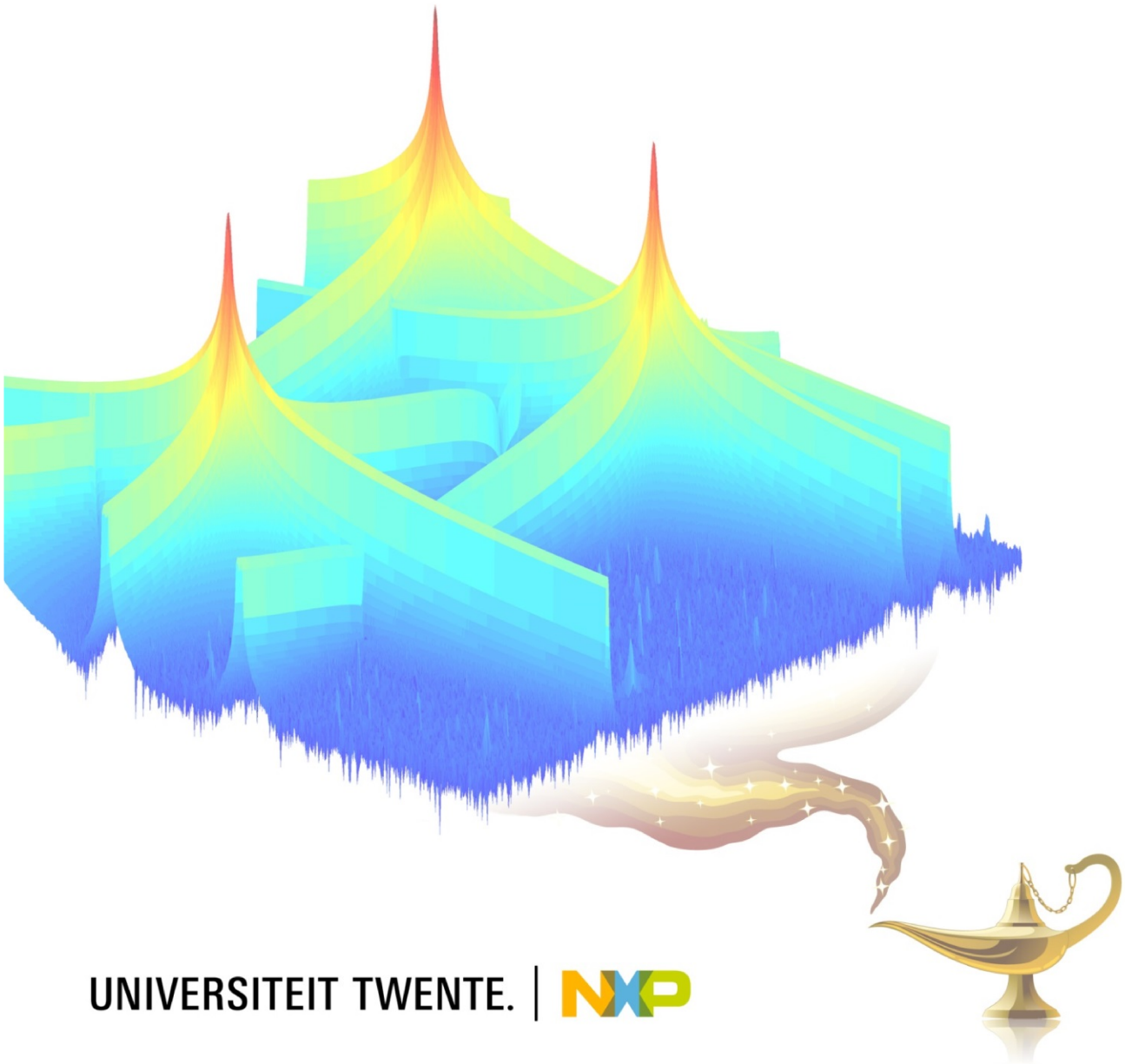
Embedded Data Compression in Automotive FMCW Radar System

Liang Li



UNIVERSITEIT TWENTE. | NXP

EMBEDDED DATA COMPRESSION IN AUTOMOTIVE FMCW RADAR SYSTEM

Liang Li

Master of Science Thesis

Computer Architecture for Embedded System Group
FACULTY OF ELECTRICAL ENGINEERING, MATHEMATICS
AND COMPUTER SCIENCE
UNIVERSITY TWENTE

Supervising Committee:

Prof.dr.ir Marco.J.G.Bekooij
Dr.ir Zoran Zivkovic
Prof.dr.ir André Kokkeler
Prof.dr.ir Gerard.J.M.Smit
Prof.dr.ir Arjan Meijerink

To my mother

Abstract

Improving safety and convenience in driving necessitates technologies for gathering information of the surroundings in real time. Range and velocity of surrounding objects are very important for collision avoidance, aided parking and so on. One of the potential candidates for obtaining this information is the Frequency Modulated Continuous Wave radar which can measure ranges and velocities of multiple targets in one measurement with reasonable accuracy and speed. However, in its baseband signal processing, large amount of memory is needed due to the special 2D FFT processing requirements. The memory requirement is in conflict with the market targeting of this system which should be a low cost, single chip solution for consumer automotive. A data compression scheme has to be used in the baseband signal processing to reduce the total system memory in order to shrink the area of the chip.

Based on the characteristics of the signal, two different compression schemes are developed in this thesis. The first one is called the Range Dependent Variable Length Encoding (RDVLE) which directly cut unused bits in the data according to the range – received power model. This bits-cutting may lead to data loss so this scheme is categorized as lossy compression. To further increase the compression ratio, the Uniform Dynamic Range Encoding (UDRE) is introduced and a new algorithm is developed to reduce the distortions generated in this RDVLE + UDRE compression. This scheme is evaluated in Maltab simulations and the compression ratio is around 2.2 while the other performances are shown in Chapter 4.8.

The second algorithm is originally lossless. Due to the combination of the Run Length Encoding (RLE) and the Huffman Encoding and the utilization of redundancy between sweeps (Doppler processing), its name is Doppler Redundancy Hybrid Encoding (DRHE). Before encoding, a prediction block will predict the input signal based on the previous input and the differences between the prediction and the input signal will be encoded. By observing the characteristics of the differences signal, the RLE is chosen to encode the large number of segments of consecutive zeroes and Huffman encoding is used to encode other values. In consideration of hardware implementation, a column based version of this scheme with memory management functionality is developed to fit all the compressed data into a regular, fixed size memory where the memory management may introduce rare data loss. The original algorithm may achieve a compression ratio as high as 9 while the compression ratio in the column based version with memory management can be set manually (typical CR is 5) in tradeoff among compression ratio, memory efficiency and the chance of data loss.

Both of these two schemes are designed with considerations of hardware implementations and overhead for possible future implementations.

Keywords:

FMCW radar, data compression.

Acknowledgements

Impossible would this thesis be without the help from many people. Hereby, I would like to express my sincere gratitude and appreciation to them. First of all I would like to thank my supervisor at NXP Semiconductors, Mr. *Zoran Zivkovic*. I benefited a lot from our daily discussions about a broad range of topics related to this thesis and I gained much inspiration from it. Not only knowledge, I also gained insights about the company and the industry itself which is crucial to my future career.

I am also very grateful to my supervisors Mr. *Marco Bekooij* and Mr. *André Kokkeler* from University Twente. In the regular meetings, they gave me many valuable feedbacks and directions on the projects. I'd like to thank them for their patience and critical attitude to review my thesis and paperwork for graduation preparation.

Mr. *Feike Jansen*, Mr. *Frank Leong* and Mr. *Stefan Drude* also gave me valuable suggestions and help during my internship and thesis project in NXP Semiconductors. I'd like to express my gratitude also to them.

During my one year stay in the company, I had a great time with the interns in our office who are Kristof Blutman, Brian Susilo, Liang Huo, Ahmet Ozgur, Dragos Amzucu, Lalit Kumar, Haris Papadopoulos, Sander Geluke, Peishuo Li and Luis Pacheco. Besides meaningful discussions during coffee break, we had a great time dining, seeing movies, attending events and doing other activities together. I wish them lots of happiness and successful careers in the future!

I would also like to thank all my friends and colleagues whom I met during this two years study in University Twente, especially Yu Tu, Jun Wang, Chao Ni, Liwei Ma, Zhenlei Li, Keli Chen, Yiyuan Zhao, Jianlei Tian, Lu Wang, Jing Xu, Yiran Wang, Xinwei Bai and Yiyuan Lin, for their support and family-like care. Many thanks also to my girlfriend Zao Ye and she made me feel she is the one who is always there for me.

My gratitude also goes to all my family members. Though apart across continents, I feel their love and care all the time and I really appreciate that they could respect my own decisions and way of life on the road to individuality. Nothing could I achieve without the help and support from you. 我爱你们。

Liang Li

Eindhoven, 13/Jul/2014

Table of Contents

ABSTRACT I

ACKNOWLEDGEMENTS III

TABLE OF CONTENTS V

1 INTRODUCTION TO DATA COMPRESSION 1

1.1 REVIEW OF INFORMATION THEORY 1
1.2 LOSSLESS COMPRESSION ALGORITHMS 2
1.2.1 STATISTICAL METHODS 3
1.2.2 DICTIONARY-BASED METHODS 4
1.3 LOSSY COMPRESSION ALGORITHMS 5
1.4 SW AND HW IMPLEMENTATIONS AND APPLICATIONS 5
1.5 DATA COMPRESSION IN RADAR 6

2 FMCW SIGNAL PROCESSING 9

2.1 PRINCIPLE OF FMCW RADAR 9
2.1.1 SIMPLE RADAR RANGE EQUATION 9
2.1.2 FMCW SIGNAL AND SYSTEM ARCHITECTURE 10
2.1.3 DERAMPING AND RANGE MEASUREMENT 11
2.1.4 DOPPLER FREQUENCY SHIFT AND VELOCITY MEASUREMENT 11
2.2 2D FFT PROCESSING 12
2.2.1 DERAMPING 12
2.2.2 FOURIER TRANSFORM PROCESSING 14
2.3 MEASUREMENT RESOLUTIONS 17
2.4 SYSTEM SPECIFICATIONS 17

3 DATA COMPRESSION IN FMCW RADAR SIGNAL PROCESSING FLOW 19

4 RANGE DEPENDENT VARIABLE LENGTH ENCODING 21

4.1 RECEIVED SIGNAL POWER PROFILE 21
4.2 CAR RCS DISTRIBUTION 23
4.3 FFT PROCESSING GAIN AND BIT LENGTH GROWTH 25
4.4 RANGE DEPENDENT VARIABLE LENGTH ENCODING 26
4.5 UNIFORM DYNAMIC RANGE ENCODING 27
4.6 CLIPPING ERROR AND CLIPPING REDUCTION METHOD 29
4.7 HARDWARE CONSIDERATIONS 32
4.8 SIMULATION AND RESULTS 33
4.8.1 MODELING OF THE SIGNAL SOURCE AND PROCESSING STEPS 33
4.8.2 SIMULATION SETUP 34
4.8.3 SIMULATION RESULTS 34

5 DOPPLER REDUNDANCY HYBRID ENCODING 39

5.1 DOPPLER REDUNDANCY AND PREDICTION MODEL 39
5.2 CHOICES OF ENCODING 42
5.3 RUN LENGTH ENCODING 42
5.4 HUFFMAN ENCODING AND CODE TABLE OVERHEAD 42
5.5 COLUMN BASED DRHE ENCODING 44
5.6 COLUMN BASED DRHE DECODING 47
5.7 MEMORY MANAGEMENT IN ENCODING PROCESS 48
5.8 HARDWARE CONSIDERATIONS 53
5.9 SIMULATION AND RESULTS 54

6 CONCLUSIONS AND RECOMMENDATIONS 57

LIST OF TABLES 59

LIST OF FIGURES 59

APPENDIX A: THE HUFFMAN CODES OF R4S4 SYMBOLS 62

BIBLIOGRAPHY 65

1 Introduction to Data Compression

Data compression research dates back to the 1950s [1] and has become standard practice in various current digital systems due to the ever-increasing amount of data to be stored and transferred. One good example is online video/image streaming [2], which would be hard and inefficient with current link speeds without data compression technology. Compression is generally achieved by representing information in a more compact form by utilizing statistical structure or removing unimportant components in the information flow.

This chapter will review some of the most successful and widely used data compression algorithms and applications to provide an overview and background of this study before diving into application specific, custom designed compression schemes. In this thesis, data compression is designed for a special application: the automotive radar system which made it necessary to understand the system structure and signal features to better fit certain designs to the system. Therefore, following the introduction of data compression, the radar systems in automotive are briefly introduced as well in this chapter.

There are two major categories of compression algorithms: lossless and lossy, where the criterion is whether the exact original input can be restored from the compressed output. Before introducing both of them, a crucial performance matric of compression algorithms has to be defined: the compression ratio.

$$CR = \frac{\text{number of input bits}}{\text{number of output bits}} \tag{1.}$$

Reciprocal of Equation (1.) might be used as compression ratio in some literatures [3], they are converted to this expression form when referenced. Another way to express the effectiveness of compression is to compare average bits per sample before and after compression.

Besides compression ratio, there are other performance metrics. Compression is often related to extra computation. Large amount of extra computation will increase its complexity no matter implemented in software of hardware. Complexity will also determine speed of the algorithm which is crucial to real time applications.

Overhead is another important factor, especially in hardware implementation. To achieve compression, various additional register, memory, function blocks etc. are needed, all of which will occupy extra chip area increasing total cost and total power consumption.

In lossy compression, the signal distortion caused by compression is also very important. The amount of distortion that is allowed is determined by application. For example, the information loss in Fax compression [4] and that in high definition image compression [5] can be very different.

Generally, the design of compression scheme for specific application is a process of **tradeoff** between different performance metrics and this will be seen so many times in this thesis.

1.1 Review of Information Theory

Information theory developed by *Claude E. Shannon* is a branch of applied mathematics that aims at quantifying information. The self-information of an event A is defined [6], associated with its probability, as:

$$i(A) = \log_{base} \frac{1}{P(A)} = -\log_{base} P(A) \tag{2.}$$

When the base of the log function is chosen to be 2, the unit of $i(A)$ is bit. If event A and B are independent, then we have:

$$i(AB) = -\log_2 P(AB) = -\log_2 P(A)P(B) = -\log_2 P(A) - \log_2 P(B) = i(A) + i(B) \quad (3.)$$

If we have a set of independent events A_i which form the sample space S of random experiment:

$$S = \cup A_i \quad (4.)$$

Then the average self-information associated with this random experiment is:

$$H = \sum_i P(A_i) i(A_i) = -\sum_i P(A_i) \log_2 P(A_i) \quad (5.)$$

H is called the entropy of the experiment. If the random experiment is a source putting out symbols A_i, *Shannon* showed that the best lossless compression scheme cannot do more than encoding the output of a source with an average number of bits equal to the entropy of the source [7]. Information theory identifies how many bits should be actually used to represent one symbol and redundancy can be eliminated by using a symbol not longer than necessary.

For example, the relative frequency of letter e in English language is about 0.116 [8], so the self-information contained in the letter e in English language is:

$$i(L_e) = \log_2 \frac{1}{0.116} = 3.11 \text{ bits} \quad (6.)$$

This means the letter e can be represented by no more than 4 bits when encoding a piece of English text. An ASCII text file uses 8 bits to encode each letter and the redundancy of 4 bits of information enables the possibility of compression.

Generally, data compression takes a stream of symbols and transforms them into certain codes. The decision to output a certain code for a certain symbol or set of symbols is based on a model. The model is simply a collection of data and rules that are used to process input symbols and determine which code(s) to output [7]. Then a coder produces appropriate code based on the model. So it is often said "Compression is a combination of model and coding". For example, in the Huffman coding compression scheme [9], symbol probabilities are obtained using some statistical model from the input data flow, then these probabilities values are used by the Huffman encoder to generate codes to represent each symbol. The basic model of compression algorithms is shown in Figure 1-1.

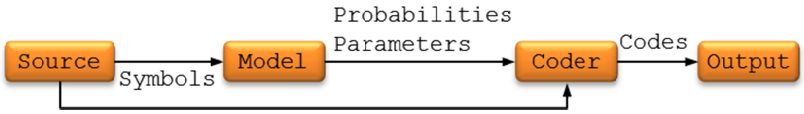


Figure 1-1 Modeling + Coding model for compression algorithm

People often use the term "coding" or "encoding" to represent the entire data compression process which might be very misleading such as "Huffman coding" and "Run length encoding". These naming conventions vary. In this thesis, they are regarded as both simply coding methods used in conjunction with a model to compress data and the whole compression scheme depending on context. Different models and coders are introduced in the following chapters.

1.2 Lossless Compression Algorithms

According to the above criterion, lossless data compression reduces the size of the input in a way such that a corresponding decompression algorithm can restore the original input exactly with no loss of data. Lossless compression is widely used in program code and text compression where strict restoration is required to guarantee execution correctness and readability [10]. For example, one letter change in a sentence may totally change the meaning and one single digit change in bank account numbers is not acceptable. Images, sound and other type of information can also be compressed using general purpose lossless compression algorithms but the compression ratio is usually not as good as that using lossy algorithms.

The "model + coder" model is mainly used to describe the two main components in lossless compression algorithms. Model is the brain of the compression algorithm which dictates the coder to do data encoding. Generally, two different types of models exist in lossless data compression: statistical and dictionary-based. Chapter 1.2.1 and Chapter 1.2.2 introduce these two types of models with their frequently-used coders.

1.2.1 Statistical Methods

Statistical lossless compression methods show a clearer separation between model and coder than dictionary-based methods [11]. As its name suggests, this type of compression uses statistical information from the input data to do compression. As introduced in Chapter 1.1, the amount of self-information is determined by a symbol's appearance probability in the data stream. The higher the probability, the less self-information it contains, the fewer bits are needed to encode it. Statistic models analyze the input data and produce probability information to feed the coder to generate code for each input symbol.

Statistics can be gathered based only on current input symbols which is called 0-order model. For example, regarding each byte in the input file as one symbol which corresponds to the ASCII code and count the time of appearance of different symbols, we can have a counts table that shows each symbol's frequency. If each frequency count is stored in an unsigned char, this 0-order statistical model occupies 256 bytes. As a natural improvement, a 1-order model might be used where the statistics are gathered based also on the previous one symbol. For example, the letter 'u' alone might have a probability of 0.01 in English, but if its previous symbol is 'q' then the probability of 'u' may rise to 0.95 [7].

Generally, higher order models will result in better compressions but also occupy more storage space. If a 0-order model occupies 256 bytes, a 1-order model will take up 256*256=65536 bytes. In a static model, the probability information has to be passed to the decompression algorithm for proper decoding which means the model itself has to be appended to the compressed file which wipes out some benefits of compression. To solve this problem, adaptive modeling is used where the model is built gradually as the data is fed in. As long as the decompression algorithm builds the model in the same way that the compression does, the data will be decoded correctly. Figure 1-2 shows a diagram of an adaptive model in comparison with the static model in Figure 1-1.

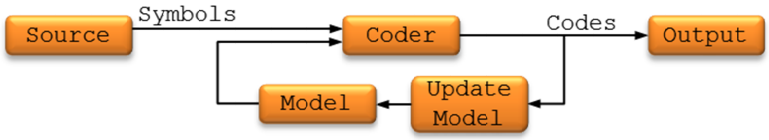


Figure 1-2 Adaptive modeling (compression)

Statistical models are often concatenated with Huffman encoder [9] which is a minimum redundancy coding scheme. It uses the statistical information from the model to assign codes to each input data sample. The Huffman coding is a coding method that generates variable integer-length codes. Its codes have unique prefix (prefix code) so they can be decoded correctly without any symbol delimitation. An example to encode symbols using the Huffman coding combined with a 0-order static statistical model is presented in the following.

Table 1-1 Example symbol frequency table

Symbol	A	B	C	D	E
Frequency	15	8	5	5	4

Table 1-1 shows a symbol set with 5 symbols in total from A to E and their frequencies which can be regarded as a 0-order statistical model. A Huffman tree is built with this information from leaf to root. The initial symbols in the table are regarded as nodes to be selected. In each iteration, two symbols with the lowest frequency counts are added as two child nodes and removed from the table. Their father node is created with the sum of their frequency counts and added back to the node list. This process continues until there is no node in the list.

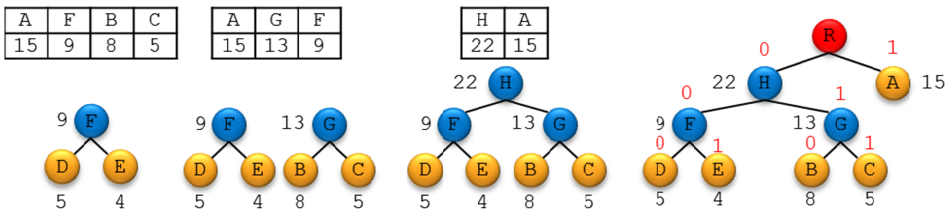


Figure 1-3 Huffman tree generation

Figure 1-3 shows the process of building a Huffman tree based on the example frequency table. Once we have the tree, code for each symbol is available just by assigning 0 to every left child and 1 to every right child.

Table 1-2 Huffman code of example model

Symbol	Huffman Code
A	1
B	010
C	011
D	000
E	001

The generated codes are shown in Table 1-2. The most frequent symbol A is assigned to the shortest symbol. No code is prefix to the others therefore each code can be decoded uniquely by just going through bit by bit of the compressed data stream. All the other codes except A have the same length which actually reveals a disadvantage of the Huffman coding: the code must have integer length. The Arithmetic coding [12] performs better than Huffman coding by encoding a stream of input symbols with a single floating point number which is actually equivalent to having fractional code length.

Both the Huffman coding and the Arithmetic coding can be modified to adaptive coding versions. As shown in Figure 1-2, the statistical model is adjusted on the fly. Adaptive statistical methods like prediction by partial matching (PPM) [13], [14] and Dynamic Markov Compression (SMC) [15] provide very good compression ratio but in general, effective higher order models can be computational and memory costly to be implemented in hardware.

1.2.2 Dictionary-based Methods

For dictionary-based method, the line between model and coder becomes vague. Coders in this type of compression usually encode several symbols as pointers pointing to dictionary items. Thus, models are of great importance in dictionary-based methods. The most famous family of dictionary based methods is the Lempel-Ziv (LZ) algorithms. The LZ77 [16] and LZ78 [17] are the basis for many variations like LZW [18], LZO [19]and so on.

Here, we present the LZ77 as a typical example to show the basic idea of dictionary-based compression. The LZ77 algorithm is shown below.

LZ77 algorithm [20]

1. *Set coding position to the beginning of the input stream.*
2. *Find the longest match of length L in the window for the rest of input data from coding position to the end of the input stream. The window is a buffer of size W indicates the number of bytes from the coding position backward.*
3. *If a match is found, output the pointer P. Move the coding position and the window L bytes forward.*
4. *If a match is not found, output a null pointer and the first byte from the coding position. Move the coding position and window one byte forward.*
5. *Go to step 2 until all input bytes are processed.*

Take the input stream “AABABCABC” as an example. The encoding steps are shown in Table 1-3. Once an input data is found in the “dictionary” (window), instead of the original data, a

pointer to the dictionary item is output. When the window is long enough, longer input data sequence can be replaced with much shorter pointer to achieve compression.

Table 1-3 LZ77 encoding steps with example input

Step	Position	Match	Output
1	1	-	(0,0)A
2	2	A	(1,1)
3	3	-	(0,0)B
4	4	AB	(2,2)
5	6	-	(0,0)C
6	7	ABC	(3,3)

The output pointer (x,y) means going back x bytes and take y bytes from that position. Decoding starts from the beginning of the compressed data and the decoded data serves as dictionary to the rest of the compressed data. Dictionary-based compression algorithms are widely used in various general purpose compression schemes both in software and hardware that are discussed in Chapter 1.4.

1.3 Lossy Compression Algorithms

Lossy compression will result in loss of information in the input and the restored information will not be exactly the same as the original input, but in return of accepting distortions in the reconstruction, higher compression ratios can often be achieved compared to lossless compression. Lossy compression is mainly used for analog data that is stored digitally and it's popular in compressing signals for human perception for the fact that humans only have limited perceptual abilities [21], [22]. We cannot hear high frequency components above 20kHz in sound, neither can we see ultra-fine details in images which can be removed from the original information without impairing proper understanding of the original information. The compression ratio also depends on what restoration quality the user accepts: speech in telephone quality and in CD quality can both be understood without ambiguity, but are significantly different in compressed size. Lossy compression makes it possible to effectively compress sound, images, videos and other signals for storage and communication.

Quantization and decimation can be regarded as simple lossy compressions which are used in converting analog signal to digital format. These are usually the first steps in compressing image and sound. Many lossy compression schemes such as JPEG [23] [24] and MP3 [25] designed for human perceptible signals use the Discrete Cosine Transform (DCT) [26] to transform signal from the time domain into the frequency domain. Reduction of information is achieved by quantization of frequency components and eliminating imperceptible frequency components according to certain human perception models.

Many kinds of signal have redundancy in itself. For example, consecutive frames in one video clip may have very similar content (temporal redundancy) and nearby channels in the same radar system might have similar responses (spatial redundancy). Taking advantage of redundancy, future data may be predicted based on the past samples. If the prediction is good, the differences between real input and prediction will have much less information. Predictions are used in various compression schemes such as MPEG2 [27], [28].

1.4 SW and HW Implementations and Applications

Various compression algorithms are implemented both in software (SW) and hardware (HW) depending on their applications and requirements. Take the MP3 encoding as an example, when it's needed in real time, embedded applications such as portable recorder, it may be implemented in dedicated chips. In applications like offline audio format conversion, software encoding library is enough and is less costly.

Some compressions are designed to be implemented only in dedicated hardware to provide very high speed such as IBM ALDC [29] and X-MatchPRO [11]. Both of them are dictionary-based lossless compression schemes that are designed for high speed digital storage and communica-

tion. Often, dictionary based algorithm often needs Content Addressable Memory (CAM) [30] in hardware to implement the "dictionary" as a lookup table which is quite power consuming and area inefficient. Despite the common JPEG, MP3 compression, hardware lossy compression schemes are also found in some specific applications like [31] which is designed to compress electrocardiogram signal based on DCT-IV.

Software implementations are more common than their hardware counterparts. Despite the general purpose software packages and libraries like zlib [32], PKZIP, some application specific schemes were also designed. [33] presents a SVD based approach for radar signal compression. In [34], a novel software-based RAM compression is presented using LZO algorithm in order to save system memory by compressing rarely used RAM contents. zram [35], zcache [36] and zswap [37] are software implementations with similar targets.

Actually, online RAM compression is gradually becoming mainstream in many modern operation systems. zram was merged into the Linux kernel mainline in version 3.14 in March 2014. Apple's compressed memory [38] in its OSX Mavericks operation system uses WKdm [39], a dictionary-based algorithm. IBM also has similar design [40].

1.5 Data Compression in Radar

To improve safety and comfort for driving, modern automotive is evolving continuously to gain the ability to know more and more about its surroundings. [41] Range and velocity of moving targets (e.g. other vehicles, pedestrians) around the automotive is important information and Radar is a very good candidate for obtaining this information. Radar is short for RAdio Detection And Ranging which uses radio waves to determine range, direction and speed of objects. Information provided by automotive Radar system can be used in aided parking systems [42], collision avoidance [43], night driving and even in automated steering. Figure 1-4 shows an aided parking concept.

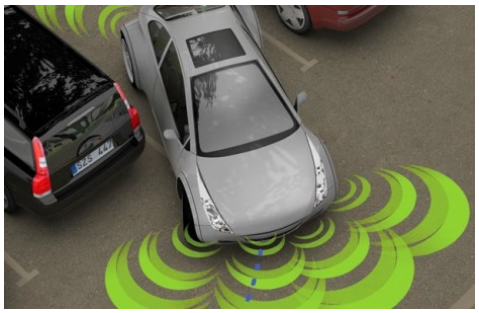


Figure 1-4 Aided parking [44]

Figure 1-4 shows a car equipped with an aided parking system. The system uses radar to measure distances between different points on the car body and the surroundings and control the car with the right steering angle and route. If the driver parks the car by himself, the system will produce warnings when it comes too close to its surroundings. In the application of collision avoidance, distance and relative speed information from radar measurement is used to determine if it's a critical situation to do automatic break and control before collision.

