Map Compression for a RFID-Based Two-Dimensional Indoor Navigation System

Tsung-Chun Tsai

A dissertation submitted to Auckland University of Technology in partial fulfilment of the requirements for the degree of Master of Computer and Information Science (MCIS)

2008

School of Computing and Mathematical Sciences

Primary Supervisor: Dave Parry

ATTESTATION OF AUTHORSHIP

Tsung-Chun Tsai

2008

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I wish to express my gratitude to my supervisors, Dr. Dave Parry and Dr. Russel Pears, for their patience, support, guidance and advice throughout this research.

I would like to thank my wife Linda, my daughter Jasmine, my parents in and family in Taiwan, and my parents-in-law in China for their support, patience and love in the process of completing this work.

Sincere thanks are also extended to helpful friends, participants and all staff of the School of Computing and Mathematical Science for their kindness and encouragement during my studying period at the Auckland University of Technology.

ABSTRACT

Radio frequency identification (RFID) tags can be used to identify a location and some can store a limited amount of data. They have been proposed as a method of storing a distributed map of an area, where a central map is not practical because of infrastructure or security concerns.

In order to navigate between a set of RFID tags in a space, location information can be stored on the tags themselves. However, because of memory constraints, it is not practical to store copies of a complete map on every tag. Use of compression techniques is a way to solve the issue of the RFID memory space limitation. The purpose of this study is to implement a compression scheme including the wavelet compression technique on map compression for a RFID based 2-dimensional navigation system and evaluate its performance. Previous work related to wavelet compression technology and location representations were reviewed to explore the background of this study from prior research findings. The methodology of this study is design science study and its testing method is experimental testing. The compression techniques adopted were combined with a navigation scheme to study whether the technique could support movement between various tag layouts. A number of experiments were carried out and the efficiency of this technique in a test environment is calculated.

LIST OF TABLES

LIST OF FIGURES

Chapter 1: Introduction

1.1 Background

Radio-frequency identification (RFID) is a wireless system to identify persons or objects automatically via radio waves. At the moment, RFID systems are becoming a core technology and applied by many enterprises for the purpose of identifying persons or objects, providing better inventory tracking and handling, and enhancing the quality of information and business process in a supply chain. An indoor navigation system is one of the applications of RFID which has been commonly proposed e.g. to overcome the lack of ability for visually impaired people, or for robotic navigation. As stated by Kulyukin, Gharpure, Nicholson and Pavithran (2004), the critical barrier for most visually impaired people to improving their quality of life is the lack of ability to navigate in new spaces which limits them to access to buildings. For instance, Kulyukin et al apply RFID technology in robot-assisted indoor navigation which gives a warning for the users when they encounter barriers. However, the technology of RFID is not without problems in this regard, as one of the major issues is the lack of storage space of RFID tag (e.g. a common data storage space of a passive RFID tag is 100 bytes). Therefore, the major objective of my study is to study and address the issue of location storage using RFID by using compression technology in order to successfully apply RFID technology in an indoor navigation system. The basic philosophy behind the compression is the procedure of encoding data using fewer bits in order to reduce large amounts of information to some which can be stored in a tag.

1.2 Motivation

A RFID navigation system based movement between set points requires information about the relative location of these points. However, an actual, complete map of these points may not be practical. A complete map stored centrally may be difficult because of infrastructure issues. For example, a central map implies access to that map wherever the person navigating is located. While this may be possible where there is e.g. wireless access to a network, this access comes at a cost of increased infrastructure. There may also be security or privacy issues, or the arrangement of items in the space may be changing rapidly. One alternative is for the map to be stored as a distributed set of data on the RFID tags themselves – for example encoding the location of the tag and its neighbours. However, if a map is to be stored on the RFID tags, the limited amount of data space makes a complete map impossible to store. On the other hand, using compression techniques is the way to overcome the limit of the size of memory in RFID tags. The main area of interest is the way in which a system could be set up so that the parts of the map contained on each RFID tag looses resolution according to the locality - that is the map gets less precise depending how far away from the tag you are. As a result, the goal of my study is to study the research question: *"How can a space efficient method of data storage using distributed data be used for navigation?"* As mentioned above, I believe that the concept of an indoor RFID navigation is worth further investigation and has vast scope for future development.

Essentially, this work deals with the situation where two constraints apply:

1. A central map is not available to the navigating person
2. A central map, or at least precise location information about the tag –group being used, is available when the tags are being loaded with the navigation information.

Although this may seem somewhat odd, this situation could occur e.g. in the situation of wool bales being stored in a shed. As the bales are being dumped, their location is recorded and compressed relative location information stored onto the tags attached to the bales. When a user then wants to recover the bales they use a very simple tag reader and computing device to navigate between them. Similarly "private" maps, which accessible only to people with access to the RFID tag information can be created.

More generally, compression techniques for distributed map storage may be of interest for very large collections of objects, e.g. a set of RFID location markers that cover a whole town, where the tags are added to add detail to a new space without updating the entire map.

1.3 Organization of the dissertation

This study is organized into the following chapters:

Chapter 2 provides a literature review of challenge and approach of location based navigation, the location representation method which includes Cartesian coordinate grid, latitude and longitude, and spherical coordinate system, RFID technology, several compression techniques, including LZW, Huffman, fractal compression, MPEG and wavelet compression, and the relevance to location and navigation.

Chapter 3 contains the concept, discussion and design of the selection of methodology of this study which includes design science and experimental testing.

Chapter 4 presents the theoretical fundamental of the research, including distance, angle and navigation.

Chapter 5 presents two experiments which are distance and angle, and navigation as well as descriptions of the algorithms and schemes.

Finally, chapter 6 provides the conclusion, discussion, strengths and limitations of this study as well as the suggestions of future research directions.

Chapter 2: Literature Review

2.1 Introduction

In this section, the challenge and the approach of the location based navigation is firstly discussed and then the existing location representation methods are described which focuses on the concept of each location representation method. The RFID technology is discussed thirdly which includes a brief description, its approaches and advantages of location based navigation, and the issues. A brief description is given of compression techniques focused on LZW, Huffman, fractal compression, MPEG and wavelet compression as well as a description of the existing successful domain and the efficiency. In the comparison, all the techniques are compared through their strengths and weaknesses. Finally, the relevance to location and navigation will be discussed.

2.2 Challenge and approach of location based navigation

Current location models can be classified into geometric or symbolic models. In geometric model, it specifies a location as an n-dimension coordinate. In symbolic model, it is a logical location which uses real-world entities to describe the location space. Both of the models provide the location information to identify a specified place. Moreover, the location based navigation is more complex than describe a location. It provides a route which includes several locations for user to navigate from one point to the destination. A map, which is the most commonly component in a location based navigation system for guiding user. Bessa, Coelho and Chalmers (2004) mention the next generation of maps should provide a more realistic representation of our world, a high quality 3D representation for the location based navigation. However, the location navigation application which contains a high quality 3D location representation has several issues such as the size of the display, the accurate of location representation, the size of storage space for the map and the cost (Bessa et al., 2004). The pixel of the display has been increased and the size storage device has been decreased by the recent technology. As a result, the cost is an issue comes with the new technology which needs to be taken into consideration. For the accurate of location representation, Global Positioning System (GPS) seems like is the way to solve this issue, which provides an absolute location to the user. GPS systems are popular nowadays but it cannot be used in the indoor environment. From other researches, there are many different ways can be implemented in the indoor navigation system and even some of them are the hybrid systems. For instance, Hub, Diepstraten and Ertl (2003), and Gryazin, Krassi and Tuominen (2003) mention use of WLAN technology, Hub, Diepstraten, and Ertl (2005) mention use of Wi-Fi technology, Feldmann, Kyamakya, Zapater and Lue (2003) mention use of Bluetooth technology and so on. Nevertheless, numbers of these technologies are complex to implement in indoor navigation system because the required devices and the assistant technologies, as well as some of these technologies are too expensive to use. In the other hand, the location representation is the most important element for any of location based navigation system. Bessa et al. (2004) indicate that the location based navigation application require a well formed representation of spatial knowledge. As a result, the location representation method is described in the next section.

2.3 Location representation method

Location representation is a fundamental function in the navigation system. The users only can recognise their current position and go to the unknown destinations from this information. Therefore, the efficiency and compatibility of a navigation system relies on the capability of the location representation. As Keys-Mathews (1998) mentions the location can be presented as the absolute location or relative location. In the absolute location, the location can be presented as an estimated location or a street address. The most common application of absolute location is GPS which receives the precise location of user that is located and transmitted by satellites. On the other hand, Korkea-aho and Tang (2002) mention the relative location is a particular types of describing position which the location is represented relative to other object. As Keys-Mathews states "relative locations are described by landmarks, time, direction or distance from one place to another and may associate a particular place with another." The location is often presented by using a Cartesian coordinate grid, latitude and longitude, and spherical coordinate system (***Figure 1***). In a Cartesian coordinate grid, the ***Figure 2*** presents the concept of the Cartesian coordinate grid which its location data contains three numbers x, y, z that are used to describe a 3D location and the 2D location that only use x and y to present which is presented in ***Figure 3***. Because of dealing with the 2D case in this study, ***Figure 4*** presents distance and angle in a polar coordinate system which is a 2D coordinate system. In latitude and longitude, the location contains two numbers which are used for resenting the latitude and longitude of the global location. In the spherical coordinate system, it uses the similar concept as the Cartesian coordinate grid but it is used to describe the location on the spherical object.

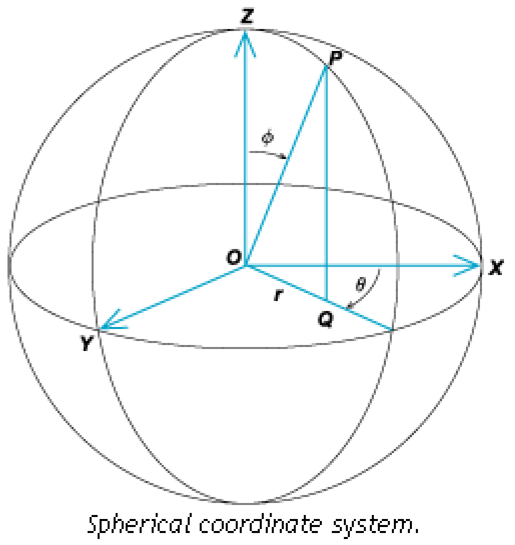


Figure 1 A spherical coordinate system (Answer.com, spherical coordinate system, n.d.)

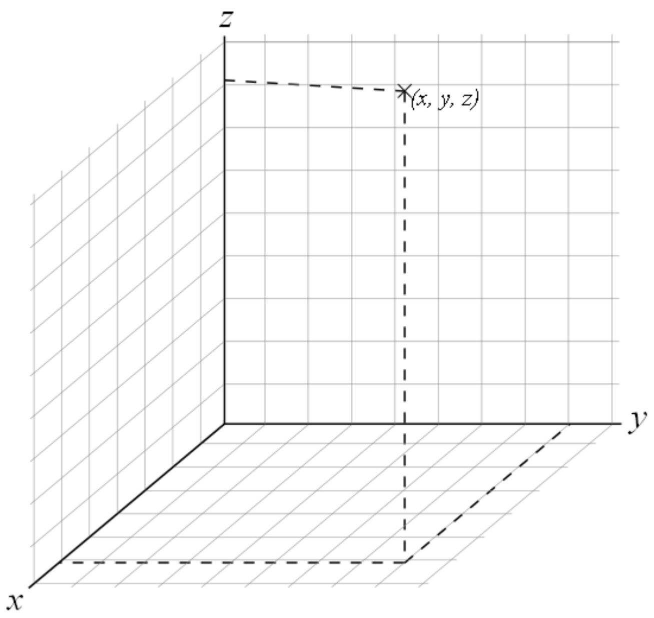


Figure 2 Cartesian coordinate grid (3-D) (Wikipedia, Cartesian with grid, 2008)

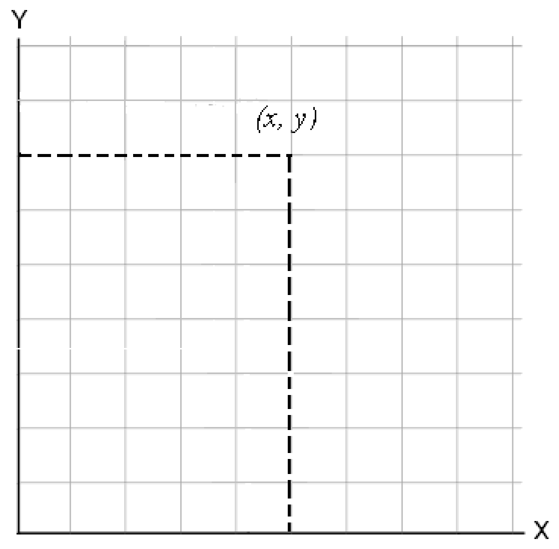


Figure 3 Cartesian coordinate grid (2-D)

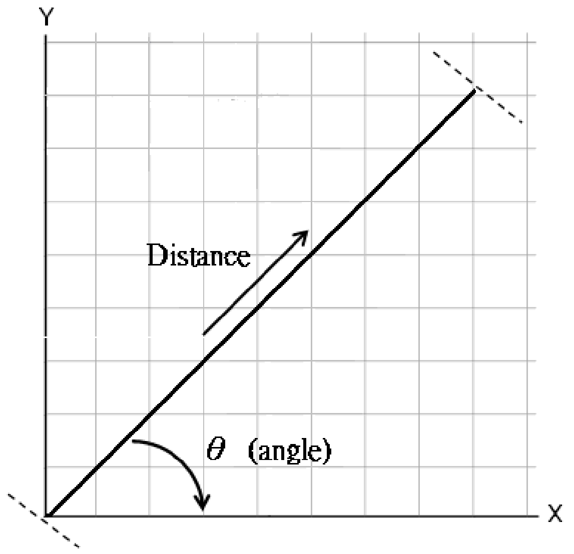


Figure 4 The representation of angle and distance in 2-dimensional system

Bessho, Kobayashi, Koshizuka and Sakamura (2007) mention GPS has been used in the recent years, and has already been made available on mobile phones. However, it has some limitations such as the accuracy and applicability. Furthermore, the need to have line-of-sight to the satellites of the GPS navigation system means it only can be used in outdoor environment. In this study, I am interested in an indoor navigation system.

Therefore, when the location representation method can only be used in the outdoor environment, this method is not useful for this study. Furthermore, the GPS uses latitude and longitude to represent its location. This method can be used to indentify the absolute location which the error can be controlled to less than 1-2 metres. However, latitude and longitude represent the global location which sometimes is less meaningful for presenting the indoor location, where walls and other obstructions are relatively small. In fact, maps of buildings may have errors relating to absolute coordinates at the same level as the error due to the GPS. Relative location, within a bounded space is more suited to indoor navigation, as higher resolution can be obtained with smaller data usage. As stated in Mehmood, Kulik and Tanin (2007), and Willis and Helal's (2004) papers, they both use grids to represent the location. In this method, the indoor environment is made up by a grid. As a result, it only requires x-coordinate and y-coordinate, and then the location can be represented. They mention the accuracy of the use of the grid depends on the dimensions of the grid. According to this, this method is simple and the designer can control the accuracy of the system as well as it performs like a latitude and longitude system for indoor navigation system. In Mehmood, Kulik, and Tanin's paper, the accuracy of their grid location representation method is high. However, from the ***Figure 5***, it can be seen that their experiment is only set up in an extremely small environment. As a result, the grid location representation method should be re-designed in order to suit this study and present the accurate location information to the users.

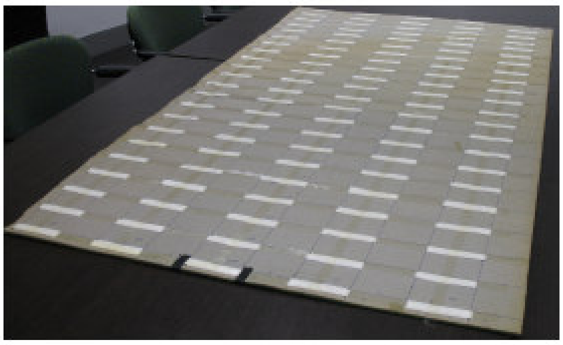


Figure 5 An example of the environment of experiment (Mehmood, Kulik, & Tanin, 2007)

2.4 RFID technology

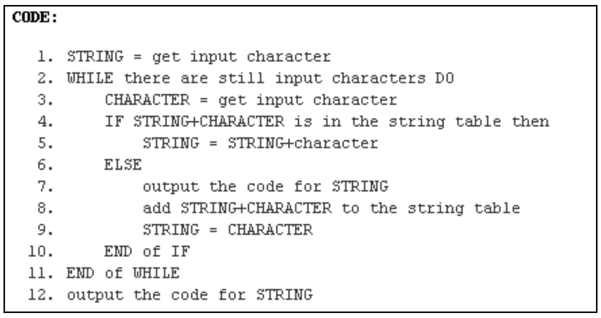
RFID is a wireless system to identify persons or objects automatically via the radio waves. An RFID system consists of a tag which is known as a transponder and a reader which is known as a transceiver. RFID features non-line-of-sight and non-contact readability (Zhou, Liu & Huang, 2007). The RFID reader can detect the existed tags within a certain range. As a result, in the research of Toshifumi (2005), and Kulyukin, Gharpure, Nicholson and Osborne (2006) both use passive RFID tags as landmarks to remind the mobile robot its position and determine next movement based on the ID information and a map of tags. There are two types of RFID tags which are active tags and passive tags. The cheapest passive tag nowadays costs less than 10 cents which can be used to reduce the cost of the system. As Renaudin, Yalak, Tomé and Merminod mention (2007) RFID technology has advantages in cost. According to above, RFID technology has several common advantages while it is applied to the location navigation system, such as low cost, quickly identifying and locating each reference object by retrieving the unique ID code, location information stored in the reference object using a transceiver and the line-of-sight is important for accurate distance measurement. However, RFID technology also contains a number of issues, such as the privacy issue, security issues and the size of storage issue. Because of a reader sends a query to a tag via the radio waves are invisible that is so difficult to identify which reader reads a particular tag. As Kim, Lee and Kim (2006) mention "Location threat is that individuals carrying unique tags can be monitored and their location revealed." As mention above, the passive tag is cost less than active tag. However, there is a limitation in the size of the storage of the passive tag. Compression is the reduction in size of data in order to save space or transmission time. As Moffat (1990), and MANZOOR and IJAZ (2008) mention compression technique is useful because it helps reduce the consumption of expensive resources. Therefore, the following section will be focused on the different compression techniques.

2.5 Compression technique

2.5.1 LZW

In 1984, T. A. Welch published Lempel-Ziv-Welch (LZW) which is a lossless data compression algorithm (Welch, 1984). It is an improvement of a dictionary coding algorithm - LZ-78. LZW is used for high-performance disk controllers in hardware. This algorithm has been widely used when it became the standard compression utility in UNIX systems. Furthermore, the format of GIF image implements the LZW as part of the compression algorithm. As a result, the LZW has been widely employed from this stage. Wu, Lonardi and Szpankowski (2006) mention other successful uses of LZW are it is used in the “V.42bis modem standard, TIFF images and PDF formatted documents.” Moreover, Nelson (1989) also states the two successful domains of the LZW which are the “file transfers over phone lines and archival storage.” Wu et al. indicate that this algorithm is unexpectedly simple. In LZW algorithm, the single codes are replaced by the character strings. However, this algorithm does not analyse these single codes and it only adds the new character string into a table of strings. As a result, the design of this algorithm is intended to be fast to run the whole compression process and it is not generally the optimal compression algorithm because it carries out only a limited analysis of the input data.

In the compression part of this algorithm, the LZW compression algorithm is shown in ***Figure 6***(Nelson, 1989).

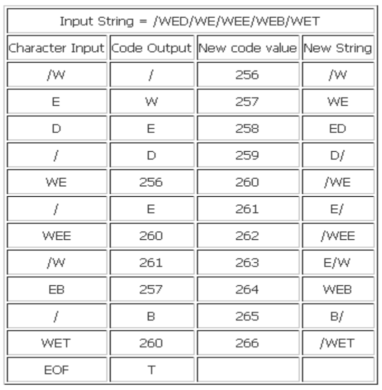


```
CODE:

STRING = get input character
WHILE there are still input characters DO
    CHARACTER = get input character
    IF STRING+CHARACTER is in the string table then
        STRING = STRING+character
    ELSE
        output the code for STRING
        add STRING+CHARACTER to the string table
        STRING = CHARACTER
    END of IF
END of WHILE
output the code for STRING
```

Figure 6 The compression algorithm of LZW (Nelson, 1989)

According to Nelson, a test of the algorithm presents that LZW always intents to produce codes for strings that are already known (in the string table already) and when a new code is produced each time, then a new string is also joined to the string table. The ***Figure 7*** shows the examination and its result of compression which is taken by Nelson.

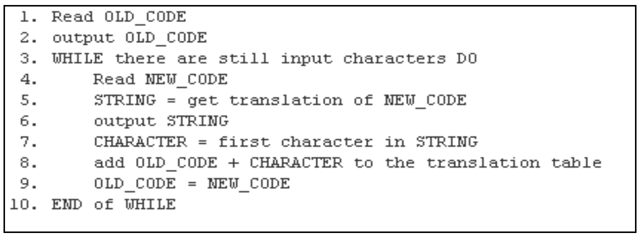


Input String = /WED/WE/WEE/WEB/WET			
Character Input	Code Output	New code value	New String
/W	/	256	/W
E	W	257	WE
D	E	258	ED
/	D	259	D/
WE	256	260	/WE
/	E	261	E/
WEE	260	262	/WEE
/W	261	263	E/W
EB	257	264	WEB
/	B	265	B/
WET	260	266	/WET
EOF	T		

Figure 7 The output of LZW compression (Nelson, 1989)

From this test, Nelson mentions when a new string starts to join to the string table each time and a code is produce, the string table fills up quickly. Furthermore, when the input is redundant, the output will be five code substitutions with seven characters. As a result, if we use 9 bit codes as the output, the 20 character input string is going to decrease to a 14.2 byte output string. However, Nelson also point out the limitation of the compression of LZW. For instance, the example of his test is chosen to express code replacement carefully. However, in the real world situation, the compression generally does not start while a sizable string table has not been built which means that the compression generally starts when at least one hundred bytes are read.

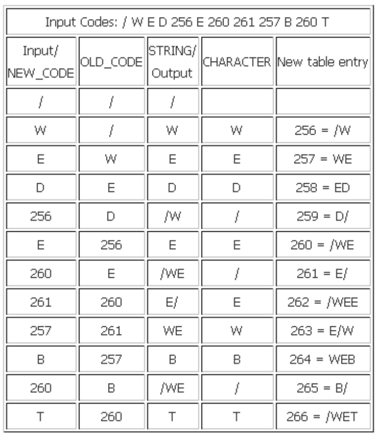
In the decompression part of this algorithm, the LZW decompression algorithm is shown in ***Figure 8*** (Nelson, 1989).



```
Read OLD_CODE
output OLD_CODE
WHILE there are still input characters DO
    Read NEW_CODE
    STRING = get translation of NEW_CODE
    output STRING
    CHARACTER = first character in STRING
    add OLD_CODE + CHARACTER to the translation table
    OLD_CODE = NEW_CODE
END of WHILE
```

Figure 8 The decompression algorithm of LZW (Nelson, 1989).

From the LZW decompression algorithm, it shows that the efficiency of this algorithm is the string table does not have to pass to the decompression code. The string table is used for decompression can be built at the same time that the compression table is build which the data is used is the input stream. Nelson indicates that this performance is possible because “the compression algorithm always outputs the string and character components of a code before it uses it in the output stream.” As a result, the compressed data is not encumbered with taking a huge string table. The ***Figure 9*** shows the examination and its result of decompression which is taken by Nelson. From the result, an important point is that the output string is identical to the input string from the LZW algorithm.



Input Codes: / W E D 256 E 260 261 257 B 260 T				
Input/ NEW_CODE	OLD_CODE	STRING/ Output	CHARACTER	New table entry
/	/	/		
W	/	W	W	256 = /W
E	W	E	E	257 = WE
D	E	D	D	258 = ED
256	D	/W	/	259 = D/
E	256	E	E	260 = /WE
260	E	/WE	/	261 = E/
261	260	E/	E	262 = /WEE
257	261	WE	W	263 = E/W
B	257	B	B	264 = WEB
260	B	/WE	/	265 = B/
T	260	T	T	266 = /WET

Figure 9 The output of LZW decompression (Nelson, 1989).

2.5.2 Huffman

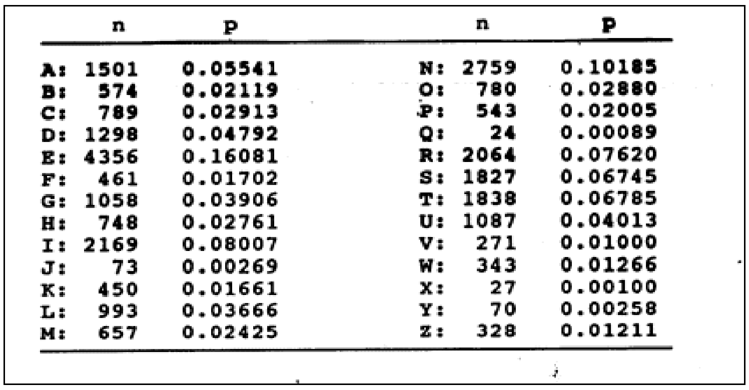
Chen, Zhang, Cao and Feng (2007) mention Huffman coding is one of the most important types of the lossless methods and its scheme was published by D. A. Huffman in 1952. Furthermore, the Huffman coding algorithm is widely used in the compression of image video and text. He, Zhang, Shen and Geng (2007) indicate that Huffman coding algorithm “has also been shown to be one of the most leading methods dealing with the data compression.” Furthermore, the Huffman coding has often been used as a support method to some other compression algorithms recently. For instance, DEFLATE algorithm and multimedia codecs such as JPEG, MP3, MPEG2, and MPEG (Chen, Pai, & Ruan, 2006).

In the Huffman coding algorithm process, the known data is passed twice. In the first pass, it gathers the statistic information of the passed on data and calculates to build a tree. In the second pass, all the codes are encoded and passed on correspondingly. Furthermore, a tree is build by using Huffman coding algorithm in the first pass. In the beginning of the first pass, a symbol and its possibility are held in each node. According to Chen et al.’s (2006) article, the implementation of the Huffman tree includes four stages which are shown in the ***Table 1***.

The stages of Huffman tree	
Stage 1	The two probable lowest free nodes or trees are joined to one tree.
Stage 2	It forms a parent node. The parent nod is allocated to the sum of these two child nodes assigns to this parent node.
Stage 3	One of the child nodes is designated as the path taken from the parent node when decoding the value 0 and other is set to the value 1.
Stage 4	A repetition of stages 1 to 4. It stops while only one tree left.

