AN ABSTRACT OF THE DISSERTATION OF

Vahid Behravan for the degree of Doctor of Philosophy in Electrical and Computer Engineering presented on February 2, 2017.

Title: Compressive Sensing for Low Power Sensor Design

Abstract approved: _____

Patrick Y. Chiang

Recent sensor System-on-Chips (SoC) have enabled significant advances in energy-efficiency by incorporating various micro-powered building blocks. Unfortunately, most of these sensor systems do not address the high power cost associated with data storage and transmission, which in some cases vastly exceeds the power consumed by the rest of the SoC. In recent years, Compressive-Sensing (CS) has been proposed as a method to accomplish significant sensor data compression, achieving compression rates up to 10x depending on the signal sparsity.

This work addresses conventional CS issues including non-adaptive compression rate and offers a solution. First, a feasibility study is conducted to investigate the sparsity variance of some biomedical signals. Then an adaptive CS framework is proposed, to adjust the compression rate based upon the input signal's sparsity on-the-fly. Thirdly, a CS framework is proposed, the reconstruction of which is aided by statistics collection. It is demonstrated how to fuse sensor data and statistics information together to improve the signal-to-error ratio (SER) of reconstruction. A test chip fabricated in TSMC 65-nm technology to implement the algorithm in a SoC incorporating statistics collection block in order to improve performance of the CS algorithm.

The final portion of this research devoted to study emerging application of time-of-flight cameras for depth measurement. These devices generate a 3 Dimensional (3D) point cloud that basically includes 3D details of objects in front of them. A framework to apply CS to 3D point cloud data is presented. Finally it demonstrates how the idea of adaptive CS can be used for 3D point cloud data compression.

Compressive Sensing for Low Power Sensor Design

by
Vahid Behravan

A DISSERTATION
submitted to
Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented February 2, 2017
Commencement June 2017

Doctor of Philosophy dissertation of Vahid Behravan presented on February 2, 2017

APPROVED:

Major Professor, representing Electrical and Computer Engineering

Director of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Vahid Behravan, Author

ACKNOWLEDGEMENTS

I would like to thank my adviser professor Patrick Chiang for supporting me in conducting this research. He is a great leader and mentor. His unconventional views on academic research and circuit design have shaped my views about my professional career in valuable ways.

I would like to express my sincere gratitude to my committee members. Dr. Huaping Liu, Dr. Matthew Johnston, Dr. Jinsub Kim and Dr. Michelle Dolgos served as my committee members and gave me their insightful comments and suggestions. A special thanks to Dr. Raviv Raich, who is a great researcher and an excellent teacher. I took all of his graduate courses and couldn't have been more satisfied.

I would like to thank all my collaborators especially Joe Crop, Neil Glover and Dr. Mohammed Shoaib. Without their support, this dissertation would not have been possible.

Finally, a special thanks to my family. I am so grateful to my parents for all of their invaluable support. I would like to express appreciation of my beloved wife Arezou, who was my constant support even when there was no one else to turn to.

CONTRIBUTION OF AUTHORS

Neil Glover and Rutger Farry helped to develop a wireless hardware to verify part of this work. Dr. Mohammed Shoaib provided some useful resources and contributed in technical discussions. Shuo Li and Chia-Hung Chen designed analog part of fabricated SoC for this research. Gurjeet Singh helped in collection of 3D point cloud data.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

1.Introduction

1.1 Motivation and Problem Definition

Recent sensor System-on-Chip (SoC) have enabled significant advances in energy-efficiency, incorporating various micro-powered building blocks, including low-power analog amplifier front-end [1], ADC [2], sub/near-threshold DSP [3], and radio [4]. Unfortunately, most of these sensor systems have ignored the problem of sensor data local storage, where the power consumed by storing (or transmitting) the data greatly exceeds the power consumed by the rest of the SoC. For example, a 14-day 10uW ECG patch sampling at 100Hz will require over 2Gb of data storage. Storing this large amount of data in the off-chip Flash RAM or sending the data via the wireless radio will be infeasible, given the I/O power of Flash-Memory or Bluetooth. Figure 1-1 shows how much energy different building blocks will consume to process this amount of data. Here, we used some commercial integrated circuits for reference. It is very clear that most of the system energy will be consumed in either radio or flash memory. A similar problem exists for other sensor applications, such as environmental monitoring or smart buildings. Therefore to reduce energy consumption significantly, one can reduce the total number of data system needs to transmit or process. In other words, solution is data compression as shown in Figure 1-2.

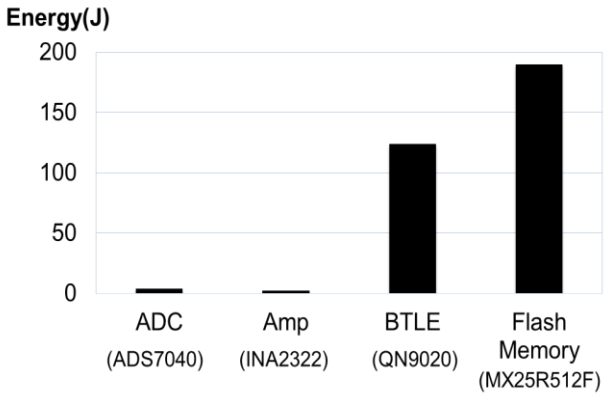

Figure 1-1: Energy consumption of some commercial integrated circuits

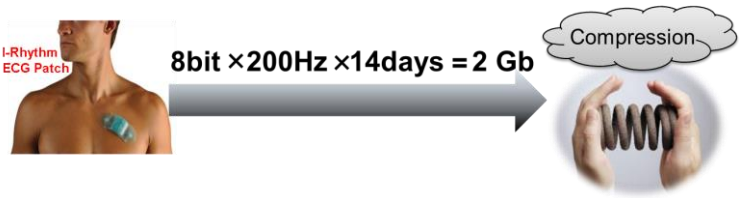

Figure 1-2: Data compression as an energy saving solution in wireless SoC

In recent years, compressive sensing (CS) has been proposed as a method to accomplish significant sensor data compression, achieving compression rates from 2-10x, depending on the signal sparsity [5]. Unfortunately, these CS systems assume an *a-priori*, blind compression rate that may miss important input signal information. The sensor input is negligible in general, however when an event occurs, it should not be overlooked by the CS compression.

1.2 An Introduction to Compressive-Sensing

In this section a brief explanation of compressive-sensing algorithm is presented. This section is adopted from chapter 1 of [6]. In recent years, compressive sensing has attracted considerable attention in many areas by suggesting that it might be possible to surpass the traditional limits of sampling theory. CS builds upon the fundamental fact that we can represent many signals using only a few non-zero coefficients in a suitable basis or dictionary. Nonlinear optimization can then enable recovery of such signals from very few measurements. The amount of data generated by emerging sensing systems has significantly grown. Unfortunately, in many important and emerging applications, the resulting Nyquist rate is so high that we end up with far too many samples. Alternatively, it may simply be too costly, or even physically impossible, to build devices capable of acquiring samples at the necessary rate. Thus, despite extraordinary advances in computational power, the acquisition and processing of signals in application areas such as imaging, video, medical imaging, remote surveillance, spectroscopy, and genomic data analysis continues to pose a tremendous challenge. To address the computational challenges involved in dealing with such high-dimensional data, we often depend on compression, which aims at finding the most concise representation of a signal that is able to achieve a target level of acceptable distortion. One of the most popular techniques for signal compression is known as transform coding, and typically relies on finding a basis or frame that provides sparse or compressible representations for signals in a class of interest. By a sparse representation, we mean that for a signal of length n, we can represent it with $k \ll n$ nonzero coefficients. By a compressible representation, we mean that the signal is well-approximated by a signal with only k nonzero coefficients. Both sparse and compressible signals can be represented with high fidelity by preserving only the values and locations of the largest coefficients of the signal. This process is called sparse approximation, and forms the foundation of transform coding schemes which exploit signal sparsity and compressibility, including the JPEG, JPEG2000, MPEG, and MP3 standards.

Leveraging the concept of transform coding, compressive sensing (CS) has emerged as a new framework for signal acquisition and sensor design. CS enables a potentially large reduction in the sampling and computation costs for sensing signals that have a sparse or compressible representation. While the Nyquist-Shannon sampling theorem states that a certain minimum number of samples is required in order to perfectly capture an arbitrary bandlimited signal, we can vastly reduce the number of measurements that need to be stored when the signal is sparse in a known basis. Consequently, when sensing sparse signals we might be able to do better than suggested by classical results. This is the fundamental idea behind CS: rather than first sampling at a high rate and then compressing the sampled data, we would like to find ways to directly sense the data in a compressed form i.e., at a lower sampling rate. The field of CS grew out of the work of Candes, Romberg, and Tao and of Donoho, who showed that a finite-dimensional signal having a sparse or compressible representation can be recovered from a small set of linear measurements [5]. The design of these measurement schemes and their extensions to practical data models and acquisition systems are central challenges in the field of CS.

Mathematically a signal x is called k-sparse if

$$\|x\|_0 \leq k. \tag{1.1}$$

And $\|x\|_0$ (norm-0) is defined a number 0f non-zero elements of x. Few real signals are exactly sparse. To quantify the compressibility of x by a k-sparse signal, we consider the following quantity:

$$\sigma_k(x)_p = \min_{\hat{x}\in\Sigma_k} \|x - \hat{x}\|_p \tag{1.2}$$

Σ_k is set of all k-sparse signals. In other words $\sigma_k(x)_p$ is the minimum approximation error when x is represented by a k-sparse signal. The objective is to estimate x based on some measurements, while restricting our estimate to be k-sparse and the estimation error is as close as possible to $\sigma_k(x)_p$. Consider the following measurement system:

$$y = \Phi x + e \quad , y \in \mathbb{R}^n , x \in \mathbb{R}^m , m < n$$

$$y\text{: Measurements}$$

$$x\text{: Sparse or compressible signal (that we want to estimate)} \tag{1.3}$$

$$\Phi\text{: Sensing matrix}$$

$$e\text{ : Measurement noise}$$

Suppose x is k-sparse with respect to certain basis Ψ such that:

$$x = \Psi\bar{x} \quad , x \in \Sigma_k \tag{1.4}$$

So we have:

$$y = \Phi(\Psi\bar{x}) + e = (\Phi\Psi)\bar{x} + e = A\bar{x} + e \quad , A \triangleq \Phi\Psi \tag{1.5}$$

The main theoretical question is for a fixed Φ what class of x can be recovered? How should we design Φ such that x can be well recovered from y. Theory of compressive-sensing answer all of these questions by setting some restrictions on sensing matrix. Eldar [6] has explained this theory in more detail.

1.3 Goal of This Work

In this research, we address two problems of conventional CS algorithm: Fixed compression rate and reconstruction error. Adaptive CS is proposed to modify compression rate based on sparsity of input signal. Also an on-chip signal processing have proposed which detects important events in the sensor input data, and uses that information during reconstruction to minimize the reconstruction error. The adaptive CS is used for two different applications of biomedical signals and depth measurement data to show that it can be used for different application. A test-chip was built to implement CS algorithm on chip for biomedical signals.

This dissertation includes five chapters. Chapter one is introduction talking about problem definition and motivation of this work. Chapter two investigates sparsity variance of biomedical signals and based on that proposes a rate-adaptive CS for biomedical signal. Appendix A at the end of this chapter gives more details about the concept of dictionary learning. The third chapter explains use of some statistics from input signal to reduce reconstruction error. A test chip fabricated in TSMC 65nm technology as proof of concept. Chapter four introduces a framework for CS of 3D point cloud data. Appendix C at the end of this chapter introduces concept of depth measurement, its application and some of research done in this area. Chapter five implements idea of adaptive CS for point cloud data. Finally chapter six concludes with a brief summary of findings and includes possible future research in continue of this work.

1.4 References

[1] X. Pu, L. Wan, Y. Sheng, P. Chiang, Y. Qin, and Z. Hong, "A Wireless 8-Channel ECG Biopotential Acquisition System for Dry Electrodes", *IEEE Radio-Frequency Integration Technology Conference*, Nov. 2012.

[2] Derek Chen, Joe Crop, Patrick Chiang, and Gabor Temes, "A 12-Bit 7uW/Channel 1 Khz/Channel Incremental ADC for Biosensor Interface Circuits", *International Symposium on Circuits and Systems (ISCAS)*, May 2012.

[3] Robert Pawlowski, Evgeni Krimer, Joseph Crop, Jacob Postman, Nariman Moezzi-Madani, Mattan Erez, Patrick Chiang, "A 530mV 10-Lane SIMD Processor With Variation Resiliency in 45nm SOI", *International Solid-State Circuits Conference,* Feb. 2012.

[4] J. Cheng, N. Qi, P. Y. Chiang, A. Natarajan, "A 1.3mW 0.6V WBAN-Compatible Sub-Sampling PSK Receiver in 65nm CMOS", *International Solid-State Circuits Conference*, Feb. 2014.

[5] Candès, E.J., & Wakin, M.B., "An Introduction To Compressive Sampling," *IEEE Signal Processing Magazine*, V. 21, March 2008.

[6] Y. C. Eldar, G. Kutyniok, "Compressed Sensing: Theory and Applications", 1st Edition, 2012.

Rate-Adaptive Compressed-Sensing and Sparsity Variance of Biomedical Signals

Vahid Behravan, Neil E. Glover, Rutger Farry,
Patrick Y. Chiang , Mohammed Shoaib

12th IEEE International Conference on Wearable and Implantable Body Sensor Networks (BSN), 2015.

2.Rate-Adaptive Compressive-Sensing and Sparsity Variance of Biomedical Signals

2.1 Abstract

Biomedical signals exhibit substantial variance in their sparsity, preventing conventional a-priori open-loop setting of the compressive sensing (CS) compression factor. In this work, we propose, analyze, and experimentally verify a rate-adaptive compressed-sensing system where the compression factor is modified automatically, based upon the sparsity of the input signal. Experimental results based on an embedded sensor platform exhibit a 16.2% improvement in power consumption for the proposed rate-adaptive CS versus traditional CS with a fixed compression factor. We also demonstrate the potential to improve this number to 24% through the use of an ultra-low power processor in our embedded system.

Keywords: adaptive compressive-sensing; low power wireless sensors; sparsity variance; embedded system

2.2 Introduction

Power consumption is one of the most critical constraints for sensor networks and future Internet-of-Things devices. The most power consuming block is the radio (transmitter), typically responsible for 60-80% of the total sensor's energy usage [7]. A known method to reduce the radio power consumption is transmitter duty cycling, sending data in quick bursts before rapidly powering down [8]. To obtain the largest benefit from transmitter duty cycling, the amount of sensor data transmitted with the radio must be reduced, such as using data compression.

Signal specific compression methods do exist (*e.g.* JPEG) that very efficiently decrease the amount of data required to recover the original signal [9]; However they typically are specific and cannot be applied to multiple sensor data types. Due to these limitations, a compression method generalizable to many different types of data is highly desirable.

Compressive-Sensing (CS) is a subsampling method that requires no a-priori information about the input signal as long as the input shows a property called sparsity (explained in detail below). Figure 2-1 shows both traditional and compressive- sensing systems and their energy consumption specifications. The complexity of the receiver in a CS system is higher than its traditional counterpart; however the receiver typically does not have as tight of a power

consumption constraint. On the other hand, the CS sensor node will exhibit significant energy savings in the transmitter (radio), outweighing any additional energy consumed by implementing the compression block.

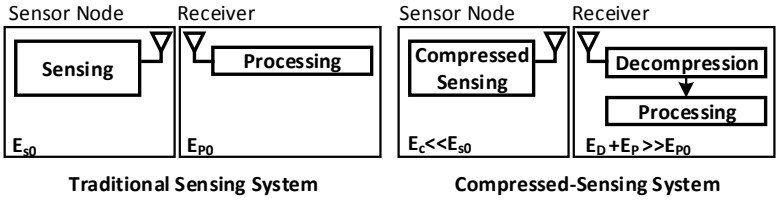

Figure 2-1: A compressive-sensing system vs. a traditional sensing system

Various methods have been previously proposed that implement a CS TX framework [10]. Unfortunately, all previous CS works assume a fixed and predetermined input signal sparsity. This assumption implies that the compression factor is statically fixed, and is not a function of a varying input signal; however, there is a trade-off between input sparsity and optimum compression factor [11]. An input signal with time-varying sparsity must adjust its compression factor continuously in order to achieve the lowest error rate during reconstruction. Using the signal sparsity to determine the sensor's optimal compression factor is critical if the sparsity of the signal varies significantly over time, across different sensor types. To our knowledge, there is no prior work that has studied the sparsity variance for several different biomedical signals.

In this paper, we first conduct a feasibility study on adaptive compressive sensing, investigating the sparsity variance of electrocardiogram (ECG), electroencephalogram (EEG) and electromyogram (EMG), using the physionet database [12]. Next, we propose an adaptive CS framework, adjusting the compression rate based upon the input signal's sparsity on-the-fly. Finally, we implement this rate-adaptive CS using a miniature, low-power wearable platform that captures ECG signals under varying sparsity conditions, calculate the sparsity variance, and adapt the CS rate.

This paper is organized as follows. Section II covers the theory of compressive-sensing, reconstruction and key system requirements. The requirements of an adaptive compressive-sensing system are then described in Section III. The miniature wearable system developed to capture test signals and implement adaptive CS framework is then outlined in Section IV, and conclusions are summarized in Section V.

2.3 CS Framework

2.3.1 Compressive-Sensing Background

Compression using CS is achieved using a simple matrix multiplication, where an uncompressed input vector $\boldsymbol{f}$ of size N multiplied by a measurement matrix $\boldsymbol{\Phi}$ of size M-by-N that produces a measurement vector $\boldsymbol{y}$ of size M. $\boldsymbol{\Phi}$ is a matrix of random numbers (*e.g.*, Bernoulli, Gaussian, uniform, etc.), such that $\boldsymbol{y}$ is a vector of random linear projections of $\boldsymbol{f}$ on $\boldsymbol{\Phi}$. CS can compress bio-signals with compression factors as large as 10x, which reduces the amount of data transmitted and therefore the total power dissipation of the sensor node by a similar factor [14]. To reconstruct the original signal $\boldsymbol{f}$ from the measured signal $\boldsymbol{y}$ in the receiver, we need to solve an underdetermined set of equations where the number of equations is much less than the number of unknown variables. In general, there is not a unique solution for these types of equations. However, CS theory shows that if the signal $\boldsymbol{f}$ is sparse in any basis (*e.g.* fourier transform, wavelet transform), then there are optimized methods to reconstruct the original signal with minimum error based on convex optimization [11]. In other words, if there is a transformation matrix $\boldsymbol{\Psi}$ such that $\boldsymbol{f} = \boldsymbol{\Psi x}$ (and therefore $\boldsymbol{y} = \boldsymbol{\Phi f} = \boldsymbol{\Phi\Psi x}$) and $\mathbf{x}$ is sparse, then reconstruction is possible. Figure 2-2 summarizes the complete CS framework [15].

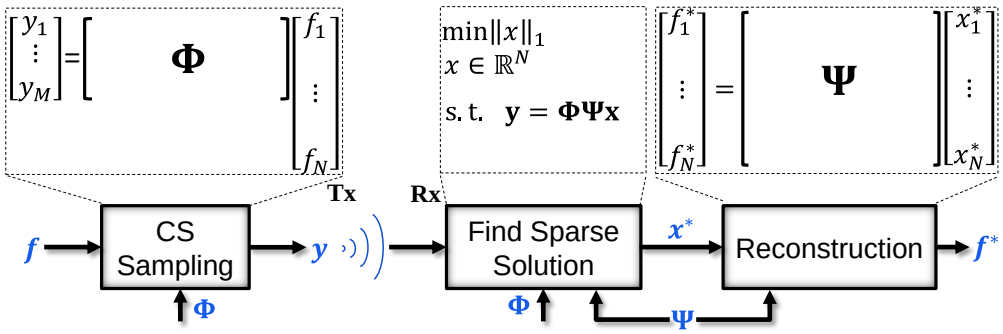

Figure 2-2: Complete CS framework

2.3.2 Sparsity Number

In theory, sparsity of a vector $\mathbf{x}$ is defined as the number of non-zero values of $\mathbf{x}$ (this is called the norm-0 of $\mathbf{x}$ and represented as $\|\boldsymbol{x}\|_0$). It is shown that if the number of measurements (M) is approximately $\mu\|\boldsymbol{x}\|_0 \log(N)$, an optimum reconstruction can be achieved with high probability where μ is a constant related to $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ [11][12]. The reconstruction process in the receiver, as shown in Figure 2-2, consists of two steps. The first step finds an approximation of the sparse vector $\mathbf{x}$ by minimizing norm-1 of $\mathbf{x}$, subject to the main equation $\boldsymbol{y} = \boldsymbol{\Phi\Psi x}$ (norm-1 also known as l1-norm is defined as the sum of absolute values of vector $\mathbf{x}$ and is represented by $\|\boldsymbol{x}\|_1$). The problem of minimizing the l1-norm has been shown to be solvable efficiently and requires only a small set of measurements (M<<N) to enable perfect recovery.

