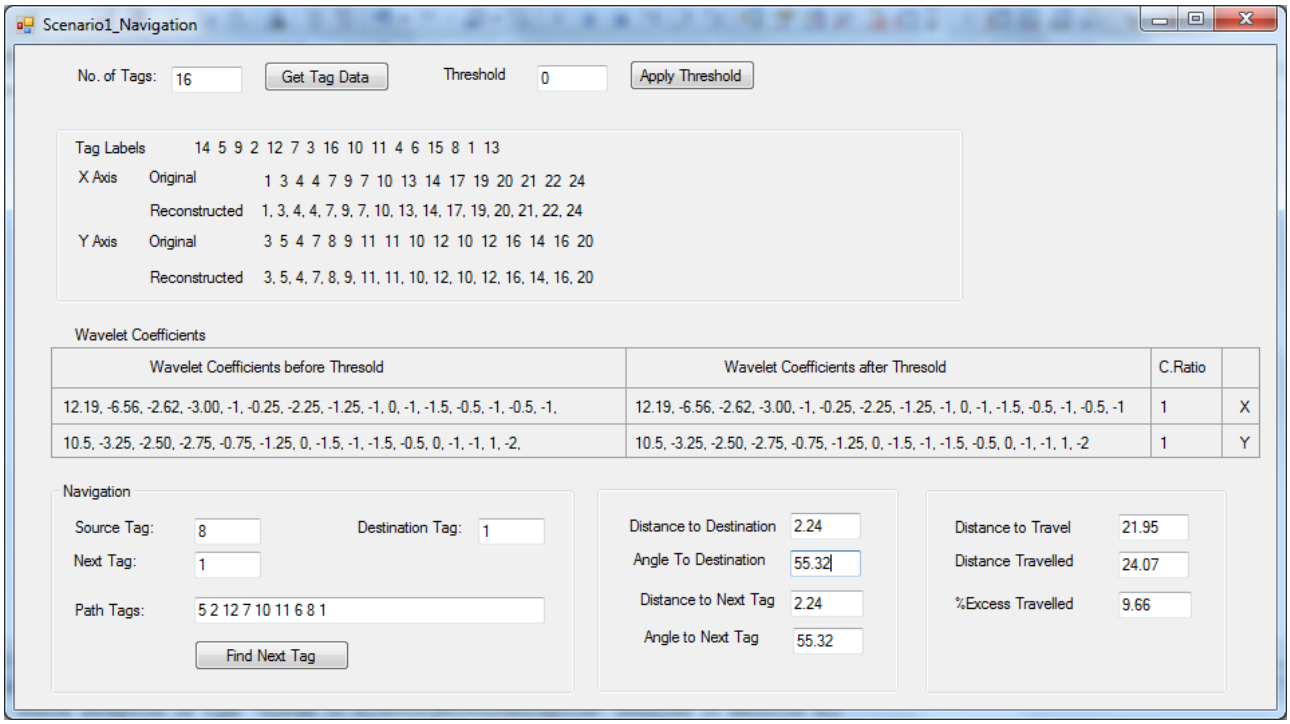


Figure 5.4: Screenshot for applying 0 threshold in Scenario 1

In the first test, the objective is to reach tag T2 from tag T15. The linear distance between these two tags is 21.95 with angle 21.77. The algorithm finds tag T4 as the first tag to move at a distance 2.24 at an angle 55.32. Now, tag T4 becomes the new source tag and the algorithm finds that tag T5 is the next tag to move to at a distance 3.16 with an angle of 10.32. Tag T6 is the next tag from tag T5 at angle of 18.45 and 2.24 units away from tag T5. Tag T9 is the next tag to move from tag T6 which is 4.12 units away from tag T6 with an angle of 5.92. The algorithm next finds tag T10 as the next tag from T9 at a distance of 2.24 units with an angle of 55.32. After tag T10, tag T12 is the next tag at a distance of 5 units from T10 at an angle of 180. From tag T12, the algorithm finds Tag T14 is the next tag to move to reach the destination tag T15 and the distance travelled from tag12 to tag T14 is 2.83 units and finally the destination tag T15 is

reached from tag T14 which is at a distance of 2.24 units. The total distance travelled is 24.07 to reach from the source tag T2 to the destination tag T15 which is 2.12 units (9.66% excess) more as compared to the distance to be travelled without compression.

Source Tag	Next Tag	Angle between source to Next	Angle between Source to Destination	Distance between Source to Destination	Distance to travel
T2	T4	55.32	21.77	21.95	2.24
T4	T5	10.32	18.33	20.12	3.16
T5	T6	18.45	20.05	17	2.24
T6	T9	5.92	20.05	14.76	4.12
T9	T10	55.32	25.78	10.82	2.24
T10	T12	180	18.33	8.94	5
T12	T14	36.89	45.26	5	2.83
T14	T15	55.32	55.32	2.24	2.24
				Total Distance Travelled	24.07
				Excess Distance Travelled	2.12
				Percentage Excess Travelled	9.66%

And the travel path from the source tag T2 to the destination tag T15 is (Figure 5.5) :

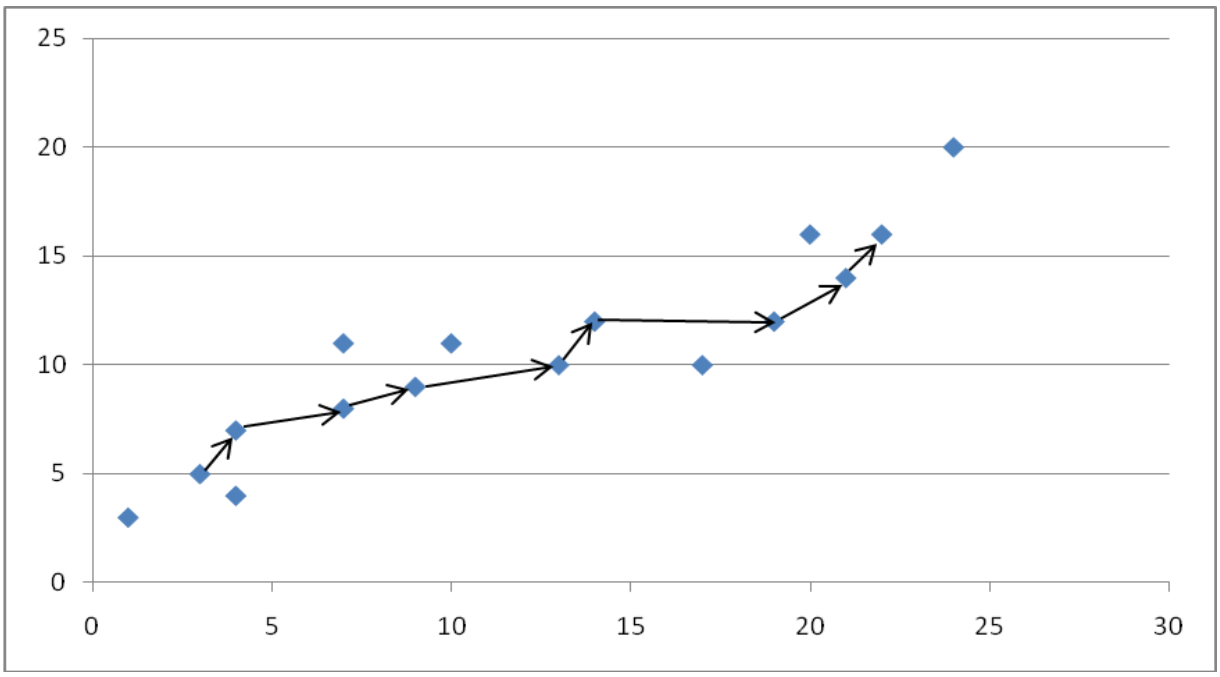


Figure 5.5 : Navigation Path for applying 0 threshold in Scenario 1

This algorithm is also tested with finding path from Tag T3 to Tag T12 and path from Tag T1 to Tag T13. Following table describes the findings and percentage excess travel to reach destination path in all these three tests :

Experiment No	Source Tag	Destination Tag	Distance to Travel	Tags Travelled	Actual Distance Travelled	Difference	Excess Percentage
1	T2	T15	21.95	T2, T4, T5, T6, T9, T10, T12, T14, T15	24.07	2.12	9.66%
2	T3	T12	17.00	T3, T4, T5, T6, T9, T10, T12	19.76	2.76	16.24%
3	T1	T13	23.02	T1, T2, T5, T6, T9, T10, T12, T13	25.55	2.53	10.99%

Table 35: Experiments Readings for applying threshold = 0 in 16-tags network in Scenario 1

Now, this testing is done again using *Lossy data compression* with threshold values 0.50 and 25%.

b. Threshold = 0.50

In threshold value 0.50, all wavelet coefficients are replaced with 0 if absolute value of coefficient is less than or equal to 0.50. Using threshold = 0.50, each tag stores (12.19,-6.56,-2.62,-3,-1,0,-2.25,-1.25,-1,0,-1,-1.5,0,-1,0,-1) and (10.5,-3.25,-2.5,-2.75,-0.75,-1.25,0,-1.5,-1,-1.5,0,0,-1,-1,1,-2) wavelet coefficients for X-Axis and Y-Axis respectively which in turn after reconstruction is giving (1,3,4,4,7,9,7,10,14,14,17,19,20,21,22,24) and (3,5,4,7,8,8,11,11,10,12,10,12,16,14,16,20) coordinates for X-Axis and Y-Axis.

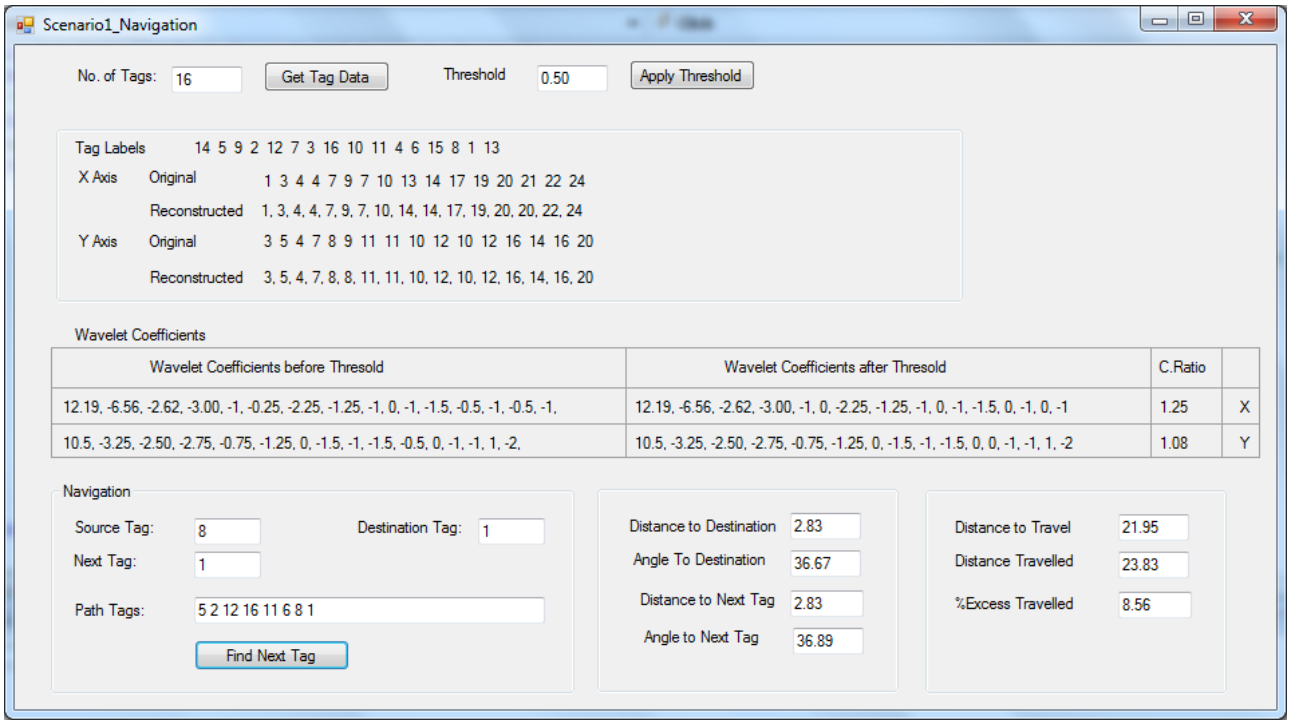


Figure 5.6 : Screenshot for applying 0.50 threshold in Scenario 1

After applying threshold = 0.50 in this scenario, the navigation algorithm gives the linear distance between the source tag T2 and the destination tag T15 as 21.95 units. Tag T4 is the first tag to move at a distance of 2.24 with the angle 55.32. The path contains tag T5 to move next from T4 which is 3.16 units from tag T4 with the angle of 10.32. From tag T5, algorithm shows tag T8 as a next tag and the distance between tag T5 and T8 is calculated as 4.24 units and angle is 38.89 between these two tags. From tag T8, tag T10 is the next tag on the navigation path at a distance of 4.12 units and angle is 5.92. The algorithm shows tag T12 is the next tag on the path, 5 units away from the current source tag T10 with angle 180. The last tag in path before the destination is tag T14 which is 2.24 units far from tag T12 and the angle between two tags is 55.32. Finally, tag T15 is reached from tag T14 with a distance of 2.83 units and angle as 36.89. The total distance travelled in the experiment is 23.83 units which is 1.88 units more (8.56% excess) than the linear distance.

The graphical representation of this navigation path is shown in Figure 5.7 below:

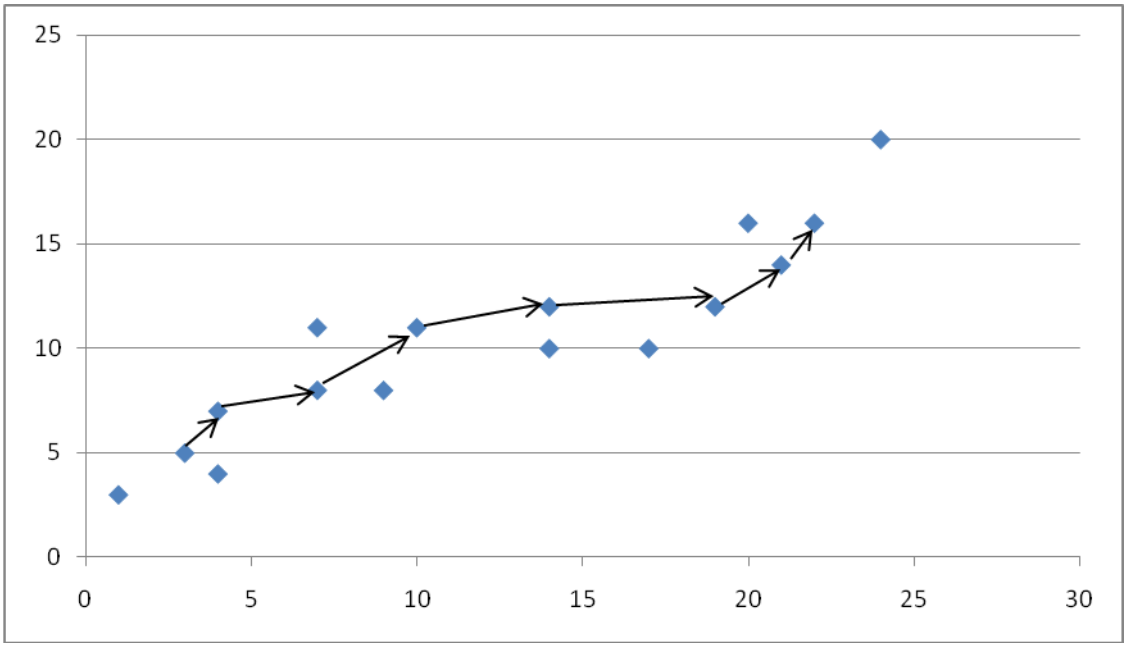


Figure 5.7 : Navigation Path for applying 0.50 threshold in Scenario 1

This algorithm is also tested with finding path from Tag T3to Tag T12 and path from Tag T1 to Tag T13. Following table describes the findings and percentage excess travel to reach destination path in all these three tests :

Experiment No	Source Tag	Destination Tag	Distance to Travel	Tags Travelled	Actual Distance Travelled	Difference	Excess Percentage
1	T2	T15	21.95	T2, T4, T5, T8, T10, T12, T15	23.83	1.88	8.56%
2	T3	T12	17.00	T3, T4, T5, T8, T10, T12	19.52	2.52	14.82%
3	T1	T13	23.02	T1, T2, T5, T7, T8, T10, T12, T13	27.07	4.05	17.59%

Table 36: Experiments Readings for applying threshold = 0.50 in 16-tags network in Scenario 1

c. Threshold = 25%

With 25% threshold, each RFID tag assume to be stored only 75% of all wavelet coefficients i.e. for 8 tags, each tag will store 6 wavelet coefficients for the data instead of all. 12 coefficients are stored if number of tags are 16. Using threshold = 25%, each tag stores (12.19,-6.56,-2.62,-3,-1,-0.25,-2.25,-1.25,-1,0,-1,-1.5) and (10.5,-3.25,-2.50,-2.75,-0.75,-1.25,0,-1.5,-1,-1.5,-0.5,0) for X-Axis and Y-Axis respectively. After wavelet coefficients reconstruction, we get X-Axis coordinates as: (1,3,4,4,7,9,7,10,14,14,18,18,20,20,23,23) and Y-Axis coordinates as: (3,5,4,7,8,9,11,11,11,11,11,11,15,15,18,18).