Other competing technology also exists, such as Light Detection and Ranging (LIDAR) and camera-based driving assistance. However, these methods suffer from bad weather conditions such as thick fog while radar does not. Other advantages of using radar include direct distance and speed measurement and invisible integration behind electromagnetically transparent materials such as car's bumper. For example, as shown in Figure 1-5, Google's self-driving car [45] employs Lidar, camera and Radar at the same time to get a comprehensive understanding of its surroundings to ensure correct behavior and safety.

Signal processing in Radar system needs a lot of computational resources as well as memory which will increase system cost. Single chip SoC solution is favored in terms of cost and sys-

tem design complexity. The amount of memory needed in a Radar system is often too much for SoC so data compression is necessary.

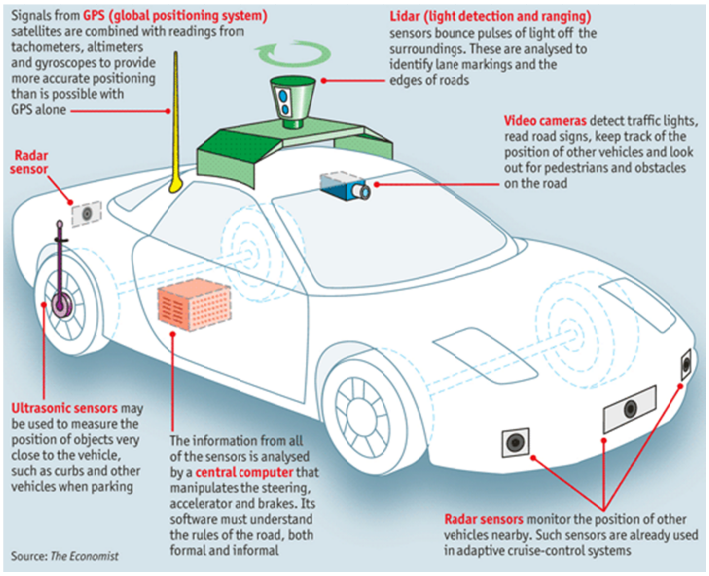


Figure 1-5 How a self-driving car works [46]

There are many works about radar signal compression in transmission and storage such as [47], [48] [49] and [50] but the study on compression as part of the radar signal processing in order to save system memory is rare which is the very topic of this work. Figure 1-6 shows the difference between the two kinds of applications.

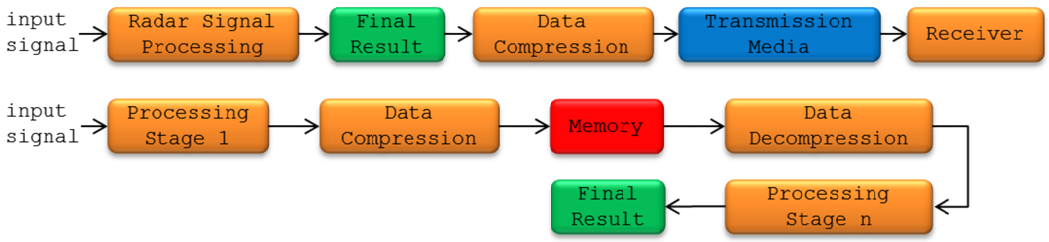


Figure 1-6 Difference between compression for transmission/storage and compression in between processing

Before getting into compression algorithm design, in the following sections, we will introduce the basic principle of FMCW radar as well as the signal processing methods employed to obtain relevant information from received radio signals.

2 FMCW Signal Processing

2.1 Principle of FMCW Radar

2.1.1 Simple Radar Range Equation

The famous radar range equation relates received echo power P_r to transmitted power P_t in a deterministic way in terms of many common system design parameters [51]. The relationship is important because radar relies on the radio wave echo to do measurement and detection. A simple step-by-step deduction may help to understand it. The following shows the process to obtain the radar range equation for a simple point target:

First, assume the transmitting antenna has an isotropic radiation pattern and the radio wave travels in a lossy medium with attenuation factor of $L_m(r)$. With the loss caused by the medium, at a range r, the total transmitted signal power spreads on the sphere with a radius of r. The power density is:

$$PD_1(r) = \frac{P_t}{4\pi r^2 L_m(r)} \left(W/m^2\right) \tag{7.}$$

However, most radar applications use directive antennas which have antenna gain of G_t to increase power in the direction of interest. The power density at r then becomes:

$$PD_2(r) = \frac{P_t G_t}{4\pi r^2 L_m(r)} \left(W/m^2\right) \tag{8.}$$

When the transmitted wave hits a target, part of the energy is scattered back, part of it is absorbed by the target itself. To characterize this process, it is imagined that the target collects all the incident energy on a fictional area σ and re-radiates it isotropically. Therefore, for monostatic radar, the power density at the collocated receiver is:

$$PD_3(r) = \frac{P_t G_t \sigma}{4\pi r^2 L_m(r) 4\pi r^2 L_m(r)} = \frac{P_t G_t \sigma}{\left[4\pi r^2 L_m(r)\right]^2} \left(W/m^2\right) \tag{9.}$$

This fictional area σ is called the Radar Cross-Section (RCS). It is not the same as the target physical cross-section area and usually depends very largely on the incident elevation and azimuth. The received power at the receiver again depends on a fictional area which characterizes the receiving antenna – effective area A_e. Then the received power is expressed as:

$$P_r(r) = \frac{P_t G_t \sigma A_e}{\left[4\pi r^2 L_m(r)\right]^2} \left(W/m^2\right) \tag{10.}$$

The effective area A_e is related to receiving antenna gain [52] as:

$$A_e = \frac{G_r \lambda^2}{4\pi} \left(m^2\right) \tag{11.}$$

where λ is the transmitted radio wave length. Substituting A_e into the received power $P_r(r)$, the received the power is written as:

$$P_r(r) = \frac{P_t G_t G_r \sigma \lambda^2}{(4\pi)^3 r^4 L_m{}^2(r)} (W) \tag{12.}$$

which is the radar range equation.

2.1.2 FMCW Signal and System Architecture

An FMCW signal is a kind of continuous waveform with frequency modulation [53]. Typically, linear frequency modulation is used where the frequency of a sine wave is linearly increased or decreased. Figure 2-1 shows the saw tooth FMCW signal.

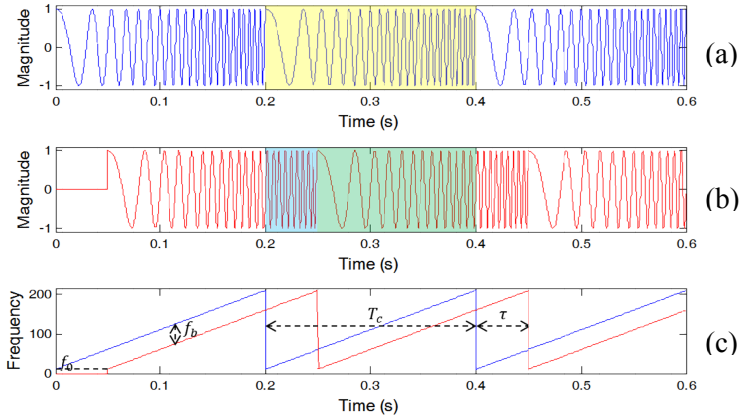


Figure 2-1 (a) Transmitted saw tooth FMCW signal (b) Returned signal with a delay (c) Time-Frequency relationship of the two signals.

The saw tooth linear frequency modulated signal with an increasing frequency is also called the (up)chirp signal. One period of the signal shown in the yellow shaded region in Figure 2-1 (a) is called one sweep of chirp of the FMCW signal which can be expressed mathematically as:

$$s_{chirp}(t) = A\cos\left[2\pi\left(f_0 + \frac{1}{2}\alpha t\right)t\right] \quad t \in (0, T_c] \tag{13.}$$

Then the instantaneous frequency can be obtained by differentiating the phase of s_{chirp} with respect to time t as:

$$f(t) = \frac{d}{dt}\left[\left(f_0 + \frac{1}{2}\alpha t\right)t\right] = f_0 + \alpha t \quad t \in (0, T_c] \tag{14.}$$

which means the instantaneous frequency is increasing linearly with time as shown in Figure 2-1 (c) where α is called chirp rate, f_0 is the initial frequency. The range between f_0 and f_{max} is the bandwidth of the FMCW signal which is

$$BW = f(T_c) - f(0) = \alpha T_c \tag{15.}$$

The other common form of linear FMCW is triangle FMCW where a frequency decreasing phase follows the increasing phase in the time-frequency relationship which is not shown here. Figure 2-2 shows a brief block diagram for FMCW radar system.

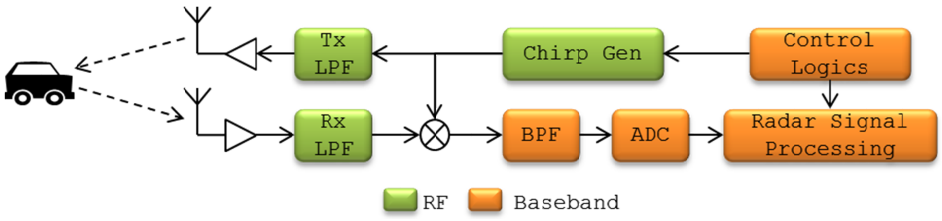


Figure 2-2 General FMCW radar system diagram

Under the control of the Control Logics block, continuous saw tooth chirp signal is generated by the Chirp Gen block and the signal is transmitted by a transmitting antenna after filtering and amplification. The incident FMCW signal is reflected by the target at some distance and the echo collected by the receiving antenna is mixed with the same FMCW signal from the transmitter. The product signal is then band pass filtered and sampled by an ADC. After sampling, it goes into the digital domain where certain algorithms are used to perform signal processing and target detection.

2.1.3 Deramping and Range Measurement

The properties of the mixed signal have to be known to employ proper signal processing algorithms in the following stages. The following conceptual deduction will shed some light on the mixed signal property. Instead of using a periodic saw tooth chirp signal which is more complex in mathematical form, we can simply multiply $s_{chirp}(t)$ with its delayed and scaled version of $s_{chirp}(t-\tau)$ as received signal as shown in Figure 2-1 (b). τ is the round trip time of flight:

$$\tau = 2 \times \frac{D}{c} \tag{16.}$$

where D is the distance between antennas and target, c is the light speed. The result, though not mathematically accurate, can reveal some information.

$$\begin{aligned} s_{mix}(t) &= s_{chirp}(t) \cdot \frac{B}{A} s_{chirp}(t-\tau) \\ &= AB\cos\left[2\pi\left(f_0 + \frac{1}{2}\alpha t\right)t\right]\cos\left[2\pi\left(f_0 + \frac{1}{2}\alpha(t-\tau)\right)(t-\tau)\right] \\ &= \frac{AB}{2}\cos\left[2\pi\left(2f_0 t + \frac{1}{2}\alpha t^2 - f_0 t + \frac{1}{2}\alpha(t-\tau)^2\right)\right] + \frac{AB}{2}\cos\left[2\pi\left(\alpha\tau t + f_0\tau - \frac{1}{2}\alpha\tau\right)\right] \\ &= S_{2f_0} + \frac{AB}{2}\cos\left[2\pi\left(\alpha\tau t + f_0\tau - \frac{1}{2}\alpha\tau\right)\right] \end{aligned} \tag{17.}$$

The first term S_{2f_0} is a high frequency term which will be filtered out by the BPF following the mixer and the input signal to the ADC will be the second term only which is:

$$s_{ADCin} = \frac{AB}{2}\cos\left[2\pi\left(\alpha\tau t + f_0\tau - \frac{1}{2}\alpha\tau\right)\right] \tag{18.}$$

The instantaneous frequency of this signal is:

$$f_b = \alpha\tau = \frac{2\alpha D}{c} \tag{19.}$$

Now, the range information is included in the frequency of the mixed signal which is often called the 'beat frequency'. This mixing process is called 'deramping' which converts a 'ramping-up' chirp signal to a constant frequency beat signal. Unlike pulse radar, the range information is contained in the frequency domain in the FMCW radar system, we could know the target distance by measuring the beat frequency which naturally leads to the signal processing method using the FFT in the following chapters.

2.1.4 Doppler Frequency Shift and Velocity Measurement

Due to the relative movement between the radar system and the targets, the round trip time of flight might change from time to time. This is especially true in automotive radar system which operates in an ever-changing environment. This relative movement will cause a frequency shift in the echo signal which is called the Doppler frequency shift given by the following equation:

$$f_D = \frac{2v_r}{\lambda} \tag{20.}$$

where v_r is the relative radial velocity between radar system and target. The Doppler frequency shift will introduce an error in the measurement of the beat frequency f_b thus influencing range measurement accuracy. However, this influence is small and by observing the signal phase shift cause by a Doppler frequency shift in consecutive FMCW sweeps, f_D can be determined and radial velocity is then also obtained.

With little influence in range measurement, speed information can be obtained which is of great use in automotive control system such as collision avoidance system

2.2 2D FFT Processing

Since both range and velocity information is contained in the frequency domain, processing in the frequency domain is a natural choice. The FFT is used to transform the received signal from time domain to frequency domain. Figure 2-3 shows the block diagram of a Radar signal processing block in Figure 2-2.

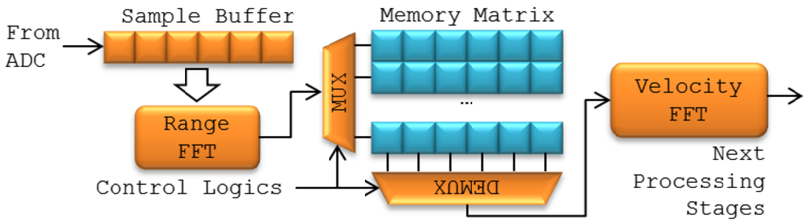


Figure 2-3 2D FFT processing diagram for the FMCW signal

The samples are first stored in the sample buffer and sent to the Range FFT block for performing the 1st FFT. The result of the 1st FFT reveals the beat frequency f_b of the deramped signal, it is therefore called range spectrum. Through multiplexer (MUX), range spectrums of several consecutive chirp sweeps are stored into consecutive rows of the storage matrix.

Once enough rows of range spectrums are collected, data are extracted column by column by the demultiplexer (DEMUX) and sent to the 2nd FFT block (or the Velocity FFT block). Due to the Doppler frequency shift f_D, phase shifts will happen among consecutive sweeps (rows) whose frequency will be revealed by this Velocity FFT. The final result contains both range and speed information, so it's called the Range-Velocity spectrum which will be used in the next processing stage to do, for example, target detection, target tracking and so on.

2.2.1 Deramping

Though simple, the previous deduction for deramping in Chapter 2.1 is not fully mathematically accurate because a real FMCW signal is a continuous periodic signal and the target is all the time moving away or towards the radar system. To illustrate the signal processing method in FMCW radar, a more accurate signal model is used.

The whole transmitted saw tooth FMCW signal with M complete chirps starting from zero time, therefore, can be expressed as:

$$s_T(t)=\sum_{m=0}^{M-1} A\cos\left[2\pi\left(f_0+\frac{1}{2}\alpha\left(t-mT_c\right)\right)\cdot\left(t-mT_c\right)\right]\Pi\left(\frac{t-\frac{T_c}{2}-mT_c}{T_c}\right) \qquad (21.)$$

where f_0 is the initial frequency, α is the chirp rate and T_c is the time duration for single chirp (it is also the signal period). The symbol Π represents the following rectangular window:

$$\Pi(t)=\begin{cases}0 & |t|>1/2\\ 1 & |t|\le 1/2\end{cases} \qquad (22.)$$

The signal is constructed by copying a windowed chirp signal every T_c second for M times. Assume there is one target at a distance r which is moving at a relative radial speed of v_r with respect to the radar system. Within the very short time of one chirp sweep, the movement of target is ignored, while between consecutive chirp sweeps, the distance in between changes by an amount of $T_c v_r$, thus we can write the returned echo as:

$$s_R(t)=\sum_{m=0}^{M-1}\cos\left[2\pi\left(f_0+\frac{1}{2}\alpha\left(t-mT_c-\tau(m)\right)\right)\left(t-mT_c-\tau(m)\right)\right]\Pi\left(\frac{t-\frac{T_c}{2}-mT_c-\tau(m)}{T_c}\right) \qquad (23.)$$

where the round trip time of flight is:

$$\tau(m) = 2(r + mv_r T_c)/c \tag{24.}$$

To simplify the expression, let $C(t)$ be:

$$C(t) = \cos\left[2\pi\left(f_0 + \frac{1}{2}\alpha t\right)t\right] \tag{25.}$$

Therefore, the mixed signal of deramping can be written as:

$$\begin{aligned} s_{mix}(t) = s_R(t)\cdot s_T(t) = &\left[\sum_{m=0}^{M-1} AC(t - mT_c)\Pi\left(\frac{t - \frac{T_c}{2} - mT_c}{T_c}\right)\right]\times \\ B&\left[\sum_{m=0}^{M-1} C(t - mT_c + T_c - \tau)\Pi\left(\frac{t - \frac{\tau}{2} - mT_c}{\tau}\right) + \sum_{m=0}^{M-1} C(t - mT_c - \tau)\Pi\left(\frac{t - \frac{T_c - \tau}{2} - mT_c - \tau}{T_c - \tau}\right)\right] \end{aligned} \tag{26.}$$

Despite of its complex form, the above equation is simply the product of yellow shaded region and blue-green shaded regions in Figure 2-1. The term in the first square bracket corresponds to the yellow shaded part, the first term in the second square bracket corresponds to the blue shaded part and the last term corresponds to the green shaded part.

It's obvious that, for the scaled and shifted rectangular gate function, the following relationships hold:

$$\begin{cases} \Pi\left(\dfrac{t - \frac{T_c}{2} - mT_c}{T_c}\right) = \Pi\left(\dfrac{t - \frac{\tau}{2} - mT_c}{\tau}\right) + \Pi\left(\dfrac{t - \frac{T_c - \tau}{2} - mT_c - \tau}{T_c - \tau}\right) \\ \Pi(t)\cdot\Pi(t) = \Pi(t) \end{cases} \tag{27.}$$

So, the mixed signal is:

$$s_{mix}(t) = AB\sum_{m=0}^{M-1}\left[\begin{array}{l} C(t - mT_c)C(t - mT_c + T_c - \tau)\Pi\left(\dfrac{t - \frac{\tau}{2} - mT_c}{\tau}\right) \\ +C(t - mT_c)C(t - mT_c - \tau)\Pi\left(\dfrac{t - \frac{T_c - \tau}{2} - mT_c - \tau}{T_c - \tau}\right) \end{array}\right] \tag{28.}$$

Now, let's observe this mixed result term by term. The first product term in the summation is:

$$C(t - mT_c)C(t - mT_c + T_c - \tau) = M_{2f_0} + \frac{1}{2}\cos 2\pi\left[\begin{array}{l} -\alpha(T_c - \tau)t + \alpha m T_c(T_c - \tau) \\ -f_0(T_c - \tau) - \dfrac{1}{2}\alpha(T_c - \tau) \end{array}\right] \tag{29.}$$

where M_{2f_0} represents a $2f_0$ high frequency term while the frequency of the second term is $-\alpha(T_c - \tau) = -BW + \alpha\tau$ which can be seen in the time-frequency relationship for the blue-shaded part in Figure 2-1. Usually, in FMCW systems, T_c is designed to be much larger than τ therefore the beat frequency $f_b = \alpha\tau$ is merely very small compared of total BW. This is to say, the frequency of the second term in Equation (29.) is also much higher than the frequency range we are interested in. Therefore, the first term in Equation (28.) will be filtered out by the BPF that is preceding the ADC. The second term in Equation (28.) which is $C(t - mT_c)C(t - mT_c - \tau)$ is expanded as:

$$C(t - mT_c)C(t - mT_c - \tau) = M_{2f_0} + \frac{1}{2}\cos 2\pi\left(a\tau t - a\tau m T_c - \frac{1}{2}a\tau^2 + f_0\tau\right) \tag{30.}$$

The high frequency term M_{2f_0} will be filtered out by the BPF. In summary, excluding noise, the signal which is present at the input of ADC will be:

$$s_{BPF}(t)=A_R\sum_{m=0}^{M-1}\cos 2\pi\left(a\tau t-a\tau mT_c-\frac{1}{2}a\tau^2+f_0\tau\right)\Pi\left(\frac{t-\frac{T_c-\tau}{2}-mT_c-\tau}{T_c-\tau}\right) \quad (31.)$$

where $A_R=\frac{1}{2}AB$. Expressed in exponential form:

$$s_{BPF}(t)=\frac{A_R}{2}\sum_{m=0}^{M-1}\left[e^{j2\pi\left(a\tau t-a\tau mT_c-\frac{1}{2}a\tau^2+f_0\tau\right)}+e^{j2\pi\left(-a\tau t+a\tau mT_c+\frac{1}{2}a\tau^2-f_0\tau\right)}\right]\Pi\left(\frac{t-\frac{T_c+\tau}{2}-mT_c}{T_c-\tau}\right) \quad (32.)$$

2.2.2 Fourier Transform Processing

The sampled signal will go through two Discrete Fourier Transforms with the help of an FFT block. To illustrate the concepts and keep its form simple, we will use the continuous Fourier Transform in the theoretical deduction. Moreover, the following deduction will only deal with one of the exponentials which come from the cosine function because the other exponential will simply have a symmetric result after the FFT processing which doesn't give extra information. In real system, with a Nyquist rate ADC, the FFT results are just frequency domain sampled version of the mathematical deduction results.

The Range FFT is done on each signal segment or each row in the memory matrix in Figure 2-3 rather than on the whole $s_{BPF}(t)$. So it's better to rearrange $s_{BPT}(t)$ into a 2D 'matrix' form that is shown in Figure 2-4:

$$s_{BPF}(m,t)=\frac{A_R}{2}e^{j2\pi\left(a\tau t-a\tau mT_c-\frac{1}{2}a\tau^2+f_0\tau\right)}\Pi\left(\frac{t-\frac{T_c+\tau}{2}}{T_c-\tau}\right) \quad (33.)$$

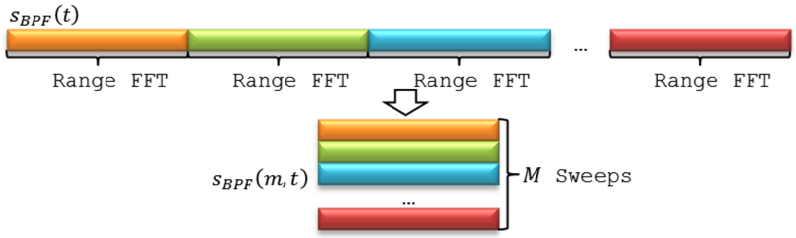


Figure 2-4 Deramped signal rearranged to 2D 'matrix' form

This 'matrix' looks a little abnormal, because each row is actually a continuous signal with $t\in(-\infty,+\infty)$ rather than finite number of discrete samples. However, this configuration is easier for mathematical deduction. There are in total M rows of this signal and the first Range Fourier Transform on each row of the continuous aperiodic signal will be:

$$\mathrm{F}\{s_{BPF}(m,t)\}=\frac{A_R}{2}e^{j2\pi\left(f_0\tau-a\tau nT_c-\frac{1}{2}a\tau^2\right)}\int_{-\infty}^{+\infty}e^{j2\pi a\tau t}\Pi\left(\frac{t-\frac{T_c+\tau}{2}}{T_c-\tau}\right)e^{-j2\pi ft}dt \quad (34.)$$

The exponential term $e^{j2\pi a\tau t}$ in the Fourier Transform will result in a shift in frequency domain. The time shift in the scaled gate function $\Pi(\frac{t}{T_c-\tau})$ will result in a phase shift in frequency domain with scaling factor which is according to the following Fourier Transform pair:

$$\mathrm{F}\{x(a(t-\tau))\}=\frac{1}{a}X\left(\frac{f}{a}\right)e^{2\pi\tau f} \quad (35.)$$

where $X(f)$ is the Fourier Transform of $x(t)$. So the Fourier Transform in Equation (34.) is then:

$$S_{RS}(m,f)=\mathrm{F}\{s_{BPF}(t,m)\}$$
$$=\frac{A_R}{2}e^{j2\pi\left(f_0\tau-a\tau mT_c-\frac{1}{2}a\tau^2\right)}e^{-j2\pi\left(\frac{T_c+\tau}{2}\right)f}(T_c-\tau)\mathrm{sinc}\left[(T_c-\tau)(f-a\tau)\right] \quad (36.)$$

which we call Range-Sweep Spectrum (sRS). The sinc function in Equation (36.) is defined as:

$$\mathrm{sinc}(x)=\frac{\sin(\pi x)}{\pi x} \quad (37.)$$

The peak of the sinc function appears at $f=a\tau$ which reveals the distance between the radar system and the target. Observing the most complex term in Equation (36.) $e^{j2\pi\left[f_0\tau-a\tau mT_c-\frac{1}{2}a\tau^2-\left(\frac{T_c+\tau}{2}\right)f\right]}$ which results in a phase shift in the Range-Sweep spectrum, we can ignore the small term $-a\tau^2/2$. Actually, in FMCW radar systems, the designed chirp time T_c is always much larger than the maximum unambiguous round trip time of flight τ which is:

$$T_c >> \tau \quad \text{for any } \tau \text{ within the designed range}$$

The terms containing T_c are due to the signal misalignment with respect to the origin, so they can also be neglected which gives a simplified expression of Range-Sweep spectrum after this first Fourier Transform:

$$S_{RS}(m,f)=\frac{A_R}{2}(T_c-\tau)e^{j2\pi f_0\tau}\mathrm{sinc}\left[(T_c-\tau)(f-a\tau)\right] \quad (38.)$$

Recall that the round trip time of flight τ is actually a function of m specified by Equation (24.):

$$S_{RS}(m,f)=A_R^{'}e^{j2\pi f_0\tau(m)}\mathrm{sinc}\left[(T_c-\tau(m))(f-a\tau(m))\right]$$
$$=A_R^{'}e^{j2\pi f_0\frac{2r+2mv_rT_c}{c}}\mathrm{sinc}\left[\left(T_c-\frac{2r+2mv_rT_c}{c}\right)\left(f-a\frac{2r+2mv_rT_c}{c}\right)\right] \quad (39.)$$

where $A_R'=\frac{A_R}{2}(T_c-\tau)=\frac{AB}{4}(T_c-\tau)$. Figure 2-5 shows a noiseless Range-Sweep spectrum of single moving target as an example.

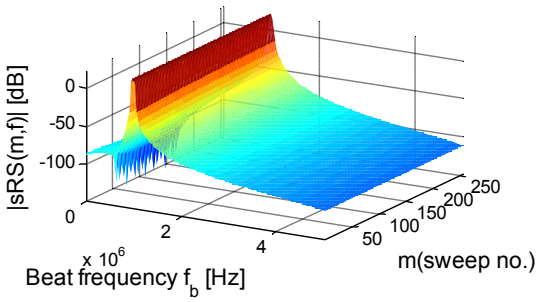


Figure 2-5 Range-Sweep spectrum after 1st FFT with single target

Though m is a discrete variable, for simplicity, it is treated here continuous as well. Then the second Fourier Transform is done on $S_{RS}(f,m)$ with respect to m:

$$S_{RV}(v,f)=\mathrm{F}\{S_{RS}(m,f)\}$$
$$=A_R^{'}e^{j2\pi f_0\frac{2r}{c}}\int_0^M e^{j2\pi f_0\frac{2v_rT_cm}{c}}\mathrm{sinc}\left[\left(T_c-\frac{2r+2mv_rT_c}{c}\right)\left(f-a\frac{2r+2mv_rT_c}{c}\right)\right]e^{-j2\pi vm}dm$$
$$(40.)$$

During one measurement which consists of M sweeps, assume that the movement of the target is small compared to the distance. We can ignore the term $2mv_rT_c$ in the sinc function and take the sinc function out of the integration as:

$$S_{RV}(\nu,f)=A_R^{'}e^{j2\pi f_0\frac{2r}{c}}\text{sinc}\left[\left(T_c-\frac{2r}{c}\right)\left(f-\alpha\frac{2r}{c}\right)\right]\int_0^M e^{j2\pi f_0\frac{2v_rT_cm}{c}}e^{-j2\pi\nu m}dm \qquad (41.)$$

Change the integral limits to infinity and plug in one gate function term:

$$\begin{aligned}S_{RV}(\nu,f)&=A_R^{'}e^{j2\pi f_0\frac{2r}{c}}\text{sinc}\left[\left(T_c-\frac{2r}{c}\right)\left(f-\alpha\frac{2r}{c}\right)\right]\int_{-\infty}^{+\infty}e^{j2\pi\frac{2f_0v_rT_c}{c}m}\Pi\left(\frac{n-\frac{M}{2}}{M}\right)e^{-j2\pi\nu m}dn\\&=A_R^{'}e^{j2\pi f_0\frac{2r}{c}}\text{sinc}\left[\left(T_c-\frac{2r}{c}\right)\left(f-\alpha\frac{2r}{c}\right)\right]M\text{sinc}\left[M\left(\nu-\frac{2f_0v_rT_c}{c}\right)\right]e^{-\frac{j2\pi M}{2}\nu} \qquad (42.)\\&=A_R^{'}Me^{j2\pi f_0\frac{2r}{c}}e^{-\frac{j2\pi M}{2}\nu}\text{sinc}\left[\left(T_c-\frac{2r}{c}\right)\left(f-\alpha\frac{2r}{c}\right)\right]\text{sinc}\left[M\left(\nu-\frac{2v_rT_c}{\lambda}\right)\right]\end{aligned}$$

There are two sinc functions in this second Fourier Transform result. Their crossing point reveals both range and velocity information, so we call this spectrum the Range-Velocity spectrum (s2D). The peak point in this 2D signal is at:

$$\begin{aligned}f_{peak}&=\alpha\frac{2r}{c}=f_b\\\nu_{peak}&=\frac{2v_rT_c}{\lambda}=f_DT_c\end{aligned} \qquad (43.)$$

Figure 2-6 shows a Range-Velocity spectrum example which is the Fourier Transform result of Figure 2-5.