After finding the optimized approximation of $\mathbf{x}$ ($\mathbf{x}^*$), the second step reconstructs the original signal using matrix multiplication $\boldsymbol{f}^* = \boldsymbol{\Psi}\mathbf{x}$.

Fortunately, some biomedical signals have sparse representation in either gabor or wavelet domains [17][18]. In general, any type of sensor data can be represented in its sparse form by using a method called dictionary learning. Based on the ideal definition of sparsity, it should be very easy to find the sparsity of a signal simply by counting the number of non-zero elements of the signal in the sparse domain. Unfortunately, for all real signals the representation of the input signal in sparse domain has non-zero elements. There are a few large values with other elements that are very small but not zero. To address this issue, Lopes in [19] introduced a new metric called *sparsity number*, as defined in Eq. (1) below, which is the lower bound of the ideal definition of sparsity:

$$sx = \frac{\|x\|_1^2}{\|x\|_2^2} = \frac{(\sum|x_i|)^2}{\sum x_i^2} \tag{2.1}$$

To find the sparsity number of a signal we need to first transfer that signal to the sparse domain. For some signals the sparse domain is known; however, for other signals there is not a specific domain for this purpose and we must use dictionary learning to train a dictionary to transfer the signal $\boldsymbol{f}$ to its sparse representation **x**.

2.3.3 Dictionary Learning

While ECG signal has known domain of sparsity, most other types of sensor signals do not have well-known domains where the signal is sparse. To extend the application of CS to different type of signals, it is necessary to have a general method for converting signals to a sparse representation. Fortunately, a method called dictionary learning exists for this purpose. Using a dictionary **D**, a vector $\boldsymbol{f}$ (that is input signal) is represented as approximation of a linear combination of a few *atoms* where *atoms* are columns of **D** (such that the dictionary in fact is a matrix). To generate this dictionary, a training process called dictionary learning is required. Dictionary learning is the problem of finding **D** such that the approximation of many known vectors (the training set) are as good as possible given a sparseness criterion on the coefficients (*i.e*. allowing only a small number of non-zero coefficients for each approximation)[20]. Different methods are available for dictionary learning, such as KSVD [20] and MOD[21]. In this paper, we use the KSVD method to check the sparsity variance of EMG and EEG signals. Having **D** any new signal in the same family of training set can be represented in the sparse domain by solving Eq. (2), where γ controls the sparseness of the answer:

$$x^* = arg\min_x \|x\|_p + \gamma\|f - Dx\|_2^2 \qquad p \in \{0,1\} \tag{2.2}$$

2.4 Sparsity-Aware Compressive Sensing

The number of measurements (M), which also defines the compression factor (M/N), is related to the sparsity level of the input signal. A trade-off exists between the compression factor and the accuracy of the reconstructed signal. Increasing the compression factor creates more errors in the reconstructed signal, while lowering the compression factor increases the amount of transmitted data without any benefit to the reconstruction. Figure 2-3 shows the output of a sparse optimization block ($\mathbf{x}^*$) for a CS system with a fixed compression factor of 16 for two different input signals. The first signal is a single tone (that is sparse in the Fourier domain) and the second signal is the summation of 5 sine waves with different frequencies. It is clear that by changing the statistics of the input signal we will see a large difference between the reconstructed sparse signal ($\mathbf{x}^*$) and actual sparse representation of the input signal in the sparse domain. This results in a large error in the output of the reconstruction algorithm.

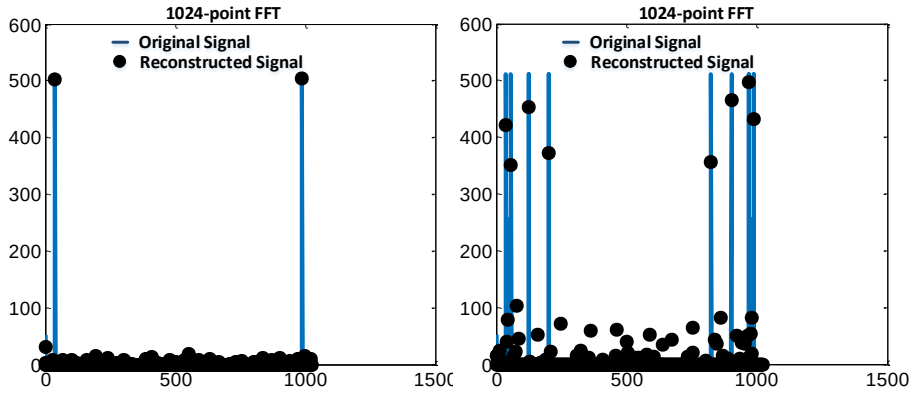

Figure 2-3: Effect of sparsity on reconstruction error for two different signals

Obtaining updated information about the sparsity of the input signal will help us to adjust the compression factor to the optimum value. Using sparsity number as a key to find the optimum compression factor is only beneficial if the sparsity of the input signal changes sufficiently over time and for different sensor data types. This means that sparsity variance is a key feature for adaptive compressive sensing.

2.4.1 Sparsity Variance

If sparsity of an input signal does not change significantly over time, we can use a fixed compression factor that corresponds to the type of input signal. To the best of our knowledge, we have found no prior works have investigated and quantified the sparsity variance for different biomedical data types. In this work, we analyze the sparsity variance for three

different types of clinical biomedical signals (ECG, EEG and EMG), using the nsrdb, mitdb, chbmit and drivedb databases of physionet [13]. We also obtained the sparsity variance of ECG data captured by own embedded system, platform. In each case, several signals divided in blocks of 256 samples and then transferred to sparse domain to calculate the sparsity numbers. Finally we plot the histogram of the sparsity number and calculate mean and standard deviation of distribution. Figure 2-4 shows the simulation results, showing that the sparsity changes drastically and that a fixed compression factor is not a good choice.

2.4.2 System Models for Sparsity-Aware Compressive-Sensing

Figure 2-5 shows two possible methods for implementing rate-adaptive compressive-sensing, using the input signal sparsity to adjust the compression factor. In the first system (Method 1), the receiver calculates the sparsity and transmits it to the sensor node of the system. The sensor then adjusts its compression factor based on this information. Because the receiver already has the sparse representation of signal, this system can easily implement CS rate adaption in the sensor node; however, one limitation with this system is the round trip delay, or the latency required to calculate the sparsity in the receiver and the transmission delay to send the result back to the sensor. If this latency is large, the sensor node must buffer recently acquired data until it has received the most up-to-date estimate of sparsity from the receiver. Another problem is that the calculation of sparsity is performed on the current data block; however this result is then used to compress the next data block, which may have a different sparsity

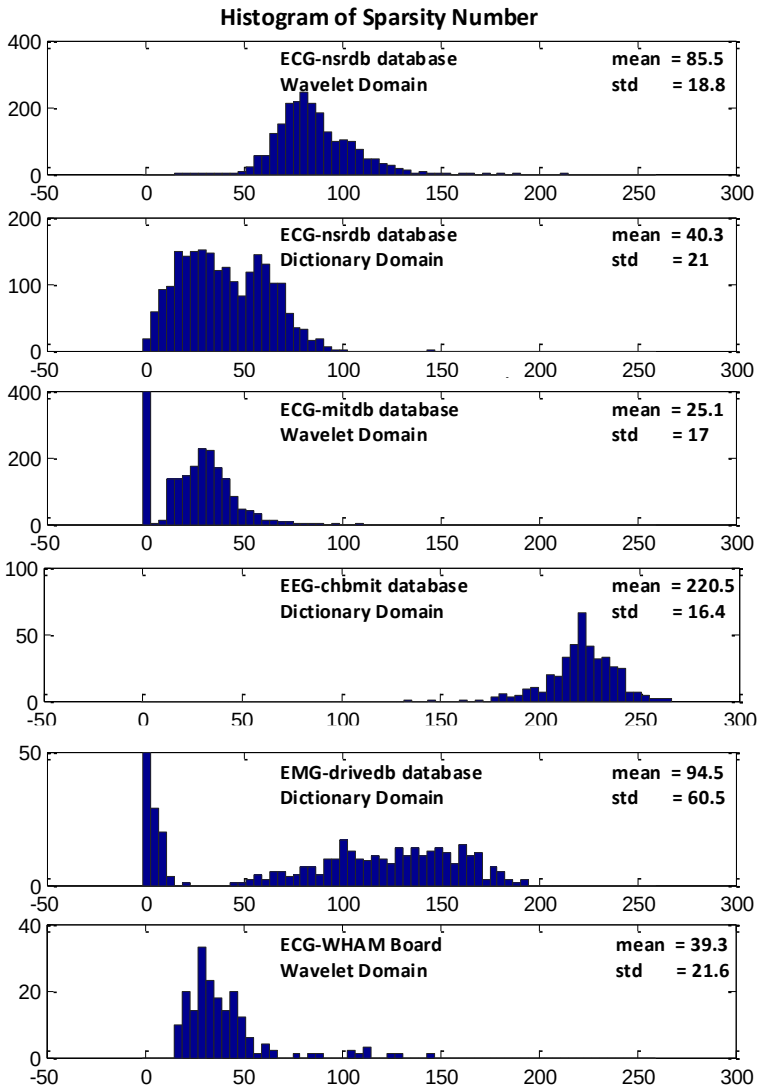

Figure 2-4: Sparsity variance for different type of biomedical signals

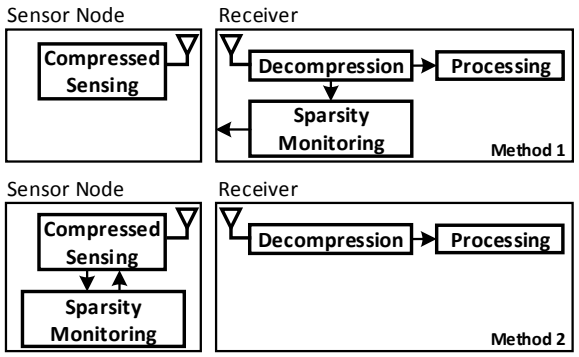

Figure 2-5: Possible methods for adaptive CS system implementation

The second system (Method 2) calculates the sparsity of the input data on the sensor node of the system. This eliminates possible problems associated with the round trip delay, and also allows for the calculated sparsity of each data block to be used in the compression of that same block. Unfortunately, this method requires additional power to operate the sparsity monitoring

block, which may not be available on the energy-constrained sensor node. In addition, this system has no direct access to the sparse representation of the input signal, and therefore requires a sparsity estimation block that obtains the sparsity of the input data directly from the time domain signal. To our knowledge, this is currently an unsolved problem and hence, we evaluate the feasibility and performance of only Method-1 here.

2.5 Experimental Results

To analyze the performance of this proposed rate-adaptive CS system, we developed an experimental wireless monitoring platform, called the Wearable Health and Activity Monitor (WHAM). We implemented the compression on the WHAM sensor node, while the corresponding reconstruction algorithm is performed in an iOS iPhone/Mac application used as the receiver device. We implemented the GPSR algorithm [22] for reconstruction, which is faster than other reconstruction algorithms like LASSO[23]. We particularly were interested in investigating the energy consumption of the wearable sensor portion of the system with and without the use of adaptive data compression. Minimizing the delay associated with the reconstruction algorithm is critical because it directly affects the size of the buffer needed at the sensor node, before the receiver can calculate and send back the sparsity number needed to compress the next data block on the sensor.

2.5.1 WHAM System Overview

The WHAM platform is a low-cost, reconfigurable, wearable, wireless platform designed specifically to collect, process and display data related to the user's activity and health. Utilizing wireless Smartphone connectivity using Bluetooth, the WHAM was designed to be operated by a wide variety of users. The complete system includes a functional custom hardware, a fully functioning iOS application, which can control and display data from the sensor from a variety of Apple products. The system weighs just 9.5 grams and is small enough to be easily stitched into clothing. Using a small CR2032 coin cell battery, the WHAM can provide continuous streaming ECG data collection for over 100 hours. The system currently consists of 3 physical components: The main board, the battery, and the flexible ECG board. The dime-sized main board consists of a stack of two miniature two-layer PCBs which contain the SoC, accelerometer, printed circuit board antenna, and other necessary passive components. The flexible ECG board contains the electrodes and instrumentation amplifier necessary for ECG data acquisition. Because of the flexibility and simplicity of its development platform, iOS was chosen as the operating system. The iOS application plots the real-time ECG graph

and Beats per Minute (BPM) calculation. Figure 2-6 shows the WHAM main sensor board, and top and bottom sides of the flexible electrode ECG patch.

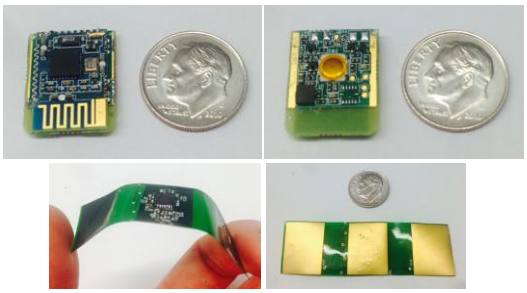

Figure 2-6: WHAM main board (top), Flexible electrode patch (bottom).

2.5.2 Implementation Methodology

As illustrated in Figure 2-2, CS compression is the product of a block of N input samples (here N=256) to a random matrix. We implemented this matrix multiplication at the sensor node. For simplicity, the random matrix $\mathbf{\Phi}$ has a fixed size of 192×256, so it always generates 192 measured values from every 256 input samples. Based on the feedback from the receiver, the sensor only transmits the required number of measured samples. For example, if M=128 then only the first 128 samples will be transmitted. This is equivalent to multiplying the input vector with the first 128 rows of $\mathbf{\Phi}$. In the receiver we also use the first 128 rows of $\mathbf{\Phi}$ for reconstruction. The receiver constantly checks the sparsity number of the data it is receiving. If the sparsity of the input signal changes by more than 50% when compared to the previous block of data, it will change the value of M. Even for a normal ECG signal the sparsity number may change in a range of 40%. Analysis of the data sets investigated in this work found that a 50% variation in sparsity number is a significant change in the input signal (*e.g.* ECG abnormality such as atrial fibrillation).

For our wireless radio protocol, we adopted Bluetooth Low Energy (BLE) as it is one of the most commonly used low-power sensor radio standards, and heavily utilizes duty cycling. A BLE connection between two devices is made every T seconds where the devices exchange some data. After this exchange both devices disconnect and power down their radios for a period of time. The receiver portion of the system (*e.g.* the App running on the iPhone) is able to set T. Unfortunately, the system we developed for this work has limitations on the possible values of T and the number of samples that each BLE transmission is able to send. For this reason we only use a limited number of values for M, as shown in Table 2-1 below, where CF stands for the compression factor.

Table 2-1: Possible options for choosing M

T	N=256, ADC Sample Rate=200 sample/sec								
	1 sample per BLE Transm.			2 samples per BLE Transm.			3 samples per BLE Transm.		
	Tx Rate	F	M	Tx Rate	F	M	Tx Rate	CF	M
20ms	50	4	64	100	2	128	150	1.33	**192**
30ms	33	6	42	66	3	84	99	2	**128**
40ms	25	8	32	50	4	64	75	2.66	**96**
50ms	16.6	10	25	40	5	50	60	3.33	75

In our implementation of adaptive CS, we transmit three samples for each BLE transmission such that the possible values of M are 192, 128, 96, 75 and 63. Initially the system starts with a default value of M (*e.g.* M=96 that is equal to a compression factor of 256/96=2.7). During normal operation, anytime the receiver detects a 50% change in the sparsity number it adjusts T appropriately such that the sensor automatically switches to the next higher or lower M value.

2.5.3 Measurments Results

To calculate the average power consumption of the system, we used a 1.3Ω resistor in series with the battery to measure the voltage drop across it. Figure 2-7 contains the system setup for power measurements as well as real-time signal acquisition. Transmission of data occurs in very short time durations but with high current consumption.

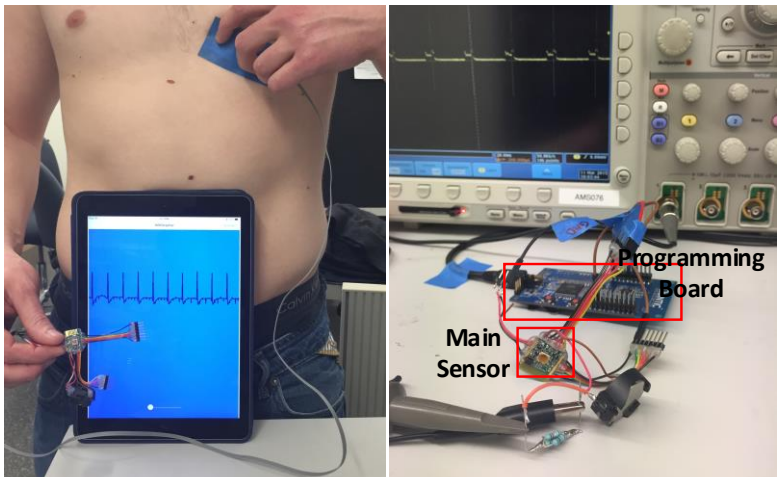

Figure 2-7: System setup for power measurement and signal acquisition

Our measurements show that the average current changes negligibly when sending 1, 2 or 3 samples with each BLE transmission. For this reason we decided to control the CS compression factor by controlling the turn-on period (T). To demonstrate and analyze the performance of

adaptive CS system, we selected two sections of an ECG signal captured by the WHAM system that showed a significant change in their sparsity number. After this signal is stored in the sensor, the data is then compressed and transmitted to the receiver, where it is then reconstructed and analyzed. Analysis of the reconstructed data in the receiver provides a new estimate of the sparsity number, where the sparsity number is used to set the new transaction time (T) of the BLE connection. The reconstructed signal is then appropriately scaled and analyzed using MATLAB in order to compare it with the original signal to calculate the reconstruction error. The reconstruction error is determined using the equation for Signal to Error Ratio (SER), as defined in Eq. (3) where $\boldsymbol{f}$, $\boldsymbol{f}^*$ are the original and reconstructed signals, respectively:

$$SER = 10\log_{10}\left(\frac{\|f\|_2^2}{\|f - f^*\|_2^2}\right) \tag{2.3}$$

Figure 2-8 shows the original and reconstructed signals. The first two blocks are normal with low sparsity where the system transmits 3 samples every 40ms, equivalent to a compression factor of 2.7. Based on the information in Table 2-1, in order to transmit these two blocks, M must be set to 96. In the 3rd block of ECG, motion artifact noise is observed. Due to this noise, its sparsity number changes by more than 50% when compared with ECG block 2. The receiver senses this change and adjusts the BLE transmission time to 30ms, equivalent to a compression factor of 2 (M=128), to compensate for this sparsity change. The sensor then applies this lower compression factor, obtaining a more accurate reconstruction in the receiver. Note that this system always exhibits a one block delay, as the 3rd block is compressed with the compression factor calculated from the 2nd block. Because of this the system experiences some error upon reconstruction.

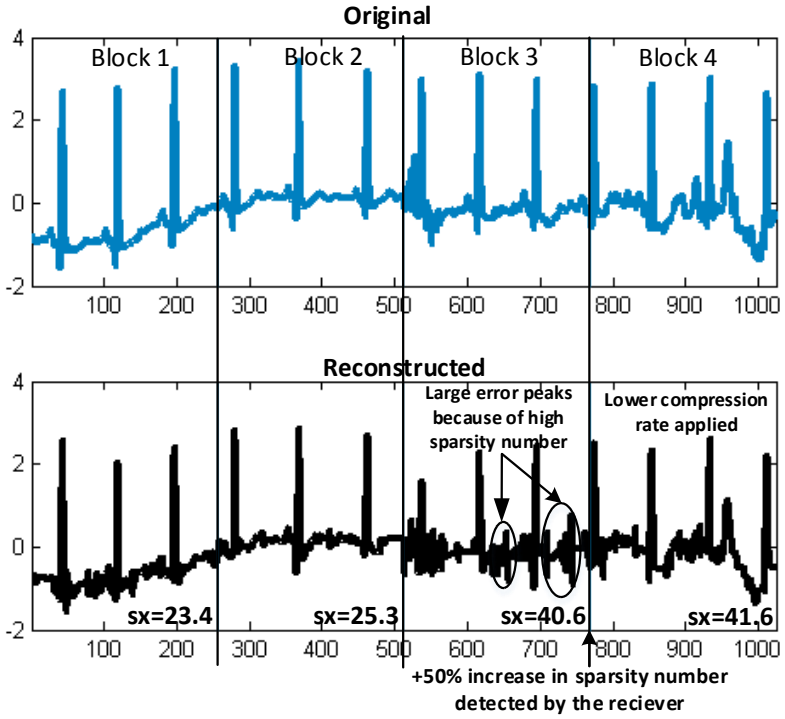

Figure 2-8: Original vs. reconstructed signal for adaptive CS system

Table 2-2 summarizes the measurement results. First we measure the average power consumption in 3 modes: Adaptive CS versus traditional CS with fixed compression factors of 2 and 2.7. We calculate the average power consumption of the adaptive CS system assuming 50% of the time the system experiences signals with a high sparsity number (*e.g.* noisy or abnormal). These results show that when targeting more than 10dB of SER, adaptive CS exhibits the same performance as a fixed compression rate system (with M=128) but consumes 14% less power. If the target is a minimum SER of 16dB for all input blocks, we will then need a fixed value of M=192 (Table 2-3). In this case, using adaptive CS that switches between M=192 and M=128 gives us the same performance but with 16.2% less power consumption. The average power consumption of the adaptive system is based on the assumption that the signal input is a normal signal with low sparsity number for a long time duration, until it abruptly changes to a signal with high sparsity number (*e.g.* because of abnormalities or motion artifact) and remains this way for a long time period. In this case, we can approximate the average power of the adaptive system using the formula in Eq. (4), where $P_{avg-fixed1}$ and $P_{avg-fixed2}$ are the average power of the CS system with old and new compression factors:

$$P_{avg-adaptive} = \frac{1}{2}(P_{avg-fixed1} + P_{avg-fixed2}) \tag{2.4}$$

.