Table 1 The five stages of implementation of Huffman tree

The ***Figure 10*** and ***Figure 11*** show an example of Huffman coding compression which is based follow the steps of ***Table 1***. ***Figure 10*** is the file that is going to encode which only contain characters and ***Figure 11*** is the Huffman tree which is generated from this file. On the other hand, the decoding process of Huffman coding algorithm is very simple. First, begin with the first bit of the stream and one uses the continuing bits from the tree to decide the next step is go right or left in the decoding tree. A character is decoded while a leaf of a tree is reached. Therefore, the decoding character is placed into the output stream and the next bit of input stream is the first bit of the next character.



	n	p		n	p
A:	1501	0.05541	N:	2759	0.10185
B:	574	0.02119	O:	780	0.02880
C:	789	0.02913	P:	543	0.02005
D:	1298	0.04792	Q:	24	0.00089
E:	4356	0.16081	R:	2064	0.07620
F:	461	0.01702	S:	1827	0.06745
G:	1058	0.03906	T:	1838	0.06785
H:	748	0.02761	U:	1087	0.04013
I:	2169	0.08007	V:	271	0.01000
J:	73	0.00269	W:	343	0.01266
K:	450	0.01661	X:	27	0.00100
L:	993	0.03666	Y:	70	0.00258
M:	657	0.02425	Z:	328	0.01211

Figure 10 The input file (VIAS Encyclopedia, 2005)

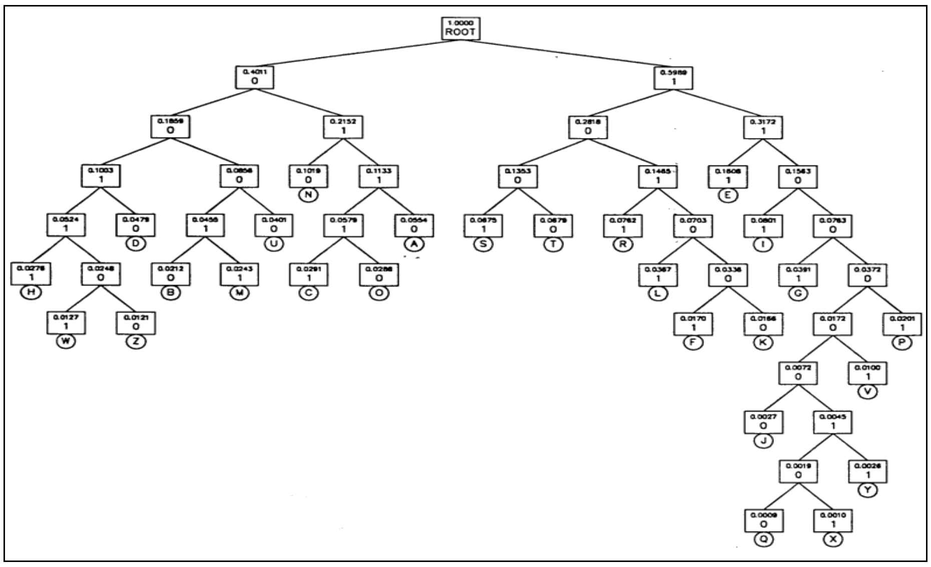


Figure 11 Huffman tree (VIAS Encyclopedia, 2005).

The effective compression ratio and simple algorithm are the main advantages of Huffman coding algorithm. He et al. (2007) mention the most important goal in some of the audio and video compression system applications is the real time performance. However, when these applications try to decode the Huffman encoded bit stream, it can be a key obstruction. In contrast, the characteristic Huffman code algorithm in general is formed by the structure of a binary tree and it gradually turns into a sparse tree because this binary tree grows from the root. The enormous memory space wasting is normally caused by this. As a result, when we use Huffman coding algorithm to discover a symbol, it might result in a long search process. On the other hand, VIAS Encyclopedia (2005) mentions “Assertions of the optimality of Huffman coding should be phrased carefully, because its optimality can sometimes accidentally be over-stated.” For instance, the mathematics coding does not involve an integer number of bits used to encode for each source symbol so it usually has preferable compression ability than Huffman coding algorithm. Furthermore, when the inputs are not allocated autonomously, LZW algorithm seems to be more efficient. According to this, the efficiency of Huffman coding relies mainly on encompassing a well estimation of the right possibility of the value of each input.

2.5.3 Fractal compression

Barnsley and Jacquin (1998) mention the fractal compression that sometimes is also called fractal image compression which is introduced in 1987. Initially, fractal compression is the production of the research iterated function systems. Davis (1998) mention the method of fractal compression technique is used to perform the compression is very different from other standard transform coder-based techniques. Moreover, the form of the fractal compression method images is very simple which a wide-sense fixed random procedure draws all the vectors. Therefore, as Davis states “they store images as quantized transform coefficients.” Furthermore, Jacquin (1992) indicates that in the block coders of fractal compression method, “the image redundancy can be exploited efficiently by the self-transformability on a blockwise basis”. Because of they store images as reduction maps of which the images are close immobile points. Furthermore, iteration of these maps is used to decode their immobile points Images. As mentioned by Poobal and Ravindran (2005), the Iterated Function Systems or Partitioned Iterated Function Systems can be used to perform fractal compression successfully.

In the efficiency, the common compression ratio of the fractal compression is up to 50:1. Furthermore, fractal compression method offers better quality in the high compression ratios. Sayood (2005) mentions fractal image compression offers same results as the DCT-based algorithms which includes JPEG. Fisher (1997) states in some form and some certain ratio, the fractal compression achieves a low compression times such as the ratio of fractal video compression is between 25:1 to 244:1 then the processing time will be around 2.4 to 66 sec per frame. This result is by comparing with fractal still image compression. Furthermore, by compared to the simple gray scale image, the efficiency of the fractal compression is increased with higher picture complicacy and depth of the colours. According to above, the successful domains of fractal compression are image compression and video compression, and it has numbers of advantages such as independent resolution, the rate of bit is high, and the speed of the decompression is also high. However, in Wang, Wu, He and Hintz's (2006) article mention "the greatest disadvantage of fractal compression method "is the high computational cost of the coding phase, which makes fractal coding cannot compete with other techniques."

From Poobal and Ravindran's (2005) article, they indicate the most common partition method is used in the fractal compression is quad-tree partitioning which uses block classification to decrease the amount of the comparisons in process. Furthermore, this method also support fractal compression achieves simple and fast capability and it performs on multiple levels of image resolution so it contains the strength of the multi-resolution decomposition. Poobal and Ravindran mention there are 8 encoding stages and 3 decoding stages in quad-tree partitioning which are shown below in ***Table 2*** and ***Table 3***:

In Encoding	
Stage 1	Tolerance factor T_{max}, is varied from 1, 2, 3…10…
Stage 2	Minimum tree depth m, maximum tree depth M, bits used for scaling factor and offset factor are fixed as 4, 6, 5 and 7 respectively.
Stage 3	Image is partitioned into four sub-nodes and is compared with domains from the domain pool D.
Stage 4	The pixels in the domain are averaged, in groups of four so that the domain is reduced to range size.
Stage 5	The root mean square (RMS) value between the transformed domain pixel values and the range pixel values is found out as, $\text{RMS} = \sqrt{\min E(R,D)/n}$, where n is the number of pixels in the range R.
Stage 6	If the RMS ≥ T_{max} and depth ≤ M, repeat the steps 2 to 4.
Stage 7	If the RMS ≤ T_{max} , the domain is mapped as W_i.
Stage 8	The collection of all such maps is given as $W = \bigcup w_i$ where W is the encoded image.

Table 2 Eight encoding stages of Fractal compression in quad-tree partitioning (Poobal and Ravindranm, 2005)

In Decoding (Decoding an image consists of iterating W from any initial image)	
Stage 1	For each range R_i, the domain D_i that maps is shrunk by two averaging non-overlapping groups of 2x2 pixels.
Stage 2	The shrunken domain pixel values are then multiplied by s_i added to o_i and placed in the location in the range determined by the orientation information.
Stage 3	This iteration is done until the fixed point is approximated by maximum number of iterations.

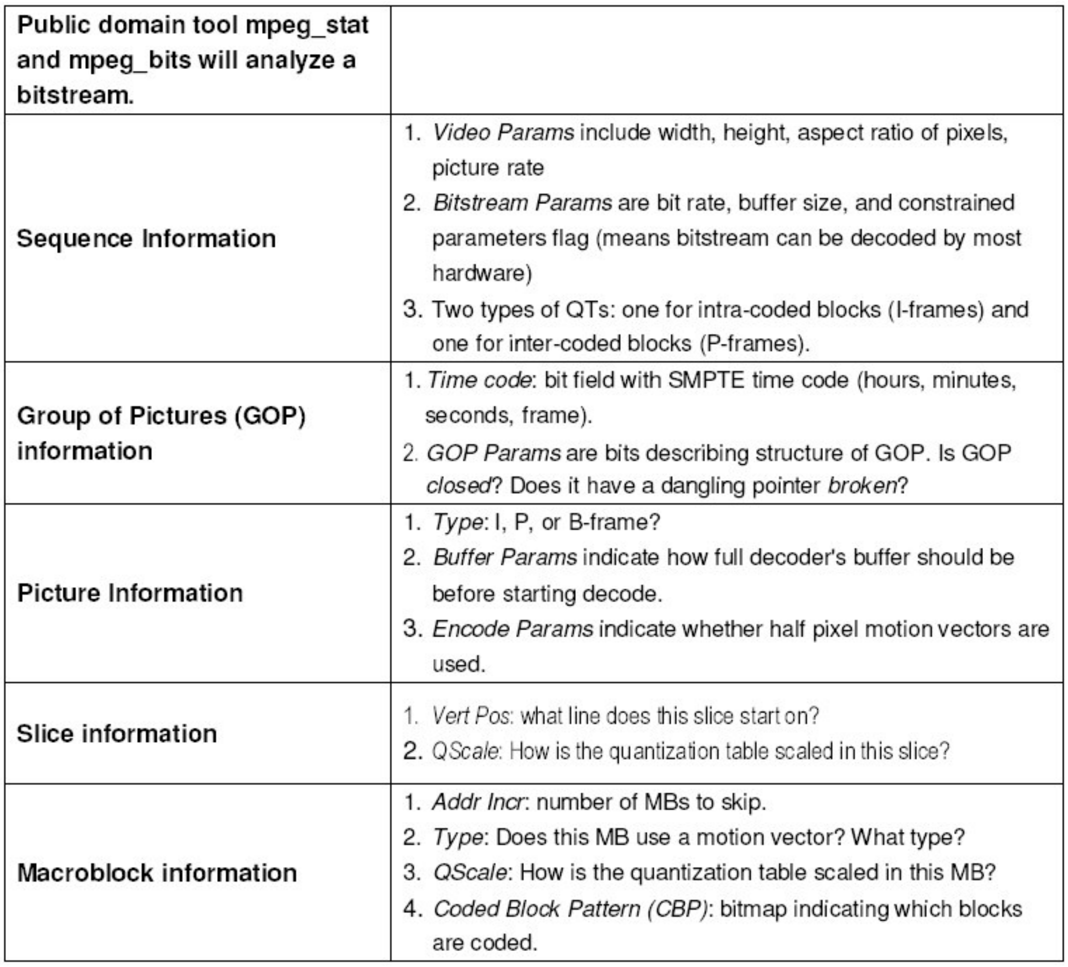
Table 3 Three decoding stages of Fractal compression in quad-tree partitioning (Poobal and Ravindranm, 2005)

2.5.4 MPEG

MPEG stands for Moving Picture Expert Group, which is a compression standard for video processing and it is widely used in multimedia applications such as VCD and DVD. Jiang, Xia, and Xiao (2006) mention the lossy compression techniques lead the researches of video compression because there are two main issues which are the accessible bandwidth for the most presently practical applications and the acceptance in the human optical system. Furthermore, it is important to introduce a particular level of distortion when using the lossy compression techniques so the best potential efficiency of the compression can be achieved. The successive standardisation activities of MPEG are introduced to represent these abilities.

One of the current implantation of the MPEG compression standard is employed for DVD which the average data rate for it is 3.5 Mbps. By using this compression method, the changeable bit-rate procedure can assign more bits for complicated scenes which contain numbers of actions, when the bits are minimized in static scenes. There are numbers of MPEG compression standards have been introduced to public such as MPEG-2, MPEG-4, MPEG-7, MPEG-21, and so on. Furthermore, the successful domain of these standards is in the multimedia platforms. However, MPEG-4 moves away from traditional successful domain of MPEG compression which moves to the implementation of software image construct descriptors and its goal is to make the bit-rates in the extremely low range. As Sayood (2005) mentions the MPEG Layer III algorithm has been extremely successful.

However, it still has some weakness because the limitation has been designed at the start point. The principle limitation is the requirements of this algorithm should be backward compatible. However, this limitation also brings some advantage to the MPEG compression method such as it has a major development in the capability of the hardware. As a result, the Advantage Audio Coding has been approved as a better quality multi-channel choice to the backward compatible MPEG compression. The summary of the MPEG video bit-stream is showing in the ***Table 4*** and ***Figure 12***.



Public domain tool mpeg_stat and mpeg_bits will analyze a bitstream.	
Sequence Information	1. *Video Params* include width, height, aspect ratio of pixels, picture rate 2. *Bitstream Params* are bit rate, buffer size, and constrained parameters flag (means bitstream can be decoded by most hardware) 3. Two types of QTs: one for intra-coded blocks (I-frames) and one for inter-coded blocks (P-frames).
Group of Pictures (GOP) information	1. *Time code*: bit field with SMPTE time code (hours, minutes, seconds, frame). 2. *GOP Params* are bits describing structure of GOP. Is GOP *closed*? Does it have a dangling pointer *broken*?
Picture Information	1. *Type*: I, P, or B-frame? 2. *Buffer Params* indicate how full decoder's buffer should be before starting decode. 3. *Encode Params* indicate whether half pixel motion vectors are used.
Slice information	1. *Vert Pos*: what line does this slice start on? 2. *QScale*: How is the quantization table scaled in this slice?
Macroblock information	1. *Addr Incr*: number of MBs to skip. 2. *Type*: Does this MB use a motion vector? What type? 3. *QScale*: How is the quantization table scaled in this MB? 4. *Coded Block Pattern (CBP)*: bitmap indicating which blocks are coded.

Table 4 The summary of the MPEG video bit-stream (Marshall, 2001)

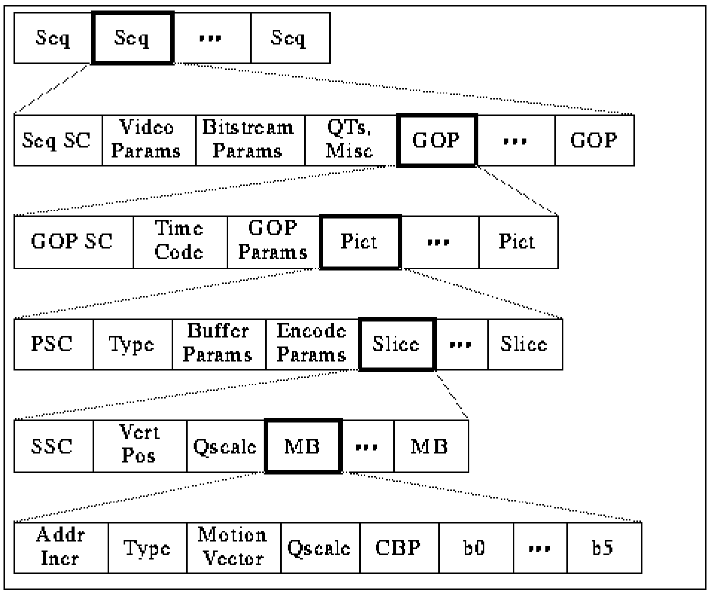


Figure 12 The MPEG video bit-stream (Marshall, 2001)

2.5.5 Wavelet compression

Wavelets are a class of functions used to localize a given function in both space and scaling. A family of wavelets can be created from a function $\psi(x)$. Sometimes it is known as a mother wavelet which is restricted in and defined over a finite interval. Daughter wavelets $\psi^{ab}(x)$ are then formed by translation (*b*) and contraction (*a*). Weisstein (n.d.) describes this by indicating its three steps which is showing in ***Table 5***.

STEP 1	$\psi^{ab}(x) = \|a\|^{-1/2}\,\psi\left(\frac{x-b}{a}\right).$
STEP 2	$W_\psi(f)(a,b) = \frac{1}{\sqrt{a}}\int_{-\infty}^{\infty} f(t)\,\psi\left(\frac{t-b}{a}\right)dt,$
STEP 3	CALDERÓN'S FORMULA GIVES: $f(x) = C_\psi \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\langle f, \psi^{ab}\rangle\,\psi^{ab}(x)\,a^{-2}\,da\,db.$

Table 5 The three steps of wavelet compression (Weisstein, n.d.)

There are a large number of wavelet transforms which the most common ones are: Discrete wavelet transform (DWT), Wavelet packet decomposition (WPD), Continuous wavelet transform (CWT), Fast wavelet transform (FWT), Stationary wavelet transform (SWT), and Lifting scheme and each suitable for different applications. On the other hand, wavelet is a useful mathematical tool which is efficient and theoretically stable for hierarchically decomposing functions. In addition, the wavelet decomposition of a function is composed of a coarse overall close with detail coefficients that affect the function at a range of scales. The wavelet has exceptional compact energy and de-correlation properties, which can be used to effectively produce concise representations that exploit the structure of data (Minos & Phillip, 2004). Moreover, wavelet transforms can commonly be computed in linear time. In the compression techniques, there are several ways to identify a wavelet compression which includes from the scaling function, scaling filter and wavelet function. By indentify from the scaling function and wavelet function, as Addison (2002) mentions the wavelet function "is in effect a band-pass filter and scaling it for each level halves its bandwidth." Furthermore, in the identification of wavelet filter, a low-pass finite impulse response filter scaling filter is the fundamental way to indentify the wavelet which the length is 2N and sum is 1. ***Figure 13*** shows the 2-D image wavelet transform.

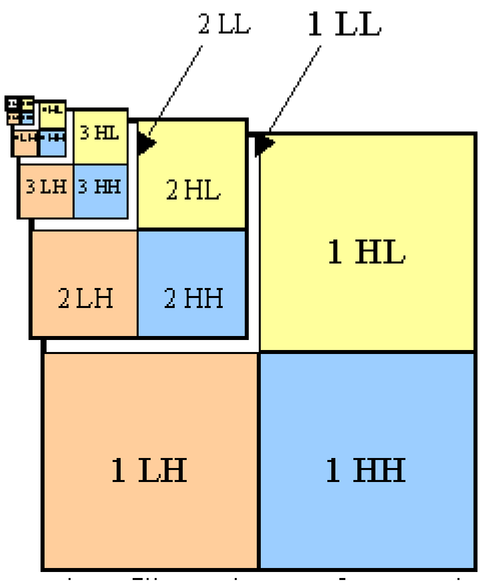


Figure 13 The 2-D image wavelet transform (Olivier, n.d.)

Wavelet transform is an iterative de-correlating process which decomposes a tile into a series of sub-bands. Each sub-band comprise tile information limited to a given frequency range. One level of wavelet decomposition allows creating 4 sub-bands from the low-pass sub-band obtained during the prior decomposition steps. From ***Figure 13***, “L” means result of low-pass filtering in a given direction which is horizontal or vertical and “H” means result of high-pass filtering in a given direction. The sub-bands 1HL, 1LH, 1HH, 1LL are the result of the wavelet decomposition applied on the complete tile and sub-bands 2HL, 2LH, 2HH, 2LL are the result of the wavelet decomposition applied on sub-band 1LL. This procedure classifies the same frequency range together which allows selectively weighting the quantization of these data. Each sub-band can go through separate quantization by a programmable factor for lossy compression. Bypassing the quantization yields lossless operation. Then, the consequential quantized sub-bands are divided into smaller blocks which are separately entropy encoded. This process is achieved by an adaptive Arithmetic Encoder which is a Modeller and an MQ-coder. The Modeller tests all bit planes of the present block which start from the most significant non-zero bit plane. It scans the present bit plane in a zigzag order with three progresses per plane. It calculates a context to the present bit in each progress. This context reflects the principal value of the neighbouring bits. Finally, the adaptive Arithmetic Encoder encodes each scanned bit using a possibility value obtained from the connected context. The Arithmetic Encoder brings up to date its possibility tables after each bit encoding. The Modeller also calculates compression metrics reflecting the image distortion occupied by reproducing the block only with its recent encoded part.

Minos and Phillip (2004) mention the success of wavelet compression technique in reducing large amounts of data has been demonstrated by numbers of the recent works. They also indicate that the wavelet compression technique also with an extensive successful history of signal and image applications. For instance, its successful applications in the still images are JPEG 2000, ECW, MrSID, SPIHT, Embedded Zerotrees of Wavelet transforms and Progressive Graphics File, and successful applications in the video are Dirac, Pixlet, Tarkin, Rududu, Bink Video, and Motion Compensated Temporal Filtering. Furthermore, Calderbank, Daubechies, Sweldens and Yeo (1998) state “Invertible wavelet transforms that map integers to integers have important applications in lossless coding”. As a result, the wavelet compression seems like has the strength of the resolution handling on the integers. As Luo, Li, Li, Zhuang and Zhang (2001) indicate the advantage of the wavelet transform is a lower resolution signal is generated with each level of wavelet decomposition and the goal of the wavelet compression technique is to store data in as little space as possible in a file. However, there is a major issue in the existing wavelet compression techniques is the quality of each compressing result from the same data are very different from these techniques, even for the same queries on the same values in different parts of the data.

The wavelet compression can be either lossless or lossy and it is only considered in the form of lossy compression when the certain loss of quality is accepted. As Ramaswamy, Namuduri, and Ranganathan (1996) mention the quality of the compression and decompression from the wavelet compression technique is perfect in the form of lossless. However, the compression result of all the types of data is not all good by using wavelet compression. For instance, Marks (2000) indicates that the momentary signal characteristics are good to be compressed by using wavelet compression but the periodic, smooth, signals are better to be compressed by using other algorithms.

2.5.6 Comparison

This section presents the comparison of all the compression techniques (from section 2.5.1 to 2.5.5). ***Table 6*** represents the type of compression technique, successful domain, strength and weakness of each compression techniques which is used to do the comparison with the requirement and contents of this study. The detailed comparison will be showed in the section 2.6.

	Type of Compression	Successful Domain	Strength	Weakness
LZW	Lossless	● Image. ● Pdf formatted documents. ● File transfers over phone lines. ● Archival storage.	● Fast run.	● Limited analysis of input data.
Huffman	Lossless	● Support some other compression algorithms. ● Image video. ● Text.	● The effective compression ratio. ● Simple algorithm.	● Long search process ● When the inputs are not allocated autonomously, LZW algorithm is seems to be more efficient.
Fractal	Lossy	● Image. ● Video	● Simple ● The image redundancy can be exploited efficiently by the self-transformability on a blockwise basis ● Low compression and decompression times in ● higher picture complicacy and depth of the colours	● High computational cost of the coding phase ● Hard to compete with other techniques
MPEG	Lossy	● VCD ● DVD.	● Make the bit-rates in the extremely low range	● The principle limitation is that the requirements of this algorithm should be backward compatible
Wavelet	Lossless and lossy	● Still image. ● Video. ● File transfers over phone lines.	● Efficient of resolution handling on the integers. ● Lower resolution signal is generated with each level of wavelet decomposition. ● Store data in as little space as possible in a file.	● The quality of each compressing result from the same data are very different

Table 6 The comparison of the compression technologies in section 2.5.1 to 2.5.5

2.6 Relevance to location and navigation

In this study, the data file which is stored in each RFID tag is a text file and the data are all numbers. Furthermore, these numbers might contain large digits in order to show the location accurately. As a result, if we use lossy compression technique, some digits of the number will become inaccurate after decompression. According to the work described above, the compression technique is focused on lossless compression techniques. On the other hand, the memory space of RFID has a small limit. By comparing with LZW and Hoffman lossless compression techniques, the wavelet technique has an advantage in that it can be used to compress a data as small as possible. Moreover, a map which is stored in a RFID tag contains two data such as the neighbours' locations and the resolution of these neighbours. As shown in the ***Figure 14***, A, B, C, D, E, F, G and H are RFID tags which are set up in the same floor of a building. In ***Table 7***, it shows the content of the map in each tag.

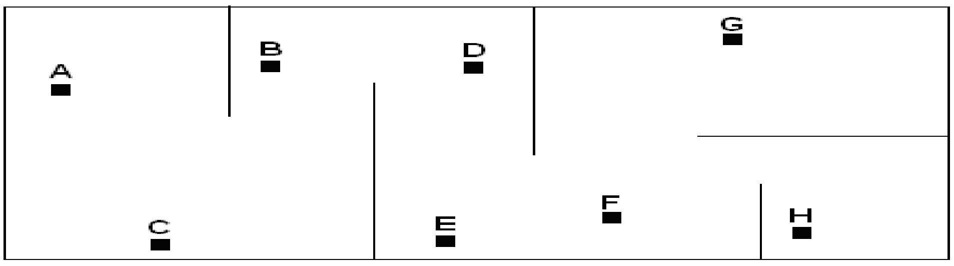


Figure 14 An example of tags in an indoor environment

	HIGH RESOLUTION	LOW RESOLUTION	LOWEST RESOLUTION
TAG A	B, C	D, E	F, G, H
TAG B	A, D	C, E	F, G, H
TAG C	A, E	B, F	D, G, H
TAG D	B, E, F	G	A, C, H
TAG E	F	B, D, H	A, C, G
TAG F	E, H	D, G	A, B, C
TAG G	D, F, H	E	A, B, C
TAG H	E, F	D, G	A, B , C

Table 7 The content of the map in each tag

For instance, in terms of data on A: B and C are high resolution, D and E are low resolution, and F, G and H are lowest resolution. In terms of data on F: E and H are high resolution, D and G are low resolution, and A, B and C are lowest resolution. ***Figure 15*** shows four possible major types of relationship between resolution and distance.