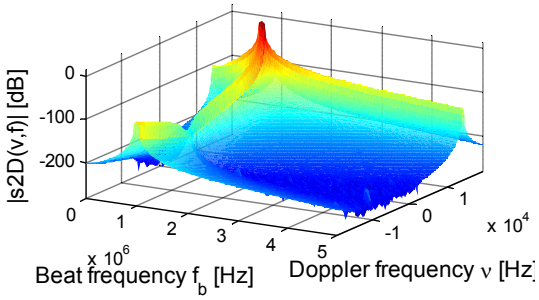


Figure 2-6 Range-Velocity spectrum after 2nd FFT

Equation (42.) is for single moving target, in real situations, there are usually multiple targets moving with different speeds, so the Range-Velocity spectrum is a superposition of several single target spectra like:

$$S_{RV}(\nu,f)=\sum_{i=0}^{K-1}A_R^{'}(i)Me^{j2\pi f_0\frac{2r_i}{c}}e^{-\frac{j2\pi M}{2}\nu}\text{sinc}\left[\left(T_c-\frac{2r_i}{c}\right)\left(f-\alpha\frac{2r_i}{c}\right)\right]\text{sinc}\left[M\left(\nu-\frac{2v_{ri}T_c}{\lambda}\right)\right] \qquad (44.)$$

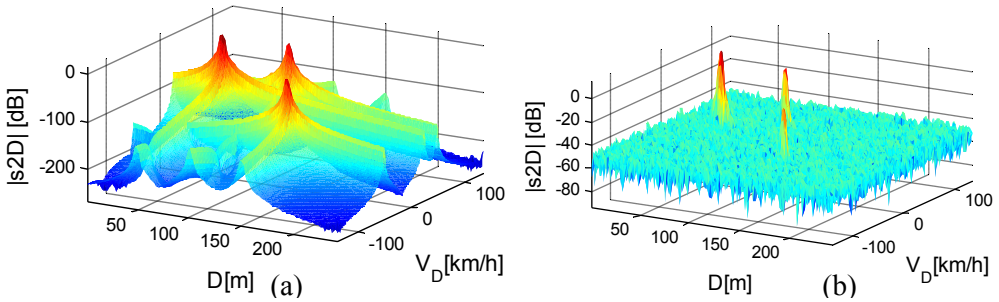


Figure 2-7 Range-Velocity spectrum of 3 targets with different speeds (a) without noise (b) with AWGN

Figure 2-7 shows two Range-Velocity spectrums of 3 targets with different speeds. Up to this point, the 2D Fourier Transform Processing has been theoretically deducted and by locating the

peak position in the Range-Velocity spectrum, the distance between the target and the radar and target radial speed can be obtained. For FMCW radar system, this spectrum will be processed by the next stages to achieve such functions as target recognition and tracking.

2.3 Measurement Resolutions

Resolutions are the limitations of how close targets can be to each other so that they can still be distinguished. It's one of the most important design targets in radar system. It is known from the previous chapters that important target information is in the frequency domain, therefore the measurement resolutions depend on the frequency resolutions.

From both Equation (19.) and Equation (43.), we know that

$$D = \frac{cf_b}{2\alpha} \tag{45.}$$

According to the Nyquist sampling theorem, the maximum unambiguous frequency the sampled signal can represent is $f_s/2$, so the maximum range is:

$$R_{\max} = \frac{cf_s}{4\alpha} = \frac{cf_sT_c}{4BW} = \frac{cN}{4BW} \tag{46.}$$

where $N = f_sT_c$ is the number of samples in one chirp sweep. For a real Fourier Transform, only half of the spectrum is used where there're $N/2$ frequency lines. So the range resolution is:

$$\Delta R = \frac{D_{\max}}{0.5N} = \frac{c}{2BW} \tag{47.}$$

From Equation (45.) and Equation (46.), we can see that maximum range and range resolution is a **tradeoff**, for a fixed N, the higher the chirp bandwidth, the better range resolution will be but the maximum range will suffer then.

The radial speed v_r comes from the measurement of the Doppler frequency shift f_D, the phase shift between consecutive chirp sweeps is:

$$\Delta\phi = 2\pi f_DT_c \tag{48.}$$

Phase has an unambiguous range between 0 to 2π and for the Velocity FFT, both halves of spectrum contain information because targets can be leaving or approaching the radar system. So the boundary of f_D is:

$$f_{D\max}^{+} = \frac{1}{2T_c} \quad f_{D\max}^{-} = -\frac{1}{2T_c} \tag{49.}$$

Therefore:

$$v_{r\max} = \pm\frac{\lambda}{4T_c} = \pm\frac{\lambda f_s}{4N} \tag{50.}$$

By collecting M consecutive chirp sweeps and doing Fourier Transforms along the direction of m shown in the previous deductions, the spectrum for velocity is divided into $M/2$ cells for both positive and negative parts, so the radial velocity resolution is:

$$\Delta v_r = \frac{\lambda}{4T_c} \Big/ \frac{M}{2} = \frac{\lambda}{2MT_c} = \frac{\lambda f_s}{2MN} \tag{51.}$$

2.4 System Specifications

In the RF front end shown in Figure 2-2, the frequency of $f_c = 77GHz$ is chosen as a signal carrier which resides in the W band (75-110GHz) [54]. This radio frequency band is generally

used for satellite communications, millimeter-wave radar research and military radar targeting and tracking applications. One reason for choosing this band is due to the congestion in lower frequency bands, for example, the 2.4GHz ISM band is heavily used for various communication applications such as WiFi and Bluetooth. Also ultra wide band in 77GHz band will provide fine range resolution due to the relationship between ΔR and BW.

In terms of the deramping block, the best way is quadrature dereamping which is shown in Figure 2-8. It's similar to a quadrature demodulator where the only difference is that one of the input signals is not a constant frequency sine wave but the same chirp signal which is transmitted to the target.

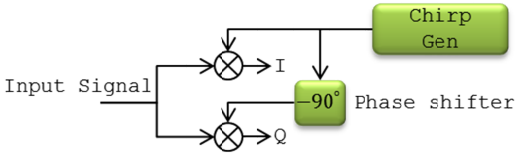


Figure 2-8 Quadrature deramping

The resulting I/Q signal forms a complex representation of the deramped signal which only has a single side spectrum while the noise is occupying the whole spectrum. Therefore the SNR is 3dB higher than non-IQ deramping. The other advantage of quadrature deramping is that it relaxes the ADC sampling rate requirements at the price of doubling the number of ADCs because the highest frequency in the I/Q signal is half of that in the non-I/Q signals. However, besides those advantages, this design does not use quadrature deramping because in 77GHz frequency band, 90 degrees phase shifters are not easy to design.

The designed chirp bandwidth is 1 GHz. and for one complete deramped signal, $N = 1024$ samples will be received by the ADC. The sampling rate of the ADC is $f_s = 10MHz$, so the length of one complete chirp in time is $1024 \times 1/(10 \times 10^6) = 102400\, ns$. An FFT with the same number of points is used to perform the range FFT operation on each sweep.

$M = 128$ complete sweeps are processed through the range FFT and collected to form a 128×512 2D matrix which is the Range-Sweep spectrum. The 2nd FFT processing, which is the Velocity FFT, is 128-points long (the same as M). Substituting these design parameters into the equations in the previous chapter, we have:

Table 2-1 System specifications and parameters overview

Parameter	Value
RF Frequency	77 GHz
Sweep Bandwidth	1 GHz
Range Resolution	0.15 m
Radial Velocity Resolution	0.15 m/s
Max Unambiguous Range	76.75 m
Max Radial Velocity	9.51 m/s
Range FFT Length	1024 points
Velocity FFT Length	128 points
Sampling Frequency	10 MSps
Sweep Time	102.4 us

3 Data Compression in FMCW Radar Signal Processing Flow

From Chapter 2, it can be seen that a certain amount of memory is necessary for performing the 2D FFT signal processing. The memory is used to store the first FFT results (Range-Sweep spectrum or sRS) which serve as intermediate values for performing the second FFT. Figure 3-1 shows the memory blocks in this system. Not much can be done to reuse the memory space because the 2 FFT operations are done in a transposed way which is mathematically described in Chapter 2. This is to say, all the sweeps have to be collected before starting the second FFT operation column by column.

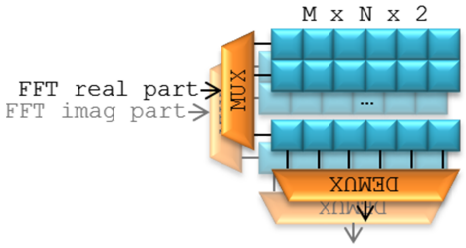


Figure 3-1 Memory in the 2D FFT processing

The first FFT operation on one sweep will produce *N* complex values, however, due to the symmetrical spectrum, only half of it is enough to extract the range information which is N/2 complex data. When doing the second FFT on each column, the full M-long complex data is useful because velocity can be either positive or negative. Each complex value has 16-bit data in its real and imaginary parts. So the total memory size is:

$$\begin{aligned}\text{Size of Mem} &= M \times \frac{N}{2} \times 2 \times 16 \text{ bits} \\ &= 128 \times 1024 / 2 \times 2 \times 16 = 2097152 \text{ bits} = 256 \text{ KBytes}\end{aligned} \tag{52.}$$

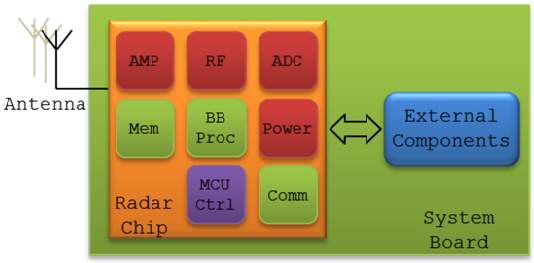


Figure 3-2 Single chip SoC radar solution

This automotive radar system is designed for consumer market vehicles therefore cost is one of the most sensitive issues. It is designed to be a system-on-chip (SoC) radar solution which means all blocks such as RF transceivers, amplifiers, digital filters, baseband processor, control logic and communication ports will be integrated into one chip. The concept is shown in Figure 3-2. SoC also simplifies system design by reducing the number of external components.This will reduce the design complexity and time to market thus giving advantages to automotive electronic system companies in the highly competitive market.

However, for a SoC design, the calculated amount of system memory is relatively large. For example, the high-end C667x [55] multicore DSP from Texas Instrument which costs more than 120 USD only has 64-256KB on-chip L1 SRAM memory. Figure 3-3 shows a chip die photo of a DSP with 64Kbytes ROM, 19Kbytes RAM. Almost half of the die area is occupied by memory blocks.

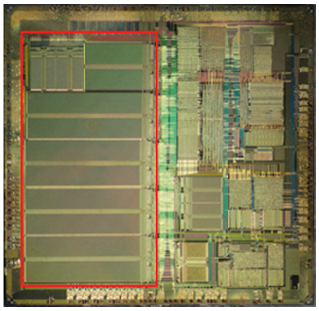


Figure 3-3 Chip die photo of TI TMS320C50 DSP [56] (memory blocks shown in red box)

Furthermore, it's possible that multiple transceivers and baseband processors will be put into one chip to form phased array configuration which will make the memory demands even more intense. Therefore memory compression in the radar signal processing flow is necessary.

4 Range Dependent Variable Length Encoding

In this Chapter, we will discuss the design of one of the two data compression schemes in this thesis – the Range Dependent Variable Length encoding (RDVLE). It's a lossy compression scheme taking advantage of the received power-range model. To further improve the compression ratio, the noise floors in different encoding regions are raised differently according to the Uniform Dynamic Range encoding (UDRE). Due to the statistical nature of the incoming data, clipping errors might happen resulting in fake target peaks. To deal with this problem, a Clipping Reduction Method (CRM) algorithm is proposed with consideration of hardware implementation complexity and cost.

4.1 Received Signal Power Profile

Chapter 2.1.1 has shown how the simple radar equation is obtained. The received echo power of a point target in a lossy medium is characterized by the following equation.

$$P_r(r) = \frac{P_t G_t G_r \sigma \lambda^2}{(4\pi)^3 r^4 L_m^2(r)} (W) \tag{53.}$$

where P_t is the transmitted power, G_t and G_r are the radar transmitting and receiving antenna gain respectively, σ is the RCS, λ is the wave length, r is the range between the radar and the target and L_m is the attenuation factor of the lossy medium.

It can be clearly seen that the received power $P_r(r)$ is inversely proportional to the 4th power of the range when other conditions are the same. This means different echoes from targets at different ranges will have different power levels which is governed by the radar equation. In terms of the incoming data, the sRS will have a decreasing magnitude along the range (or f_B) direction.

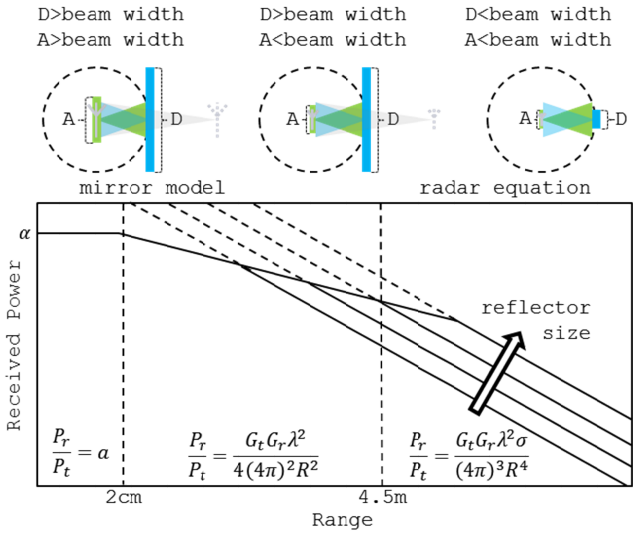


Figure 4-1 Range vs. Received power in different scenario [57]

In reality, the received power and range relationship is not that ideal as inversely proportional to the 4th power of range and Figure 4-1 shows three different scenarios.

If the distance between the radar and the target is small, the target acts like an electromagnetic wave mirror and there will be much more energy scattered back to the radar so the range-received power curve drops down slower in the beginning part which is shown in the first two cases in the above figure. For a faraway target, only part of the beam illuminates the target and the curve drops much faster than the previous two cases.

Though more sophisticated than a single radar equation, this 3-section model is still far from realistic. The range-received power relationship will be even more complicated with considera-

tion of reflections from environmental objects and multi-path effects. However, our target is not to develop a superior range–received power model but to use a model with certain degree of accuracy to help data compression. From the introduction to data compression in Chapter 1 we know that information about signal properties (such as symbol probability) is crucial to achieve compression. Here, this range–received power profile is another extra piece of information about the signal and will help to achieve compression.

In terms of data compression, this range–received power model gives a deterministic rough estimation of the data sample magnitude for different range which means the sample magnitude in the Range-Sweep spectrum is roughly known according to the position. By knowing the rough magnitude for each sample, the number of bits needed to store a certain sample can be roughly known and instead of using fixed bit length for every sample, variable bit length data is stored to save the total memory.

In the system every point in the complex Range-Sweep spectrum (Range-Sweep spectrum or sRS) is represented by two signed fixed point values. As shown in Figure 4-2, the length of the fixed point data including sign bit is in total WL, with FL bits in the fractional part and 1 bit for the sign. The data is stored in 2's complement format.

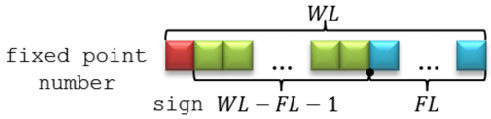


Figure 4-2 Fixed point data format

Dynamic range of a fixed point number format is the ratio of the maximum absolute value representable and the minimum absolute value representable. For the signed fixed point number mentioned above, the dynamic range is:

$$DR_1 = \frac{2^{WL-1}-1}{2^{FL}} / 2^{-FL} = 2^{WL-1} - 1 \approx 2^{WL-1} \approx 6.02(WL-1)\,\text{dB} \qquad (54.)$$

One very similar but different concept is the Signal to Noise ratio (SNR) which is defined as the ratio of the signal power to the noise power. The minimum possible noise value is the quantization noise which is non-linear and signal dependent. For a full scale perfect sine wave, when quantized to the above signed fixed point number format, the SNR [58] is:

$$SNR_1 = 20\log\left(2^{WL-1}\sqrt{3/2}\right) \approx 6.02(WL-1) + 1.76\,\text{dB} \qquad (55.)$$

which is very close to the DR of the number format. One detail to be noted is that, under this definition of the number dynamic range, an unsigned number with the same total bit length WL provides 2 times the DR of its signed counterpart. However, the data in sRS is signed, so signed fixed point data format has to be used. In the rest part of this thesis, the representation of the number format like $[sign, WL, FL]$ is used to represent the fixed point number format, for example [1,16,14] means a signed 16-bit number with 14 bits in the fractional part, 1 bit in the sign and 1 bit in the integral part.

There is a **tradeoff** between model accuracy and compression performance. The more accurate the model is, the more exact the bit length can be assigned to data from different ranges and the less bits or memory space is wasted. However, the more accurate the model is, the more overhead there will be, both in memory and computational requirements because a more complex model has to be stored inside the system.

To balance the performance and overhead, a model designed by NXP based on the segmented model is used. The model is shown in Figure 4-3.

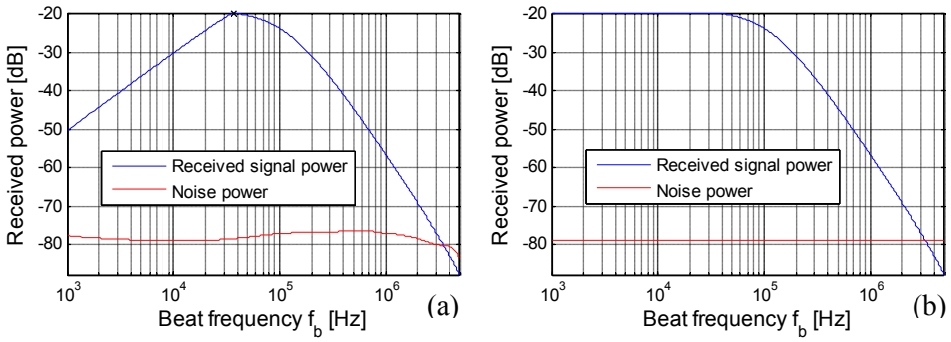


Figure 4-3 (a) Received power–range model developed by NXP based on the segmented model (b) simplified version which is actually used

Figure 4-3 (a) shows the original model. There is an obvious drop in the very low frequency (low range) region which is due to the attenuation from the BPF shown in Figure 2-2. This attenuation is intended to remove the echo from car parts that are relatively static to the radar, for example, the front bumper. Another obvious feature is the non-uniform noise floor which is dependent on range. For simplicity, a simplified version of this model shown in Figure 4-3 (b) is used. First the attenuated segment in the low frequency region is made flat and in simulation because relatively static car parts are not considered. Second, the noise floor is made flat across the whole spectrum which eliminates the need to store the range dependent noise floor.

4.2 Car RCS Distribution

Beside the model inaccuracy, another important factor influencing the compression performance is the automotive Radar Cross Section. As described in Chapter 2.1.1, RCS, which is usually written as σ, is a fictional area. It is often not the same as the target physical cross-section area and usually depends very largely on the incident elevation and azimuth. [59] RCS represents the ability of the target to reflect radar signals in the direction of the radar receiver.

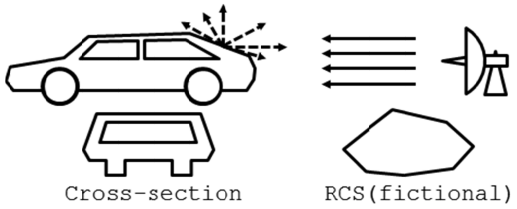


Figure 4-4 Object cross-section and the RCS

The RCS expressions of two regular shape metal objects are listed below [60]:

$$\text{Flat Plate: } \sigma_{fp} = \frac{4\pi w^2 l^2}{\lambda^2} \qquad \text{Sphere: } \sigma_s = \pi r^2 \tag{56.}$$

where w and l are the width and length of the plate and r is the radius of the sphere. The RCS of a sphere is independent of frequency if both the range and the sphere radius are much larger than incident wavelength. It's obvious from the above two equations that the unit for RCS is m^2 and usually RCS is expressed in $dBsm = 10\,log_{10}\,RCS(m^2)$ which is decibels referenced to a square meter to be directly added to the received power expressed in dB. Figure 4-5 shows a real measurement of RCS of a human dummy in 24GHz and 77GHz bands for 0-360 degree azimuth with 1 degree step. The blue line and the red dash line represent two consecutive measurements in the same condition. It can be seen that the RCS depends largely on azimuth and even consecutive measurements in time have some difference.

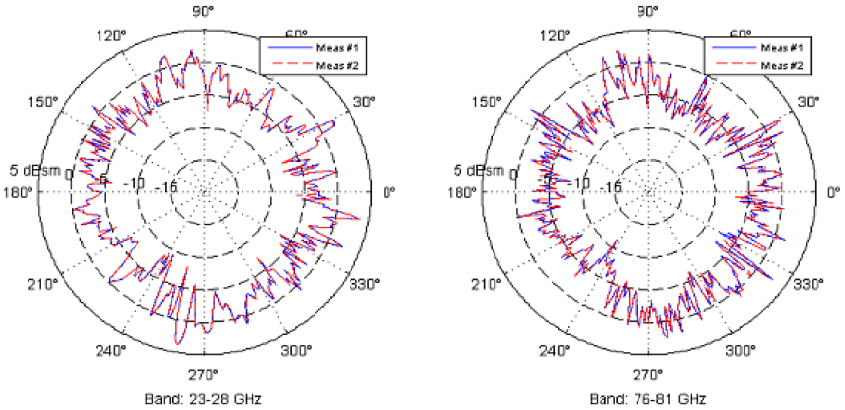


Figure 4-5 RCS of a human dummy 0-360 degree azimuth in 24GHz and 77GHz band [61]

Besides incident elevation and azimuth, the RCS is influenced by many other factors such as the shape and material of the target, incident wave frequency and so on. According to [62], the major reflector locations on the back of a car are bumper, license plate, tail lights, muffler, rear suspension and roof/rear window join shown in Figure 4-6. All the above factors make it actually not a deterministic value even for the same target across consecutive measurements, let alone in real automotive radar application scenario, both the radar and the target are moving causing the environment to be different from time to time.

Figure 4-6 Major RCS contributions from the back of a car

Due to the impracticability to have a mathematical model for very complex target, many efforts were devoted to real measurement and statistical modeling of the RCS. The RCS of pedestrians and dummies are measured in [61] and [62]. The RCS of cars are measured in [62] and [63]. In terms of statistical modeling, there are several conventional ways such as chi-square model, log-normal model and Rice model etc. Take the chi-square model for example which is a general target model, the probability density of RCS σ is:

$$p(\sigma)=\frac{m}{\Gamma(m)\sigma_{av}}\left(\frac{m\sigma}{\sigma_{av}}\right)^{m-1}e^{-\frac{m\sigma}{\sigma_{av}}}I_{[0,\infty)}(\sigma) \tag{57.}$$

where σ_{av} is the mean value of RCS, m is half of the "number of degrees of freedom", $\Gamma(m)$ is the gamma function. The parameters are determined through data fitting using the measurement RCS data.

Instead of the conventional model, researchers from Toyota CSRS in [62] measured 24 real vehicles, each 30 times and fit the measured RCS into Weibull distributions. The result is shown in Figure 4-7. The reason why the Weibull distribution is used is that it's very flexible and the shape of the distribution can change drastically according to its two parameters: scale (β) and shape (α). Mathematically, the probability density function of Weibull distribution is:

$$p(X)=\begin{cases}\frac{\alpha}{\beta}\left(\frac{X}{\beta}\right)^{\alpha-1}e^{-\left(\frac{X}{\beta}\right)^{\alpha}} & (x\geq 0)\\ 0 & (x<0)\end{cases} \tag{58.}$$

where $\alpha>0$ is the shape parameter and $\beta>0$ is the scale parameter. By changing the shape parameter, the Weibull distribution interpolates between exponential distribution ($\alpha=1$) and the Rayleigh distribution ($\alpha=2$).

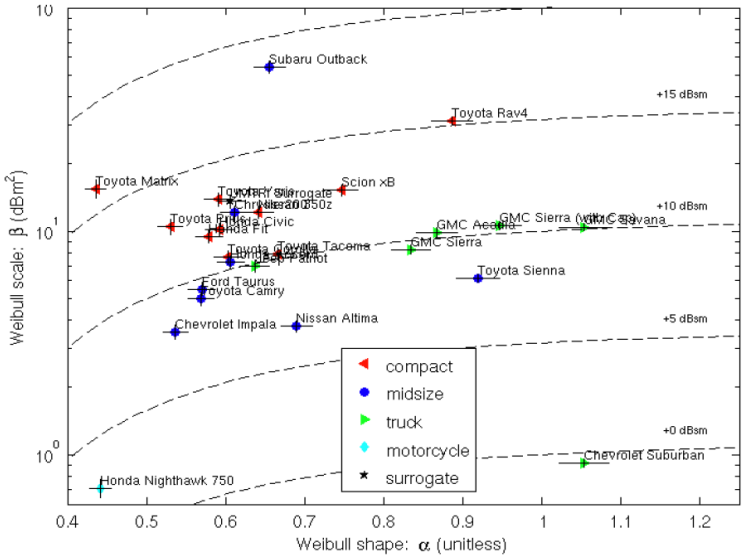


Figure 4-7 Weibull distribution parameters of the RCS of 24 measured cars [62]

Figure 4-8 shows the probability density function and cumulative distribution function of Weibull distribution with different parameters.

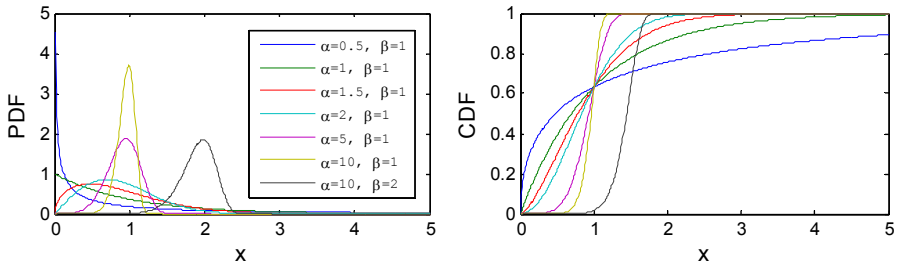


Figure 4-8 PDF and CDF of Weibull distribution under different parameters

It can be clearly seen that the shape of the PDF depends greatly on the shape parameter α in which way it may fit many different measurements well. With the two parameters known, the mean and variance of a Weibull distributed random variable are:

$$E(X)=\alpha\Gamma\left(1+\frac{1}{\beta}\right)$$
$$\mathrm{var}(X)=\alpha^2\left[\Gamma\left(1+\frac{2}{\beta}\right)-\left(\Gamma\left(1+\frac{1}{\beta}\right)^2\right)\right] \quad (59.)$$

Treating the RCS as a random variable further complicates the compression problem in the way that even if the most accurate range-received power model is used, the randomness in RCS will make the estimation of the word length inaccurate and only part of the samples can be guaranteed to fit completely into the designed word length. Here comes another **tradeoff**: the longer word length one chooses, the less chance the samples will overflow due to insufficient length but obviously the compression ratio will be lower.

