CHAPTER 2

LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then next steps is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system. A literature review is a body of text that aims to review the critical points of current knowledge including substantive findings as well as theoretical and methodological contributions to a particular topic.

2.1 Data Compression

In computer science and information theory, data compression, source coding, or bit-rate reduction involves encoding information using fewer bits than the original representation. Compression can be either lossy or lossless. Lossless compression reduces bits by identifying and eliminating statistical redundancy. No information is lost in lossless compression. Lossy compression reduces bits by identifying unnecessary information and removing it. The process of reducing the size of a data file (text, audio, video, etc.) is popularly referred to as data compression, although its formal name is source coding.

There are two major categories of compression algorithms: lossy and lossless. Lossy compression algorithms involve the reduction of a file’s size usually by removing small details that require a large amount of data to store at full fidelity. In lossy compression, it is impossible to restore the original file due to the removal of essential data. Lossy compression is most commonly used to store image and audio data, and it can achieve very high compression ratios through data removal. Lossless data compression is the size reduction of a file, such that a decompression function can restore the original file exactly with no loss of data. Lossless data compression is used ubiquitously in computing, from saving space on your personal computer to sending data over the web, communicating over a secure shell, or viewing a PNG or GIF image.

The basic principle that lossless compression algorithms work on is that any non-random file will contain duplicated information that can be condensed using statistical modeling techniques that determine the probability of a character or phrase appearing. These statistical models can then be used to generate codes for specific characters or phrases based on their probability of occurring, and assigning the shortest codes to the most common data. Such techniques include entropy encoding, run-length encoding, and compression using a dictionary. Using these techniques and others, an 8-bit character or a string of such characters could be represented with just a few bits resulting in a large amount of redundant data being removed.

Given this vast amount of different techniques, there are different ways how to classify compression techniques:

- With respect to the type of data to be compressed
- With respect to the target application area
- With respect to the fundamental building blocks of the algorithms used

2.2 Brief History of Compression Techniques

Data compression has only played a significant role in computing since the 1970s, when the Internet was becoming more popular and the Lempel-Ziv algorithms were invented, but it has a much longer history outside of computing. Retracing history, it can be observed that the first ever form of data compression was introduced in 1838 in Morse code for telegraphy. In the English language, most frequently occurring letters such as “e” and “t” are given shorter Morse codes.

The development of Information theory in 1940s ignited drastic evolution of numerous techniques in the field of data compression. In 1949, Claude Shannon and Robert Fano devised a technique for compression by assigning codes words based on probabilities of blocks in the data to be compressed. These compressions were confined to hardware implementations. In the 50ies of the 20th century compression technologies exploiting statistical redundancy were developed – bit-patterns with varying length were used to represent individual symbols according to their relative frequency.

During mid-1970s, Huffman put forward the technique of dynamic updation of code words based on accurate data. The key difference between Shannon-Fano coding and Huffman coding is that in the former the probability tree is built bottom-up, creating a suboptimal result, and in the latter it is built top-down.

In 1977, Abraham Lempel and Jacob Ziv published their groundbreaking LZ77 algorithm, the first algorithm to use a dictionary to compress data. More specifically, LZ77 used a dynamic dictionary oftentimes called a sliding window.

During mid-1980s, the pioneering work done by Terry Welch led to innovation of Lempel–Ziv–Welch (LZW) algorithm, which later became the most popular algorithm for many general purpose compression systems. This has been used for implementation in programs like PKZIP and also used in hardware devices like modems.

Compressive Sensing (CS) is an innovative process of acquiring and reconstructing a signal that is sparse or compressible. Around 2004 Emmanuel Candès, Terence Tao and David Donoho discovered important results on the minimum number of data needed to reconstruct an image even though the Nyquist–Shannon criterion, would deem the number of data insufficient.

2.3 Different Sampling Techniques

In signal processing, **sampling** is the reduction of a continuous signal to a discrete signal. A common example is the conversion of a sound wave (a continuous signal) to a sequence of samples (a discrete-time signal). A **sample** refers to a value or set of values at a point in time and/or space.