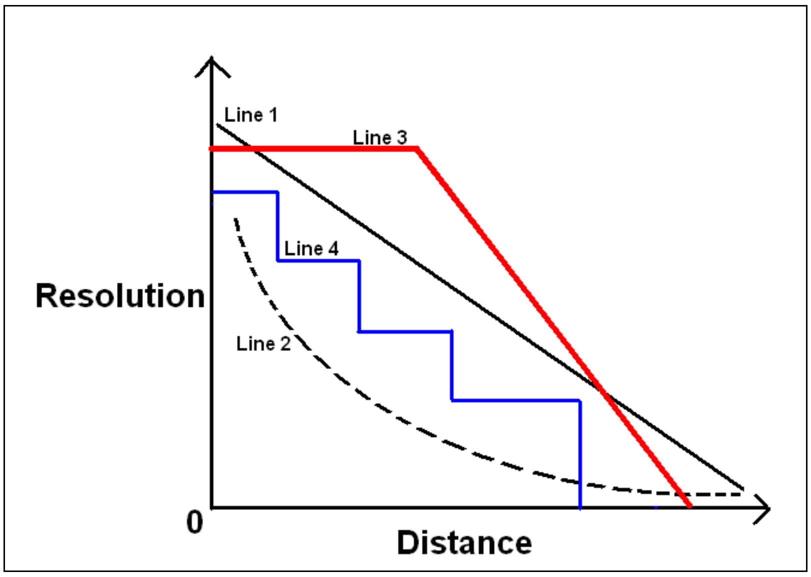


Figure 15 The relationship between resolution and distance

Line 1 represents the relationship between resolution and distance is linear which means when the distance is increased, the resolution is decreased. Line 2 represents the relationship between resolution and distance is nonlinear which means when the distance is increased, the resolution is decreased. Line 3 represents the situation where resolution does not change within a certain distances but the relationship changes to linear one above a certain threshold. Line 4 represents the relationship like downward steps which means the resolution undergoes step changes, at different distances. Later this technique will be used for angle representation

The reason why a changing resolution map is used and needed in this study is there is a limited amount of memory in RFID, so only a fixed number of neighbour's location can be stored in one tag

The wavelet approach as mentioned in the beginning of section 2.5.5, the filter of the wavelet compression classifies the same frequency range together. As a result, when the wavelet compression is used, the low and high resolution data are organized and efficiently compressed. For this reasons, because of the wavelet approaches, the wavelet compression technique was chosen for this study. First approach is the capable of resolution handling on the integers which is already described above. Second approach is it is to store data in as little space as possible in a file which is satisfactory to solve the limited amount of space issue of RFID tags. The third approach is the lower resolution signal is generated with each level of wavelet decomposition which means the low resolution map data is not ignored and not lose in any stage of wavelet compression. The fourth approach is a lossless compression technique if all coefficients are kept, thus it keeps all digits of each data so the map data is meaningful and accurate. The fifth approach is the processing time is fast which the map compression and decompression can be executed in real time.

Chapter 3: Methodology

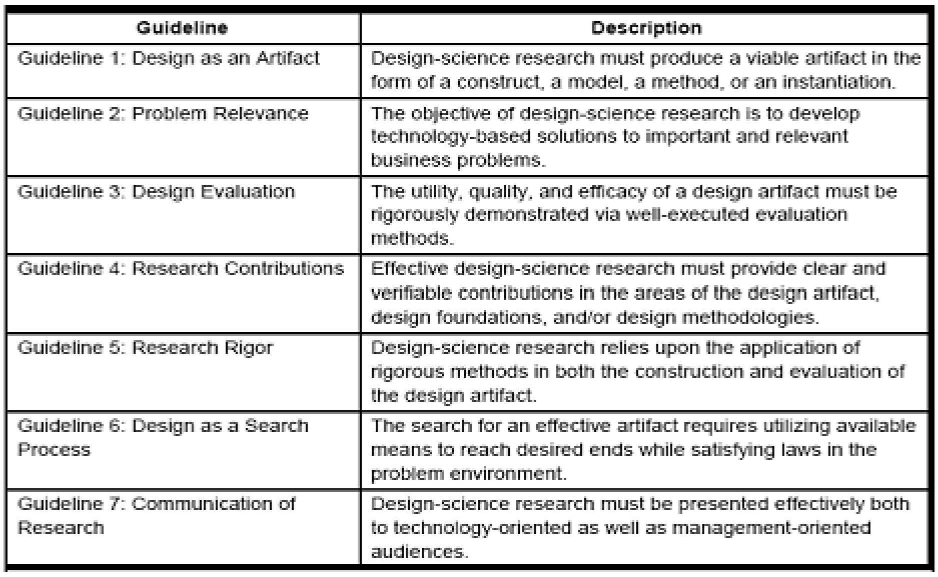
3.1 Selection of methodology

There are two methodologies are employed in this study which are design science and experimental study. Furthermore, the design science as the core methodology in this study and the experimental study is used as the testing methodology to generate the experimental result. Therefore, the concept, the discussion and comparison with other methodologies, and the design of each methodology are presented.

3.2 Design science

3.2.1 Concept and discussion

In design science paradigm, the understanding and knowledge of a problem field and its solution are accomplished in the construction and application of the designed artifact. As Nunamaker, Chen and Purdin (1990) mention design science is technology-oriented as applied research which applies knowledge to address practical problems. Furthermore, according to March and Smith (1995), design science attempts to serve human intentions by designing and creating things. Thus, it attempts to understand reality which opposes to social and natural sciences. March and Smith describe design science outputs as four types which are constructs, models, methods and implementations. Moreover, a construct represent a conceptualisation which is used to express problems within the area and to state their solutions. A model is a set of statements or suggestions describing relations between the constructs which specifies situations as problem and solution statements. A method is an algorithm or a guideline which is employed to execute a task. An implementation is an instantiated operation on constructs, models and methods. Furthermore, design science consists of two basic activities which are building and evaluation. ***Table 8*** summarises the seven guidelines which should be solved in some way for design science study to be complete (Hevner, March, Park & Ram, 2004).



Guideline	Description
Guideline 1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Guideline 2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Guideline 6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Guideline 7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Table 8 Design science research guidelines

Furthermore, the discussion is focused on the reasons of use of design science as the methodology in this study which is based on the comparison of design science, survey and case study methodology and it focuses on their ontology, epistemology, method, and axiology. Firstly, in ontology, Hevner et al. (2004) mention the ontology of design science is multiple and contextually situated alternative world-states, and socio-technology enabled. In ontology of survey, it is a single reality which is probabilistic and knowable. In ontology of case study, it is a socially constructed multiple realities. The ontology in this study is not from any reality and it is based situated alternative world-states. In epistemology, the design science is known through making, survey is that objective which is detached from researcher of truth, and case study is that subjective which the knowledge and values appear from the interaction of researchers and participants. As a result, this study is to build a RFID indoor navigation system and the epistemology is through this creation. In methodology, the design science is measure an artefact's impact on the composite system (Hevner et al., 2004), the survey is quantitative methodology, and case study is qualitative methodology. This study is a developmental study which is not only focuses on either quantitative or qualitative. In axiology, the value of design science is from the creation, understanding, and improvement which can be controlled, the value of survey is from truth which is a universal or a prediction, the value of case study is from a description of the understanding. In this study, the value is from the creative system. According to above discussion and the concept section, the use of the methodology in this study is design science.

3.2.2 Design

In design science, there are two fundamental parts of design which are creating and evaluating the artifact. In the creation of the artifact, the navigation system includes a map, and a programme. Firstly, a programme is used to compress and decompress the map will be written. This programme is based on the wavelet compression technique which the Java language is employed. Furthermore, a map which is stored in a RFID tags and it contains the distances of, the angles of and the labels of its neighbours. On the other hand, I use experimental study to do the test in order to evaluate the artifact. Therefore, this part is discussed in the next section 3.3.

3.3 Experimental testing

3.3.1 Concept and discussion

Experimental study supports the scientific method which is a procedure including, suggesting and testing hypotheses. The first step is to develop a theory. This theory can be from observations or an inference from prior research. Then, a hypothesis is composed which is a statement that can be examined. As Key (1997) states the experimental study is "an attempt by the researcher to maintain control over all factors that may affect the result of an experiment. In doing this, the researcher attempts to determine or predict what may occur". Furthermore, the general process of experiment study is one or more independent variables are conducted to decide their outcome on a dependent variable and a true experimental design needs an artificial environment. As Howell (1997) mentions the outcomes of an experimental study are recognized as the dependent variables which rely on the action of the independent variable. According to above, there are numbers of steps in this study which includes identify the problem, express the hypotheses and realize their consequences, construction, and perform the experiment. In the discussion, the reason of the use of experimental testing as the methodology in the evaluation part of design science is indicated. From Hevner et al.'s (2004) paper, the author states there are several methods which can be employed to evaluate the designed artifact such as case study, analytical and experimental. As a result, this part focuses on the comparison of these three methods. By using case study method, it is to study artifact in depth in a specific case. By using analytical method, it is to demonstrate inherent optimal properties of artifact or provide optimality bounds on artifact behaviour. In experimental testing method, it is a controlled experiment to study artifact in controlled environment for qualities such as usability, accuracy, applicability and so on. In the testing part of this study, the qualities of a RFID indoor navigation system are study and examined. In Milella, Vanadia, Cicirelli and Distante's (2007) paper, they use experimental testing to examine their design artifact which supports them to evaluate the accuracy and efficiency of the artifact. According to above, the experimental testing is a better testing method which is used in this study.

3.3.2 Design

In the first part of the experiment, the algorithms are used to reduce the storage space of distance and angle will be applied for a number of sets of random number tag and then the efficiency of reduction of the storage space will be test. In the second part of the experiment, the distance and angle data sets are used in the first experiment will be applied and the algorithms for the navigation will also be implemented. Then, the accuracy and will be tested.

Chapter 4: Theoretical Fundamental of the Research

In this section, the theoretical foundation of the research is discussed which involves choosing a method for representing relative location that can be stored on tags, and then used for navigation. The distance, angle and tag label are three sets of the data which contained in the tags. The distance and angle occupy most storage space in the tag. Therefore, the schemes of reducing the size of data for these two data sets are described respectively in section 4.1 and 4.2. Furthermore, the system should be available to navigate after the data have been stored in each tag correctly. As a result, the method and scheme of the navigation is also discussed in section 4.3.

The scheme used has the following characteristics:

1. Every tag contains location information about every other tag
2. The further away a tag is from the original tag, the lower the resolution of the distance and location
3. Relative location is stored as a distance and angle from the tag in question – effectively a vector.
4. Each tag therefore stores a different set of values.
5. The angle and distance to the nearest neighbour group are stored at the highest possible resolution
6. When navigating, the user can only move to one of the nearest neighbour group.
7. Only 2-D data is being stored.

4.1 Distance

In an indoor environment, a number of RFID tags are deployed and each one is used as a landmark to mark a specific location. Each of tag contains three sets of data which represent as the distance, angle and label of the neighbour tags respectively which associate with each other. Assuming there are 33 tags which have been placed which are denoted as S_1 to S_{33}. For each tag, it is able to calculate the distances between it to all other tags in order to obtain a list of the neighbour tags, then those distances is sorted in an increasing order. Furthermore, the process of this navigation system is: the user inputs the destination and current position tag first then the system will calculate the next point which is from one of the tags in the nearest neighbour group that towards to the destination for user to go to. This process repeats continually until the user reaches the destination. However, because of the limited memory space of the RFID tag, only a fixed amount of data can be stored in it. Therefore, it is a limited number of nearest neighbour (NN) tags that have complete information stored on each tag. As demonstrated in ***Figure 16***, the centre is the current position tag S_1 and the central two circles at full resolution in distance. The outer circle has the lower resolution by comparing with the central two circles.

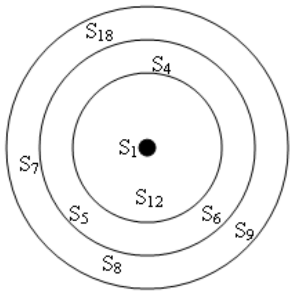


Figure 16 The neighbour tags of S1 in different distance resolutions (3 levels only)

The tag S_{12} is the tag in nearest neighbour group of S_1, the S_4 and S_6 are the tags in second close neighbour group of S_1, S_5 is the tag in nearest neighbour group of S_1, and so on. Because of there is only one point will be generated from the tags that with the full resolution for the next movement by the system each time. As a result, other tags only need the approximate value in distance and angle. According to this, all the tags in the NN group are with full resolution and other tags are in lower resolution which is compared with the NN group.

The advantages of wavelet compression have already been discussed in chapter 2. As a result, the wavelet compression method is used to reduce the size of distance data set. ***Figure 17*** represents the Haar Wavelet Coefficient Tree of S_1. The number 1 to 32 is the coefficients which are generated from the wavelet compression which can be used to reconstruct the data of all the tags. For instance, if the data of S_{12} and S_4 need to be reconstructed, the coefficient 1, 2, 3, 5, 9, and 17 is needed. If the data of S_6 and S_5 need to be reconstructed, the coefficient 1, 2, 3, 5, 9, and 18 is needed. At the bottom of the tree, the reconstructive values of all tags are shown from the highest resolution to the lowest resolution because the original list of neighbour tags are compressed in the same way. As mentioned previously, the data of the low resolution ones can be at a lower resolution, thus only the certain amount of coefficients is need to be stored. The coefficient 1, 2, 3, 4, 5, 6, 9, 17, 18, 19 and 20 will be stored. Therefore, the distance of the tags in the NN group (S_{12}, S_4, S_6 and S_5) can be exactly reconstructed from those coefficients. However, for the tags in the second close neighbour group (S_{18}, S_7, S_9 and S_8), the coefficient 19 and 20 have not been stored which means there is one missing value in the calculation for these tags. Therefore, the reconstructive data of S_{18}, S_7, S_9 and S_8 is fuzzier than the NN group. Furthermore, the distance of the tags in the third close neighbour group (S_2, S_{11}, S_3, S_{17}, S_{19}, S_{33}, S_{25} and S_{24}) can be reconstructed from these 10 coefficients as well. However, the coefficient 11, 12, 21, 22, 23 and 24 have not been stored in the tag which means there are two missing values in the calculation for these tags, so the reconstructive data of S_2, S_{11}, S_3, S_{17}, S_{19}, S_{33}, S_{25} and S_{24} is fuzzier than the second close neighbour group. For the tags in the fourth close neighbour group (S_{32}, S_{10}, S_{23}, S_{13}, S_{22}, S_{21}, S_{28}, S_{16}, S_{15}, S_{14}, S_{29}, S_{30}, S_{20}, S_{31}, S_{27} and S_{26}), the coefficient 7, 8, 13, 14, 15, 16, 25, 26, 27, 28, 29, 30, 31 and 32 have not been stored which means there are three missing values in the calculation for these tags. Therefore, the reconstructive data of the tags in fourth close neighbour group is fuzzier than the third close neighbour group. According to above, once all the coefficients for the NN group are stored, only one coefficient is required for each of other groups. As a result, the wavelet compression can be used to solve the memory space issue and it also can be used to achieve another goal which is that the father away a tag is from any other tag, the lower the distance resolution of the distant tag, stored on the original tag.

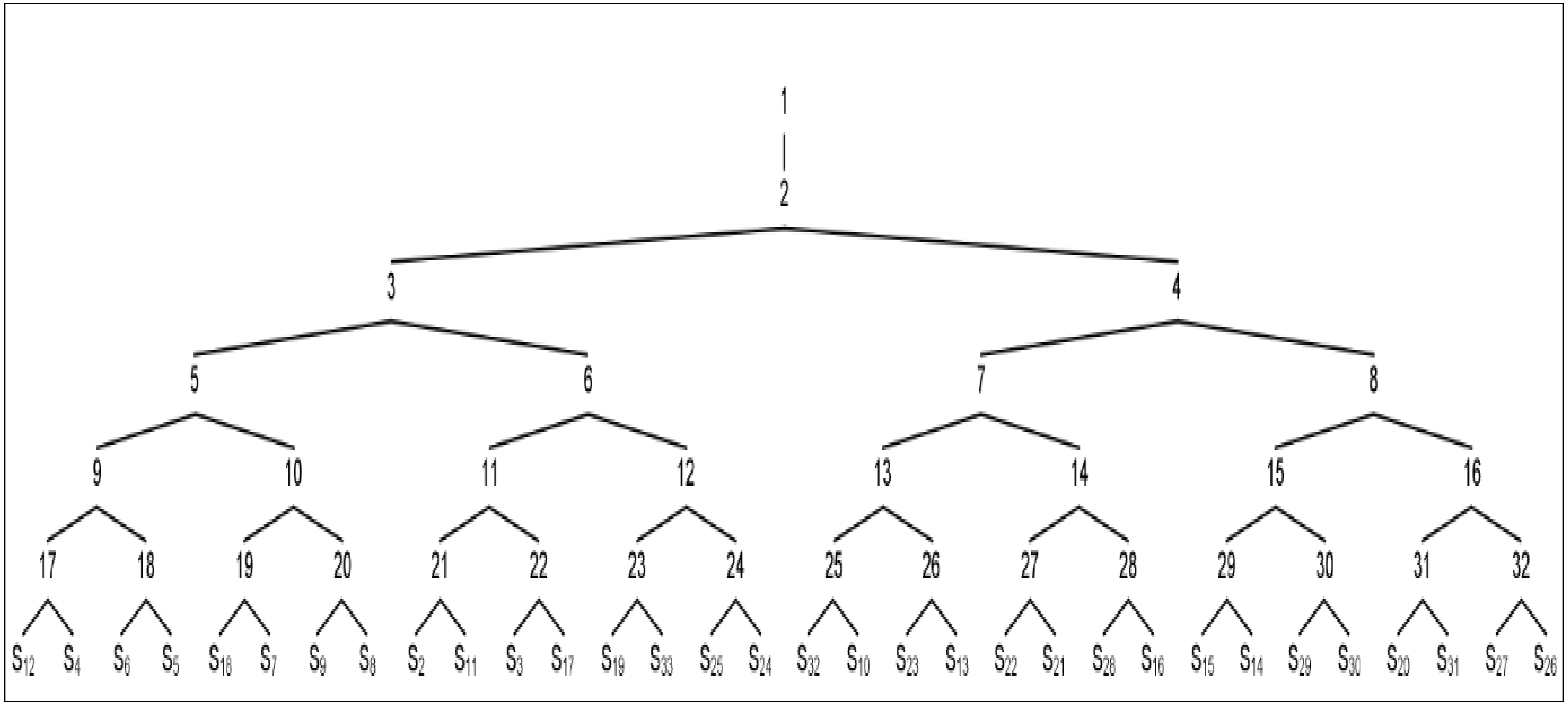


Figure 17 Haar Wavelet Coefficient Tree of S1

4.2 Angle

The wavelet has been used for reducing the data storage space of distance which is discussed in previous section 4.1. The angles of the neighbour tags are another large amount of data which is stored in each tag. As a result, the angle can guide the user move toward the correct direction. In order to reduce the use of the storing space, the amount of the data and the size of each single value should be reduced. Is the wavelet compression technique suitable for this data set? For distance data set, the data has already been sorted in distance order. Therefore, it is achievable that the distance resolution is getting fuzzier and fuzzier for the reconstructive data because one less from the previous level of the coefficients for each level of resolution is stored. However, for the angle data set, the data cannot be sorted in such a meaningful way, thus the order of the data will not match the reconstructive values of distance data. Moreover, while the wavelet compression algorithm is applied to the un-sorted angle data set as used on the distance data set, the result will be different.

Figure 18 represents the part of the Haar Wavelet Compression Tree for the NN group (A and B) and the second close neighbour group (C and D).

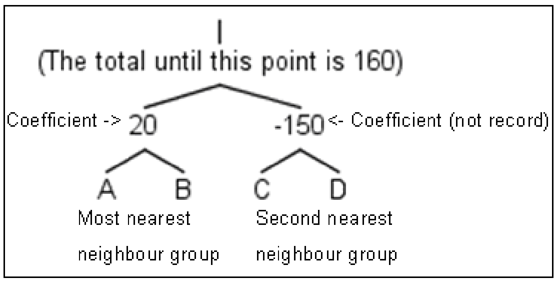

(The total until this point is 160)
Coefficient -> 20
-150 <- Coefficient (not record)
A
B
C
D
Most nearest neighbour group
Second nearest neighbour group

Figure 18 Haar Wavelet Coefficient Tree of the first two neighbour groups

Through the calculation for the second close neighbour group, it can be seen that the total number of the coefficients is one less from the nearest neighbour group. Assume that the two original angles are 10 and 310, the coefficient which is not record is -150, and the storing value for this group is 160. As a result, the original calculation for the first number is 160 + (-150) and 10, and the second number is 160 - (-150). However, by comparing 160 with the original numbers of this group (10 and 310), the meaning of the angles have the big different. According to above, the reconstructive value might not get fuzzier and fuzzier (level by level) in this case, and the ratio of fuzziness is only depends on the value of each pair of numbers. Therefore, another method should be considered and applied to the angle data set.

The new compression technique which is suitable for using in angle data has been designed which has been given a name “quadrant compression technique” in this study. ***Figure 19*** represents the neighbour tags of S_1 (the first 4 level of angle resolution only) and the angle ranges of each level.

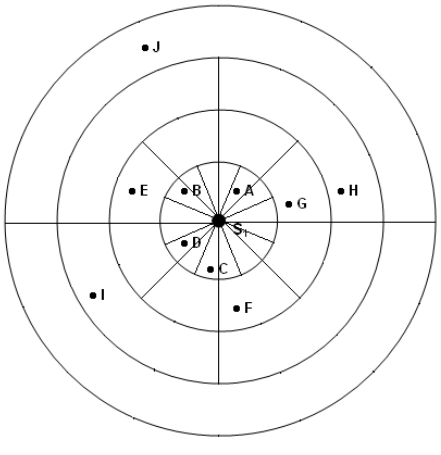


Figure 19 The neighbour tags of S1 (4 different resolution levels only) and the angle ranges of each level

Tag A, B, C and D are the tags of the nearest neighbour group, E, F, and G are the tags of the second close neighbour group, H and I are the tags of the third close neighbour group, and J is the tags of the fourth close neighbour group. The angle of each tag is represented by a binary number (e.g. 00100111). In order to express the angle with the different resolution from high to low for the neighbour tags, each level group of tags use the bits which one bit less than previous group. For instance, the angles of the tags of the nearest neighbour group are represented with the highest resolution which use the most bits to present the angle, the tags of the second close neighbour group use the bits which is one bit less than the most nearest group to present the angle, the tags of the third close neighbour group use the bits which is one bit less than the second nearest group to present the angle, and so on. Furthermore, as demonstrated in ***Figure 19***, the 4 bits binary number is applied in the nearest neighbour group, so there are $2^4 = 16$ angle ranges which are $0^\circ – 22.5^\circ$, $22.5^\circ – 45^\circ$, $45^\circ – 67.5^\circ$, $67.5^\circ – 90^\circ$, 90° - 112.5°, $112.5^\circ – 135^\circ$, $135^\circ – 157.5^\circ$, $157.5^\circ – 180^\circ$, $180^\circ – 202.5^\circ$, $202.5^\circ – 225^\circ$, $225^\circ – 247.5^\circ$, $247.5^\circ – 270^\circ$, $270^\circ – 292.5^\circ$, $292.5^\circ – 315^\circ$, $315^\circ – 337.5^\circ$ and $337.5^\circ – 360^\circ$ have been used to represent the directions in this group. The number of bits of the current group is always less than previous one, so the number of angle ranges of the current group is always half of the previous group (divide by 2^1). For instance, in ***Figure 19***, the number of angle range from nearest neighbour group to the fourth close neighbour group is 16, 8, 4 and 2. As a result, the resolution of the angle is getting fuzzier and fuzzier from nearest neighbour to the farthest neighbour, as well as the storing space has been reduced. Thus, it corresponds to the original concept.

4.3 Navigation

The schemes for reducing data storage for the data sets of distance and angles are discussed in the section 4.1 and 4.2. As a result, the navigation system is ready to provide the navigational information and to be used by the users after the RFID tags are deployed and the related data files are stored. Therefore, the navigation scheme will be described in this section. There are three steps in the whole processing of the navigation. Before start to navigate, the user needs to stop at any one of the existing tags and choose a destination tag from the list of current position tag. In the first step, the system calculates the mean of the angle range of the destination tag. In the second step, the system calculates and finds the next target tag which is in the highest angle range resolution group that is closest in angle to the destination. In the third step, after the next target tag is found, the user will move to that tag. The user repeats the second and third step until the system shows the destination is the next tag to move to (the destination is in the highest angle range resolution group of current position tag). As a result, the user only need one more move from the end of third step, and then the user reaches the destination.

Figure 20 presents an example of twenty-six tags (Tag A – Tag Z) that are deployed in an area. As a result, there are twenty-five stored distances, angles and labels in each tag.

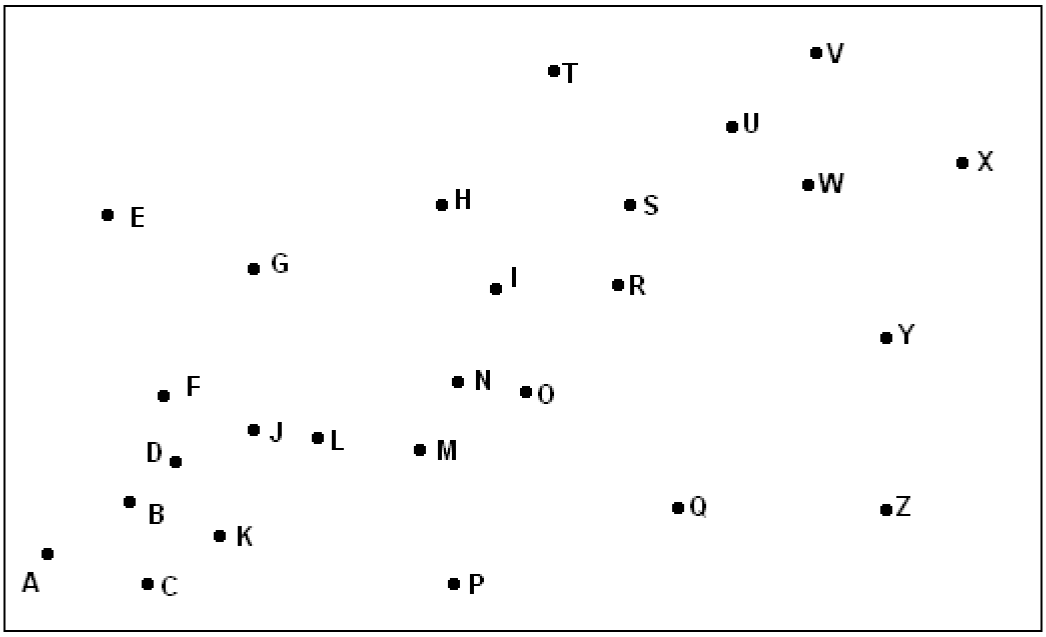


Figure 20 An example of twenty-six tags are deployed in an area: twenty-five stored distances, angles and labels in each tag

In the following paragraph, the steps of navigation will be demonstrated by using ***Figure 20***. It assumes that the each tag has three NN tags, the start point is Tag B and the destination point is Tag X. ***Figure 21*** presents the NN tags in the highest angle range resolution group for Tag B are D, K and C, and the uncertainty of angle of X, D, K and C. The angle range for X, D, K and C are $\angle X_{LOW}$ - $\angle X_{HIGH}$, $\angle D_{LOW}$ - $\angle D_{HIGH}$, $\angle K_{LOW}$ - $\angle K_{HIGH}$ and $\angle C_{LOW}$ - $\angle C_{HIGH}$. In order to find the next point that is going to move to, the tag in the NN group that is closest in angle to the destination tag should be identified. Furthermore, the angle data are the uncertainty of angle because only the range of angle is displayed which is converted from a binary number. For instance, the binary number 11100011 means the angle range is 319.22° – 320.62°, 0110 means the angle range is 135.00° – 157.50°, and 0 means the angle range is 0.00° – 180.00°. The angle binary numbers of all the neighbour tags for each tag have already been calculated by CalAngleBinary.java (appendix 1) after all the tags have been deployed which is demonstrated in the previous angle experiment section. The equation – "($\angle Range_{LOW}$ + $\angle Range_{HIGH}$) / 2" is used to calculate the mean of each angle range in order to calculate the angle between each NN tag in the highest angle range resolution group and the destination. As a result, there are three angles need to be compared in ***Figure 21*** which are $\angle D_{MEAN}BX_{MEAN}$, $\angle K_{MEAN}BX_{MEAN}$ and $\angle C_{MEAN}BX_{MEAN}$. After the comparison, the angles from smallest to largest are $\angle D_{MEAN}BX_{MEAN}$, $\angle K_{MEAN}BX_{MEAN}$ and $\angle C_{MEAN}BX_{MEAN}$. Therefore, the tag D is the tag in the highest angle range resolution group that is closest in angle to the destination tag X which means D is the next tag to move to.

