# TABLE OF CONTENTS

# LIST OF TABLES

**Table Number** **Page**

# LIST OF FIGURES

**Figure Number** **Page**

## LIST OF ACRONYMS

| Acronym | Meaning |
|---|---|
| CS | Compressive Sensing |
| LDS | Linear Dynamical Systems |
| 2D | Two Dimensional |
| SNR | Signal to Noise Ratio |
| SPC | Single Pixel Camera |
| SMC | Spatial Multiplexing Camera |
| TMC | Temporal Multiplexing Camera |
| P2C2 | Programable Pixel Compressive Camera |
| OMP | Orthogonal Matching Pursuit |
| CoSAMP | Compressive Sampling Matching Pursuit |
| N4SID | Subspace State Space System Identification |
| PCA | Principal Component Analysis |
| LED | Light Emitting Diode |
| DynTex | Dynamic Texture |
| DCT | Discrete Cosine Transform |
| SVD | Singular Value Decomposition |
| i.i.d | Independent Identically distributed |

# CHAPTER 1

# INTRODUCTION

## 1.1. Background

Nyquist theorem is the cardinal principle of modern day digital communication systems. The ground-breaking theory postulates that to reconstruct any signal accurately and unambiguously it has to be sampled at rate that is twice the highest frequency component of the signal. This criterion put strict constraints in higher frequency spectrum sensing. Images and videos have highly redundant contents that can be compressed using compression algorithms. The success of compression algorithms led to the question, why sample at twice the rate when during compression most of it will simply be discarded? This question was finally answered by scientists in the form of introduction of new technique for sensing, called the Compressive Sensing(CS) in 2006. Compressive Sensing, since its introduction has seen massive research in various areas of application including image and videos sensing at a rate much lower than the dictates of Nyquist criteria. In this novel technique, compression and sensing is done in a single step. Single step Compression and sensing plays vital role in terms of saving storage space and reduction of transmission time. Another benefit of the technique is its independent reconstruction algorithms that facilitate signal processing, image or video reconstruction at the receiver side independent of the time of sensing.

Like the Nyquist that relies on knowledge of highest frequency component, the compressive sensing theory rests on the assumption that the signal is either itself

sparse or is sparse in another transform basis like the Discrete Cosine or the Fourier etc.

Signal reconstruction in compressive sensing is governed by two conditions: Sparsity is the first condition which requires the signal to be sparse in some domain. Incoherence is the second condition which is applied through the Restricted Isometric Property (RIP) [1][2].

Generally, Compressive sensing takes weighted linear combination of samples also called compressive measurements in a basis different from the basis in which the signal is known to be sparse. Emmanuel Candès et al, showed that number of the compressive measurements can be small and still contain nearly all the useful information required to reconstruct the image. The task of reconstructing the image into requisite domain requires solving a system of underdetermined linear equations. It is because the number of compressive measurements is smaller than the number of pixels in the full image. The restriction that the initial signal is sparse, enables solution of the underdetermined system. Sparsity is achieved by minimizing the number of nonzero components of the solution. The $l_0$-norm was defined as the function counting the number of non-zero components of a vector by Donoho, Candès et al. However, it was also proved by them that for many problems $l_1$-norm is equivalent to the $l_0$-norm. A linear program is thus a much easier and efficient way to determine $l_1$-norm, compared with $l_0$-norm, for which efficient solution methods already exist. Over the years many algorithms like Basis Pursuits have been developed that are faster than linear programming and also preserve sparsity even in noisy environment.

Sampling under the Nyquist differs from CS in three important aspects. First of all, it considers infinite length and continuous-time signals as compared with CS, that is based on mathematical theory for computing vectors having finite dimensions in $\mathbb{R}^n$ . Second, instead of using time based sampling of the signal as in case of Nyquist, inner products between the signal and more general test functions is used to obtain measurements in the CS systems. Lastly, the methods of signal recovery differentiate the two paradigms. Nyquist protocol employs sinc interpolation for signal recovery based on a linear process with little computation and has a simple interpretation. In CS, however, signal recovery is typically achieved using highly nonlinear methods.