Nyquist Shannon Sampling Theory

The Nyquist Shannon Sampling theory is the conventional theory used for sampling any given signal.

In 1949, Shannon presented his famous proof that any band- limited time-varying signal with ‘n’ Hertz highest frequency component can be perfectly reconstructed by sampling the signal at regular intervals of at-least 1/2n seconds. In traditional signal processing techniques, we uniformly sample data at Nyquist rate, prior to transmission, to generate ‘n’ samples. These samples are then compressed to ‘m’ samples; discarding n-m samples.

At the receiver end, decompression of data takes place to retrieve ‘n’ samples from ‘m’ samples. The paradigm of Shannon’s sampling theory is cumbersome when extended to the emerging wide-band signal systems since high sampling rates may not be viable for implementation in circuitry: high data- rate A/D converters are computationally expensive and require more storage space.

The sampling theorem specifies that to avoid losing information when capturing the signal, the sampling rate must be at least twice the signal bandwidth. In many applications the Nyquist rate is so high that too many samples result, making compression a necessity prior to storage or transmission.

Sample-then-compress framework has three inefficiencies: large *N (number of samples)*, all *N* coefficients must be computed, and the locations of large coefficients must be encoded. In other applications due to large signal bandwidth it is difficult to sample at Nyquist rate.

After reviewing the conventional sampling theorem one may wonder: why should we go through all computation when we only need ‘m’ samples in the end for trans- mission? Are the real world signals always band limited? How can we get ‘n’ samples efficiently, especially if we need a separate hardware sensor for each sample? The alternative theory of compressive sensing by Candes, Tao, Romberg and Donoho has made a significant contribution to the body of signal processing literature, by giving sampling theory a new dimension.

Compressive Sensing

Compressive sensing is a method to capture and represent compressible signals at a rate significantly below the Nyquist rate. Compressive sensing, also known as compressed sensing or sparse sensing or CS, is a novel sensing/sampling paradigm that goes against the common wisdom in data acquisition. CS theory asserts that one can recover certain signals and images from far fewer samples or measurements than tradition- al methods use. To make this possible, CS relies on two principles: *sparsity*, which pertains to the signals of interest, and *incoherence*, which pertains to the sensing modality.

The following figure (Figure 2.1) represents concept of traditional data sampling and compressive sensing. Further elaboration is done in subsequent sections.