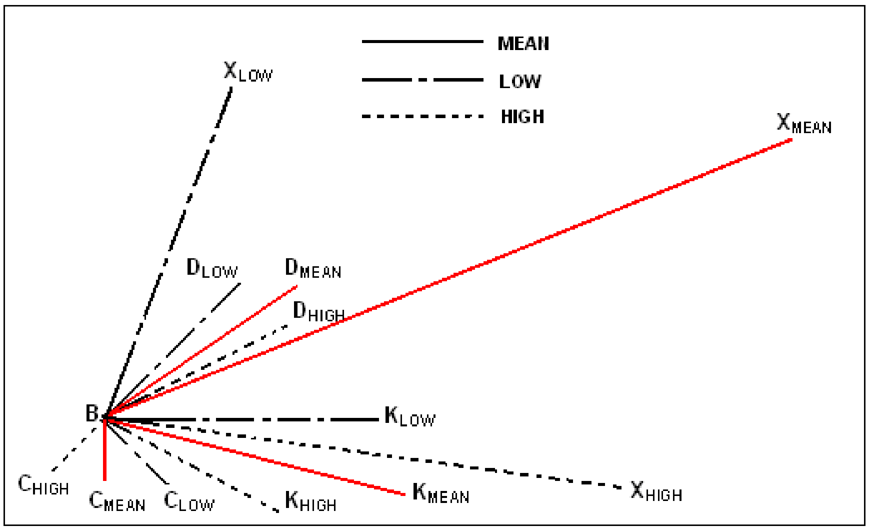


Figure 21 The angle range of each NN tag for Tag B

After moving to Tag D, the process is used to find the next tag to move to in Tag B should be repeated in Tag D. However, there are two key points need to be checked before run the process which are whether the destination is in the highest angle range resolution group of the current position tag and whether any tag in the highest angle range resolution group of the current position tag has already been in the route before. ***Figure 22*** presents the NN tags in the highest angle range resolution group for Tag D are F, B and J, and the uncertainty of angle of X, F, B and J.

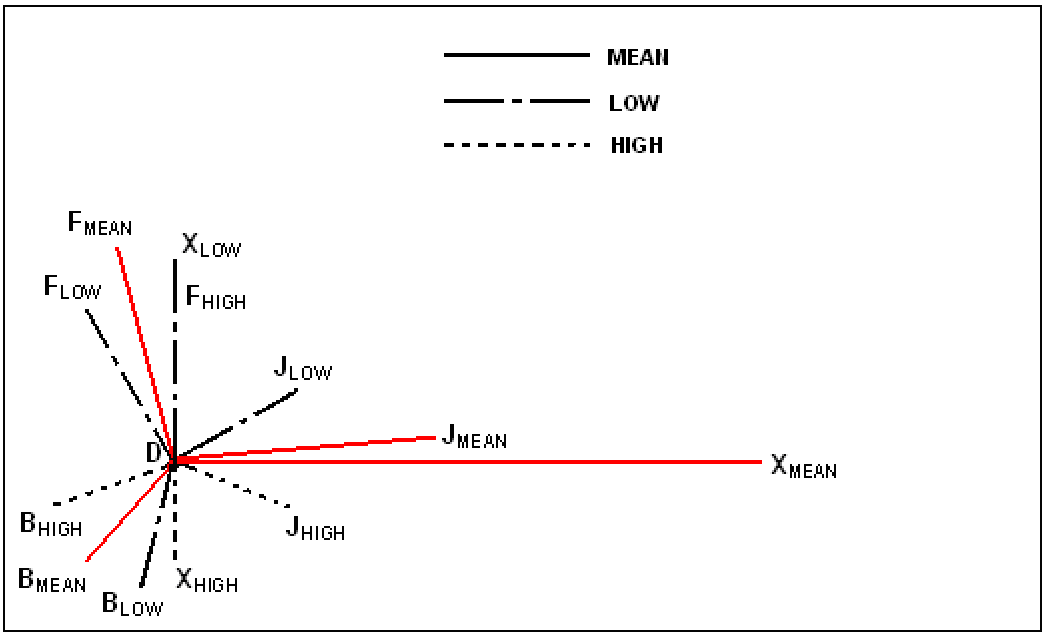


Figure 22 The angle range of each NN tag for Tag D

The Tag B has already been in the route, so this tag is not used in the comparison. As a result, there are two angles need to be compared which are $\angle J_{MEAN}DX_{MEAN}$ and $\angle F_{MEAN}DX_{MEAN}$. The angle $\angle J_{MEAN}DX_{MEAN}$ is smaller than $\angle F_{MEAN}DX_{MEAN}$. Therefore, the next tag to move to is Tag J. The process in the Tag D should be repeated in Tag J. As ***Figure 23*** presents, the NN tags in the highest angle range resolution group for Tag J are D, L and M, and the angle $\angle L_{MEAN}JX_{MEAN}$ and $\angle M_{MEAN}JX_{MEAN}$ need to be compared (D has already been in the route). As a result, Tag L is the next tag to move to.

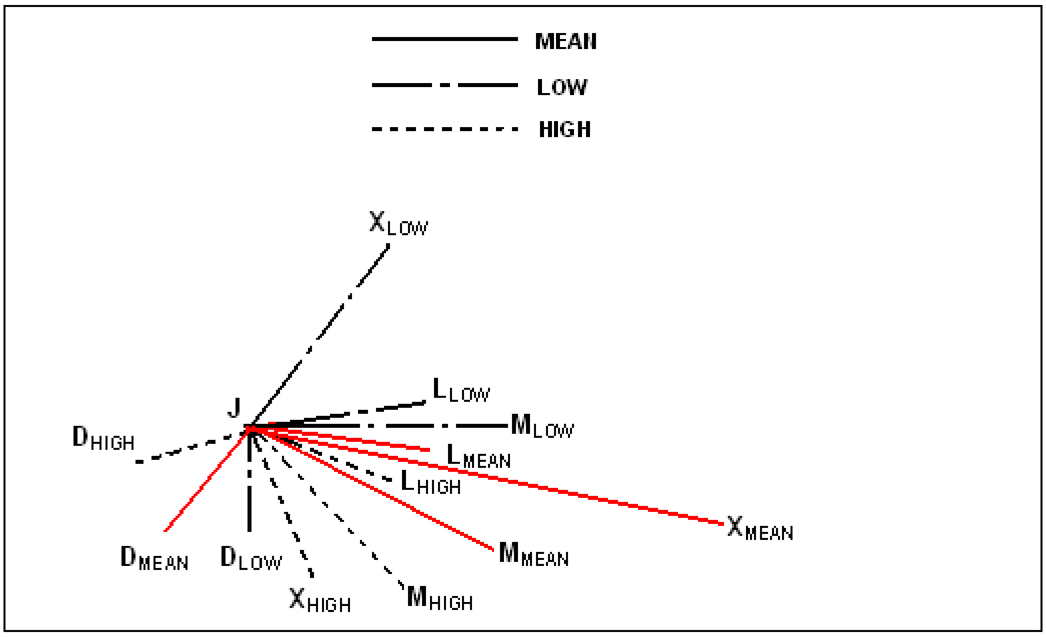


Figure 23 The angle range of each NN tag for Tag J

As ***Figure 24*** presents, the NN tags in the highest angle range resolution group for Tag L are J, M and N, and the angle $\angle M_{MEAN}LX_{MEAN}$ and $\angle N_{MEAN}LX_{MEAN}$ need to be compared (J has already been in the route). As a result, Tag N is the next tag to move to.

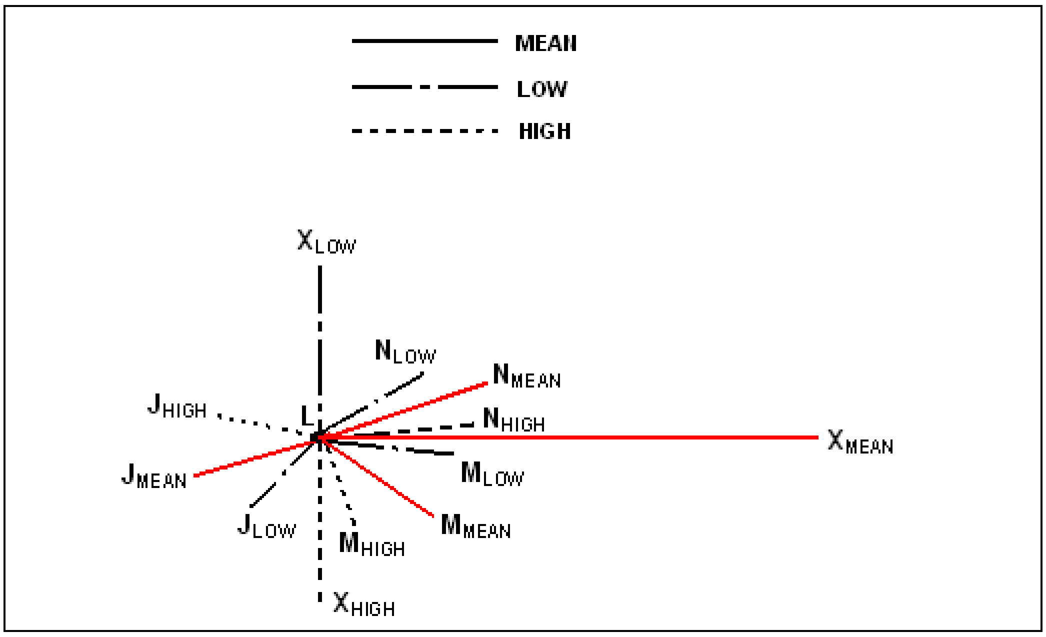


Figure 24 The angle range of each NN tag for Tag L

As ***Figure 25*** presents, the NN tags in the highest angle range resolution group for Tag N are I, M and O, and the angle ∠$I_{MEAN}NX_{MEAN}$, ∠$M_{MEAN}NX_{MEAN}$ and ∠$O_{MEAN}NX_{MEAN}$ need to be compared. As a result, Tag O is the next tag to move to.

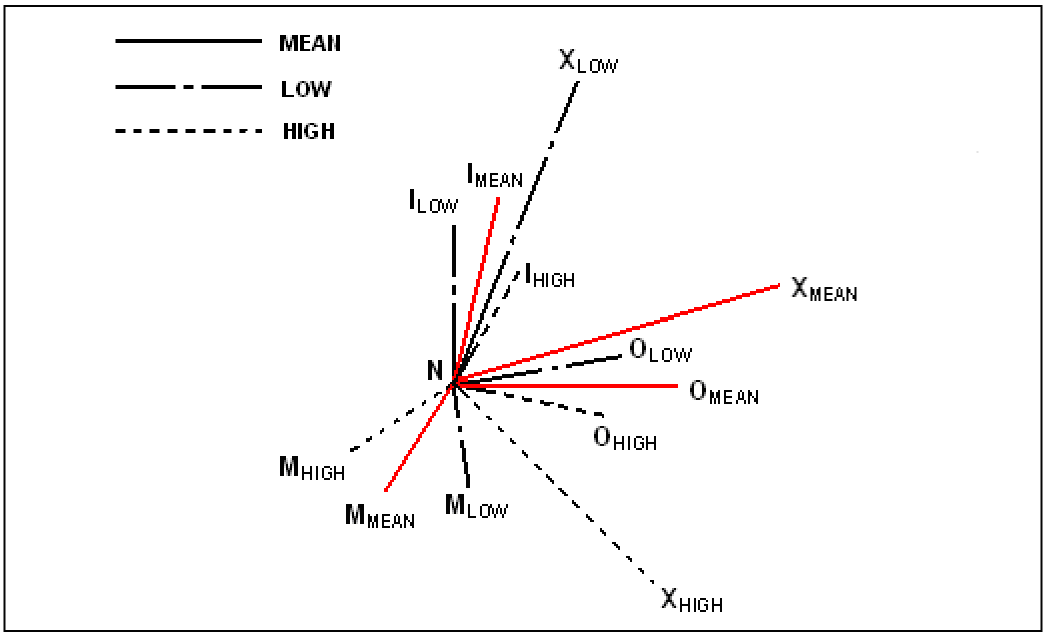


Figure 25 The angle range of each NN tag for Tag N

Figure 26 presents the NN tags in the highest angle range resolution group for Tag O are N, M and I, and the angle ∠$M_{MEAN}OX_{MEAN}$ and ∠I_{MEAN}OXMEAN need to be compared (N has already been in the route). As a result, Tag I is the next tag to move to.

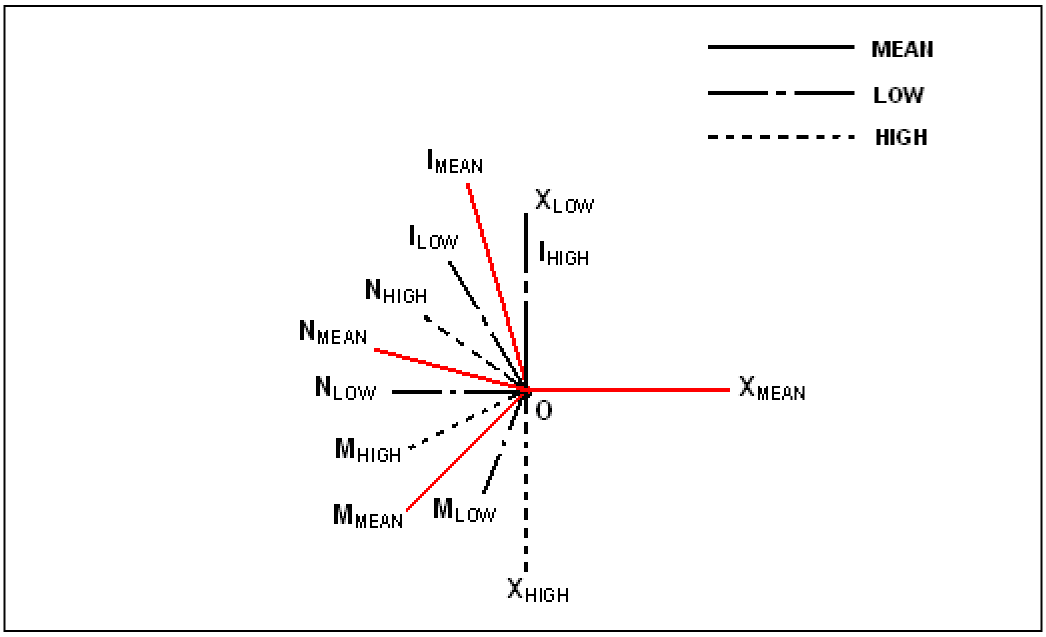


Figure 26 The angle range of each NN tag for Tag O

Figure 27 presents the NN tags in the highest angle range resolution group for Tag I are H, R and N, and the angle ∠$H_{MEAN}IX_{MEAN}$ and ∠$R_{MEAN}IX_{MEAN}$ need to be compared (N has already been in the route). As a result, Tag R is the next tag to move to.

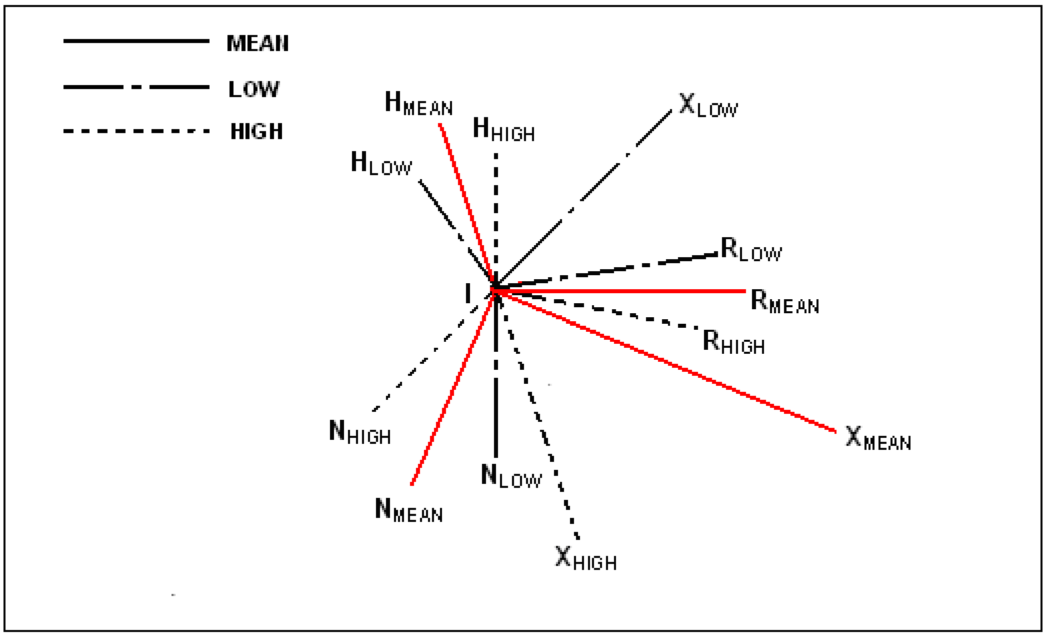


Figure 27 The angle range of each NN tag for Tag I

Figure 28 presents the NN tags in the highest angle range resolution group for Tag R are S, H and I, and the angle $\angle S_{MEAN}RX_{MEAN}$ and $\angle H_{MEAN}RX_{MEAN}$ need to be compared (I has already been in the route). As a result, Tag S is the next tag to move to.

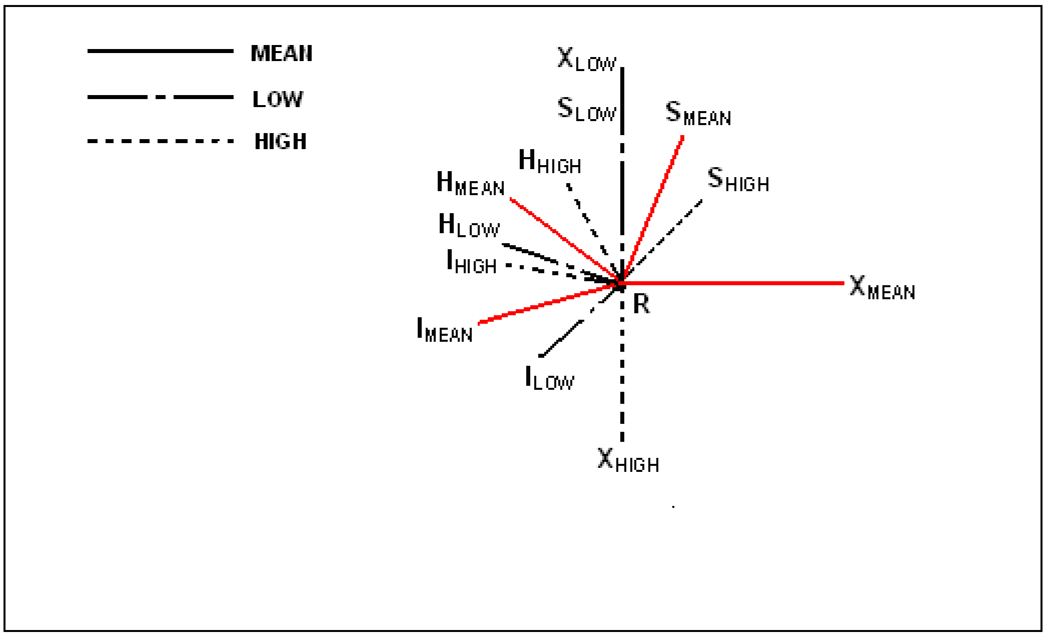


Figure 28 The angle range of each NN tag for Tag R

Figure 29 presents the NN tags in the highest angle range resolution group for Tag S are U, W and R, and the angle $\angle U_{MEAN}SX_{MEAN}$ and $\angle W_{MEAN}SX_{MEAN}$ need to be compared (R has already been in the route). As a result, Tag W is the next tag to move to.

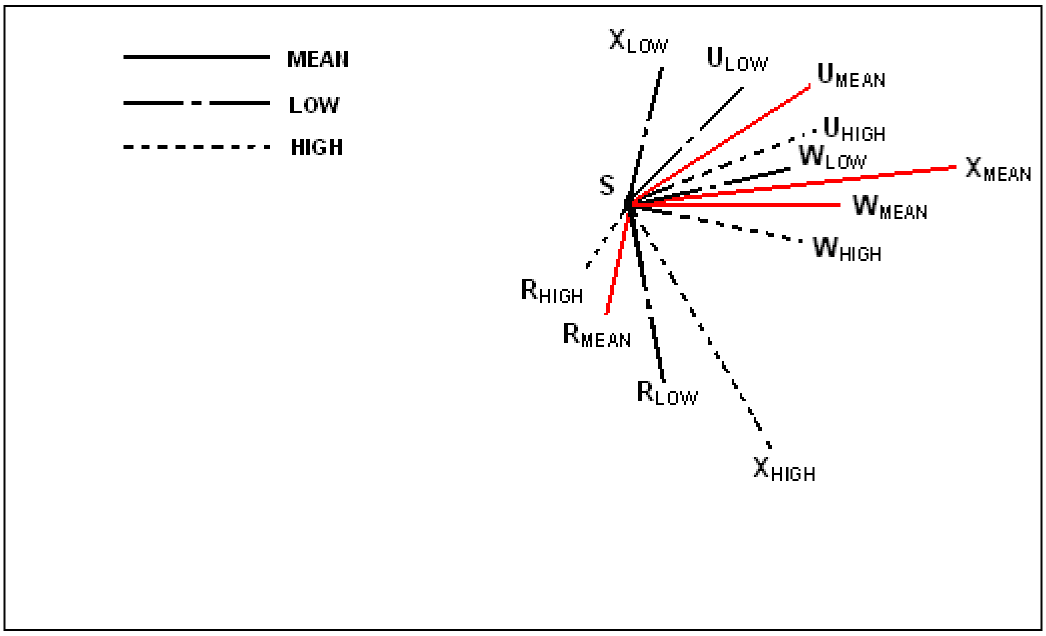


Figure 29 The angle range of each NN tag for Tag S

Figure 30 presents the NN tags in the highest angle range resolution group for Tag W are U, V and X. As a result, the destination - Tag X is included in the highest angle range resolution group of Tag W. Therefore, the final movement is from Tag W to the destination - Tag X.

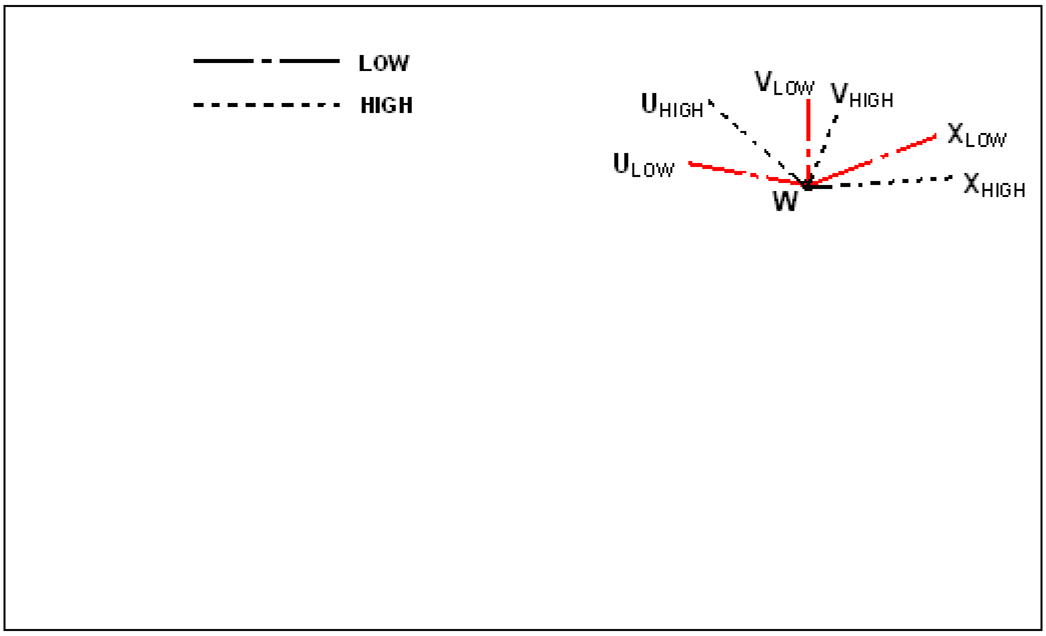


Figure 30 The angle range of each NN tag for Tag W

According to the whole process of this navigation, the route of the navigation is B (start) → D → J → L → N → O → I → R → S → W →X (finish) which is presented in ***Figure 31***.