Table 2-2: Performance of the Adaptive CS System for SER_{min}=10dB

CS method	**SER(dB)**				
	Block1	**Block2**	**Block3**	**Block4**	**Average Power**
Adaptive	13.2	12.5	4.9	**10.5**	**2.5mW**
Fixed (M=128)	14.9	14.7	10.2	**10.5**	2.9mW
Fixed (M=96)	13.2	12.5	4.9	**6**	2.1mW

Table 2-3: Performance of the Adaptive CS System for SER_{min}=16dB

CS method	**SER(dB)**				
	Block1	**Block2**	**Block3**	**Block4**	**Average Power**
Adaptive	14.9	14.7	10.2	**16.3**	**3.6mW**
Fixed (M=192)	22	19	16.1	**16.3**	4.3mW
Fixed (M=128)	14.9	14.7	10.2	**10.5**	2.9mW

Figure 2-9 shows the measured average power consumption of the sensor for different transmission rates of BLE. Unfortunately, the results show that the power consumption of the embedded processor dominates after some point, such that decreasing transmission rate (increasing T) does not improve power consumption proportionally. Using an ultra-low power processor we could expect 1.5mW power consumption for T=40ms giving average power of 2.2mW for the adaptive CS system, equivalent to a 24% improvement when compared with a fixed CS system with M=128.

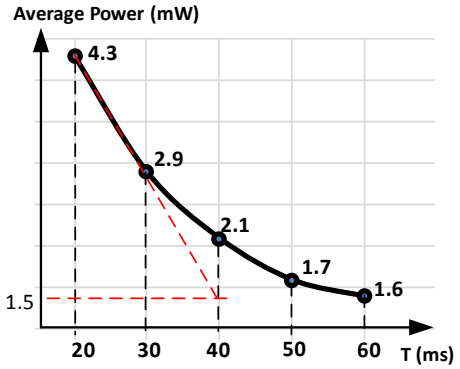

Figure 2-9: Measured average power for different BLE transmission rates

Finally, we measured the execution time of the reconstruction algorithm on the receiver (a MacBook Air), where the worst case reconstruction latency was found to be 24 ms, or less than the time required by the ADC to generate 5 samples.

2.6 Conclusion

In this paper we propose an adaptive compressive-sensing system that enables real-time adjustment of the compression factor, depending on the variance in the sparsity number of the input signal. We also investigated the sparsity variance of different biomedical signals to provide justification for implementing an adaptive compressive-sensing system. We then validated our proposed system through experiments conducted on a custom embedded hardware. We showed that our adaptive compressive-sensing system can reduce power consumption by 16.2%, without any major change on the sensor node. Despite this initial success, the criteria for which the adaptive system should change the compression factor has not been fully explored and is the subject of future work Furthermore, the relationship between sparsity and the desired compression factor is also still unknown for real biomedical signals, and requires future research.

2.7 Appendix A: Dictionary Learning

This appendix adopted from [21]. Dictionary learning is a topic in signal processing area. Dictionary is usually used for sparse representation or approximation of signals. In sparse representation a vector x is represented or approximated as a linear combination of a few atoms of dictionary D. Each column of D is called atom.

$$x_a = D.w \ , \qquad D \in \mathbb{R}^{N \times K} \tag{2.5}$$

Where w is a vector containing mostly zero coefficients. Dictionary is the problem of finding D such that the approximation of many vectors (the training set) are as good as possible given a sparseness criterion on the coefficients (*i.e.* allowing only a small number of non-zero coefficients for each approximation). Here is an example:

$$\begin{gathered} x_a = D.w = w_1 d_1 + w_4 d_4 + w_7 d_7 \\ x = x_a + r \ \ \therefore \ \ r = x - x_a = x - D.w \end{gathered} \tag{2.6}$$

Where w_1, w_4, w_7 are respectively first, fourth and seventh element of vector w and d_1, d_4, d_7 are respectively first, fourth and seventh column (atom) of dictionary. If most of entries in w are zero, this is a sparse approximation. We can find the optimum w by solving the following optimization problem:

$$w_{opt} = \arg\min_{w} \left(\|w\|_p + \gamma \|x - D.w\|_2^2 \right) \ , \quad p \in \{0,1\} \tag{2.7}$$

The process of dictionary learning starts with collection (L) of training vectors (training set) collected in a matrix $X \in \mathbb{R}^{N \times L}$:

$$\{D_{opt}, W_{opt}\} = \arg\min_{D,W} \sum_{l=1}^{L} \left(\|w_l\|_p + \gamma \|X - D.W\|_2^2 \right) \ , p \in \{0,1\}$$
$$W_{opt} \in \mathbb{R}^{K \times L}, D_{opt} \in \mathbb{R}^{N \times K} \quad (2.8)$$

Different methods are available for dictionary learning such as MOD, K-SVD, RSL-DLA. K-SVD is an iterative method including two steps in each iteration:

1- Keeping D fixed and find W. This gives L independent problems.
2- Keeping only non-zero positions in W fixed find D and W using Singular Value Decomposition (SVD) decomposition.

2.8 References

[7] F. Chen, "Energy-efficient Wireless Sensors: Fewer Bits, More MEMS," PhD Thesis, Department of Electrical Eng. And Comp. Scince, Massachusetts Inst. Technol., Cambridge, MA, USA, Sep. 2011.

[8] J. Cheng; et al., "A near-threshold, multi-node, wireless body area sensor network powered by RF energy harvesting," Custom Integrated Circuits Conference (CICC), 2012 IEEE , vol., no., pp.1,4, 9-12 Sept. 2012

[9] S.K. Mukhopadhyay, M. Mitra, "An ECG data compression method via R-Peak detection and ASCII Character Encoding," Computer, Communication and Electrical Technology (ICCCET), 2011 International Conference on , vol., no., pp.136,141, 18-19 March 2011.

[10] F. Chen, A.P. Chandrakasan, V. Stojanovic, "A signal-agnostic compressed sensing acquisition system for wireless and implantable sensors," Custom Integrated Circuits Conference (CICC), 2010 IEEE , vol., no., pp.1,4, 19-22 Sept. 2010

[11] D. Donobo, "Compressed Sensing," Trans. on Information Theory, 2006.

[12] Candes, E.J.; Tao, T., "Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies?," *Information Theory, IEEE Transactions on* , vol.52, no.12, pp.5406,5425, Dec. 2006

[13] A. L. Goldberger, et al., "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals,[Circulation Electronic Pages; http://circ.ahajournals.org/cgi/content/full/101/23/e215], 2000.

[14] http://www.physionet.org/physiobank/database/

[15] D. Gangopadhyay, et al., "Compressed Sensing Analog Front-End for Bio-Sensor Applications," Solid-State Circuits, IEEE Journal of , vol.49, no.2, pp.426,438, Feb. 2014.

[16] O. Abari, F. Chen, F. Lim, V. Stojanovic, “Performance trade-offs and design limitations of analog-to-information converter front-ends,” Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on , March 2012.

[17] S. Miaou and S. Chao, “Wavelet-Based Lossy-to-Lossless ECG Compression in a Unified Vector Quantization Framework,” IEEE Transactions on Biomedical Engineering , vol. 52, pp. 539-543, 2005.

[18] S. Aviyente, “Compressed Sensing Framework for EEG Compression,” IEEE 14th Workshop on Statistical Signal Processing , 2007.

[19] M. Lopes, “Estimating Unknown Sparsity in Compressed Sensing,” International Conference on Machine Learning (ICML), 2013.

[20] M. Aharon, M. Elad, A. Bruckstein, “K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation,” Signal Processing, IEEE Transactions on , vol.54, no.11, pp.4311,4322, Nov. 2006.

[21] http://www.ux.uis.no/~karlsk/dle/

[22] http://www.lx.it.pt/~mtf/GPSR/

[23] http://statweb.stanford.edu/~tibs/lasso.html

A Compressed-Sensing Sensor-on-Chip IncorporatingStatistics Collection to Improve Reconstruction Performanc

Vahid Behravan, Shuo Li, Neil E. Glover, Chia-Hung Chen, Mohammed Shoaib, Gabor C. Temes and Patrick Y. Chiang

IEEE Custom Integrated Circuits Conference (CICC), 2015.

3.A Compressive-Sensing Sensor-on-Chip Incorporating Statistics Collection to Improve Reconstruction Performance

3.1 Abstract

Reconstructing signals accurately is a critical aspect of compressive sensing. We propose a compressive-sensing sensor-on-chip that compresses and also extracts key statistics of the input signal at sampling time. These statistics can be used at the receiver to significantly improve the accuracy of reconstruction. When compared against a conventional compressive-sensing system, our experimental measured results demonstrate an improvement of as much as 9-18 dB in the signal-to-error (SER) of the reconstructed signal, depending on input data type and compression factor.

Keywords: compressive sensing, sensor-on-chip, reconstruction error, statistics collection.

3.2 Introduction

Power consumption is one among the most critical constraints for sensor networks and future Internet-of-Things devices. The most power-consuming block in a sensor system-on-chip is the radio (transmitter), typically responsible for more than 60-80% of the total sensor's energy usage [24]. Thus, one direct way to reduce the radio's energy consumption is to minimize the amount of transmitted data, such as by using data compression. Compressive sensing is an attractive way of compressing data since it requires no prior knowledge about the input signal, as long as the signal exhibits a property called sparsity. It is broadly applicable because many types of sensor data exhibit sparsity, including biomedical signals [25].

Although compressive sensing is a signal-agnostic method of compression, incorporating additional statistics about the input signal (along with its sparsity) can provide more information that can aid reconstruction algorithms at the receiver. Hence, such statistics collection (SC) can improve the accuracy of signal reconstruction. However, to realize systems based on this principle, sensor nodes will need to transmit additional data (statistics) to the receiver, which may increase transmission-energy costs.

In the literature, various systems have previously implemented the compressive-sensing framework [26][27]. However, all of them employ blind reconstruction with no SC. They do not incorporate any extra information about the input signal to aid reconstruction, resulting in less-than-optimal SERs at the receiver.

In this paper, we implement for the first time a compressive-sensing framework whose reconstruction is aided by statistics collection. We demonstrate how to fuse sensor data and statistics information together to improve the signal-to-error ratio (SER) of reconstruction by as much as 18 dB. Furthermore, using various low-power design techniques, we show that for an ECG data input, SC can be performed with just 0.4µW/12.6µW (dynamic/static power) at a sampling rate of 96 kHz (an overhead of <1% of the communication energy assuming a 1mW radio).

The paper is organized as follows. In Section II, we provide background on compressive sensing, reconstruction, and the design of the proposed SC approach. We then describe the micro-architectural details of the various sub-blocks of the sensor-on-chip (SoC) in Section III. We present the digital circuit implementation of the SoC in Section IV, measurement results in Section V, and conclusions in Section VI.

3.3 System Overview

3.3.1 Background on Compressive Sensing

In compressive sensing, compression of digitized data is performed using matrix multiplications. In particular, an uncompressed input vector $\boldsymbol{f}$ of size N is multiplied by a measurement matrix $\boldsymbol{\Phi}$ of size M×N, producing a measurement vector $\boldsymbol{y}$ of size M. Since $\boldsymbol{\Phi}$ is typically a matrix of random numbers, $\boldsymbol{y}$ is a vector of random linear projections of $\boldsymbol{f}$ onto $\boldsymbol{\Phi}$. To reconstruct the original signal $\boldsymbol{f}$ from the measured signal $\boldsymbol{y}$, we need to solve a set of equations, where the number of equations is much less than the number of unknown variables. In general, there is no unique solution for these types of equations. However, the theory of compressive sensing [5] shows that if the signal $\boldsymbol{f}$ is sparse in any basis $\boldsymbol{\Psi}$ (*e.g.* time, Fourier, wavelet), then there exist known techniques to reconstruct the original signal with minimum error using convex optimization 0. In other words, if there is a transformation matrix $\boldsymbol{\Psi}$ such that $\boldsymbol{f} = \boldsymbol{\Psi}\mathbf{x}$ (and therefore $\boldsymbol{y} = \boldsymbol{\Phi f} = \boldsymbol{\Phi\Psi}\mathbf{x}$) and $\mathbf{x}$ is sparse, then reconstruction is possible. Since reconstruction is complex, in typical sensing systems, it is usually performed at an energy-unconstrained receiver, such as in a cellphone or the cloud.

3.3.2 SC-Assisted Reconstruction

Various algorithms are available for reconstruction [28]. Most of them attempt to minimize the l1-norm of $\mathbf{x}$ using an iterative approach. The gradient-projection for sparse reconstruction (GPSR) algorithm is one such algorithm that minimizes the l1-norm and optimization error simultaneously, as shown below [30]:

$$x^* = arg\min_x \tau\|\boldsymbol{x}\|_1 + 0.5\|\boldsymbol{y} - \boldsymbol{\Phi x}\|_2^2 \quad (3.1)$$

where τ is a non-negative real parameter. In the above formulation, note that τ can also be a column vector with the same number of non-negative entries as **x**. In this case, the objective function then weighs each element of **x** differently such that the following relationship holds true:

$$x^* = arg\min_x \tau^T abs(x) + 0.5\|y - \boldsymbol{\Phi} x\|_2^2 \quad (3.2)$$

In this paper, our system first extracts some statistics about the sensor's input signal (*i.e.* timing location of the R-peaks in the ECG). Based on these statistics, during reconstruction at the receiver, we choose smaller values for the elements of vector τ that correspond to more important timing locations of the input signal. Using this technique, we can significantly decrease the reconstruction error (results shown in Section V). One drawback of this method is that the input signal **x** must be sparse in time. To the best of our knowledge, there are currently no similar algorithms for signals that are sparse in other domains (*e.g.*, wavelet), so we leave this for future work.

3.4 Sensor-on-Chip Architecture

Figure 3-1 shows the block diagram of the proposed SC-assisted compressive-sensing SoC. It consists of an analog front-end (AFE), analog-to-digital converter (ADC), and digital-processing module. The digital-processing module comprises of specialized elements for compressive sensing (CS), statistics collection (SC), and packet management (PM) that organizes all of the required data into a packet format and relays them to the serial interface. Next we present implementation details on each of the sub-blocks.

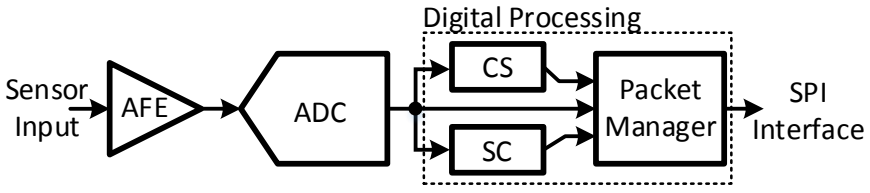

Figure 3-1: Block diagram of the proposed SC-assisted compressive-sensing SoC.

3.4.1 Analog Front-end

The AFE comprises of a chain of current-feedback instrumentation amplifier (CFIA), a fixed-gain amplifier (FGA) and a programmable-gain amplifier (PGA). It also includes a DC servo-

loop (DSL) and a differential output buffer (X1). Figure 3-2 shows block diagram of the AFE and its output for an ECG input. The FGA is used to adjust the bandwidth of the amplifier from 250-935 Hz, while the PGA is used to adjust the gain between 40-53.7 dB. The DSL rejects the offset of the input signal and also sets the high pass corner of the amplifier. The schematic of the CFIA is shown in Figure 3-3. It includes three transconductance stages, one each for the input signal (V_{in}), feedback input (V_{fb}), and DSL output (V_{os}). The current through the input and feedback differential pairs are equal in order to obtain equivalent gains, while a common-source stage is used to drive the load. The measured AFE performance is listed in Table 3-1.

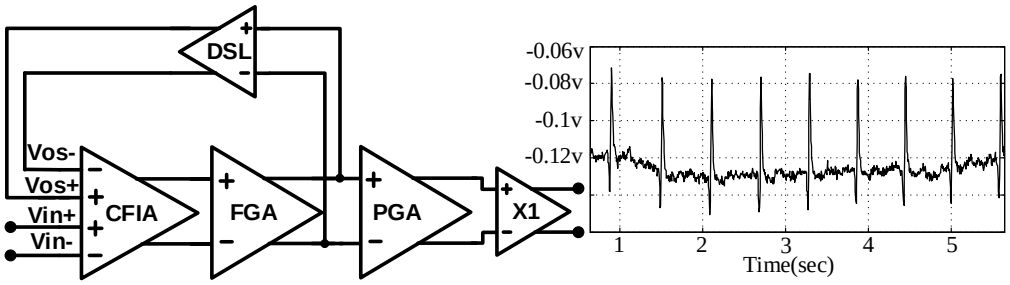

Figure 3-2: Block diagram of the AFE.

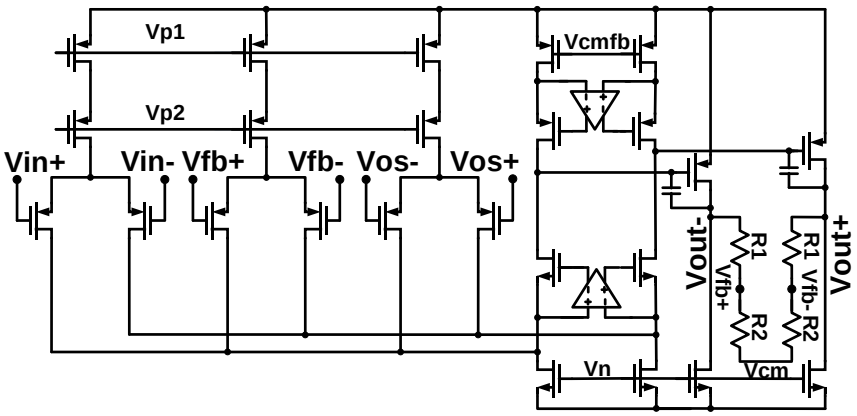

Figure 3-3: Schematic of the CFIA.

3.4.2 Analog-to-Digital Converter

For high-resolution data conversion, higher-order ΔΣ ADCs are more prone to instability, and exhibit a limited non-overloaded input range. Therefore, we integrate a two-step incremental ADC (IADC), which was previously presented in [31]. The two-step operation achieves nearly third-order noise shaping performance while using only second-order IADC circuitry, improving the energy efficiency significantly. The measured performance shows that at 500S/s, the ADC achieves a 100 dB dynamic range and 91 dB SNDR for a 250 Hz input signal. The device consumes only 10.7 µW of power with an active area of only 0.2 mm^2, which is the smallest among similarly performing ADCs.

3.4.3 Digital-Processing Module

On-chip digital processing includes three main blocks. First, the compressive-sensing (CS) block implements compression, where the number of measurements (M) is fixed to 50 in this design. To change the compression factor (CF), the user needs to adjust the length of the input signal (N). Second, the SC block extracts the maximum, threshold point, and for an ECG input, the location and width of the R-peaks. Third, the PM organizes all of the required data into a packet and relays them out of the chip through the SPI port. In the next section, we present more details on each of these blocks.

3.5 Digital System Implementation

3.5.1 Compressive-Sensing Block

This block implements the multiplication of a set of N input samples (coming from the ADC) with a random M×N matrix **Φ** with randomly selected 0 and 1 values. To guarantee that the columns of **Φ** are independent, we combine the outputs of two linear-feedback shift registers (LFSR)[24]. The compressive-sensing block thus generates M output samples for every N input samples. Each of M output samples are inner products of one row of **Φ** and the N input samples. Since the elements of **Φ** are either 0 or 1, we use a select-and-accumulate (SAC) unit to implement this inner-product computation. The clock signal of the digital blocks has a frequency of 96 kHz and is provided by the ADC. If the corresponding element of **Φ** is 0, the clock is gated and the output of the SAC does not change. If it is 1, then the value of the ADC sample will be added to the output of the SAC. After N ADC samples, the SAC contains the final value and is then stored in a secondary register, resetting the SAC for the next data block. Figure 3-4 and Figure 3-5 show the compressive-sensing block and LFSR-based random matrix generation block, respectively.