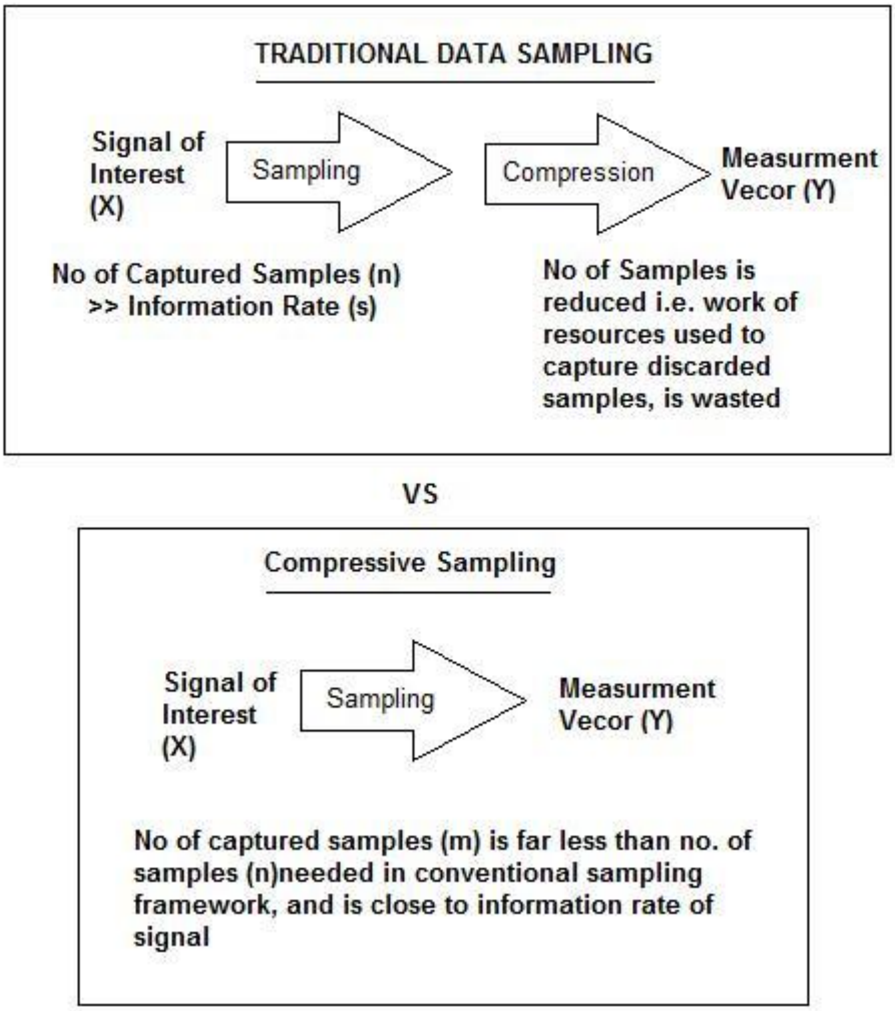
TRADITIONAL DATA SAMPLING
Signal of Interest (X)
Sampling
Compression
Measurment Vecor (Y)
No of Captured Samples (n) >> Information Rate (s)
No of Samples is reduced i.e. work of resources used to capture discarded samples, is wasted
VS
Compressive Sampling
Signal of Interest (X)
Sampling
Measurment Vecor (Y)
No of captured samples (m) is far less than no. of samples (n)needed in conventional sampling framework, and is close to information rate of signal

Figure. 2.1 Traditional Data Sampling and Compression versus Compressive Sensing

2.4 Compressive Sensing

Compressive sensing theory asserts that we can recover certain signals from fewer samples than required in Nyquist paradigm. This recovery is exact if signal being sensed has a low information rate (means it is sparse in original or some transform domain). Number of samples needed for exact recovery depends on particular reconstruction algorithm being used. If signal is not sparse, then recovered signal is best reconstruction obtainable from s largest coefficients of signal. CS handles noise gracefully and reconstruction error is bounded for bounded perturbations in data.

CS relies on two principles: ***sparsity,*** which pertains to the signals of interest, and ***incoherence***, which pertains to the sensing modality.

- **Sparsity:** Natural signals such as sound, image or seismic data can be stored in compressed form, in terms of their projection on suitable basis $\boldsymbol{\psi}$. When basis is chosen properly, a large number of projection coefficients are zero or small enough to be ignored. If a signal has only s non-zero coefficients, it is said to be s-Sparse. If a large number of projection coefficients are small enough to be ignored, then signal is said to be compressible. Sparsity expresses the idea that the "information rate" of a continuous time signal may be much smaller than suggested by its bandwidth, or that a discrete-time signal depends on a number of degrees of freedom, which is comparably much smaller than its (finite) length.
- **Incoherence:** Incoherence extends the duality between time and frequency and expresses the idea that objects having a sparse representation in $\boldsymbol{\psi}$ must be spread out in the domain in which they are acquired, just as a Dirac or a spike in the time domain is spread out in the frequency domain. Put differently, incoherence says that unlike the signal of interest, the sampling/sensing waveforms have an extremely dense representation in $\boldsymbol{\psi}$. Coherence measures the maximum correlation between any two elements of two different matrices. These two matrices might represent two different basis representation do- mains. If ψ is an $n \times n$ matrix with $\psi_1, \psi_2 \ldots \psi_n$ as columns and ϕ is an $m \times n$ matrix with $\phi_1, \phi_2 \ldots \phi_m$ as rows then coherence μ is defined as

$$\boldsymbol{\mu(\phi, \psi) = \sqrt{n}\ max \,|<\ \phi_k\,, \psi_j >|}$$

For, $1 \leq j \leq n$

$1 \leq k \leq m$

It follows from linear algebra that:

$$\boldsymbol{1 \leq \mu(\phi, \psi) \leq \sqrt{n}}$$

Within the CS framework, low coherence between $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ translates to fewer samples required for reconstruction of signal. An example of low coherence measurement - representation basis pair is sinusoids and spikes that are incoherent in any dimension, and can be used for compressively sensing signals having sparse representation in terms of sinusoids.