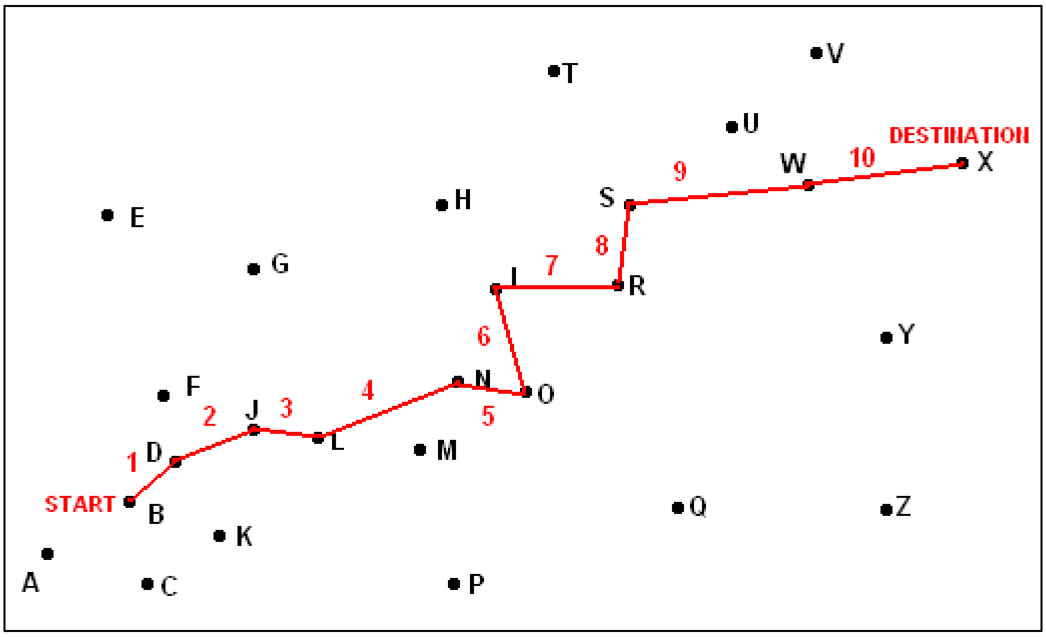


Figure 31 The route of the navigation

Chapter 5: Experimental Methodology and Results

The theoretical fundamental of the research is already discussed in the previous section: chapter 4. As a result, the next work is to implement it. There are two experiments are carried. The experimental methodology and the discussion of the results are also provided for each experiment.. The first experiment is the distance and angle experiment which the seventeen locations of RFID tags are randomly generate. The distance and angle for the sixteen neighbour tags are calculated for each of tag and the data is sorted by the distance. Then, the Haar wavelet compression technique is applied on each tag to compress the distance of its sixteen neighbour tags which generates the Haar wavelet coefficients. Because of the limitation of the storage space, there are only eight coefficients are stored in each tag. The distances of the sixteen neighbour tags are re-calculated by using these eight coefficients in order to compare with the original distance data and calculate the percentage of error in resolution of the distance data. For the angle data, the sixteen neighbour tags in each tag are divided into three different resolution levels which are eight digits, four digits and one digit. Each of angle data for the sixteen neighbour tags is converted into binary number. The angles of the sixteen neighbour tags are converted again from their binary number in order to compare with the original angle data and calculate the percentage of error in resolution of the distance data. The second experiment is navigation experiment. There are two sets of data are used which the first set of data is same as the data is used in the first experiment and the second set of data is a new set of random data. There are ten tests of navigation for each set of data. Each test is assigned an origin tag and a destination tag and there is a Java programme which is used to calculate the next point to move to from the current location. After each of the tests, the total moving distance is calculated in order to calculate the difference and the percentage of the excess from the actual distance.

5.1 Experiment 1: Distance and angle

It appears reasonable to assume that half the storage space is used for distance and half for angle, except the label of tag name. Because of these three types of data are only provided in the navigation system, and the integer format is used for tag name and the double format is used for both distance and angle. The average storage space in the passive RFID tag is approximately one hundred bytes. The size of double format of a single value can be five to eight bites. Therefore, the usage of the storage space needs to be managed efficiently. On compressed term, the scheme for different amount of tags is different. For instance, for one hundred tags, the amount of memory space are used for the tag label can be three hundred eighty-seven bits, three hundred eighty-eight bits or three hundred eighty-nine bits which depends on the number of bits of label of the current position tag. As a result, the remaining memory space for distance data and angle data are approximate to fifty bytes. The wavelet coefficients take approximate to fifteen bytes. Therefore, there are thirty-five bytes (280 bits) can be used for angle data which uses quadrant compression technique. If one hundred tags are divided into three resolution groups which are eight bits, four bits and one bit, it is possible to have eight tags in eight bits group, eight tags in four bits group and eighty-four tags in one bit group. However, for the twenty tags, the label data takes approximate to nine bytes of memory space. As a result, the remaining memory space approximate to eighty bytes. Thus, the distance data and angle data both can have more number of tags in the high resolution groups. Furthermore, ***Figure 32*** presents the procedures for producing the files of experimental distance, angle and tag label which are stored in the tags.

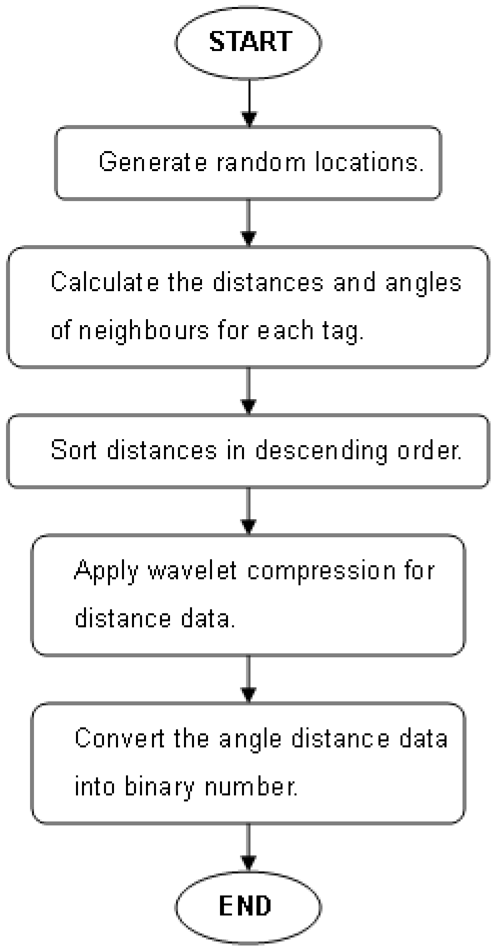

START
Generate random locations.
Calculate the distances and angles of neighbours for each tag.
Sort distances in descending order.
Apply wavelet compression for distance data.
Convert the angle distance data into binary number.
END

Figure 32 Flowchart of the production of experimental data

In the first section of this experiment, the seventeen locations of the tags which I decide are randomly generated by a java programme – RandomMap.java (appendix 2). It can be seen that the tags can be deployed in any location so the locations of the tags are randomly generated. Furthermore, the seventeen locations are generated, so there will be sixteen neighbour tags for each tag. As a result, ***Figure 33*** presents the Haar Wavelet Tree for Tag1 which the number 1 to 16 represents the wavelet coefficients which are in five different levels, and N1 to N16 represents the neighbours of Tag1 from the nearest to farthest. The wavelet coefficients are calculated by WaveletConstruction.java (appendix 3).

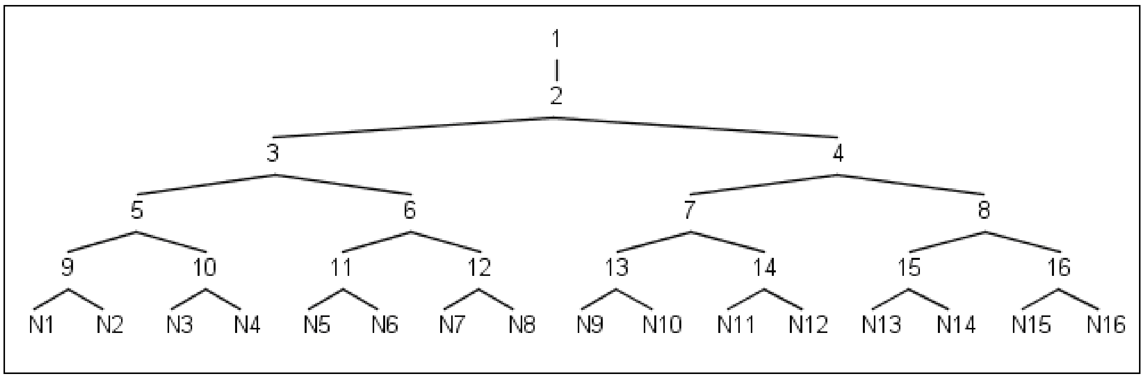
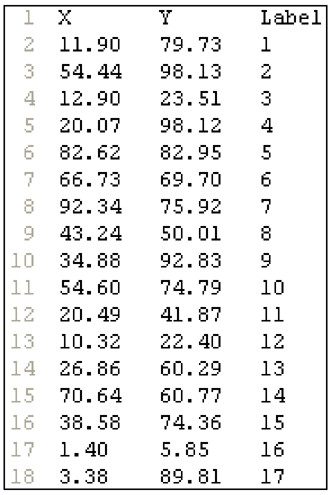


Figure 33 Haar Wavelet Tree for Tag1: total of 17 tags including Tag1

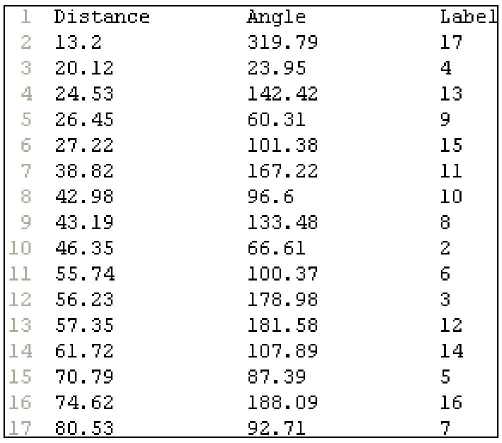
From the left half of tree, it can be divided into three groups which N1, N2, N3 and N4 are the group 1, N5, N6, N7 and N8 are the group 2, and N9 to N16 are the group 3. If the resolution of reconstructive values needs to be less and less, this part of tree can provide three different resolution levels. In order to achieve this, the coefficient 11 and 12 will not be record for group 2, and the coefficient 7, 8, 13, 14, 15 and 16 will not be record for group 3. Furthermore, if there are only nine locations of tags are randomly generated, they also can be divided into three groups but there will only two tags in the highest resolution level. According to this, the testing of seventeen tags has more tags in each resolution group to examine and compare the change and the affection of the resolution. Moreover, the seventeen tags is the smallest number that can be used to generate at least three different resolution groups with at least four tags in each group. In RandomMap.java (appendix 2) programme, the duplicate locations will be checked in order to assign a different location for each of tag. ***Figure 34*** presents the output file of this programme - ranTags.txt which is the X-axial value and the Y-axial value of each tag.



```
X       Y       Label
11.90   79.73   1
54.44   98.13   2
12.90   23.51   3
20.07   98.12   4
82.62   82.95   5
66.73   69.70   6
92.34   75.92   7
43.24   50.01   8
34.88   92.83   9
54.60   74.79   10
20.49   41.87   11
10.32   22.40   12
26.86   60.29   13
70.64   60.77   14
38.58   74.36   15
1.40    5.85    16
3.38    89.81   17
```

Figure 34 ranTags.txt

The Calculations.java (appendix 4) is run subsequently in order to generate the distances and angles between each tag to other tags. There are seventeen assumed tags are tested in this section, as a result seventeen output files will be generated by Calculations.java which are Tag1.txt, Tag2.txt, Tag3.txt and so on. Each of these seventeen output files contains three columns of data which the third column represents as tag label, whereas the first column (sorted) and second column are the distance and the angle of tag (display in the third column), respectively. For instance, in Tag1.txt, the target tag is tag1, thus the distance and angle is calculated between tag1 and each remaining sixteen tags. ***Figure35*** presents the distance, angle and label of each neighbour tag of Tag1 which the row one to row sixteen shows the sorted distances (ascending order) with the angles between tag1 and tag2 to tag17.



Distance	Angle	Label
13.2	319.79	17
20.12	23.95	4
24.53	142.42	13
26.45	60.31	9
27.22	101.38	15
38.82	167.22	11
42.98	96.6	10
43.19	133.48	8
46.35	66.61	2
55.74	100.37	6
56.23	178.98	3
57.35	181.58	12
61.72	107.89	14
70.79	87.39	5
74.62	188.09	16
80.53	92.71	7

Figure 35 The distance, angle and label of each neighbour tags of Tag1 (sorted by distance)

In this experiment, the Tag1.txt has been chosen and it is used to examine the efficiency of use of the storage space. For the distance data and angle data, all the sixteen data from both sets will be retained because the memory space allows it to do so. The wavelet compression is applied to the distance data. Because of the storage space and concern of resolution, there are only eight wavelet coefficients are stored in a file for Tag1. As showing in the ***Figure 33*** (page 62), from N1 to N4 are the nearest neighbour (NN) group, from N5 to N8 are the second close neighbour group, and from N9 to N16 are the third close neighbour group. The eight coefficients will be stored are coefficient 1, 2, 3, 4, 5, 6, 9 and 10. According to this, the resolution of the tags is getting fuzzier and fuzzier because the value of N1, N2, N3 and N4 can be exactly reconstructed, the value of N5, N6, N7 and N8 is deficiency of one coefficient in the calculation, and the value of N9, N10, N11, N12, N13, N14, N15 and N16 is deficiency of two coefficients in the calculation.

The equation – "((|Original Value – Reconstructive Value|) / Original Values) x 100%" is applied for calculate the percentage of the error in resolution for each distance. ***Table 9*** presents the percentage of the error in resolution for each distance for Tag1.

Neighbour Number	**% Error in Resolution**
N1	0.00 %
N2	0.00 %
N3	0.00 %
N4	0.00 %
N5	21.31 %
N6	14.94 %
N7	0.24 %
N8	0.24 %
N9	16.33 %
N10	3.27 %
N11	4.11 %
N12	5.98 %
N13	16.52 %
N14	1.59 %
N15	3.63 %
N16	10.70 %

Table 9 The percentage of the error in resolution for each distance for Tag1

In ***Table 9***, the percentage of the error in resolution for NN group are all 0.00% because the reconstruction values in this group are exactly been reconstructed without any missing coefficient. The lowest value and highest value in second close neighbour group are 0.24% and 21.31%. The lowest value and highest value in third close neighbour group are 1.59% and 16.52%. The saved storage space for each resolution level group for distance data can be calculate by the equation – "(the number of saved coefficient x 7)", because the size of one coefficient is approximate to seven bits. ***Figure 36*** presents the saved storage space of distance from nearest distance group to farthest distance group for Tag1. In ***Figure 36***, the gradient between Group2 and Group3 is bigger than the gradient between Group1 and Group2 which means the saved storage space of distance in Group3 is more than Group2. Compare with ***Figure 15*** (page 32) in section 2.6, ***Figure 36*** is similar to Line 3 where resolution is inversely relation to storage.

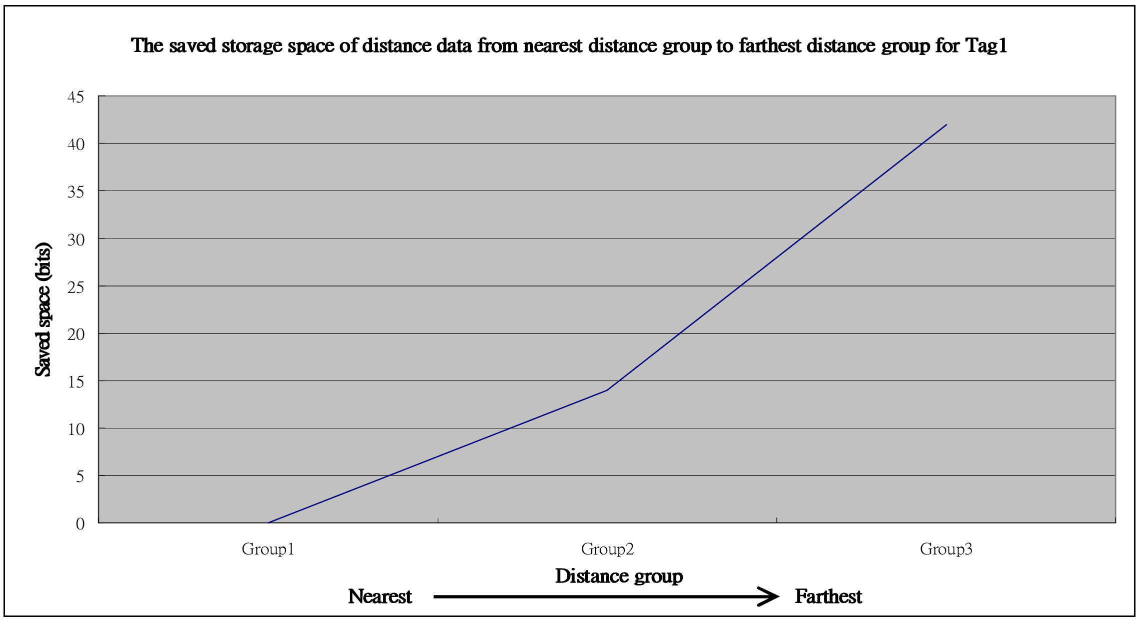


Figure 36 The saved storage space of distance data from nearest distance group to farthest distance group for Tag1

Furthermore, the fifteen different groups of random number are generated for the next examination. As a result, there are fifteen different distance data of Tag1. ***Table 10*** presents the percentage of the error in resolution of distance for all sixteen neighbour tags of Tag1 for fifteen different random number groups from nearest to farthest (N1 to N16), and the average (AVG) value of and the Standard Deviation value (STDEV) of the percentage of the error in resolution of all groups within each neighbour.

	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	N16
Group1	0.00%	0.00%	0.00%	0.00%	21.31%	14.94%	0.24%	0.24%	16.33%	3.27%	4.11%	5.98%	16.52%	1.59%	3.63%	10.70%
Group2	0.00%	0.00%	0.00%	0.00%	3.69%	3.43%	3.94%	3.65%	4.81%	3.75%	1.68%	6.09%	31.33%	1.05%	1.53%	18.92%
Group3	0.00%	0.00%	0.00%	0.00%	19.80%	14.18%	8.00%	6.90%	4.27%	1.63%	1.54%	3.96%	15.79%	1.62%	5.28%	8.81%
Group4	0.00%	0.00%	0.00%	0.00%	5.91%	5.28%	14.07%	10.98%	8.96%	5.83%	5.32%	7.51%	7.08%	6.16%	5.12%	14.74%
Group5	0.00%	0.00%	0.00%	0.00%	20.09%	14.33%	2.66%	2.52%	12.07%	4.81%	4.71%	9.43%	18.64%	3.00%	0.06%	15.74%
Group6	0.00%	0.00%	0.00%	0.00%	3.72%	3.46%	13.55%	10.66%	25.28%	4.90%	6.05%	7.92%	13.72%	6.57%	2.68%	17.25%
Group7	0.00%	0.00%	0.00%	0.00%	11.21%	9.16%	3.94%	3.66%	12.51%	0.19%	4.74%	5.95%	10.49%	4.54%	1.58%	13.34%
Group8	0.00%	0.00%	0.00%	0.00%	2.06%	1.97%	3.34%	3.13%	8.57%	7.48%	5.09%	8.66%	5.84%	1.80%	0.34%	7.09%
Group9	0.00%	0.00%	0.00%	0.00%	9.18%	7.76%	8.73%	7.43%	10.80%	0.65%	0.84%	8.71%	25.38%	3.28%	0.69%	19.42%
Group10	0.00%	0.00%	0.00%	0.00%	0.99%	0.97%	3.63%	3.39%	12.43%	1.88%	1.23%	10.44%	28.86%	4.08%	0.27%	21.01%
Group11	0.00%	0.00%	0.00%	0.00%	2.19%	2.10%	8.13%	6.99%	3.84%	0.18%	1.38%	2.07%	22.65%	4.99%	7.37%	13.23%
Group12	0.00%	0.00%	0.00%	0.00%	16.73%	12.54%	7.29%	6.36%	5.07%	0.82%	2.10%	3.37%	16.65%	1.62%	5.57%	9.07%
Group13	0.00%	0.00%	0.00%	0.00%	3.53%	3.30%	0.94%	0.92%	5.13%	0.95%	0.04%	5.53%	25.18%	2.48%	2.63%	12.94%
Group14	0.00%	0.00%	0.00%	0.00%	9.14%	7.73%	3.76%	3.49%	17.74%	5.53%	8.37%	10.05%	9.81%	5.56%	3.01%	14.62%
Group15	0.00%	0.00%	0.00%	0.00%	2.41%	2.30%	2.69%	2.56%	11.52%	4.41%	1.01%	13.46%	15.22%	8.76%	4.76%	20.51%
AVG	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**8.80%**	**6.90%**	**5.66%**	**4.86%**	**10.62%**	**3.08%**	**3.21%**	**7.28%**	**17.54%**	**3.81%**	**2.97%**	**14.49%**
STDEV	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**7.33%**	**5.03%**	**4.18%**	**3.23%**	**5.95%**	**2.35%**	**2.42%**	**3.00%**	**7.77%**	**2.26%**	**2.26%**	**4.38%**

Table 10 The percentage of the error in resolution of distance, and the AVG of and the STDEV of % error in resolution for fifteen different groups of random number

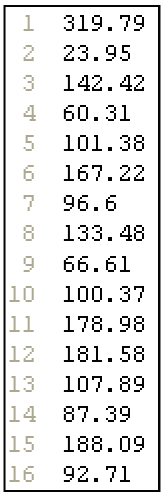
In ***Table 10***, the AVG and STDEV for NN group are both 0.00%. From N6 to N 8 are second close neighbour group and from N9 to N 16 are third close neighbour group. However, the lowest AVG in third close neighbour group is 2.97% which is lower than all AVG in second close neighbour group as well as another three AVG values from the third close neighbour group. The AVG of the resolution from nearest tag group to farthest tag group is not less and less as well as the STDEV. As a result, the change of resolution in distance is instable. Furthermore, ***Table 11*** presents the original size of the distance data, the size of the distance data by using Wavelet compression technique and the size of the distance data by using Zip compression technique for each of seventeen testing tags.

	Original size of distance data (byte)	**After Wavelet compression (byte)**	**After Zip Compression (byte)**
Tag 1	111	66	182
Tag 2	112	69	184
Tag 3	110	69	184
Tag 4	110	71	184
Tag 5	112	71	185
Tag 6	111	68	182
Tag 7	109	70	185
Tag 8	111	68	181
Tag 9	110	70	182
Tag 10	108	69	185
Tag 11	110	68	187
Tag 12	109	68	184
Tag 13	110	68	184
Tag 14	110	67	186
Tag 15	112	69	187
Tag 16	111	65	187
Tag 17	110	67	184

Table 11 The original size of the distance data, the size of the distance data by using Wavelet compression technique and the size of the distance data by using Zip compression technique for each of seventeen testing tags

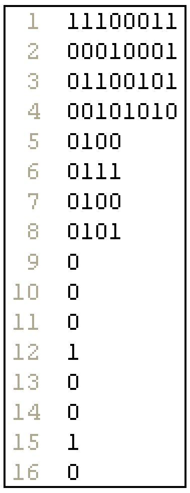
In Zip compression technique, the reconstructive data is exactly same as the original data. However, in ***Table 11*** shows the size of the data which employs the Zip compression technique is increased. On the other hand, the size of the data which uses the Wavelet compression technique is decreased. According to this, the size of the data did not reduce using Zip compression technique which is the opposite of the goal of this study, although it gets the zero percentage of error in resolution of all the reconstructive data. The distance section of the experiment has been described. Therefore, the angle section of the experiment is going to be discussed in the next.

The seventeen tags which are generated in distance section is used for angle section, thus the amount of NN tags in angle is same as in distance. The quadrant compression technique is used for the angle data. Furthermore, there are three different angle levels are assumed for the angle data which are eight bits, four bits and one bit. In order to match the resolution of the distance which is getting more uncertain by each group, the first four angles are assigned to eight bits, the next four angles are assigned to four bits, and the last eight angles are assigned to one bit. According to this, the CalAngleBinary.java (appendix 1) is applied to convert the original angle data to their binary angle data. ***Figure 37*** presents the original angle data of Tag1 and ***Figure 38*** presents the converted angle data of the first eight values of Tag1.



```
319.79
23.95
142.42
60.31
101.38
167.22
96.6
133.48
66.61
100.37
178.98
181.58
107.89
87.39
188.09
92.71
```

Figure 37 The original angle data of Tag1



```
11100011
00010001
01100101
00101010
0100
0111
0100
0101
0
0
0
1
0
0
1
0
```

Figure 38 The converted angle data of Tag

The equation – "((|the original angle – the mean of the converted angle range) / the original angle) x 100%" is applied for calculate the percentage of the error in resolution for each angle. ***Table 12*** presents the percentage of the error in resolution for each angle for Tag1. In ***Table 12*** the lowest value and highest value of the percentage of the error in resolution for NN group are 0.04% and 2.76%. The lowest value and highest value in second close neighbour group are 0.13% and 7.29%. The lowest value and highest value in third close neighbour group are 2.92% and 49.72%. Accordingly, the highest value of the percentage of the error in resolution is increasingly from NN group to the third close neighbour group.

Neighbour Number	% Error in Resolution
N1	0.04 %
N2	2.76 %
N3	0.22 %
N4	0.90 %
N5	0.13 %
N6	0.91 %
N7	4.81 %
N8	7.29 %
N9	35.11 %
N10	10.33 %
N11	49.72 %
N12	48.69 %
N13	16.58 %
N14	2.99 %
N15	43.55 %
N16	2.92 %

Table 12 The percentage of the error in resolution for each angle for Tag1

The saved storage space for each resolution level group for the angle data can be calculated by the equation – "(the number of saved bit of one tag x the number of tag in the group)". Moreover, the number of saved bit of one tag depends on the difference of the number of bits in one tag between the current group and the highest resolution angle group. ***Figure 39*** presents the saved storage space of angle from nearest distance group to farthest distance group for Tag1.