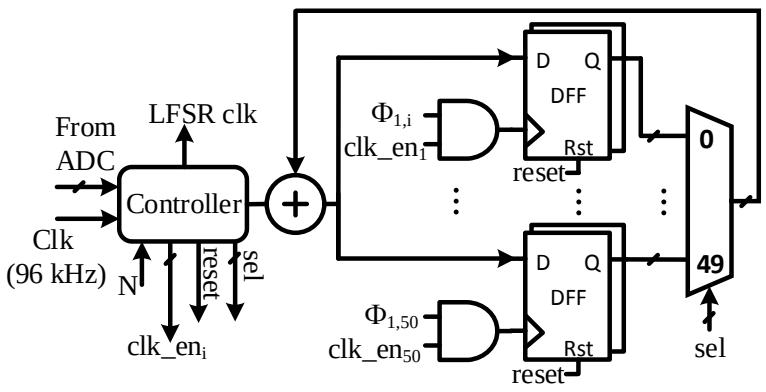

Figure 3-4: Block diagram of the compressive-sensing block.

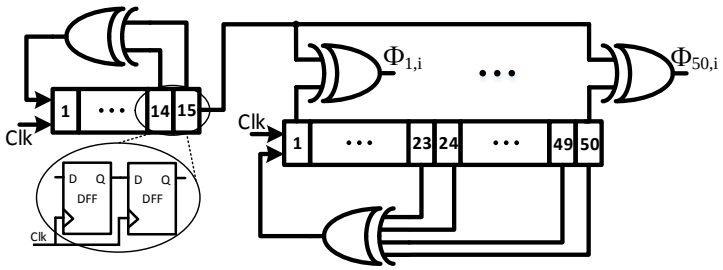

Figure 3-5: Hardware to generate elements of the random matrix **Φ.**

3.5.2 Statistics-Collection Block

SC operates by first finding the index and value of the largest element in each set of the N input samples. It also calculates the start index and width of a consecutive number of data samples that are above a user-defined threshold level. To limit the amount of data overhead created by the SC, the SC only extracts the first two locations of where the threshold is passed. The SC also finds the location and width of the first two R-peaks in the N samples. Thus, for every block of N ADC samples, the SC generates ten 16-bit values for the above statistics, resulting in a 20% data overhead. In practice, this overhead can be easily further reduced by using a larger value of M. Figure 3-6 shows block diagram of SC block.

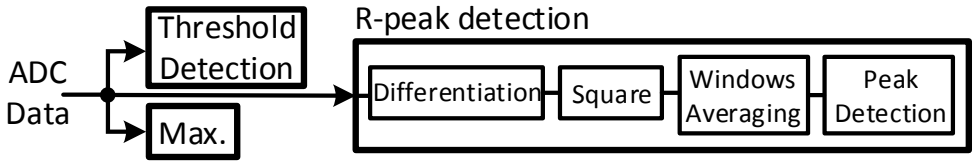

Figure 3-6: Block diagram of statistics-collection block.

3.5.3 Packet Manager Block

The PM organizes all of the data into a single packet by adding a header and footer, and then relaying the data out of the chip using an SPI interface. SPI is a three-wire serial interface with a clock, serial data, and select signals. Serial bits are transmitted during the clock's rising edges when the select bit is low.

3.6 Measurement Results

To verify the SoC's measured performance, we generate a random sparse-in-time signal pulse using an external source applied to the SoC input. Next, we collect statistics from the SC module to find the location and width of this pulse. Figure 3-7 shows the reconstructed signal with and without using the proposed SC-assisted reconstruction. As can be seen, the reconstruction using SC is almost perfect versus the conventional CS-only version. Note that

as discussed in Section II, the current GPSR algorithm currently works only with signals that are sparse-in-time (*i.e.,* $\boldsymbol{\Psi}$ is a unity matrix), such as UWB signals, neural spikes, and time-sparse ECG waveforms. As part of future work, we plan to use a modification of this algorithm for signals that are sparse in other domains.

Figure 3-8 shows the SER for different compression factors with and without using statistics for the same pulse-like input. The SER is defined in Eq. (3) where $\boldsymbol{f}$, $\boldsymbol{f}^*$ are the original and reconstructed signals, respectively:

$$SER = 10\log_{10}\left(\frac{\|f\|_2^2}{\|f - f^*\|_2^2}\right) \tag{3.3}$$

As shown in Figure 3-8 , SC-assisted reconstruction improves SER by as much as 18 dB.

Finally, we demonstrate the effect of using statistic collection on the SER for a real ECG signal input, where timing samples that are not located near the R-peak location and below an adjustable threshold value are converted to zero, thereby providing sparsity in the time domain. Measurements of the reconstructed ECG R-peak waveform are shown in Figure 3-9. For this real ECG waveform experiment, we observe a 9 dB improvement in the SER. The die photograph of the SoC is shown in Figure 3-10, and a comparison with other similar compressive-sensing systems is summarized in Table 3-1.

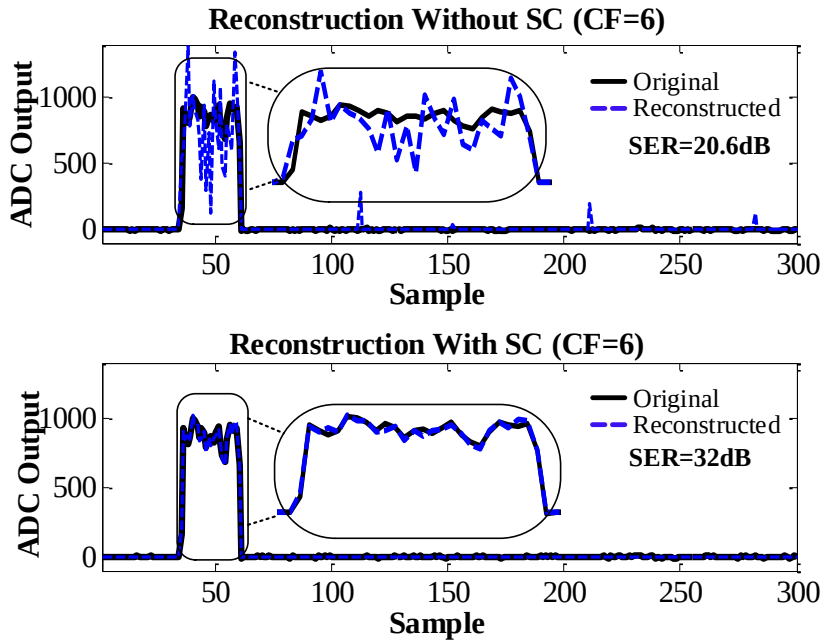

Figure 3-7: SC-assisted versus conventional reconstruction for a pulse waveform.

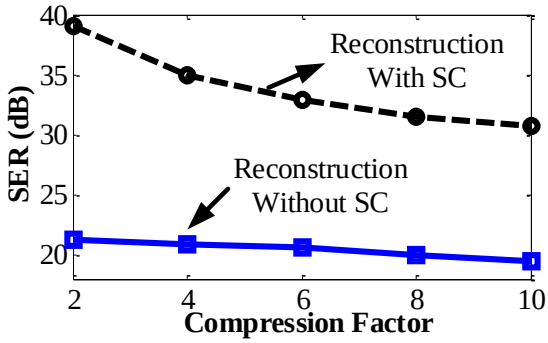

Figure 3-8: SER vs. compression factor with and without using signal statistics.

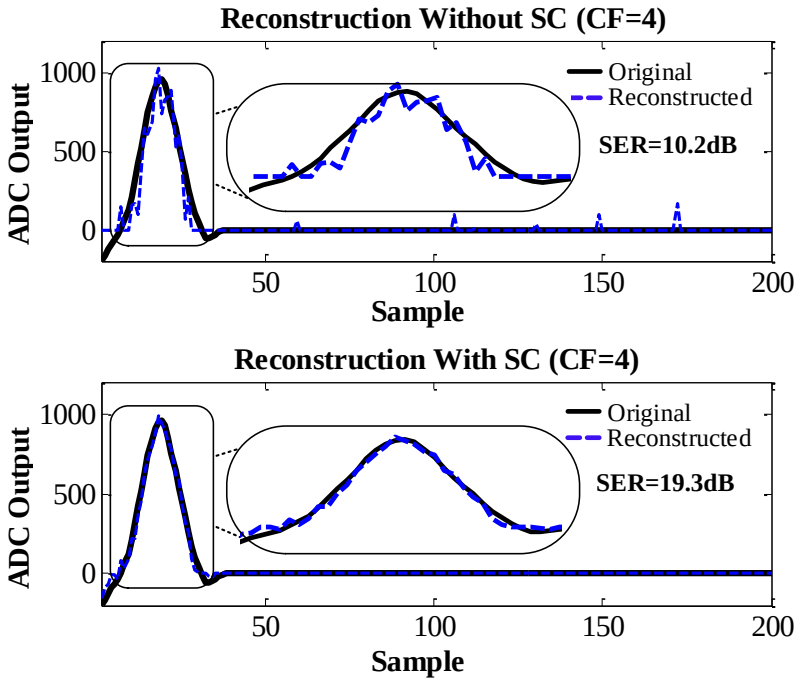

Figure 3-9: Impact of SC on reconstruction for an ECG signal.

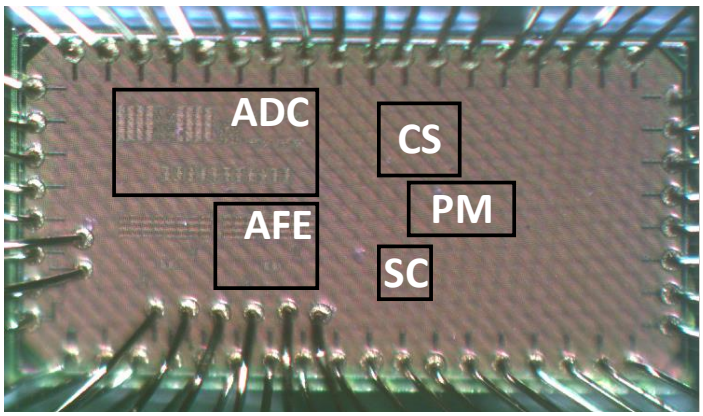

Figure 3-10: Die photograph of the proposed SoC

3.7 Conclusion

In this paper we presented a compressive-sensing SoC for compressing biomedical sensor data that incorporates statistics collection (SC). We showed that using some statistics from the input signal can dramatically improve the performance of reconstruction algorithms for signals that are sparse in the time domain. As part of future work, we are modifying the GPSR algorithm such that it can be used for signals that are sparse in non-time domain bases.

Table 3-1: Performance comparison with previous compressive-sensing SoCs

	Previous CS Sensor-on-Chip		**This Work**
	[3]	**[4]**	
AFE	No	No	Power: 10uW Gain: 40/53.7 dB BW:250/476/721 Hz Input Impedance: 2.5G CMRR: 73 dB @50Hz
ADC	SAR 5-bits ENOB < 1.1µW	SAR 6.5-bits ENOB	Incremental 15-bits ENOB 10.7 µW
Compression	2.5µW @ 100 kHz	28nW @ 2kHz	10.7 µW Static 0.7 µW Dynamic @ 96kHz
Statistics Collection	No	No	12.6 µW Static 0.4 µW Dynamic @ 96kHz
Technology	90 nm CMOS	0.13 µm CMOS	65nm CMOS

3.8 Acknowledgement

This work was partially funded by Center for Design of Analog-Digital Integrated Circuits (CDADIC), and the NSF CAREER program.

3.9 Appendix B: Hardware Implementation Details

The major benefit of CS algorithm is its simple implementation. Here we explain in more details the hardware implementation of CS block. CS as shown in Figure 3-11 is responsible for multiplication of random matrix Φ and vector f of input signal samples.

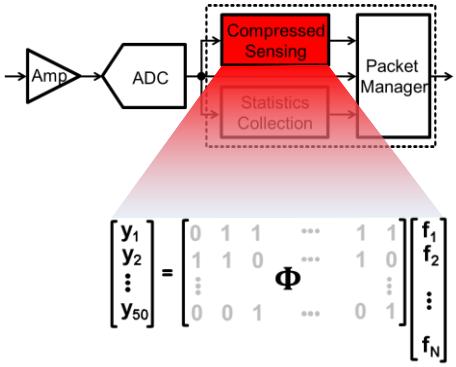

Figure 3-11: CS block implements a matrix-vector implementation

The point is all elements of Φ are either 1 or 0. By rewriting matrix-vector multiplication we have equation (3.4). Φ_i is the i[th] column of Φ.

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{50} \end{bmatrix} = f_1\Phi_1 + f_2\Phi_2 + \cdots + f_N\Phi_N \qquad (3.4)$$

For the simplicity of implementation the number of output samples is always fixed and equal to 50. So to change the compression rate we change the number of samples in input vector f (N). Figure 3-12 shows the hardware used for implementation of (3.4). It consists of 50 (number of output samples) accumulator. The input clock to these accumulators are masked with elements of Φ. If one element is 0, it means the output of accumulator shouldn't change and this is what exactly the AND gate is for. The LFSR block generates one column of Φ in each clock cycle. At the first clock, when the first input sample (f_1) is ready the LFSR also generates the first column of Φ (Φ_1) as shows in Figure 3-13. At this time any output which its corresponding element in Φ is 1 will change based on the input sample. For particular Φ shown in Figure 3-11 for example y_2 will change but y_1 will be fixed after first clock cycle. The same process repeats for the second clock cycle and this time both y_1 and y_2 will change as shown in Figure 3-14 because their corresponding elements in Φ is 1. Finally after N[th] clock, all 50 outputs have their final values and ready to transmit as show in Figure 3-15.

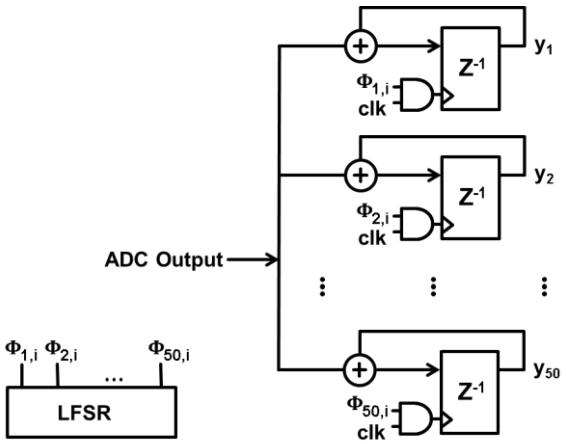

Figure 3-12: Hardware block diagram for CS implementation

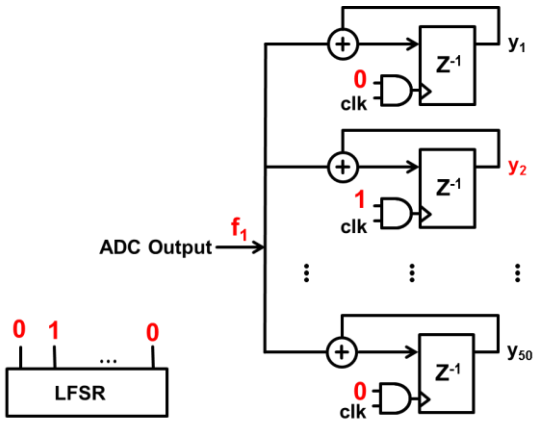

Figure 3-13: Signal values after the first clock cycle

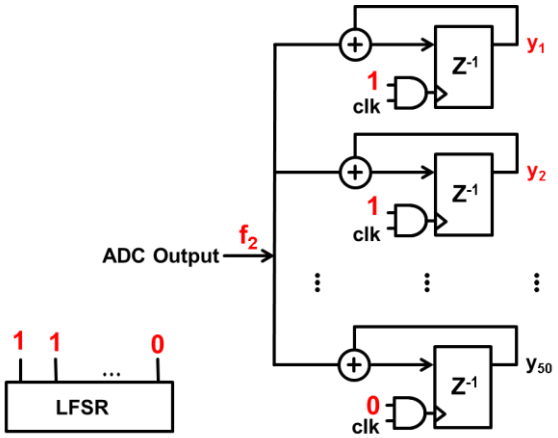

Figure 3-14: Signal values after the second clock cycle

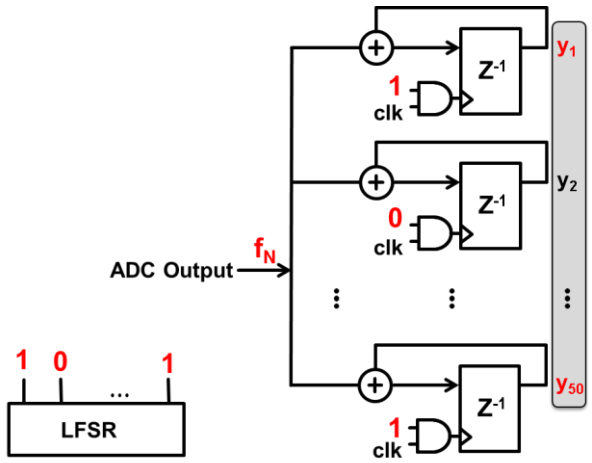

Figure 3-15: Outputs are ready to transmit after the N[th] clock cycle

The same process will continue with the next block of N input samples. There is a timing block in system to reset hardware at the end of each data block. Packet manager block creates a data packet and sends the whole packet to serial interface to send data out of the chip. Figure 3-16 shows format of data packet. Finally SPI used as a serial interface to transmit data out of chip. SPI is a 3-wire interface and Figure 3-17 shows SPI signaling.

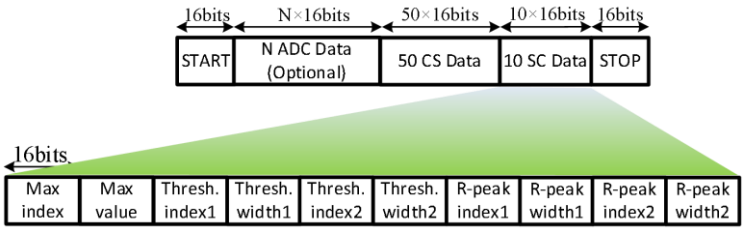

Figure 3-16: Data packet format

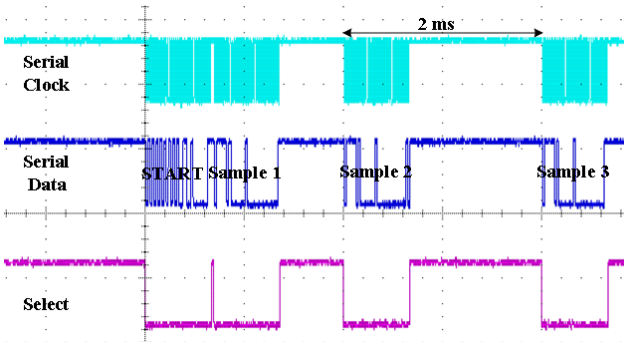

Figure 3-17: SPI interface signaling

3.10 References

[24] F. Chen, “Energy-efficient Wireless Sensors: Fewer Bits, More MEMS,” PhD Thesis, MIT, Sep. 2011.

[25] V. Behravan, et al., “Rate-Adaptive Compressed-Sensing and Sparsity Variance of Biomedical Signals,” Body Sensor Networks 2015 (to appear).

[26] F. Chen, et al., “A signal-agnostic compressed sensing acquisition system for wireless and implantable sensors,” Custom Integrated CCkts. Conf., Sep. 2010.

[27] Gangopadhyay, et al., “Compressed Sensing Analog Front-End for Bio-Sensor Applications,” IEEE J. Solid-State Circuits, vol.49, no.2, pp.426,438, Feb. 2014.

[28] E. Candes, et al., “Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies?,” IEEE Trans. Information Theory, Dec. 2006

[29] http://statweb.stanford.edu/~tibs/lasso.html

[30] M. Figueiredo, R. Nowak and S. Wright, “Gradient Projection for Sparse Reconstruction: Application to Compressed Sensing and Other Inverse Problems, ” IEEE J. Sel. Topics in Signal Processing, Dec. 2007.

[31] C.H. Chen, et al., “ A 11 µW 250 Hz BW two-step incremental ADC with 100 dB DR and 91 dB SNDR for integrated sensor interfaces,” Custom Integrated CCkts. Conf., Sep. 2014.

A Framework For Compressive-Sensing of 3D Point Clouds

Vahid Behravan, Gurjeet Singh and Patrick Y. Chiang

Presented in The 12th International Conference on Computational Intelligence and Security (CIS), 2016.

4.A Framework For Compressive-Sensing of 3D Point Clouds

4.1 Abstract

In this paper we propose a framework for analyzing 3D point cloud data compression using compressive-sensing. This framework uses real 3D point clouds and investigates different error sources that may affect performance of the system. Experimental results show that excluding edge points from error calculation gives us better criteria to decide the best compression ratio in the system.

Keywords: compressive-sensing; point cloud; LiDAR

4.2 Introduction

In recent years depth sensing data has been widely used in many computer vision applications, such as 3D reconstruction, augmented reality (*e.g.* Microsoft HoloLens), autonomous driving (*e.g.* Google car), collision avoidance (*e.g.* DJI drones), and gesture recognition (*e.g.* Microsoft Kinect). The ability for a machine to sense both color (RGB) and depth (Depth Map) has been shown to significantly improve image classification.