The applications of Compressive Sensing in image and signal processing are immense. Apart from image processing the technique is a major contender for the future 5G telecommunication networks.

Compressive sensing has been successfully applied to still images, medical imaging, and sensors networks, however, its application in video sensing is still under intensive research.

### 1.2. Motivation of Research

This research is intended to develop a novel algorithm for compressively sensing videos by modelling them as linear dynamical systems. The two of the major factors that are the motivation behind this research are success of compression algorithms and characterization of dynamic textures as linear dynamical systems. Firstly, the sensing videos compressively is motivated by the success of video compression algorithms that indicate that videos are high

redundant. Compressively sensing video in a single step implies that sensing and compression can be done in a single step significantly reducing the amount of data. It can thus lead to compelling new camera or sensors design, especially in domains where sensing is inherently costly like the far infrared. Secondly, dynamic textures, activity, and video clustering have been successfully modelled as Linear Dynamical Systems (LDSs). It notably reduces the number of required parameters to be estimated by offering low dimensional representations for otherwise high-dimensional videos, thus reducing the amount of data that is needed to be sensed. LDSs characterise the video signal as combination of time-varying and time-invariant parameters. Its addresses the ephemeral nature of videos to a large extent due to its generative nature that provides a prior for the evaluation of the video in both forward and reverse time. The combination of the two techniques thus results into an effective compressive sensing video protocol that addresses both ephemeral and compressive nature of video signals.

### 1.3. Problem Statement

Presently, there are many compressive video sensing techniques that have tried to address the fundamental problems being faced by videos sensed through Spatial Multiplexing Cameras (SMC). First, ephemeral nature, i.e., scene changes with each compressive measurement. Second, dimensionality of videos is much higher as compared to static images. In case of SPCs it gets more complex as it has only one sensor. This imperative dimension is missing from almost all approaches for CS-based video recovery so far. The proposed methods, appropriate for general scenes, consider scenes as time-invariant series of frames

(i.e. video) instead of continuously changing scenes. Some are designed specifically for time-varying periodic scenes and few exploit optical flows in CS architecture based TMCs. Multi-scale sensing and sensing secluded pieces of each frame rely on static models and ignore time –varying nature of videos. The purpose is to define a frame work for CS videos that can address the ephemeral nature of video.

### 1.4. Objectives of Research

This research inteds to review existing literature of the research work already accomplished on compressive sensing of videos and then to propose a novel protocol for compressively sensing videos modelled as linear dynamical system(LDS). It intends to evaluate its performance like Compression Ratio (CR), reduction in measurment rate compared with Nyquist rate, noise robustness as reconstruction SNR against similar i.e.SMC based CS video algorithms.

### 1.5. Organization of Thesis Document

The thesis comprises of five chapters. Chapter 1 introduces the subject, its novelty, importance, some research background, and motivations of research. It also concisely states the problem and clearly mentions scopes of research.

Chapter 2 deals with the brief literature review of current research in the field of Compressive Sensing, its applications in image and video sensing and reconstruction, video compressive sensing methodologies and linear dynamical systems.

Chapter 3 discusses the modalities of compressively sensing videos by modelling them as linear dynamical systems.

Experimental results and simulation results based on the proposed algorithm are presented in chapter 4 along with the analyses and performance comparison with another CS video algorithm. The thesis is concluded in chapter 5 with recommendation for future direction of work.

### 1.6. Notations used in the thesis:

Matrices, vectors and scaler quantities are denoted by boldface italic uppercase, boldface italic lowercase and normal italic letters respectively. $[\boldsymbol{z}]_{\boldsymbol{t}}$ represent the value of $\boldsymbol{z}$ at time *t.* $\boldsymbol{\Phi}$, $\boldsymbol{\Psi}$ are the measurement matrix and sparsifying matrix respectively. $\mathbb{R}$ represents set of real numbers. In $[\mathbb{R}]^{M\times N}$ here *M*x*N* is the order of the matrix having elements from real numbers. *K* is a number of non-zero elements in a sparse vector. Estimated values are indicated by accentuation $[\hat{z}]$. The superscript and subscript $[.]_H^T$ indicate transpose and Hankel matrix parameter. $\|.\|p$ represents the $lp-norm$.