4.3 FFT Processing Gain and Bit Length Growth

As mentioned in Chapter 2, 2D FFT processing is the core of the signal processing for FMCW radar, these 2 FFT operations introduce extra gains and make the bit length of a sample grow after each operation which has to be considered when designing the right word length for the RDVLE compression.

Now, let's consider the FFT or the Discrete Fourier Transform. An N-point FFT operation will generate N complex results which represent N frequency bins from $-f_s/2$ to $f_s/2$. For a real input, the spectrum is Hermitian symmetrical (or $X(f) = X^*(-f)$). For a single tone, there will

be two lines in the whole power spectrum which are symmetrical according to the y-axis and the whole signal energy is 'concentrated' in that two bins. However, the spectrum of noise spreads across the whole power spectrum. Especially when the noise is white, it will uniformly spread in the whole spectrum of N bins resulting in a gain of:

$$G_{FFT} = 10\log_{10}(N/2) \tag{60.}$$

Figure 4-9 shows an example of FFT processing gain. The input signal is a 100Hz sine wave sampled at $f_s = 1000Hz$ with white noise of variance 1:

$$x[n] = \sqrt{2}\sin\left(2\pi \times 100 \times \frac{n}{1000}\right) + n[n] \tag{61.}$$

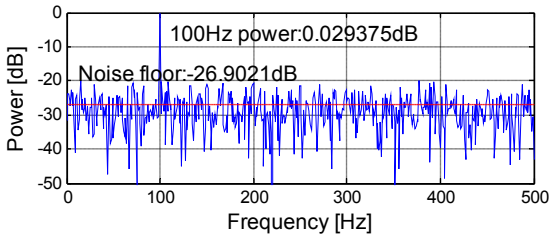


Figure 4-9 FFT processing gain example (half spectrum)

The power of the sine wave is: $0dB$, the FFT processing gain is: $G_{FFT} = 10\,log_{10}(1000/2) = 27.00dB$ which is in correspondence with the value calculated from simulation.

In terms of FMCW radar, on one hand, this FFT processing gain is beneficial to target detection because it literarily lowers the noise floor and makes signals buried in the noise visible to the system, on the other hand, the dropped noise floor will necessitate more bits for storing the spectrum (sRS, exactly speaking) to capture all the details in the signal.

Assume the analog front end in this system is perfect and the noise floor is uniform. Assume also the AGC subsystem works perfectly so the last bit in the ADC output represents the signal noise floor (or the noise floor is normalized to 0dB). In this case, to represent the ADC result, no FL bit is needed and WL is determined by the number of ADC bits. As described in Chapter 2.4, a 16-bit ADC ([1,16,1] format) and 1024 points FFT are used. So the FFT processing gain is: $G_{FFT} = 10\,log_{10}(1024/2) = 27.09dB$ which means the noise floor drops down from 0dB to -27.09dB. Then about 4.2 more bits are needed in the FL parts to represents the FFT result.

4.4 Range Dependent Variable Length Encoding

After the 1st FFT, or the Range FFT, targets at different ranges will be separated and the magnitude of each bin of the Range-Sweep spectrum represents the received echo power. With all the above considerations, the RDVLE can be designed.

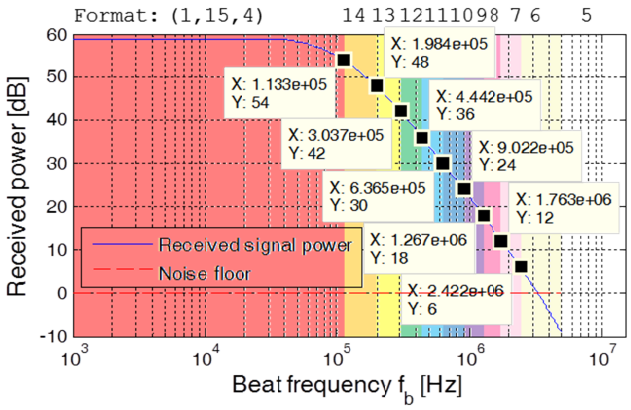


Figure 4-10 The RDVLE with bit length zones (noise floor is normalized to 0dB)

Figure 4-10 shows the RDVLE scheme with bit lengths in different zones. There are in total 11 coding zones which are marked with difference colors. The starting point of each zone is 6dB lower than the starting point of previous zone so 1 bit can be reduced. Though the ADC is 16 bits, the received power model we used here only has a maximum power of 60dB which cannot occupy the whole ADC range. So 10 bits in the integer part are needed, with the FFT processing gain, 4 extra bits in the fractional part are needed. This is the reason why the data in the leftmost zone is in the format of [1,15,4]. The WL reduces in the following zones from 14 to 5 with FL is equal to 4 constantly. With this scheme, the total number of bits needed for the complex 1st FFT result is $128 \times 7596\,\text{bits/sweep} = 972288$ bits. So the CR under this scheme is:

$$CR_{RDVLE} = 256\,\text{KBytes}/972288\,\text{bits}=2.16 \tag{62.}$$

4.5 Uniform Dynamic Range Encoding

As shown in Figure 4-10, due to the received power-range relationship, the peaks which represent near targets are much higher than those which represent faraway targets, or more precisely speaking, to the target detection stage following the 2D FFT processing, the SNR of the signal representing each target is not the same. It is generally known that detection performance is in positive correlation with the SNR, so for those targets with high SNR, it's easier for the system to detect and vice versa.

Generally speaking, the higher the SNR, the better the detection performance is. However, detection rate will saturate beyond some point with an increasing SNR. This means, for near targets, the SNR might well exceed the saturation points and the excess SNR does not contribute much to detection rate but still occupies more bits in the memory simply because it has higher magnitude. To cope with this situation and further compress the signal, we proposed Uniform Dynamic Range Encoding (UDRE) based on RDVLE.

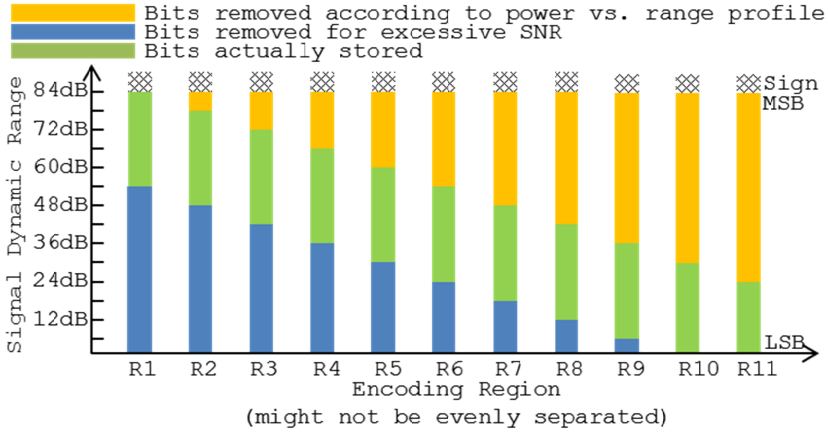


Figure 4-11 The Uniform Dynamic Range Encoding

Figure 4-11 shows a general case of UDRE based on RDVLE. Each vertical bar represents one bit-removing scheme for the fixed point data in that encoding region in the 1st FFT result. The encoding regions are determined by the received power-range model and do not need to be evenly separated on x-axis. All bars have the same lengths which represent the original data length after 1st FFT. The length of fractional part, as stated in Chapter 4.3, is determined by the FFT length.

The yellow parts in the bars have different length and they represent the bits that are removed due to RDVLE scheme discussed in the previous Chapter according to received power-range model. In correspondence to the model, the yellow parts are growing with increasing distances (or beat frequency). The blue parts represent the bits removed due to excessive SNR. More bits are removed in the low range regions. The green parts represent the bits that are left which are actually stored in the memory. Removing bits from the LSBs means to raise the noise floor which reduces the SNR. All green parts have the same length which means targets in each re-

gion have the same maximum SNR if RCS variation is not considered. This is why this encoding scheme is called the Uniform Dynamic Range Encoding. The last encoding region doesn't have the same number of bits in the green part simply because it doesn't have enough power in the original signal itself.

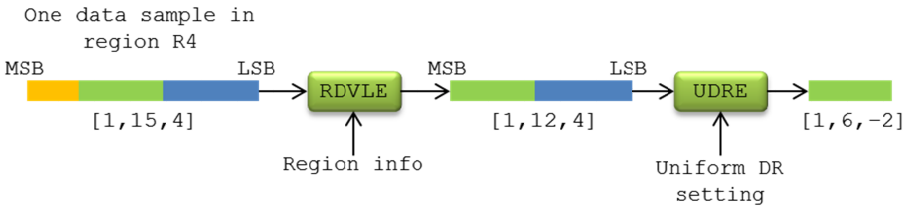


Figure 4-12 The RDVLE + UDRE encoding flow (with example in R4 region)

Figure 4-12 shows an example of the whole encoding flow in R4 region. In this example, after 1st FFT, the data format for every sample in the encoding region is [1,15,4]. The region information is provided to the RDVLE encoder. According to the received power-range model in the RDVLE encoder, the first 3 MSB bits (except the sign bit) are meaningless due to power attenuation. Then the first 3 MSB bits which are colored in yellow are removed by the RDVLE encoder. The output data becomes [1,12,4] which is sent to the UDRE encoder. A preset of Uniform DR setting of 5 bits is used by UDRE encoder so 6 last LSB bits are removed by UDRE. The final result is in format [1,6,-2] which is signed 6 bits fixed point number. Note that the FL length of -2 means the LSB represents 2^2 but not 2^0.

Figure 4-13 shows an example of final Range-Velocity spectrum using RDVLE + UDRE. It's very clear that the first three target peaks reside in higher noise floors than the rest targets. The peaks heights compared to their own noise floors are more or less the same reflecting the name "Uniform Dynamic Range".

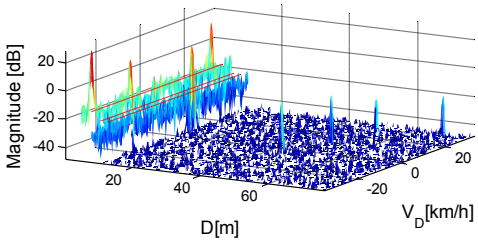


Figure 4-13 Example result after the RDVLE + UDRE

Though it saves extra bits, UDE makes the noise floor non-uniform which is a little problematic for the following detection stages for locating the exact positions of the peaks. As in the figure, the leftmost noise floor is even higher than the far-away target peaks, so a single threshold cannot distinguish all peaks. Target or peak detection is not a topic for this thesis project, so it won't be detailed here. One potential way to do peak detection in non-uniform noise floor like the above figure is regional detection.

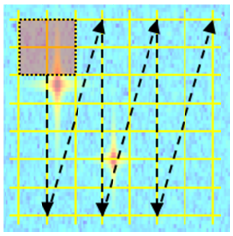


Figure 4-14 Regional scan peak (target) detection

The concept of regional scan peak detection is shown in Figure 4-14. An $n \times m$ grid (shown in red, m=n=2) scans in the noted direction as the sweeps comes in. The average magnitude or a weighted sum of the $n \times m$ sample magnitudes are compared to a regional threshold according to the VLE + UDRE scheme to determine if there is a peak. By choosing proper m, n and regional threshold, the influence of the steps in the noise floor on target detection can be reduced.

4.6 Clipping Error and Clipping Reduction Method

When LSB bits are removed, the error between the encoded signal and the original one is not so large but when MSB bits are removed, the error can be quite significant due to their higher weights which is shown in Figure 4-15. Note that MSBs are removed in a way that the modified value will saturate if the original value cannot fit into the new format after bit removal.

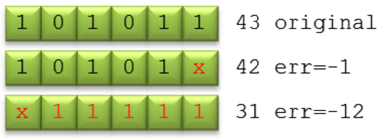


Figure 4-15 Error cause bit removing (data in unsigned binary, x means removed)

When the received power-range model is accurate, there won't be any errors because the MSB bits that are removed are totally unused, however, as described in Chapter 4.2, accurate modeling is neither possible nor economical in terms of storage space. So there will always be cases where MSB bits are removed information is lost and the signal is clipped.

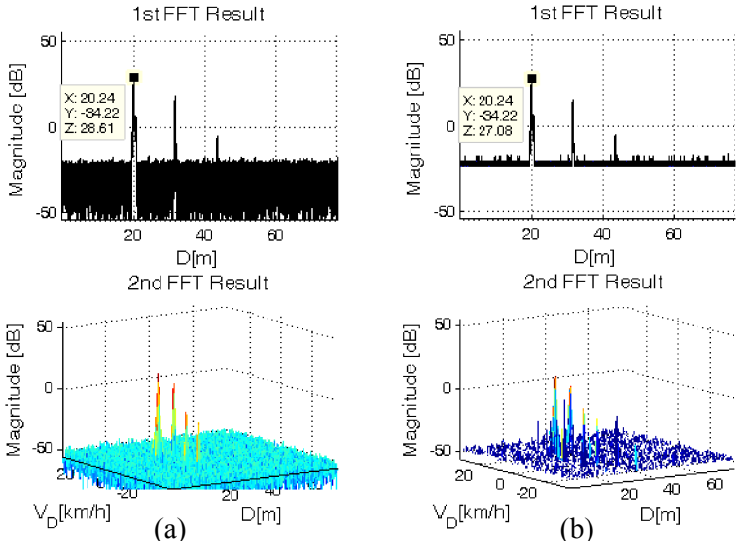


Figure 4-16 (a) All FP processing, no bits removed (b) RDVLE scheme with clipping happened

Figure 4-16 shows one case of clipping introduced by RDVLE. Figure 4-16 (a) shows the reference results from all floating point processing without any data compression and bits removing while Figure 4-16 (b) shows results with RDVLE encoding. It is clear in the two 1st FFT results that the signal from the target at a range of 20m is clipped because of MSB removing in RDVLE encoding. When the clipped 1st FFT result is used in the 2nd FFT operation, new peaks appears near the 20m targets in the second FFT result which is the Range-Speed spectrum (s2D). The new peaks near the target peak will influence the detection performance by making the detector recognize fake targets.

When a signal is clipped, there will be harmonics generated which means new frequency components are present. The general idea of reducing distortions is to keep the signal as similar as original one, however, the 2D FFT processing makes other approaches possible.

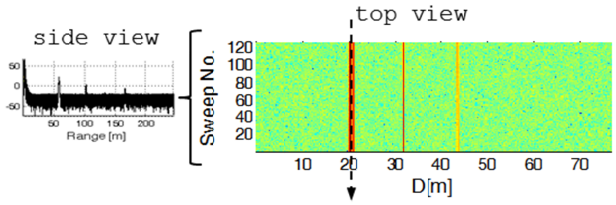


Figure 4-17 Direction of the 2nd FFT (shown in dash arrow line)

Beyond the general idea, as discussed in Chapter 2.2.2 and shown in Figure 4-17, the 2nd FFT is done in orthogonal direction with respect to the 1st FFT due to the fact that velocity information is contained in the phase change between different sweeps. This means our target to reduce clipping and clipping induced harmonics is not in rows but in columns.

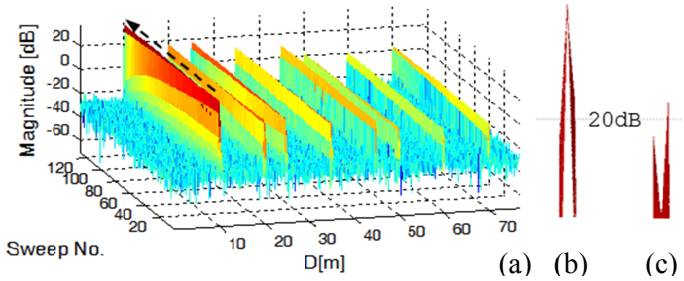


Figure 4-18 Consistent scaling in column (a) sRS (b) detail view of the first peak unscaled in the noted viewing direction with dash arrow line (c) detail view of the first peak scaled down by 4

The unique nature of the 2D FFT makes it possible to scale columns in the result of 1st FFT without influencing the position of the peak in the final Range-Velocity spectrum (s2D). Figure 4-18 (a) shows the result of 1st FFT (or sRS), Figure 4-18 (b) shows a detail view of the peak at 10-meter in the noted viewing direction. This peak is then scaled down by a factor of 4 shown in (c). Viewing in terms of row, it's seriously distorted, however, in terms of column, the scaled column still preserve the phase change information with a lower magnitude.

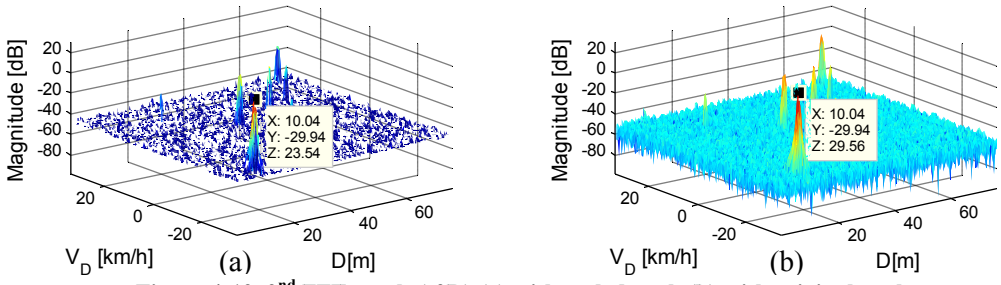


Figure 4-19 2nd FFT result (s2D) (a) with scaled peak (b) with original peak

Figure 4-19 shows the 2nd FFT result (or s2D) for the scaled and unscaled situation. The velocity of the target at 10-meter is revealed, and the target peak of the scaled input is about 6dB lower than the unscaled one. Here comes another **tradeoff**: SNR and distortion. The unscaled high peak in the 1st FFT result will result in higher target peaks in the s2D which will benefit the following detection stages but at the risk of generating harmonics due to clipping. However, those peaks that are clipped are usually already too high to fit in the data format of that encoding region. So sacrificing 3-6 dB SNR in trade of low distortion is reasonable.

Based on the above discussion, we proposed the following Clipping Removal Method. It's a dynamic scaling algorithm by keeping record for each column where the scaling has happened.

CRM algorithm:

1. *Initialize column scaling position record cPos[N/2][K]*
2. *Initialize scaling factor sc[N/2] to all 1, sc increment factor to b.*
3. *Initialize magnitude threshold cTh[N/2] according to the RDVLE scheme.*
4. *Read in N samples in the current sweep and do the 1st FFT on this sweep (row).*
5. *Divide each FFT result in current sweep with corresponding sc[n].*
6. *Compare each result in current sweep with corresponding cTh[n].*
7. *If result n is larger than cTh[n], clipping happens. Increase scaling factor to b*sc[n], divide the original current sample with the new sc[n], append current sweep number in cPos[n][end].*
8. *Repeat step 3-6 until required number of sweeps are processed.*
9. *Scale the previous unscaled parts using cPos[N/2][K] information.*

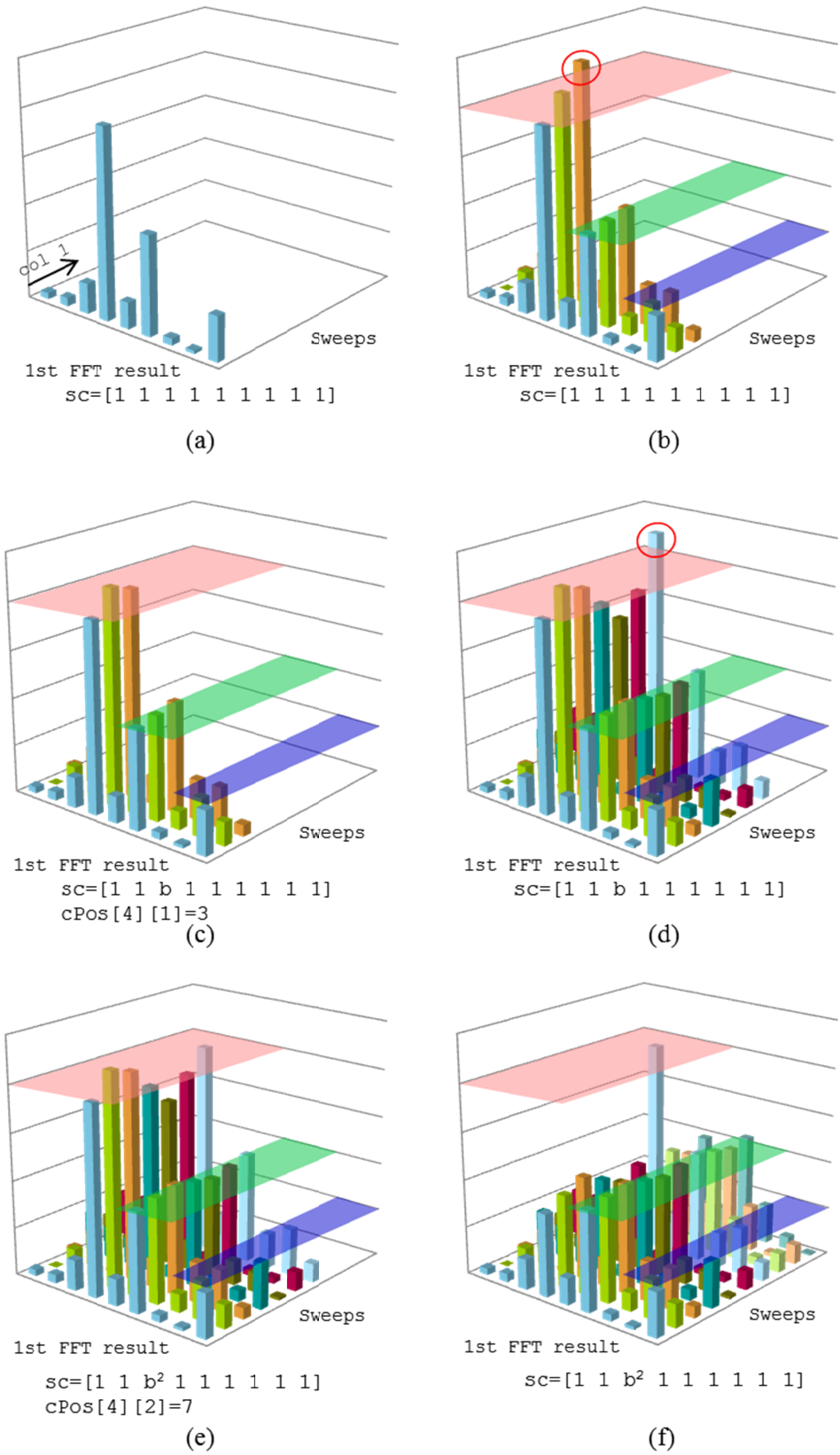


Figure 4-20 CRM in action (a) sc are initialized to all 1, 1st FFT result from the first sweep comes in. (b) In sweep 3, the 4th column has one data above the regional threshold. (c) sc[4] is set to b and the clipped data is scaled by b, cPos[4][1] is set to the sweep number – 3. (d) All new data in the column of 4 is scaled by b until at sweep 7, the data is still clipped even after the first scaling. (e) sc[4] is further increased to b^2 and the clipped data is scaled again, cPos[4][2] is set to 7 (f) All sweeps are received, according to cPos record, data[3-6][4] (4th column, from 3rd to 6th row) is further scaled by b, data[1-2][4] are further scaled by b^2.

Figure 4-20 shows a step by step example for this CRM algorithm. The data from 1st sweep and the position of 1st column are shown in Figure 4-20 (a). The three half-transparent planes are the thresholds derived from RDVLE scheme. In Figure 4-20 (b), one value (4th column) in the 3rd sweep has the risk to be clipped so sc[4] is set to b and that data is scaled down by b which is shown in Figure 4-20 (c) and the first slot in cPos[4] is set to the sweep number 3 representing for column 4 that the 3rd data is the first position that is scaled.

As the sweeps come in, all incoming data in column 4 is scaled by b, however, in Figure 4-20 (d) a new data is above the threshold even after scaling down by b. Then this data is further scaled down by b again in Figure 4-20 (e) and the sc[4] is set to b^2. cPos[4][2] is set to 7 meaning in sweep 7, the scaling factor is increased again. After all the 10 sweeps are received, cPos information is used to correct the previous unscaled data. Specifically, data[3-6][4] (4th column, from 3rd to 6th row) is further scaled by b, data[1-2][4] are further scaled by b^2 making every data in column 4 scaled by b^2 uniformly. Figure 4-20 (f) shows the final data of the 1st FFT result with every point under the regional threshold.

4.7 Hardware Considerations

Bearing in mind that this data compression scheme is part of a SoC, it should be designed to be optimal in terms of hardware implementation.

It's very straightforward to implement the RDVLE + UDRE which are cutting bits. The overhead is the space to store the received power-range model and the region boundaries. As described in previous chapters, one sweep contains N data points and it will generate N/2 useful complex data by the 1st N-point FFT. Assume the whole range is divided into k regions, so k-1 boundaries need to be stored. Each boundary needs at most $\lceil log_2 N/2 \rceil$ bits to be stored. For these k regions, k fixed point number format masks have to be stored. The simplest scheme is to store bit-mask patterns (assume 16-bit mask is used). Overheads are shown in Figure 4-21.

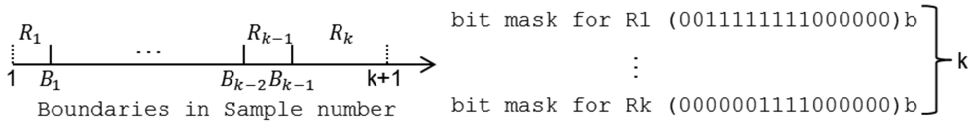


Figure 4-21 Overhead for the RDVLE + UDRE

In total, the space overhead is:

$$OH_{VLE} = (k-1)\lceil \log_2(N/2) \rceil + 16k \tag{63.}$$

Take $k = 11$, $N = 1024$ as an example, the total space overhead is 266 bits. Comparing to Equation (52.), it's only 0.012% of the original data size.

Equation (63.) gives the worst case estimation of the overhead which is already reasonably small. To further reduce the overhead, more sophisticated schemes can be used. For example the RDVLE bit-mask can be increased linearly to avoid the storage of bit-mask for every encoding region.

For implementing the CRM algorithm, the overhead is higher compared to the RDVLE but still at a reasonable level. Two additional sets of values have to be stored in the CRM. The first one is the set of scaling factors sc. One scaling factor is kept for each column, so the length of sc is N/2.

The size of each scaling factor sc depends on b. Actually, the choice of b not only influences the space overhead it also influences the computation complexity overhead which is directly related to hardware area. If b is chosen to be some random small integer or fractional number, an additional divider will be needed to perform scaling operation. A good choice of b is a power of two which reduces scaling operation to bit shifting operation and the scaling factor can be simply the power of 2, for example, sc[n]=1 means right shift by 1 bit which is scaling down by 2. Assume 3 bits are assigned to each sc, then sc can represent 8 different scaling factors which are 1 2 4 8 16 32 64 128 respectively. The number of bits to store all sc[N/2] is: 3N/2.

The second additional set of values is cPos[N/2][K] which has K elements for each column. For M sweeps, $\lceil log_2 M \rceil$ bits are needed at maximum to store the sweep number. So the number of bits to store cPos[N/2][K] is $\lceil log_2 M \rceil NK/2$. It's not predictable at which sweep the incoming data will be clipped and how many positions are needed to be recorded for one column. K is chosen so that it covers most clipping cases but not all cases. For example, if K is chosen to be 4, the maximum scaling factor is 16 which provides an extra of 12dB in max magnitude for each encoding region. The total space overhead for implementing the CRM algorithm is:

$$OH_{CRM} = \frac{3N}{2} + \frac{\lceil \log_2 M \rceil NK}{2} \tag{64.}$$

When $N = 1024$, $M = 128$ and $K = 4$, the total overhead from CRM algorithm is 15872 bits which is 0.76% of the original size. Combine the above two aspects, the total number is $266 + 15872 = 16138$ bits which is about 0.77% of the original size or 3.85% of the compressed size assuming a CR of 5 can be achieved. In summary, the total space overhead for implementing RDVLE + UDE + CRM is well within reasonable range and the computation overhead is very low if the scaling factor increment b is chosen properly.

4.8 Simulation and Results

All the simulations are done in Matlab, to assist explanation, various code snippets and block diagrams are shown in the following accompanying simulation results.