The conventional method for acquiring depth information is called stereo vision. It uses two RGB cameras to obtain a pair of stereo images and then uses computational algorithms to extract depth data from those images. Although this method provides a high resolution depth map and uses simple and cheap hardware, however, it uses a large amount of computational DSP hardware. More importantly, stereo vision requires a large illumination that makes it inaccurate to use it in dark environments or poor contrast light scenarios [32]. As a result this method is not accurate for some applications like autonomous driving.

Light Detection And Ranging (LiDAR) is an emerging mechanism that uses time-of-flight (ToF) measurements to compute the distance from the laser source to the target. Although the concept of ToF measurement is very simple, because of high speed of light its hardware implementation was not possible until recent years that high speed electronics are available.

There are two types of LiDAR system. A scanning LiDAR typically acquires multiple samples of a volume by scanning the region at a specified scan rate. The scanning device can be a brushless motor or a MEMS mirror. Flash LiDAR uses an array of imager in which a 2 dimensional array of sensors (*e.g.* PD or SPAD) are used to simultaneously acquire points, just

as a 2 dimensional scene is acquired with an array of detectors in a conventional camera [33]. For either single-beam scanning or flash LiDAR, The resulting measurement is called a point cloud of the region. Scanning LiDAR is generally slower than flash LiDAR but it is usually able to operate up to longer distances since all energy of the laser is focused at one point.

In either scanning or flash LiDAR the light source can be pulsed or modulated by a continuous-wave (CW) source, typically a sinusoid or square wave. Pulsed LiDAR requires a fast photo-detector, usually a single-photon avalanche diode (SPAD) and ToF can be calculated by using a Time to Digital Converter (TDC)[34]. This approach necessitates fast electronics, since achieving 1 millimeter accuracy requires timing a pulse of 6.6 picoseconds in duration. In CW Lidar, the laser is modulated by a continuous sine wave and ToF will be calculated by measuring the phase difference between transmitted and received signal. CW LiDAR doesn't need high frequency hardware but the maximum depth is limited by modulation frequency [32].

Unfortunately, there are two key limitations to LiDAR sensing. The first limitation is the amount of output data. LiDAR data cannot use the traditional JPG encoding (used for traditional 2D-images), so the output data is typically uncompressed. The result is that there is huge amount of data for large scanning LiDAR, and currently there is no way to compress that data.

The second issue is the limited frame rate. Frame rate here means how many point cloud can LiDAR device capture in a second. The frame rate is usually limited by scanning device or readout circuit and for many applications (e.g. autonomous driving) high frame rate is required. A key question in any power efficient LiDAR system (*e.g.* wireless sensor applications) is how many points we need to capture to fully obtain the scene point cloud. The lower points we need to capture the lower energy will be needed to transmit this data to the receiver. Also it increases the frame rate.

Compressive sensing is a method that enables reduction of the LiDAR data. Previous papers [35][36][37] attempted to do this, but didn't propose a comprehensive framework to analyze a compressive-sensing LiDAR system. Lau [35] proposes a new way of applying compressive sensing, called re-sampling compressive sampling to reduce LiDAR sampling rate required to capture 3 dimensional scenes. This method needs to generate and process a matrix with the order of n^4 elements for a $n \times n$ point-cloud that makes it computationally impractical for many applications. Hawe [36] proposes a conjugate subgradient method for solving the arising compressive-sensing problem of a depth map from a stereo vision camera. This method efficiently subsamples the depth map by randomly choosing points just from edges of the object

in the depth map, however, this method is just useful for stereo vision applications that depth map is extracted from original RGB images, so the edges points are known. For ToF camera finding edge points in a complex point cloud is not trivial and this method can't be used. Liu [37] shows that stereo vision depth maps are sparser using a dictionary of both wavelet and contourlet atoms. He also proposed a new reconstruction algorithm that results in smaller error in the reconstruction depth map. However this method also doesn't investigate the performance on a point cloud from a ToF camera. A key question that none of the above works answer is what sub-sampling ratio (compression rate) is actually enough for a given point cloud. We know that a higher compression rate results in higher reconstruction error, however, we need to know how much error is acceptable for a given point cloud. More importantly we are interested in finding the best criterion for the error measurement. To the best of our knowledge all literatures proposed compressive-sensing for depth map or point cloud use Mean Absolute Error (MAE) over all data points that may not be the best criterion.

This paper presents a framework and methodology for compressive sensing of a LiDAR system. In this work we investigate different error criteria and the effect of different compression rates on these errors for multiple point clouds. Point clouds used in this research are taken by a Google Tango tablet that uses a CW ToF depth sensor [33]. We investigate two different error criteria: MAE over the whole point cloud with and without excluding errors at the edge points. The goal is to create a framework for point cloud compressive-sensing that gives us a better understanding about the best criterion for the selection of the optimum compression rate for a given scene.

This paper is organized as follows. Section II introduces a basic background for compressive-sensing. Section III explains our proposed framework to analyze a compressive-sensing LiDAR system. Experimental results will be shown in Section IV and finally Section V provides conclusion.

4.3 Compressive-sensing Background

Compression using compressive-sensing is achieved using a simple matrix multiplication, where an uncompressed input vector $\boldsymbol{f}$ of size N multiplied by a measurement matrix $\boldsymbol{\Phi}$ of size M-by-N that produces a measurement vector $\boldsymbol{y}$ of size M. $\boldsymbol{\Phi}$ is a matrix of random numbers (*e.g.* Bernoulli, Gaussian, uniform, etc.), such that $\boldsymbol{y}$ is a vector of random linear projections of $\boldsymbol{f}$ on $\boldsymbol{\Phi}$. Compressive-sensing has shown to be able to compress different type of sensor signal with compression factors as large as 10x, which reduces the amount of data transmitted and

therefore the total power dissipation of the sensor node by a similar factor. To reconstruct the original signal $\boldsymbol{f}$ from the measured signal $\boldsymbol{y}$ in the receiver, we need to solve an underdetermined set of equations where the number of equations is much less than the number of unknown variables. In general, there is not a unique solution for these types of equations. However, compressive-sensing theory shows that if the signal $\boldsymbol{f}$ is sparse in any basis (*e.g.* Fourier transform, wavelet transform), then there are optimized methods to reconstruct the original signal with minimum error based on convex optimization [39]. In other words, if there is a transformation matrix $\boldsymbol{\Psi}$ such that $\boldsymbol{f} = \boldsymbol{\Psi}\mathbf{x}$ (and therefore $\boldsymbol{y} = \boldsymbol{\Phi f} = \boldsymbol{\Phi\Psi}\mathbf{x}$) and $\mathbf{x}$ is sparse, then reconstruction is possible. Figure 4-1 summarizes the complete compressive-sensing framework [40].

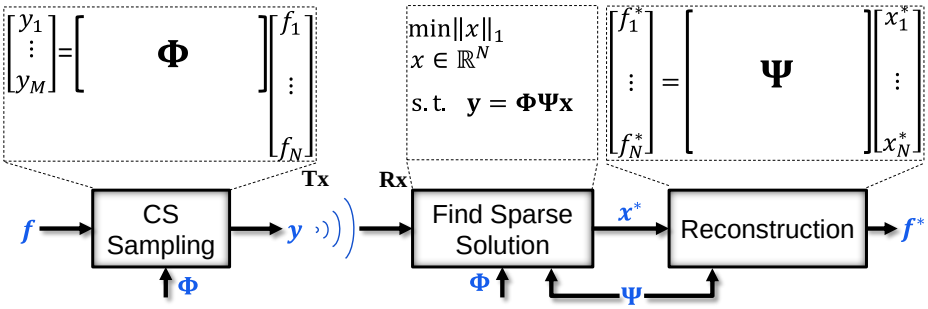

Figure 4-1: Complete compressive-sensing framework

4.4 Compressive-sensing Framework For LiDAR

Figure 4-2 shows the proposed framework for analysis of a compressive-sensing LiDAR system. After taking point cloud we need to remove singular points. These points are created because of LiDAR measurement error. Figure 4-3 shows one sample point cloud before removing singular points. As it can be seen these points usually happen at the edge of the objects and are result of light diffraction at the edges. Singular points create unwanted error when we convert point cloud to gray scale image.

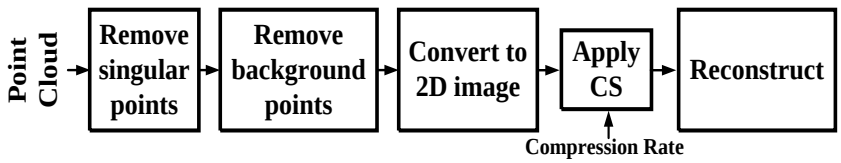

Figure 4-2: Compressive-sensing framework for LiDAR

The next step is removing background points. Background points are usually not the points of interest and affect analysis of the compressive-sensing LiDAR system. Background usually includes a shadow of front objects and creates dummy edge points after converting point cloud

to a 2D image. Figure 4-3 shows how background points create dummy edges in 2D mapping of the point cloud. After removing background points we convert point cloud to a 2D gray scale image. This step helps to treat point cloud as an image and makes compressive-sensing implementation and analysis much easier. The only point here is that we have to make sure the resolution of 2D image is high enough such that multiple points of point cloud don't map to a single pixel of 2D image. Having a 2D image implementation of compressive-sensing is straightforward and explained in different papers [36][37]. In this work we assume we have no a-priori knowledge of edge points and random samples of point cloud for compressive-sensing are chosen based on a uniform random pattern.

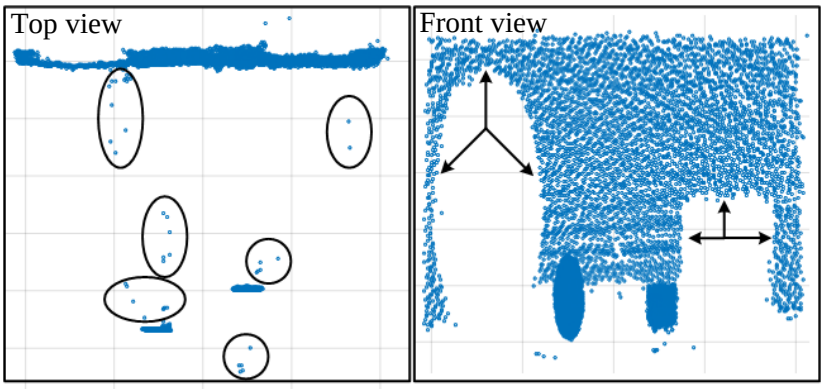

Figure 4-3: Singular points and dummy edges in point cloud of scene 2

After compressive-sensing we need to reconstruct the original 2D image from the compressed data by solving an optimization problem as explained in Section II. It is shown that point cloud data is sparse in wavelet and contourlet domains [37]. In this work we consider wavelet domain sparsity for reconstruction. Finally we calculate the reconstruction error. This error is actually the only parameter available to analyze how compressive-sensing affects the point cloud and what is the best compression rate based on this error. All available literature use MAE as an error criterion. This is a reasonable criterion but the problem is that when we calculate MAE over the whole point cloud the error from non-important points may exceed the error resulted from important points and affect our analysis. In this work we look at another error criterion that is MAE over the whole point cloud without considering errors from edge points. In other words we exclude edge points from error calculation.

4.5 Experimental Results

We capture point cloud from four different scenes with different complexity. After removing singular and background points we compress point clouds with different rates and for each case

calculate two different errors to understand how they are related to each other. Figure 4-3 shows front and top view of a 3D point cloud (from scene 2 in Figure 4-4). It can be seen that how singular points exist because of measurement error of LiDAR device at the edge of the objects. Front view shows how background points lead to false edges that create dummy edges in the resulting 2D image. As a result, our framework proposes to remove singular points and background points before next steps to avoid extra error in compressive-sensing system because of these artifact effects.

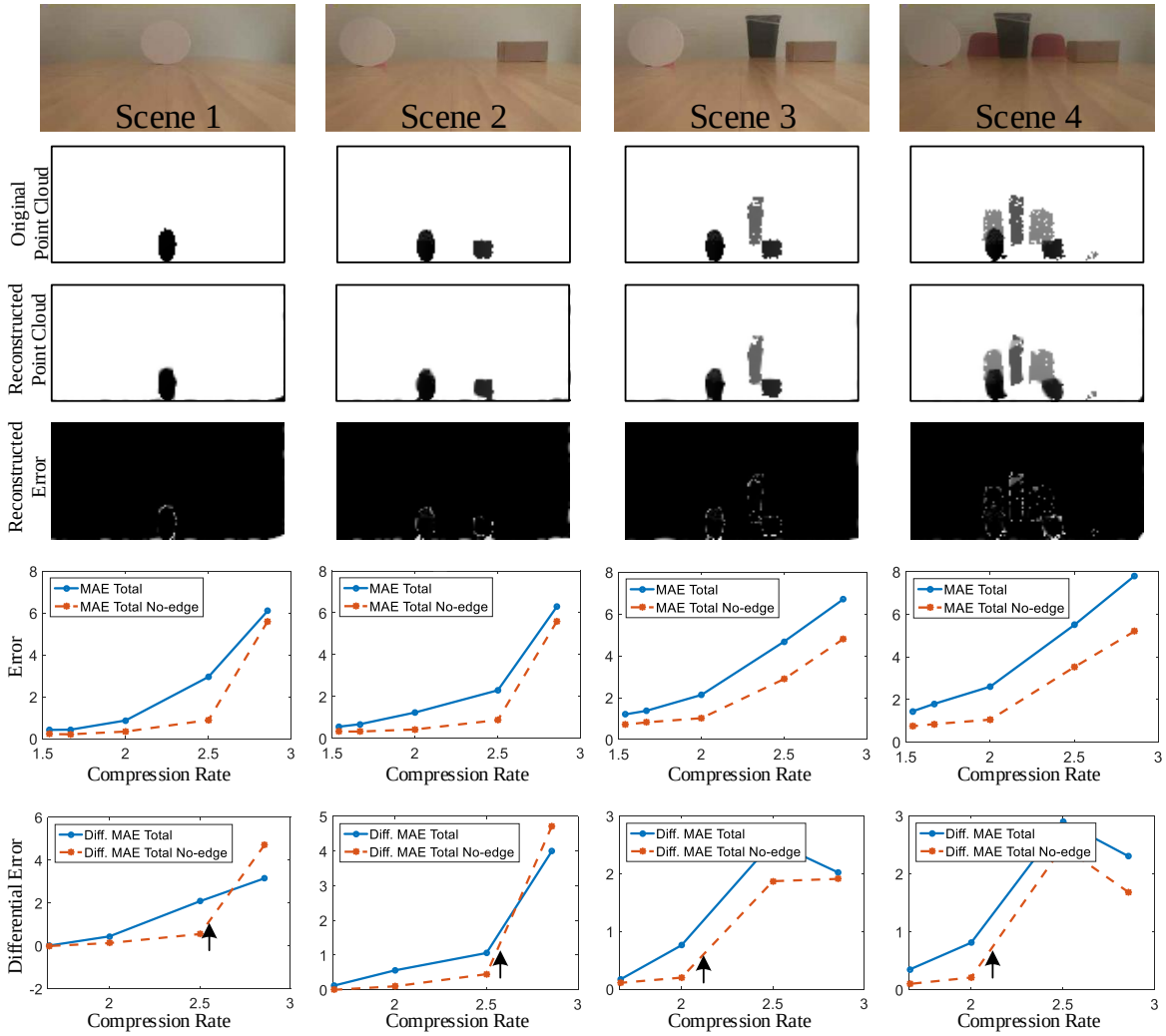

Figure 4-4: Four different scenes and their point clouds

Figure 4-4 shows experimental results. The first row is RGB image of each of four scenes. The second row is gray scale 2D version of 3D point cloud after removing singular and background points. The third row shows the reconstructed gray scale image after compressive-sensing. The fourth row is the reconstruction error or in other word the difference between second and third rows. The darker pixels mean smaller errors. It can be seen that the largest errors happen at the

edge points. That is why it intuitively makes scene to exclude these points from error calculation. Remember that what we really need to know is distance of each object to LiDAR device so removing few edge points doesn't create any problem. The fifth row shows plots of two different errors. We calculate Mean Absolute Error (MAE) over all pixels of the point cloud with and without considering the errors at the edge points. For each point cloud we calculate error at five different compression rates of 1.54, 1.67, 2, 2.5, 2.86 that are respectively correspond to the under sampling ratios of 0.65, 0.6, 0.5, 0.4, 0.35. In the sixth row we plot the differential error. Starting at compression rate of 1.67 we calculate the difference between the current error and the reconstruction error at the previous compression rate.

The question here is what the optimum compression rate for a given scene is without actually looking at the reconstructed point cloud. We suggest that using differential error helps to find this optimum rate. As you can see for scene 2 and 4 the highest increase rate at the error happens at compression rate of 2.5. We can find this by looking at the slope of the differential error plot. For scenes 3 and 4 this highest slope happens at the compression rate of 2 that can be selected as the optimum rate. Figure 4-5 shows reconstructed point clouds of scenes 1 and 4 at 2.5x compression. It can visually be seen that at this compression rate, scene 1 reconstructed well, however, scene 4 shows multiple extra objects because of error. This is compatible with our previous finding that 2.5x is not an optimum rate for scene 4.

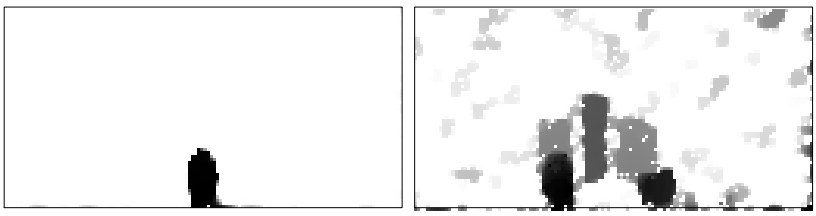

Figure 4-5: Point clouds of scene 1 and 4 after reconstruction at compression rate of 2.5

Having such criterion for automatically choosing the optimum compression rate helps to develop machine learning or deep learning algorithm to find the optimum compression rate by looking at the input scene. It also can be seen that excluding edge points from error calculation slightly increases the slope of differential error plot at the optimum compression rate that make it easier to find this point.

4.6 Conclusion

In this paper we proposed a framework to analyze compressive-sensing LiDAR system. Experiments are done using different point clouds captured by a Google Tango tablet. We

showed that the maximum error generated by compressive-sensing happens at the edge points of point cloud. We also showed the effect of scene background and intrinsic measurement error of LiDAR at the edge points to the resulting point cloud. We finally showed that excluding edge points from error calculation can helps us to find an optimum compression rate for each point cloud in these specific scenes.

4.7 Appendix C: Basics of Depth Measurement

Depth measurement has found many applications in recent years including 3D scanning, gesture recognition and autonomous driving as shown in Figure 4-6. There are three major methods to measure distance to an object. Stereo matching is a cost-efficient but computationally intensive method that uses two RGB cameras located in a fixed position. By finding difference between locations of a particular point of object in two images, distance of that point to both cameras can be calculated. Structured light projects a structured pattern of light to the object as shown in Figure 4-7. An algorithm figures out how much the original pattern is distorted and based on that 3D shape of object can be extracted. The third method is called Time of Flight (ToF). A transmitter send a narrow pulse of laser beam. A receiver detects the received pulse. Since light travels in fixed speed, having time of flight easily calculates distance. Figure 4-8 shows a block diagram of this system. A device that uses ToF technique to extract depth information in called LiDAR.

Figure 4-6: Some applications of depth measurement

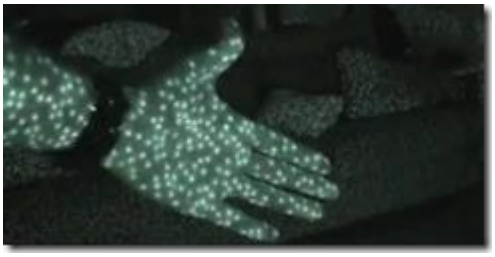

Figure 4-7: A structured pattern light (Source: Microsoft Robotics Blog)

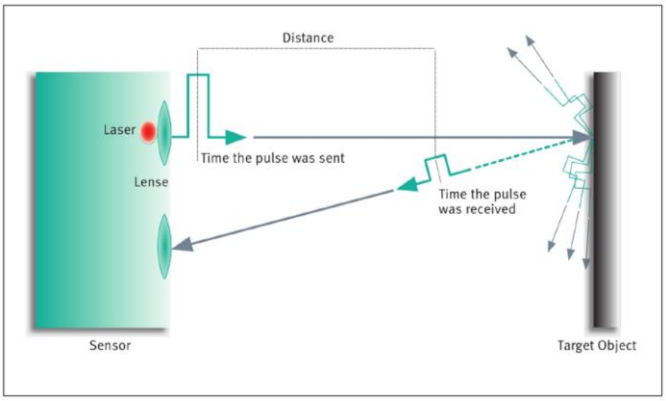

Figure 4-8: A simplified ToF camera system [Source: www.automationworld.com]

A LiDAR device always includes a laser and a light detector and it may have a scanning device to scan a field of view. The result of scanning a field of view by LiDAR is called 3D point cloud that has distance to different points in the area of scanning. Figure 4-9 shows a sample LiDAR system setup.