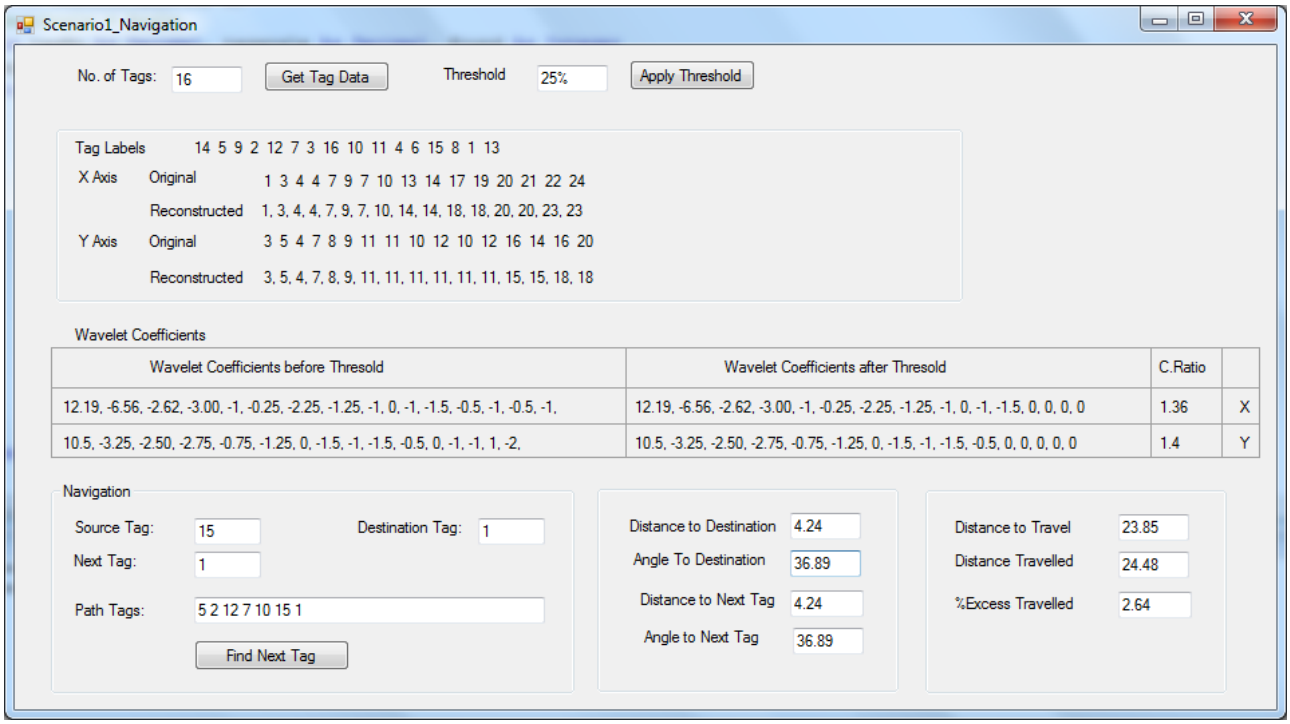


Figure 5.8 : Screenshot for applying 25% threshold in Scenario 1

The total path length (linear distance) from tag T2 to tag T15 is 23.85 units. From source tag T2 to the destination T15, the path travelled through tags T4, T5, T6, T9, T13 i.e. number of hops is 5. The total distance travelled is 24.48 units which is 2.64% excess as compare to the

linear distance which is 23.85 units. The graphical representation of the travel using threshold = 25% is shown below in Figure 5.9 :

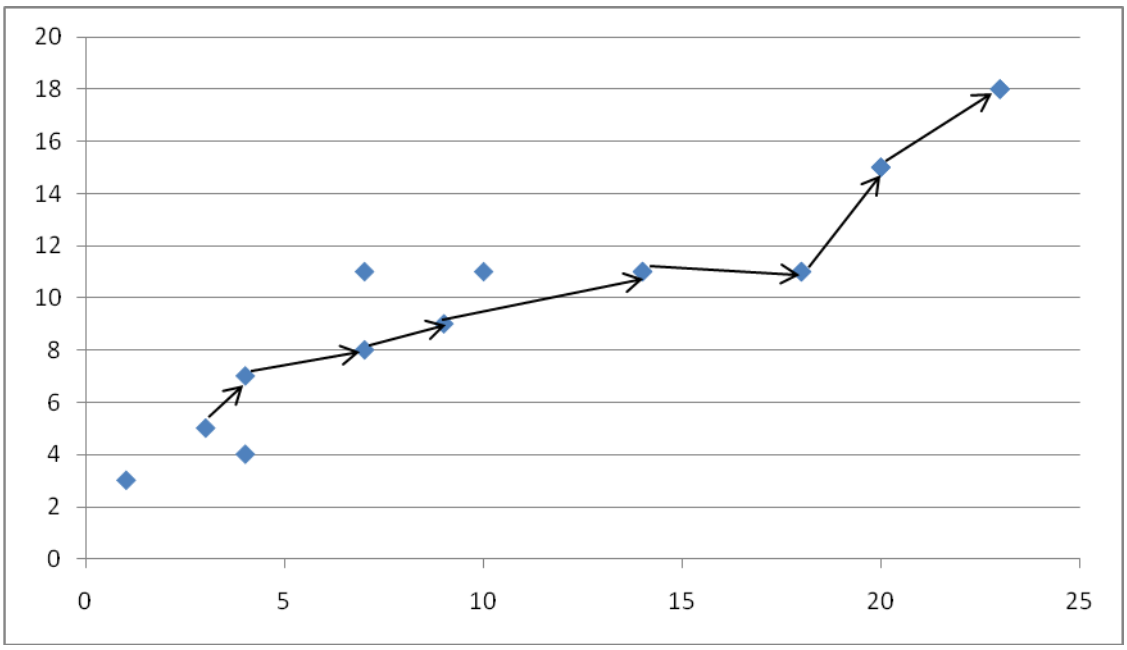


Figure 5.9 : Navigation Path for applying 25% threshold in Scenario 1

This algorithm is also tested with finding the path from Tag T3 to Tag T12 and the path from Tag T1 to Tag T13. Following table describes the findings and percentage excess travel to reach destination path in all these three tests :

Experiment No	Source Tag	Destination Tag	Distance to Travel	Tags Travelled	Actual Distance Travelled	Difference	Excess Percentage
1	T2	T15	23.85	T2, T4, T5, T6, T9, T13	24.48	0.63	2.64
2	T3	T12	15.65	T3, T4, T5, T6, T9, T11, T12	17.79	1.54	13.67%
3	T1	T13	22.47	T1, T2, T5, T6, T8, T9, T13	23.52	1.05	4.67%

Table 37: Experiments Readings for applying threshold = 25% in 16-tags network in Scenario 1

5.3.2 Scenario 2: Standard Decomposition method

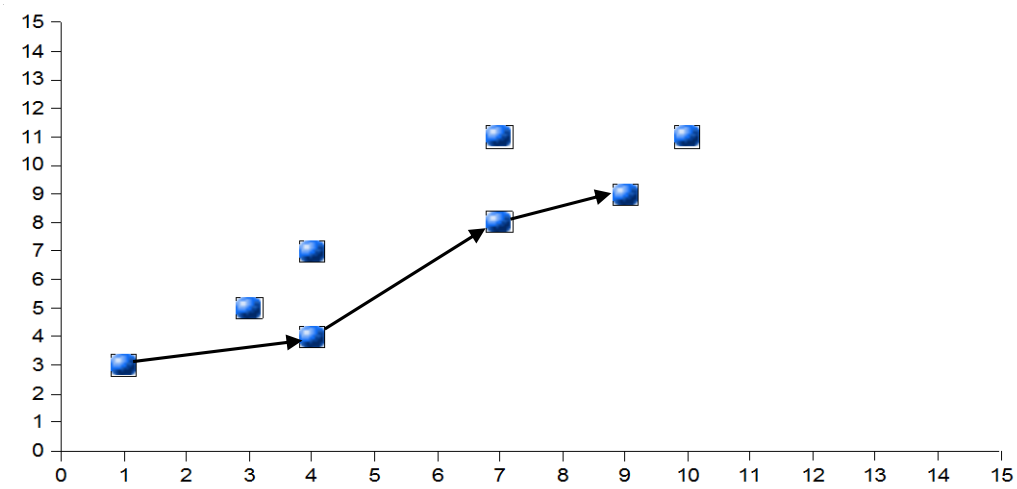
Each tag will store all wavelet coefficients in the lossless data compression mode using the standard decomposition method.

In the first test, we are finding the path between T1 (source tag) and T6 (destination tag). The distance between T1 and T6 is 10 and angle is 28.76. The algorithm calculates the next tag to move towards T6 and it is tag T3. The distance between T3 and T1 is 3.16 and the angle between T1 and T3 is 10.32. Now, source tag is T3 and the destination tag is still T6. So next nearest tag from T3 towards T6 is tag T5 which is 5 units away from tag T3 at the angle of

45.02. The next run is having T5 as the source tag. The algorithm gives tag T6 (which is a destination tag) which is 2.24 units away from tag T5 with angle of 18.45. So, to reach from tag T1 to tag T6, we have to travel a total of 10.40 units as compared to the original 10 units. (See table below).

Source Tag	Next Tag	Angle between source to Next	Angle between Source to Destination	Distance between Source to Destination	Distance to travel
T1	T3	10.32	28.76	10	3.16
T3	T5	45.02	36.89	7.07	5.00
T5	T6	18.45	18.45	2.24	2.24
				Total Distance Travelled	10.40
				Excess Distance Travelled	0.40
				Percentage Excess Travelled	4%

The path diagram of this travel is:



This algorithm is also tested with finding the path from tag T2 to tag T7 and the path from tag T4 to tag T8. The following table describes the findings and percentage of excess travel to reach destination path in all these three tests.

Experiment No	Source Tag	Destination Tag	Distance to Travel	Tags Travelled	Actual Distance Travelled	Difference	Excess Percentage
1	T1	T6	10	T1, T3, T5, T6	10.40	0.40	4%
2	T2	T7	9.22	T2, T4, T5, T6, T7	9.88	0.88	9.54%
3	T4	T8	7.21	T4, T5, T6, T8	7.63	0.42	5.83%

Table 38: Experiment Readings using 8-tags network in Scenario 2

Now, experiment 1 above is repeated in the lossy compression mode with threshold values of 0.50 and 25%. With threshold value 0.50, all wavelet coefficients are replaced with 0 if absolute value of coefficient is less than or equal to 0.50. With 25% threshold, each RFID tag will store the largest 75% of coefficients in value.

Using threshold = 0.50, the wavelet coefficient reconstruction functions returns (1,3,4,4,7,8,7,11) and (3,5,4,7,8,10,10,11) coordinates for X-Axis and Y-Axis respectively. The algorithm gives T2, T4, and T5 tags in the path of travel from source tag T2 to destination tag T7. The distance travelled in this path is 8.64 as compare to original distance of 6.40. Using a threshold of 25%, after reconstructing the wavelet coefficients obtained are (1,3,4,4,8,9,9,11) and (3,5,4,7,8,9,9,911) coordinates for X-Axis and Y-Axis respectively. So, to reach tag T7 from tag T2, the algorithm has to pass through T4, T5, and T6 tags to each the destination. Using 25% threshold, tag T6 and tag T7 end up with the same coordinates but algorithm considers the T6 tag ahead of the T7 tag because of its position in the network. In keeping with the coalescing of T6 and T7, the algorithm indicates 0 units need to be travelled between T6 and T7.

Threshold	Source Tag	Destination Tag	Distance to Travel	Tags Travelled	Actual Distance Travelled	Difference	Excess Percentage
0.50	T2	T7	6.40	T2, T4, T5,T6, T7	8.64	2.24	35%
25%	T2	T7	7.21	T2, T4, T5,T6, T7	7.77	0.56	7.77%

Table 39: Experiment Readings using 8-tags network for applying different threshold values in Scenario 2

Experiment 2: In this experiment, we use 16 tags.

In this experiment we assess the effects of lossless and lossy compression on path length. The standard decomposition method is used and the coefficients are stored in a 4 x 8 matrix as:

$$\begin{pmatrix} 11.34,-4.91,-0.82,2.5 \\ -2.72,0.16,-0.06,-0.13 \\ -1,0.13,-0.13,-1.13 \\ -1.06,0.31,0.50,0.13 \\ -0.88,-0.13,0,0.25 \\ -0.88,0.13,0.75,0 \\ -0.25,-0.50,-0.25,-0.75 \\ -1.12,0.38,-0.75,0.5 \end{pmatrix}$$

a. Threshold = 0

After reconstructing the data, we get X-Axis coordinates as (1, 3, 4, 4, 7, 9, 7, 10, 13, 14, 17, 19, 20, 21, 22, 24) and Y-Axis coordinates as (3, 5, 4, 7, 8, 9, 11, 11, 10, 12, 10 , 12, 16, 14,

16, 20). Our objective is to reach tag T15 from tag 2. The distance between these two tags is 21.95 with an angle of 21.96.

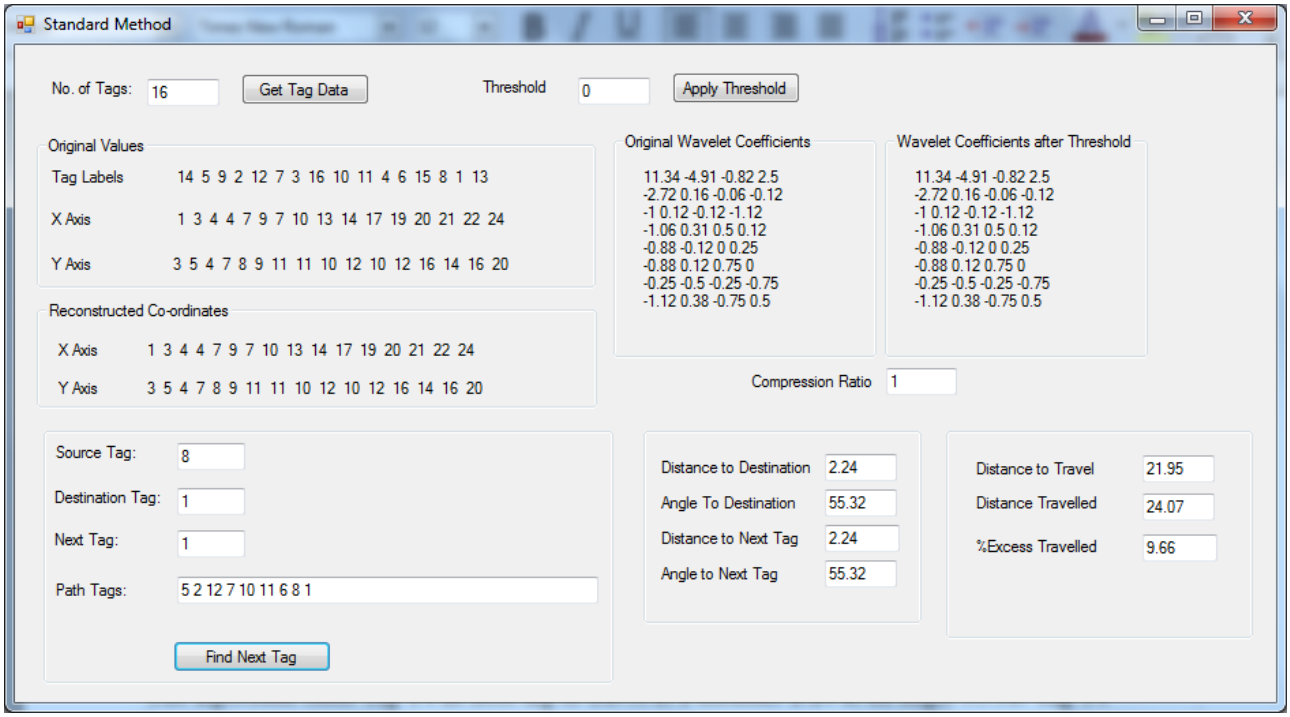


Figure 5.10 : Screenshot for applying 0 threshold in Scenario 2

As this is *Lossless* data compression, this experiment shows similar results to the *Lossless* data compression configuration in Scenario 1 above. The algorithm finds tag T4 as the next tag to travel at a distance 2.24 and at an angle of 55.58. Tag T4 then becomes the source tag and from this point, the algorithm finds that tag T5 is the next tag to move to at a distance of 3.16 and with an angle of 10.32. Tag T6 is the next tag from tag T5 at angle of 18.33 and distance 2.24 units away from tag T5. Tag T9 is the next tag to move to from tag T6 which is 4.12 units away from tag T9 with an angle of 5.73. After tag T9, tag T10 is the next tag at a distance of 2.24 units from T9 and at an angle of 55.58. From tag T10, the algorithm finds tag T12 is the next tag to move to reach the destination tag T15 and the distance to travel from tag10 to tag T12 is 5 units with an angle of 179.91. After tag T12, algorithm gives T14 tag at a distance of 2.83 units having angle of 36.67 and finally the destination tag T15 is reached from tag T14 which is at a distance of 2.24 units with angle of 55.32. The total distance travelled is 24.07 to reach from the source tag T2 to the destination tag T15 which is 2.12 units more than the shortest path between source and destination.

Source Tag	Next Tag	Angle between source to Next	Angle between Source to Destination	Distance between Source to Destination	Distance to travel
T2	T4	55.32	21.96	21.95	2.24
T4	T5	10.32	18.45	20.12	3.16
T5	T6	18.45	19.96	17	2.24
T6	T9	5.92	20.19	14.76	4.12
T9	T10	55.32	25.58	10.82	2.24
T10	T12	180	18.45	8.94	5
T12	T14	36.89	45.02	5	2.83
T14	T15	55.32	55.32	2.24	2.24
				Total Distance Travelled	24.07
				Excess Distance Travelled	2.12
				Percentage Excess Travelled	9.66%

The following figure (Figures 5.11) gives the graphical representation of the travel path from tag T2 to tag T15:

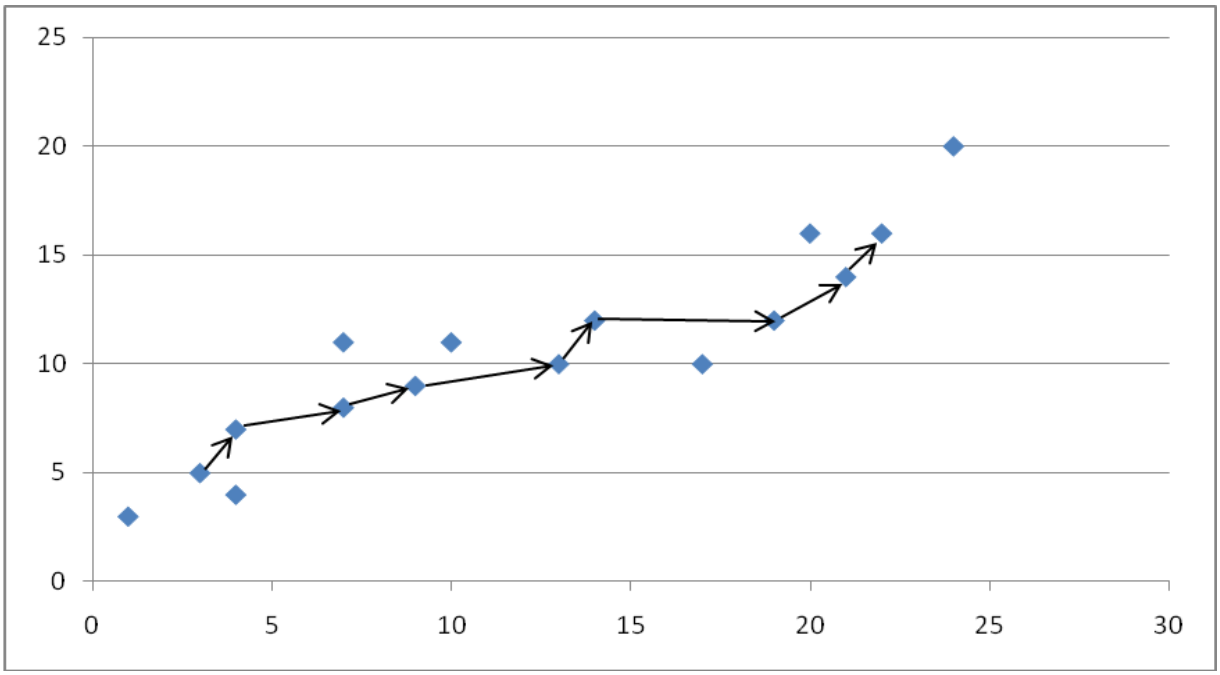


Figure 5.11 : Navigation Path for applying 0 threshold in Scenario 2

This algorithm is also tested with finding path from Tag T3 to Tag t11 and path from Tag T1 to Tag T13. Following table describes the findings and percentage excess travel to reach destination path in all these three tests :

Experiment No	Source Tag	Destination Tag	Distance to Travel	Tags Travelled	Actual Distance Travelled	Difference	Excess Percentage
1	T2	T15	21.95	T2, T4, T5, T6, T9, T10, T12, T14, T15	24.07	2.12	9.66%
2	T3	T12	17.00	T3. T4, T5, T6, T9, T10, T12	19.76	2.76	16.24%
3	T1	T13	23.02	T1, T2, T5, T6, T9, T10, T12, T13	25.55	2.53	10.99

Table 40: Experiments Readings for applying threshold = 0 in 16-tags network in Scenario 2

We now assess the effect of lossy data compression with thresholds of 0.50 and 25%.

b. Threshold = 0.50

After applying the 0.50 threshold, we get the following wavelet coefficient matrix:

$$\begin{pmatrix} 11.34,-4.91,-0.82,2.5 \\ -2.72,0,0,0 \\ -1,0,0,0 \\ -1.06,0,0,0 \\ 0,0,0,0 \\ 0,0,0.75,0 \\ 0,0,0,-0.75 \\ -1.12,0,-0.75,0 \end{pmatrix}$$

Reconstructing these *lossy* wavelet coefficients we get following coordinates for X-Axis ad Y-Axis:

X : 1,3,4,4,7,7,8,11,13,15,17,19,20,21,21,24
Y: 3,4,4,7,9,9,11,11,10,12,10,12,16,15,16,19

Using this data, the algorithm finds all tags to travel from tag T2 to tag T15. The distance between these two tags is 21.63 units with an angle of 25.58.