4.8.1 Modeling of the Signal Source and Processing Steps

In consideration of complexity and simulation speed, the simulation is carried out in baseband. The signal source that is modeled is the signal after deramping and LPF which is represented by Equation (31.). The only little difference is that the rectangular window is removed for simplicity. Removing the window means making the length of every section of deramped signal change from $T_c - \tau$ to T_c, the change is small and neglectable because τ is much smaller than T_c.

```
%generate deramping response.
sigDeramp = zeros(Sweeps,N);
%total sample number
totSNo = 0;
for sweep = 0:Sweeps-1
    for n= 0:N-1
        % update distance to account for moving reflector
        R   = R - vD*ts;
        % update time of flight to account for moving reflector
        tau = 2*R/c;
        sigDeramp(ramp_idx+1,n+1) = ...
              sum(...
                  10.^(Sig_pow/20).*...
                  cos( 2*pi*( fSlope.*tau*(totSNo*ts-...
                       ramp_idx*Tchirp)- fc*tau) )...
                  );
        %accumulate sample no.
        totSNo = totSNo + 1;
    end
end
```

The above Matlab snippet is used to generate the signal source. Sig_pow is derived from the received power-range model according to the distance of the target. After this, white noise is added to the generated signal and the total signal is normalized to the noise floor (scaling the total signal to make the noise floor at 0dB level) then quantized to the format of [1,16,0]. An example of a generated signal is shown in Figure 4-22.

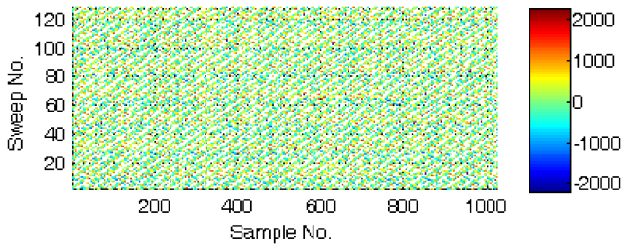


Figure 4-22 Example of generated deramped signal

The generated signal is fed into the first FFT block and the FFT is done in each sweep (row), then the FFT result is encoded using above-mentioned schemes. The encoded signal is stored in a matrix and the second FFT is done on each column after which the final result of the Range-Velocity spectrum is output.

4.8.2 Simulation Setup

Figure 4-23 shows the simulation setup. The upper part above the dashed line is the reference system where all operations and intermediate results are in floating point. The lower part is the system combined with compression schemes. The first FFT result is encoded in the 1st FFT Encoding block and the 1st RMSE is the root mean squared error introduced by this encoding. The 2nd FFT Encoding bock is not necessary for the system because the result is directly used by the following detection stages and it will not be stored in the memory again especially if the detection algorithm works in a region-based way. But to make the evaluation complete, the block is there. Both of the two blocks can be bypassed by the switches of Bypass1 and Bypass2. The total RMSE is the root mean square error between the reference system and the system with compression.

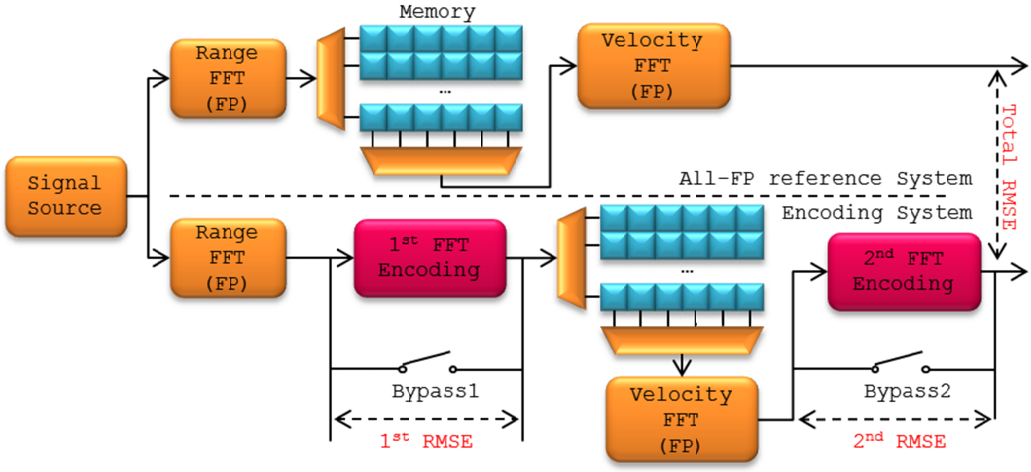


Figure 4-23 Simulation setup

One important thing to mention is that the ultimate criteria of this system should be the detection performance such as genuine detection rate and false detection rate. However, target detection is not the topic of this work. Instead, Root Mean Squared Error is used to characterize the data distortion in this compression scheme. RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(X_i - X_{iref}\right)^2} \qquad (65.)$$

where X_i is the encoded values and X_{iref} is the corresponding reference values.

4.8.3 Simulation Results

System parameters in simulations are listed in Table 2-1. Specific parameters such as RCS distribution parameters are shown in related results.

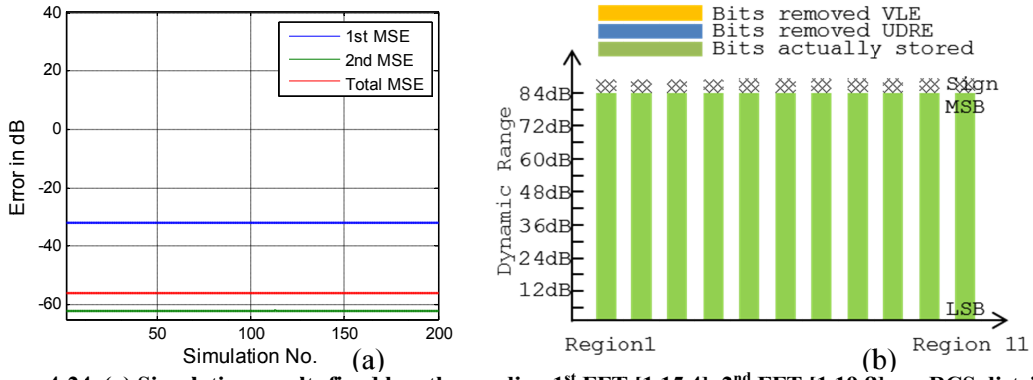


Figure 4-24 (a) Simulation result: fixed length encoding 1st FFT [1,15,4], 2nd FFT [1,19,8], no RCS distribution, fixed ranges and velocities. (b) Fixed length encoding visualization for 1st FFT

Figure 4-24 shows the simulation results using fixed length encoding i.e. all results are in the same number format. For the 1st FFT result, it is in the format of [1,15,4] and [1,19,8] for the 2nd FFT result. The data format is the same across all encoding regions which is shown in Figure 4-24 (b). In this simulation, 200 iterations are run, in each run, the ranges and distances of the targets are fixed values:

$$R = \begin{bmatrix} 10 & 20 & 25 & 35 & 45 & 50 & 60 & 70 \end{bmatrix} \text{m}$$
$$vD = \begin{bmatrix} -30 & 20 & -20 & 5 & -5 & 10 & 15 & 30 \end{bmatrix} / 3.6 \text{ m/s} \quad (66.)$$

In addition, RCS variation is not considered and no CRM is applied. The solid lines in Figure 4-24 (a) are various error values in 200 simulations and the dashed lines are their average values respectively. There are merely variations meaning that all values are well in range of the fixed length number format and the errors are due to the accuracy differences between floating point and fixed length format.

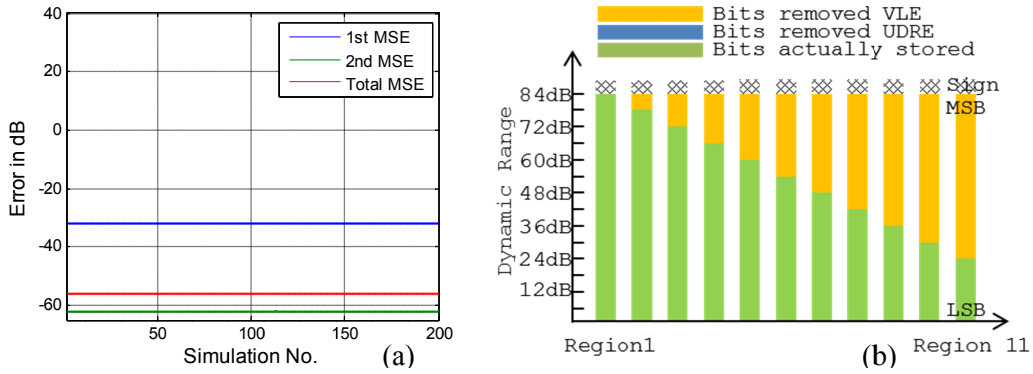


Figure 4-25 (a) Simulation result: VLE for both FFT results, no RCS distribution, statistical range and velocity generation for 5 targets (b) VLE visualization for 1st FFT

Figure 4-25 shows the simulation results with the VLE scheme. RCS variation is not considered, but instead of fixed ranges and velocities, randomly generated ranges from 0 to 70m and velocities from -30km/h to 30km/h for 5 targets are used. The number formats in different encoding regions are shown in Figure 4-25 (b) starting from [1,15,4] to [1,5,4]. Under this scheme, the compression ratio is fixed at:

$$CR_{VLE1} = 2.16 \quad (67.)$$

Errors shown in Figure 4-25 (a) are the same as those in Figure 4-24 (a). This is because the encoding regions are designed exactly according to the received power-range model which is also used to generate the signal itself, so every sample can fit into the number format of its region.

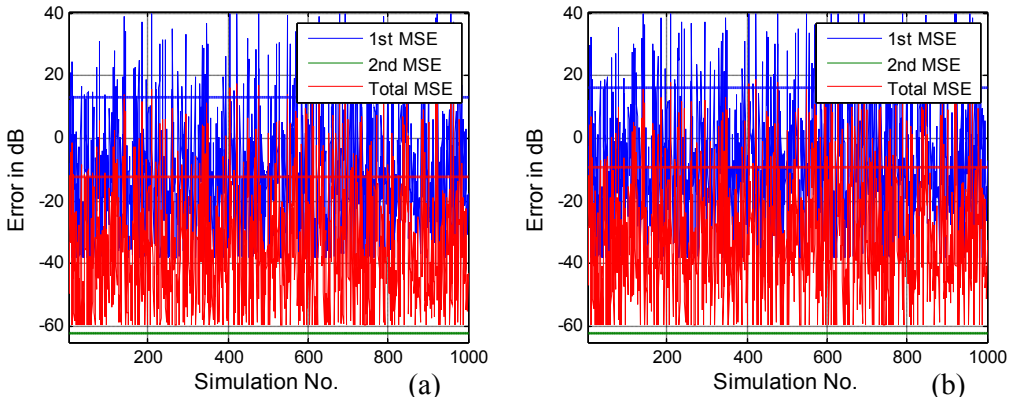


Figure 4-26 (a) Simulation result: VLE for both FFT results, fixed RCS distribution (Prius), statistical range and velocity generation for 5 targets, without CRM (b) Simulation result: conditions are the same as those in (a) but with CRM

Figure 4-26 shows the simulation results with the same VLE scheme shown in Figure 4-25 (b) with 5 randomly generated targets. However, RCS variation is considered this time, as shown in Figure 4-7, the measured Toyota Prius RCS which has a weibull distribution ($\alpha = 0.535$, $\beta = 11\text{dBsm}$) random variable is added to the generated signals. This randomness will result in some magnitude overflow cases as discussed previously. Figure 4-26 (a) shows the result without CRM, comparing to Figure 4-25 (a) where RCS is not considered, the 1st RMSE and the total RMSE are much higher, but the 2nd RMSE remains low because the second encoding stage provides a much longer format.

Figure 4-26 (b) is the result with CRM. The 1st RMSE and total RMSE are even worse (increased by approximately 3dB). This number is pessimistic because the result with CRM is compared directly to unscaled all-FP reference result. Though harmonics can be largely reduced, the target peaks are lower than the reference value and this difference is also counted in the RMSEs. To see the effect of CRM, one must look into individual measurements. There are many high peaks above -10dB in Figure 4-26 resulting in large RMSEs, one case of them is shown in Figure 4-27.

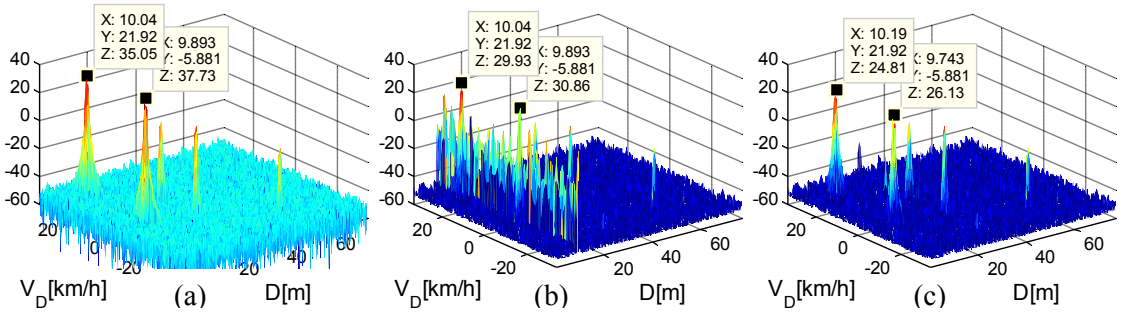


Figure 4-27 (a) All-FP reference s2D (b) s2D without CRM, total RMSE is -7.50dB (c) s2D with CRM total RMSE is -9.90dB (all three z-axes are in dB scale)

Due to RCS variation, the nearest two targets generate very large signals that are clipped in RDVLE without CRM in Figure 4-27 (b). When CRM is applied in Figure 4-27 (c), the s2D becomes more recognizable, but the amplitudes of the two peaks are significantly lower resulting in even higher RMSE. In terms of detection, the s2D with CRM is apparently better than the one without CRM.

It's better to view from another perspective to see the effects of CRM. Figure 4-28 shows SNR plots derived from the same Range-Velocity spectrum (s2D) determined in the same simulation as in Figure 4-26.

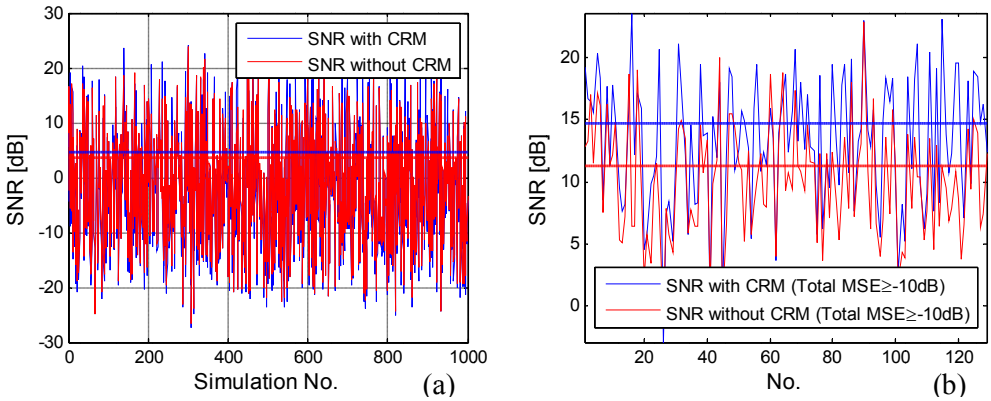


Figure 4-28 (a) SNR of s2D in all simulations (b) SNR of s2D with Total RMSE greater than or equal to -10dB

SNR of the 2D spectrum is **defined** as the ratio between total power of 14 × 14 samples around each target peak and the total power of the rest samples. Figure 4-28 (a) shows the SNR of s2D in all 1000 simulations and the average SNR with CRM is higher than that without CRM. Figure 4-28 (b) shows the SNR extracted from Figure 4-28 (a) for simulations of which the Total RMSE (shown in Figure 4-26) is greater than or equal to -10dB which is often a result from clipping. There are 129 cases out of 1000 simulations. The average SNR with CRM when total RMSE is greater than or equal to -10dB is much higher than that without CRM even with the fact that target peaks' power with CRM can be much lower than that without CRM. This result shows the meaning of CRM in RDVLE by providing better SNR for the following detection stages especially in high RMSE cases.

After applying CRM, UDRE is also added to the compression scheme and simulated. The results are shown below.

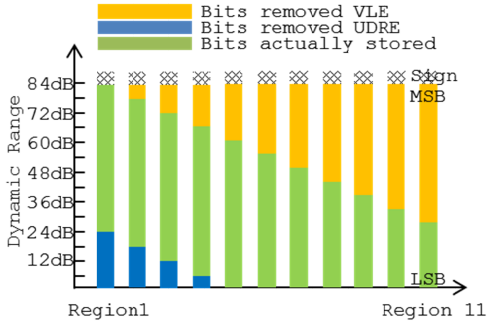


Figure 4-29 RDVLE + UDRE (10 bits) formats in different coding regions

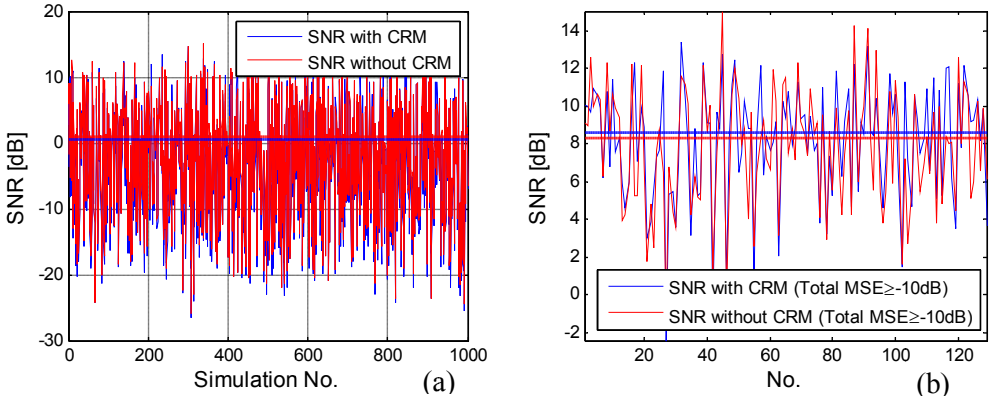


Figure 4-30 SNR: RDVLE + UDRE (10bits): (a) all simulations (b) RMSE greater than or equal to -10 dB

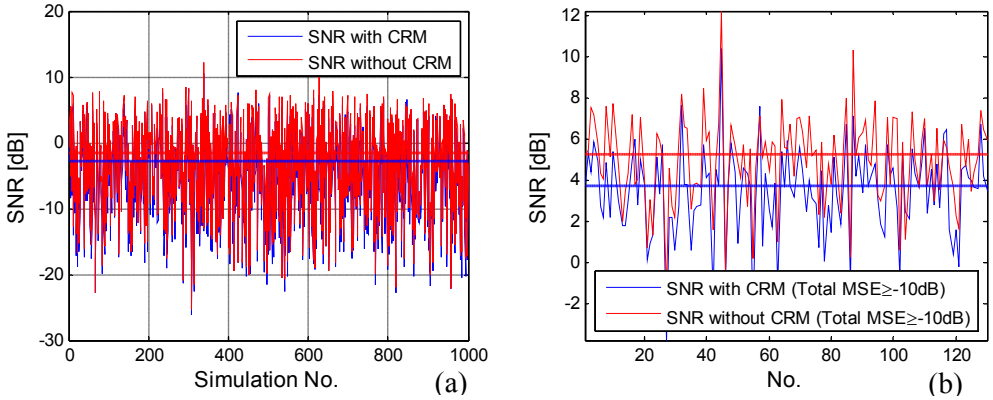


Figure 4-31 SNR: RDVLE + UDRE (8bits): (a) all simulations (b) RMSE greater than or equal to -10 dB

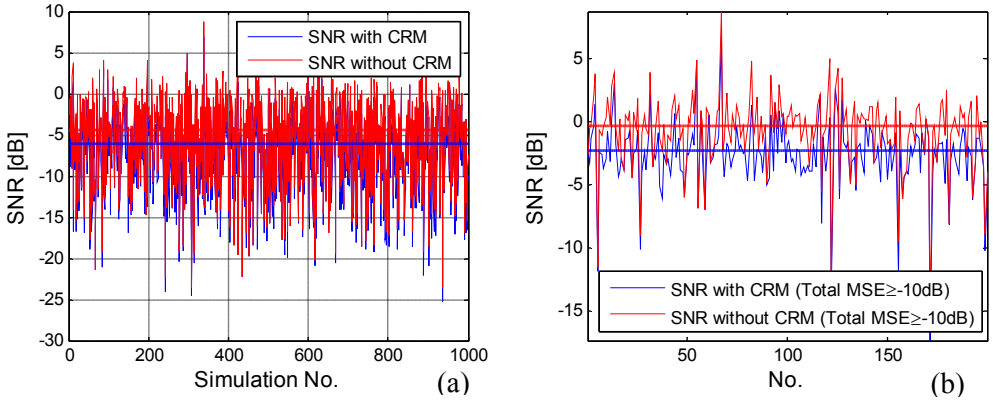


Figure 4-32 SNR: RDVLE + UDRE (6bits): (a) all simulations (b) RMSE greater than or equal to -10 dB

First of all, the formats in different encoding regions under RDVLE + 10-bit UDRE scheme are shown in Figure 4-29, formats for UDRE with different number of bits are similar. The SNR results are shown in Figure 4-30, Figure 4-31 and Figure 4-32. The CRs under these 3 different schemes are **2.27**, **2.43** and **2.81** respectively. The average SNRs of all simulations are dropping as expected with the reduction in number of bits in UDRE. The SNRs in simulations with Total RMSE greater than or equal to -10dB are also dropping with the reduction of bits in UDRE. In Figure 4-30 (b), the average SNR with CRM is higher than that without CRM but it's the opposite case in Figure 4-31 (b) and Figure 4-32(b). This is due the reduction of signal power caused by the CRM and the rise of the noise floor by UDRE. Though the UDRE can further increase CR, but the increase in CR is small with a quick drop in SNR. RDVLE + 10-bit UDRE + CRM is a practical choice but UDRE with less number of bits might not be suitable.

5 Doppler Redundancy Hybrid Encoding

In this chapter, we will discuss the design of the other compression scheme called the Doppler Redundancy Hybrid Encoding (DRHE) which is, in general, lossless. This scheme takes advantage of the redundancy in signals between consecutive sweeps i.e. the similarity of signals in one sweep after another to do compression. Thanks to the Run Length Encoding and the Huffman coding, the signal can be compressed in a lossless manner which preserves all the information in range and velocity. The original DRHE algorithm is adjusted into a column based algorithm to fit the sweep by sweep nature of the incoming signal. Due to the use of the Huffman coding, the total length of each column is unpredictable, so a memory management method is proposed to arrange all data into a fixed column size memory by dynamically allocating new memory space for overflowed column. Similar to previous chapter, at the end, the hardware implementation considerations and simulation results are presented.

5.1 Doppler Redundancy and Prediction Model

Redundancy in signals is the key in all kinds of compression algorithms. It can mean repeating structures, regular arrangements, similar values and so on.

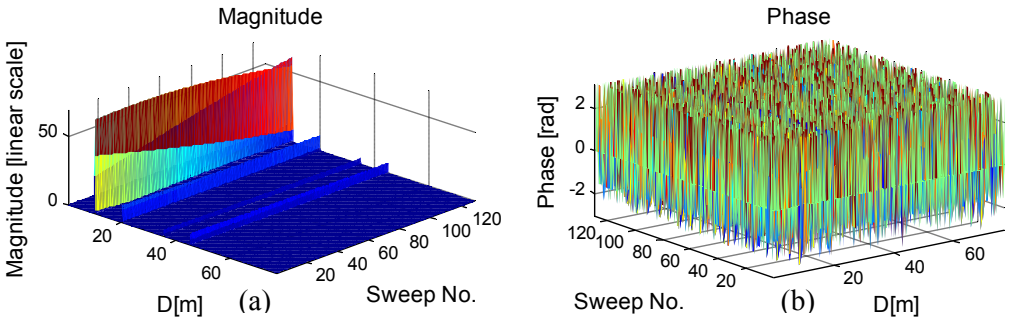


Figure 5-1 An example of magnitude and phase of sRS

Figure 5-1 (a) shows the magnitude of one example sRS signal. It's easy to have the observation that the changes in magnitude across sweeps are actually small though it's not constant. This is one example of the redundancy in the signal. We call this redundancy between different sweeps the Doppler (Spectrum) Redundancy because it exists in the direction of Doppler frequency shift discussed in Chapter 2.1.4. After identifying the redundancy, the natural next step is to try to predict the incoming sweep based on the existing data.

However, sRS, as the result of a FFT operation, is originally in Real/Imaginary format. One example is shown in Figure 5-2.

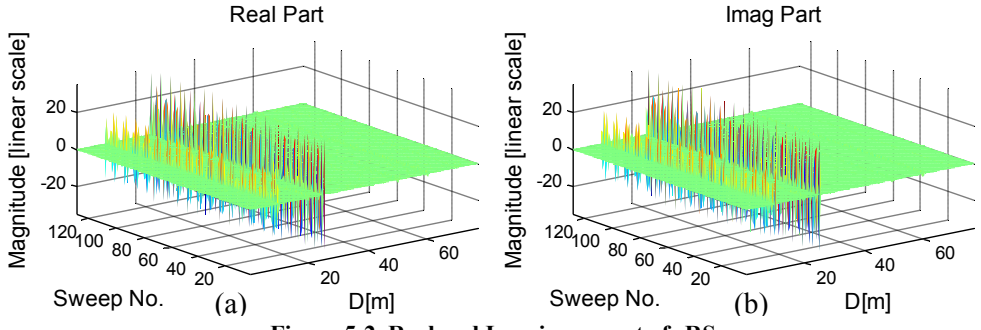


Figure 5-2 Real and Imaginary part of sRS

Unlike the magnitude of sRS, both real part and imaginary of sRS change drastically across the sweeps which makes it hard to do prediction. This problem also exists in Magnitude/Phase representation, but only in the phase part which exhibits even more unpredictability (see Figure 5-1 (b)). One good property of phase is that it is always within the range of $[-\pi, \pi]$. Even if the prediction does not work well for phase, storing it with certain accuracy will not be a big problem.

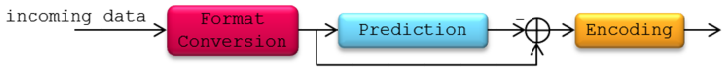


Figure 5-3 General structure of the DRHE compression scheme

Figure 5-3 shows the general structure of DRHE compression scheme. The incoming data is in Real/Imaginary format and after proper data format conversion, a prediction is made based on previous data and the difference between the prediction and the real current input data is computed and sent to be further encoded.

For each incoming complex 1st FFT result value, the prediction is in the following form:

$$\hat{X}[m,f]=\hat{a}[m,f]e^{j\hat{\theta}} \tag{68.}$$

where â and $\hat{\theta}$ are predicted magnitude and phase respectively. For magnitude:

$$\hat{a}[m,f]=\alpha\hat{a}[m-1,f]+(1-\alpha)a[m-1,f] \tag{69.}$$

The current predicted value â$[m,f]$ is a linear combination of previous predicted value â$[m-1,f]$ and the last incoming magnitude value a$[m-1,f]$. For phase prediction:

$$\begin{aligned}\hat{\theta}[m,f]&=\theta_{mdl}[m,f]+\theta_{incMdl}[m,f]\\ \theta_{mdl}[m,f]&=\theta[m-1,f]\\ \theta_{incMdl}[m,f]&=\beta\theta_{incMdl}[m-1,f]+(1-\beta)(\theta[m-1,f]-\theta_{mdl}[m-1,f])\end{aligned} \tag{70.}$$

The phase prediction $\hat{\theta}$ is the sum of the phase model value θ_{mdl} and the increment model θ_{incMdl}. The phase model is simply the last phase of the incoming data. The phase increment model is a linear combination of the last value of the increment model and the difference between the last incoming data and the last model value. Substitute θ_{mdl} and θ_{incMdl} into $\hat{\theta}$ and we get:

$$\begin{aligned}\hat{\theta}[m,f]&=\theta[m-1,f]+\beta\theta_{incMdl}[m-1,f]+(1-\beta)\theta[m-1,f]-(1-\beta)\theta_{mdl}[m-1,f]\\ &=\theta[m-1,f]+\beta(\theta_{incMdl}[m-1,f]+\theta_{mdl}[m-1,f])\\ &\quad+(1-\beta)\theta[m-1,f]-\theta[m-2,f]\\ &=\beta\hat{\theta}[m-1,f]+(2-\beta)\theta[m-1,f]-\theta[m-2,f]\end{aligned} \tag{71.}$$

From Equations (69.) and (71.), it's obvious that the predictions are two IIR filters where the incoming magnitude and phase are the inputs and the predictions are the outputs. Figure 5-4 shows the two filters in dataflow graphs.