# LITERATURE REVIEW

# CHAPTER 2

# COMPRESSIVE SENSING(CS) AND LINEAR DYNAMICAL SYSTEMS

## 2.1 Fundamentals of CS

It is considered imperative to introduce fundamental terminologies and the principles of the CS as well as Linear Dynamical Systems before their application in this work. The important terms as defined in the 'Introduction to Compressed Sensing' are reproduced below for effective comprehension of the subject [3].

**'Sparse Signal**. A signal of length $n$ that can be represented by $k \ll n$ , non-zero coefficients.

**$l_p$-norm.** 'The norm is defined for $p \in [1, \propto]$ as follows:

$$\|x\|_p = \begin{cases} (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}, & p \in [1, \infty) \\ \max\limits_{i=1,2,..,n} |x_i|, & p = \infty \end{cases} \quad (2.1)$$

**Basis**. A set $\{\boldsymbol{\phi}_i\}_{i=1}^{n}$ is called a basis for set of real numbers $\mathbb{R}^n$if the vectors in the set span $\mathbb{R}^n$ and are linearly independent. This implies that each vector in the space has a unique representation as a linear combination of these basis vectors. For any $x \in \mathbb{R}^n$, there exist unique coefficients $\{c_i\}_{i=1}^{n}$ such that

$$x = \sum_{i=1}^{n} \boldsymbol{c}_i \boldsymbol{\phi}_i \quad (2.2)$$

**Orthonormal Basis**. It is an important special case of a basis defined as a set of vectors $\{\boldsymbol{\phi_i}\}_{i=1}^{n}$ satisfing the following:

$$\langle \boldsymbol{\phi_i}, \boldsymbol{\phi}_j \rangle = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \quad (2.3)$$

It has the advantage that the coefficients $\boldsymbol{c}$ can be easily calculated as

$$\boldsymbol{c}_i = \langle x, \boldsymbol{\phi_i} \rangle, \ or \quad \boldsymbol{c} = \boldsymbol{\Phi}^T x \tag{2.4}$$

Where $\boldsymbol{c}$ is length *n* vector with entries $\boldsymbol{c}_i$ and $\boldsymbol{\Phi}$ denote the *n x n* matrix with columns given by $\boldsymbol{\phi}_i$.

**Frame.** It is defined as sets of possibly linearly dependent vectors. Frame is a set of vectors $\{\boldsymbol{\phi}_i\}_{i=1}^{n}$ in $\mathbb{R}^d$ , $d < n$ corresponding to a matrix $\boldsymbol{\Phi} \in \mathbb{R}^{d \times n}$ , such that for all vectors $\boldsymbol{x} \in \mathbb{R}^d$,

$$\boldsymbol{A} \, \|\boldsymbol{x}\|_2^2 \ \leq \ \|\boldsymbol{\Phi}^T \boldsymbol{x}\|_2^2 \leq \boldsymbol{B} \, \|\boldsymbol{x}\|_2^2 \tag{2.5}$$

with $0 < \boldsymbol{A} \leq \boldsymbol{B} < \propto$.

***p*-Flat**. The series of geometrical objects, point, line, plane and space from the three-dimensional Euclidian geometry are extended and termed as 0-flat, 1-flat, 2-flat, 3-flat respectively and extended upto N-flats. These are boundary less regions determined by 1, 2, 3, 4,…N+1 linearly independent points. A p-Flat is determined by p+1 linearly independent points, and every q-Flat, with q<p, which is determined by q+1 of these points, lies entirely within the p-Flat.

**Ambient Space**. It is defined as an N-Flat containing the considered p-flat and q-flat as proper subset. i.e. p and q are less than N.