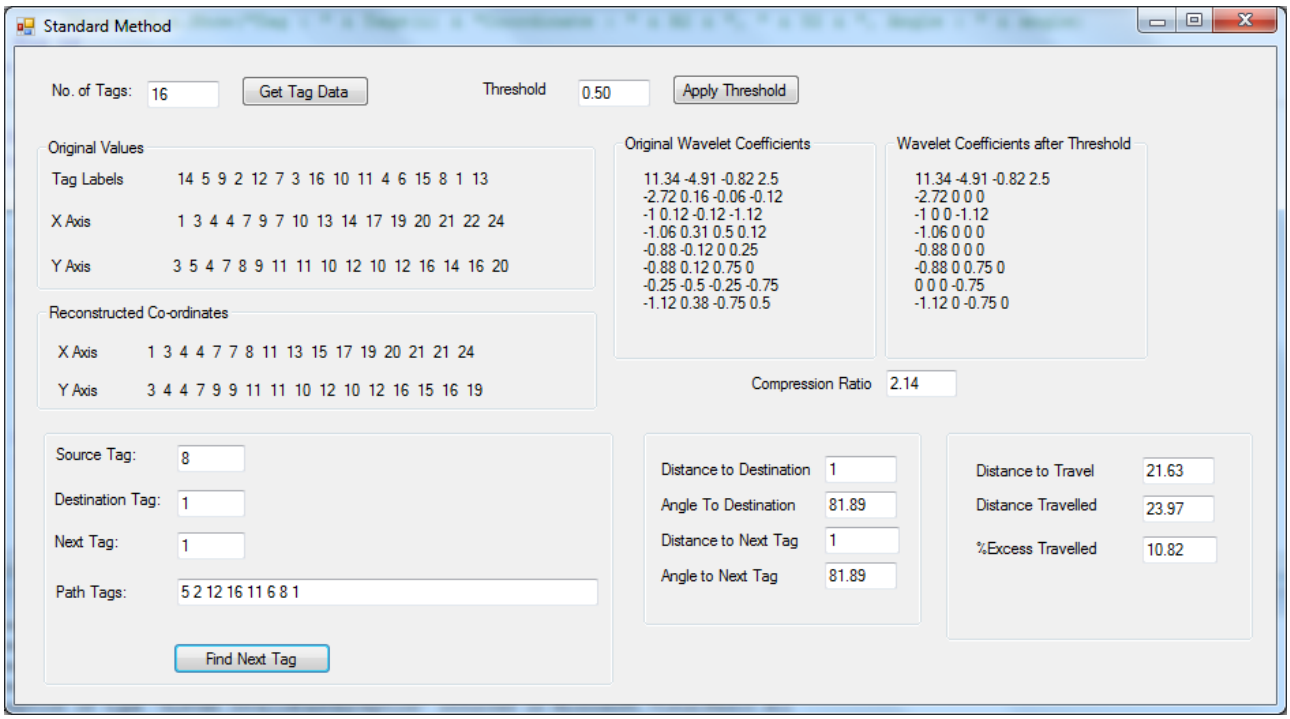


Figure 5.12 : Screenshot for applying 0.50 threshold in Scenario 2

First tag in the path is T4 at a distance of 3.16 units with an angle of 63.60. From tag T4, the next tag to travel to is tag T5 which is at a distance of 3.61 units from tag T4 with an angle of 25.78. The next in the sequence is tag T5 which is 4.47 units away and at an angle of 18.33. From tag T8, the next tag is T10 which is 4.12 units from T8 with an angle of 5.73. T12 is the next tag to be found by the algorithm with a distance of 4 units from T10 and with an angle of 179.91. Tag T14 is the final tag before reaching the destination, is 3.61 units away from T14 with an angle of 48.13. Finally, tag T15 is reached from T14 at a distance of 1 unit and angle between these two tags is 81.93. The total distance travelled from tag T2 to T15 is 23.97 which is 10.82% more than the shortest path.

Source Tag	Next Tag	Angle between source to Next	Angle between Source to Destination	Distance between Source to Destination	Distance to travel
T2	T4	63.45	25.58	21.63	3.16
T4	T5	25.58	19.78	19.24	3.61
T5	T8	18.45	18.45	15.65	4.47
T8	T10	5.92	18.45	11.18	4.12
T10	T12	180	25.58	7.21	4
T12	T14	48.20	55.32	4.47	3.61
T14	T15	81.89	81.93	1	1
				Total Distance Travelled	23.97
				Excess Distance Travelled	2.34
				Percentage Excess Travelled	10.82%

Following is the graphical representation (Figure 5.13) of the travel for tag T2 to tag T15 using the standard decomposition method in lossy mode with threshold set at 0.50.

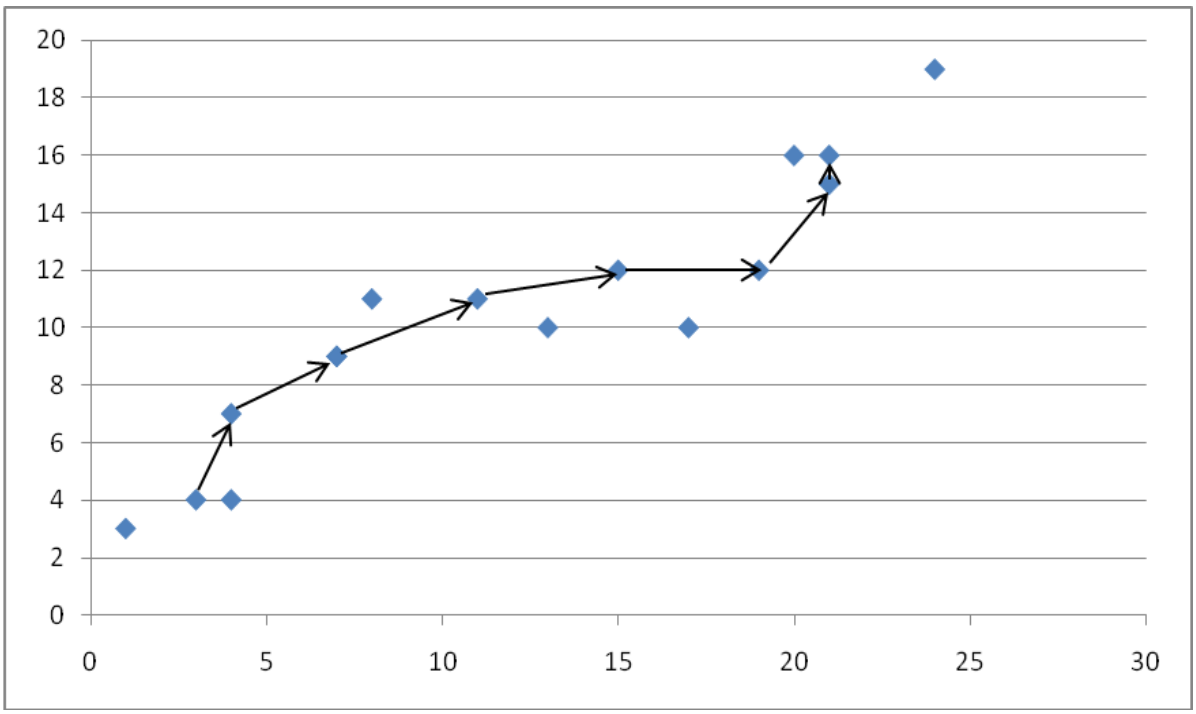


Figure 5.13 : Navigation Path for applying 0.50 threshold in Scenario 2

This algorithm is also tested with finding path from Tag T3 to Tag t11 and path from Tag T1 to Tag T13. Following table describes the findings and percentage excess travel to reach destination path in all these three tests :

Experiment No	Source Tag	Destination Tag	Distance to Travel	Tags Travelled	Actual Distance Travelled	Difference	Excess Percentage
1	T2	T15	21.63	T2, T4, T5, T8, T10, T12, T14, T15	23.97	2.34	10.82%
2	T3	T12	17.00	T3, T5, T8, T10, T12	18.42	1.42	8.35%
3	T1	T13	23.02	T1, T2, T5, T8, T10, T12, T13	25.35	2.33	10.12%

Table 41: Experiments Readings for applying threshold = 0.50 in 16-tags network in Scenario 2

c. Threshold = 25%

Now, 25% threshold is applied on the wavelet coefficients. After applying this threshold, we get the following wavelet coefficient matrix:

$$\begin{pmatrix} 11.34,-4.91,-0.82,0 \\ -2.72,0.16,-0.063,0 \\ -1,0.13,-0.13,0 \\ -1.06,0.31,0.50,0 \\ -0.88,-0.13,0,0 \\ -0.88,0.13,0.75,0 \\ -0.25,-0.50,-0.25,0 \\ -1.12,0.38,-0.75,0 \end{pmatrix}$$

The reconstructed coefficients are:

X : 1,3,4,4,7,9,7,10,11,13,13,15,18,18,19,22
Y : 3,5,4,7,8,9,11,11,11,13,13,15,18,18,19,22

Using this data, we find the distance between tag T2 and T15 as 21.16 units with an angle of 33.07.

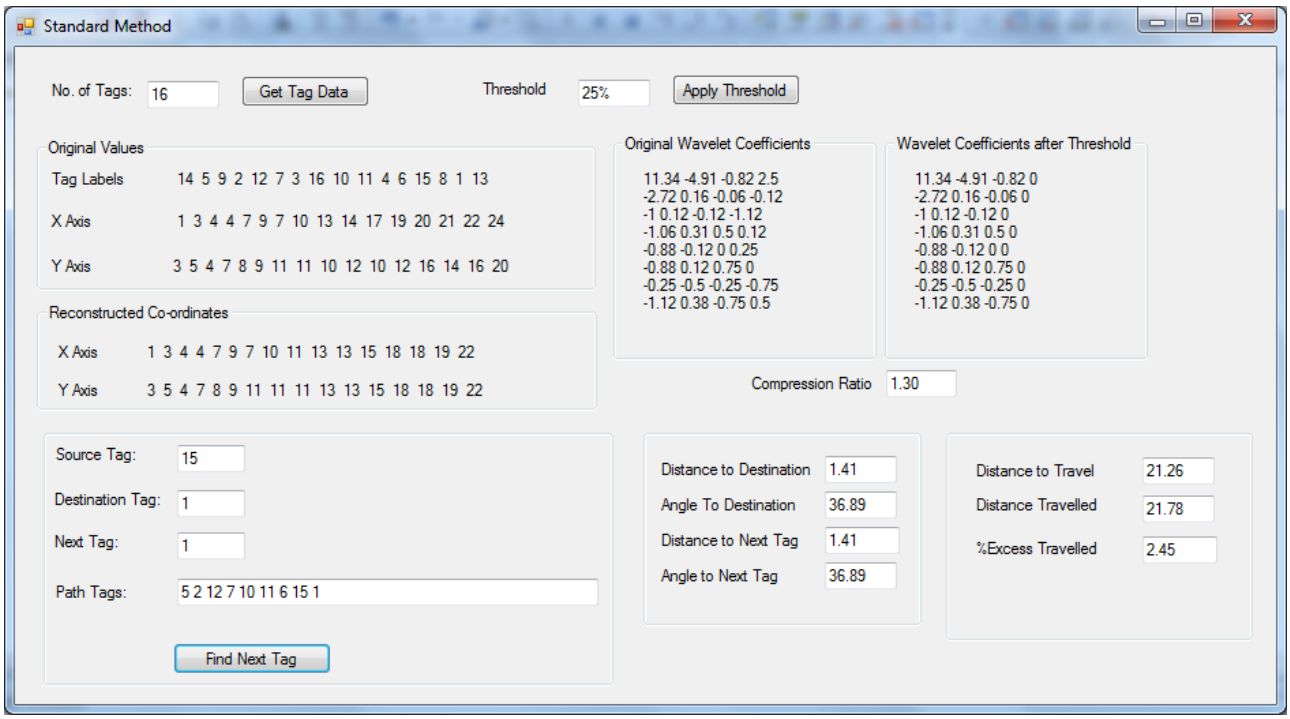


Figure 5.14 : Screenshot for applying 25% threshold in Scenario 2

Number of tags in the path from T2 to T15 are 7, i.e. hops = 7. The distance travelled between these two tags is 21.26 units which is just 0.52 units more than the shortest path, i.e. 2.45% excess which is near optimal.

Source Tag	Next Tag	Angle between source to Next	Angle between Source to Destination	Distance between Source to Destination	Distance to travel
T2	T4	55.58	33.07	21.26	2.24
T4	T5	10.31	30.55	19.21	3.16
T5	T6	18.33	34.40	16.28	2.24
T6	T9	36.67	36.89	14.14	2.83
T9	T10	36.67	36.89	11.31	2.83
T10	T12	36.67	36.89	8.49	2.83
T12	T13	36.67	36.89	5.66	4.24
T13	T15	36.89	36.89	1.41	1.41
				Total Distance Travelled	21.78
				Excess Distance Travelled	0.52
				Percentage Excess Travelled	2.45%

Following figure (Figure 5.15) is the graphical representation of the travel from tag T2 to tag T15 using the standard decomposition method operating in lossy mode with threshold = 25%.

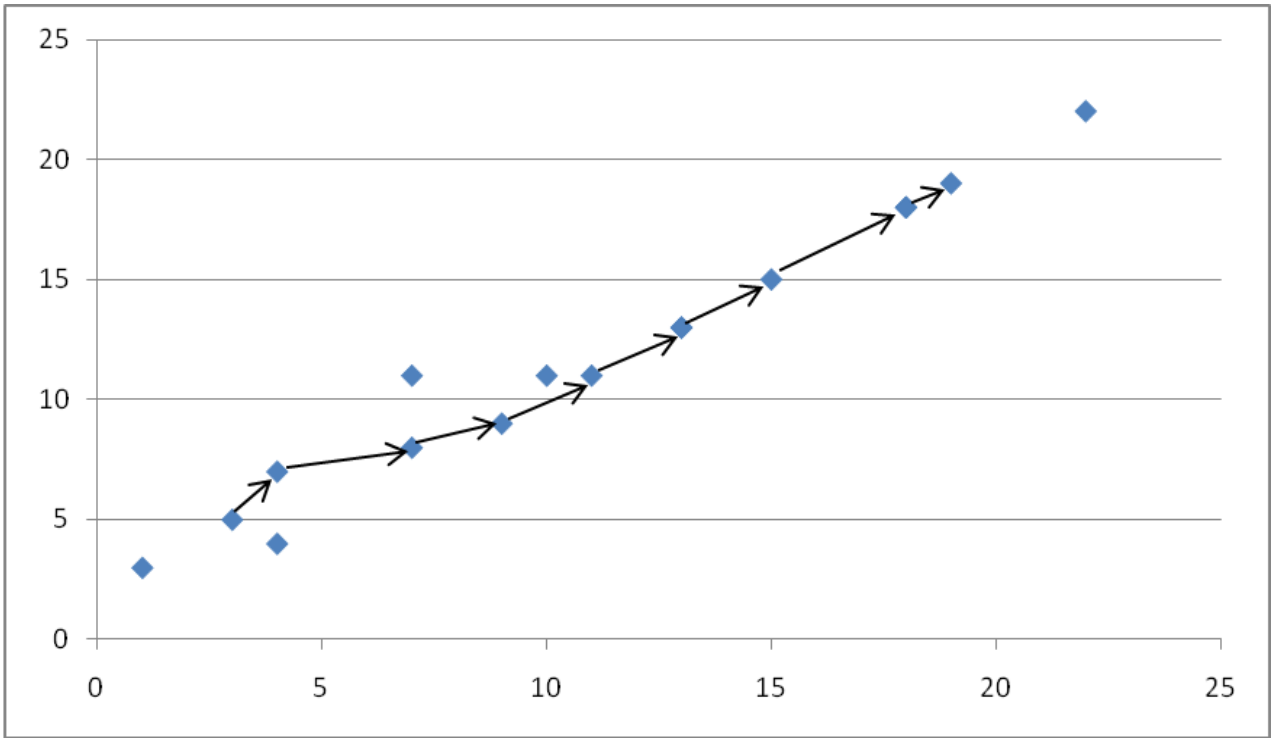


Figure 5.15 : Navigation Path for applying 25% threshold in Scenario 2

This algorithm is also tested with finding path from Tag T3 to Tag t11 and path from Tag T1 to Tag T13. Following table describes the findings and percentage excess travel to reach destination path in all these three tests :

Experiment No	Source Tag	Destination Tag	Distance to Travel	Tags Travelled	Actual Distance Travelled	Difference	Excess Percentage
1	T2	T15	21.16	T2, T4, T5, T6, T9, T10, T12, T13, T15	21.78	0.52	2.45%
2	T3	T12	15.56	T3, T4, T5, T6, T9, T10, T12	16.89	1.33	8.55%
3	T1	T13	22.67	T1, T2, T5, T6, T9, T10, T12, T13	22.80	0.13	0.57%

Table 42: Experiments Readings for applying threshold = 25% in 16-tags network in Scenario 3

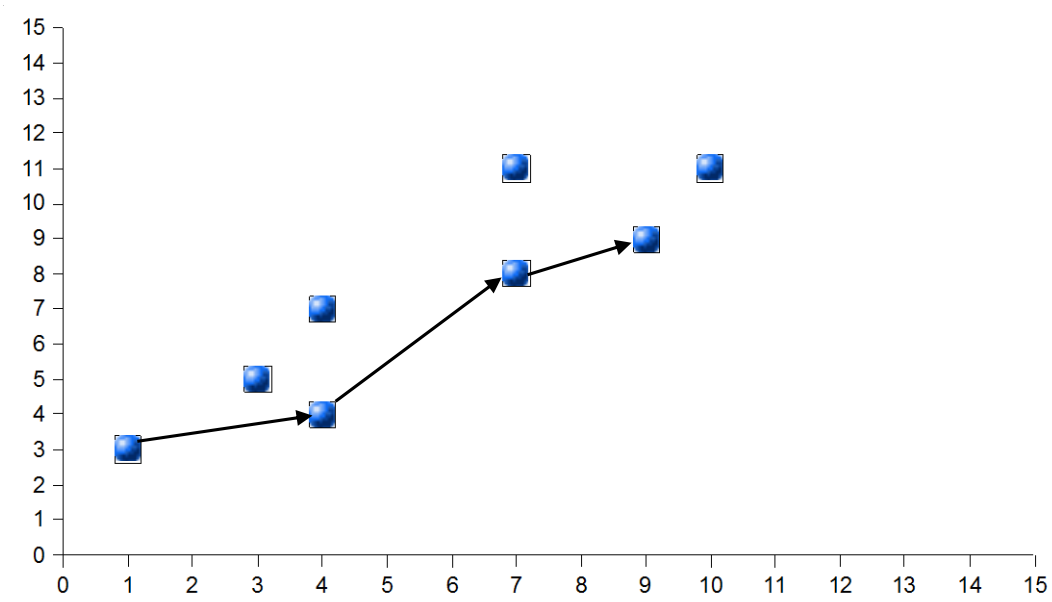
5.3.3 Scenario 3: Store Wavelet Coefficients using tag position

In this scenario, lossy data compression is applied through the use of positional information, rather than the use of a fixed size threshold. We first experiment with 8 tags.

Navigation (8-Tags Network)

In this experiment, the source tag was taken as T1 and the destination tag as T6. Tag 1 at position (1, 3) stores exact information about tag T2 at (3, 5), tag T3 at (4, 4), and tag T4 at (4, 7). Approximate information was stored on tags T5 (6, 2), T6 (7, 4), T7 (7, 5) and T8 (8, 3).

We found tag T3 (4, 4) to be the nearest tag in the path to the destination tag, i.e. tag T6 (7,4). Applying this procedure recursively, we found that tag T5 (6, 2) was the next tag in the path. So, to reach tag T6 from tag T1, the pointer went through tags T3 and T5. In this case, we have 3 hops to reach the destination tag. So, Navigation from T1 to T6 is:



As mentioned earlier, each tag stores a different set of wavelet coefficients and program reconstructs data during runtime, each time we get different values depending on source tag position. The following table contains the wavelet coefficients stored in these tags and the corresponding reconstructed values.