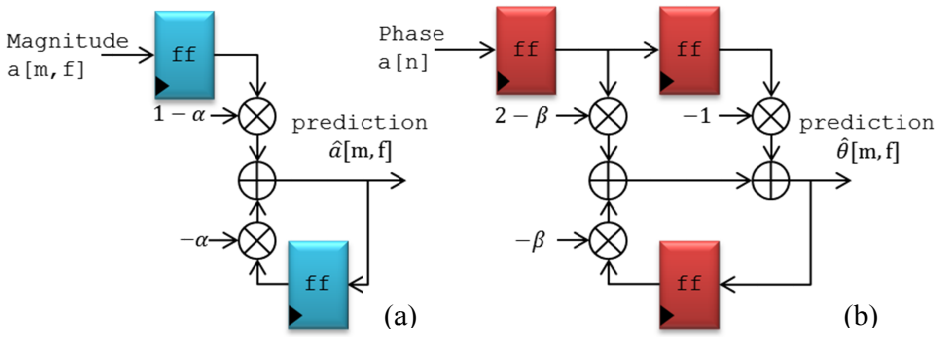


Figure 5-4 Predictions viewed as IIR filters: (a) Magnitude prediction (b) Phase prediction

The two predictions act like two IIR filters with tunable parameters, namely α and β. Seen from Equation (69.) and (70.), intuitively, the two parameters control how fast the predictions are influenced and modified by the real input signal.

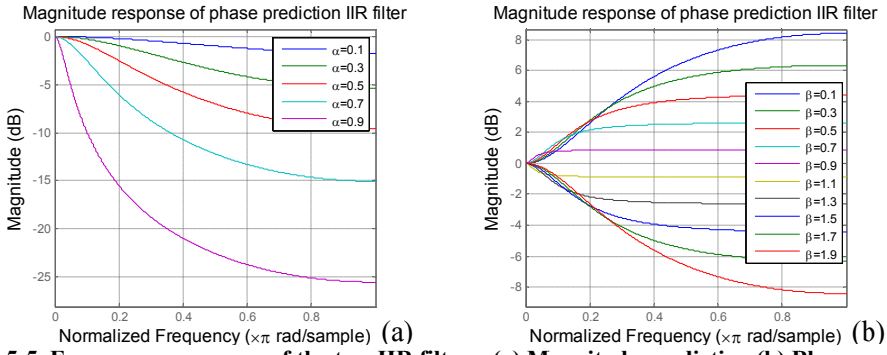


Figure 5-5 Frequency responses of the two IIR filters: (a) Magnitude prediction (b) Phase prediction

The magnitude prediction filter is a low pass filter (LPF) because the change in magnitude shown in Figure 5-1 (a) is rather slow and mild. The phase prediction filter can change from low pass to high pass (with gain) depending on β. According to Equation (39.), the phase at Sweep m is $2\pi f_0 (2r + 2mv_r T_c)/c$. Taking r as a constant between consecutive rows and a maximum speed of 9 m/s, the phase change in consecutive Sweeps is:

$$\Delta Phase = 2\pi f_0 \frac{2v_r T_c}{c} = 2\times 3.14\times 7.7\times 10^{10}\times 2\times 9\,\mathrm{m/s}\times 1.02\times 10^{-4} \div c = 2.97\,\mathrm{rad} \quad (72.)$$

which is a quite large change which can also be observed in Figure 5-1 (b). This observation leads to the choice of a HPF for the phase prediction filter.

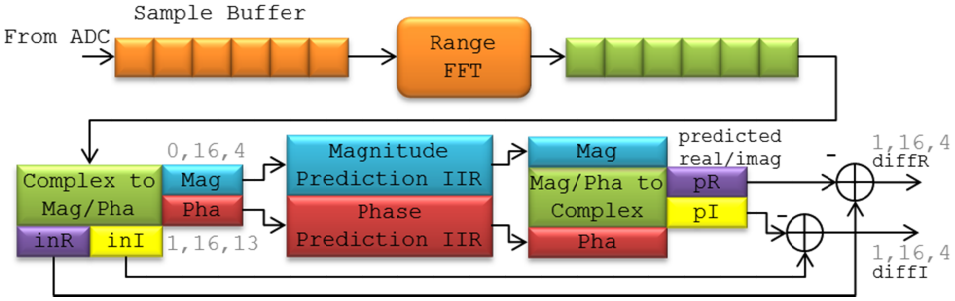


Figure 5-6 Prediction and difference stage

Figure 5-6 shows the details of the prediction stage with complex to Mag/Pha conversion. The prediction is done in mag/pha but the prediction result is converted back to complex representation and the difference between the prediction and the incoming data (diffR and diffI) will be further encoded.

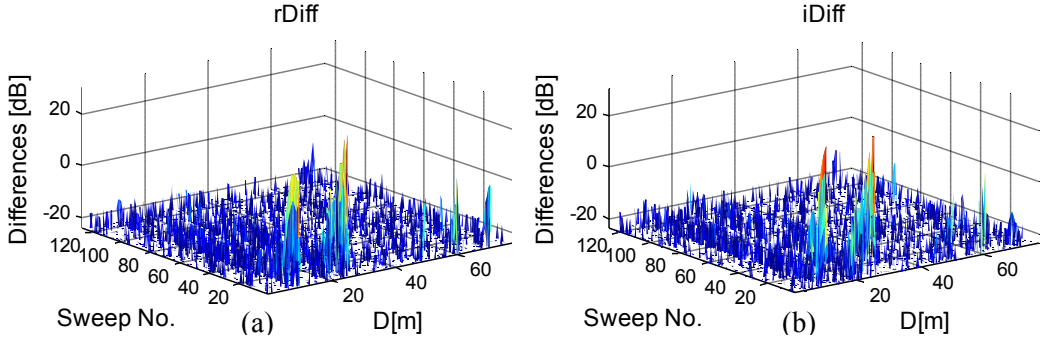


Figure 5-7 Differences between prediction and real input: (a) difference in real part (b) difference in imaginary part in both figure, $\alpha = 0.3\ \beta = 0.7$

Figure 5-7 shows an example of rDiff and iDiff with prediction parameters α = 0.3 and β = 0.7. For both the real and imaginary parts, the differences are large for the first several sweeps especially where there is an object in near range which has high peak. As the sweeps come in, the model is better adapted to the input signal, so the differences drop down to as low as -20dB and at some points, the difference is even 0.

The differences will be further encoded and stored. In decoding, the decoded differences can be losslessly restored by reverse operations of adding differences to prediction values. Last but not least, for hardware implementation, the filters have to be in fixed point format which will be briefly discussed in Chapter 5.8.

5.2 Choices of Encoding

After the prediction stage, the input signal is converted to differences which have to be encoded in a proper way. Design choices should be made based on the characteristics of the data to be encoded.

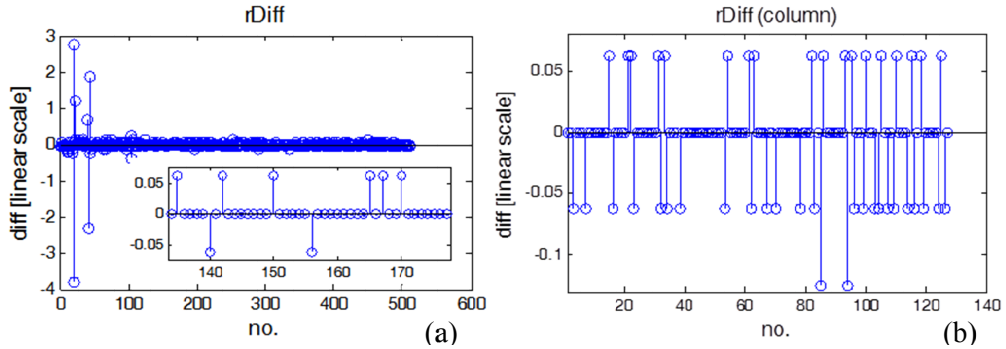


Figure 5-8 (a) One sweep of rDiff (b) One column of rDiff

Figure 5-8 shows one sweep of the rDiff and one column of rDiff, the small graph shows the data details around no.130-180. There are two obvious characteristics in rDiff. First, the difference is generally small but there are some rare high peaks. The peaks appear due to the inability of the prediction filters to predict the exact value of the incoming signal when there is an object, especially at near range with high received power. Second, there are many segments that are filled with consecutive zeroes.

For the first characteristic, entropy coding such as the Huffman coding and the arithmetic coding might be suitable. They will generate shorter codes for frequent symbols and long codes for rare symbols which will optimize the overall storage space. For the second characteristic, run length encoding or dictionary based encoding might be suitable which are capable of encoding long repeating symbols.

5.3 Run Length Encoding

RLE is used mainly for its simplicity. RLE is a lossless compression which is widely used in all kinds of image compressions. It will run through the input data and encode it as single values followed by its counts. It's very useful and effective for simple data and patterns especially those with repeating symbols.

AAAABAABBBBCCCCCCDDDDDDDDEEEAE A4BA2B4C6D8E3AE (73.)

The above example gives a basic idea of how RLE works. The beginning four As are encoded as A4 and the rest of the string are encoded in the same way yielding the right side result which is apparently shorter than the original string. The decoding is straightforward – read in the data in the compressed stream one after another and repeat it the following number times.

From Figure 5-8 we can see that there are many repeating zeroes but other values do not appear so frequent in a consecutive way which is to say RLE might be efficient for encoding zeroes in the rDiff and iDiff but not for other values. This may limit the usage of RLE and for other values (symbols), other encoding has to be used.

5.4 Huffman Encoding and Code Table Overhead

As discussed in Chapter 5.2, for other symbols, the Huffman or the arithmetic encoding might be suitable. In terms of implementation, arithmetic encoding is much more complex than Huffman encoding but the gain in compression ratio is limited, so the Huffman coding is the first choice.

The basic principle of the Huffman coding is explained in Chapter 1.2.1. Besides all of its advantages, the Huffman coding will cause some overhead: one of which is the code table which stores pairwise mapping from symbol (input data) to code (compression output). Each data point in rDiff and iDiff is a 16-bit fixed point number which is to say there is $2^{16} = 65536$ different symbols. If we choose to store the whole pairwise table, it will have 65536 entries. Assume the CR can reach 5, the total size of the code table is:

$$65536*16/5/8=26214\text{bytes} \quad (74.)$$

which is already 10% of the original total size of rDiff and iDiff. For better compression performance, separate code tables for rDiff and iDiff may be needed which will double the storage overhead. Another option is to store the probability table which still has 65536 entries while extra computation is needed to build the Huffman tree. The overhead for using 16-bit symbol is too high and is not an affordable solution.

Since 16-bit symbol is too long, it can be divided into 2 8-bit symbols. For 8-bit symbols, the number of code table entries is greatly reduced to only 256.

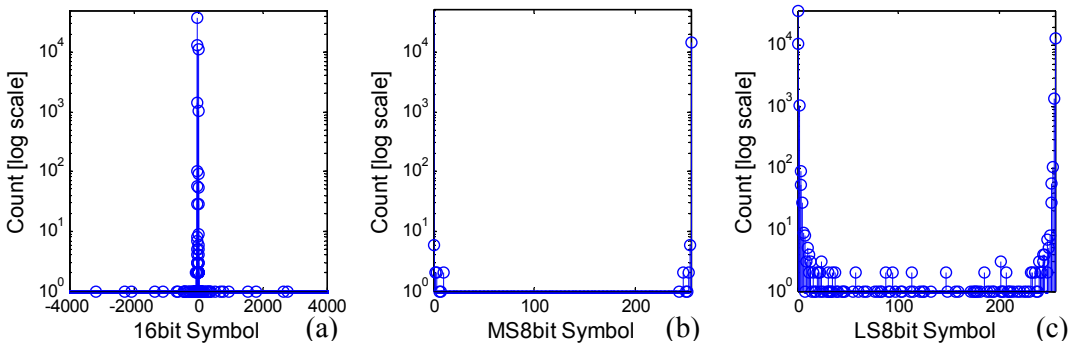


Figure 5-9 Symbol histograms: (a) original 16-bit symbol (b) least significant 8 bits from 16-bit symbol (c) most significant 8 bits from 16-bit symbol

Table 5-1 Entropy of example signal with different symbols

Symbol	Entropy	Theoretical CR
16-bit	1.647bits	9.715
8-bit	1.323 bits	6.047
LS8-bit	1.646 bits	4.860
MS8-bit	0.773 bits	10.349

Figure 5-9 shows three histograms, (a) is the histogram of the original 16-bit symbol, the symbols themselves are the data in rDiff. As described above, 16-bit signed symbols are from -32768 to 32767. In the above example, no data falls below -4000 or above 4000, so those parts are omitted on x-axis in (a). Figure 5-9 (a) is consistent with our observations in Chapter 5.2 that most difference data is small and around 0. The entropy of 16-bit symbols of the example signal is 1.647 bits as listed in Table 5-1 which leads to a theoretical compression ratio of 9.715.

By separating a 16-bit symbol in the middle into two unsigned 8-bit symbols, we have the MS8b (most significant 8 bits) data and the LS8b data. Their histograms are shown in Figure 5-9 (b) and (c). In general, most data concentrates at the two ends on x-axis. This is due to the data type conversion. Data is stored in 2's complement format in 16-bit symbol, when converted to 2 8-bit unsigned symbols, small positive numbers will be around 0 and negative numbers with small absolute values will be on the other end.

However, despite similar trend, the MS8b symbol histogram is quite different from the LS8b symbol histogram which is also reflected in the entropy items listed in Table 5-1. The LS8b has much more information in it than the MS8b because the prediction works well and the difference has very little chance to be large enough to fill the most significant 8 bits. If we encode these two parts of data using the same Huffman code table, the theoretical CR is 6.047 which is not as good as the average CR of $(4.860 + 10.439)/2 = 7.650$ when using an individual table for each symbol set.

Though it may help to increase the compression ratio, using individual code table for LS8b and MS8b adds more overhead and the CR still cannot compete with that using 16-bit symbol. So separating a 16-bit symbol into two 8-bit symbols might not be a good option. Another way of using the Huffman coding with reasonable overhead will be presented in the following chapters.

5.5 Column Based DRHE Encoding

The problem in Huffman coding is discussed above and it has to be used in another way to fully utilize its capability. The signal characteristics discussed in Chapter 5.2 are similar to that of the quantized coefficients in the JPEG encoding algorithm shown in Figure 5-10.

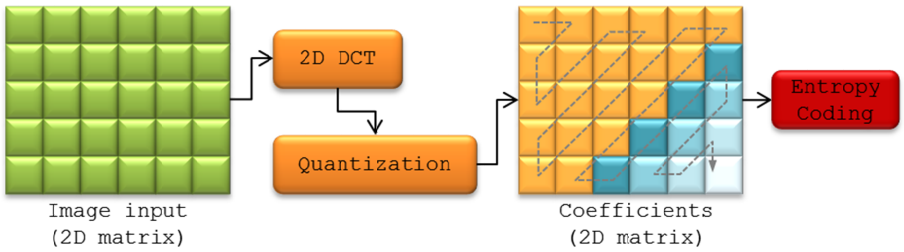


Figure 5-10 JPEG encoding algorithm [24]

In the JPEG encoding, an image is treated as a 2D matrix. A 2D Discrete Cosine Transform (2D DCT) is performed on this matrix generating a coefficient matrix with the same size. JPEG is a lossy algorithm, the loss of information happens in the quantization of coefficients. Those coefficients are quantized with different step sizes. The right-bottom coefficients represent high frequency image details which are not perceivable by human eyes. These coefficients are quantized with larger steps often leading to 0s in their places which are colored with gradually changing blue in Figure 5-10. Finally, the quantized coefficients are encoded by the Huffman coding in the sequence shown by the gray dash line in Figure 5-10.

The similarities between rDiff/iDiff and JPEG quantized coefficients make it possible to learn from the Huffman coding used in JPEG algorithm. However, the JPEG algorithm takes the whole image as a 2D matrix while our FMCW radar signal comes in sweep by sweep. Another important difference is that the second FFT is done in column direction therefore it's better to encode difference data in column direction and put encoded data in the same column together which makes memory addressing easier in decoding. The concept is shown in Figure 5-11.

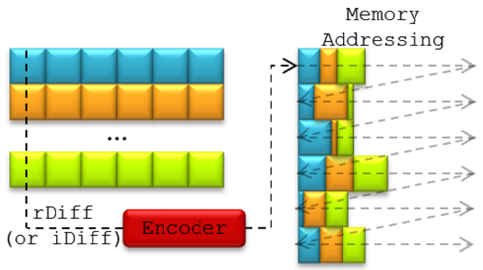


Figure 5-11 Column based compression and easy memory addressing for decoding

With all the considerations for this particular FMCW radar system and its signal characteristics, the following column based DRHE is designed. Take [1 0 3 0 0 0 0 253] as an example of the difference output (rDiff or iDiff) from the prediction block. Each number is a 16-bit fixed point value in this input data stream will be encoded into a binary form of [RRRRSSSS+APPEND] where RRRRSSSS is an 8-bit field and APPEND is an SSSS-bit field (for example, is SSSS is 0101, then APPEND is 5-bit long). For convenience, the name R4S4 might be used instead of RRRRSSSS in the following text.

The 4-bit RRRR field in R4S4 is the number of zeroes preceding a non-zero data while SSSS is the region that the non-zero data is in and is also the number of bits that APPEND needs to encode the exact data value. Table 5-2 gives the SSSS values and corresponding ranges.

Table 5-2 SSSS regions table

SSSS	Value range	Number of possible values
1	-1,1	2
2	-3,-2,2,3	4
3	[-7,-4], [4,7]	8
4	[-15,-8], [8,15]	16
5	[-31,-16], [16 31]	32
…	$-[2^n-1,2^{n-1}],[2^{n-1},2^n-1]$	2^{SSSS}
15	[-32767, -16384], [16384, 32767]	32768

The value ranges are arranged in an 'expanding' way where the higher level is always at the outer range in number axis shown in Figure 5-12 (a). Before generating APPEND bits, the input data is shifted. For negative numbers, the data is added with $2^{SSSS}-1$ and for non-negative numbers, 2^{SSSS} is subtracted from the data.

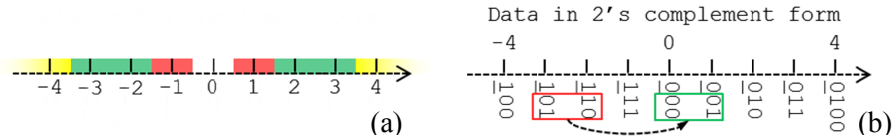


Figure 5-12 (a) Expanding ranges (b) Data shifting for generating APPEND bits, underscores note sign bits

Take 2, 3, -2, -3 as example. They are all in the SSSS=2 region meaning only 2 bits are needed in APPEND bits. Their APPEND bits are [10], [11], [01], [00] respectively. To achieve this, -2 and -3 are first shifted to 1 and 0 respectively (see Figure 5-12) and after that binary encoded. This shifting guarantees that all negative data has binary strings starting with 0 and all non-negative data has binary strings starting with 1 in the APPEND field. The previous example is then encoded into the following binary string shown in Figure 5-13.

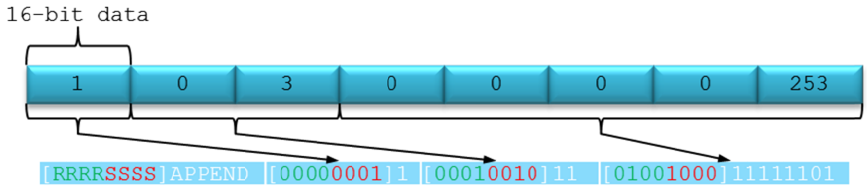


Figure 5-13 The R4S4+APPEND encoding for example data

Eight 16-bit values are encoded into 3 groups of R4S4+APPEND bits. A little problem for RRRR field is that there might be cases of more than 15 consecutive zeroes. To solve this problem, the symbol R4S4 of [11110000] is used to represent 15 consecutive zeroes followed by a 0 which is in total 16 zeroes and this symbol can appear multiple times if more than 16 consecutive zeroes need to be encoded. The length of the APPEND field bits is variable but the R4S4 field are fixed (8 bits) where Huffman coding is well suitable.

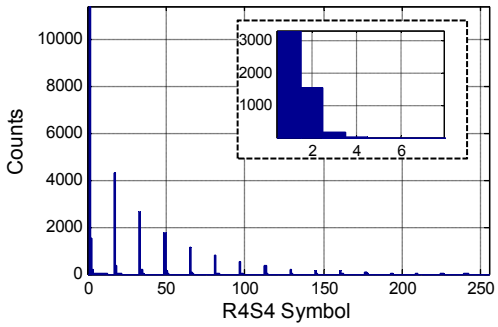


Figure 5-14 R4S4 symbol histogram

Figure 5-14 shows the R4S4 symbol histogram example. The general trend of the histogram is like an exponential decay in counts as R4S4 increases and for each peak value, there is a small micro decay structure shown in the small figure inside Figure 5-14. The local peaks are 1, 17, 33, 49, 65, 81, 97, 113, 129, 145, 161, 177, 193, 209, 225 and 240.

Table 5-3 shows the first 4 local peaks and their meaning. The distribution of the symbols shows very meaningful information of the signal: the number of occurrences decreases as the number of consecutive zeroes (RRRR) increases and the largest amount of data is small (in SSSS region 1). With this probability information, the Huffman coding may have good performance.

Table 5-3 Some local peaks in R4S4 symbol histogram and their meanings

Local Peak (Decimal, Binary)	Meaning
1, 00000001	No 0 preceding, value in SSSS region 1 (-1 or 1)
17, 00010001	One 0 preceding, value in SSSS region 1 (-1 or 1)
33, 00100001	Two 0s preceding, value in SSSS region 1 (-1 or 1)
49, 00110001	Three 0s preceding, value in SSSS region 1 (-1 or 1)
65, 01000001	Four 0s preceding, value in SSSS region 1 (-1 or 1)

Once we have the distribution of the symbols, we can generate the Huffman code table. In this design, we use a fixed Huffman code table from the example signal we have modeled. The code table proves to be suitable for all kinds of circumstances in the following simulations and for real recorded signals. The Huffman code table we generated is listed in Appendix A.

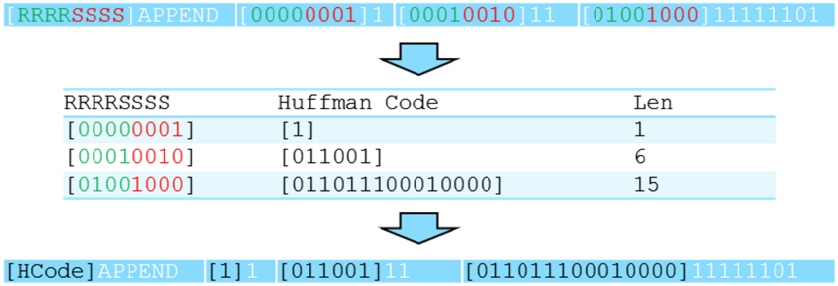


Figure 5-15 The Huffman coding of R4S4 field

Figure 5-15 shows Huffman coding of the R4S4+APPEND data from the example input. The final result is a 33 bits string representing the 128 bits original data. When there is more input data, the compression ratio will further increase.

With the above description, the DRHE itself is clear but there is still one element missing – how to perform the encoding column-wise when the signal comes in sweep by sweep? This problem is solved by the introduction of an addition mechanism called zCounter (zCNT). The name literally explains its usage: counting the number of zeroes. One zCounter is kept for each column and once the corresponding position in the current sweep is a zero, the zCNT is increased by 1 until there is non-zero data. Once there is non-zero data after some consecutive zeroes in one column or the zCNT reaches 16, the zCNT will be written into the RRRR part and the non-zero data is encoded by SSSS+APPEND bits with Huffman coding.

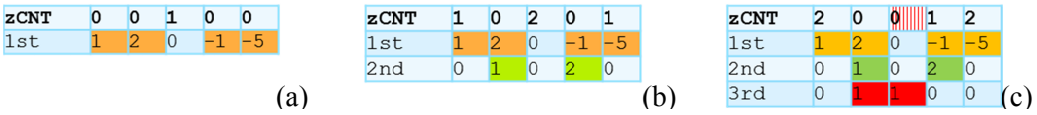


Figure 5-16 zCounter in action

Figure 5-16 shows the zCounter mechanism working in action. Figure 5-16 (a) shows the status after the first sweep of data. The 3rd column has one 0, and the zCNT[3] is set to 1, other data shaded in orange is encoded. Figure 5-16 (b) shows the incoming of the second sweep of data, zCounters continue to accumulate. In Figure 5-16 (c) the first non-zero data comes at column 3, zCNT[3] is reinitialized to 0 and 2 is written into RRRR field with corresponding SSSS and APPEND bits for the incoming data 1.

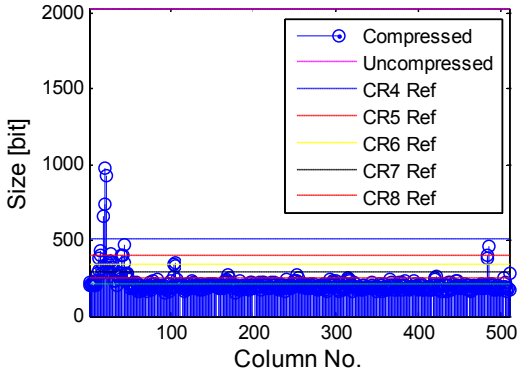


Figure 5-17 Compressed size of each column (CRx=Compression Ration of x)

Figure 5-17 shows the compressed size for each column using the example data. The overall compression ratio is 9.54. For the column representing the range at which there is a target, the compressed length is longer than where there is not, due to the fact that it contains more information. This result gives a first impression that column based DRHE encoding results in higher compression ratios than 16-bit or 2 8-bit Huffman coding.

5.6 Column Based DRHE Decoding

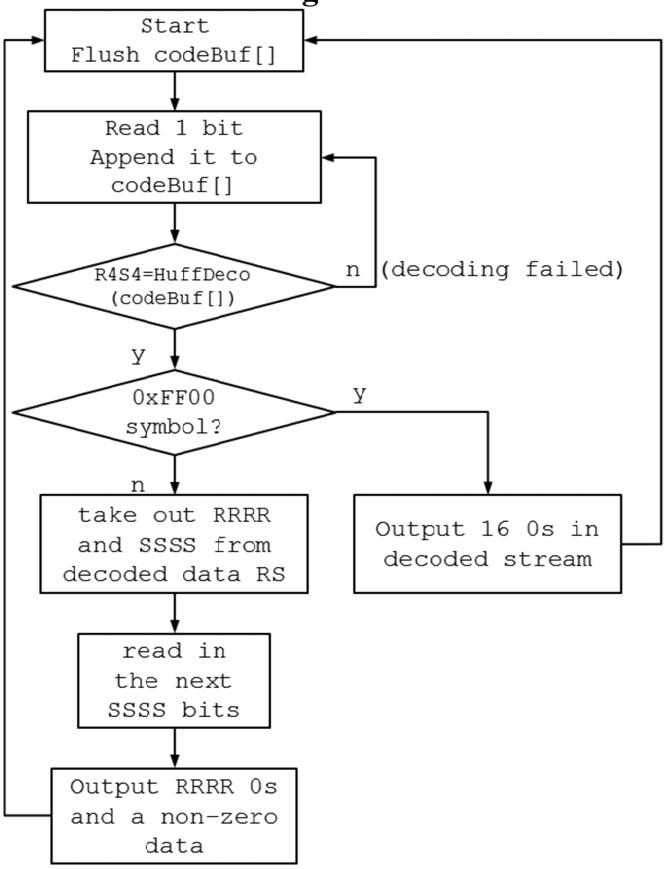


Figure 5-18 Column based DRHE decoding flow graph

Comparing to encoding, column based DRHE decoding is much easier because the Huffman is a prefix code (see the introduction chapters) and all prefix codes can be decoded uniquely without any ambiguity. Once the Huffman coded R4S4 is decoded, known number of APPEND bits can be read from the compressed bit stream and reconstruct the exact original data.

Figure 5-18 shows the flow chart of the decoding algorithm which is quite self-explanatory. The HuffDeco() function is used to decode the input using the Huffman code table. codeBuf[] is used to temporarily store the input bit stream. The encoding – decoding combination ensures

lossless compression of the difference data output by prediction block. The prediction process is also fully reversible which means the original input signal can be reconstructed losslessly.