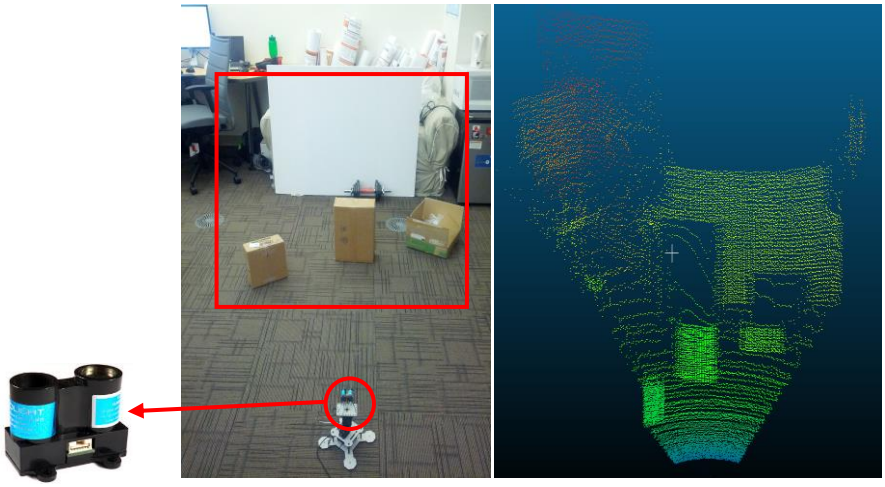

Figure 4-9: A sample LiDAR system setup

Figure 4-10 shows a history of commercial devices used by distance measurements. As you can see new devices mostly use ToF mechanism.

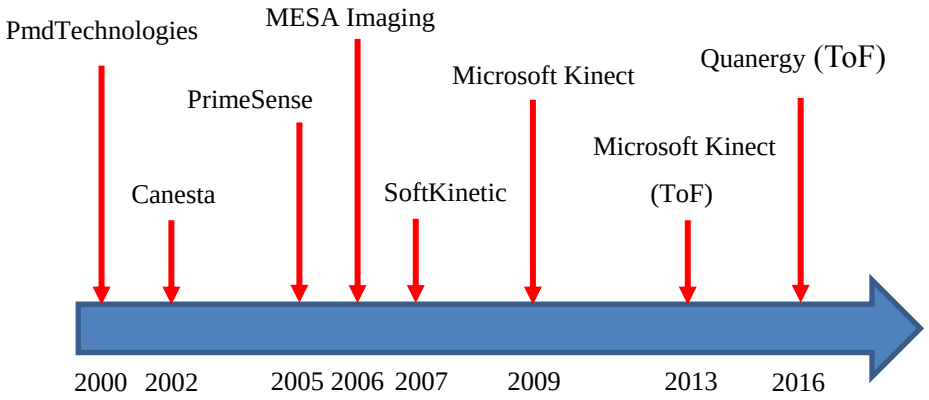

Figure 4-10: History of depth measurement devices

As mentioned before a LiDAR device includes a laser and a light detector as receiver. Light detector can be array based (*e.g.* PMD 19k-S3 from PMD technologies) or just a single photo diode. In this case usually a scanning device (*e.g.* MEMS mirror). Table 4-1 shows an example of these devices and their specifications.

Table 4-1: Some typical parts used in a LiDAR device

Part number	**Type**	**Vendor**	**Specifications**	
PMD 19k-S3	Sensor	PMD	120x160 Pixels, Up to 80 MHz 175 mW (@ 3 ms Integration time & 5MHz readout speed)	Flash
SLD1239JL-54	Laser Diode	Sony	2.5vx150mA = 375 mW (To generate 100 mW optical power)	Flash Scanning
MEMS Mirror/Driver	Scanning Device	Mirrorcle	~150 mW	Scanning
C30659-900	APD	Excelitas	35mA*5v = 175 mW	Scanning

Photo detector is the main part of a LiDAR device. Many recent detectors are designed as array of single detectors (shown as Pixel Matrix in Figure 4-11). The benefit of this array is that there is no need for a scanning device. Figure 4-11 shows a block diagram of an advanced LiDAR (depth) camera.

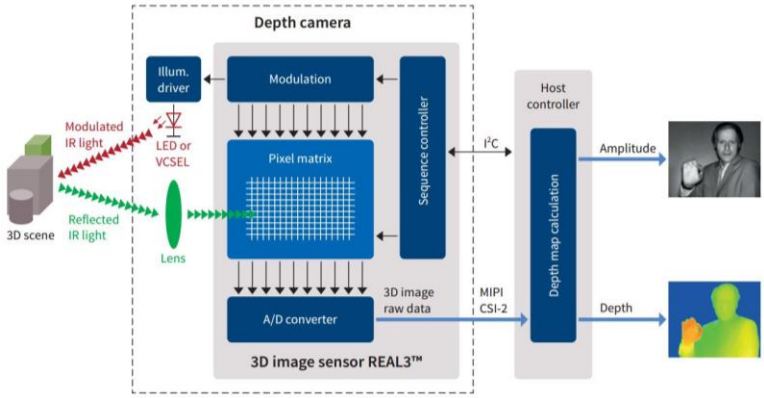

Figure 4-11: A simplified block diagram of an advanced Lidar camera [41]

As mentioned before a LiDAR device can work in CW or pulse mode. In both cases after photo diode there is a circuit to calculate ToF. In CW LiDAR a method called 4-bucket approach is used to calculate ToF. This method will be explained in more detail in Appendix D. In pulse LiDAR a Time to Digital Converter (TDC) is used to directly calculate the time difference between transmit and received pulse. A detail explanation of TDC design for LiDAR application can be found in [42]. Figure 4-12 shows a basic block diagram of a pulse LiDAR with TDC circuit as proposed in [42].

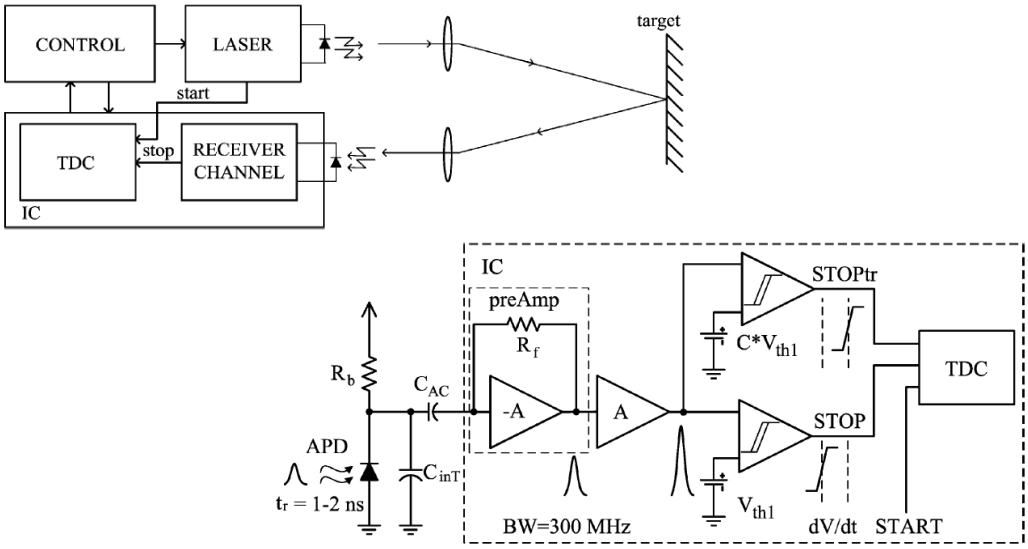

Figure 4-12: A pulsed puled LiDAR with TDC [42]

There are other publications about design of TDC for LiDAR. Jihyun Cho [43] has a good work for reference. Recently different companies has made effort to make LiDAR cameras with higher range and lower cost. Innoluce [44] achieved 250 meters target range with their LiDAR shown in Figure 4-13. This is a scanning LiDAR and there is a MEMS mirror at its core. Quanergy [45] is another startup has made low cost LiDAR camera.

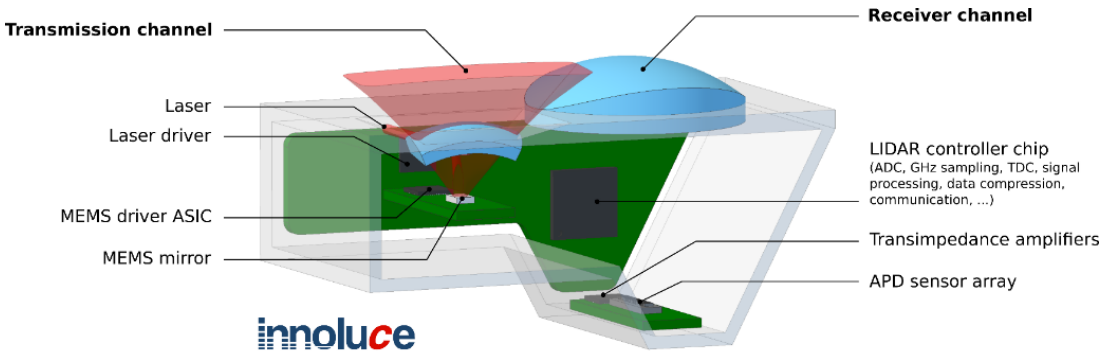

Figure 4-13: High range LiDAR from Innoluce

Many researchers have done works in this area as well both in LiDAR hardware design and point cloud data processing. Ahmed Kirmani [46] proposed a compressive depth acquisition camera framework that uses a single sine wave light source and different spatial light modulator to extract multiple depth values from a single receive. Figure 4-14 shows basic block diagram of this work.

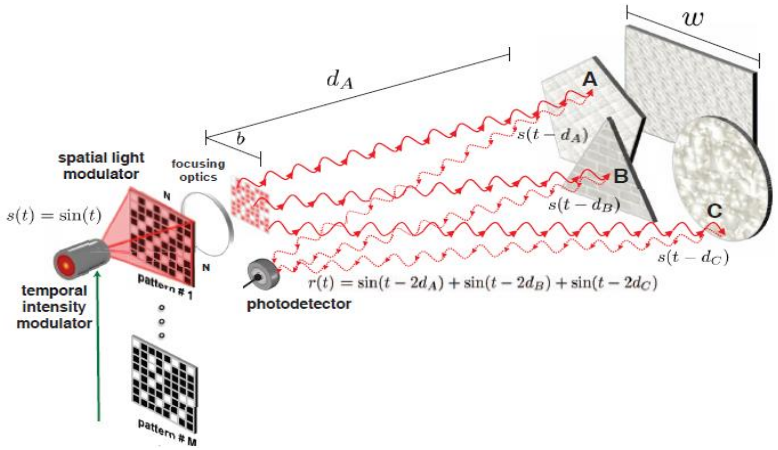

Figure 4-14: A compressive depth acquisition camera framework

Robert Moss [47] implemented a complete LiDAR camera system using a 2D MEMS mirror as shown in Figure 4-15. The FPGA is responsible for driving MEMS mirror and processing digital samples of received light. Milanovic [23] proposed a method for optical 3D tracking using a MEMS mirror that also can be used as a LiDAR camera. It uses a beam splitter to conduct receive light to the receiver as shown in Figure 4-16.

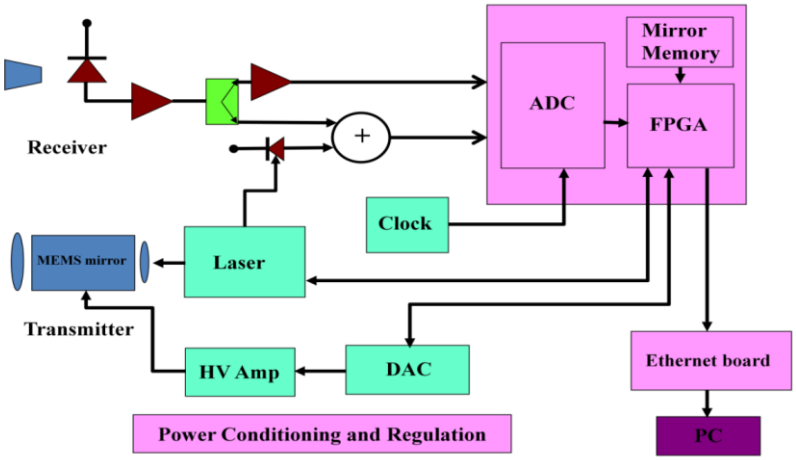

Figure 4-15: A LiDAR camera system using 2D MEME mirror

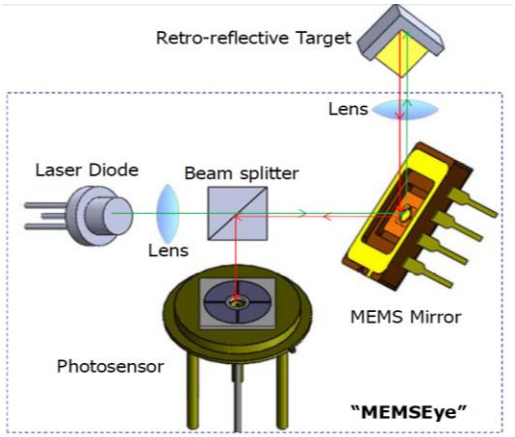

Figure 4-16: MEMSEye system for optical tracking and depth measurement

Mutto [49] gives very detailed understanding of CW ToF cameras. It also explains 4-bucket approach to calculate phase difference between transmitted and received signal. Unfortunately details of calculation is not provided. Appendix D shows necessary calculation for 4-bucket approach. Kadambi [50] proposes a new depth sensing architecture that explains a fixed coded aperture to reduce the number of sensors needed for depth measurement. He also proposed coded ToF cameras [51] that uses sparse deconvolution to address multipath interference and recovery of time profiles. Two major applications of this work are visualizing light sweeping a scene and recovering depth of transparent objects. Kadambi [52] also proposes concept of frequency domain ToF that extracts object depth in frequency domain. In other words in his proposed system depth measurement is only relied on frequency not phase. Bhandari [53] used modulation frequency diversity and sparse regularization for resolving multi-path interference in ToF imaging. Kadambi [54] introduced a complete survey of 3D depth cameras and their

applications in computer vision. Lange [55] has one of the earliest work that adequately formulates problem of depth measurement using ToF cameras. Shcherbakova [56] designed a 3D camera based on linear-mode gain-modulated avalanche photodiodes. Foix [57] reviewed the state of the art in the field of ToF cameras. Castorena [58] proposed a new mechanism for compressive sampling of CW LiDAR by using concept of finite rate of innovation. He also introduced a mechanism to model scene sparsity in LiDAR camera using compressive-sampling [59]. Laky [60] formulated sparse representation of CW LiDAR point clouds. Niclass [61] implemented a CMOS SoC in 0.18 μm thechnology for a 100-m range 10 frames/s 200×96 pixel ToF camera. Wang [62] proposed a mechanism for edge extraction by merging 3D point cloud and 2D image data. A survey of LiDAR technology and its application in spacecraft relative navigation presented by Christian [63]. One issue of ToF cameras is that they need calibration. Kahlmann [64] explained the calibration process to increase accuracy of CW LiDAR. Uriarte [65] modeled distance nonlinearity in ToF cameras and corrected these nonlinearities using integration time offset.

4.8 Appendix D: Detail of ToF Calculation

As mentioned before ToF for pulse LiDAR can directly be measured by measuring time difference between transmitted and received pulse. For CW we need a mechanism to find phase difference between transmitted and received wave. We can calculate ToF by having phase since the light speed is fixed and known. A method called 4-bucket can be used to measure phase difference. Assume the transmit signal is a sine waveform in the form of equation (3.5).

$$x(t) = \sin(w_0 t) \tag{3.5}$$

The received signal will have an amplitude and phase offset as equation (3.6).

$$s(t) = A\sin(w_0 t + \varphi) + B \tag{3.6}$$

The goal is finding A, B, φ. In 4-bucket approach we sample $s(t)$ with a clock frequency equal to 4 times of input frequency (w_0) as shown in Figure 4-17. As a result at the receiver we have four samples with equations as in (3.7).

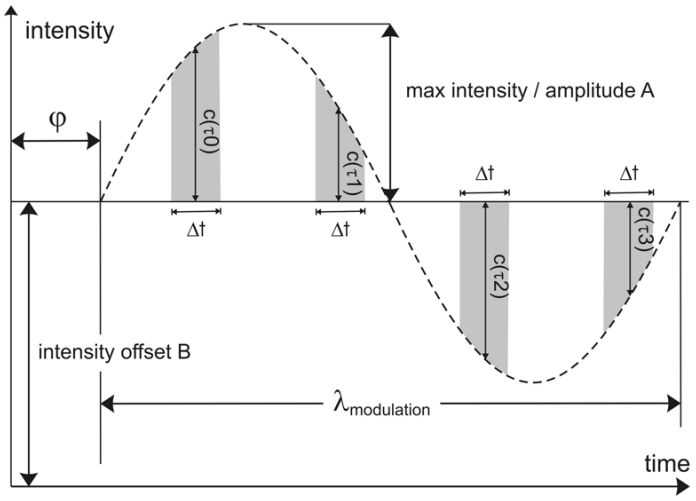

Figure 4-17: 4-bucket approach to measure phase difference [64]

$$
\begin{aligned}
s_0 &= c(0) = A\sin(\varphi) + B \\
s_1 &= c(1) = A\sin\left(2\pi f_0 \times \frac{1}{4f_0} + \varphi\right) + B = A\cos(\varphi) + B \\
s_0 &= c(2) = A\sin\left(2\pi f_0 \times \frac{2}{4f_0} + \varphi\right) + B = -Asin(\varphi) + B \\
s_0 &= c(3) = A\sin\left(2\pi f_0 \times \frac{3}{4f_0} + \varphi\right) + B = -Acos(\varphi) + B
\end{aligned}
\tag{3.7}
$$

We can write (3.7) in a matrix form of (3.8) that e is measurement error.

$$
\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} sin\varphi & 1 \\ cos\varphi & 1 \\ -sin\varphi & 1 \\ -cos\varphi & 1 \end{bmatrix} \cdot \begin{bmatrix} A \\ B \end{bmatrix} + e \Rightarrow y = Kx + e \tag{3.8}
$$

The goal is finding A, B, φ such that estimation error is minimized.

$$
arg\min_x \|y - Kx\|_2^2 \tag{3.9}
$$

This is a standard least square estimation problem we know that the solution of (3.9) is equation (3.10).

$$
x = (K^T K)^{-1} K^T y \tag{3.10}
$$

Interestingly $K^T K$ is independent of φ .

$$
K^T K = \begin{bmatrix} sin\varphi & cos\varphi & -sin\varphi & -cos\varphi \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} sin\varphi & 1 \\ cos\varphi & 1 \\ -sin\varphi & 1 \\ -cos\varphi & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \tag{3.11}
$$

By solving (3.10) one can find:

$$x = \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(s_0 - s_2)sin\varphi + \frac{1}{2}(s_1 - s_3)cos\varphi \\ \frac{1}{4}(s_0 + s_1 + s_2 + s_3) \end{bmatrix} \quad (3.12)$$

By replacing A, B in (3.9) we finally find that:

$$\varphi = \tan^{-1}(\frac{s_0 - s_2}{s_1 - s_3}) \quad (3.13)$$

4.9 References

[32] Larri Li, "Time-of-Flight Camera – An Introduction," Technical White Paper, May 2014.

[33] www.pmdtech.com

[34] J. Nissinen, I. Nissinen and J. Kostamovaara, "Integrated Receiver Including Both Receiver Channel and TDC for a Pulsed Time-of-Flight Laser Rangefinder With cm-Level Accuracy," in IEEE Journal of Solid-State Circuits, vol. 44, no. 5, pp. 1486-1497, May 2009.

[35] Richard C. Lau ; T. K. Woodward, "A new approach to apply compressive sensing to LIDAR sensing," Proc. SPIE 9109, Compressive Sensing III, May 2014.

[36] S. Hawe, M. Kleinsteuber and K. Diepold, "Dense disparity maps from sparse disparity measurements," 2011 International Conference on Computer Vision, Barcelona, 2011, pp. 2126-2133.

[37] L. K. Liu, S. H. Chan and T. Q. Nguyen, "Depth Reconstruction From Sparse Samples: Representation, Algorithm, and Sampling," in IEEE Transactions on Image Processing, vol. 24, no. 6, pp. 1983-1996, June 2015.

[38] https://get.google.com/tango

[39] Candes, E.J.; Tao, T., "Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies?," Information Theory, IEEE Transactions on , vol.52, no.12, pp.5406,5425, Dec. 2006

[40] F. Chen, "Energy-efficient Wireless Sensors: Fewer Bits, More MEMS," PhD Thesis, Dept. Electr. Eng. Comp. Sc., Massachusetts Inst. Technol., Cambridge, MA, USA, Sep. 2011.