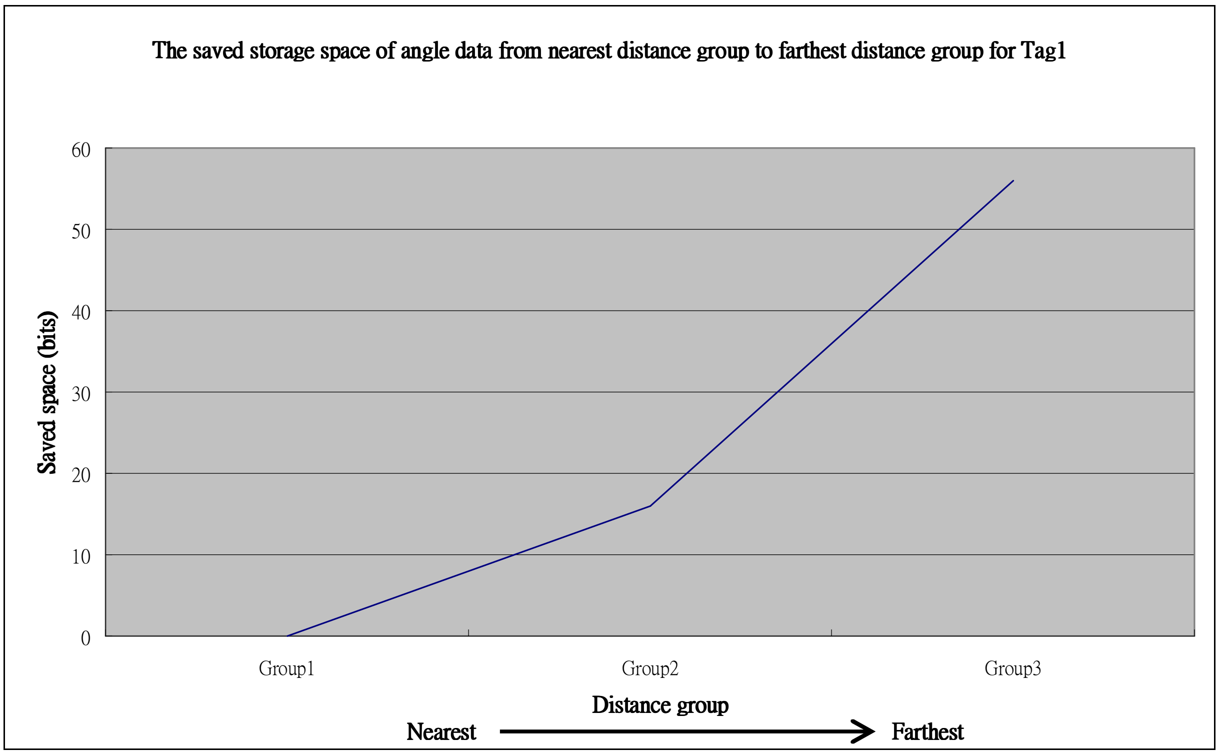


Figure 39 The saved storage space of angle data from nearest distance group to farthest distance group for Tag1

The obliquity between Group2 and Group3 is more oblique than the obliquity between Group1 and Group2 which means the total saved storage space in Group3 is more than Group2. Furthermore, from the equation of saved space of angle, the two facts that impact on the amount of saved space are the number of saved bit of one tag and the number of tag in the group. According to this, the more storage space is saved while the more number of tag in the lower resolution. Furthermore, the fifteen different groups of random number are generated in previous section are also used for the next examination. As a result, there are fifteen different angle data of Tag1. ***Table 13*** presents the percentage of the error in resolution of angle for all neighbour tags of Tag1 for fifteen different random number groups from nearest to farthest (N1 to N16), and the average (AVG) value of and the Standard Deviation value (STDEV) of the percentage of the error in resolution of all groups within each nearest neighbour. In ***Table 13***, from N1 to N4 is the nearest neighbour group, from N5 to N8 are second close neighbour group and from N9 to N16 are third close neighbour group. The highest AVG value in group 1 is 1.32% which is lower than the lowest AVG value 5.54% in group 2. The highest AVG value in group 2 is 22.50% which is lower than the lowest AVG value 28.90% in group 3. As a result, the resolution becomes lower and lower from the nearest neighbour tag group to the farthest tag group.

	N1	**N2**	**N3**	**N4**	**N5**	**N6**	**N7**	**N8**	**N9**	**N10**	**N11**	**N12**	**N13**	**N14**	**N15**	**N16**
Group1	0.04%	2.76%	0.22%	0.90%	0.13%	0.91%	4.81%	7.29%	35.11%	10.33%	49.72%	48.69%	16.58%	2.99%	43.55%	2.92%
Group2	0.40%	0.22%	0.18%	0.31%	16.56%	8.58%	0.82%	6.27%	26.10%	44.94%	25.69%	43.62%	14.52%	47.16%	30.74%	28.78%
Group3	0.04%	0.09%	0.56%	1.63%	13.61%	0.40%	2.79%	2.59%	109.50%	19.91%	4.63%	779.77%	27.97%	771.25%	24.03%	55.20%
Group4	0.29%	0.67%	0.34%	0.11%	1.34%	0.38%	0.50%	0.28%	49.58%	389.13%	21.64%	0.67%	33.59%	8.75%	13.51%	15.10%
Group5	0.07%	0.17%	1.25%	0.08%	0.68%	3.67%	5.86%	1.77%	19.27%	44.73%	103.07%	15.25%	37.17%	29.47%	44.10%	14.42%
Group6	0.24%	0.16%	0.18%	4.61%	0.09%	2.91%	1.07%	186.99%	14.58%	24.20%	21.09%	18.99%	8.76%	18.60%	15.33%	13.82%
Group7	0.01%	0.42%	0.01%	0.98%	6.35%	4.99%	0.61%	0.99%	38.02%	40.03%	20.49%	39.33%	17.99%	31.27%	17.90%	39.39%
Group8	0.30%	0.02%	0.07%	0.34%	2.88%	2.38%	37.50%	1.03%	20.29%	12.77%	20.83%	7.55%	15.31%	19.13%	11.71%	13.08%
Group9	0.16%	0.03%	0.00%	0.18%	4.49%	1.68%	2.64%	0.24%	7.22%	0.67%	0.31%	41.32%	0.80%	43.17%	25.85%	34.17%
Group10	0.44%	0.41%	0.34%	0.16%	5.76%	8.51%	1.30%	1.35%	18.48%	23.38%	21.53%	19.23%	18.32%	170.76%	121.13%	55.47%
Group11	0.10%	11.75%	0.30%	6.12%	7.42%	1.37%	6.50%	6.83%	25.47%	124.22%	24.57%	8.38%	644.42%	47.71%	108.00%	15.21%
Group12	0.72%	2.66%	0.05%	0.30%	12.73%	7.85%	9.57%	47.25%	249.38%	111.96%	993.56%	47.40%	161.25%	2.24%	56.79%	41.00%
Group13	0.18%	0.17%	1.48%	0.11%	1.36%	118.87%	4.89%	0.06%	58.26%	14.60%	24.85%	68.92%	13.69%	215.46%	817.43%	83.04%
Group14	0.16%	0.13%	0.01%	0.06%	48.42%	3.58%	4.08%	71.76%	24.31%	21.24%	1.44%	15.57%	17.74%	7.67%	11.61%	10.56%
Group15	0.09%	0.08%	0.04%	0.06%	1.89%	2.26%	0.22%	2.81%	8.57%	21.59%	1.63%	2.01%	4.14%	6.32%	9.02%	11.34%
AVG	**0.22%**	**1.32%**	**0.34%**	**1.06%**	**8.25%**	**11.22%**	**5.54%**	**22.50%**	**46.94%**	**60.25%**	**89.00%**	**77.11%**	**68.82%**	**94.80%**	**90.05%**	**28.90%**
STDEV	**0.19%**	**3.02%**	**0.45%**	**1.83%**	**12.28%**	**29.91%**	**9.24%**	**49.94%**	**61.55%**	**97.65%**	**251.51%**	**195.43%**	**163.77%**	**197.25%**	**204.11%**	**22.36%**

Table 13 The percentage of the error in resolution of angle, and the AVG of and the STDEV of % error in resolution for fifteen different groups of random number

Furthermore, ***Table 14*** presents the original size of the angle data, the size of the angle data by using quadrant compression technique and the size of the angle data by using Zip compression technique for each of seventeen testing tags.

	Original size of angle data (byte)	**After quadrant compression (byte)**	**After Zip compression (byte)**
Tag 1	121	88	189
Tag 2	126	88	188
Tag 3	114	88	186
Tag 4	127	88	191
Tag 5	128	88	186
Tag 6	125	88	189
Tag 7	127	88	183
Tag 8	121	88	190
Tag 9	126	88	191
Tag 10	123	88	190
Tag 11	118	88	190
Tag 12	111	88	183
Tag 13	118	88	189
Tag 14	126	88	191
Tag 15	121	88	193
Tag 16	110	88	187
Tag 17	121	88	189

Table 14 The original size of the angle data, the size of the angle data by using quadrant compression technique and the size of the angle data by using Zip compression technique for each of seventeen testing tags

Using Zip compression technique, the reconstructive angle data is exactly same as the original angle data. However, ***Table 14*** shows using the Zip compression technique is increases the size of the data required. In the other hand, the size of the angle data which uses the quadrant compression technique is decreased. According to this, the size of the data is successfully reduced by quadrant compression technique which is one of the most important approaches used in this study.

5.2 Experiment 2: Navigation

The previous section 5.1 discussed the first experiment which includes distance and angle. As a result, the experiment is ready to carry on for the navigation section after the data are stored correctly in each tag. In this section, the algorithm of the navigation for the experiment is described first and then demonstration of the navigation.

In the navigation section, the most essential part is the algorithm for detecting the next tag to move to which is in the highest resolution angle range group that is closest in angle to the destination. Therefore, this algorithm is described in this section. ***Figure 40*** presents the procedure for finding next tag. The procedure of this algorithm is implemented in NextTag.java (appendix 5).

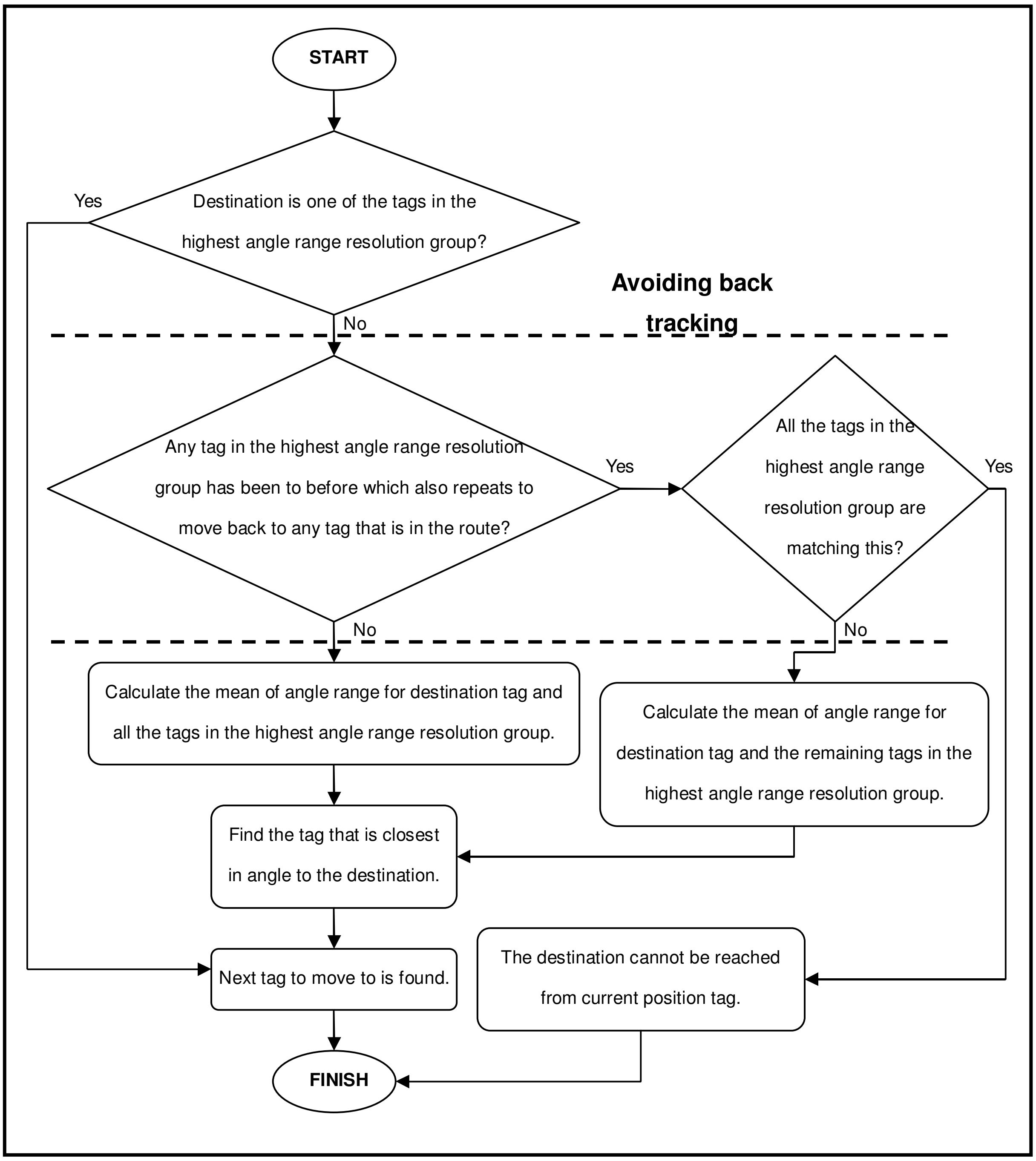
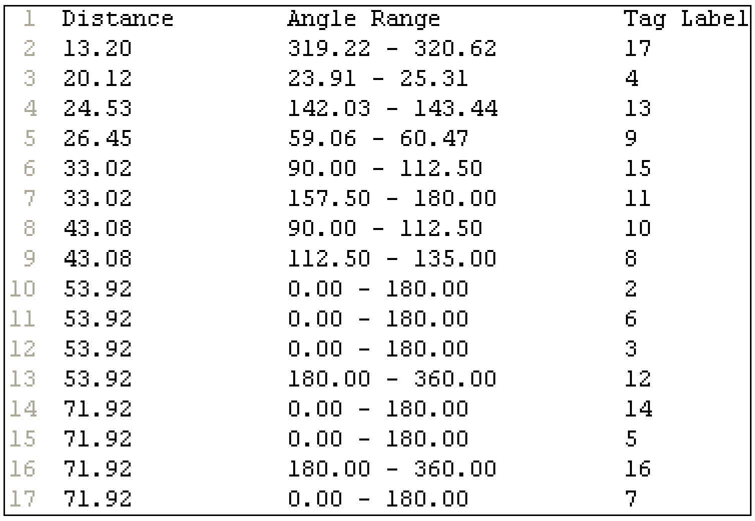

START
Destination is one of the tags in the highest angle range resolution group?
Yes
No
Avoiding back tracking
Any tag in the highest angle range resolution group has been to before which also repeats to move back to any tag that is in the route?
Yes
No
All the tags in the highest angle range resolution group are matching this?
Yes
No
Calculate the mean of angle range for destination tag and all the tags in the highest angle range resolution group.
Calculate the mean of angle range for destination tag and the remaining tags in the highest angle range resolution group.
Find the tag that is closest in angle to the destination.
Next tag to move to is found.
The destination cannot be reached from current position tag.
FINISH

Figure 40 Flowchart of finding next tag

After the next tag to move to is calculated by NextTag.java, the user accordance with the distance, angle and the tag label from the output file which is generated by NavigationOutput.java (appendix 6) to move. ***Figure 41*** presents the output file for Tag1 which is generated by NavigationOutput.java.



```
Distance        Angle Range          Tag Label
13.20           319.22 - 320.62      17
20.12           23.91 - 25.31        4
24.53           142.03 - 143.44      13
26.45           59.06 - 60.47        9
33.02           90.00 - 112.50       15
33.02           157.50 - 180.00      11
43.08           90.00 - 112.50       10
43.08           112.50 - 135.00      8
53.92           0.00 - 180.00        2
53.92           0.00 - 180.00        6
53.92           0.00 - 180.00        3
53.92           180.00 - 360.00      12
71.92           0.00 - 180.00        14
71.92           0.00 - 180.00        5
71.92           180.00 - 360.00      16
71.92           0.00 - 180.00        7
```

Figure 41 The navigation output file for Tag1

The first step in algorithm is to check whether the destination tag is one of the tags in the highest angle range resolution group of the current position tag. If the destination tag matches this criterion, the next tag to move to is found. However, if it does not match this criterion, the next step is to check whether any tag in the highest angle range resolution group is been to before which also repeats to move back to the tag that is in the route. This process is needed to avoid back tracking. If any tag matches this criterion, it means that this tag is been to before and no new next tag to move to so it moves back to the same tag in the route. Furthermore, if all the tags in the highest angle range resolution group match this criterion, this means there is no next tag to move to. According to this, the destination cannot be reached from the current position tag. However, if there is any number of tags remains, the final step of finding the next tag to move to is executed. In the final step, the mean of angle range for destination tag and all the remaining tags in the highest angle range resolution group will be calculated in order to have exactly angle for the next calculation. After all the mean values are calculated, the next calculation is to calculate and find the angle of the tag that is closest in angle to the destination. As a result, the next tag to move to is found after this calculation. Furthermore, if there is any remaining tag is in the back of current position tag which is in the different side from the destination tag. The calculation for calculating the angle between this tag and destination tag is different. In this situation, this tag is in the back of the current tag, so the angle should calculate anti-clockwise while the mean value of the calculating angle is greater than 180° from the angle of current tag and destination. For instance, ***Figure 42*** presents the current position tag is A, the destination is B, and B_{MEAN}, C_{MEAN}, D_{MEAN} and E_{MEAN} are the mean value for tags B, C, D and E of Tag A. The calculation for the angle between $C_{MEAN}A$ and AB_{MEAN}, $D_{MEAN}A$ and AB_{MEAN}, and $E_{MEAN}A$ and AB_{MEAN} are $\angle B_{MEAN}AC_{MEAN}$, $\angle B_{MEAN}AD_{MEAN}$, $\angle B_{MEAN}AE_{MEAN}$. The calculation for the angle between $C_{MEAN}A$ and AB_{MEAN}, $D_{MEAN}A$ and AB_{MEAN} are correct. However, the correct calculation for angle between $E_{MEAN}A$ and AB_{MEAN} should be calculate anti-clockwise which is $\angle B_{MEAN}AE_{MEAN\_2}$ not $\angle B_{MEAN}AE_{MEAN\_1}$ because it is more accurately and logically.

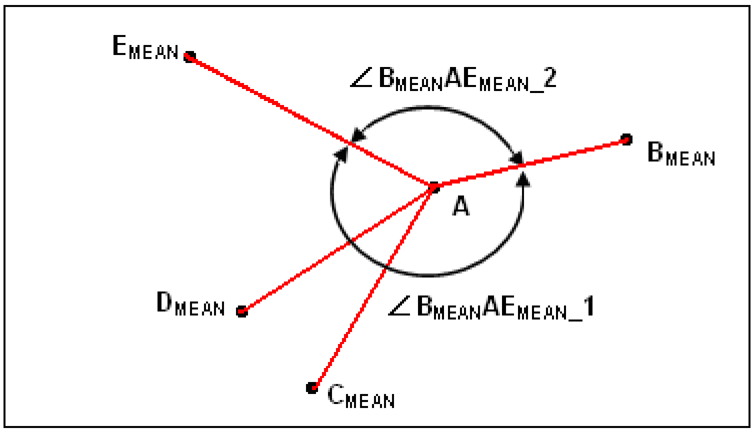


Figure 42 The example of current position Tag A, the destination Tag B, and the mean value for NN tags (C, D and E) for Tag A

The algorithm of navigation is already described. Therefore, in the following paragraph, the experiment of the navigation is presented. The data are used in the first example are the existing data which have already been generated in the previous experiment section. For these seventeen tags, Tag 16 (T16) is assumed as the origin position and Tag 5 (T5) is assumed as the destination. The programme is used to generate the next tag to move to from the current position tag is NextTag.java (appendix 5). There are three inputs for this programme which are current position tag, destination tag and the route which has already been travelled (includes current position tag). Because of the destination is a fixed tag, so the input for the current position tag needs to be changed after move to new tag, as well as the route. In the first execution of NextTag.java, the current position tag is T16 and the already visited list includes: T16. The result of output shows the next tag to move to is T3. In the second execution of NextTag.java, the current position tag is T3 and the already visited list includes: T16 and T3. The result of output shows the next tag to move to is T11. In the third execution of NextTag.java, the current position tag is T11 and the already visited list includes: T16, T3 and T11. The result of output shows the next tag to move to is T8. In the fourth execution of NextTag.java, the current position tag is T8 and the already visited list includes: T16, T3, T11 and T8. The result of output shows the next tag to move to is T10. In the fifth execution of NextTag.java, the current position tag is T10 and the already visited list includes: T16, T3, T11, T8 and T10. The result of output shows the next tag to move to is T6. In the sixth execution of NextTag.java, the current position tag is T6 and the already visited list includes: T16, T3, T11, T8, T10 and T6. The result of output shows the next tag to move to is the destination T5. Therefore, the last movement is move to the destination Tag T5. According to this navigation, the route for navigating from T16 to T5 is T16 → T3 → T11 → T8 → T10 → T6 → T5 which is showed in ***Figure 43***.

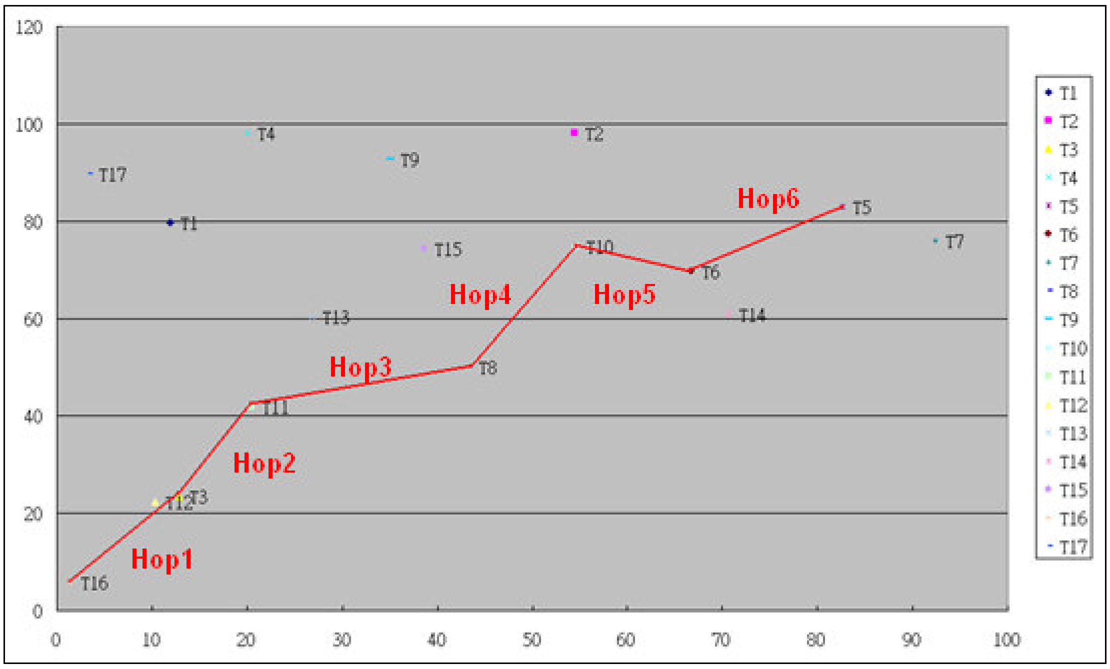


Figure 43 The hops in the navigation of T16 to T5

Table 15 presents all the tags are recorded in this navigation, the distance between each movement, the distance between T16 and T5, the difference between the total distance of the route and the actual distance between T16 and T5, and the percentage of the access of the navigation. The equation is used to calculate the percentage of the excess of the navigation is - "(The difference between actual distance and sum of the distance of each hop / actual distance) x 100%". In ***Table 15***, the percentage of the excess is 12.69, thus it means the difference between the sum of all hops and actual distance is not very great.

	Movement
T16	0
T3	21.07
T11	19.87
T8	24.16
T10	27.26
T6	13.15
T5	20.69
SUM	126.2
Actual distance from T16 to T5	111.99
Difference	14.21
% EXCESS	12.69 %

Table 15 The movement to each Tag, and the SUM and the % EXCESS of the navigation

The navigation will be tested ten more times and each test contains a different set of origin tag and destination. ***Table 16*** presents the origin tag, the destination, number of hop in the route, the sum of the movement, the actual distance between origin tag and destination, the difference between the sum of the movement and actual distance, and the percentage of excess of each test for these ten navigations. In ***Table 16***, the lowest percentage of excess is 1.05% which is in run number 1 and its number of hop is 3. However, the highest percentage of excess is 188.94% which is in run number 6 and its number of hop is 6. The number of hop in run number 9 and 10 are 9 which are both more than run number 6 but their percentage of excess are both lower than run number 6.

Run Number	Origin Tag	Destination	Number of Hop in Route	SUM	Actual Distance	Difference	% EXCESS
1	T12	T15	3	59.77	59.15	0.62	1.05 %
2	T7	T4	4	79.78	75.6	4.18	5.53 %
3	T1	T14	4	87.44	61.72	25.72	41.67 %
4	T15	T7	3	55.53	53.78	1.75	3.25 %
5	T17	T6	4	79.66	66.47	13.19	19.84 %
6	T8	T17	6	162.76	56.33	106.43	188.94 %
7	T11	T2	3	74.76	65.71	9.05	13.77 %
8	T6	T1	4	72.02	55.74	16.28	29.21 %
9	T2	T12	9	205.11	87.64	117.47	134.04 %
10	T14	T16	9	191.35	88.38	102.97	116.51 %

Table 16 The result of ten navigations

In second part of this experiment, another set of seventeen random numbers are generated for the testing of navigation. Tag 4 (T4) is assumed as the origin position and Tag 11 (T11) is assumed as the destination. The NextTag.java (appendix 5) is executed to find the next tag to move to repeatedly until the output shows the next tag to move to is the destination tag. ***Figure 44*** presents the route of this navigation which is T4 → T7 → T3 → T6 → T15 → T6 → T3 → T16 → T10 → T5 → T14 → T11.

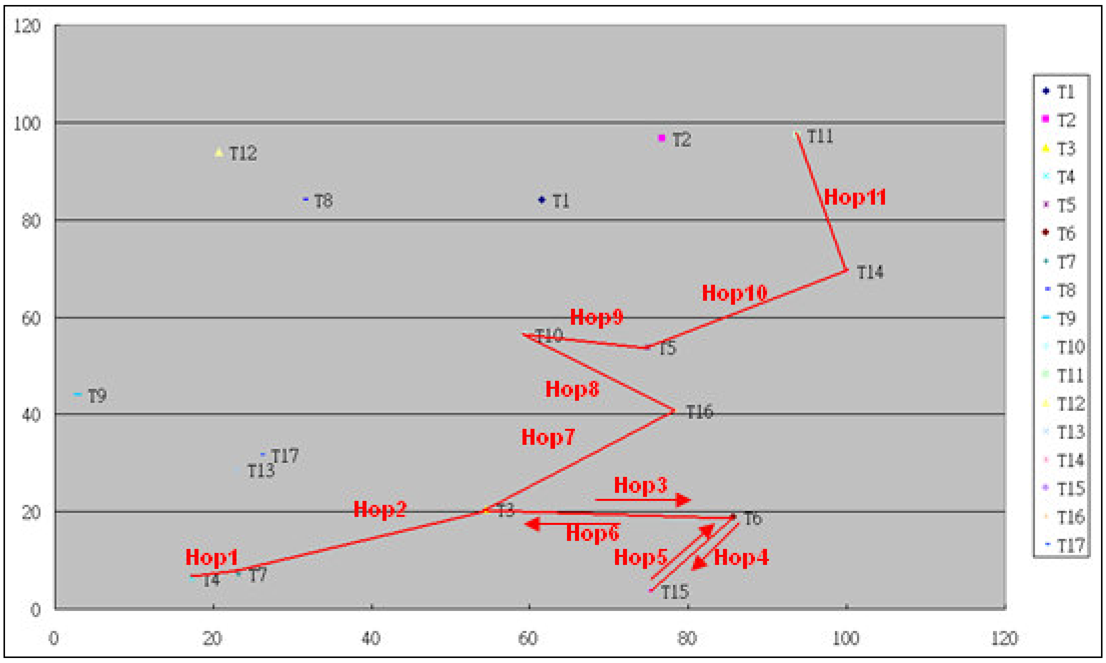


Figure 44 The hops in the navigation of T4 to T11

Table 17 presents all the tags are recorded in this navigation, the distance between each movement, the distance between T4 and T11, the difference between the total distance of the route and the actual distance between T4 and 11, and the percentage of the excess of the navigation. The equation is used to calculate the percentage of the excess of the navigation is same as the previous testing of navigation. In ***Table 17***, the percentage of excess is 126.42% which is higher than 100%, thus it means this navigation has too many unnecessary hops in the route which can be seen in ***Figure 44***.