Restricted Isometry Property (RIP)

Restricted Isometry Property has been the most widely used tool for analyzing the performance of CS recovery algorithms; a key notion that has proved to be very useful to study the general robustness of CS.

Definition

For each integer s=1,2,3…. define the isometric constant δ_s of the matrix ϕ as the smallest number such that,

$$(\boldsymbol{1}-\boldsymbol{\delta_s})||\boldsymbol{x}||_{\boldsymbol{l2}}^{\boldsymbol{2}} \leq ||\boldsymbol{\phi_x}||_{\boldsymbol{l2}}^{\boldsymbol{2}} \leq (\boldsymbol{1}+\boldsymbol{\delta_s})||\boldsymbol{x}||_{\boldsymbol{l2}}^{\boldsymbol{2}}$$

holds for all s-sparse vectors. A vector is said to be s-sparse if it has at most s non-zero entries. Then, the matrix ϕ is said to satisfy the *s*-restricted isometric property with restricted isometric constant δ_s.We will loosely say that a matrix ϕobeys the RIP of order S ifδ_s is not too close to one. When this property holds, ϕ approximately preserves the Euclidean length of S-sparse signals, which in turn implies that S-sparse vectors cannot be in the null space of ϕ.

2.5 Reconstruction of Sparse Signals

A nonlinear algorithm is used in CS, at receiver end to reconstruct original signal. This nonlinear reconstruction algorithm requires knowledge of a representation basis (original or transform domain) in which signal is sparse (exact recovery) or compressible (approximate recovery). Signal of interest or the raw signal can be regarded as a vector $'\boldsymbol{f}'$with millions of components. We assume that '***f***' can be represented as a linear combination of certain *basis functions*:

$$\boldsymbol{f} = \boldsymbol{\psi}\,\boldsymbol{c}$$

The basis functions must be suited to a particular application. Here, $\boldsymbol{\psi}$ is the discrete cosine transform.We also assume that most of the coefficients $\boldsymbol{c}$ are effectively zero, so that $\boldsymbol{c}$ is sparse.

Sampling the signal involves another linear operator,

$$\boldsymbol{b} = \boldsymbol{\phi}\,\boldsymbol{c}$$

Here, $'\boldsymbol{b}'$ is a few random samples of $'\boldsymbol{f}'$, so $'\boldsymbol{\phi}'$ is a subset of the rows of the identity operator. But more complicated sampling operators are possible. They include:

- Random Fourier sub-matrix
- Fast Johnson-Linden Strauss transform
- Randomly chosen matrix

To reconstruct the signal, we must try to recover the coefficients by solving

$$\boldsymbol{A\,x = b}$$

Where $\boldsymbol{A = \phi\,\psi}$

Once we have the coefficients, we can recover the signal itself by computing

$$\boldsymbol{f = \psi\,x}$$

Since this is a compression technique, $\boldsymbol{A}$ is rectangular, with many more columns than rows. Computing the coefficients $\boldsymbol{x}$ involves solving an underdetermined system of simultaneous linear equations, $\boldsymbol{A\,x = b}$. In this situation, there are many more unknowns than equations.

The key to the almost magical reconstruction process is to impose a ***nonlinear*** regularization involving the ***l_1*** **norm**. This is a variant of the Convex Relaxation reconstruction scheme. The “Manhattan” norm, ***l_1***, is named after travel time along the square grid formed by city streets.

l_1: norm(x,1) = sum(abs(x))

In principle, computing this reconstruction should involve counting non-zeros with l_0. This is a combinatorial problem whose computational complexity makes it impractical. It is NP-hard. However, l_0 can be replaced by l_1 as the two problems have the same solution. The l_1 computation is practical because it can be posed as a linear programming problem and solved with the traditional simplex algorithm or modern interior point methods.