**General Position**. A set of M-points in N-dimensional ambient space is in general position if no sub-collection of, at most, N points are linearly dependent i.e. iff any p+2 of them do not lie on a p-Flat.

**Null Space**. It is the space of a matrix ***A*** that consist of all vectors z such that their product is zero or null.

$$\mathcal{N}(\boldsymbol{A}) = \{\boldsymbol{z} : \boldsymbol{A}\boldsymbol{z} = 0\} \tag{2.6}$$

**Solution Space**. The solution space set of M-linear equations with N variables SLE(M,N) is represented by the intersection of all M-hyperplanes ((N-1)-Flats, which are solution spaces of particular linear equations) in the N-dimensional ambient space.

**Spark of the Matrix**. The spark of a given matrix ***A*** is the smallest number of columns of A that are linearly dependent.

**Null Space Property**. A matrix ***A*** satisfies the null space property (NSP) of order k if there exists a constant C>0 such that

$$\|h_{\Lambda}\|_2 \ \leq C \ \frac{\|h_{\Lambda^c}\|_1}{\sqrt{k}} \ , \tag{2.7}$$

holds for all $h \in \mathcal{N}(\boldsymbol{A})$ and for all Λ such that $|\Lambda| \leq k$.

**Restricted Isometric Property (RIP)**. A matrix ***A*** satisfies the restricted isometric property (RIP) of order k if there exists $\delta_k \in (0,1)$ such that

$$(1-\delta_k)\,\|\boldsymbol{x}\|_2^2 \ \leq \ \|\boldsymbol{A}\boldsymbol{x}\|_2^2 \ \leq (1+\delta_k)\,\|\boldsymbol{x}\|_2^2 \tag{2.8}$$

holds for all $\boldsymbol{x} \in \ \Sigma_k$. It means that if a matrix ***A*** satisfies the RIP of order 2k, then it approximately preserves the distance between any pair of k-sparse vectors.

**Coherence**. The coherence of matrix ***A*****,** ***μA*** is the largest absolute inner product between any two columns $\boldsymbol{a_i}, \boldsymbol{a_j}$ of **A**:

$$\mu(A) = \max_{1 \leq i < j \leq n} \frac{|\langle \boldsymbol{a_i}, \boldsymbol{a_j} \rangle|}{\|\boldsymbol{a_i}\|_2 \|\boldsymbol{a_j}\|_2} \ , \tag{2.9}$$

These definitions will help the reader understand the dynamics of mathematics involved in compressive sensing. The whole procedure from sensing to

reconstructions along with all variables and budgets has been summarised in the following figure adopted from the ULM course material on CS [4].