Tag	Wavelet Coefficients	Reconstructed values
T1	X : 5.62, -2.62, -1, -0.25, -1, 0, 0, 0 Y : 7.25, -2.50, -0.75, -1.25, -1, -1.5, 0, 0	X : 1, 3, 4, 4, 8, 8, 8, 8 Y : 3, 5, 4, 7, 8, 9, 11, 11
T3	X : 5.62, -2.62, -1, -0.25, -1, 0, 0, 0 Y : 7.25, -2.50, -0.75, -1.25, -1, -1.5, 0, 0	X : 1, 3, 4, 4, 8, 8, 8, 8 Y : 3, 5, 4, 7, 8, 9, 11, 11
T5	X : 5.62, -2.62, -1, -0.25, 0, 0, -1, -1.5 Y : 7.25, -2.50, -0.75, -1.25, 0, 0, -0.5, 0	X : 2, 2, 4, 4, 7, 9, 7, 10 Y : 4, 4, 6, 6, 8, 9, 11, 11
T6	X : 5.62, -2.62, -1, -0.25, 0, 0, -1, -1.5 Y : 7.25, -2.50, -0.75, -1.25, 0, 0, -0.5, 0	X : 2, 2, 4, 4, 7, 9, 7, 10 Y : 4, 4, 6, 6, 8, 9, 11, 11

The distance between T1 and T6 is 8.60 and the angle between them is 27.43. The navigation algorithm finds the four nearest neighbours of the source tag and then calculates the angle difference of each of the nearest neighbour tags to destination tag. The next tag to navigate to is the tag which has closest angle to the destination as shown in the following table.

Source Tag	Next Tag	Angle between source and next	Angle between Source and Destination	Distance between Source to Destination	Distance to travel
T1	T3	10.32	32.49	9.23	3.16
T3	T5	36.89	36.89	5.66	5.66
T5	T6	18.45	18.45	2.24	2.24
				Total Distance Travelled	11.06
				Excess Distance Travelled	1.83
				Percentage Excess Travelled	19.83%

Three experiments are done using the 8-Tags network. The table below describes the path and distance travelled for each experiment in moving from the source to the destination tag.

Experiment No	Source Tag	Destination Tag	Distance to Travel	Tags Travelled	Actual Distance Travelled	Difference	Excess Percentage
1	T1	T6	9.23	T1, T3, T5, T6	11.06	1.83	19.83%
2	T2	T7	7.81	T2, T4, T5, T6, T7	11.43	3.62	46.35%
3	T4	T8	5.66	T4, T5, T6, T8	8.60	2.94	51.94%

Table 43: Experiments Readings in 8-tags network in Scenario 3

The same data is used in the next set of experiments with a 16-tags network. In the 16-Tags network, each group stores exact information on its member tags, less precise information about tags of its neighbouring group and even less precision information on the remaining two groups. With respect to Group1 (the reference, or baseline group), it stores exact information for

tags T1 to T4, stores fuzzier information about tags T5 to T8 and the fuzziest information about tags T9 to T16.

Group 2 stores exact information about tags T5 to T8, fuzzier information about tags T1 to T4 and the fuzziest information about tags T9 to T16. Group3 stores exact information about tags T9 to T12, fuzzier information about tags T13 to T16 and the fuzziest information on tags T1 to T8. In case of Group 4, it stores exact information about tags T13 to T16, fuzzier information on tags T9 to T12 and the fuzziest information about tags T1 to T8.

The following table displays the wavelet coefficients stored in these tags and the corresponding reconstructed values. The bold font denotes those wavelet coefficients that had exact reconstructions.

Tag	Wavelet Coefficients Stored	Reconstructed Coordinate Values
01	X: 12.19, -6.56, -2.62, -3, -1, 0.25, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0 Y: 10.5, -3.25, -2.50, -2.75, -0.75, -1.25, 0, 0, -1, -1.5, 0, 0, 0, 0, 0, 0	X: **1, 3, 4, 4**, 8, 8, 8, 8, 16, 16, 16, 16, 22, 22, 22, 22 Y: **3, 5, 4, 7**, 8, 8, 11, 11, 11, 11, 11, 11, 16, 16, 16, 16
02	X: 12.19, -6.56, -2.62, -3, -1, 0.25, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0 Y: 10.5, -3.25, -2.50, -2.75, -0.75, -1.25, 0, 0, -1, -1.5, 0, 0, 0, 0, 0, 0	X: **1, 3, 4, 4**, 8, 8, 8, 8, 16, 16, 16, 16, 22, 22, 22, 22 Y: **3, 5, 4, 7**, 8, 8, 11, 11, 11, 11, 11, 11, 16, 16, 16, 16
03	X: 12.19, -6.56, -2.62, -3, -1, 0.25, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0 Y: 10.5, -3.25, -2.50, -2.75, -0.75, -1.25, 0, 0, -1, -1.5, 0, 0, 0, 0, 0, 0	X: **1, 3, 4, 4**, 8, 8, 8, 8, 16, 16, 16, 16, 22, 22, 22, 22 Y: **3, 5, 4, 7**, 8, 8, 11, 11, 11, 11, 11, 11, 16, 16, 16, 16
04	X: 12.19, -6.56, -2.62, -3, -1, 0.25, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0 Y: 10.5, -3.25, -2.50, -2.75, -0.75, -1.25, 0, 0, -1, -1.5, 0, 0, 0, 0, 0, 0	X: **1, 3, 4, 4**, 8, 8, 8, 8, 16, 16, 16, 16, 22, 22, 22, 22 Y: **3, 5, 4, 7**, 8, 8, 11, 11, 11, 11, 11, 11, 16, 16, 16, 16
05	X: 12.19, -6.56, -2.62, -3, -1, -0.25, 0, 0, 0, 0, -1, -15, 0, 0, 0, 0 Y: 10.5, -3.25, -2.50, -2.75, 0.75, -1.25, 0, 0, 0, 0, -0.5, 0, 0, 0, 0, 0	X: 2, 2, 4, 4, **7, 9, 7, 10**, 16, 16, 16, 16, 22, 22, 22, 22 Y: 4, 4, 6, 6, **8, 9, 11, 11**, 11, 11, 11, 11, 16, 16, 16, 16
06	X: 12.19, -6.56, -2.62, -3, -1, -0.25, 0, 0, 0, 0, -1, -15, 0, 0, 0, 0 Y: 10.5, -3.25, -2.50, -2.75, 0.75, -1.25, 0, 0, 0, 0, -0.5, 0, 0, 0, 0, 0	X: 2, 2, 4, 4, **7, 9, 7, 10**, 16, 16, 16, 16, 22, 22, 22, 22 Y: 4, 4, 6, 6, **8, 9, 11, 11**, 11, 11, 11, 11, 16, 16, 16, 16
07	X: 12.19, -6.56, -2.62, -3, -1, -0.25, 0, 0, 0, 0, -1, -15, 0, 0, 0, 0 Y: 10.5, -3.25, -2.50, -2.75, 0.75, -1.25, 0, 0, 0, 0, -0.5, 0, 0, 0, 0, 0	X: 2, 2, 4, 4, **7, 9, 7, 10**, 16, 16, 16, 16, 22, 22, 22, 22 Y: 4, 4, 6, 6, **8, 9, 11, 11**, 11, 11, 11, 11, 16, 16, 16, 16
08	X: 12.19, -6.56, -2.62, -3, -1, -0.25, 0, 0, 0, 0, -1, -15, 0, 0, 0, 0 Y: 10.5, -3.25, -2.50, -2.75, 0.75, -1.25, 0, 0, 0, 0, -0.5, 0, 0, 0, 0, 0	X: 2, 2, 4, 4, **7, 9, 7, 10**, 16, 16, 16, 16, 22, 22, 22, 22 Y: 4, 4, 6, 6, **8, 9, 11, 11**, 11, 11, 11, 11, 16, 16, 16, 16
09	X: 12.19, -6.56, -2.62, -3, 0, 0, -2.25, -1.25, 0, 0, 0, 0, -0.5, -1, 0, 0 Y: 10.5, -3.25, -2.50, -2.75, 0, 0, 0, -1.5, 0, 0, 0, 0, -1, -1, 0, 0	X: 3, 3, 3, 3, 8 8, 8, 8, **13, 14, 17, 19**, 20, 20, 23, 23 Y: 5, 5, 5, 5, 10, 10, 10, 10, **10. 12, 10, 12**, 15, 15, 18, 18
10	X: 12.19, -6.56, -2.62, -3, 0, 0, -2.25, -1.25, 0, 0, 0, 0, -0.5, -1, 0, 0 Y: 10.5, -3.25, -2.50, -2.75, 0, 0, 0, -1.5, 0, 0, 0, 0, -1, -1, 0, 0	X: 3, 3, 3, 3, 8 8, 8, 8, **13, 14, 17, 19**, 20, 20, 23, 23 Y: 5, 5, 5, 5, 10, 10, 10, 10, **10. 12, 10, 12**, 15, 15, 18, 18
11	X: 12.19, -6.56, -2.62, -3, 0, 0, -2.25, -1.25, 0, 0, 0, 0, -0.5, -1, 0, 0 Y: 10.5, -3.25, -2.50, -2.75, 0, 0, 0, -1.5, 0, 0, 0, 0, -1, -1, 0, 0	X: 3, 3, 3, 3, 8 8, 8, 8, **13, 14, 17, 19**, 20, 20, 23, 23 Y: 5, 5, 5, 5, 10, 10, 10, 10, **10. 12, 10, 12**, 15, 15, 18, 18
12	X: 12.19, -6.56, -2.62, -3, 0, 0, -2.25, -1.25, 0, 0, 0, 0, -0.5, -1, 0, 0 Y: 10.5, -3.25, -2.50, -2.75, 0, 0, 0, -1.5, 0, 0, 0, 0, -1, -1, 0, 0	X: 3, 3, 3, 3, 8 8, 8, 8, **13, 14, 17, 19**, 20, 20, 23, 23 Y: 5, 5, 5, 5, 10, 10, 10, 10, **10. 12, 10, 12**, 15, 15, 18, 18
13	X: 12.19, -6.56, -2.62, -3, 0, 0, -2.25, -1.25, 0, 0, 0, 0, 0, 0, -0.5, -1 Y: 10.5, -3.25, -2.50, 0, 0, 0, -1.5, 0, 0, 0, 0, 0, 0, 1, -2	X: 3, 3, 3, 3, 8, 8, 8, 8, 14, 14, 18, 18, **20, 21, 22, 24** Y: 5, 5, 5, 5, 10, 10, 10, 10, 11, 11, 11, 11, **16, 14, 16, 20**
14	X: 12.19, -6.56, -2.62, -3, 0, 0, -2.25, -1.25, 0, 0, 0, 0, 0, 0, -0.5, -1 Y: 10.5, -3.25, -2.50, 0, 0, 0, -1.5, 0, 0, 0, 0, 0, 0, 1, -2	X: 3, 3, 3, 3, 8, 8, 8, 8, 14, 14, 18, 18, **20**, **21, 22, 24** Y: 5, 5, 5, 5, 10, 10, 10, 10, 11, 11, 11, 11, **16, 14, 16, 20**
15	X: 12.19, -6.56, -2.62, -3, 0, 0, -2.25, -1.25, 0, 0, 0, 0, 0, 0, -0.5, -1 Y: 10.5, -3.25, -2.50, 0, 0, 0, -1.5, 0, 0, 0, 0, 0, 0, 1, -2	X: 3, 3, 3, 3, 8, 8, 8, 8, 14, 14, 18, 18, **20, 21, 22, 24** Y: 5, 5, 5, 5, 10, 10, 10, 10, 11, 11, 11, 11, **16, 14, 16, 20**
16	X: 12.19, -6.56, -2.62, -3, 0, 0, -2.25, -1.25, 0, 0, 0, 0, 0, 0, -0.5, -1 Y: 10.5, -3.25, -2.50, 0, 0, 0, -1.5, 0, 0, 0, 0, 0, 0, 1, -2	X: 3, 3, 3, 3, 8, 8, 8, 8, 14, 14, 18, 18, **20, 21, 22, 24** Y: 5, 5, 5, 5, 10, 10, 10, 10, 11, 11, 11, 11, **16, 14, 16, 20**

As we can see from the Table above the reconstructed values for the neighboring group are without error while groups there are further away incur errors that increase with the distance from the baseline (reference group). For example, Y coordinates for Group 2 are actually 8, 8, 11

and 11 instead of 8, 9, 11 and 11, which are their reconstructions. These tags have fuzziest information for Groups 3 and 4 and so we get 16, 16, 16, 16, 22, 22, 22, 22 and 11, 11, 11, 11, 16, 16, 16, 16 instead of 13, 14, 17, 19, 20, 21, 22, 24 and 10, 12, 10, 12, 16, 14, 16, 20. If we carefully observe these values, we can notice that for Group 2, the first two members have same data and the next two members have same data whilst in Groups 3 and 4, each member has exactly the same coordinates as all other tags in the group. This is because tag T1 has the fuzziest information about tags in Group 4.

In this experiment, T2 is the source tag and T15 is the destination tag.

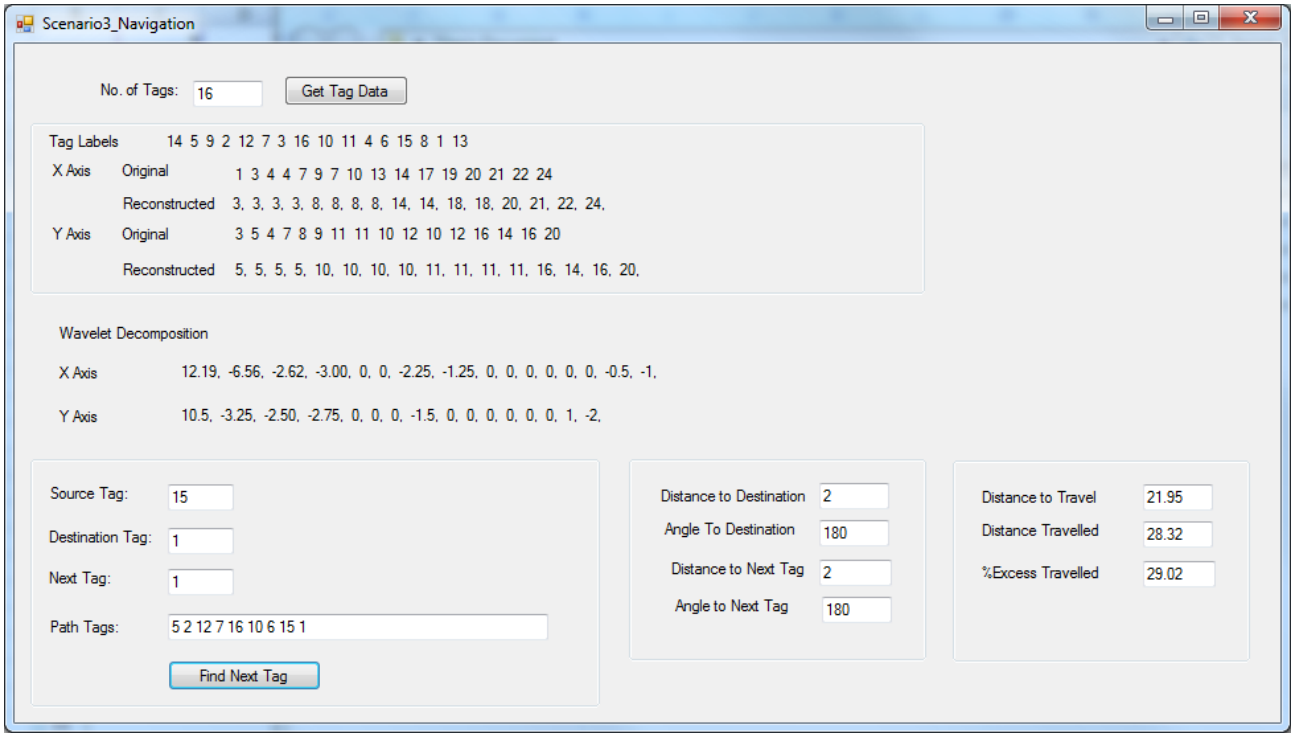


Figure 5.16: Navigation Path in Scenario 3

For the first run of the algorithm, T5 is indicated as the next tag to travel in the path. Subsequent executions give T6, T8, T9, T12 and T13 as the tags to move next. So, to reach T13 from T4 tag, it has to travel through T5, T6, T8, T9, and T12. The number of hops in the path is 6, T4-->T5-->T6-->T8-->T9-->T12-->T13. The details about angle and distance between the tags appear in the following table.

Source Tag	Next Tag	Angle between source to Next	Angle between Source to Destination	Distance between Source to Destination	Distance to travel
T2	T3	55.32	21.96	21.95	2.24
T3	T5	5.92	18.45	20.12	4.12
T5	T6	18.45	19.96	17	2.24
T6	T8	55.32	20.19	14.76	2.24
T8	T9	180	14.51	13	6
T9	T12	10.32	30.55	12.81	6.32
T12	T13	63.45	48.20	7.21	3.16
T13	T15	180	180	2	2
				Total Distance Travelled	28.32
				Excess Distance Travelled	6.36
				Percentage Excess Travelled	28.96%

The graphical representation of the above travel path is shown below in Figure 5.17 :

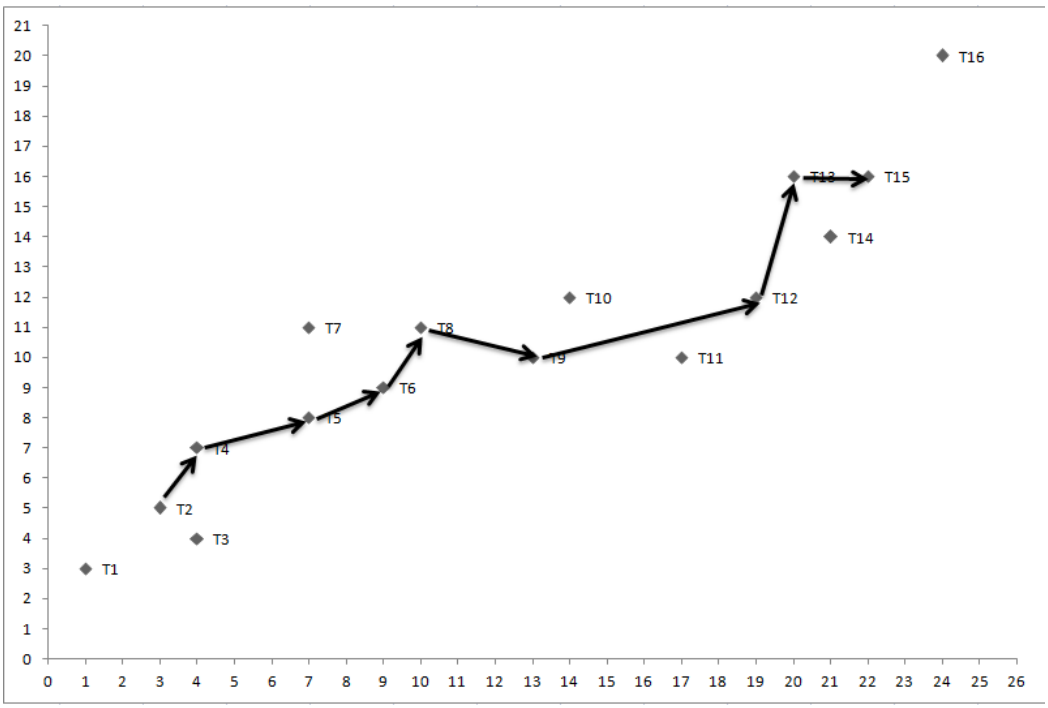


Figure 5.17: Screenshot for applying 0 threshold in Scenario 4

The same experiment is done on different paths, one is travelling from tag T4 to tag T13 and then from tag T6 to tag 16. Following table describes the summary of travelled path for these experiments is :

Experiment No	Source Tag	Destination Tag	Distance to Travel	Tags Travelled	Actual Distance Travelled	Difference	Excess Percentage
1	T4	T13	20.12	T4, T5, T6, T8, T9, T12, T13	23.76	3.64	18.09%
2	T2	T15	21.95	T2, T3, T5, T6, T8, T9, T12, T13, T15	28.32	6.36	28.96%
3	T6	T16	14.76	T6, T8, T9, T12, T14, T15, T16	21.55	6.79	46%

Table 44: Experiments Readings in 16-tags network in Scenario 3

5.3.4 Scenario 4: Non-Standard Wavelet Decomposition Method

16 tags are used to test Non-Standard Wavelet Decomposition method with 0, 0.50 and 25% threshold values.

i. Threshold = 0

In this experiment, source Tag was taken as T2 and destination tag as T15. The distance between the source and destination is 21.95 units and angle between then is 21.77.