Another effective reconstruction scheme is a variant of *Greedy Iterative method* called as the ***Orthogonal Matching Pursuit (OMP)***. It constructs an approximation by going through an iteration process. In each iteration, the locally optimum solution is determined by finding the column vector of $\boldsymbol{A}$ which is most correlated with the residual vector ***r***. Initially the residual vector is equal to the vector that is to be approximated i.e. ***r=b*** and it is adjusted at each iteration to take into account the previously chosen vector.

OMP is a stepwise forward selection algorithm and is easy to implement. A key component of OMP is the stopping rule, which depends on the noise structure. In the noiseless case the stopping rule is that the residual becomes zero i.e. $\boldsymbol{r_i = 0}$. The algorithm stops when this condition is achieved.

Another class of algorithms with low computational complexity is the *Iterative Thresholding schemes*. ***Approximate Message Passing (AMP)*** is a variant of this scheme. AMP reconstructs the signal as effectively as $\boldsymbol{l_1}$while running much faster. The idea behind these algorithms is that when a signal is represented in terms of a suitable basis, smaller coefficients are set to zeroes while the larger coefficients above a given threshold are possibly shrunk.

In every iteration, a residual is calculated along with a new threshold. With every step, these values change and the algorithm breaks when the stopping condition is reached. As in the case of OMP, the stopping rule is dependent on the noise structure. For the noiseless case, the stopping rule is that the residual reaches zero.

2.6 MATLAB

The name MATLAB stands for MATrix LABoratory. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects. MATLAB is a high-level language and interactive environment for numerical computation, visualization, and programming. Using MATLAB, you can analyze data, develop algorithms, and create models and applications. The language, tools, and built-in math functions enable you to explore multiple approaches and reach a solution faster than with spreadsheets or traditional programming languages, such as C/C++ or Java.

2.6.1Introduction to MATLAB

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran. MATLAB can be used for a range of applications, including signal processing and communications, image and video processing, control systems, test and measurement, computational finance, and computational biology.

2.6.2 MATLAB Environment

The MATLAB environment (on most computer systems) consists of menus, buttons and a writing area similar to an ordinary word processor. There are plenty of help functions that you are encouraged to use. The writing area that you will see when you start MATLAB is called the *command window*. In this window you give the commands to MATLAB.

In the command window you will see a prompt that looks like >> . You type your commands immediately after this prompt. Once you have typed the command you wish MATLAB to perform, press <enter>. If you want to *interrupt a command* that MATLAB is running, press<ctrl> + <c>.

The commands you type in the command window are stored by MATLAB and can be viewed in the *Command History* window. To repeat a command you have already used, you can simply double-click on the command in the history window, or use the <up arrow> at the command prompt to iterate through the commands you have used until you reach the command you desire to repeat.

2.6.3 Obtaining Help in MATLAB

To obtain help on any of the MATLAB commands, you simply need to type

help <command>

at the command prompt. For example, to obtain help on the gamma function, we type at the command prompt:

help gamma

You may also get help about commands using the "Help Desk", which can be accessed by selecting the MATLAB Help option under the Help menu.

Note that the description MATLAB returns about the command you requested help on contains the command name in ALL CAPS. This does not mean that you use this command by typing it in ALL CAPS. In MATLAB, you almost always use all lower case letters when using a command.

2.6.4 Advantages of MATLAB

- MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment.
- MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. These factors make it an excellent tool for teaching and research.
- MATLAB is an interactive system whose basic data element is an array that does not require dimensioning.
- It has powerful built-in routines that enable a very wide variety of computations.
- It also has easy to use graphics commands that make the visualization of results immediately available.
- Specific applications are collected in packages called as *toolbox.* There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization and other fields of applied science and engineering.