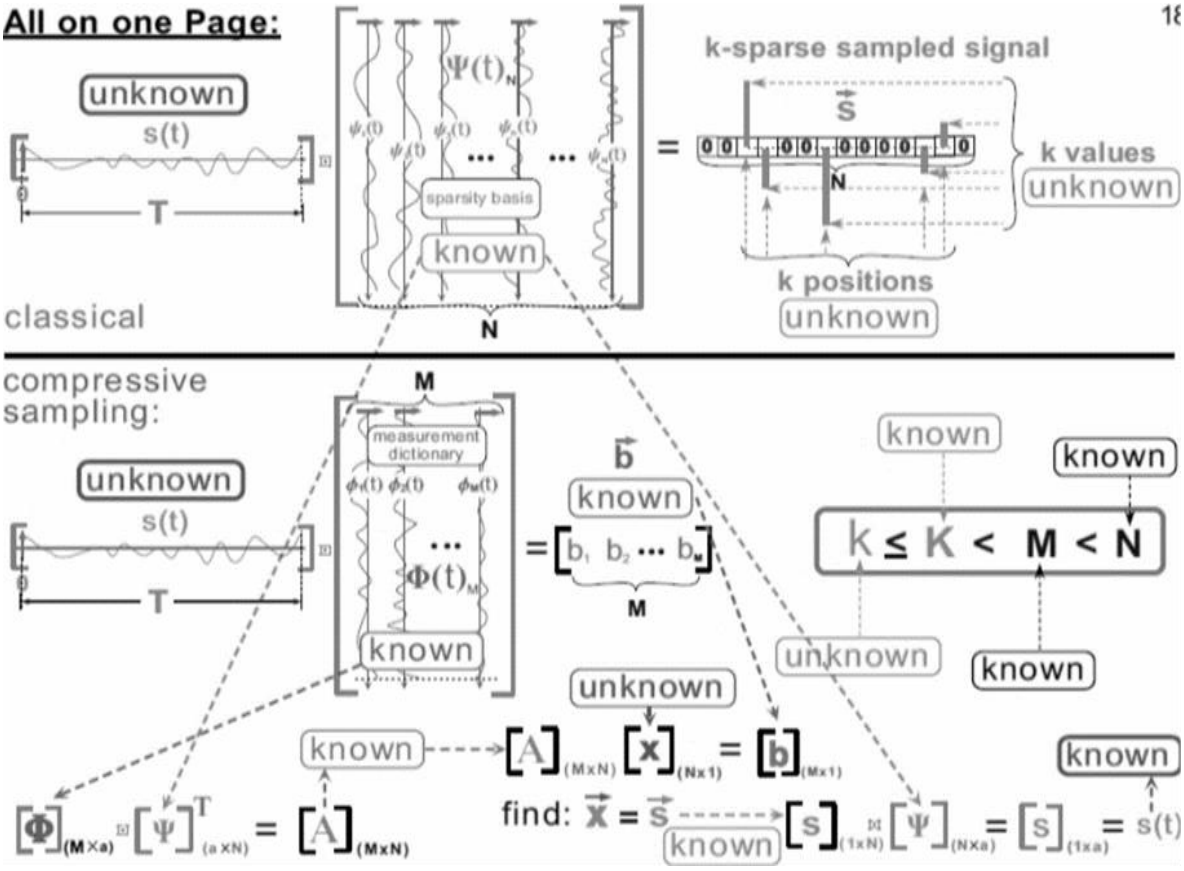



**Figure 2-1** Mathematical overview of CS showing sensing and reconstruction processes.

## 2.2 Compressive Sensing

CS states that a signal of the form $\boldsymbol{a} \in \mathbb{R}^{N}$ can be linearly measured at much smaller sampling rate in the form,

$$\boldsymbol{b} = \boldsymbol{\Phi}\, \boldsymbol{a} + \boldsymbol{\varepsilon}\,, \tag{2.10}$$

and can be successfully recovered. Here $\boldsymbol{\Phi} \in \mathbb{R}^{M \times N}$ represents the measurement matrix, $\boldsymbol{\varepsilon}$ is the measurement noise and $M < N$ [1,2]. The linear system of the form $\boldsymbol{b} = \boldsymbol{\Phi}\,\boldsymbol{a}$ is under-determined and is poorly conditioned to estimate $\boldsymbol{a}$ from the measurements $\boldsymbol{b}$. However, CS postulates that signal $\boldsymbol{s}$ is sparse in a sparsifying basis $\boldsymbol{\Psi}$ that is signal $\boldsymbol{a}$ defined as $\boldsymbol{a} = \boldsymbol{\Psi}\,\boldsymbol{s}$, has $K$ non-zero components at the maximum. Hence, if the matrix $\boldsymbol{\Phi\Psi}$ satisfies the restricted isometry property (RIP) then the signal $\boldsymbol{a}$ can be precisely reconstructed from $M = O\,(K \log (N/K))$ measurements [5]. Matrix $\boldsymbol{\Phi\Psi}$ satisfies the RIP when $\boldsymbol{\Psi}$ is an orthonormal basis and the elements of the matrix $\boldsymbol{\Phi}$ are i.i.d. samples from a sub-Gaussian distribution. Finally, the solution of the convex optimization problem of the form as in (2) can recover the signal $\boldsymbol{a}$ from $\boldsymbol{b}$.