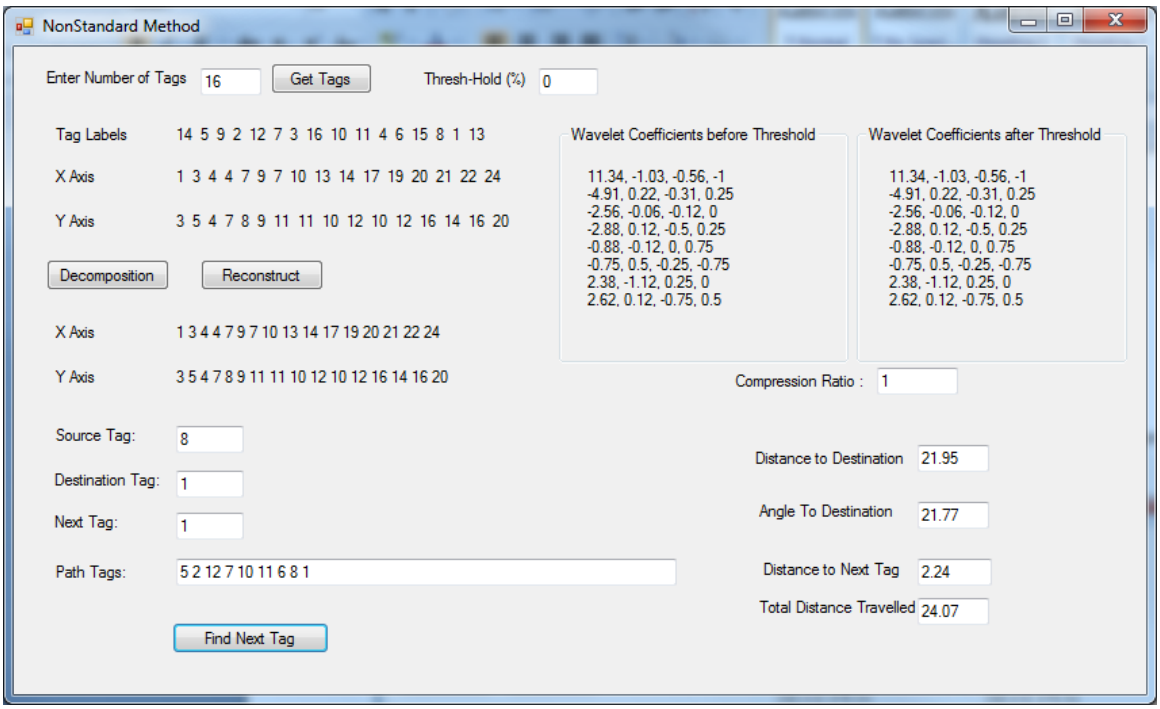


Figure 5.18 : Screenshot for applying 0 threshold in Scenario 4

The next tag in the path is tag T4 which is 2.24 units away from tag T2. Now, source tag is T4 and we again search for next tag to move. The algorithm gives tag T5 as the next tag to move towards the destination at a distance of 3.16 units from the current source tag T4. Tag T6 is the next tag to move to next from the current source tag T5 which is at a distance of 2.24 units.

Now, when source tag is T6, the algorithm indicates T9 as the next tag, which is 4.21 units away from tag T6. From tag T9, we get next tag as tag T10 with a distance of 2.24 units from tag T9. Tag T12 is given is then indicated as the next from the current source tag T9 and is at a distance of 5 units away. This then leads to tag T4, at a distance of 2.83 units from the current source tag. Finally, the destination tag T15 is reached from the source tag T14 at a distance of 2.24 units. In total the number of hops required is 8 to reach tag T15 from T2. The total distance travelled is 24.07 units which is 2.12 units more (9.66% excess) than the shortest path from source to destination, thus only incurring a small overhead in path length due to compression. The travel path is shown in the following figure (Figure 5.19) :

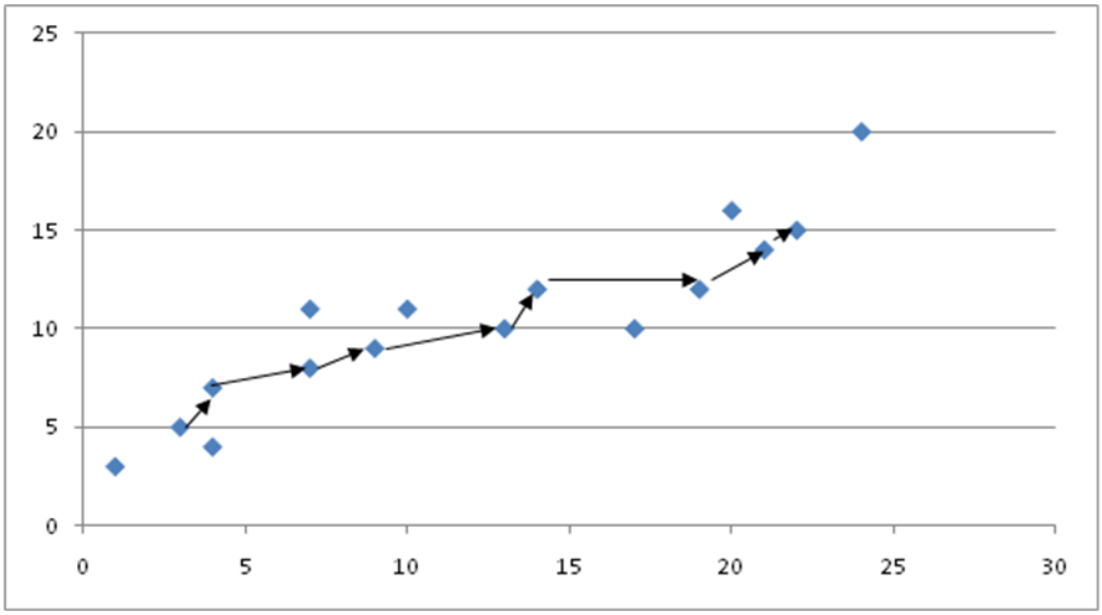


Figure 5.19 : Navigation Path for applying 0 threshold in Scenario 4

This algorithm is also tested with finding path from Tag T3 to Tag t11 and path from Tag T1 to Tag T13. Following table describes the findings and percentage excess travel to reach destination path in all these three tests :

Experiment No	Source Tag	Destination Tag	Distance to Travel	Tags Travelled	Actual Distance Travelled	Difference	Excess Percentage
1	T2	T15	21.95	T2, T4, T5, T6, T9, T10, T12, T14, T15	24.07	2.12	9.66%
2	T3	T12	17.00	T3, T4, T5, T6, T9, T10, T12	19.76	2.76	16.24%
3	T1	T13	23.02	T1, T2, T5, T6, T9, T10, T12, T13	25.55	2.53	10.19%

Table 45: Experiments Readings for applying threshold = 0 in 16-tags network in Scenario 4

ii. Threshold = 0.50

Here, the same data has been used at threshold level of 0.50 in order to see the effect of more compression on navigation performance. We preserve the source tag as T2 and the destination Tag as T15 in order to maintain the integrity of the comparison. The distance between the source tag T2 and destination tag T15 is 23.02 units and the angle between them is 26.36.

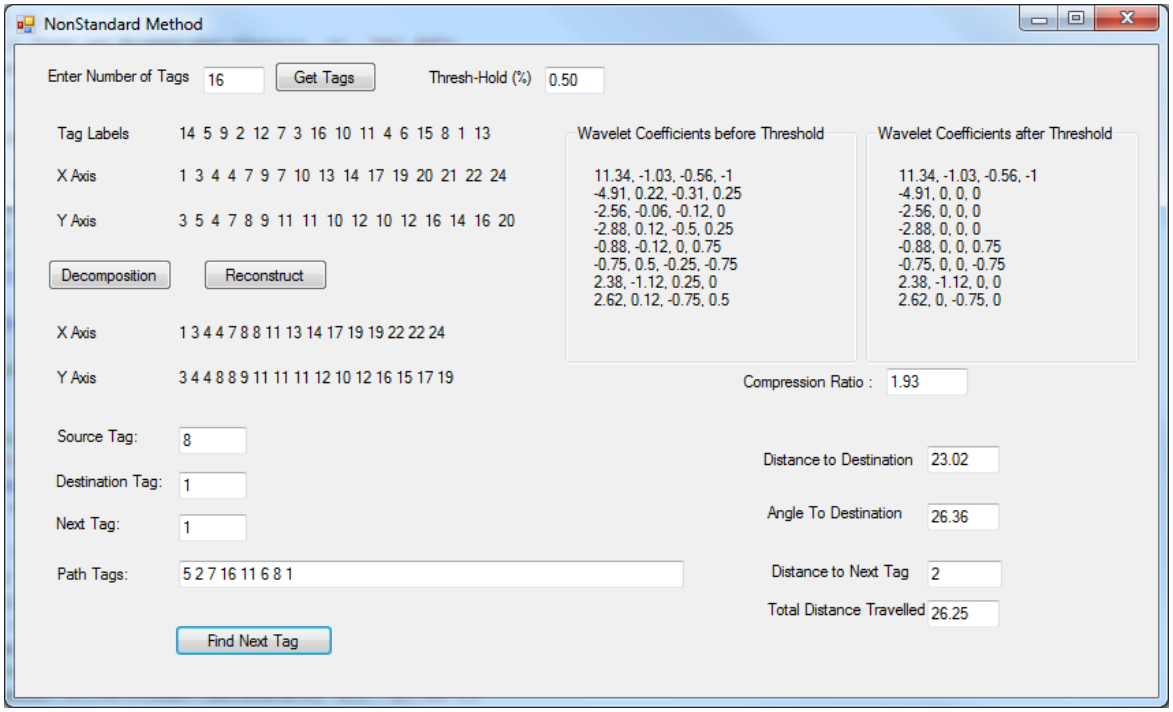


Figure 5.20 : Screenshot for applying 0.50 threshold in Scenario 4

The path travelled in this case is different from the path taken with no thresholding (Figure 5.21). The path sequence is T2, T4, T6, T8, T10, T12, T14, T15 which requires a total of 7 hops to reach the destination as compared to 8 hops with no thresholding. However, the total distance travelled was 26.25 units, which represents an excess of 14.03% as opposed to a lower excess of 9.66% with no thresholding.

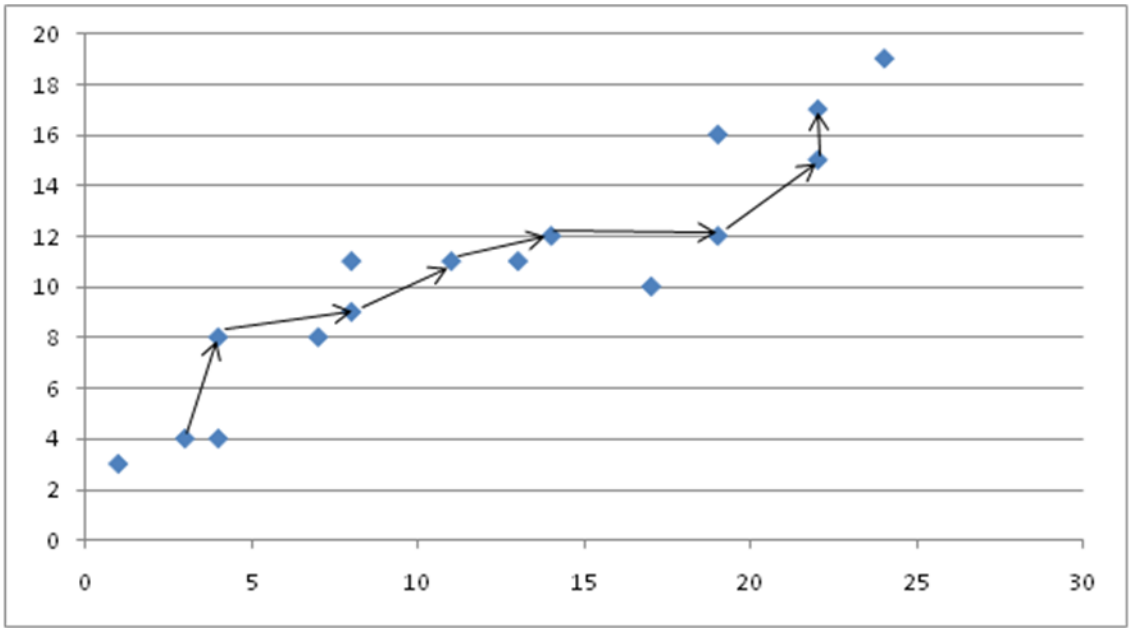


Figure 5.21 : Navigation Path for applying 0.50 threshold in Scenario 4

This algorithm is also tested with finding path from Tag T3 to Tag t11 and path from Tag T1 to Tag T13. Following table describes the findings and percentage excess travel to reach destination path in all these three tests :

Experiment No	Source Tag	Destination Tag	Distance to Travel	Tags Travelled	Actual Distance Travelled	Difference	Excess Percentage
1	T2	T15	23.02	T2, T4, T6, T8, T10, T12, T14, T15	26.25	3.23	14.03%
2	T3	T12	17.00	T3, T5, T8, T10, T12	18.16	1.16	6.82%
3	T1	T13	22.02	T1, T2, T5, T6, T8, T10, T12, T13	25.08	3.06	13.90%

Table 46: Experiments Readings for applying threshold = 0.50 in 16-tags network in Scenario 4

iii. Threshold = 25%

Same data has been used again with the same source and destination tags but with the threshold set to 25%.

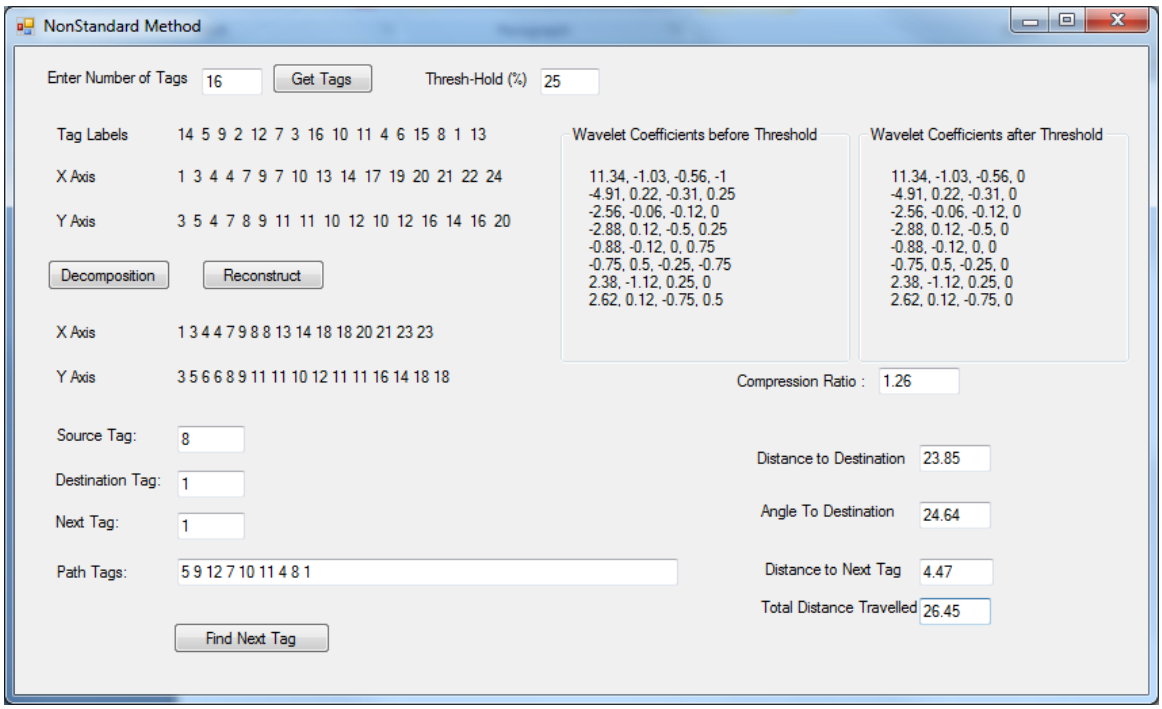


Figure 5.22 : Screenshot for applying 25% threshold in Scenario 4

The path sequence in this case is T2, T3, T5, T6, T9, T10, T11, T14 and T15. The total number of hops in this case is 8 which is 1 more than with thresholding set at 0.5. The total distance travelled is lesser at 26.45 units, thus incurring an excess of 10.9% as opposed to an excess of 14.03% with thresholding at the 25% level. The Navigation path for this experiment is shown in following figure (Figure 5.23):

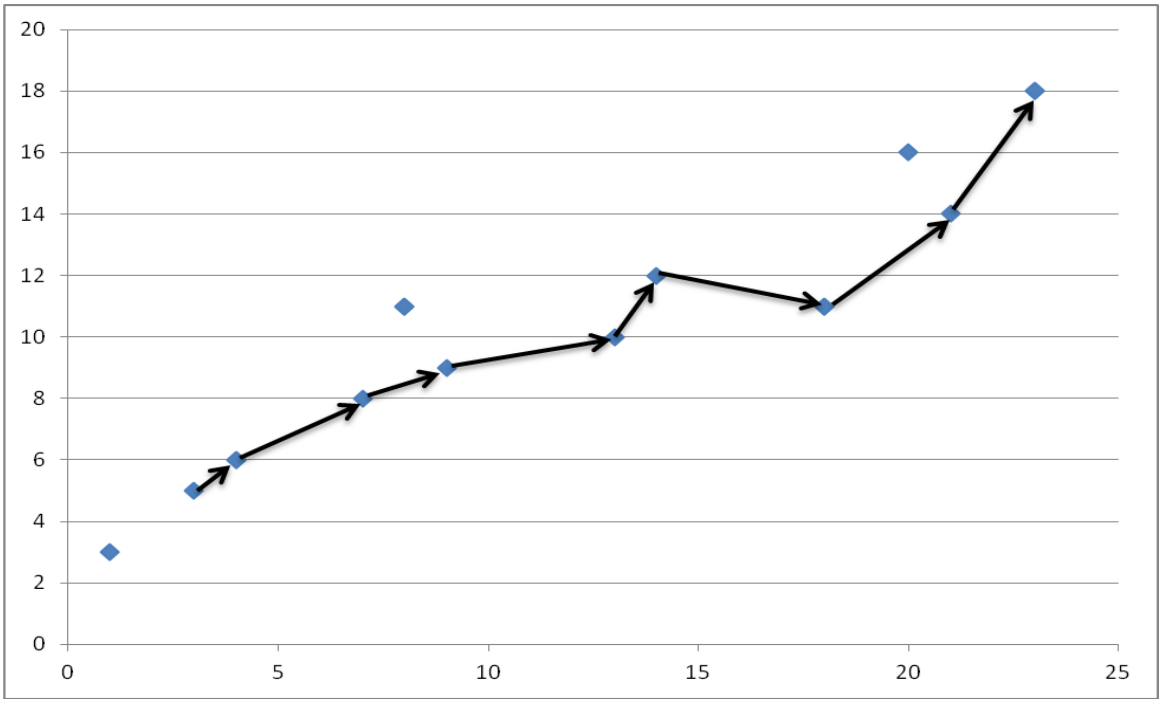


Figure 5.23 : Navigation Path for applying 25% threshold in Scenario 4

This algorithm is also tested with finding path from Tag T3 to Tag t11 and path from Tag T1 to Tag T13. Following table describes the findings and percentage excess travel to reach destination path in all these three tests :

Experiment No	Source Tag	Destination Tag	Distance to Travel	Tags Travelled	Actual Distance Travelled	Difference	Excess Percentage
1	T2	T15	23.85	T2, T3, T5, T6, T9, T10, T11, T14, T15	26.45	2.60	10.90%
2	T3	T12	14.87	T3, T5, T6, T9, T10, T11, T12	16.33	1.46	9.82%
3	T1	T13	23.02	T1, T2, T5, T6, T9, T10, T11, T13	24.67	1.65	7.17%

Table 47: Experiments Readings for applying threshold = 25% in 16-tags network in Scenario4

Chapter 6: Experimental Analysis

This chapter comprises of two sections. The first section will discuss the findings of compression experiments done on the 8 tag and 16 tag environments from the twin perspectives of compression efficiency and navigation performance. Based on the analysis conducted in the first section, the second section will identify the best compression scheme to be used in a real-world scenario.