	Movement
T4	0
T7	6.03
T3	33.99
T6	31.21
T15	18.48
T6	18.48
T3	31.21
T16	31.29
T10	24.67
T5	15.67
T14	29.61
T11	28.6
SUM	269.24
Actual Distance from T4 to T11	118.91
Difference	150.33
% EXCESS	126.42 %

Table 17 The movement to each Tag, and the sum and the accuracy of the navigation

Furthermore, the navigation will be tested ten more times and each test contains a different set of origin tag and destination. ***Table 18*** presents the origin tag, the destination, number of hop in the route, the sum of the movement, the actual distance between origin tag and destination, the difference between the sum of the movement and actual distance, and the percentage of the excess of each test for these ten navigations. In ***Table 18***, the lowest percentage of the excess is 0.78% which his in run number 9, thus the navigation route is very close to the actual distance. However, the highest percentage of the excess is 264.32% which his in run number 4 that has fifteen hops in its navigation. In run number 3 and 6, the number of hop are both 10 but the percentage of the excess is 201.50% for run number 6 which is much higher than the percentage of the percentage of the excess in run number 3 which is 148.35%. Furthermore, each of the navigation tests is successfully complete which means each test finds the route from the origin tag to the destination tag. However, the percentage of the excess distance is unstable which has already been mentioned above. According to this, efficiency of the path finding is also unstable because it is affected by the deployment of the RFID tags, the origin tag, the destination tag, the algorithm of finding the next tag to move to and the algorithm for back tracking avoidance.

Run Number	Origin Tag	Destination	Number of Hop in Route	SUM	Actual Distance	Difference	% EXCESS
1	T9	T14	6	182.38	100.29	82.09	81.85 %
2	T7	T2	8	232.19	104.23	127.96	122.77 %
3	T11	T7	10	284.19	114.43	169.76	148.35 %
4	T6	T12	15	361.41	99.2	262.21	264.32 %
5	T16	T4	4	118.6	70.13	48.47	69.11 %
6	T13	T2	10	261.16	86.62	174.54	201.50 %
7	T17	T11	11	287.87	94.22	193.65	205.53 %
8	T12	T15	8	221.49	105.35	116.14	110.24 %
9	T15	T9	3	83.51	82.86	0.65	0.78 %
10	T14	T4	5	139.21	104.07	35.14	33.77 %

Table 18 The result of ten navigations for second set of random number

Chapter 6: Discussion and Conclusion

6.1 Conclusion

As mentioned earlier (section 1.2) the objective of this study is to address the issue of location storage using RFID by using compression technology in order to successfully apply RFID technology in an indoor navigation system. From the experimental results, the distance data storage required for each tag has been reduced to half of the original size. After the reconstruction of the distance data, the first four data has percentage of the error in resolution are all zero which are the nearest neighbour distance data for each tag. As a result, it can be seen that the distance is stored efficiently when applying the wavelet compression technique. For the angle data, the quadrant compression technique has been applied on each of tag which the storage size has been reduced to one third of the original data size. After the reconstruction of the angle data, the average of the percentage of the error in resolution for the first four data from all tags is around 1% which is the nearest neighbour angle data for each tag. As a result, it can be seen that the angle has also been stored efficiently.

The navigation experiment involved developing an algorithm to use compressed data to navigate between data-containing tags. As a result, the algorithm is used to find the next tag to move to from the current tag and the method is used to avoid the back tracking are both successfully designed and implemented in this study. According to this, the goal of this study is achieved.

6.2 Discussion

The two experiments have been carried in the previous section: chapter 5 which have produced a number of results. However, there are several results which are not in accordance with the expectation in the beginning of this study. Accordingly, any of these differences and the findings from the results of experiments are discussed in this section.

The distance data set is examined in the first part of experiment 1 (section 5.1) which the wavelet compression technique is applied to reduce its storage size. It expects a smooth change in resolution and storage for the distance data set but the change of the resolution is not smooth in practice. As mentioned in the section 4.1, the amount of the coefficient is one less than the previous group in the calculation for each group. Thus, the resolution of the distance should be less and less from the closest to farthest. However, as shown in ***Table 9*** (page 65), the lowest value and highest value of the percentage of the error in resolution in the second close neighbour group are 0.24% and 21.31%. The lowest value and highest value of the percentage of the error in resolution in the third close neighbour group are 1.59% and 16.52%. According to this, the resolution for the distance resolution is not become less and less from closest to farthest neighbour group. It can be seen that the value of each coefficient in wavelet compression cannot be controlled because the equation to be used to calculate the coefficient is fixed. Therefore, the percentage of the error in resolution for each neighbour tag depends on the value of the missing coefficient of each group. I.e. the distribution of distances in the dataset will affect the result. In particular, if a group of points that have the same resolution are widely separated, the percentage error at the beginning and end of this group will be high, whereas the error of the central group will be small as the value calculated is close to the mean of the distance of this group. There is another issue need to be discussed for the distance data. In this study, the seventeen random location tags are generated for using in the experiment and I decided the first four neighbour tags are the highest resolution group (full resolution). However, if the total tags are 100, is four NN tags are for the navigation? What is the reasonable and number of NN tags for using to choose the next tag to move to in navigation? This should be reconsidered and redesigned carefully and logically, and may be domain-specific, for example in the case of a system used to map routes through doorways, a relatively lower number of NN may be appropriate, whereas in areas with higher tag density in particular areas e.g. finding objects in distinct clusters, a higher number may be needed.

The angle data set is examined in the second part of experiment 1 (section 5.1) which the quadrant compression technique is applied to reduce its storage size.. The change of the angle is as we expect which contains a smooth change in resolution and storage. When designing the storage system, of the angle data set, the number of the tags in the highest angle resolution needs to be the same as the number of tags in the highest distance resolution group. As a result, the number of the tags in the highest angle resolution is four. In this study, I decided the angles of all the tags are divided into three groups which are in the resolution of 8 bits, 4 bits and 1 bit. As ***Table 12*** (page 71) shows, the angle resolution becomes less and less from closest to farthest neighbour group which matches the concept of design of this study. However, if the amount of tags is 100, then there are many possible schemes to represent 100 tags. The angle resolution in each group and the number of group need to be reconsidered and redesigned while the amount of the tags is changed as well as while the number of tags in highest distance resolution group is changed in order to make the use of the storage space efficiently and make it meaningful for use in the navigation.

After the experiment of the distance and angle, the second experiment is focused on the navigation. As shown in ***Table 16*** (page 82) and ***Table 18*** (page 85) in section 5.2, the test for all the navigations are successful which means the destination in each navigation is reachable from the origin tag of each navigation. Furthermore, because of the geometry of the map in the experiment, it results in the navigation starting to back track. The avoidance of back tracking is one of the essential points in the design and algorithm of the navigation in this study because it stops the navigation repeating the permanent movement between two tags. As shown in ***Figure 44*** (page 83), the algorithm of the avoidance of the back tracking is successfully performed in this navigation system. However, if the amount of tags is different from this study, the algorithm of the avoidance of the back tracking for this study might not be suitable. For instance, if the amount of the tags is 100, its geometry of the map is more complex than this study (seventeen tags). As a result, if the same algorithm for this study is applied to an experiment of 100 tags, it might result in multiple loops or back-tracks around the tags which means the destination is unreachable. Thus, this kind of navigation system is ineffective in this case. According to this, the algorithm of the avoidance of the back tracking should be reconsidered for different experiment while the amount of the tags is changed. Furthermore, the actual angle of the destination tag and the mean values of tags in highest angle resolution group is also another factor which affects the back tracking. For instance, the value is used in the comparison in the navigation is the mean value of the angle range. If the destination is in the lowest angle resolution group of the current position tag, the mean values of the destination is either 90 or 270 depends on which side of the current position tag that the destination is in. It assumes that the actual angle of the destination is 10 and the mean values for tags in the highest angle resolution group are 15, 20, 100 and 120. As a result, after the comparison in the navigation, the system will choose the tag with the mean value of 100 for its next tag to move to which is much different from the comparison by using the actual value of destination. If this kind of situation appears several times in navigation, it will result in too much back tracking which means the percentage of excess on distance will be very high.

6.3 Strengths and Limitations

There are several strengths and limitations in this study. The simple navigation system is one of the strengths, thus mean this system is simple on its design and easy to use. Low cost to implement this navigation system is another strength because this navigation system only requires a RFID reader and a number of RFID tags.

There are several limitations in this study. For instance, the two sets of seventeen random distributed tags are used in the experiment, thus some particular map have not been generated (e.g. one tag is isolated with other sixteen tags) which the results might be biased. The amount of tags in the experiment is one of limitations, thus it means some algorithms might not be applicable. The back tracking can be seen as a limitation which increases the percentage of excess on distance in the navigation. Furthermore, this system has not been applied in real world, thus it becomes very difficult to predict whether the system can be effectively and accurately applied to the real case and it also might contain some factors from the environment which affects to the system.

6.4 Future research directions

As I discussed in the previous section (section 6.3), there are numbers of limitations that need to be addressed and further studies performed in the future. For instance, all kinds of particular maps need to be discussed and applied in the experiment in order to get the results in all kinds of situations. More experiment as well as more tags need to be involved in order to get more effective and accurate result. The compression method for the angle data and the algorithm for the avoidance of the back tracking are both need to be improved in order to make the total distance in the navigation is close to the actual distance. In particular the use of a 2d wavelet compression technique may be appropriate. Furthermore, the navigation system needs to be deployed in real world in order to solve the issues in the real world situation.

REFERENCES

Addison, P. S. (2002). *The illustrated wavelet transform handbook: applications in science, engineering, medicine and finance*. Bristol: Institute of Physics Publishing.

Answer.com, spherical coordinate system [Image] (n.d.). Retrieved August 16, 2008, from http://www.answers.com/topic/coordinate-system?cat=technology

Barnsley, M. F., & Jacquin, A. (1998). Application of recurrent iterated function systems to images. *Visual Comm. and Image Processing, Pro. SPIE 1001*(3), 122–131.

Bessa, M., Coelho, A., & Chalmers, A. (2004). *Alternate feature location for rapid navigation using a 3D map on a mobile device*. Paper presented at the 2004 3rd Proceedings of the International Conference on Mobile and Ubiquitous Multimedia, Maryland, USA.

Bessho, M., Kobayashi, S., Koshizuka, N., & Sakamura, K. (2007, August 19-22). *A pedestrian navigation system using multiple space-identifying devices based on a unique identifier framework*. Paper presented at 2007 International Conference on Machine Learning and Cybernetics, Hong Kong, China.

Calderbank, A. R., Daubechies, I., Sweldens, W., & Yeo, B .L. (1998). Wavelet transforms that map integers to integers. *Applied and Computational Harmonic Analysis*, *5*(3), 332-369.

Chen, B., Zhang, H. C., Cao, W. L., & Feng, J. H. (2007, August 19-22). *Huffman coding method based on number character*. Paper presented at the 2007 International Conference on Machine Learning and Cybernetics, Hong Kong, China.

Chen, C. Y., Pai, Y. T., & Ruan, S. J. (2006, November). *Low power Huffman coding for high performance data transmission*. Paper presented at the 2006 International Conference on Hybrid Information Technology, Cheju Island, South Korea.

Davis, G. M. (1998). A wavelet-based analysis of fractal image compression. *IEEE Transactions on Image Processing*, *7*(2), 141-154.

Feldmann, S., Kyamakya, K., Zapater, A., & Lue, Z. (2003). *An indoor Bluetooth-based positioning system: Concept, implementation and experimental evaluation.* Paper presented at the 2003 International Conference on Wireless Networks, Las Vegas, USA.

Gryazin, E. A., Krassi, B. A., & Tuominen, J. O. (2003, November 27-28). *WLAN technology for indoor positioning and navigation*. Paper presented at the 2003 4th International Science Conference on New Information Technologies: Development and Applications, Taganrog, Russia.

He, Y. J., Zhang, D. L., Shen, B., & Geng, L. F. (2007, October 22-25). *Implementation of fast Huffman decoding algorithm*. Paper presented at the 2007 7th International Conference on ASIC, Guilin, China.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly, 28*(1), 75-105.

Howell, D. C. (1997). *Statistical Methods for Psychology* (4th ed.). Boston: Duxbury Press.

Hub, A., Diepstraten, J., & Ertl, T. (2003). Design and development of an indoor navigation and object identification system for the blind. *ACM SIGACCESS Accessibility and Computing, 77-78*, 147-152.

Hub, A., Diepstraten, J., Ertl, T. (2005). *Augmented indoor modeling for navigation support for the blind*. Paper presented at the Proceedings of the 2005 International Conference on Computers for People with Special Needs, Las Vegas, Nevada, USA.

Jacquin, A. (1992). Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Trans. Image Processing*, 1, 18–30.

Jiang, J., Xia, J., & Xiao, G. (2006, April 6). MPEG-2 based lossless video compression. *IEEE Proceedings of Vision-Image and Signal Processing, 153*(2), 244-252.

Keys-Mathews, L. (1998). *The five themes of geography*. Retrieved August 10, 2008, from http://www2.una.edu/geography/statedepted/themes.html

Key, J. P. (1997). *Research design in occupational education*. Retrieved August 24, 2008, from http://www.okstate.edu/ag/agedcm4h/academic/aged5980a/5980/newpage2.htm

Kim, I., Lee, B., & Kim, H. (2006, Feb 20-22). *Privacy Protection based on User-defined Preferences in RFID System*. Paper presented at the 8th International Conference Advanced Communication Technology, Phoenix Park, Korea.

Kulyukin, V., Gharpure, C., Nicholson, J., & Pavithran, S. (2004). RFID in robot-assisted indoor navigation for the visually impaired. *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004*(2), 1979-1984.

Kulyukin, V., Gharpure, C., Nicholson, J., & Osborne, G. (2006). RobotAssisted wayfinding for the visually impaired in structured indoor environments. *Autonomous Robots, 21*, 29-41.

Luo, L., Li, J., Li, S., Zhuang, Z., & Zhang, Y. Q. (2001). *Motion compensated lifting wavelet and its application in video*. Paper presented at the 2001 Electronic Proceedings of IEEE International Conference on Multimedia and Expo, Tokyo, Japan.

Manzoor, U., & Ijaz, K. (2008, February 20-22). *Optimizing bandwidth usage and supporting block level synchronization in MultiSync: a multiagent system for ubiquitous file synchronization Source*. Paper presented at 2008 7th Proceedings of the WSEAS International Conference on Artificial intelligence, knowledge engineering and data bases, Cambridge, UK.

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support systems, 15*(5), 251-266.

Marks, S. K. (2000). *Joint source/channel coding for mobile audio streaming*. PhD Thesis. University of Melbourne.

Marshall, D. (2001). *The MPEG video bitstream*. Retrieved August 10, 2008, from http://www.cs.cf.ac.uk/Dave/Multimedia/node255.html

Mehmood, M. A., Kulik, L., & Tanin, E. (2007). Navigation and interaction in physical spaces using RFID enabled spatial sensing. *The Proceedings of the 5th international conference on Embedded networked sensor systems*, *5*, 379-380.

Milella, A., Vanadia, P., Cicirelli, G., & Distante, A. (2007, September 4-7). *RFID-based environment mapping for autonomous mobile robot applications*. Paper presented at 2007 IEEE/ASME international conference on Advanced intelligent mechatronics, ETH Zurich, Switzerland.

Minos, G., & Phillip, B. G. (2004). Probabilistic wavelet synopses. *ACM Transactions on Database Systems*, *29*(1), 43-90.

Moffat, A. (1990). Design and natural science research on information technology. *IEEE Transactions on Communications, 38*(11), 1917-1921.

Nelson, M. (1989). Implementing the PPM data compression scheme. *Decision Support systems, 15*(5), 251-266..

Nunamaker, J., Chen, M., & Purdin, T. D. M. (1990). Systems development in information systems research. *Journal of Management Information Systems, 7*(3), 89-106.

Olivier, C. (n.d.). *IP cores for accelerating JPEG2000*. Retrieved August 15, 2008, from http://www.us.design-reuse.com/articles/6691/ip-cores-for-accelerating-jpeg2000.html

Poobal, S., & Ravindran, G. (2005, May 13). *Analysis on the effect of tolerance criteria in fractal image compression*. Paper presented at the 2005 IEEE International Workshop on Imaging Systems and Techniques, Niagara Falls, Ontario, Canada.

Ramaswamy, V. N., Namuduri, K. R., & Ranganathan, N. (1996). Lossless image compression using wavelet decomposition. *13th International Conference on Pattern Recognition, ICPR'96*(3), 924.

Renaudin, V., Yalak, O., Tomé, P., & Merminod, B. (2007). *Indoor navigation of emergency agents*. Retrieved October 24, 2008, from http://infoscience.epfl.ch

Sayood, K. (2005). *Introduction to data compression* (3rd ed.). San Fransisco, USA: Morgan Kaufmann.

Schafer, R. (1998). MPEG-4: a multimedia compression standard for interactive applications and services. *Electronics & Communication Engineering Journal*, *10*(6), 253-262.

Toshifumi, T. (2005). World map based on RFID tags for indoor mobile robots. *2005 Proceedings of the International Society for Optical Engineering, 6006*, 600613.1-600613.8.

VIAS Encyclopedia (2005). *Huffman coding*. Retrieved August 10, 2008, from http://www.vias.org/encyclopedia/huffman_coding.html

Wang, H., Wu, Q., He, X. J., & Hintz, T. (2006, August 30-01). A novel interactive progressive decoding method for fractal image compression. *2006 First International Innovative Computing-Information and Control, 3*, 613-617.

Weisstein, E. W. (n.d.). *Wavelet*. Retrieved August 15, 2008, from http://mathworld.wolfram.com/Wavelet.html

Welch, T. A. (1984). A technique for high-performance data compression. *IEEE Computer, 17*(6), 8–19.

Wikipedia , Cartesian with grid [Image] (2008). Retrieved August 16, 2008, from http://en.wikipedia.org/wiki/Image:Cartesian_with_grid.svg

Willis, S., & Helal, S. (2004). *A passive RFID information grid for location and proximity sensing for the blind user*. Retrieved September 27, 2008, from http://www.cise.ufl.edu/submit/files/file_355.pdf

Wu, Y., Lonardi, S., & Szpankowski, W. (2006, March 28-30). *Error-resilient LZW data compression*. Paper presented at the Proceedings of the DCC 2006 Data Compression Conference, Utah, USA.

Zhou, Y., Liu, W., & Huang, P. (2007, April 10-14). *Laser-activated RFID-based indoor localization system for mobile robots*. Paper presented at the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy.

APPENDICES

Page

Appendix 1: JAVA CODE – CalAngleBinary.java

```
import java.util.*;
import java.io.*;
public class CalAngleBinary
{       public static void main(String[] args) {
                double [] anglArray = new double [16];
                int [] anglRanArray = new int [16];
                String [] anglBinArray = new String [16];
                String element;
                int level1Num = 4;
                int level2Num = 4;
                int level3Num = 8;
                int level1Binary = 8;
                int level2Binary = 4;
                int level3Binary = 1;
                int count = 0;
                File fIn = new File("C:\\D_files\\Calculations_Angle\\ran1\\Tag1.txt");
                File fOut = new File("C:\\D_files\\Angle_Binary\\ran1\\Tag1.txt");
                try
                {       FileReader fReader = new FileReader(fIn);
                        BufferedReader bReader = new BufferedReader(fReader);

                        while ((element = bReader.readLine()) != null)
                        {       anglArray[count] = Double.parseDouble(element);
                                count++;}
                        bReader.close();}
                catch (Exception e)
                {       System.err.println ("Error reading from file");}
                count = 0;

                for (int i = 0; i < level1Num; i++)
                {       anglRanArray[count] = AngleRange(anglArray[count], level1Binary);
                        anglBinArray[count] = AngleBinary(anglRanArray[count]-1, level1Binary);
                        count++;}

                for (int i = 0; i < level2Num; i++)
                {       anglRanArray[count] = AngleRange(anglArray[count], level2Binary);
                anglBinArray[count] = AngleBinary(anglRanArray[count]-1, level2Binary);
                count++;}

        for (int i = 0; i < level3Num; i++)
        {       anglRanArray[count] = AngleRange(anglArray[count], level3Binary);
                anglBinArray[count] = AngleBinary(anglRanArray[count]-1, level3Binary);
                count++;}
        try
        {       FileWriter fWriter = new FileWriter(fOut);
                BufferedWriter bWriter = new BufferedWriter(fWriter);

                for(int i = 0; i < anglBinArray.length; i++)
                {       bWriter.write(anglBinArray[i]);
                        bWriter.newLine();}
                bWriter.close();}
        catch (Exception e)
        {       System.err.println ("Error writing to file");}
}

public static int AngleRange(double currentAngle, int numBits) {
    double ranges = Math.pow(2, numBits);
    double rangeVaule = 360 / ranges;
    double checkRange;
    int anglRange;
    checkRange = 0;
    anglRange = 0;

    for (int i = 0; i < ranges; i++)
    {       if (currentAngle > checkRange && currentAngle < (checkRange + rangeVaule))
            {       anglRange = i + 1;
                    i = (int)ranges;}
            else if (currentAngle == checkRange)
            {       anglRange = i + 1;
                    i = (int)ranges;}
            checkRange += rangeVaule;}
            return anglRange;}

public static String AngleBinary(int angle, int length) {
    String temp = Integer.toBinaryString(angle); // change decimal number to binary
```

```
        String temp2 = "0";
        int tempLength = length - temp.length(); // check how many bits miss from the original
binary number
        // add missing bits (0) in front of the binary number
        for (int k = 0; k < tempLength; k++)
        {       temp = temp2 + temp;
        }
                return temp;
   }
```

Appendix 2: JAVA CODE – RandomMap.java

```
import java.util.*;
import java.text.*;
import java.io.*;

public class RandomMap{
        public static void main(String[] args){
                double[] xValue = new double[17];
                double[] yValue = new double[17];
                int[] lableValue = new int[17];
                double x_ranNum = 0;
                double y_ranNum = 0;
                String newFormatX, newFormatY;
                int check = 0;
                DecimalFormat df = new DecimalFormat("##0.00");
                File fOut = new File("C:\\D_files\\ranTags15.txt");

                for (int i = 0; i < xValue.length; i++) {
                        while (check == 0){
                                x_ranNum = Math.random() * 100;
                                newFormatX = df.format(x_ranNum);
                                x_ranNum = Double.parseDouble(newFormatX);
                                y_ranNum = Math.random() * 100;
                                newFormatY = df.format(y_ranNum);
                                y_ranNum = Double.parseDouble(newFormatY);
                                if (i == 0)
                                {       check = 1;}
                                else {          check = Search(xValue, yValue, x_ranNum, y_ranNum, i + 1);}
                        }
                        check = 0;
                        xValue[i] = x_ranNum;
                        yValue[i] = y_ranNum;
                        lableValue [i] = i + 1;}

                // write the values of two arrays into file ranTags.txt
                try
                {       FileWriter fWriter = new FileWriter(fOut);
```

```
                    BufferedWriter bWriter = new BufferedWriter(fWriter);
                    bWriter.write("X            Y            Label");
                    bWriter.newLine();

                    for(int k = 0; k < xValue.length; k++)
                    {       newFormatX = df.format(xValue[k]);
                            newFormatY = df.format(yValue[k]);
                            bWriter.write(newFormatX + "\t" + newFormatY + "\t" + lableValue[k]);
                            bWriter.newLine();}
                bWriter.close();}
            catch (Exception e)
            {       System.err.println ("Error writing to file");}
        }

        // check any duplicate point
        public static int Search(double[] xArray, double[] yArray, double xKey, double yKey, int
currentLength) {for (int j = 0; j < currentLength; j++) {
                    if (xKey == xArray[j] && yKey == yArray[j])
                    {       return 0;}
                    else{}
        }
            return 1;
        }
}
```

Appendix 3: JAVA CODE – WaveletConstruction.java

```
import java.util.*;
import java.io.*;
import java.text.*;

public class WaveletConstruction
{       public static void main(String[] args) {
       ArrayList disArray = new ArrayList();
        ArrayList coeffiArray = new ArrayList();
        int count, tagNum;
        int count2 = 0;
        String element;
        File fIn = new File("C:\\D_files\\Calculations\\ran1\\Tag1.txt");
        File fOut = new File("C:\\D_files\\Distance_Coefficient\\ran1\\Tag1.txt");

        try
                {       FileReader fReader = new FileReader(fIn);
                        BufferedReader bReader = new BufferedReader(fReader);
                        while ((element = bReader.readLine()) != null)
                        {       if (count2 == 0)
                                {       count2++;}
                                else{ String temp [] = element.split("\\t");
                                        disArray.add(temp[0]); }
                        }
                    bReader.close();
        }
        catch (Exception e)
        {       System.err.println ("Error reading from file");}

        // calculate wavelet coefficient
        coeffiArray = constructAverage(disArray);
        try
        {       FileWriter fWriter = new FileWriter(fOut);
                BufferedWriter bWriter = new BufferedWriter(fWriter);
                bWriter.write((String)coeffiArray.get(0));
                bWriter.newLine();
```

```
            bWriter.write((String)coeffiArray.get(1));
            bWriter.newLine();
            bWriter.write((String)coeffiArray.get(2));
            bWriter.newLine();
            bWriter.write((String)coeffiArray.get(3));
            bWriter.newLine();
            bWriter.write((String)coeffiArray.get(4));
            bWriter.newLine();
            bWriter.write((String)coeffiArray.get(5));
            bWriter.newLine();
            bWriter.write((String)coeffiArray.get(8));
            bWriter.newLine();
            bWriter.write((String)coeffiArray.get(9));
            bWriter.newLine();
            bWriter.close();}
      catch (Exception e)
      {     System.err.println ("Error writing to file");}
}

public static ArrayList constructAverage(ArrayList alDataItems){
   ArrayList arrCoefficients = new ArrayList();
   ArrayList alTemp = new ArrayList();
   double iWave1, iWave2, iDiff, iAverage = 0;
   int iIndex = 0, iLength = alDataItems.size(), iPos;

   /* while the list length is greater than one */
   while(alDataItems.size() > 1){
      iIndex = 0;
      iPos = 0;

      /* While not the end of the current list */
      while(iIndex < iLength){
         /* Get the numbers of the values to average */
         iWave1 = Double.parseDouble((String)alDataItems.get(iIndex));
         iIndex++;
         iWave2 = Double.parseDouble((String)alDataItems.get(iIndex));
         iIndex++;
         /********************************************************************/
         /* Calculate the difference and the average */
```

```
            iAverage = (iWave1 + iWave2)/2;
            iDiff = iWave1 - iAverage;
            iDiff = Math.round(iDiff * 10000) / 10000d;
            /* Add Difference between values to coefficent array */
            arrCoefficients.add(iPos, Double.toString(iDiff));
            /* Add average to new list */
            alTemp.add(Double.toString(iAverage));
            iPos++;}

          /* If there is an uneven number of numbers add the last one to the end */
          if((alDataItems.size()%2)!= 0){
            alTemp.add(alDataItems.get(iIndex));}

          /* Replace old list with new list */
          alDataItems.clear();
          alDataItems = (ArrayList)alTemp.clone();
          iLength = alDataItems.size();
          alTemp.clear();}
        iAverage = Math.round(iAverage * 10000) / 10000d;

        /* Add Final average value to front of co-efficient list */
        arrCoefficients.add(0, Double.toString(iAverage));

        return arrCoefficients;
    }
}
```