$$\min \|\boldsymbol{s}\|_1 \;\; | \;\; \|\boldsymbol{b} - \boldsymbol{\Phi\Psi s}\|_2 \leq \boldsymbol{e}. \qquad (2.11)$$

Here $\boldsymbol{e}$ is an upper bound on the measurement noise $\boldsymbol{\varepsilon}$. The solution to (2.11) with high probability is the required $K$-sparse solution. CS protocol has also demonstrated that the sorted coefficients of the $\boldsymbol{s}$ swiftly degenerate according to a power-law in compressible signal [6]. Multiple algorithms exist that can solve (2.11) e.g. [1,7]. Sparse approximation problems are also efficiently handled by greedy algorithms like Orthogonal Matching Pursuits (OMP) [8] and CoSAMP [9]. A variation, model based CoSAMP, exploits fast convergence, computational efficiency and simplicity of structural constraints like block sparsity [10].

### 2.3 Status of research in CS

Nyquist rate demands sensing of features at twofold the specific frequency. However, Compressive Sensing (CS) facilitates reconstruction of signals, sparse

in some basis, at a far lower sampling rate then the Nyquist criteria [1,2]. Nyquist only takes into account the band-limitedness whereas CS exploits the structure based on sparsity. This Nyquist feature results into costly sensors/camera designs in non-visible spectrum. This research addresses sensing of videos through CS. Videos are highly redundant as is evident from the success of compression algorithms. Exploitation of compression and sensing in single step can lead to innovative sensor designs especially in infrared and beyond. It can considerably cut down sensed data and decrease costs. Sensors based on CS theory are already in place. Spatial Multiplexing Cameras (SMC) boost spatial resolution optically such as the single-pixel camera (SPC) [11] and the flexible voxels camera [12]. Temporal Multiplexing Cameras (TMCs) boost temporal, resolution optically like the P2C2 camera [13]. Interested readers may see Survey of compressive video sensing [14] for details of sensors based on CS architecture. Replacing full-frame sensor with far fewer optical sensors in non-visible wavelengths scene acquisition is highly beneficial.

SMCs measure scenes many times successively for accurate sensing. The approach delivers encouraging results for SPC and still images [11]. However, it delivers poorly for videos acquisition. Video CS by SMCs faces two major challenges; First, ephemeral nature, i.e., scene changes with each compressive measurement. Second, dimensionality of videos is much higher as compared to static images. In case of SPCs it gets more complex as it has only one sensor. This imperative dimension is missing from almost all approaches for CS-based video recovery (e.g., [15–19]). The proposed methods, appropriate for general