5.7 Memory Management in Encoding Process

To use this column based DRHE in a real hardware setup, another problem has to be dealt with: the memory management. Seen from Figure 5-11 and Figure 5-17, the length of each compressed column is not even close. Some columns can be significantly longer than others. Figure 5-17 is just a benign case scenario. A very unfavorable bad case scenario is when there are two targets at the same range (probably with different velocities).

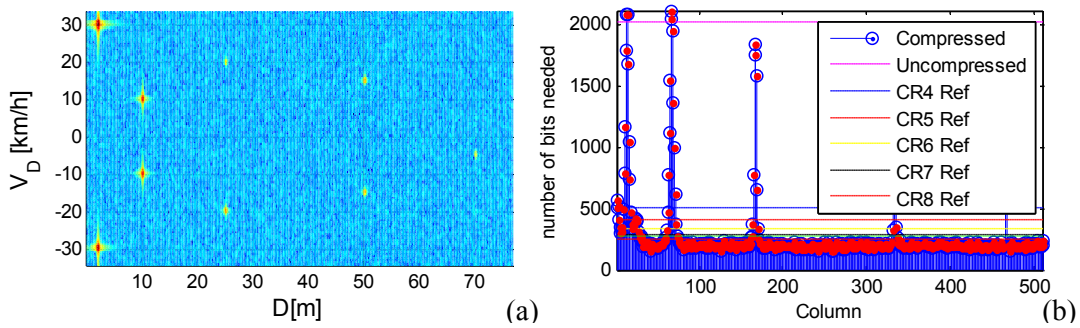


Figure 5-19 (a) Bad case target distribution (b) Compressed column length

Figure 5-19 shows one example of many bad case scenarios. There are two targets at each of the following ranges: [2, 10, 25, 50, 70] m. Figure 5-19 (a) shows the all-FP reference range-velocity spectrum where all targets can be seen as clear peaks in it. Figure 5-19 (b) is the compressed size of each column. It's very clear that wherever there are targets, the corresponding column size is much larger than the average size. For the 6 nearest targets, the compressed sizes are even larger than the original sizes.

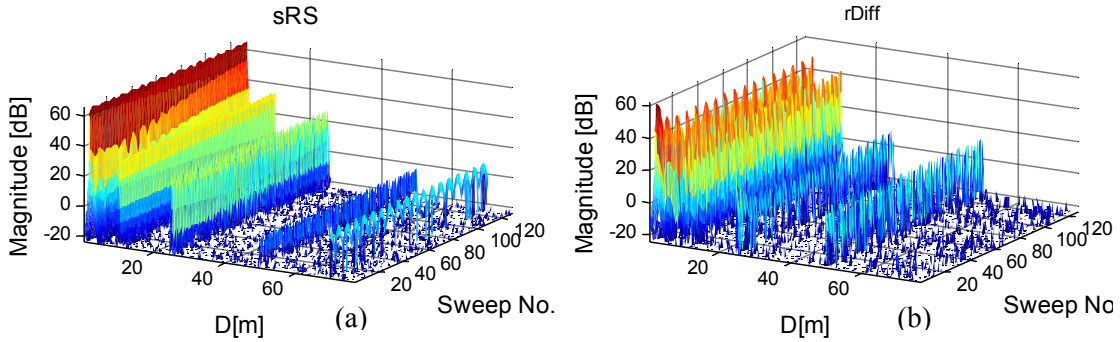


Figure 5-20 (a) Magnitude of sRS in bad case scenario (b) rDiff in bad case scenario

The fundamental cause of this problem is that when there are more than 1 targets at a range, the magnitude of sRS will start to change drastically among sweeps. The sRS signal of the above bad case is shown in Figure 5-20 (a), comparing to the benign case shown in Figure 5-1 (a), the change is quite significant. The prediction block cannot predict the signal with high probability so the difference signal is also large as shown in Figure 5-20 (b). Large values are rare according to R4S4 histogram shown in Figure 5-14, so large values are assigned to Huffman codes that are longer than average causing that the length of the corresponding column to be far beyond average.

This exceptional behavior will reduce the overall compression ratio to around 7 but this CR is usable and still much better than that achieved by the RDVLE. The serious influence is on memory configuration. The unknown length for each column makes the memory space allocation hard. For a fixed number of columns, if we use a large column length, the total memory is large and CR is sacrificed with a lot of memory unused and wasted. If a smaller column length is used, the CR will be good but there will be chances that certain columns cannot fit into the limited length column and have data losses. So a memory management function is needed to make the best use of memory and fit all the data in an organized way at the same time.

The target of this memory management function is to place compressed data of each column in a way that data in the same column is put close to each other in a block of memory with fixed column length, and a fixed number of rows of memory (thus the total size of memory is fixed).

The general idea of this algorithm is to reserve some memory space in the beginning and reallocate it dynamically when there is not enough space in some column to store the incoming compressed data stream. Before introducing the memory management algorithm, a data structure called emMap (extended memory map) has to be shown which is used to record the information of columns with insufficient space and newly allocated spaces to assist memory management to achieve its goal.

Table 5-4 emMap

orgCol	4	5	3	21	…
startPos	513	523	531	539	…
len	10	8	8	8	…
curPos	521	530	537	546	…

Table 5-4 shows the emMap with some example values. There are in total 4 rows in it. orgCol is the original column number where there is not enough free space to store the incoming compressed data stream. startPos is the starting column position to store the extra data that cannot be stored in orgCol. The extra space is in total len columns. curPos is the current column writing position in the newly allocated extra memory space. Besides emMap, there is an array of memory pointer called memPointer (one for each column including extra columns) pointing to the current writing position in the columns.

According to simulations, the bad case CR is around 6.5 if there isn't any constraint on memory shape (column can have arbitrary length). Instead of trying to obtain this bad case CR, we desire even lower CR, for example, target at $CR_{tar} = 5$. Another CR called CR threshold is set to $CR_{th} = 7$. Instead of allocating 512 columns of 406 bits memory to achieve CR5, we allocate 717 columns of 290 bits memory (which is still CR5) and reserve 205 columns for dynamic memory allocation. The memory reshaping concept is shown in Figure 5-12.

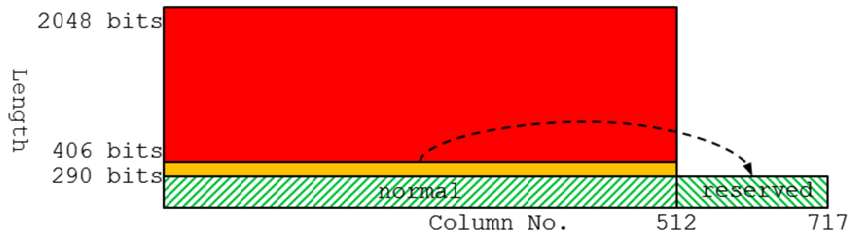


Figure 5-21 Memory reshaping

The red, yellow and left-inclined filled regions (normal memory) in total represent the uncompressed data which are 512 columns of 2048 bits data stream. The yellow and normal memory in total is the size to achieve a CR of 5. To reserve extra space for overflowed columns, the yellow region is reserved and is appended to the end of the normal memory region (the area of yellow region and right-inclined filled region are the same). So the memory actually has 717 columns of 290 bits space resulting in a fixed CR of 5.

Figure 5-22 shows the flow chart of the memory management algorithm for one sweep and this flow chart will be repeated for every incoming sweep. We use the numbers in the above example. numCol is 717, the maximum length for each column is 290 bits. fullCol is an array storing all original column numbers where all normal memory and newly allocated reserved memory is full.

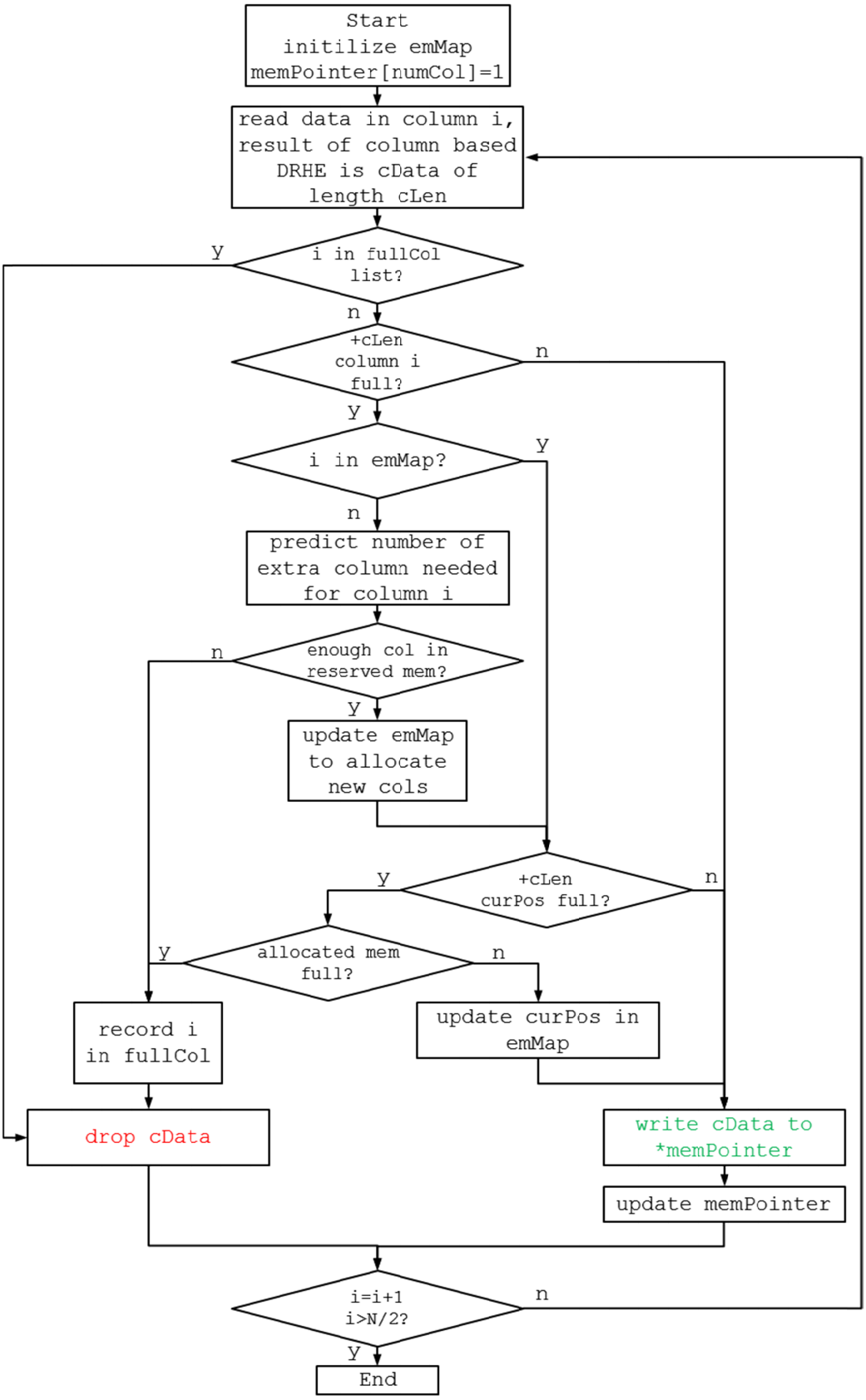


Figure 5-22 Flow chart of memory management algorithm for one sweep

In the middle of the chart, prior to updating emMap, a prediction is done. This step predicts the number of columns that are still needed in the reserved memory to store the rest of the incoming data when a certain column in normal memory is full. It is a simple linear prediction:

$$N_{rest} = \left\lceil \frac{curLen \cdot \left(\frac{M}{curS} - 1 \right)}{colBits} \right\rceil \tag{75.}$$

where N_{rest} is the number of columns that is needed for the rest of data in that column, curLen is the current length in bits of that column, M is the total number of sweeps, curS is the current sweep number, colBits is the column size in bits. The accuracy of this prediction depends on the time it happens. It's easier to understand that the prediction accuracy is higher when column overflow happens in the later stage of one measurement because the accumulated incoming encoded data with various lengths have an averaging effect on the prediction thus making it more realistic. Figure 5-22 shows the prediction error of each column vs. sweep no. (which also represents time).

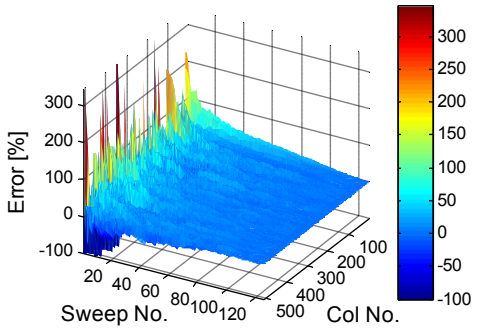


Figure 5-23 Final length prediction error

As more sweeps come in, the prediction error drops to an acceptable level. With large prediction error, not enough or more than necessary reserved memory might be allocated to overflowed columns resulting in data loss or memory waste.

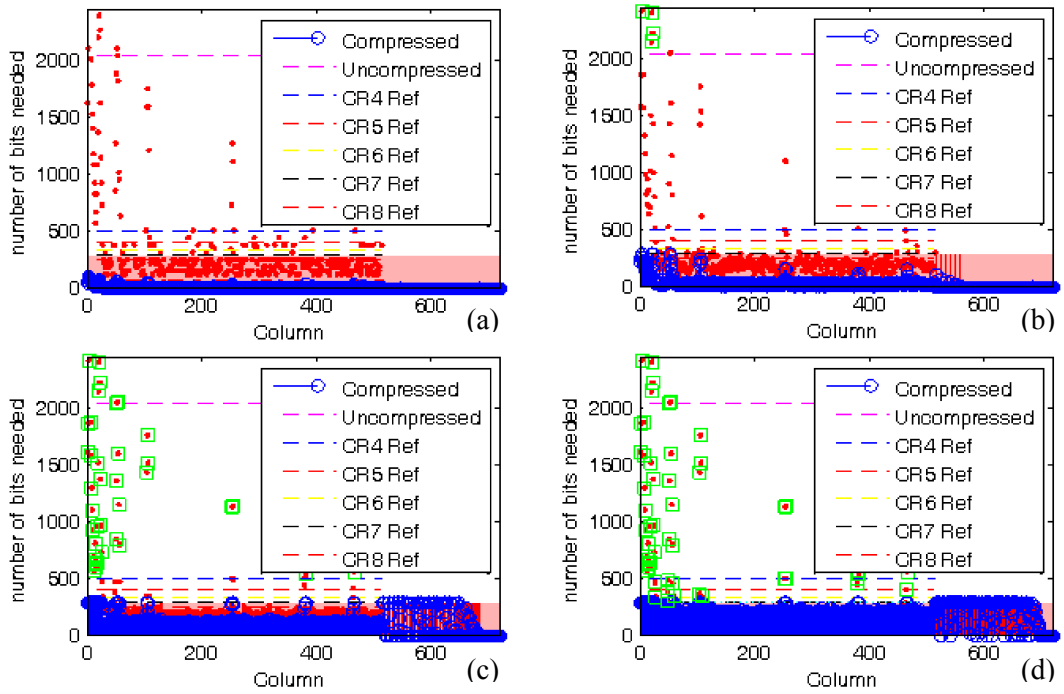


Figure 5-24 Memory management in action with bad case input data: (a) First several sweeps, no overflow (b) Some columns starts to overflow, new columns are allocated in reserved memory based on prediction (c) More columns overflow, more reserved memory are used. (d) All sweeps have come in, all data is fit into the fixed size memory

Figure 5-24 shows the memory management algorithm working in action in four graphs. The red-shaded area is the normal memory and reserved memory; the red dots are the column total length predictions at figure-capture time; green boxes are the predictions that are fixed and used

to allocate new columns in the reserved memory. The red vertical lines in the reserved memory are the boundaries of newly allocated memory for different orgCols.

In Figure 5-24 (a), data begins to come in, no overflow happens due to the small amount of data. In Figure 5-24 (b) as sweeps come in, some columns overflow very early because it's the bad case scenario. Extra columns are allocated based on predictions at this time shown in green boxes. As more sweeps come in, more and more columns overflow in Figure 5-24 (c) therefore more reserved space is used. Figure 5-24 (d) is the final situation with all compressed data fit into the fixed size memory.

Decoding under this memory management scheme is very similar to that without memory management discussed in Chapter 5.6. The only extra step is to use emMap to read the data in the reserved memory space for overflowed columns. Take the emMap shown in Table 5-4 as an example. The first column in it is orgCol:4, startPos:513, len:10, curPos:521. When decoding the 4^{th} column of memory, after reading all the data in 4^{th} column, it will read data from column 513 until the end of column 521 to finish decoding all the data that is originally in column 4 in the sRS difference.

One detail in decoding is that not all the newly allocated columns in reserved memory will be fully occupied and one or more columns can be empty even after storing all the compressed data due to the inaccuracy and ceiling function in the simple linear prediction. So an 'endmark' is needed to notify the memory management block to stop reading in data when it has all necessary bits in. Fortunately, there are some unused R4S4 codes which can be used as 'endmark'. The R4S4 code with SSSS=0000 and RRRR from 0000 to 1110 are not used and any of them can be used as an 'endmark'.

Another important aspect of this memory management algorithm is that it may compromise the integrity of the data. Data loss may happen when the algorithm cannot allocate more space for newly overflowed columns thus turning the original DRHE into a lossy compression scheme. Data loss will happen in the 'drop cData' step in Figure 5-22. Before the first time of dropping in a certain column, the column number will be written into the fullCol list. Once a column is recorded in fullCol list, the new incoming data in this column will be lost. This case usually only happens in bad case scenarios. Table 5-5 gives an example fullCol list under bad case scenario and also shows at which sweep it becomes full.

Table 5-5 Example fullCol list (with bad case input signal)

fullCol	50	55	253
Full at Sweep No.	121	121	127

Take column 50 as an example, the emMap has its record as: orgCol: 50, startPos: 660, len: 2, curPos: 661 meaning 2 extra columns starting at 660^{th} column are allocated to store the overflowed data in column 50. The final length of column 660 is 290 bits which is totally full and the length of column 661 is 288 bits. The 2 bits free space is not enough for storing the compressed data at sweep 121 any data belongs to column 50 after sweep 121 is discarded. The same applies to column 55 and 253 in the example fullCol list.

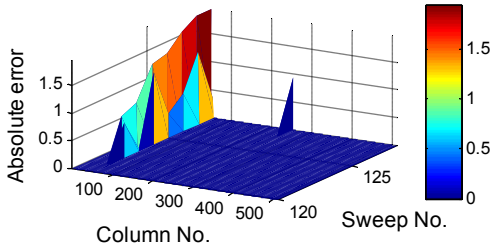


Figure 5-25 Absolute error between original s2D and s2D computed from uncompressed sRS in bad case (sweeps that are not shown are all 0)

When decompressing the compressed data with loss, the lost data is replaced with 0s and the decompressed difference is converted back to sRS followed by the 2^{nd} FFT which will generate s2D. Figure 5-25 shows the absolute error between original s2D and the s2D with data loss in

compression. Because the prediction is done only based on the previous one sweep, so data loss will only influence the sweeps that are after its beginning and in this example they're 121, 121 and 127 for column 50, 55 and 253 respectively.

5.8 Hardware Considerations

First of all, the overhead of implementing this compression scheme has to be evaluated. DRHE is quite different from the VLE scheme. Compared to the first scheme which has almost only memory overhead but no computation overhead, DRHE will introduced much more computation overhead due to the use of Huffman coding.

In terms of memory overhead, it comes from various tables and records that are needed to record the memory mappings. The length of emMap cannot be predicted, but its maximum length is definitely smaller than the number of columns in normal memory and smaller than the number of columns in reserved memory. According to simulations, the number of records (one record is one column in Table 5-4) is around 1/8 of the number of columns in normal memory. So we allocate $N_{norm}/4$ records in emMap. For each field in one record, $\lceil log_2 N_{norm} \rceil$ bits are used. In total, the memory overhead for emMap is:

$$OH_{emMap} \approx \frac{N_{norm}}{4} \times 4 \times \lceil \log_2 (N_{norm}) \rceil = \mathrm{N}_{norm} \lceil \log_2 (N_{norm}) \rceil \text{ bits} \quad (76.)$$

Similarly, the memory overhead for fullCol list is:

$$OH_{fullCol} = \frac{N_{norm} + N_{reserve}}{4} \times \lceil \log_2 (N_{norm} + N_{reserve}) \rceil \text{ bits} \quad (77.)$$

Memory overhead for memPointer is:

$$OH_{memP} = (N_{norm} + N_{reserve}) \times \lceil \log_2 (colBits) \rceil \text{ bits} \quad (78.)$$

The zCounters also need memory to store their counter values. Each zCNT needs 4 bits and every column in normal memory has one zCNT, so the total overhead is:

$$OH_{zCNT} = 4N_{norm} \text{ bits} \quad (79.)$$

Another memory overhead is the space for storing the Huffman code table which has 256 items. Each item is at most 16 bits long so the total space for the table is:

$$OH_{hTable} = 256 \times 16 = 4096 \text{ bits} \quad (80.)$$

When CR_{tar} is set to 5 and CR_{th} is set to 7, N_{norm} is 512, $N_{reserve}$ is set to 205 according to the previous discussion. colBits is set to 290. So the total memory overhead is:

$$OH_{DRHE} = OH_{emMap} + OH_{fullCol} + OH_{memP} + OH_{zCNT} + OH_{hTable} = 18486\ bits \quad (81.)$$

which is 1.76% of the original size or 8.80% of the compressed size. This memory overhead is significantly higher than that of the first compression scheme but still in an acceptable range.

Computation overheads are mainly from the use of the Huffman encoding and the search operations in various records such as emMap and fullCol. The Huffman encoding and decoding can both be implemented in software and hardware depending on the requirements which makes design choice quite flexible.

Besides reasonable overhead, other considerations have been taken into account for hardware implementation. Column based encoding and the memory management algorithm are already introduced in Chapters 5.5 and 5.7. These two techniques make the scheme suitable for the stream-like signals coming in sweep by sweep.

Another practical issue is the prediction block. As shown in Figure 5-4, the prediction block can be implemented as two IIR filters, but in hardware, floating point representation is too costly. Proper fixed point representation has to be defined for each block inside Figure 5-6 to pre-

serve enough accuracy and save register length at the same time. This is again a **tradeoff**: a longer number format gives better accuracy but has larger area and consumes more power. According to Chapter 4.3, the processing gain from the 1024 points FFT will lower the noise floor of the input signal by 4 bits. To preserve this noise floor, a magnitude format is chosen as [0,16,4] and phase format is chosen as [1,16,13] after the data conversion. The final difference results diffR and diffI are in [1,16,4] format as shown in Figure 5-6. The design space for this prediction block is not fully explored in this thesis. It's possible to move the two IIR into software that is running on a processor using much longer data format. For the complex to mag/pha and mag/pha to complex conversion, there are also many possible implementations in both hardware and software which remains to be explored.

For the Huffman code table, it's good to have all the codes within a certain maximum length. A known maximum length will make decoding more robust. To generate the code table in Appendix A, the length limited Huffman coding can be used [64]. This variant of Huffman coding can make sure that each code in the Huffman code table is within a certain length, 16 bits, for example, is used here. Codes in Appendix A are generated with the statistics from one signal in a normal scenario. This fixed table is used throughout the design and test of the DRHE and performs well. Of course, more advanced techniques like the adaptive Huffman coding can be used but they have to be evaluated and modified carefully in order not to add too much overhead to the system.

Last but not least, all above calculations and designs like overhead estimation, memory allocation and various records are done **in the unit of bit**. However, in a real system, byte or word addressing is more realistic. So bits have to be packed into bytes before storing into the memory which may cause even more overhead and space waste but the general idea of this compression scheme still applies.

5.9 Simulation and Results

The simulation of column based DRHE with memory management in bad case is already shown in Figure 5-24. The system specification is also the one shown in Table 2-1. In the following simulation, 5 targets with randomly generated ranges and velocities are used. RCS variation is also considered using the Toyota Prius parameter also used to generate Figure 4-7. The target CR is set to $\mathrm{CR}_{\mathrm{tar}} = 5$ and the threshold is set to $\mathrm{CR}_{\mathrm{th}} = 7$. So the normal memory region has 512 columns of 290 bits and reserved memory region has 205 columns of 290 bits.

Figure 5-26 (a) shows the column based DRHE with memory management in a benign case:

$$\begin{aligned} R &= \begin{bmatrix} 10 & 20 & 35 & 45 & 60 & 70 \end{bmatrix} \mathrm{m} \\ vD &= \begin{bmatrix} 30 & -10 & 20 & -20 & 15 & -5 \end{bmatrix} / 3.6 \ \mathrm{m/s} \end{aligned} \tag{82.}$$

In this case, only several columns exceeded $\mathrm{CR}_{\mathrm{th}}$ and most of the reserved memory space is empty shown in Figure 5-26 (b) comparing to Figure 5-24.

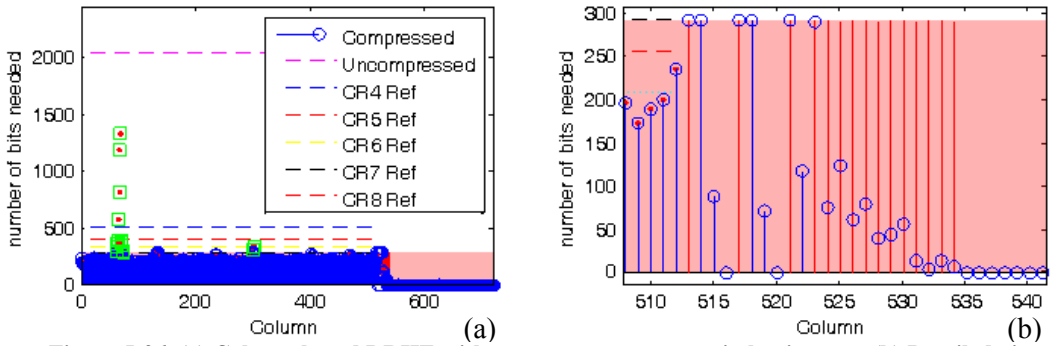


Figure 5-26 (a) Column based DRHE with memory management in benign case (b) Detailed view

Figure 5-27 shows the test result using one set of real measurement data from NXP Semiconductor. It was obtained in the initial designs of the analog front end after deramping. Figure 5-27(a) is the Range-Velocity spectrum showing one moving target while Figure 5-27 (b) is the

column based DRHE result. It successfully fits all the data into a fixed size memory (achieving CR=5 as stated before).

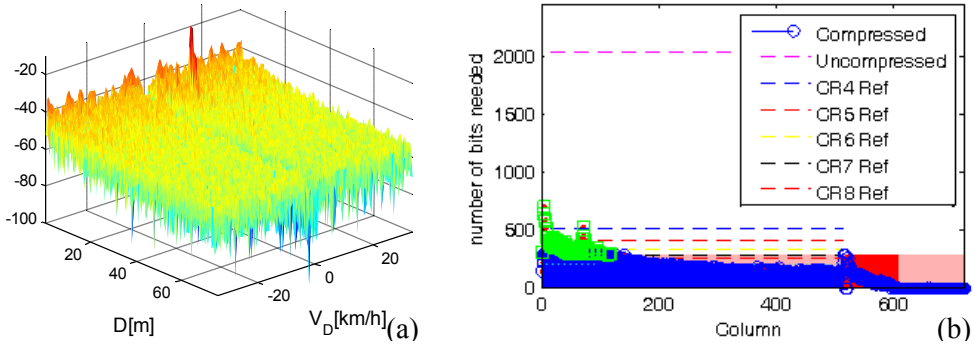


Figure 5-27 (a) s2D of real measurement data (b) DRHE result of real measurement data

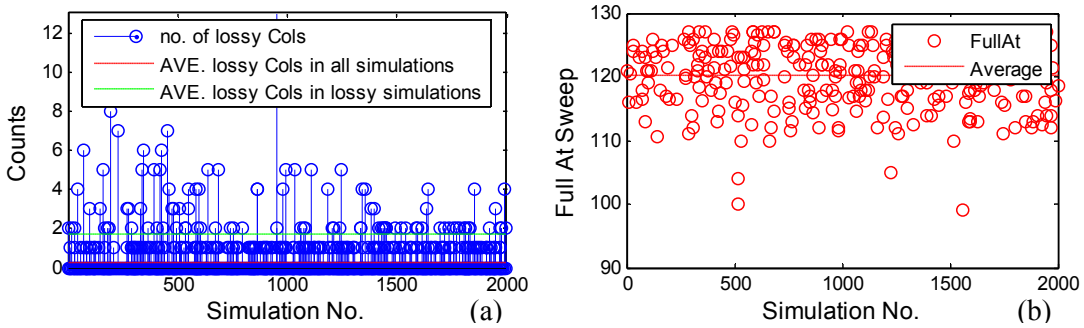


Figure 5-28 (a) Lossy columns in simulations (b) Full-at-sweep graph for all lossy columns

Though in normal cases, this compression scheme is lossless, due to the memory management, data loss may happen in some bad case scenarios. It's good to know the frequency that data loss will happen and how much data will be lost. Figure 5-28 shows two relevant graphs of data loss.