Appendix 4: JAVA CODE – Calculations.java

```
import java.util.*;
import java.io.*;
import java.text.*;

public class Calculations
{	public static void main(String[] args) {
	ArrayList arrayX = new ArrayList();
	ArrayList arrayY = new ArrayList();
	int count, tagNum;
	int count2 = 0;
	String element;
	String fileFrontNam = "C:\\D_files\\Calculations\\ran1\\Tag";
	String fileFrontNam2 = "C:\\D_files\\Calculations_Distance\\ran1\\Tag";
	String fileFrontNam3 = "C:\\D_files\\Calculations_Angle\\ran1\\Tag";
	String fileFrontNam4 = "C:\\D_files\\Calculations_Label\\ran1\\Tag";
	String fileMidNam;
	String fileEndNam = ".txt";
	String fileName, fileName2, fileName3, fileName4;
	File fIn = new File("C:\\D_files\\ranTags1.txt");
	try
		{	FileReader fReader = new FileReader(fIn);
			BufferedReader bReader = new BufferedReader(fReader);

			while ((element = bReader.readLine()) != null)
			{	if (count2 == 0)
				{	count2++;}
				else{ String temp [] = element.split("\\t");
					arrayX.add(temp[0]);
					arrayY.add(temp[1]); }
			}
		   bReader.close();}
		catch (Exception e)
		{System.err.println ("Error reading from file");}
```

```
        // calculations
        String [] tempArrayX = (String[])arrayX.toArray(new String[arrayX.size()]);
        String [] tempArrayY = (String[])arrayY.toArray(new String[arrayX.size()]);
        final double[] disCalResults = new double[arrayX.size()-1];
        double[] anglCalResults = new double[arrayX.size()-1];
        int[] labeCalResults = new int[arrayX.size()-1];
        for (int i = 0; i < arrayX.size(); i++)
        {       count = 0;
                for (int j = 0; j < arrayX.size(); j++)
                {       if (i != j)
                        {       disCalResults [count] = Distance(tempArrayX[i], tempArrayY[i],
    tempArrayX[j], tempArrayY[j]);
                                anglCalResults [count] = Angle(tempArrayX[i], tempArrayY[i],
    tempArrayX[j], tempArrayY[j]);
                                labeCalResults [count] = j + 1;
                                count++;}
                }
                Integer [] sortOrder = new Integer[disCalResults.length];
                for(int k=0; k <sortOrder.length; k++){
                    sortOrder[k] = k;}

                // sort by distance
                Arrays.sort(sortOrder,new Comparator<Integer>() {
                    public int compare(Integer a, Integer b){
                                 if(disCalResults[b]<disCalResults[a]){
                                       return 1; }
                                if(disCalResults[b]>disCalResults[a]){
                                       return -1; }
                                return 0; }});
                tagNum = i + 1;
                fileMidNam = Integer.toString(tagNum);
                fileName = fileFrontNam + fileMidNam + fileEndNam;
                fileName2 = fileFrontNam2 + fileMidNam + fileEndNam;
                fileName3 = fileFrontNam3 + fileMidNam + fileEndNam;
                fileName4 = fileFrontNam4 + fileMidNam + fileEndNam;

                File fOut = new File(fileName);
                File fOut2 = new File(fileName2);
                File fOut3 = new File(fileName3);
```

```
            File fOut4 = new File(fileName4);
            String rr;
            try
            {       FileWriter fWriter = new FileWriter(fOut);
            BufferedWriter bWriter = new BufferedWriter(fWriter);
                    FileWriter fWriter2 = new FileWriter(fOut2);
            BufferedWriter bWriter2 = new BufferedWriter(fWriter2);
                    FileWriter fWriter3 = new FileWriter(fOut3);
            BufferedWriter bWriter3 = new BufferedWriter(fWriter3);
                    FileWriter fWriter4 = new FileWriter(fOut4);
            BufferedWriter bWriter4 = new BufferedWriter(fWriter4);
                    bWriter.write("Distance             Angle                 Label");
                    bWriter.newLine();

                    for(int m = 0; m < anglCalResults.length; m++)
                    {       bWriter.write(disCalResults[sortOrder[m]] + "\t\t" +
anglCalResults[sortOrder[m]] + "\t\t" + labeCalResults[sortOrder[m]]);
                            bWriter.newLine();
                            rr = Double.toString(disCalResults[sortOrder[m]]);
                            bWriter2.write(rr);
                            bWriter2.newLine();
                            rr = Double.toString(anglCalResults[sortOrder[m]]);
                            bWriter3.write(rr);
                            bWriter3.newLine();
                            rr = Integer.toString(labeCalResults[sortOrder[m]]);
                            bWriter4.write(rr);
                            bWriter4.newLine();}
                bWriter.close();
                bWriter2.close();
                    bWriter3.close();
                    bWriter4.close();}
            catch (Exception e)
            {       System.err.println ("Error writing to file");}
    }
  }

  public static double Distance(String curX, String curY, String tarX, String tarY) {
            DecimalFormat df = new DecimalFormat("##0.00");
            String newFormat;
```

```
                double tempCurX = Double.parseDouble(curX);
                double tempCurY = Double.parseDouble(curY);
                double tempTarX = Double.parseDouble(tarX);
                double tempTarY = Double.parseDouble(tarY);
                double calResult;

                calResult = Math.sqrt(Math.pow((tempTarX - tempCurX),2) + Math.pow((tempTarY -
tempCurY),2));
                newFormat = df.format(calResult);
                calResult = Double.parseDouble(newFormat);
                return calResult; }

   public static double Angle(String curX, String curY, String tarX, String tarY) {
                double tempCurX = Double.parseDouble(curX);
                double tempCurY = Double.parseDouble(curY);
                double tempTarX = Double.parseDouble(tarX);
                double tempTarY = Double.parseDouble(tarY);
                DecimalFormat df = new DecimalFormat("##0.00");
                String newFormat;
                double disX = tempTarX - tempCurX;
             double disY = tempTarY - tempCurY;
             double degrees = 0.0d;

             // Calculate angle
             if (disX == 0.0)
             {     if (disY > 0.0)
                          degrees = 0.0;
                    else if (disY < 0.0)
                               degrees = 180.0; }
             else if (disY == 0.0)
             {      if    (disX > 0.0)
                          degrees = 90.0;
                    else if (disX < 0.0)
                    {   degrees = 270.0; }
             }
                else
             {         if    (disX < 0.0){
                          degrees = Math.atan(disY/disX) + Math.PI;
                               // convert to degrees
                               degrees = degrees * 180 / Math.PI;
                               degrees = 450 - degrees; }
                    else if (disY < 0.0){
                          degrees = Math.atan(disY/disX) + (2*Math.PI);

                               // convert to degrees
                               degrees = degrees * 180 / Math.PI;
                               degrees = 450 - degrees; }
                    else{
                          degrees = Math.atan(disY/disX);

                               // convert to degrees
                               degrees = degrees * 180 / Math.PI;
                               degrees = 90 - degrees; }}

                newFormat = df.format(degrees);
                degrees = Double.parseDouble(newFormat);
                return degrees;
   }
}
```

Appendix 5: JAVA CODE – NextTag.java

```
import java.util.*;
import java.io.*;
import java.text.*;

public class NextTag
{       public static void main(String[] args) {
                ArrayList angBinArray = new ArrayList();
                ArrayList tagLabel = new ArrayList();
                ArrayList dupArray = new ArrayList();
                String desTagLabel = "";
                String currTagLabel = "";

                // Including the current tag (the last position of this array)
                String routeArray [] = {"","","","","","","","","","","",""};
                String tempValue, element;
                int count = 0;
                int check = 0;
                int found = 0;
                int nextTag = -1;
                double desXValue, desYValue, currXValue, currYValue;
                double compareAngle = 0.00d;
                double calValue;
                double minValue = 0.00d;
                int min = 0;
                int tempCheck = 0;
                int addCheck = 0;
                int checkNN = 0;
                double tempLow,tempHigh;

                // set up an array to record duplicated tags in route
                for (int i = 0; i < routeArray.length; i++)
                {       for (int j = 0; j < routeArray.length; j++)
                        {       if (i != j && routeArray[i].equals(routeArray[j]))
                                {       // if the tag is already existing in dupArray, don't record this tag in
checkArray
                                        for (int k = 0; k < dupArray.size(); k++)
                        {       if ((i + 1) < routeArray.length)
                                {       if (routeArray[i+1].equals((String)dupArray.get(k)))
                                        {       addCheck = 1; }}
                                if ((i + 1) >= routeArray.length)
                                {       addCheck = 1; }}
                        if (addCheck != 1)
                        {       dupArray.add(routeArray[i+1]); }
                        addCheck = 0; }}
}

String fileName = "C:\\D_files\\Angle_Binary\\ran2\\Tag";
String fileMidName = currTagLabel;
String fileEndName = ".txt";
String fileName2 = "C:\\D_files\\Storing_Tag_Labels\\ran2\\Tag";
fileName += fileMidName + fileEndName;
fileName2 += fileMidName + fileEndName;
File fIn = new File(fileName);
File fIn2 = new File(fileName2);
try
        {       FileReader fReader = new FileReader(fIn);
        BufferedReader bReader = new BufferedReader(fReader);
                FileReader fReader2 = new FileReader(fIn2);
        BufferedReader bReader2 = new BufferedReader(fReader2);
                while ((element = bReader.readLine()) != null)
                {       angBinArray.add(element); }

                while ((element = bReader2.readLine()) != null)
                {       tagLabel.add(element); }
                bReader.close();
                bReader2.close();}
        catch (Exception e)
        {       System.err.println ("Error reading from file");}

        // Remove the tag in highest angle resolution that has already been in the route
        for (int i = 0; i < dupArray.size(); i++)
        {       for (int j = 0; j < 4; j++)
                {       if (((String)dupArray.get(i)).equals((String)tagLabel.get(j)))
                        {       tagLabel.remove(j);
                                angBinArray.remove(j);
```

```
                              checkNN++;}}
                  }
                  // Two strings for storing low angle range value and high angle range value
                  String [] rangArray1 = new String [angBinArray.size()];
                  String [] rangArray2 = new String [angBinArray.size()];

                  // Convert binary numbe of angle into angle range and load values of range into
two arrays
                  for (int i = 0; i < angBinArray.size(); i++)
                  {       String tempRange [] = reconRange((String)angBinArray.get(i)).split(" ");
                          rangArray1 [i] = tempRange [0];
                          rangArray2 [i] = tempRange [1]; }
                  double [] meanArray = new double [4 - checkNN];

                  // calaulate the mean for each tag in the highest angle resolution group
                  for (int i = 0; i < meanArray.length; i++)
                  {meanArray [i] = (Double.parseDouble(rangArray1[i]) +
Double.parseDouble(rangArray2[i])) / 2; }
                  count = 0;

                  // if all the NN have already in the route, the destination cannot be reached (no
next tag to move to)
                  if (checkNN == 4)
                  {       System.out.println("All the NN tags in the highest angle resolution group
have been in the route.");
                          System.out.println("---The destination cannot be reached---");}
                  else{
                          // first check whether the destination is in the highest angle resolution
group
                          // if it is, just go to that tag and then finished.
                          while (check != 1)
                          {       if (desTagLabel.equals((String)tagLabel.get(count)))
                                  {       compareAngle = (Double.parseDouble(rangArray1 [count]) +
Double.parseDouble(rangArray2 [count])) / 2;
                                          if (count < (4 - checkNN)) //in the highest angle resolution
group
                                          {       found = 1;
                                                  nextTag = count;
                                                  check = 1; }}
```

```
					if (count == rangArray1.length - 1)
					{	check = 1; }
					count++;}
				count = 0;
				check =0;
				if (found == 1)
				{	System.out.println("The next tag to move to is destination: Tag "
+desTagLabel); }
				else{ if (compareAngle <= 180)
					{	// check any angle of tag is also smaller than 180
						for (int i = 0; i < meanArray.length; i++)
						{	if (meanArray[i] < 180)
							{	nextTag = i;
								count++;}}
						if (count == 1)
						{	System.out.println("The next tag to move to is: Tag " +
tagLabel.get(nextTag)); }
						else if (count > 1)
						{	check = 1; }
						else if (count == 0)
						{	for (int i = 0; i < meanArray.length; i++)
							{ if (i == 0) // Assign the initial value for min and minValue
								{	min = 0;
									minValue = Math.abs(meanArray[i] -
compareAngle);
									/* when minValue is greater than 180, the
calculation for angle should
									calculate from current calcaulation tag to
compareAngle (calculate anti-clockwise)*/
									if (minValue > (compareAngle + 180))
									{	minValue = 360 - minValue; }}
								else{ calValue = Math.abs(meanArray [i] -
compareAngle);
									/* when minValue is greater than 180, the
calculation for angle should
									calculate from current calcaulation tag to
compareAngle (calculate anti-clockwise)*/
									if (calValue > (compareAngle + 180))
```

```
                                        {       calValue = 360 - calValue; }

                                        if (calValue < minValue)
                                        {       min = i;
                                                minValue = calValue; }}
                                }
                                System.out.println("The next tag to move to is: Tag " +
tagLabel.get(min));
                        } // end of else if (count == 0)
                } // end of if (compareAngle < 180)
                if (compareAngle > 180)
                {       // check any angle of tag is also greater than 180
                        for (int i = 0; i < meanArray.length; i++)
                        {       if (meanArray[i] > 180)
                                {       nextTag = i;
                                        count++;}}
                        if (count == 1)
                        {       System.out.println("The next tag to move to is: Tag " +
tagLabel.get(nextTag)); }
                        else if (count > 1)
                        {       check = 1; }
                        else if (count == 0)
                        {       for (int i = 0; i < meanArray.length; i++)
                                {       if (i == 0) // Assign the initial value for min and
minValue
                                        {       min = 0;
                                                minValue = Math.abs(meanArray[i] -
compareAngle);

                                                /* when minValue is greater than 180, the
calculation for angle should
                                                calculate from current calcaulation tag to
compareAngle (calculate anti-clockwise)*/
                                                if (minValue > (meanArray[i] + 180))
                                                {       minValue = 360 - minValue; }
                                        }
                                        else{ calValue = Math.abs(meanArray[i] -
compareAngle);
```

```
                                                    /* when minValue is greater than 180, the
calculation for angle should
                                                    calculate from current calcaulation tag to
compareAngle (calculate anti-clockwise)*/
                                                    if (calValue > (meanArray[i] + 180))
                                                    {       calValue= 360 - calValue; }
                                                    if (calValue < minValue)
                                                    {       min = i;
                                                            minValue = calValue; }}
                                            }
                                            System.out.println("The next tag to move to is: Tag " +
tagLabel.get(min)); } // end of else if (count == 0)
                                } // end of if (compareAngle > 180)
                                if (check == 1)
                                {       for (int i = 0; i < meanArray.length; i++)
                                        {       if (i == 0) // Assign the initial value for min and minValue
                                                {       min = 0;
                                                        minValue = Math.abs(meanArray[i] -
compareAngle); }
                                                else{ calValue = Math.abs(meanArray[i] -
compareAngle);
                                                        if (calValue < minValue)
                                                        {       min = i;
                                                                minValue = calValue; }}
                                        }
                                        System.out.println("The next tag to move to is: Tag " +
tagLabel.get(min)); }}}
  }
  /*******************************************/
  /* Reconstrcture angle ranges function */
  /*******************************************/
  public static String reconRange(String binary)
  {   int length = binary.length(); ////this one is for the length of the string
      double ranges = Math.pow(2, length);
      double rangeValue = 360 / ranges;
      String tempRange;
      DecimalFormat df = new DecimalFormat("##0.00");
      String newFormat;
      String newFormat2;
```

```
        int decimal = Integer.parseInt(binary,2); // change binary number to decimal
        double rangeStart = decimal * rangeValue;
        double rangeEnd = rangeStart + rangeValue;
        newFormat = df.format(rangeStart);
        newFormat2 = df.format(rangeEnd);
        tempRange = newFormat + " " + newFormat2;
        return tempRange;}
}
```

Appendix 6: JAVA CODE – NavigationOutput.java

```
import java.util.*;
import java.io.*;
import java.text.*;

public class NavigationOutput
{       public static void main(String[] args) {
                ArrayList coefArray = new ArrayList();
                ArrayList angBinArray = new ArrayList();
                ArrayList labelArray = new ArrayList();
                String element, element2, element3;
                File fIn = new File("C:\\D_files\\Distance_Coefficient\\ran1\\Tag1.txt");
                File fIn2 = new File("C:\\D_files\\Angle_Binary\\ran1\\Tag1.txt");
                File fIn3 = new File("C:\\D_files\\Calculations_Label\\ran1\\Tag1.txt");
                File fOut = new File("C:\\D_files\\Storing_files\\ran1\\Tag1.txt");
                File fOut2 = new File("C:\\D_files\\Navigation_Output_files\\ran1\\Tag1.txt");
                File fOut3 = new File("C:\\D_files\\Storing_Tag_Labels\\ran1\\Tag1.txt");
                try
                        {       FileReader fReader = new FileReader(fIn);
                        BufferedReader bReader = new BufferedReader(fReader);
                                FileReader fReader2 = new FileReader(fIn2);
                        BufferedReader bReader2 = new BufferedReader(fReader2);
                                FileReader fReader3 = new FileReader(fIn3);
                        BufferedReader bReader3 = new BufferedReader(fReader3);
                                while ((element = bReader.readLine()) != null)
                                {       coefArray.add(element);}
                                while ((element = bReader2.readLine()) != null)
                                {       angBinArray.add(element); }
                                while ((element = bReader3.readLine()) != null)
                                {       labelArray.add(element); }
                                bReader.close();
                                bReader2.close();
                                bReader3.close();}
                        catch (Exception e)
                        {       System.err.println ("Error reading from file");}
                        String [] tempCoefArray = (String[])coefArray.toArray(new
String[coefArray.size()]);
                        String [] tempAngArray = (String[])angBinArray.toArray(new
String[angBinArray.size()]);
                        String [] tempLabeArray = (String[])labelArray.toArray(new
String[labelArray.size()]);
                        String [] actuCoefArray = new String [tempAngArray.length];
                        String [] naviDisOutput = new String [tempAngArray.length];
                        String [] naviAngOutput = new String [tempAngArray.length];
                        for (int i = 0; i < actuCoefArray.length; i++)
                        {       actuCoefArray [i] = "0";}
                        actuCoefArray [0] = tempCoefArray [0];
                        actuCoefArray [1] = tempCoefArray [1];
                        actuCoefArray [2] = tempCoefArray [2];
                        actuCoefArray [3] = tempCoefArray [3];
                        actuCoefArray [4] = tempCoefArray [4];
                        actuCoefArray [5] = tempCoefArray [5];
                        actuCoefArray [8] = tempCoefArray [6];
                        actuCoefArray [9] = tempCoefArray [7];
                        naviDisOutput = reconstructionAll(tempAngArray.length, actuCoefArray);
                        for (int i = 0; i < naviAngOutput.length; i++)
                        {       naviAngOutput[i] = reconRange(tempAngArray[i]); }
                        try
                        {       FileWriter fWriter = new FileWriter(fOut);
                        BufferedWriter bWriter = new BufferedWriter(fWriter);
                                FileWriter fWriter2 = new FileWriter(fOut2);
                        BufferedWriter bWriter2 = new BufferedWriter(fWriter2);
                                FileWriter fWriter3 = new FileWriter(fOut3);
                        BufferedWriter bWriter3 = new BufferedWriter(fWriter3);
                                bWriter.write(" Coefficient       |       Angle (Binary)        Tag Label");
                                bWriter.newLine();
                                for (int i = 0; i < tempAngArray.length; i++)
                                {       if (i < tempCoefArray.length    )
                                        {       bWriter.write(" " +tempCoefArray[i] + "    \t|" + "        " +
tempAngArray[i] + "    \t\t" + tempLabeArray[i]);
                                            bWriter.newLine();}
                                        else{ bWriter.write("\t\t|" + "        " + tempAngArray[i] + "    \t\t" +
tempLabeArray[i]);
                                            bWriter.newLine();}}
                                bWriter2.write("Distance             Angle Range                  Tag Label");
                                bWriter2.newLine();
                                for(int i = 0; i < tempAngArray.length; i++)
                                {       bWriter2.write(naviDisOutput[i] + "\t\t" + naviAngOutput[i] + "\t\t"
+tempLabeArray[i]);
                                        bWriter2.newLine();
                                        bWriter3.write(tempLabeArray[i]);
                                        bWriter3.newLine();}
                                bWriter.close();
                                bWriter2.close();
                                bWriter3.close();}
                        catch (Exception e)
                        {       System.err.println ("Error writing to file");}
            }
            /******************************************/
            /* Reconstrcture distance values function */
            /******************************************/
            public static String[] reconstructionAll(int iNumNodes, String[] coefficientValues){
                ArrayList reconstrucValue = new ArrayList();
                double calcuTemp, tempValue;
                int calcuLevel, leftHalf, rightHalf, calcuCoeffi, calcuChek, temp1, count;
                DecimalFormat df = new DecimalFormat("##0.00");
                String newFormat;
                calcuTemp = Math.log(iNumNodes)/Math.log(2) - 1;
                calcuLevel = (int)calcuTemp;
                leftHalf = iNumNodes / 2;
                rightHalf = iNumNodes / 2;
                int [] coefNum = new int [calcuLevel];
                int [] calcuSymbol = new int [calcuLevel];
                int [] checkNum = new int [calcuLevel];
                temp1 = calcuLevel;

                /* assign value of coefNum and checkNum array*/
                for(int i = 0; i < calcuLevel; i++) {
                    calcuCoeffi = 2;
                    calcuChek = 0;
                    for (int j = 0; j <= i ; j++ ) {
                        calcuCoeffi += Math.pow(2, j); }
                    calcuChek += Math.pow(2, temp1);
                    coefNum [i] = calcuCoeffi;
                    checkNum [i] = calcuChek;
                    temp1--;}
                /* assign value of calcuSymbol array*/
                for(int i = 0; i < calcuLevel; i++) {
                        calcuSymbol [i] = 0; }
                count = 1;

                /**************************************************/
                /* Reconstruction of the left half of value in tree*/
                while (leftHalf != 0){
                        tempValue = Double.parseDouble(coefficientValues[0]) +
Double.parseDouble(coefficientValues[1]);
                        for (int k = 0; k < calcuLevel; k++) {
                                /* add current level number to tempValue*/
                                if (calcuSymbol [k] == 0){
                                        tempValue += Double.parseDouble(coefficientValues[coefNum[k] - 1]);
                                }
                                else {tempValue -= Double.parseDouble(coefficientValues[coefNum[k] - 1]); }

                                /* check the current level number for the next use*/
                                /*check next turn of current level is + or -*/
                                if (count % (checkNum[k]/2) == 0){
                                        if (calcuSymbol [k] == 0)
                                        {    calcuSymbol [k] = 1; }
                                        else {calcuSymbol [k] = 0; }}

                                /*check the value of the next turn of current level need to be changed or not*/
                                if (count % checkNum[k] == 0){
                                        coefNum [k] = coefNum [k] + 1; } }
                        tempValue = Math.round(tempValue * 1000) / 1000d;
                        newFormat = df.format(tempValue);
                        reconstrucValue.add(newFormat);
                        tempValue = 0;
                        count++;
                        leftHalf--;}

                count = 1;

                /**************************************************/
                /* Reconstruction of the right half of value in tree*/
                for(int i = 0; i < calcuLevel; i++) {
                        calcuCoeffi = 0;
                        calcuChek = 0;
                        calcuCoeffi += Math.pow(2, i + 2);
                        calcuChek += Math.pow(2, temp1);
                        coefNum [i] = calcuCoeffi; }

                /* assign value of calcuSymbol array*/
                for(int i = 0; i < calcuLevel; i++) {
                        calcuSymbol [i] = 1; }
                while (rightHalf != 0){
                        tempValue = Double.parseDouble((String)coefficientValues[0]) -
Double.parseDouble((String)coefficientValues[1]);
                        for (int k = 0; k < calcuLevel; k++) {
                                /* add current level number to tempValue*/
                                if (calcuSymbol [k] == 1){
                                       tempValue -= Double.parseDouble((String)coefficientValues[coefNum[k]
- 1]); }
                                else {     tempValue +=
Double.parseDouble((String)coefficientValues[coefNum[k] - 1]); }

                                /* check the current level number for the next use*/
                                /*check next turn of current level is + or -*/
                                if (count % (checkNum[k]/2) == 0){
                                        if (calcuSymbol [k] == 0)
                                        {    calcuSymbol [k] = 1; }
                                        else { calcuSymbol [k] = 0; }}

                                /*check the value of the next turn of current level need to be changed or not*/
                                if (count % checkNum[k] == 0){
                                        coefNum [k] = coefNum [k] - 1; }}
                        tempValue = Math.round(tempValue * 1000) / 1000d;
                        newFormat = df.format(tempValue);
                        reconstrucValue.add(iNumNodes / 2, newFormat);
                        tempValue = 0;
                        count++;
                        rightHalf--;}
                String [] temp = (String[])reconstrucValue.toArray(new String[reconstrucValue.size()]);
                return temp; }
```

```
    /* Reconstrcture angle ranges function */
    public static String reconRange(String binary)
    {   int length = binary.length(); ////this one is for the length of the string
        double ranges = Math.pow(2, length);
        double rangeValue = 360 / ranges;
        String tempRange;
        DecimalFormat df = new DecimalFormat("##0.00");
        String newFormat;
        String newFormat2;
        int decimal = Integer.parseInt(binary,2); // change binary number to decimal
        double rangeStart = decimal * rangeValue;
        double rangeEnd = rangeStart + rangeValue;
        newFormat = df.format(rangeStart);
        newFormat2 = df.format(rangeEnd);
        tempRange = newFormat + " - " + newFormat2;

        return tempRange;
    }
}
```