[41] http://image-sensors-world.blogspot.com/2015/12/pmd-and-infineon-present-improved-tof.html

[42] Jan Nissinen, et al, “Integrated Receiver Including Both Receiver Channel and TDC for a Pulsed Time-of-Flight Laser Rangefinder With cm-Level Accuracy”, JSSC 2009

[43] Jihyun Cho, et al, “A 3-D Camera With Adaptable Background Light Suppression Using Pixel-Binning and Super-Resolution”, JSSC 2014.

[44] http://www.innoluce.com/

[45] http://quanergy.com/

[46] A. Kirmani, A. Colaço, F. N. C. Wong and V. K. Goyal, "CoDAC: A compressive depth acquisition camera framework," *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, 2012

[47] Robert Moss, et al., Low-cost compact MEMS scanning LADAR system for robotic Applications, SPIE 2012.

[48] V. Milanovic, et al. ,“MEMSEYE” FOR OPTICAL 3D TRACKING AND IMAGING APPLICATIONS, Transducers 2011 Conference

[49] C. D. Mutto, et al. , “Time-of-Flight Cameras and Microsoft Kinect”, Springer, 2013.

[50] A. Kadambi and P. T. Boufounos, "Coded aperture compressive 3-D LIDAR," *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, South Brisbane, QLD, 2015.

[51] A. Kadambi, et al., “Coded time of flight cameras: sparse deconvolution to address multipath interference and recover time profiles”, ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2013.

[52] A. Kadambi, et al., “Frequency Domain TOF: Encoding Object Depth in Modulation Frequency”, arXiv, 2015.

[53] A. Bhandari, et al., “Resolving Multi-path interference in Time-of-Flight Imaging via Modulation Frequency Diversity and Sparse Regularization”, Optics Letter, Volume 39, Issue 06, 2014.

[54] A. Kadambi, et al., “3D Depth Cameras in Vision: Benefits and Limitations of the Hardware”, Springer, 2014.

[55] R. Lange and P. Seitz, "Solid-state time-of-flight range camera," in *IEEE Journal of Quantum Electronics*, vol. 37, no. 3, pp. 390-397, Mar 2001.

[56] O. Shcherbakova, L. Pancheri, G. F. D. Betta, N. Massari and D. Stoppa, "3D camera based on linear-mode gain-modulated avalanche photodiodes," *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, San Francisco, CA, 2013.

[57] S. Foix, G. Alenya and C. Torras, "Lock-in Time-of-Flight (ToF) Cameras: A Survey," in *IEEE Sensors Journal*, vol. 11, no. 9, pp. 1917-1926, Sept. 2011.

[58] J. Castorena and C. D. Creusere, "Compressive sampling of LIDAR: Full-waveforms as signals of finite rate of innovation," *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, Bucharest, 2012.

[59] J. Castorena, C. D. Creusere and D. Voelz, "Modeling lidar scene sparsity using compressive sensing," *2010 IEEE International Geoscience and Remote Sensing Symposium*, Honolulu, HI, 2010.

[60] S. Laky, P. Zaletnyik, C. Toth and B. Molnar, "Sparse representation of full waveform lidar data," *2012 IEEE International Geoscience and Remote Sensing Symposium*, Munich, 2012.

[61] C. Niclass, M. Soga, H. Matsubara, M. Ogawa and M. Kagami, "A 0.18- um CMOS SoC for a 100-m-Range 10-Frame/s 200×96-Pixel Time-of-Flight Depth Sensor," in *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, pp. 315-330, Jan. 2014.

[62] Y. Wang, D. Ewert, D. Schilberg and S. Jeschke, "Edge extraction by merging 3D point cloud and 2D image data," *2013 10th International Conference and Expo on Emerging Technologies for a Smarter World (CEWIT)*, Melville, NY, 2013, pp. 1-6.

[63] J. A. Christian, S. Cryan, "A Survey of LIDAR Technology and its Use in Spacecraft Relative Navigation", AIAA Guidance, Navigation, and Control (GNC) Conference Boston, MA, 2013.

[64] T. Kahlmann, et al., "Calibration for Increased Accuracy of the Range Imaging Camera SwissRangerTM", Proceedings of the ISPRS Com. V Symposium, 2006.

[65] C. Uriarte, et al., "Modelling Distance Nonlinearity in ToF Cameras and Correction Based on Integration Time Offsets", CIARP 2011, Springer.

5.Adaptive Compressive-Sensing of 3D Point Clouds[1]

5.1 Abstract

In this paper we propose a mechanism for adaptive compressive-sensing of 3D point cloud. One problem with conventional compressive-sensing algorithm is a-priori fixed compression rate. In this work we show that there is a correlation between sparsity of point cloud and number of edge points. The algorithm uses this information to estimate sparsity of input point cloud and adaptively change compression rate to achieve the same reconstruction error.

Keywords: adaptive compressive-sensing; point cloud; LiDAR

5.2 Introduction

In recent years, depth sensing data has been widely used in many computer vision applications, such as 3D reconstruction, augmented reality (*e.g.* Microsoft HoloLens), autonomous driving (*e.g.* Google car), collision avoidance (*e.g.* DJI drones), and gesture recognition (*e.g.* Microsoft Kinect). The ability for a machine to sense both color (RGB) and depth (Depth Map) has been shown to significantly improve image classification. The conventional method for acquiring depth information is called stereo vision. It uses two RGB cameras to obtain a pair of stereo images and then uses computational algorithms to extract depth data from those images. Although this method provides a high resolution depth map and uses simple and cheap hardware, however, it needs a large amount of computational DSP hardware. More importantly, stereo vision requires enough illumination that makes it inaccurate to use it in dark environments or poor contrast light scenarios [66]. As a result this method is not accurate for some applications like autonomous driving.

Light Detection And Ranging (LiDAR) is an emerging mechanism that uses time-of-flight (ToF) measurements to compute the distance from the laser source to the target. Although the concept of ToF measurement is very simple, because of high speed of light its hardware implementation was not possible until recent years that high speed electronics are available. There are two types of LiDAR system. A scanning LiDAR typically acquires multiple samples of a volume by scanning the region at a specified scan rate. The scanning device can be a brushless motor or a MEMS mirror. Flash LiDAR uses an array of imager in which a 2

[1] This paper submitted to the 2nd International Conference on Signal and Image Processing (ICSIP), 2017.

dimensional array of sensors (*e.g.* PD or SPAD) are used to simultaneously acquire points, just as a 2 dimensional scene is acquired with an array of detectors in a conventional camera [67]. Scanning LiDAR is generally slower than flash LiDAR but it is usually able to operate up to longer distances since all energy of the laser is focused at one point.

In either scanning or flash LiDAR the light source can be pulsed or modulated by a continuous-wave (CW) source, typically a sinusoid or square wave. Pulsed LiDAR requires a fast photo-detector, usually a single-photon avalanche diode (SPAD) and ToF can be calculated by using a Time to Digital Converter (TDC) [68]. This approach necessitates fast electronics, since achieving 1 millimeter accuracy requires timing a pulse of 6.6 picoseconds in duration. In CW Lidar, the laser is modulated by a continuous sine wave and ToF will be calculated by measuring the phase difference between transmitted and received signal. CW LiDAR doesn't need high frequency hardware but the maximum depth is limited by modulation frequency [66]. For either single-beam scanning or flash LiDAR, the resulting measurement is called a point cloud of region.

Many emerging technologies generate huge amount of data. Google self-driving car for example, gathers 750MB of sensor data per second. The LiDAR device used by Google car – a Velodyne 64 beam laser – can take up to 1.3 million data points per second. There are many efficient and standard algorithms (*e.g.* JPEG) to compress 2D images, however, unfortunately there is no standard way to compress 3D data. We can compress 3D point cloud if we convert it to a 2D gray-scale image and then apply JPEG to that 2D image. The problem is by converting 3D to 2D we will lose some data, since 3D points may be non-uniformly distributed in space. Figure 5-1 shows a sample 3D point cloud an object with 15915 points and process of converting it to 2D image. It can be seen that all 3D points in the black box should be averaged and map to a single pixel value in 2D image plane. As shown in Table 5-1 for this point cloud even with a 1024x2048 pixels image we still lose some data points. As a result, this method of compression may be acceptable for some applications but it is not suitable for applications that need all data points like high resolution 3D shape extraction.

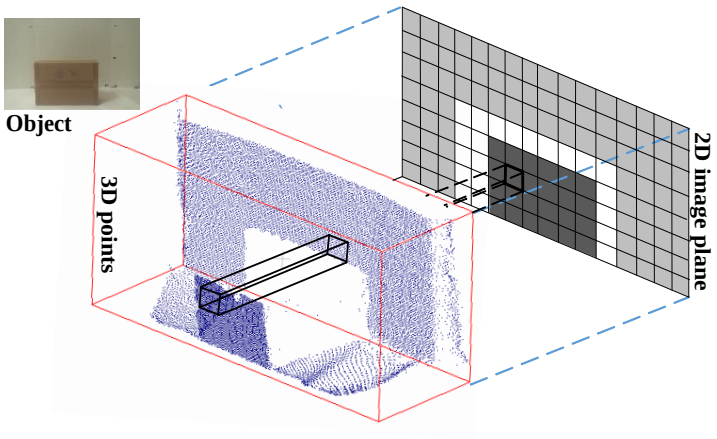

Figure 5-1: Process of converting 3D point cloud to 2D image

Table 5-1: Maximum number of 3D points mapped to a single 2D Pixel

2D Image Size	# of 2D Pixels	Max # of 3D points mapped to a single 2D pixel
64x128	8k	42
128x256	32k	22
256x512	128k	10
512x1024	512k	5
1024x2048	2M	2

A key question in any power efficient LiDAR system (*e.g.* wireless sensor applications) is how many points we need to capture to fully obtain the scene point cloud. The lower points we need to capture the lower energy will be needed to transmit this data to the receiver. Compressive-sensing is a method that enables direct compression of the LiDAR data. Previous papers [35],[36],[37] attempted to apply CS to point cloud data. Lau [69] proposes a new way of applying compressive sensing, called re-sampling compressive sampling to reduce LiDAR sampling rate required to capture 3 dimensional scenes. This method needs to generate and process a matrix with the order of n^4 elements for a $n \times n$ point-cloud that makes it computationally impractical for many applications. Hawe [70] proposes a conjugate subgradient method for solving the arising compressive-sensing problem of a depth map from a stereo vision camera. This method efficiently subsamples the depth map by randomly choosing points just from edges of the object in the depth map, however, this method is just useful for stereo vision applications that depth map is extracted from original RGB images, so the edges points are known. For ToF camera finding edge points in a complex point cloud is not trivial and this method can't be used. Liu [71] shows that stereo vision depth maps are sparser using a dictionary of both wavelet and contourlet atoms. He also proposed a new

reconstruction algorithm that results in smaller error in the reconstruction depth map. However, this method also doesn't investigate the performance on a point cloud from a ToF camera. None of these papers try to address the optimum rate for compression. As theory of CS suggests there is a relationship between sparsity of the input signal and optimum compression rate [72]. In other words, a fixed compression rate is not appropriate for CS. Instead, we need to adaptively change the rate based on the sparsity of point cloud, which gives us the idea of adaptive CS. In our previous paper [73] we applied adaptive CS to biomedical signals by calculating sparsity at the receiver node. This is an easy method since reconstruction algorithm at the receiver recovers that sparse representation of signal and we exactly calculate the sparsity and modify the compression rate based on that, however as we explained in [73], this method creates a delay in the system that may not be acceptable in some applications.

In this paper, we show that there is a correlation between sparsity of point cloud and number of edge points in that point cloud and we can use number of edge points as an estimation of sparsity and modify the compression rate based on that. Since we directly estimate sparsity at the sensor node there is no delay in this mechanism.

This paper is organized as follows. Section II introduces a basic background for compressive-sensing. Section III explains system models for adaptive CS. Section IV introduces system architecture for proposed adaptive CS of point cloud. Experimental results will be shown in Section V and finally Section VI provides conclusion.

5.3 System Models for Adaptive Compressive-Sensing

Figure 5-2 and Figure 5-3 show two possible methods for implementing adaptive compressive-sensing, using the input signal sparsity to adjust the compression factor. In the first system shown in Figure 5-3, the LiDAR sensor transmit compressed data to the cloud and reconstruction algorithm in the cloud exactly calculates the sparsity and transmits it to the sensor. The sensor then adjusts its compression factor based on this information. Because the reconstruction algorithm already has the sparse representation of signal, this system can easily implement CS rate adaption in the sensor node; however, one limitation with this system is the round trip delay, or the latency required to calculate the sparsity in the cloud and the transmission delay to send the result back to the sensor. If this latency is large, the LiDAR sensor must buffer recently acquired data until it has received the most up-to-date estimate of sparsity from the receiver. Another problem is that the calculation of sparsity is performed on

the current data block; however, this result is then used to compress the next data block, which may have a different sparsity.

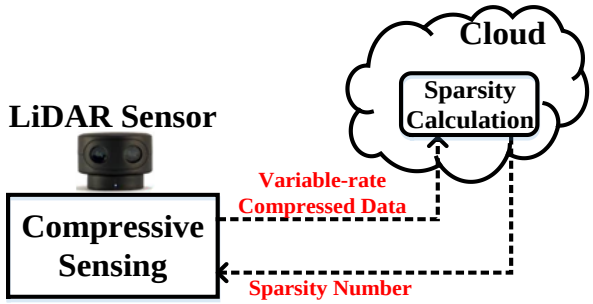

Figure 5-2: Sparsity-calculation adaptive CS system

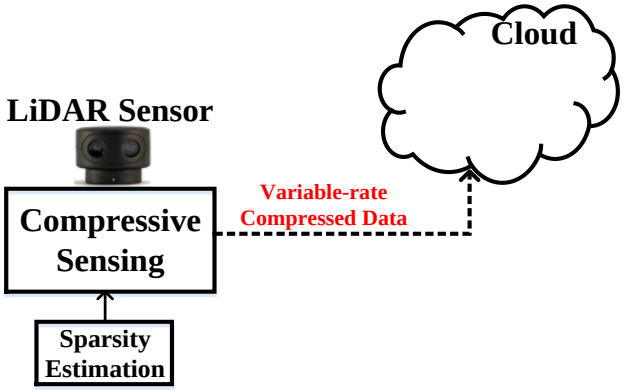

Figure 5-3: Sparsity-estimation adaptive CS system

The second system shown in Figure 5-3 estimates the sparsity of input data on the sensor node of the system. This eliminates possible problems associated with the round trip delay, and also allows for the calculated sparsity of each data block to be used in the compression of that same block. Unfortunately, this method requires additional power to operate the sparsity monitoring block, which may not be available on the energy-constrained sensor node. In addition, this system has no direct access to the sparse representation of the input signal, and therefore requires a sparsity estimation block that obtains the sparsity of the input data directly from the time domain signal. In our previous paper [73] we studied the feasibility and performance of first method. In this work we evaluate performance of second method for a different application. We show that we can use number of edge points in the point cloud to estimate sparsity of point cloud.

5.4 System Architecture

We found that there is a correlation between sparsity of point cloud data and number of edge points. Since there is no accurate algorithm to extract edge points directly from 3D points we

use a secondary sensor (RGB) to calculate this. Many LiDAR systems already has a cheap low resolution RGB camera beside LiDAR device (e.g. Google Tango). It is shown that if the LiDAR and camera are in a fixed position with respect to each other a projection matrix can convert 3D point clouds to correspondent 2D image captured by RGB camera. In other words, number of edge points in 2D image captured by RGB camera is approximately equal to number of edge points in 3D point cloud. Figure 5-4 shows the system architecture used in this project.

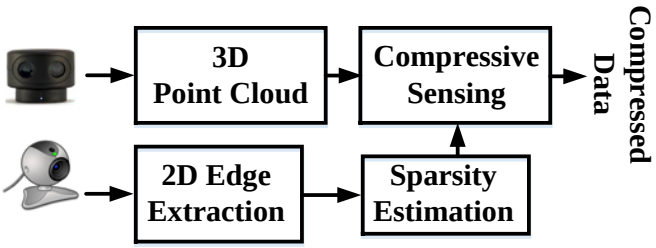

Figure 5-4: System Architecture

We used two different hardware sets to evaluate this system. The first set uses a Logitech C920 HD webcam and LiDAR-Lite TOF sensor. It is mounted on turret which performs the scan of an area in 2 dimensions. It scans the scene and provides a point cloud with 12288 points. The second set uses a Google Tango that has an RGB sensor integrated in to the device. Figure 5-5 shows these two hardware sets.

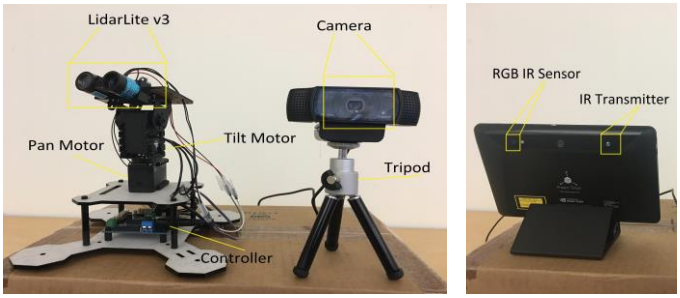

Figure 5-5: Two hardware sets used to capture data: LiDAR-Lite (Left), Google Tango (Right)

5.5 Experimental Results

We capture point cloud from six different scenes with different complexity using LiDAR-Lite and Google Tango. Then we transfer point cloud to wavelet domain and count number of significant coefficients as a measure of sparsity. We also use Canny algorithm to extract edges from 2D image and count number of edge points. The next step is applying CS to point cloud. The initial compression rate assumed to be 2.5. We continue to compress point clouds with the same rate and the goal is to achieve similar reconstruction error that is calculated as Mean

Absolute Error (MAE). When error changes significantly we change the compression rate appropriately to get the similar error as before.

Figure 5-6 and Figure 5-7 shows dataset captured by both LiDAR-Lite and Tango respectively. The first row is the 2D image captured by image sensor. The second row shows the output of edge detection algorithm on 2D image labeled with the total number of edge points. The third row shows the point cloud that is converted to 2D for better visualization and is labeled with the sparsity number of point cloud. As mentioned before this number represents number of major wavelet coefficients of point cloud data. The forth row shows reconstructed point clouds when conventional CS is used and we compress all point clouds with the fixed initial rate of 2.5. It is clear that without adaptive system the reconstruction error is so large that easily can be seen by naked eye. Finally the last row shows the reconstructed point cloud and associated compression rate to maintain the initial MAE for adaptive CS system.