6.1 Effects of Different Compression Schemes

Different compression algorithms are used in the four different experimental scenarios. After compressing the original data, threshold values have been applied on the wavelet coefficients to get a final compressed version of the wavelet coefficients. The following table displays the compression ratios derived from each compression scheme for the 16 tag network configuration.

Scenario	**Threshold Value**	**Compression Ratio**
Scenario 1 : All Wavelet Coefficients are stored	0	1.00
	0.50	1.25, 1.08 Avg 1.17
	25%	1.36, 1.40 Avg 1.38
Scenario 2 : Standard Decomposition Method	0	1.00
	0.50	2.14
	25%	1.30
Scenario 3 : Coefficients according to tag resolution	--	2.00
Scenario 4 : Non-Standard Decomposition Method	0	1.00
	0.50	1.93
	25%	1.26

From the above table, it can be observed that the 0.50 threshold compressed data was very effective in all four scenarios, though the 25% threshold gave the best result in Scenario 1. A threshold level of 1 was applied in Scenario 1 but gave a high excess in percentage terms for the distance travelled and was therefore abandoned. While testing different threshold values on wavelet coefficients, it was observed that the compression ratio is dependent on the distribution

of wavelet coefficients values. In some cases, the 25% threshold yielded better results than thresholding at the absolute level of 0.5. As Talukder et al have observed (Talukder, K. H., Harada, K., 2007) the setting of the threshold needs to be carried out empirically on a trial and error basis as compression ratios obtained are very dependent on the nature of the underlying data distribution.

In Scenario 1, which assumes that all wavelet coefficients can be stored in a single RFID tag, three threshold values were applied on the array of wavelet coefficients. Threshold = 1 gave the worst results and had to be abandoned, as mentioned before (see Tables 10 and 11). Overall, with Scenario 1, the lowest reconstruction error is 8%, while the highest is 11%. When 25% thresholding was applied in Scenario 1, the last half of the coefficients was affected. This is due to the fact that 25% thresholding results in the last 25% of the coefficients being dropped, which in turn only affects the reconstruction of the last half of the coefficient array (i.e. the last set of tags in the array (see Tables 15 and 16).

Scenario 2 experiments also gave very similar results. The standard decomposition method is used for compression in this scenario. A multi-dimensional array was used to stored wavelet coefficients and reconstructed values. But here the number of values that changed with Threshold = 0.50 are more than that in Scenario 1 (i.e. storing all coefficients). Thresholding at the 25% level also gave poorer results as compared to Scenario 1. The maximum %Err is 33% here as compared to 10% in Scenario 1. Compression ratio was better for 16 tags when the 0.50 threshold value was applied. Compression ratio obtained was 2.14 with threshold 0.50 as opposed to 1.30 for the 25% thresholding level.

As Scenario 3 is inherently lossy in nature, no thresholding applied in this scenario. The number of coefficients stored in each tag depends on how many tags exist in the environment. Here 8 tags and 16 tags environments are tested. In the 8-tag environment, 6 wavelet coefficients are stored while in the 16-tag environment, 8 wavelet coefficients are stored for tag labels and (X-Axis, Y-Axis) coordinates. The greater the number of tags in the environment the better is the compression ratio. Compression ratio is 2 for 16 tags as compared to 1.33 for 8 tags (Table 24).

After experimenting on different N number of tags, following formula has been derived t calculate number of wavelet coefficients to be stored in each tag using Scenario 3 :

Number of Tags (N) = 2^S

Here, for 8 tags we store 6 wavelet coefficients and for 16 tags, 8 wavelet coefficients are stored in each tag, i.e.

$8 = 2^3$ i.e. 3 x 2 wavelet coefficients
$16 = 2^4$ i.e. 4 x 2 wavelet coefficients

So, Number of Wavelet Coefficients in a tag = 2×S. Using this formula, we can easily calculate number of wavelet coefficients to be store for any N number of tags and the resulting compression ratio. The following table illustrates the relationship between the size of the network (i.e. number of tags) and compression ratio.

Number of Tags	Number of Wavelet Coefficients in a tag	Compression Ratio
8	6	1.33
16	8	2.00
32	10	3.20
64	12	5.33

From the above table, it can be seen that when the number of tags increases in this scenario, the compression ratio also increases. Thus, it is clear that better efficiencies can be achieved in large RFID tag environments.

Non-standard decomposition method has been applied to compress the data in Scenario 4. After applying threshold = 0.50 on wavelet coefficients, we get %Err values ranging from 5% to 20% while the corresponding range with 25% thresholding was 4% to 50. At the same time, the compression ratio with the 0.50 threshold at 1.93 was better than the compression ratio at 25% thresholding, which happened to be 1.26.

After compressing the wavelet coefficients, they are stored in text files. One text file was created for all Scenarios except for Scenario 3 where wavelet coefficients are stored depending on tag resolution. The algorithm created 16 different text files; one for each different tag as each tag stores a different set of wavelet coefficients. The following table describes the storage space required (in bytes) for each tag for each scenario and with a different number of tags in the network.

Scenario	Threshold	8 tags		16 tags		XML (16 tags)	
		A	B	A	B	A	B
1	Lossless	70	101	137	173	602	864
	25%	55	86	110	156	451	646
	0.50	63	94	124	168	598	860
2	Lossless	89	119	180	240	588	1065
	25%	66	96	141	201	488	974
	0.50	70	101	114	174	582	1065
3	No threshold as it already has lossy data	54	85	76	136	214	684
4	Lossless	82	112	171	231	531	1001
	25%	60	91	140	200	433	903
	0.50	63	94	126	186	525	995

A: If only Coefficients are stored for X-Axis and Y Axis in the tag and treat tag positions as tag labels
B: If Tag Labels, X-Axis, and Y-Axis wavelet Coefficients are stored in the tag

As active RFID tags have storage capacities that can up to 128 Kilobytes while passive RFID tags can have up to 32 KB of storage capacity, the data compression schemes used in this research easily meet the storage constraints involved with both types of tags. The storage size for the16 tag-environment is 208 bytes. Thus, it can be seen that, except for the standard decomposition method operating in lossless mode (Scenario 2) which requires 240 byes to store tag data, the other scenarios require lesser storage space than with the original data. If it is assumed that all tags are deployed sequentially and labels are 1,2,3... and so on are assigned to the tags, then only the X-Axis and Y-Axis coordinates data need be stored, thus saving a great deal of storage space. But if tags are not deployed in sequential order, tag label data will need to be stored for each tag; which will still not cause a storage overflow.

Two file types are tested here to store tag data, i.e. CSV and XML. The XML file type takes much larger space to store the data which incurs a storage overhead which is to 4 to 5 times greater than with the CSV file format. Accordingly, CSV format is recommended to store the wavelet coefficients data in an RFID tag.

Thus, taking into consideration the above storage capacity table, Scenario 3 (i.e. wavelet coefficients stored as per tag resolution) gives better results as it requires less memory capacity to store tag information. But when compression ratio is taken into consideration, the standard decomposition technique with 0.50 thresholding tends to give better results when compared to the other modes of compression.

6.2 Navigation Experiment

After successfully conducting experimentation with data compression for the four different scenarios, the next goal was to test the effect of compression on navigation performance. For discussion of the navigation experimentation, the 16 tag configuration is selected. The experimental results are summarized below:

(Here Actual Distance means the distance measured on the shortest path between source and destination tags)

Scenario	Compression Mode	Source Tag	Destination Tag	Actual Distance	Distance Travelled	No. of Hops	% Excess
1	Lossless	T2	T15	21.95	24.07	07	9.66%
	0.50 Threshold	T2	T15	21.95	23.83	06	8.56%
	25% Threshold	T2	T15	23.85	24.48	05	2.64%
2	Lossless	T2	T15	21.95	24.07	07	9.66%
	0.50 Threshold	T2	T15	21.63	23.97	06	10.82%
	25% Threshold	T2	T15	21.26	21.78	07	2.45%
3	---	T2	T15	21.95	28.32	07	28.96%
4	Lossless	T2	T15	21.95	24.07	07	9.66%
	0.50 Threshold	T2	T15	23.02	26.25	06	14.03%
	25% Threshold	T2	T15	23.85	26.45	07	10.90%

Thus it is clear from the above table that if all wavelet coefficients are stored in lossless and lossy modes in Scenario 1, the results will be exactly the same as with no compression applied. The threshold value 25% gives better results as compared to threshold value 0.50 in the first scenario. The user needs to travel 2.64% greater distance at 25% thresholding as compared to 8.56% in the case of 0.50 thresholding.

If the standard decomposition method is applied, the lossless mode gives the same results as with no compression applied. In the lossy mode, 25% thresholding gives better results than with thresholding at 0.50. The distance to travel decreased by 1.19 units when thresholding at 0.50 was applied while it is decreased by 2.74 units.

Scenario 3 gives the worse results when compared to other scenarios. The user has to travel 28.96% more to reach tag T15 from tag T2.

In Scenario 4 where we applied the non-standard wavelet decomposition method, once again the 25% thresholding value gave better results than with the 0.50 threshold value. With the 25% threshold value, the user needs to travel 10.90% more as compared to a 14.03% excess using the 0.50 threshold value.

The number of hops was also less when 0.50 threshold value was applied in all experiments. In the lossless mode, the navigation required 7 hops to travel from tag T2 to tag T15. But, when 0.50 threshold value was applied, the number of hops reduced to 6 for all cases from 7, except for the first scenario where the number of hops was 5.

Thus, from a navigation viewpoint, data compression using 25% thresholding with the standard decomposition method gives the best results. We observe that the optimal thresholding level differs depends on the viewpoint; from the compression perspective 0.5 thresholding gives the best compression, whereas from the navigation viewpoint the 25% thresholding gives the best results. The ultimate choice between the two thresholds should thus be determined on a case-by- case basis depending on the user's preference between storage efficiency and navigation performance.

Chapter 7: Conclusion

As mentioned earlier, the focus of this study was to design efficient data compression methods using the Haar wavelet scheme when applied on RFID tags in an indoor or underground navigation environment. The study also investigated whether data compressed by using different methods are sufficiently efficient to navigate through RFID tags deployed in the environment as compared to navigation with non-compressed data. Three different scenarios were discussed in this study. Storage capacity was assessed with the different data compression schemes employed. Thresholding was also applied on the wavelet coefficients. Thresholding played was shown to play a significant role in data compression. A proper threshold value results in a better compression ratio with less error, i.e. with minimal changes in reconstructed values. Storage requirements after data compression is much reduced from the original non compressed form.

Raviraj, P., et. al., (2007) designed a computationally efficient algorithm using the *Haar* wavelet. They used the two dimensional discrete wavelet transform on image data. They experimented with thresholding levels of 25%, 10%, 5% and 1% and obtained promising results as they were able to reconstruct the images with very minor loss in quality while achieving high compression rates. However, to date there has been no study reported in the literature that the author is aware of that applies wavelet compression to location data stored on RFID tags. Thus this research provides new insights into the use of wavelets by providing a new area for its application, particularly in view of its success in both storage efficiency and navigation performance.

Scalability of these scenarios is with respect to Compression Ratio and Network Size. As we discussed in our experimental study in Chapter 6, Compression Ratio increased when we implemented data compression with Scenario 3. The 8-tag network has compression ratio of 1.33 while it steadily increases to 2.00, 3.20, 5.33 for 16-tag, 32-tag and 64-tag networks. This shows that as the number of tags is increased in this Scenario, Compression Ratio increased as well, thus demonstrating compression ration scales very well under Scenario 3. In other Scenarios as well, proper utilization of threshold values will help to implement data compression algorithm effectively in a larger network.

Application of the system in real world scenarios

The system as described in this research was designed to operate in indoor and underground environments where RFID tags can be installed. The system can thus be implemented in large hospitals where RFID tags are installed in different localities such as ICU, blood banks, emergency units, operating theaters, drug stores and so on. The patient will have navigation devices such as PDAs or mobile devices. The patient will enter the location of the place that they want to visit such as a blood bank. Then the navigation system operating with the help of the RFID tag network will respond by guiding the patient to reach the desired destination. This will reduce the search time in very large hospitals and be helpful in emergency situations.

This system can also be used in underground mining projects where we need to find the location of miners working in different parts of the mine. Very large warehouses are candidates for the use of such technology. In a large warehouse, it can help employees to find locations of certain product sections, or departments. It can also be helpful to museum visitors and act as a guide to navigate visitors to certain parts of the museum that they are interested in.

Strengths and Limitations

There are certain strengths and limitations to this research study. One of the strengths is that this study analyzed data compression techniques in three different realistic scenarios, involving lossless compression, lossy standard decomposition, lossy non-standard decomposition and a novel lossy compression scheme that we introduced that is based on tag resolution. Different threshold values are also applied in the lossy schemes to ascertain their effect on the accuracy of reconstructed values. Though data compression was tested on the 8-tag and 16-tag environments, in principle they can be applied across much larger environments as well, without any need to change the algorithms involved. All wavelet coefficients generated in different experiments are stored in CSV files which are then used in the navigation experiment. We also devised a novel navigation algorithm that was designed to operate with sparse approximate maps produced with the application of lossy data compression. Our experimentation showed that the navigation algorithm was able to achieve high levels of performance with the right choice of thresholding level used for data compression.

Most researchers studied different wavelet decomposition methods and their efficacy separately. There is no study known to the author that compared the performance of these methods viz., standard and non-standard wavelet decomposition methods. This research thus helped to fill the gap in this regard.

Just as there are a number of strengths to the study, some limitations exist, as well. The

main limitation is that the system is not implemented and tested in a real environment. Such testing was considered to be outside the scope of the study but there is every reason to believe that the compression and navigation efficiencies that were obtained will carry through to real world usage as our experimentation was generic in nature and did not assume certain specialized application environments.

Future Research Directions

As mentioned in the previous section about the strengths and limitations of the study, the future direction is to implement and analyze the system in real case scenario and test the outcome and navigation path against standards. The feasibility and reliability of the system in a real world needs to be established.

A future extended study, apart from testing in the real world will be how to avoid storing duplicate data in the RFID tags. During the study, it was observed that nearest neighbour tags end up storing wavelet coefficients that are similar in value. A, further study needs to test the implementation of an RFID reference tag which will store the duplicate (or near duplicate) wavelet coefficients, thus enabling a higher data compression ratio to be achieved.

REFERENCES

Balevic, A., Rockstroh, L., Wroblewski, M., Simon, S. (2008). "Using Arithmetic Coding for Reduction of Resulting Simulation Data Size on Massively Parallel GPGPUs." Recent Advances in Parallel Virtual Machine and Message Passing Interface **5205**: 295-302.

Barr, K. C., Asanovic, K. (2006). "Energy-aware Lossless Data Compression." ACM Transactions on Computer Systems (TOCS) **24(**3): 250-291.

Caplinskas, A., Vasilecas, O. (2004). Information Systems Research Methodologies and Models. The 5th International Conference on Computer Systems and Technologies. Bulgaria**:** 1-6.

Chakrabarti, K., Garofalakis, M., Rastogi, R., Shim, K. (2001). "Approximate query processing using wavelets." The VLDB Journal **10**(2-3): 199-233.

Chon, H. D., Jun, S., Jung, H., Won An, S. (2004). "Using RFID for Accurate Positioning." Journal of Global Positioning Systems **3**(1-2): 32-39.

de Villiers, M. R. (2005). Three approaches as pillars for interpretive information systems research: development research, action research and grounded theory. Proceedings of the 2005 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries, South Africa.

Deligiannakis, A., Garofalakis, M., Roussopoluos, N. (2007). "Extended Wavelets for Multiple Measures." ACM Transactions on Database Systems **32(**2).

Domdouzis, K., Kumar, B., Anumba, C. (2007). "Radio-Frequencey Identification (RFID) Applications: A brief introducdtion." Advanced Engineering Informatics **21**(4): 350-355.

Dvorsky, J., Pokorny, J., Snasel, V. (1999). "Word-Based Compression methods and Indexing for Text Retrieval Systems." Advances in Databases and Information Systems **1691/1999**: 76-84.

Fakeh, R., Ghani, A.A.A. (2009). "Empirical Evaluation of Decomposition Strategy of Wavelet Video Compression." International Journal of Image Processing (IJIP) **3**(1): 31-54.

Fazzinga, B., Flesca, S., Masciari, E., Furfaro, F. (2009). Efficient and Efffective RFID Data Warehousing. IDEAS '09 Proceedings of the 2009 International Database Engigineering & Applications Syposium, Calabria (Italy), ACM.

Garofalakis, M., Kumar, A. (2004). Deterministic wavelet thresholding for maximum-error metrics. Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of database systems, Paris, France.

Garofalakis, M., Gibbons, P.B. (2004). "Probabilistic Wavelet Synopses." Transactions on Database Systems (TODS) **29**(1): 43-90.

Hevner, A. R., March, S.T., Park, J., Ram, S. (2004). "Design Science in Information Systems Research." MIS Quarterly **28**(1): 75-105.

Howard, P. G., Vitter, J.S. (1994). Arithmetic Coding for Data Compression. Proceedings of the IEEE. **82(6):** 857-865.

Hu, Y. C., Chang C. (2000). "A new lossless compression scheme based on Huffman coding scheme for image compression." Signal Processing: Image Communication **16**(4): 367-372.

Huffmire, T., Sherwood, T. (2006). Wavelet-based Phase Classification. Proceedings of the 15th International Conference on Parallel Architectures and Compilation Techniques, New York, ACM.

Iivari, J. (2007). "A Paradigmatic Analysis of Information Systems As a Design Science." Scandinavian Journal of Information Systems **19**(2): 39-64.

Jain, A., Sen Gupta, I. (2007). A JPEG compression resistant steganography scheme for raster graphics images. TENCON 2007 - 2007 IEEE Region 10 Conference**:** 1-4.

Kambli, M., Bhatia, S. (2010). "Comparison of different Fingerpring Compression Techniques." Signal & Image Processing : An International Journal (SIPIJ) **1**(1): 27-39.

Khalifa, O. O., Harding, S.H., Hashim, A.A. (2008). "Compression using Wavelet Transform." Signal Processing: An International Journal **2**(5): 17-26.

Khorrami, H., Moavenian (2010). "A Comparative Study of DWT, CWT and DCT transformations in EDG arrhythmias classification." Expert Systems with Applications **37**: 5751-5757.

Koumaras, H., Kourtis, A., Lin, C.H., Shieh, C.K. (2008). "End-to-End Prediction Model of Video Quality and Decodable Frame Rate for MPEG Broadcasting Services." International Journal on Advances in Networks and Services **1**(1): 19-29.

Latu, G. (2010). "Sparse data structure design for wavelet-based methods." Retrieved 25-07-2011, from http://icps.u-strasbg.fr/~latu/wavelet/course_note.pdf.

Lionel, M. N., Yunhao, L., Yiu, C.L., Patil, A.P. (2004). "LANDMARC: Indoor Location Sensing Using Active RFID." Wireless Networks **10**: 701-710.