Figure 5-28 (a) shows the number of lossy columns in each simulation out of 2000 simulations. For example, in the 120th simulation, 3 columns have data loss. 296 cases of data loss happened in 2000 simulations. The average number of lossy columns is 0.252 for all simulations and is 1.706 for all simulations with data loss. Figure 5-28 (b) shows the sweep numbers (Full-At-Sweep) at which data losses begun in the lossy simulations. In the total 296 cases, the average number of Full-At-Sweep is 120 (128 sweeps in total).

Overall, 3873 data points in 505 columns are lost in 2000 simulations resulting in an average of 1.937 points loss per simulation which is well in acceptable range.

6 Conclusions and Recommendations

The main aim of this thesis is to design a compression algorithm that can be used to reduce the memory that is needed for storing the intermediate data in the FMCW radar signal processing flow. In Chapter 1, the principle of data compression is briefly reviewed and various implementations and applications are discussed.

With a brief explanation of the importance and necessity of the radar in automotive in Chapter 1, Chapter 2 presents the principle of the FMCW radar and the mathematical deductions of the FMCW radar signal processing flow. Range and velocity is extracted from the beat frequency and the Doppler shift by two FFT operations. Based on this theoretical deduction, measurement resolutions are calculated. This Chapter concludes by presenting the system specification and parameters used throughout the whole design process.

Based on the characteristics of the Range-Sweep spectrum signal in the signal processing, in Chapters 4 and 5, two compression schemes are designed i.e. the Range Dependent Variable Length Encoding and the Doppler Redundancy Hybrid Encoding. The former scheme takes advantage of the power profile in signal transmission and stores signals from difference ranges in different bit-length formats. In addition, the clipping reduction method and the Uniform Dynamic Range encoding are designed to further increase the compression ratio. Simulation shows that a reasonable combination is RDVLE + CRM + 10-bit UDRE which provides a CR of 2.27 with acceptable loss. This scheme is simple to implement but cannot provide enough CR. Harmonics and distortions introduced by clipping and bit-removing are also potential problems for the following target recognition stages.

The second compression scheme is designed to be lossless. It takes advantage of the redundancy residing in the Doppler spectrum or in other words, the similarities in the signals of consecutive sweeps. Predictions are made based on previous samples and the differences between predictions and real inputs are encoded and stored. Due to the use of the Run Length Encoding and the Huffman encoding, the length of the encoded data cannot be predicted and the raw compression ratio is between 6.5 and 9.5. By introducing a memory management block, the compressed data can be filled in a fixed size 'rectangle-shape' memory with fixed preset CR. However, this memory management block may introduce some data losses in extreme cases. In simulations when target CR is set to 5 and threshold CR is set to 7, only less than 2 samples are lost in one measurement (128 sweeps). As discussed in Chapter 1, lossy compression usually has better compression ratio than lossless compression but this is not the case in this thesis because RDVLE does not fully exploit the characteristics of the signals. With the capability of reducing memory requirements to 20% of its original, the column based DRHE with memory management is promising in FMCW radar signal processing flow with proper hardware/software partition.

Besides the above work, many improvements and additional designs and researches remain to be done. Some recommendations and future work are listed below.

Encoding in the RDVLE

Current RDVLE design stores the bit-removed signals without any further encoding. Inspired by the DRHE scheme, further encoding might be used to increase the compression ratio (or in other words, we can combine RDVLE with DRHE to provide a more efficient lossy compression scheme).

Test with Target Recognition Stages

The ultimate performance metric for an automotive radar system is its target detection rate. Both RDVLE and DRHE will cause data loss to different extents which will have influence on signal shape and eventually on detection rate. Recognition stages are not available during this thesis work so the compression schemes cannot be tested with it. In consideration with algo-

rithms like the UDRE which will raise noise floors in different regions, recognition algorithm has to be designed together with compression scheme to have a good match.

Signal Prediction

The quality of the signal prediction in the DRHE determines the CR directly, in current design, simple linear prediction is used based only on one previous sweep. In benign cases, the prediction is good but it cannot predict well in the bad cases where two or more targets appear at the same range. New prediction methods can be used, for example, to predict based on more number of previous sweeps or to use other prediction methods with reasonable computation overhead.

Adaptive Huffman Coding

Current DRHE uses the static Huffman coding with a fixed code dictionary derived from the example signal. To eliminate the need of dictionary storage, the adaptive Huffman coding can be used. However, the overhead introduced has to be carefully evaluated.

More Sophisticated Memory Management

In the memory management block used in column based DRHE, dynamic memory allocation in the reserved memory will happen no more than once as shown in the flow chart in Figure 5-22. This means when allocated reserved memory is full again, new incoming data will be lost even there is still free space in reserved memory. Memory management can be modified to allow more times of memory allocation which compensates the space prediction inaccuracy, however, this will make the logics even more complex and more columns in emMap are needed to store memory pointers.

Hardware/Software Partition

Hardware implementation for the RDVLE is straightforward, but the column based DRHE seems too complex to be implemented only in hardware especially the Huffman encoder, decoder and the memory management blocks. Part of the algorithms can be implemented in a MCU or DSP accompanying high speed hardware processing blocks such as FFT. Besides assisting compression algorithm execution, the processor can perform other tasks like communication and interrupts handling for the whole radar system. To achieve optimal speed and chip area balance, proper hardware/software partition has to be done.

List of Tables

Table 1-1 Example symbol frequency table 3
Table 1-2 Huffman code of example model 4
Table 1-3 LZ77 encoding steps with example input 5
Table 2-1 System specifications and parameters overview 18
Table 5-1 Entropy of example signal with different symbols 43
Table 5-2 SSSS regions table 45
Table 5-3 Some local peaks in R4S4 symbol histogram and their meanings 46
Table 5-4 emMap 49
Table 5-5 Example fullCol list (with bad case input signal) 52

List of Figures

Figure 1-1 Modeling + Coding model for compression algorithm 2
Figure 1-2 Adaptive modeling (compression) 3
Figure 1-3 Huffman tree generation 4
Figure 1-4 Aided parking [44] 6
Figure 1-5 How a self-driving car works [46] 7
Figure 1-6 Difference between compression for transmission/storage and compression in between processing 7
Figure 2-1 (a) Transmitted saw tooth FMCW signal (b) Returned signal with a delay (c) Time-Frequency relationship of the two signals. 10
Figure 2-2 General FMCW radar system diagram 10
Figure 2-3 2D FFT processing diagram for the FMCW signal 12
Figure 2-4 Deramped signal rearranged to 2D 'matrix' form 14
Figure 2-5 Range-Sweep spectrum after 1st FFT with single target 15
Figure 2-6 Range-Velocity spectrum after 2nd FFT 16
Figure 2-7 Range-Velocity spectrum of 3 targets with different speeds (a) without noise (b) with AWGN 16
Figure 2-8 Quadrature deramping 18
Figure 3-1 Memory in the 2D FFT processing 19
Figure 3-2 Single chip SoC radar solution 19
Figure 3-3 Chip die photo of TI TMS320C50 DSP [56] (memory blocks shown in red box) 20
Figure 4-1 Range vs. Received power in different scenario [57] 21
Figure 4-2 Fixed point data format 22
Figure 4-3 (a) Received power–range model developed by NXP based on the segmented model (b) simplified version which is actually used 23
Figure 4-4 Object cross-section and the RCS 23
Figure 4-5 RCS of a human dummy 0-360 degree azimuth in 24GHz and 77GHz band [61] 24
Figure 4-6 Major RCS contributions from the back of a car 24
Figure 4-7 Weibull distribution parameters of the RCS of 24 measured cars [62] 25
Figure 4-8 PDF and CDF of Weibull distribution under different parameters 25
Figure 4-9 FFT processing gain example (half spectrum) 26
Figure 4-10 The RDVLE with bit length zones (noise floor is normalized to 0dB) 26
Figure 4-11 The Uniform Dynamic Range Encoding 27
Figure 4-12 The RDVLE + UDRE encoding flow (with example in R4 region) 28
Figure 4-13 Example result after the RDVLE + UDRE 28
Figure 4-14 Regional scan peak (target) detection 28
Figure 4-15 Error cause bit removing (data in unsigned binary, x means removed) 29
Figure 4-16 (a) All FP processing, no bits removed (b) RDVLE scheme with clipping happened 29

Figure 4-17 Direction of the 2nd FFT (shown in dash arrow line) 29
Figure 4-18 Consistent scaling in column (a) sRS (b) detail view of the first peak unscaled in the noted viewing direction with dash arrow line (c) detail view of the first peak scaled down by 4.....30
Figure 4-19 2nd FFT result (s2D) (a) with scaled peak (b) with original peak.....30
Figure 4-20 CRM in action (a) sc are initialized to all 1, 1st FFT result from the first sweep comes in. (b) In sweep 3, the 4th column has one data above the regional threshold. (c) sc[4] is set to b and the clipped data is scaled by b, cPos[4][1] is set to the sweep number – 3. (d) All new data in the column of 4 is scaled by b until at sweep 7, the data is still clipped even after the first scaling. (e) sc[4] is further increased to b^2 and the clipped data is scaled again, cPos[4][2] is set to 7 (f) All sweeps are received, according to cPos record, data[3-6][4] (4th column, from 3rd to 6th row) is further scaled by b, data[1-2][4] are further scaled by b^2. 31
Figure 4-21 Overhead for the RDVLE + UDRE 32
Figure 4-22 Example of generated deramped signal.....34
Figure 4-23 Simulation setup.....34
Figure 4-24 (a) Simulation result: fixed length encoding 1st FFT [1,15,4], 2nd FFT [1,19,8], no RCS distribution, fixed ranges and velocities. (b) Fixed length encoding visualization for 1st FFT 35
Figure 4-25 (a) Simulation result: VLE for both FFT results, no RCS distribution, statistical range and velocity generation for 5 targets (b) VLE visualization for 1st FFT 35
Figure 4-26 (a) Simulation result: VLE for both FFT results, fixed RCS distribution (Prius), statistical range and velocity generation for 5 targets, without CRM (b) Simulation result: conditions are the same as those in (a) but with CRM.....36
Figure 4-27 (a) All-FP reference s2D (b) s2D without CRM, total RMSE is -7.50dB (c) s2D with CRM total RMSE is -9.90dB (all three z-axes are in dB scale).....36
Figure 4-28 (a) SNR of s2D in all simulations (b) SNR of s2D with Total RMSE greater than or equal to -10dB 37
Figure 4-29 RDVLE + UDRE (10 bits) formats in different coding regions 37
Figure 4-30 SNR: RDVLE + UDRE (10bits): (a) all simulations (b) RMSE greater than or equal to -10 dB.....37
Figure 4-31 SNR: RDVLE + UDRE (8bits): (a) all simulations (b) RMSE greater than or equal to -10 dB 38
Figure 4-32 SNR: RDVLE + UDRE (6bits): (a) all simulations (b) RMSE greater than or equal to -10 dB 38
Figure 5-1 An example of magnitude and phase of sRS.....39
Figure 5-2 Real and Imaginary part of sRS 39
Figure 5-3 General structure of the DRHE compression scheme 40
Figure 5-4 Predictions viewed as IIR filters: (a) Magnitude prediction (b) Phase prediction ...40
Figure 5-5 Frequency responses of the two IIR filters: (a) Magnitude prediction (b) Phase prediction 41
Figure 5-6 Prediction and difference stage 41
Figure 5-7 Differences between prediction and real input: (a) difference in real part (b) difference in imaginary part in both figure, $\boldsymbol{\alpha = 0.3\ \beta = 0.7}$ 41
Figure 5-8 (a) One sweep of rDiff (b) One column of rDiff.....42
Figure 5-9 Symbol histograms: (a) original 16-bit symbol (b) least significant 8 bits from 16-bit symbol (c) most significant 8 bits from 16-bit symbol.....43
Figure 5-10 JPEG encoding algorithm [24].....44
Figure 5-11 Column based compression and easy memory addressing for decoding 44
Figure 5-12 (a) Expanding ranges (b) Data shifting for generating APPEND bits, underscores note sign bits 45
Figure 5-13 The R4S4+APPEND encoding for example data.....45
Figure 5-14 R4S4 symbol histogram 45
Figure 5-15 The Huffman coding of R4S4 field.....46
Figure 5-16 zCounter in action 46

Figure 5-17 Compressed size of each column (CRx=Compression Ration of x) 47
Figure 5-18 Column based DRHE decoding flow graph 47
Figure 5-19 (a) Bad case target distribution (b) Compressed column length 48
Figure 5-20 (a) Magnitude of sRS in bad case scenario (b) rDiff in bad case scenario 48
Figure 5-21 Memory reshaping 49
Figure 5-22 Flow chart of memory management algorithm for one sweep 50
Figure 5-23 Final length prediction error 51
Figure 5-24 Memory management in action with bad case input data: (a) First several sweeps, no overflow (b) Some columns starts to overflow, new columns are allocated in reserved memory based on prediction (c) More columns overflow, more reserved memory are used. (d) All sweeps have come in, all data is fit into the fixed size memory 51
Figure 5-25 Absolute error between original s2D and s2D computed from uncompressed sRS in bad case (sweeps that are not shown are all 0) 52
Figure 5-26 (a) Column based DRHE with memory management in benign case (b) Detailed view 54
Figure 5-27 (a) s2D of real measurement data (b) DRHE result of real measurement data 55
Figure 5-28 (a) Lossy columns in simulations (b) Full-at-sweep graph for all lossy columns 55

Appendix A: The Huffman Codes of R4S4 Symbols

R4S4	Huffman Code
0	0000111000001011
1	1
2	0101
3	000010
4	010001
5	0000110
6	0000011
7	00000011
8	00000010
9	0110110
10	0110100
11	00000000
12	000011101
13	0000111111
14	0000111000001010
15	000011111011101
16	000011111011100
17	001
18	011001
19	011000101
20	01101110110
21	000011111011111
22	0000111000000
23	000011111011110
24	000011111011001
25	000011111011000
26	000011111011011
27	000011111011010
28	011000110110101
29	011000110110100
30	011000110110111
31	011000110110110
32	011000110110001
33	0001
34	0110101
35	000011100100
36	011000110110000
37	011000110110011
38	011000110110010
39	011000110111101
40	011000110111100
41	011000110111111
42	011000110111110
43	011000110111001
44	011000110111000
45	011000110111011
46	011000110111010
47	011000110100101
48	011000110100100
49	0111
50	01100000
51	011000110100111
52	011000110100110
53	011000110100001
54	011000110100000
55	011000110100011
56	011000110100010
57	011000110101101
58	011000110101100
59	011000110101111
60	011000110101110
61	011000110101001
62	011000110101000
63	011000110101011
64	011000110101010
65	01001
66	000011110
67	011011100010101
68	011011100010100
69	011011100010111
70	011011100010110
71	011011100010001
72	011011100010000
73	011011100010011
74	011011100010010
75	011011100011101
76	011011100011100
77	011011100011111
78	011011100011110
79	011011100011001
80	011011100011000
81	010000
82	011011111
83	011011100011011
84	011011100011010
85	011011100000101
86	011011100000100
87	011011100000111
88	011011100000110
89	011011100000001
90	011011100000000
91	011011100000011
92	011011100000010
93	011011100001101
94	011011100001100
95	011011100001111
96	011011100001110
97	0000010
98	0110001100
99	011011100001001
100	011011100001000
101	011011100001011
102	011011100001010
103	011011100110101
104	011011100110100
105	011011100110111
106	011011100110110
107	011011100110001
108	011011100110000
109	011011100110011
110	011011100110010
111	011011100111101
112	011011100111100
113	00000001
114	00001111100
115	011011100111111
116	011011100111110
117	011011100111001
118	011011100111000
119	011011100111011
120	011011100111010
121	011011100100101
122	011011100100100
123	011011100100111
124	011011100100110
125	011011100100001
126	011011100100000
127	011011100100011
128	011011100100010
129	01100001
130	000011100101
131	011011100101101
132	011011100101100
133	011011100101111
134	011011100101110
135	011011100101001
136	011011100101000
137	011011100101011
138	011011100101010
139	011000111010101
140	011000111010100
141	011000111010111
142	011000111010110
143	011000111010001
144	011000111010000
145	011000100
146	0000111000011
147	011000111010011
148	011000111010010
149	011000111011101
150	011000111011100
151	011000111011111
152	011000111011110
153	011000111011001
154	011000111011000
155	011000111011011
156	011000111011010
157	011000111000101
158	011000111000100
159	011000111000111
160	011000111000110

161	011011110
162	011000111000001
163	011000111000000
164	011000111000011
165	011000111000010
166	011000111001101
167	011000111001100
168	011000111001111
169	011000111001110
170	011000111001001
171	011000111001000
172	011000111001011
173	011000111001010
174	011000111110101
175	011000111110100
176	011000111110111
177	00001110011
178	011000111110110
179	011000111110001
180	011000111110000
181	011000111110011
182	011000111110010
183	011000111111101
184	011000111111100
185	011000111111111
186	011000111111110
187	011000111111001
188	011000111111000
189	011000111111011
190	011000111111010
191	011000111100101
192	011000111100100

193	00001110001
194	011000111100111
195	011000111100110
196	011000111100001
197	011000111100000
198	011000111100011
199	011000111100010
200	011000111101101
201	011000111101100
202	011000111101111
203	011000111101110
204	011000111101001
205	011000111101000
206	011000111101011
207	011000111101010
208	011011101010101
209	0000111000010
210	011011101010100
211	011011101010111
212	011011101010110
213	011011101010001
214	011011101010000
215	011011101010011
216	011011101010010
217	011011101011101
218	011011101011100
219	011011101011111
220	011011101011110
221	011011101011001
222	011011101011000
223	011011101011011
224	011011101011010

225	011011101111
226	011011101000101
227	011011101000100
228	011011101000111
229	011011101000110
230	011011101000001
231	011011101000000
232	011011101000011
233	011011101000010
234	011011101001101
235	011011101001100
236	011011101001111
237	011011101001110
238	011011101001001
239	011011101001000
240	0000111110101
241	0000111110100
242	000011100000100
243	011011101001011
244	011011101001010
245	011011101110101
246	011011101110100
247	011011101110111
248	011011101110110
249	011011101110001
250	011011101110000
251	011011101110011
252	011011101110010
253	0000111000001101
254	0000111000001100
255	000011100000111

Bibliography

[1] S. Wolfram, A new kind of science, Wolfram Media, 2002.

[2] S.-G. Choi, J.-H. Yoo, D.-O. Kang and J.-W. Lee, "Implementation of MPEG2-TS streaming system over wireless 1394 network," in *Advanced Communication Technology, 2004. The 6th International Conference on*, 2004.

[3] J. Nunez and S. Jones, "Run-length coding extensions for high performance hardware data compression," *IEE Proceedings-Computers and Digital Techniques,* vol. 150, no. 6, pp. 387-395, 2003.

[4] N. Jayant, J. Johnston and R. Safranek, "Signal compression based on models of human perception," *Proceedings of the IEEE,* vol. 81, no. 10, pp. 1385-1422, 1993.

[5] C. Christopoulos, A. Skodras and T. Ebrahimi, "The JPEG2000 still image coding system: an overview," *Consumer Electronics, IEEE Transactions on,* vol. 46, no. 4, pp. 1103-1127, 2000.

[6] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review,* vol. 5, no. 1, pp. 3-55, 2001.

[7] M. Nelson and J. Gailly, The Data Compression Book, Wiley, 1996.

[8] "What is the frequency of the letters of the alphabet in English?," [Online]. Available: http://www.oxforddictionaries.com/words/what-is-the-frequency-of-the-letters-of-the-alphabet-in-english.

[9] D. A. Huffman and others, "A method for the construction of minimum redundancy codes," *proc. IRE,* vol. 40, no. 9, pp. 1098-1101, 1952.

[10] K. Sayood, Introduction to Data Compression, Morgan Kaufmann, 2012.

[11] J. L. Nunez and S. Jones, "Gbit/s lossless data compression hardware," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 11, no. 3, pp. 499-510, 2003.

[12] I. H. Witten, R. M. Neal and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM,* vol. 30, no. 6, pp. 520-540, 1987.

[13] J. G. Cleary and I. Witten, "Data compression using adaptive coding and partial string matching," *Communications, IEEE Transactions on,* vol. 32, no. 4, pp. 396-402, 1984.

[14] A. Moffat, "Implementing the PPM data compression scheme," *Communications, IEEE Transactions on,* vol. 38, no. 11, pp. 1917-1921, 1990.

[15] G. V. Cormack and R. Horspool, "Data compression using dynamic Markov modelling," *The Computer Journal,* vol. 30, no. 6, pp. 541-550, 1987.

[16] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on information theory,* vol. 23, no. 3, pp. 337-343, 1977.

[17] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *Information Theory, IEEE Transactions on,* vol. 24, no. 5, pp. 530-536, 1978.

[18] T. A. Welch, "A technique for high-performance data compression," *Computer,* vol. 17,

no. 6, pp. 8-19, 1984.

[19] "LZO library 2.08," [Online]. Available: http://www.oberhumer.com/opensource/lzo/.

[20] "LZ77 Compression Algorithm," [Online]. Available: http://msdn.microsoft.com/en-us/library/ee916854.aspx.

[21] A. B. Watson, J. Hu and J. F. MCGOwAN, "Digital video quality metric based on human vision," *Journal of Electronic imaging,* vol. 10, no. 1, pp. 20-29, 2001.

[22] F. W. Campbell and J. Robson, "Application of Fourier analysis to the visibility of gratings," *The Journal of physiology,* vol. 197, no. 3, p. 551, 1968.

[23] G. K. Wallace, "The JPEG still picture compression standard," *Consumer Electronics, IEEE Transactions on,* vol. 38, no. 1, pp. xviii--xxxiv, 1992.

[24] I. IEC, "Information technology-digital compression and coding of continuous-tone still images: Requirements and guidelines," *Standard, ISO IEC,* pp. 10918-1, 1994.

[25] I. Iso, "IEC 11172-3 Information technology-coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s-Part3: Audio," *Motion Picture Experts Group,* 1993.

[26] N. Ahmed, T. Natarajan and K. R. Rao, "Discrete cosine transform," *Computers, IEEE Transactions on,* vol. 100, no. 1, pp. 90-93, 1974.

[27] I. Rec, "H. 262| ISO/IEC 13818-2," *Information technology—Generic coding of moving pictures and associated audio information—Video,* 2000.

[28] P.-W. Cheng, C.-C. Shen and P.-C. Li, "MPEG compression of ultrasound RF channel data for a real-time software-based imaging system," *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on,* vol. 59, no. 7, pp. 1413-1420, 2012.

[29] D. J. Craft, "A fast hardware data compression algorithm and some algorithmic extensions," *IBM Journal of Research and Development,* vol. 42, no. 6, pp. 733-746, 1998.

[30] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," *Solid-State Circuits, IEEE Journal of,* vol. 41, no. 3, pp. 712-727, 2006.

[31] S.-C. Lai, W.-C. Chien, C.-S. Lan, M.-K. Lee, C.-H. Luo and S.-F. Lei, "An efficient DCT-IV-based ECG compression algorithm and its hardware accelerator design," in *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, 2013.

[32] "zlib 1.2.8," [Online]. Available: http://www.zlib.net/.

[33] Z. Zhou, "Data compression for radar signals: an SVD based approach," 2001.

[34] L. Yang, R. P. Dick, H. Lekatsas and S. Chakradhar, "Online memory compression for embedded systems," *ACM Transactions on Embedded Computing Systems (TECS),* vol. 9, no. 3, p. 27, 2010.

[35] "compcache," [Online]. Available: https://code.google.com/p/compcache/.

[36] "zcache: a compressed page cache," [Online]. Available: http://lwn.net/Articles/397574/.

[37] "zswap: compressed swap caching," [Online]. Available: http://lwn.net/Articles/528817/.

[38] "Apple Compressed Memory," [Online]. Available:

https://www.apple.com/media/us/osx/2013/docs/OSX_Mavericks_Core_Technology_Overview.pdf.

[39] P. R. Wilson, S. F. Kaplan and Y. Smaragdakis, "The Case for Compressed Caching in Virtual Memory Systems.," in *USENIX Annual Technical Conference, General Track*, 1999.

[40] R. B. Tremaine, P. A. Franaszek, J. T. Robinson, C. O. Schulz, T. B. Smith, M. E. Wazlowski and P. M. Bland, "IBM memory expansion technology (MXT)," *IBM Journal of Research and Development,* vol. 45, no. 2, pp. 271-285, 2001.

[41] L. Vlacic, M. Parent and F. Harashima, Intelligent vehicle technologies: theory and applications, Butterworth-Heinemann, 2001.

[42] Z. Wei, Z. Lihua and L. Kun, "Research of Electronic Parking Radar Based on Ultrasonic Ranging System [J]," *Computer Automated Measurement \& Control,* vol. 4, 2004.

[43] K. A. Lukin, "Noise radar with correlation receiver as the basis of car collision avoidance system," in *Microwave Conference, 1995. 25th European*, 1995.

[44] "Valeo PARK4U system," [Online]. Available: http://www.valeo.com/en/page-transverses-gb/popin-diaporama-en/popin-diaporama-cda-en/diaporama-park4u.html.

[45] E. Guizzo, "How google's self-driving car works," *IEEE Spectrum Online, October,* vol. 18, 2011.

[46] "Inside story: Look, no hands," [Online]. Available: http://www.economist.com/node/21560989.

[47] V. Lakshmanan, "Overview of radar data compression," in *Optical Engineering+ Applications*, 2007.

[48] U. Benz, K. Strodl and A. Moreira, "A comparison of several algorithms for SAR raw data compression," *Geoscience and Remote Sensing, IEEE Transactions on,* vol. 33, no. 5, pp. 1266-1276, 1995.

[49] B. J. Sundersingh, "Qualitative evaluation of data compression in real-time ultrasound imaging," 2000.

[50] Y.-F. Li and P.-C. Li, "Ultrasound beamforming with compressed data," in *Ultrasonics Symposium (IUS), 2010 IEEE*, 2010.

[51] J. L. Eaves and E. K. Reedy, Principles of modern radar, vol. 197, Springer, 1987.

[52] C. A. Balanis, "Fundamental parameters of antennas," *Antenna Theory Analysis and Design,* pp. 27-132, 1982.

[53] A. G. Stove, "Linear FMCW radar techniques," in *IEE Proceedings F (Radar and Signal Processing)*, 1992.

[54] IEEE, "IEEE Standard Letter Designations for Radar-Frequency Bands," *IEEE STANDARD,* 2002.

[55] "TI C667x DSP Product Page," [Online]. Available: http://www.ti.com/lsds/ti/dsp/keystone/c667x/products.page.

[56] "TI TMS320C50 DSP die photo," [Online]. Available: http://commons.wikimedia.org/wiki/File:TI_TMS320C50_DSP_die.JPG.

[57] F. Jansen, "Signal Processing for Automotive Rardar Systems (NXP internal)".

[58] J. G. Proakis, Digital signal processing: principles algorithms and applications, Pearson Education India, 2001.

[59] M. A. Richards, Fundamentals of radar signal processing, Tata McGraw-Hill Education, 2005.

[60] E. C. Desk, "Electronic Warfare and Radar Systems Engineering Handbook," 1997.

[61] J. Fortuny-Guasch and J.-M. Chareau, "Radar Cross Section Measurements of Pedestrian Dummies and Humans in the 24/77 GHz Frequency Bands: Establishment of a Reference Library of RCS Signatures of Pedestrian Dummies in the Automotive Radar Bands," 2013.

[62] Rini Sherony, "Radar Scanning for Development of Vehicle and Pedestrian Surrogate Targets for Vehicle Pre-Collision System (PCS) Testing," [Online]. Available: http://www.sae.org/events/gim/presentations/2013/sherony_rini.pdf.

[63] N. Yamada, "Three-Dimensional High Resolution Measurement of Radar Cross," *R&D Review of Toyota CRDL,* vol. 36, 6 2001.

[64] L. L. Larmore and D. S. Hirschberg, "A fast algorithm for optimal length-limited Huffman codes," *Journal of the ACM (JACM),* vol. 37, no. 3, pp. 464-473, 1990.