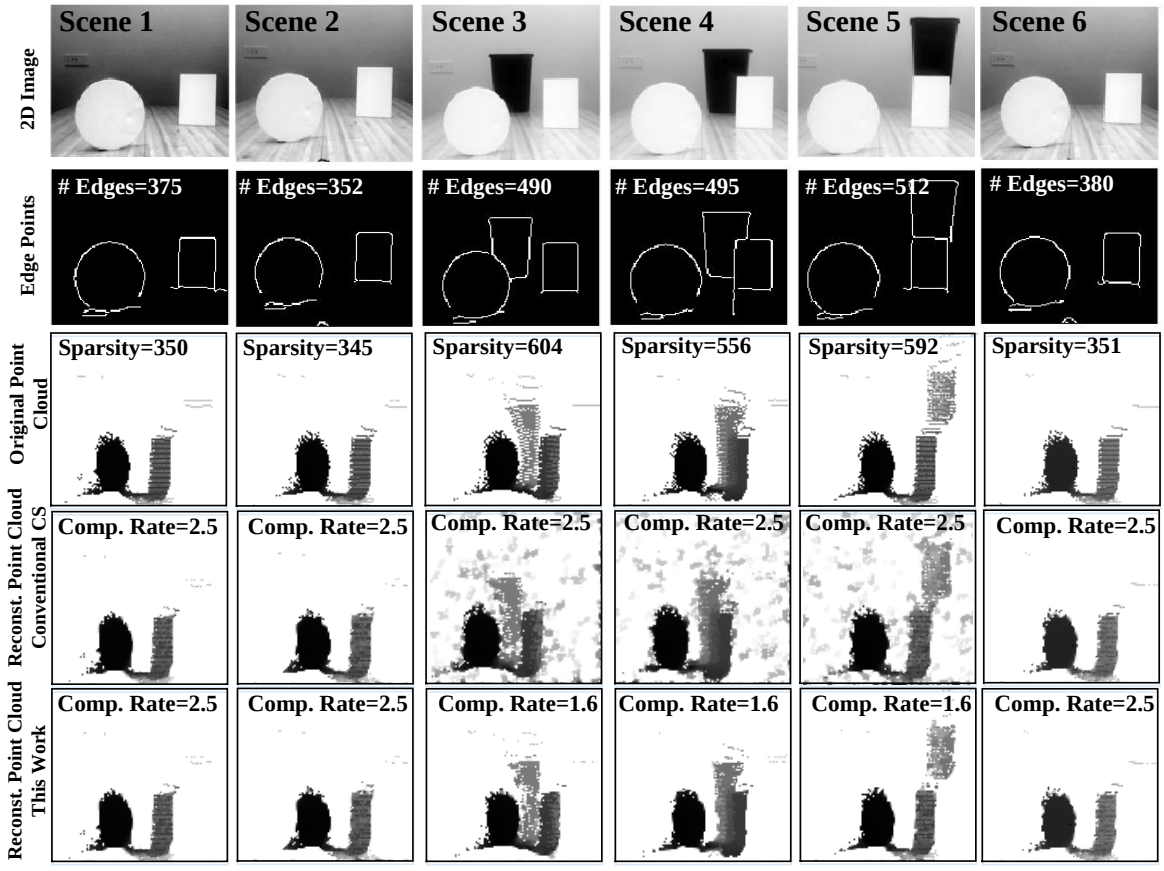

Figure 5-6: LiDAR-Lite: Six different scenes and their point clouds and related 2D image

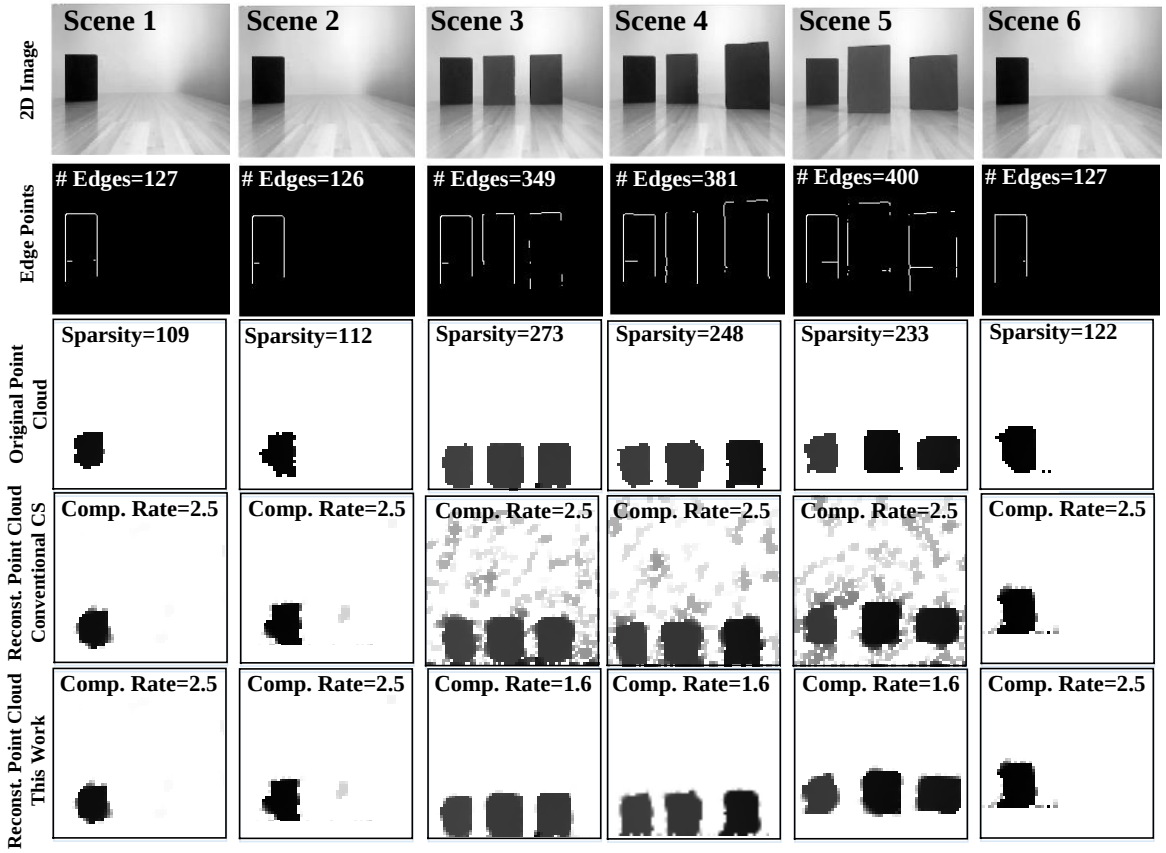

Figure 5-7: Google Tango: Six different scenes and their point clouds and related 2D image

It can be seen that sparsity significantly changes from scene 2 to 3 and we changed compression rate from 2.5 to 1.6 to keep the same MAE. At scene 6 the sparsity returns to the initial level and we can increase rate to 2.5 again. Figure 5-8 shows the sparsity of point cloud and total number of edge points in 2D image for each scene and for each hardware set. These plots first show how sparsity is correlated with the number of edge points and secondly it shows that in order to keep the same MAE, number of edge points provide information about point cloud sparsity that eventually can be used by CS algorithm to change the compression rate.

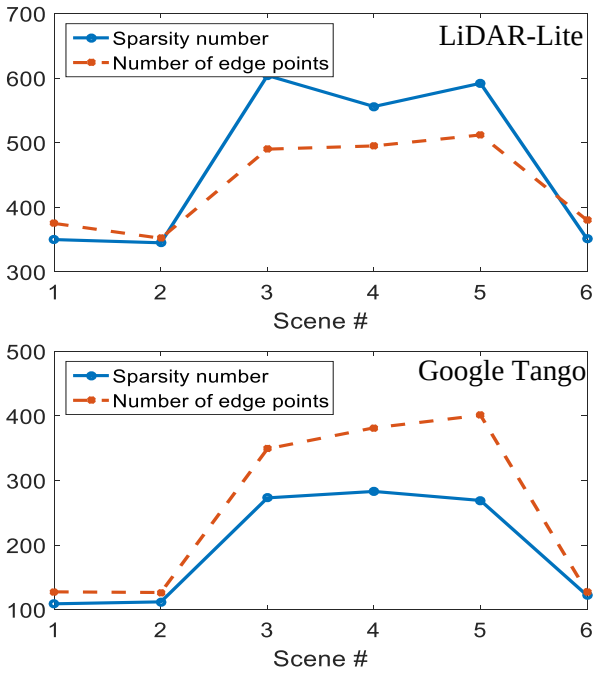

Figure 5-8: Sparsity vs number of edge points for each scene

Finally, two problems need to be addressed that are subject to future work. First problem is how we set the threshold level to calculate sparsity and number of edge points. Second problem is how to translate sparsity change to the compression rate change in order to keep the same MAE. In other words, is sparsity changes 50% from scene 3 to scene 4 how do we know that compression rate should change to 1.6 from 2.5 to achieve the same MAE. We believe some machine learning algorithms can help to solve these problems by training them with a larger set of point cloud and image training pairs.

5.6 Conclusion

In this paper we proposed a mechanism for adaptive compressive-sensing 3D point cloud. We showed that there is a correlation sparsity of point cloud and number of edge points. This correlation helps us to modify the compression rate based on complexity of 3D point cloud. We evaluated this mechanism using two different hardware sets. As experimental results showed using conventional CS can cause very large error during reconstruction. Finally further work needs to be done to find suitable threshold levels to calculate sparsity number and number of edge points automatically. We believe machine learning algorithm such as deep learning can help to find these thresholds.

5.7 References

[66] Larri Li, “Time-of-Flight Camera – An Introduction,” Technical White Paper, May 2014.

[67] www.pmdtech.com

[68] J. Nissinen, et al., “Integrated Receiver Including Both Receiver Channel and TDC for a Pulsed Time-of-Flight Laser Rangefinder With cm-Level Accuracy,” in IEEE Journal of Solid-State Circuits, vol. 44, no. 5, pp. 1486-1497, May 2009.

[69] Richard C. Lau ; T. K. Woodward, “A new approach to apply compressive sensing to LIDAR sensing,” Proc. SPIE 9109, Compressive Sensing III, May 2014.

[70] S. Hawe, M. Kleinsteuber and K. Diepold, “Dense disparity maps from sparse disparity measurements,” 2011 International Conference on Computer Vision, Barcelona, 2011, pp. 2126-2133.

[71] L. K. Liu, at al. “Depth Reconstruction From Sparse Samples: Representation, Algorithm, and Sampling,” in IEEE Transactions on Image Processing, vol. 24, no. 6, pp. 1983-1996, June 2015.

[72] Candes, E.J.; Tao, T., “Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies?,” Information Theory, IEEE Transactions on , vol.52, no.12, Dec. 2006

[73] V. Behravan, et al., “Rate-adaptive compressed-sensing and sparsity variance of biomedical signals, ” 2015 IEEE 12th International Conference on Wearable and Implantable Body Sensor Networks (BSN), Cambridge, MA, 2015, pp. 1-6.

[74] F. Chen, “Energy-efficient Wireless Sensors: Fewer Bits, More MEMS,” PhD Thesis, Dept. Electr. Eng. Comp. Sc., MIT, Cambridge, MA, USA, Sep. 2011.

6.Conclusion

This work addresses conventional CS issues including non-adaptive compression rate and offers solutions for using in low power SoCs.

In chapter 2 an adaptive compressive-sensing system was presented that enables real-time adjustment of the compression factor, depending on the variance in the sparsity number of the input signal. Also sparsity variance of different biomedical signals investigated to provide justification for implementing an adaptive compressive-sensing system. This system validated through experiments conducted on a custom embedded hardware. It was shown that this adaptive compressive-sensing system can reduce power consumption by 16.2%, without any major change on the sensor node. Despite this initial success, the criteria for which the adaptive system should change the compression factor has not been fully explored and is the subject of future work. Furthermore, the relationship between sparsity and the desired compression factor is also still unknown for real biomedical signals, and requires future research.

In chapter 3 a compressive-sensing SoC for compressing biomedical sensor data that incorporates statistics collection (SC) was presented. Is was shown that using some statistics from the input signal can dramatically improve the performance of reconstruction algorithms for signals that are sparse in the time domain. As part of future work, one can try to modify the GPSR algorithm such that it can be used for signals that are sparse in non-time domain bases.

In chapter 4 a framework to analyze compressive-sensing LiDAR system was presented. Experiments were done using different point clouds captured by a Google Tango tablet. It is shown that the maximum error generated by compressive-sensing happens at the edge points of point cloud. It is finally shown that excluding edge points from error calculation can helps us to find an optimum compression rate for each point cloud in these specific scenes.

In chapter 5 a mechanism for adaptive compressive-sensing of 3D point clouds were presented. It was shown that there is a correlation between sparsity of point cloud and number of edge points. This correlation helps us to modify the compression rate based on complexity of 3D point cloud. The proposed mechanism evaluated using two different hardware sets. As experimental results showed using conventional CS can cause very large error during reconstruction. Finally further work needs to be done to find suitable threshold levels to calculate sparsity number and number of edge points automatically. We believe machine learning algorithm such as deep learning can help to find these thresholds.

Bibliography

X. Pu, L. Wan, Y. Sheng, P. Chiang, Y. Qin, and Z. Hong, "A Wireless 8-Channel ECG Biopotential Acquisition System for Dry Electrodes", *IEEE Radio-Frequency Integration Technology Conference*, Nov. 2012.

Derek Chen, Joe Crop, Patrick Chiang, and Gabor Temes, "A 12-Bit 7uW/Channel 1 Khz/Channel Incremental ADC for Biosensor Interface Circuits", *International Symposium on Circuits and Systems (ISCAS)*, May 2012.

Robert Pawlowski, Evgeni Krimer, Joseph Crop, Jacob Postman, Nariman Moezzi-Madani, Mattan Erez, Patrick Chiang, "A 530mV 10-Lane SIMD Processor With Variation Resiliency in 45nm SOI", *International Solid-State Circuits Conference,* Feb. 2012.

J. Cheng, N. Qi, P. Y. Chiang, A. Natarajan, "A 1.3mW 0.6V WBAN-Compatible Sub-Sampling PSK Receiver in 65nm CMOS", *International Solid-State Circuits Conference*, Feb. 2014.

Candès, E.J., & Wakin, M.B., "An Introduction To Compressive Sampling," *IEEE Signal Processing Magazine*, V. 21, March 2008.

Y. C. Eldar, G. Kutyniok, "Compressed Sensing: Theory and Applications", 1st Edition, 2012.

F. Chen, "Energy-efficient Wireless Sensors: Fewer Bits, More MEMS," PhD Thesis, Department of Electrical Eng. And Comp. Scince, Massachusetts Inst. Technol., Cambridge, MA, USA, Sep. 2011.

J. Cheng; et al., "A near-threshold, multi-node, wireless body area sensor network powered by RF energy harvesting," Custom Integrated Circuits Conference (CICC), 2012 IEEE , vol., no., pp.1,4, 9-12 Sept. 2012

S.K. Mukhopadhyay, M. Mitra, "An ECG data compression method via R-Peak detection and ASCII Character Encoding," Computer, Communication and Electrical Technology (ICCCET), 2011 International Conference on , vol., no., pp.136,141, 18-19 March 2011.

F. Chen, A.P. Chandrakasan, V. Stojanovic, "A signal-agnostic compressed sensing acquisition system for wireless and implantable sensors," Custom Integrated Circuits Conference (CICC), 2010 IEEE , vol., no., pp.1,4, 19-22 Sept. 2010.

D. Donobo, "Compressed Sensing," Trans. on Information Theory, 2006.

Candes, E.J.; Tao, T., "Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies?," *Information Theory, IEEE Transactions on* , vol.52, no.12, pp.5406,5425, Dec. 2006

A. L. Goldberger, et al., "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals,[Circulation Electronic Pages; http://circ.ahajournals.org/cgi/content/full/101/23/e215], 2000.

D. Gangopadhyay, et al., "Compressed Sensing Analog Front-End for Bio-Sensor Applications," Solid-State Circuits, IEEE Journal of , vol.49, no.2, pp.426,438, Feb. 2014.

O. Abari, F. Chen, F. Lim, V. Stojanovic, "Performance trade-offs and design limitations of analog-to-information converter front-ends," Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, March 2012.

S. Miaou and S. Chao, "Wavelet-Based Lossy-to-Lossless ECG Compression in a Unified Vector Quantization Framework," IEEE Transactions on Biomedical Engineering, vol. 52, pp. 539-543, 2005.

S. Aviyente, "Compressed Sensing Framework for EEG Compression," IEEE 14th Workshop on Statistical Signal Processing, 2007.

M. Lopes, “Estimating Unknown Sparsity in Compressed Sensing,” International Conference on Machine Learning (ICML), 2013.

M. Aharon, M. Elad, A. Bruckstein, “K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation,” Signal Processing, IEEE Transactions on, vol.54, no.11, pp.4311,4322, Nov. 2006.

V. Behravan, et al., “Rate-Adaptive Compressed-Sensing and Sparsity Variance of Biomedical Signals,” Body Sensor Networks 2015 (to appear).

D. Gangopadhyay, et al., “Compressed Sensing Analog Front-End for Bio-Sensor Applications,” IEEE J. Solid-State Circuits, vol.49, no.2, pp.426,438, Feb. 2014.

C.H. Chen, et al., “A 11 µW 250 Hz BW two-step incremental ADC with 100 dB DR and 91 dB SNDR for integrated sensor interfaces,” Custom Integrated CCkts. Conf., Sep. 2014.

Larri Li, “Time-of-Flight Camera – An Introduction,” Technical White Paper, May 2014.

J. Nissinen, I. Nissinen and J. Kostamovaara, “Integrated Receiver Including Both Receiver Channel and TDC for a Pulsed Time-of-Flight Laser Rangefinder With cm-Level Accuracy,” in IEEE Journal of Solid-State Circuits, vol. 44, no. 5, pp. 1486-1497, May 2009.

Richard C. Lau ; T. K. Woodward, “A new approach to apply compressive sensing to LIDAR sensing,” Proc. SPIE 9109, Compressive Sensing III, May 2014.

S. Hawe, M. Kleinsteuber and K. Diepold, “Dense disparity maps from sparse disparity measurements,” 2011 International Conference on Computer Vision, Barcelona, 2011, pp. 2126-2133.

L. K. Liu, S. H. Chan and T. Q. Nguyen, “Depth Reconstruction From Sparse Samples: Representation, Algorithm, and Sampling,” in IEEE Transactions on Image Processing, vol. 24, no. 6, pp. 1983-1996, June 2015.

Jan Nissinen, et al, “Integrated Receiver Including Both Receiver Channel and TDC for a Pulsed Time-of-Flight Laser Rangefinder With cm-Level Accuracy”, JSSC 2009.

Jihyun Cho, et al, “A 3-D Camera With Adaptable Background Light Suppression Using Pixel-Binning and Super-Resolution”, JSSC 2014.

A. Kirmani, A. Colaço, F. N. C. Wong and V. K. Goyal, "CoDAC: A compressive depth acquisition camera framework," *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, 2012

Robert Moss, et al., Low-cost compact MEMS scanning LADAR system for robotic Applications, SPIE 2012.

V. Milanovic, et al., “MEMSEYE” FOR OPTICAL 3D TRACKING AND IMAGING APPLICATIONS, Transducers 2011 Conference

C. D. Mutto, et al., “Time-of-Flight Cameras and Microsoft Kinect”, Springer, 2013.

A. Kadambi and P. T. Boufounos, "Coded aperture compressive 3-D LIDAR," *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, South Brisbane, QLD, 2015.

A. Kadambi, et al., “Coded time of flight cameras: sparse deconvolution to address multipath interference and recover time profiles”, ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2013.

A. Kadambi, et al., “Frequency Domain TOF: Encoding Object Depth in Modulation Frequency”, arXiv, 2015.

A. Bhandari, et al., “Resolving Multi-path interference in Time-of-Flight Imaging via Modulation Frequency Diversity and Sparse Regularization”, Optics Letter, Volume 39, Issue 06, 2014.

A. Kadambi, et al., “3D Depth Cameras in Vision: Benefits and Limitations of the Hardware”, Springer, 2014.

R. Lange and P. Seitz, "Solid-state time-of-flight range camera," in IEEE Journal of Quantum Electronics, vol. 37, no. 3, pp. 390-397, Mar 2001.

O. Shcherbakova, L. Pancheri, G. F. D. Betta, N. Massari and D. Stoppa, "3D camera based on linear-mode gain-modulated avalanche photodiodes," *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, San Francisco, CA, 2013.

S. Foix, G. Alenya and C. Torras, "Lock-in Time-of-Flight (ToF) Cameras: A Survey," in *IEEE Sensors Journal*, vol. 11, no. 9, pp. 1917-1926, Sept. 2011.

J. Castorena and C. D. Creusere, "Compressive sampling of LIDAR: Full-waveforms as signals of finite rate of innovation," *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, Bucharest, 2012.

J. Castorena, C. D. Creusere and D. Voelz, "Modeling lidar scene sparsity using compressive sensing," *2010 IEEE International Geoscience and Remote Sensing Symposium*, Honolulu, HI, 2010.

S. Laky, P. Zaletnyik, C. Toth and B. Molnar, "Sparse representation of full waveform lidar data," *2012 IEEE International Geoscience and Remote Sensing Symposium*, Munich, 2012.

C. Niclass, M. Soga, H. Matsubara, M. Ogawa and M. Kagami, "A 0.18- um CMOS SoC for a 100-m-Range 10-Frame/s 200×96-Pixel Time-of-Flight Depth Sensor," in *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, pp. 315-330, Jan. 2014.

Y. Wang, D. Ewert, D. Schilberg and S. Jeschke, "Edge extraction by merging 3D point cloud and 2D image data," *2013 10th International Conference and Expo on Emerging Technologies for a Smarter World (CEWIT)*, Melville, NY, 2013, pp. 1-6.

J. A. Christian, S. Cryan, “A Survey of LIDAR Technology and its Use in Spacecraft Relative Navigation”, AIAA Guidance, Navigation, and Control (GNC) Conference Boston, MA, 2013.

T. Kahlmann, et al., “Calibration for Increased Accuracy of the Range Imaging Camera SwissRangerTM”, Proceedings of the ISPRS Com. V Symposium, 2006.

C. Uriarte, et al., “Modelling Distance Nonlinearity in ToF Cameras and Correction Based on Integration Time Offsets”, CIARP 2011, Springer.

Richard C. Lau ; T. K. Woodward, “A new approach to apply compressive sensing to LIDAR sensing,” Proc. SPIE 9109, Compressive Sensing III, May 2014.

S. Hawe, M. Kleinsteuber and K. Diepold, “Dense disparity maps from sparse disparity measurements,” 2011 International Conference on Computer Vision, Barcelona, 2011, pp. 2126-2133.

L. K. Liu, at al. “Depth Reconstruction From Sparse Samples: Representation, Algorithm, and Sampling,” in IEEE Transactions on Image Processing, vol. 24, no. 6, pp. 1983-1996, June 2015.

Candes, E.J.; Tao, T., “Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies?,” Information Theory, IEEE Transactions on , vol.52, no.12, Dec. 2006

V. Behravan, et al., “Rate-adaptive compressed-sensing and sparsity variance of biomedical signals,” 2015 IEEE 12th International Conference on Wearable and Implantable Body Sensor Networks (BSN), Cambridge, MA, 2015, pp. 1-6.