Ng, R., Ramamoorthi, R., Hanrahan, P. (2003). "All-frequency shadows using non-linear wavelet lighting approximation." ACM Transactions on Graphics (TOG) **22**(3): 376-381.

Pais, S., Symonds, J. (2011). "Data Storage on a RFID Tag for a distributed system." International Journal of UbiComp (IJU) **2**(2): 1-14.

Papageorgiou, C., Poggiom, T. (2000). "A Trainable System for Object Detection." International Journal of Computer Vision **38**(1): 15-33.

Potdar, M., Chang, E., Potdar, V. (2006). Applications of RFID in Pharmaceutical Industry. ICIT 2006 : IEEE International Conference on Industrial Technology. Mumbai, India**:** 2860-2865.

Raviraj, P., Sanavullah, M.Y. (2007). "The Modified 2D-Haar Wavelet Transformation in Image Compression." Middle-East Journal of Scientific Research **2**(2): 73-78.

Schomer, D. F., Elekes, A.A., Hazle, J.D., Huffman, J.C., Thompson, S.K., Chui, C.K., Murphy, W.A. (1998). "Introduction to wavelet-based compression of medical images." RadioGraphics - The Journal of continuing medical education in radiology **18**: 469-481.

Singh, S., Sharma, R.K., Sharma, M.K. (2009). "Use of Wavelet Tranform Extension for Graphics Image Compression using JPEG2000 Framework." International Journal of Image Processing (IJIP) **3**(1): 55-60.

Singh, T., Chopra, S., Kaur, H., Kaur, A. (2010). "Image Compression Using Wavelet and Wavelet packet Transformation." International Journal on Computer Science and Technology **1**(1): 97-99.

Singla, V., Singla, R., Gupta, S. (2008). "Data Compression Modelling: Huffman and Arithmetic." International Journal of the Computer, the Internet and Management **16**(3): 64-68.

Stabno, M., Wrembel, R. (2009). "RLH: Bitmap compression technique based on run-length and Huffman encoding." Information Systems **34**(4-5): 400-414.

Stollnitz, E. J., DeRose, A.D., Salesin, D.H. (1995). "Wavelets for Computer Graphics: A Primer 1." Computer Graphics and Applications, IEEE **15**(3): 76-84.

Stollnitz, E. J., DeRose, A.D., Salesin, D.H. (1995). "Wavelets for Computer Graphics: A Primer 2." Computer Graphics and Applications, IEEE **15**(4): 75-85.

Talukder, K. H., Harada, K. (2007). "Haar Wavelet Based Approach for Image Compression and Quality Assessment of Compressed Image." IAENG International Journal of Applied Mathematics **36**(1).

Tesoriero, R., Gallud, J.A., Lozano, M., Penichet, V.M.R. (2008). "Using Active and Passive RFID Technology to Support Indoor Location-Aware Systems." IEEE Transactions on Consumer Electronics **54**(2): 578-583.

Tichy, W. F. (1998). "Should computer scientists experiment more?" Computer **31**(5): 32-40.

Vitter, J. S., Wang, M. (1999). Approximate computation of multidimensional aggregates of sparse data using wavelets. SIGMOD '99 Proceedings of the 1999 ACM SIGMOD International Conference of Management of Data, NY, USA, ACM.

Walker, W. (2005). "The strengths and weaknesses of research designs involving quantitative measures." Journal of Research in Nursing **10**(5): 571-582.

Want, R. (2006). "An Introduction to RFID Technology." PERVASIVE Computing: 25-33.

Ward, M., Van Kranenburg, R., (May 2006). "RFID: Frequency, standards, adoption and innovation." JISC Technology and Standards Watch.

Witten, I. H., Neal, R.M., Cleary, J.G. (1987). Arithmetic Coding for Data Compression. Communications of the ACM. New York, USA, ACM, New York. **30:** 520-540.

Xia, J., Jiang, J., Yang, S.Y., Hou, C.H. (2003). An empirical study to turn MPEG-2 into lossless video compression. International Conference on Visual Information Engineering, VIE 2003.

ANNEXURES

Annexure 1 : Tag Creation Code

```
 N = Convert.ToInt32(txtTags.Text)
Dim orandom As System.Random
ReDim Tags(N - 1), XAxis(N - 1), YAxis(N - 1)
Dim tagdata(N - 1, 3), T As Integer
orandom = New System.Random
fullpath = " F:\ThesisLATEST\HaarWavelet\Data\Tags_16_New.txt"
objReader = New StreamWriter(fullpath)
For i = 0 To N - 1
   Tags(i) = i + 1
   ok = False
   While ok = False
      X = orandom.Next(100)
      Y = orandom.Next(100)
      Xindex = Array.IndexOf(XAxis, X)
      YIndex = Array.IndexOf(YAxis, Y)
      If Xindex <> YIndex Then
         ok = True
      Else
         If Xindex = -1 Then
            ok = True
         End If
      End If
   End While
   XAxis(i) = X
   YAxis(i) = Y
   tagdata(i, 0) = i + 1
   tagdata(i, 1) = X
   tagdata(i, 2) = Y
   tagdata(i, 3) = Math.Sqrt(X * X + Y * Y)
Next
For i = 0 To N - 2
   For j = i + 1 To N - 1
      If XAxis(i) > XAxis(j) Then
         Xindex = XAxis(i)
         XAxis(i) = XAxis(j)
         XAxis(j) = Xindex
         YIndex = YAxis(i)
         YAxis(i) = YAxis(j)
         YAxis(j) = YIndex
      End If
      If tagdata(i, 3) > tagdata(j, 3) Then
         T = tagdata(i, 0)
         X = tagdata(i, 1)
         Y = tagdata(i, 2)
         dist = tagdata(i, 3)
         tagdata(i, 0) = tagdata(j, 0)
         tagdata(i, 1) = tagdata(j, 1)
         tagdata(i, 2) = tagdata(j, 2)
         tagdata(i, 3) = tagdata(j, 3)
         tagdata(j, 0) = T
         tagdata(j, 1) = X
```

```
            tagdata(j, 2) = Y
            tagdata(j, 3) = dist
        End If
    Next
Next
For i = 0 To N - 1
    objReader.Write(tagdata(i, 0) & ", " & tagdata(i, 1) & ", " & tagdata(i, 2) & " | ")
Next
objReader.Close()
Label2.Text = "Tags created...."
```

Annexure 2 : Wavelet Decomposition in Scenario 1

```
 lblTagCoeff.Text = ""
lblXCoeff.Text = ""
lblYCoeff.Text = ""
'       S = 3
N = Tags.Length
For i = 1 To 100
   If N = 2 ^ i Then
      S = i
      Exit For
   End If
Next
ReDim Tavg(N - 1), Xavg(N - 1), Yavg(N - 1)
For i = 0 To N - 1
   Tavg(i) = Tags(i)
   Xavg(i) = XAxis(i)
   Yavg(i) = YAxis(i)
   Tsum += Tags(i)
   Xsum += XAxis(i)
   Ysum += YAxis(i)
Next
avg1 = Tsum / N
avg2 = Xsum / N
avg3 = Ysum / N
ReDim TagsCoeff(N - 1), XCoeff(N - 1), YCoeff(N - 1)
For cnt = S To 1 Step -1
   ReDim Preserve Tavg1(N / 2 - 1)
   ReDim Preserve Xavg1(N / 2 - 1)
   ReDim Preserve Yavg1(N / 2 - 1)
   ReDim Preserve diff1(N / 2 - 1)
   ReDim Preserve diff2(N / 2 - 1)
   ReDim Preserve diff3(N / 2 - 1)
   j = 0
   k = 0
   For i = 0 To N - 2
      Tavg1(j) = (Tavg(i) + Tavg(i + 1)) / 2
      Xavg1(j) = (Xavg(i) + Xavg(i + 1)) / 2
      Yavg1(j) = (Yavg(i) + Yavg(i + 1)) / 2
      diff1(k) = Tavg(i) - Tavg1(j)
      diff2(k) = Xavg(i) - Xavg1(j)
      diff3(k) = Yavg(i) - Yavg1(j)
      i += 1
      j += 1
      k += 1
   Next
   j = k
   For i = 0 To k - 1
      TagsCoeff(j) = diff1(i)
      XCoeff(j) = diff2(i)
      YCoeff(j) = diff3(i)
      j += 1
   Next

   ReDim Tavg(N / 2 - 1), Xavg(N / 2 - 1), Yavg(N / 2 - 1)
   For i = 0 To N / 2 - 1
      Tavg(i) = Tavg1(i)
      Xavg(i) = Xavg1(i)
```

```
            Yavg(i) = Yavg1(i)
        Next
        N = N / 2
    Next
    TagsCoeff(0) = avg1 'Tsum / N
    XCoeff(0) = avg2 'Xsum / N
    YCoeff(0) = avg3 'Ysum / N

    N = Tags.Length
    For i = 0 To N - 1
        TagsCoeff(i) = Math.Round(TagsCoeff(i), 2)
        XCoeff(i) = Math.Round(XCoeff(i), 2)
        YCoeff(i) = Math.Round(YCoeff(i), 2)
    Next
    ' ------------ Writing to text file
    fullpath = "C:\ThesisLATEST\HaarWavelet\Data\WaveletCoeff_S1_" & N & ".txt"
    objReader = New StreamWriter(fullpath)
    For i = 0 To N - 1
        objReader.Write(TagsCoeff(i))
        If i <> N - 1 Then
            objReader.Write(",")
        End If
    Next
    objReader.Write("|")
    For i = 0 To N - 1
        objReader.Write(XCoeff(i))
        If i <> N - 1 Then
            objReader.Write(",")
        End If
    Next
    objReader.Write("|")
    For i = 0 To N - 1
        objReader.Write(YCoeff(i))
        If i <> N - 1 Then
            objReader.Write(",")
        End If
    Next
    objReader.Close()
    '-------------
    For i = 0 To Tags.Length - 1
        lblTagCoeff.Text &= TagsCoeff(i) & "  "
        lblXCoeff.Text &= XCoeff(i) & "  "
        lblYCoeff.Text &= YCoeff(i) & "  "
    Next
```

Annexure 3 : Wavelet Decomposition in Scenario 2 (Standard Decomposition)

```
 lblTagCoeff.Text = ""
lblXCoeff.Text = ""
lblYCoeff.Text = ""

' L = N / 2
L = txtTags.Text / 2
i = 0
ReDim myArr(3, L - 1), CoeffArr(3, L - 1), AxisData(L - 1)
For r = 0 To 3 Step 2
   For c = 0 To L - 1
      myArr(r, c) = XAxis(i)
      myArr(r + 1, c) = YAxis(i)
      i = i + 1
   Next
Next

'----- First Pass Row Decomposition
N = L
For i = 1 To 100
   If N = 2 ^ i Then
      S = i
      Exit For
   End If
Next
For r = 0 To 3
   For i = 0 To L - 1
      AxisData(i) = myArr(r, i)
   Next

   N = L
   Tsum = 0
   ReDim Tavg(N - 1), diff1(N - 1), Tavg1(N - 1)
   For i = 0 To N - 1
      Tavg(i) = AxisData(i)
      Tsum += AxisData(i)
   Next
   avg1 = Tsum / N
   ReDim TagsCoeff(N - 1)
   For cnt = S To 1 Step -1
      ReDim Preserve Tavg1(N / 2 - 1)
      ReDim Preserve diff1(N / 2 - 1)
      j = 0
      k = 0
      For i = 0 To N - 2
         Tavg1(j) = (Tavg(i) + Tavg(i + 1)) / 2
         diff1(k) = Tavg(i) - Tavg1(j)
         i += 1
         j += 1
         k += 1
      Next
      j = k
      For i = 0 To k - 1
         TagsCoeff(j) = diff1(i)
         j += 1
      Next
```

```
        ReDim Tavg(N / 2 - 1)
        For i = 0 To N / 2 - 1
            Tavg(i) = Tavg1(i)
        Next
        N = N / 2
    Next
    TagsCoeff(0) = avg1 'Tsum / N
    For i = 0 To N - 1
        TagsCoeff(i) = Math.Round(TagsCoeff(i), 2)
    Next

    For i = 0 To L - 1
        CoeffArr(r, i) = TagsCoeff(i)
    Next
Next

'----- Second Pass Column Decomposition

N = 4 'L
ReDim AxisData(3), CoeffArr2Pass(L - 1, 3)
For i = 1 To 100
    If N = 2 ^ i Then
        S = i
        Exit For
    End If
Next
For c = 0 To L - 1
    For i = 0 To 3
        AxisData(i) = CoeffArr(i, c)
    Next

    N = 4 'L
    Tsum = 0
    ReDim Tavg(N - 1), diff1(N - 1), Tavg1(N - 1)
    For i = 0 To N - 1
        Tavg(i) = AxisData(i)
        Tsum += AxisData(i)
    Next
    avg1 = Tsum / N
    ReDim TagsCoeff(N - 1)
    For cnt = S To 1 Step -1
        ReDim Preserve Tavg1(N / 2 - 1)
        ReDim Preserve diff1(N / 2 - 1)
        j = 0
        k = 0
        For i = 0 To N - 2
            Tavg1(j) = (Tavg(i) + Tavg(i + 1)) / 2
            diff1(k) = Tavg(i) - Tavg1(j)
            i += 1
            j += 1
            k += 1
        Next
        j = k
        For i = 0 To k - 1
            TagsCoeff(j) = diff1(i)
            j += 1
        Next

        ReDim Tavg(N / 2 - 1)
        For i = 0 To N / 2 - 1
            Tavg(i) = Tavg1(i)
```

```
            Next
            N = N / 2
        Next
        TagsCoeff(0) = avg1 'Tsum / N
        For i = 0 To N - 1
            TagsCoeff(i) = Math.Round(TagsCoeff(i), 2)
        Next

        For i = 0 To 3
            CoeffArr2Pass(c, i) = TagsCoeff(i)
        Next
    Next

    '-------
    fullpath = "C:\ThesisLATEST\HaarWavelet\Data\WaveletCoeff_S2_" & Tags.Length & ".txt"
    objReader = New StreamWriter(fullpath)

    For r = 0 To L - 1
        For c = 0 To 3
            objReader.Write(Math.Round(CoeffArr2Pass(r, c), 2))
            If c <> 3 Then objReader.Write(",")
        Next
        objReader.WriteLine()
    Next
    objReader.Close()
```

Annexure 4 : Wavelet Decomposition in Scenario 3

```
    Private Sub btnWaveDecompo_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnWaveDecompo.Click
    Dim PathArray() As Integer, ind As Integer, newXCoeff() As Decimal, newYCoeff() As Decimal, k1 As
Integer, TagInd As Integer
    WaveletCalc()
    For i = 0 To Tags.Length - 1
      TagsCoeff(i) = Math.Round(TagsCoeff(i), 2)
      XCoeff(i) = Math.Round(XCoeff(i), 2)
      YCoeff(i) = Math.Round(YCoeff(i), 2)
    Next
    For TagInd = 0 To Tags.Length - 1
      lblTagCoeff.Text = XCoeff.Length
      lblXCoeff.Text = ""
      lblYCoeff.Text = ""
      For i = 0 To XCoeff.Length - 1
        lblXCoeff.Text &= XCoeff(i) & ","
        lblYCoeff.Text &= YCoeff(i) & ","
      Next

      CoeffPath(Tags.Length, TagInd + 1)
      ind = path.Length + pair.Length - 2
      ReDim PathArray(ind)
      For j = 0 To path.Length - 1
        PathArray(j) = Math.Abs(path(j))
      Next
      k1 = 0
      For k = j To ind
        PathArray(k) = Math.Abs(pair(k1))
        k1 += 1
      Next
      Array.Sort(PathArray)
      '---------------------
      ReDim newXCoeff(PathArray.Length - 1), newYCoeff(PathArray.Length - 1)
      For j = 0 To PathArray.Length - 1
        ind = PathArray(j)
        newXCoeff(j) = XCoeff(ind)
        newYCoeff(j) = YCoeff(ind)
      Next
      ' ------------ Writing to text file
      fullpath = "C:\ThesisLATEST\HaarWavelet\Data\Experiment Data\WaveletCoeff_S3_" & N & "_" &
TagInd + 1 & ".txt"
      objReader = New StreamWriter(fullpath)
      Dim m As Integer

      For m = 0 To newXCoeff.Length - 1
        objReader.Write(newXCoeff(m))
        If m < newXCoeff.Length - 1 Then
          objReader.Write(",")
        End If
      Next
      objReader.Write("|")
      For m = 0 To newYCoeff.Length - 1
        objReader.Write(newYCoeff(m))
        If m < newYCoeff.Length - 1 Then
```

```
            objReader.Write(",")
        End If
    Next
    objReader.Close()
    '-------------

  Next
End Sub

Public Sub WaveletCalc()
  N = Tags.Length
  For i = 1 To 100
    If N = 2 ^ i Then
      S = i
      Exit For
    End If
  Next
  ReDim Tavg(N - 1), Xavg(N - 1), Yavg(N - 1)
  For i = 0 To N - 1
    Tavg(i) = Tags(i)
    Xavg(i) = AxisData(i, 0)
    Yavg(i) = AxisData(i, 1)
    Tsum += Tags(i)
    Xsum += AxisData(i, 0)
    Ysum += AxisData(i, 1)
  Next
  avg1 = Tsum / N
  avg2 = Xsum / N
  avg3 = Ysum / N
  ReDim TagsCoeff(N - 1), XCoeff(N - 1), YCoeff(N - 1)
  For cnt = S To 1 Step -1
    ReDim Preserve Tavg1(N / 2 - 1)
    ReDim Preserve Xavg1(N / 2 - 1)
    ReDim Preserve Yavg1(N / 2 - 1)
    ReDim Preserve diff1(N / 2 - 1)
    ReDim Preserve diff2(N / 2 - 1)
    ReDim Preserve diff3(N / 2 - 1)
    j = 0
    k = 0
    For i = 0 To N - 2
      Tavg1(j) = (Tavg(i) + Tavg(i + 1)) / 2
      Xavg1(j) = (Xavg(i) + Xavg(i + 1)) / 2
      Yavg1(j) = (Yavg(i) + Yavg(i + 1)) / 2
      diff1(k) = Tavg(i) - Tavg1(j)
      diff2(k) = Xavg(i) - Xavg1(j)
      diff3(k) = Yavg(i) - Yavg1(j)
      i += 1
      j += 1
      k += 1
    Next
    j = k
    For i = 0 To k - 1
      TagsCoeff(j) = diff1(i)
      XCoeff(j) = diff2(i)
      YCoeff(j) = diff3(i)
      j += 1
    Next

    ReDim Tavg(N / 2 - 1), Xavg(N / 2 - 1), Yavg(N / 2 - 1)
    For i = 0 To N / 2 - 1
      Tavg(i) = Tavg1(i)
      Xavg(i) = Xavg1(i)
```

```
            Yavg(i) = Yavg1(i)
        Next
        N = N / 2
    Next
    TagsCoeff(0) = avg1 'Tsum / N
    XCoeff(0) = avg2 'Xsum / N
    YCoeff(0) = avg3 'Ysum / N
End Sub
Public Sub CoeffPath(ByRef nValue As System.String, ByRef iValue As System.String)
    Dim n1 As Decimal, root As Integer, k1 As Integer, k2 As Integer, pos As Integer, sign As Integer
    N = Convert.ToInt32(nValue)
    pos = Convert.ToInt32(iValue)
    For i = 1 To 100
        If N = 2 ^ i Then
            S = i
            Exit For
        End If
    Next
    n1 = N / 2
    Root = 1
    'pos = i
    k1 = 2
    k2 = 0
    ReDim path(S + 1)
    ReDim pair(S - 1)
    ReDim coeff(S * 2)
    path(0) = 0
    If pos <= N / 2 Then
        sign = 1
        path(1) = 1
        Root = 2
    Else
        Root = 3
        path(1) = -1
        pos = pos - N1
        sign = -1
    End If

    For j = 2 To S
        If pos <= N1 / 2 Then
            path(j) = Root
            If Root Mod 2 = 0 Then
                pair(j - 2) = (Root + 1) * -1
            Else
                pair(j - 2) = (Root - 1)
            End If
            Root = Root * 2
        Else
            path(j) = Root * -1
            If Root Mod 2 = 0 Then
                pair(j - 2) = (Root + 1) * -1
            Else
                pair(j - 2) = (Root - 1)
            End If
            Root = (Root * 2) + 1

            If pos > N1 Then
                pos = pos - N1
                'Root = Root + 1
            Else
                If N1 > pos Then
                    pos = N1 - pos
```

```
                'Root = Root + 1
            End If
        End If
    End If
    N1 = N1 / 2
  Next
End Sub
```

Annexure 5 : Wavelet Decomposition in Scenario 4(Non-Standard Decomposition)

```
Private Sub btn_Decomposition_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btn_Decomposition.Click
        N = txtTags.Text
        ReDim AxisData(N / 2 - 1, N / 4 - 1)
        k = 0
        For i = 0 To 7 Step 2
            For c = 0 To N / 4 - 1
                AxisData(i, c) = XAxis(k)
                AxisData(i + 1, c) = YAxis(k)
                k = k + 1
            Next
        Next
        ReDim CoeffArr(N / 2 - 1, N / 4 - 1), XCoeff(N / 4 - 1)
        '---------- Row Decomposition
        For cnt = 0 To N / 2 - 1
            k = 0
            For c = 0 To N / 4 - 1
                XCoeff(k) = AxisData(cnt, c)
                k = k + 1
            Next
            Nonstandard_Decompose()
            k = 0
            For c = 0 To N / 4 - 1
                AxisData(cnt, c) = XCoeff(k)
                k = k + 1
            Next
        Next
        '---------- Column Decomposition
        ReDim XCoeff(N / 2 - 1)
        For cnt = 0 To N / 4 - 1
            k = 0
            For c = 0 To N / 2 - 1
                XCoeff(k) = AxisData(c, cnt)
                k = k + 1
            Next
            Nonstandard_Decompose()
            k = 0
            For c = 0 To N / 2 - 1
                AxisData(c, cnt) = Math.Round(XCoeff(k), 2)
                k = k + 1
            Next
        Next
        '----------- Wavelet Coefficients before Threshold
        Label4.Text = ""
        NonZero1 = 0
        For r = 0 To 7
            For c = 0 To 3
                Label4.Text &= Format(AxisData(r, c), "#0.##")
                If AxisData(r, c) <> 0 Then NonZero1 += 1
                If c <> 3 Then Label4.Text &= ", "
            Next
            Label4.Text &= vbCrLf
        Next
        '----------- Apply Threshold
        If txtThreshHold.Text <> "0" Then
```

```
        For r = 0 To 7
            For c = 0 To 3
                'If Math.Abs(AxisData(r, c)) <= txtThreshHold.Text Then
                '    AxisData(r, c) = 0
                'End If
            Next
            AxisData(r, 3) = 0     '--- 25% Threshold
        Next
    End If
    '---------- Writing into file
    fullpath = "C:\ThesisLATEST\HaarWavelet\Data\WaveletCoeff_NS_" & Tags.Length & ".txt"
    objReader = New StreamWriter(fullpath)
    Label17.Text = ""
    NonZero2 = 0
    For r = 0 To 7
        For c = 0 To 3
            Label17.Text &= Format(AxisData(r, c), "#0.##")
            If c <> 3 Then Label17.Text &= ", "
            If AxisData(r, c) <> 0 Then NonZero2 += 1
            objReader.Write(Format(AxisData(r, c), "#0.##"))
            objReader.Write(",")
        Next
        Label17.Text &= vbCrLf
        objReader.WriteLine()
    Next
    objReader.Close()
    If NonZero2 <> 0 Then txtCratio.Text = Format(NonZero1 / NonZero2, "#0.##") Else txtCratio.Text = 0
End Sub

Private Sub Nonstandard_Decompose()
    k = XCoeff.Length
    L = XCoeff.Length ' k / 2
    While L > 1
        k = 0
        ReDim avg(L / 2 - 1), diff(L / 2 - 1)
        For i = 0 To L - 1 Step 2
            avg(k) = (XCoeff(i) + XCoeff(i + 1)) / 2
            diff(k) = XCoeff(i) - avg(k)
            k = k + 1
        Next
        k = 0
        For i = 0 To avg.Length - 1
            XCoeff(k) = avg(i)
            k = k + 1
        Next
        j = 0
        'k = k - 1
        For i = 0 To diff.Length - 1
            XCoeff(k) = diff(j)
            j = j + 1
            k = k + 1
        Next
        L = L / 2
    End While
End Sub
```

Annexure 6 : Wavelet Reconstruction in Scenario 1

```
    Public Sub DataReconstruct(ByRef S As Integer)
    N = Convert.ToInt32(txtTags.Text)
    Dim myTArr(N - 1) As Decimal, myXArr(N - 1) As Decimal, myYArr(N - 1) As Decimal, x As Decimal, y As
Decimal, ind As Integer, t As Integer
    ReDim Tavg(N - 1), Xavg(N - 1), Yavg(N - 1), diff1(N / 2 - 1), diff2(N / 2 - 1), diff3(N / 2 - 1)
    ind = 1
    k = 0
    ReDim XAxis(N - 1), YAxis(N - 1)
    XAxis(0) = XCoeff(0)
    YAxis(0) = YCoeff(0)
    For i = 0 To S - 1
      k = 0
      t = 0
      For j = 1 To 2 ^ i
        x = XCoeff(ind)
        y = XAxis(k)
        myXArr(t) = y + x
        myXArr(t + 1) = y - x

        x = YCoeff(ind)
        y = YAxis(k)
        myYArr(t) = y + x
        myYArr(t + 1) = y - x

        ind += 1
        k += 1
        t += 2
      Next
      For j = 0 To N - 1
        XAxis(j) = myXArr(j)
        YAxis(j) = myYArr(j)
      Next
    Next
  End Sub
```

Annexure 7 : Wavelet Reconstruction in Scenario 2(Standard Decomposition)

```
    Private Sub DataReconstruct()
    Dim t1 As Decimal, t2 As Decimal, tcoeff(N / 2) As Decimal, x As Decimal, y As Decimal, ind As Integer, t
As Integer
    '---- First Pass Column reconstruction
    N = 4
    L = txtTags.Text / 2
    Dim myXArr(N - 1) As Decimal, Tags1() As Decimal
    ReDim CoeffArr(3, L - 1), myArr(3, L - 1)
    For i = 1 To 100
      If N = 2 ^ i Then
        S = i
        Exit For
      End If
    Next
    For r = 0 To L - 1

      ReDim TagsCoeff(3)
      For i = 0 To 3
        TagsCoeff(i) = CoeffArr2Pass(r, i)
      Next

      ind = 1
      k = 0
      ReDim Tags1(N - 1)
      Tags1(0) = TagsCoeff(0)
      For i = 0 To S - 1
        k = 0
        t = 0
        For j = 1 To 2 ^ i
          x = TagsCoeff(ind)
          y = Tags1(k)
          myXArr(t) = y + x
          myXArr(t + 1) = y - x

          ind += 1
          k += 1
          t += 2
        Next
        For j = 0 To N - 1
          Tags1(j) = myXArr(j)
        Next
      Next

      For i = 0 To 3
        CoeffArr(i, r) = Tags1(i)
      Next
    Next
    '--------------- Second Pass Row reconstruction

    N = L
    For i = 1 To 100
      If N = 2 ^ i Then
        S = i
        Exit For
      End If
```

```
    Next
    For r = 0 To 3
      ReDim TagsCoeff(L - 1)
      For i = 0 To L - 1
        TagsCoeff(i) = CoeffArr(r, i)
      Next

      ReDim myXArr(N - 1)
      ReDim Tags1(N - 1)
      Tags1(0) = TagsCoeff(0)
      ind = 1
      k = 0
      For i = 0 To S - 1
        k = 0
        t = 0
        For j = 1 To 2 ^ i
          x = TagsCoeff(ind)
          y = Tags1(k)
          myXArr(t) = y + x
          myXArr(t + 1) = y - x

          ind += 1
          k += 1
          t += 2
        Next
        For j = 0 To N - 1
          Tags1(j) = myXArr(j)
        Next
      Next

      For i = 0 To L - 1
        myArr(r, i) = Math.Round(Tags1(i), 0)
      Next
    Next

End Sub
```

Annexure 8 : Wavelet Reconstruction in Scenario 3

```
    Public Sub DataReconstruct()
        N = Convert.ToInt32(txtTags.Text)
        Dim k As Integer, j As Integer
        Dim myTArr(N - 1) As Decimal, myXArr(N - 1) As Decimal, myYArr(N - 1) As Decimal, x As Decimal, y As
Decimal, ind As Integer, t As Integer

        For i = 1 To 100
            If N = 2 ^ i Then
                S = i
                Exit For
            End If
        Next
        ind = 1
        k = 0
        ReDim XAxis(N - 1), YAxis(N - 1)
        XAxis(0) = XCoeff(0)
        YAxis(0) = YCoeff(0)
        For i = 0 To S - 1
            k = 0
            t = 0
            For j = 1 To 2 ^ i
                x = XCoeff(ind)
                y = XAxis(k)
                myXArr(t) = y + x
                myXArr(t + 1) = y - x

                x = YCoeff(ind)
                y = YAxis(k)
                myYArr(t) = y + x
                myYArr(t + 1) = y - x

                ind += 1
                k += 1
                t += 2
            Next
            For j = 0 To N - 1
                XAxis(j) = myXArr(j)
                YAxis(j) = myYArr(j)
            Next
        Next
    End Sub
```

Annexure 9 : Wavelet Reconstruction in Scenario 4 (Non-Standard Decomposition)

```
    Private Sub btnReconstruct_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnReconstruct.Click

    '---------- Column Reconstruct
    colreconstruct = 1
    N = txtTags.Text
    For cnt = 0 To N / 4 - 1
      k = 0
      For c = 0 To N / 2 - 1
        XCoeff(k) = AxisData(c, cnt)
        k = k + 1
      Next
      NonStandard_Reconstruct()
      k = 0
      N = txtTags.Text
      For c = 0 To N / 2 - 1
        AxisData(c, cnt) = XCoeff(k)
        k = k + 1
      Next
    Next
    '---------- Row Reconstruct
    N = txtTags.Text
    colreconstruct = 0
    ReDim XCoeff(N / 4 - 1)
    For cnt = 0 To N / 2 - 1
      k = 0
      For c = 0 To N / 4 - 1
        XCoeff(k) = AxisData(cnt, c)
        k = k + 1
      Next
      NonStandard_Reconstruct()
      k = 0
      N = txtTags.Text
      For c = 0 To N / 4 - 1
        AxisData(cnt, c) = Math.Round(XCoeff(k), 0)
        k = k + 1
      Next
    Next
    '----------
    txtxAxisReconstruct.Text = ""
    txtyAxisReconstruct.Text = ""
    k = 0
    For i = 0 To 7 Step 2
      For c = 0 To N / 4 - 1
        txtxAxisReconstruct.Text &= AxisData(i, c) & " "
        txtyAxisReconstruct.Text &= AxisData(i + 1, c) & " "
        XAxis(k) = AxisData(i, c)
        YAxis(k) = AxisData(i + 1, c)
        k = k + 1
      Next
    Next
  End Sub
```

```
Private Sub NonStandard_Reconstruct()
    Dim m As Integer, x As Decimal, y As Decimal
    ReDim avg(0), diff(0)
    'N = txtTags.Text / 2
    N = XCoeff.Length '/ 2
    For i = 1 To 100
       If N = 2 ^ i Then
          S = i
          Exit For
       End If
    Next
    m = 2
    k = 2
    avg(0) = XCoeff(0)
    diff(0) = XCoeff(1)
    x = avg(0) + diff(0)
    y = avg(0) - diff(0)
    XCoeff(0) = x
    XCoeff(1) = y

    For i = 1 To S - 1
       j = avg.Length
       ReDim avg(j * 2 - 1)
       For j = 0 To avg.Length - 1
          avg(j) = XCoeff(j)
       Next
       k = avg.Length '+ 1
       ReDim diff(avg.Length - 1)
       For j = 0 To diff.Length - 1
          diff(j) = XCoeff(k)
          k = k + 1
       Next
       m = 0
       For j = 0 To i '+ 1
          x = avg(j) + diff(j)
          y = avg(j) - diff(j)
          XCoeff(m) = x
          XCoeff(m + 1) = y
          m = m + 2
       Next
       If i = S - 1 And colreconstruct = 1 Then
          x = avg(j) + diff(j)
          y = avg(j) - diff(j)
          XCoeff(m) = x
          XCoeff(m + 1) = y
          m = m + 2
       End If
    Next
End Sub
```

Annexure 10 : Navigation Algorithm Coding (Finding Next Tag to move)

```
    Private Sub btnNextTag_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnNextTag.Click

    If Path.Length = 1 Then
       Path(0) = txtSourceTag.Text
    End If

    N = Tags.Length
    For i = 1 To 100
       If N = 2 ^ i Then
          S = i
          Exit For
       End If
    Next

    For i = 0 To N - 1
       If Tags(i) = txtSourceTag.Text Then
          SourceX = XAxis(i)
          SourceY = YAxis(i)
          Exit For
       End If
    Next
    For i = 0 To N - 1
       If Tags(i) = TxtDestinationTag.Text Then
          DestX = XAxis(i)
          DestY = YAxis(i)
          Exit For
       End If
    Next

    DestAngle = Math.Round(Math.Atan2((SourceY - DestY), (SourceX - DestX)), 2)
    DestDistance = Math.Round(Math.Sqrt((DestX - SourceX) ^ 2 + (DestY - SourceY) ^ 2), 2)

    If DestAngle < 0 Then
       DestAngle = DestAngle + 3
    End If
    DestAngle = Math.Round(DestAngle * (180 / Math.PI), 2)

    If (SourceY - DestY) > 0 Then
       If SourceX > DestX Then DestAngle = DestAngle + 270
       If SourceX = DestX Then DestAngle = DestAngle + 180
       If SourceX < DestX Then DestAngle = DestAngle + 90
    End If

    If txtDestiDistance.Text = "" Then txtDestiDistance.Text = Math.Round(DestDistance, 2)
    If txtDestiAngle.Text = "" Then txtDestiAngle.Text = Math.Round(DestAngle, 2)

    DistanceCalc(txtSourceTag.Text)
    AngleCalc(txtSourceTag.Text)

    If DistanceArray(0, 0) = TxtDestinationTag.Text Then
       TagNext = DistanceArray(0, 0)
    Else
```

```
            CalculateAngleRange(DestAngle)
            TagNext = NextDistance(0, 0)
        End If
        If TagNext = 0 Then
            TagNext = TxtDestinationTag.Text
        End If

        TagNext = NextDistance(0, 0)
        If TagNext = 0 Then
            TagNext = TxtDestinationTag.Text
        End If

        Dim a As Integer = Path.Length
        If Array.IndexOf(Path, TagNext) = -1 Then
            tagindex = Array.IndexOf(Tags, txtSourceTag.Text)

            TxtNextTag.Text = TagNext
            ReDim Preserve Path(a)
            Path(a) = TagNext

            For i = 0 To 3
                If AngleArray(i, 0) = TagNext Then
                    nextTagAngle = AngleArray(i, 1)
                End If
                If DistanceArray(i, 0) = TagNext Then
                    nextTagDistance = DistanceArray(i, 1)
                End If
            Next
            txtNextTagDistance.Text = Math.Round(nextTagDistance, 2)
        End If
        TotDistTravelled = TotDistTravelled + Math.Round(nextTagDistance, 2)
        txtTotDistTravelled.Text = TotDistTravelled
        TxtPathTags.Text = ""
        For i = 0 To Path.Length - 1
            TxtPathTags.Text &= Path(i) & " "
        Next
    End Sub

    Private Sub DistanceCalc(ByRef Source As Decimal)
        Dim SourceTag As Decimal, X As Integer, Y As Integer
        SourceTag = Source
        For i = 0 To N - 1
            If Tags(i) = SourceTag Then
                X = XAxis(i)
                Y = YAxis(i)
                Exit For
            End If
        Next
        j = 0
        ReDim DistanceArray(N - 2, 1)
        For i = 0 To N - 1
            If Tags(i) <> SourceTag Then
                DistanceArray(j, 0) = Tags(i)
                DistanceArray(j, 1) = Math.Sqrt((X - XAxis(i)) ^ 2 + (Y - YAxis(i)) ^ 2) '+ 1
                j = j + 1
            End If
        Next
        SortDistanceArray()
    End Sub

    Private Sub AngleCalc(ByRef SourceTag As Decimal)
```

```
        Dim angle As Decimal, X1 As Integer, Y1 As Integer, X2 As Integer, Y2 As Integer, X As Integer, Y As
Integer
        For i = 0 To N
            If Tags(i) = SourceTag Then
                X1 = XAxis(i)
                Y1 = YAxis(i)
                Exit For
            End If
        Next
        j = 0
        ReDim AngleArray(N - 2, 1)
        For i = 0 To N - 2
            If Tags(i) <> SourceTag And Array.IndexOf(Path, Tags(i)) = -1 Then
                AngleArray(j, 0) = Tags(i)
                X2 = XAxis(i)
                Y2 = YAxis(i)
                angle = 0

                angle = Math.Atan2((Y1 - Y2), (X1 - X2))
                If angle < 0 Then
                    angle = angle + 3
                End If
                angle = angle * (180 / Math.PI)
                If (Y1 - Y2) > 0 Then
                    If X1 > X2 Then angle = angle + 270
                    If X1 = X2 Then angle = angle + 180
                    If X1 < X2 Then angle = angle + 90
                End If
                AngleArray(j, 1) = Math.Round(angle, 2)
                j = j + 1
            End If
        Next
    End Sub

    Private Sub SortDistanceArray()
        Dim i As Integer, temp1 As Integer, temp2 As Decimal
        For i = 0 To N - 3
            For j = i + 1 To N - 2
                If DistanceArray(i, 1) > DistanceArray(j, 1) Then
                    temp1 = DistanceArray(i, 0)
                    temp2 = DistanceArray(i, 1)
                    DistanceArray(i, 0) = DistanceArray(j, 0)
                    DistanceArray(i, 1) = DistanceArray(j, 1)
                    DistanceArray(j, 0) = temp1
                    DistanceArray(j, 1) = temp2
                End If
            Next
        Next
    End Sub

    Public Sub CalculateAngleRange(ByRef Angle As Decimal)
        ReDim NextDistance(3, 1)
        Dim tagNo As Decimal, tagangle As Decimal, found As Integer, tagindex As Integer
        tagindex = 0
        For i = 0 To N - 2  '--- Checking 4 NN
            tagNo = DistanceArray(i, 0)
            found = 0
            For j = 0 To N - 2
                If AngleArray(j, 0) = tagNo And Array.IndexOf(Path, tagNo) = -1 Then ' And tagNo <>
TxtDestinationTag.Text Then
                    tagangle = AngleArray(j, 1)
                    If tagangle > 0 Then found = 1
```

```
            Exit For
        End If
    Next
    If found = 1 Then
        NextDistance(tagindex, 0) = DistanceArray(i, 0)
        NextDistance(tagindex, 1) = Math.Abs(tagangle - DestAngle)
        tagindex = tagindex + 1
    End If
    If tagindex = 4 Then Exit For
  Next

  SortNextDistance()
End Sub

Private Sub SortNextDistance()
  Dim i As Integer, temp1 As Integer, temp2 As Decimal
  For i = 0 To 3
    For j = i + 1 To 2
      If NextDistance(i, 1) > NextDistance(j, 1) Then
        temp1 = NextDistance(i, 0)
        temp2 = NextDistance(i, 1)
        NextDistance(i, 0) = NextDistance(j, 0)
        NextDistance(i, 1) = NextDistance(j, 1)
        NextDistance(j, 0) = temp1
        NextDistance(j, 1) = temp2
      End If
    Next
  Next
End Sub
```