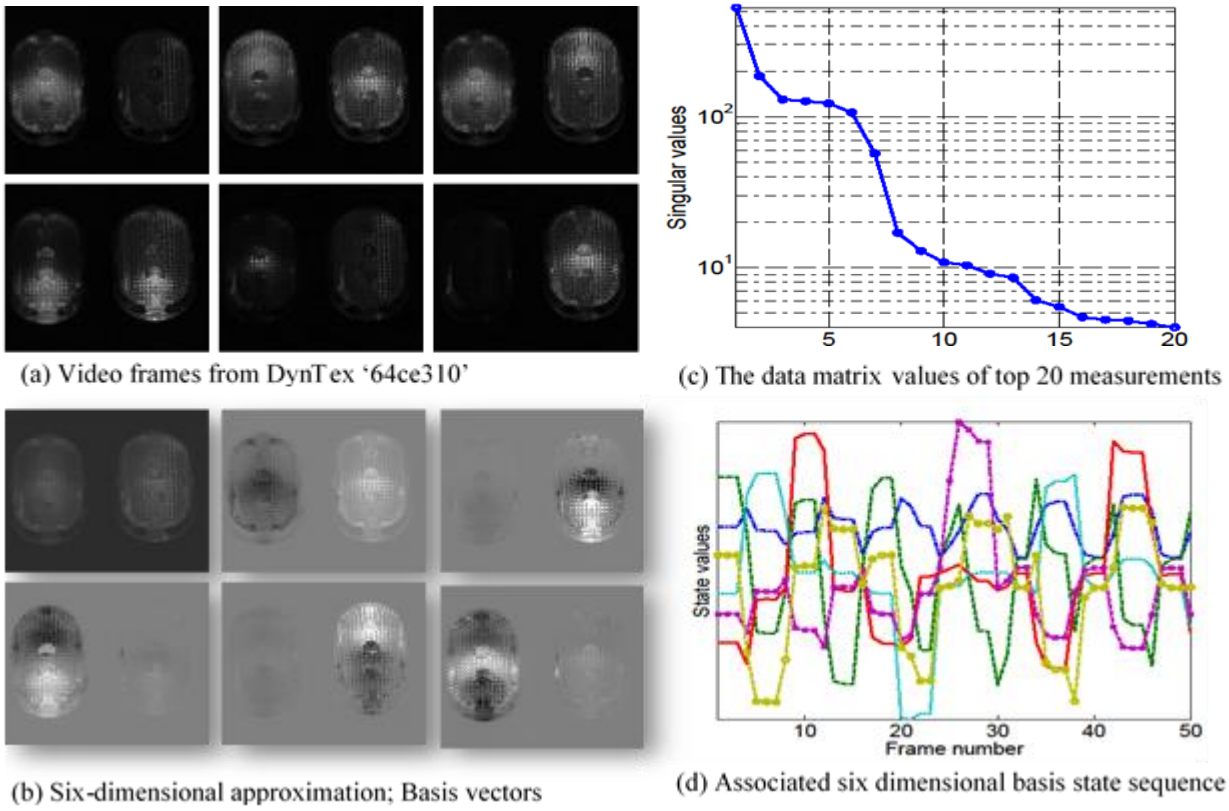
scenes, consider scenes as time-invariant series of frames (i.e. video) instead of continuously changing scenes. A noticeable exception is proposed in [20], however, it is designed specifically for time-varying periodic scenes. CS-MUVI [21] proposes sensing and recovery protocol based on optical flows.

Similarly, CS architecture based TMCs exploit motion estimates [13,19,22,23] however, suffer a fundamental problem of 'Chicken-and-egg' [21]. Multi-scale sensing [14] and sensing secluded pieces of each frame [19] rely on static models and ignore time –varying nature of videos. A Non-linear sensing architecture that optimizes system performance is suggested in [24]. Sankaranarayanan et al in [25] proposed specialized dual-scale sensing DSS matrix for robust initial scene estimates of lower spatial resolution videos sensed through SMC.

An approach to tackle the challenges is to model videos as a parametric problem. A parametric model like Linear Dynamical Systems (LDS) that fits most classes of the videos simplifies video reconstruction to parameter estimation. CS protocol for videos modeled as LDS is inspired by successful and extensive linear modeling of dynamic textures [26,27,28]. High dimension videos can be represented in much lower dimensions as LDS, hence reducing parameters to be estimated and in turn the sensing data. LDS characterizes the videos as a mix of dynamic/time varying and static/ time-invariant parameters. It also propagates a prior for progression of video in both forward and reverse time. This allows us to address the ephemeral nature of videos to a high degree.

## 2.4 Video modelling as linear dynamical systems

One of the important class of parametric modeling of time-series is Linear Dynamical Systems (LDS). Space-Time signals like traffic scenes [26], dynamic textures [27], video inpainting and human activity [29], and multi camera tracking [30] have widely been successfully realized as LDS. Application of LDS for precise modeling in computer vision problems is also presented as survey in [31]. This modelling of videos as time -indexed series of images will be exploited in this work. Most approaches fit an LDS model to videos by first estimating lower-dimensional embedded observations through principal component analyses (PCA) and then learning state transition by capturing time-varying dynamics of video. Expectation-Maximization (EM) [26], N4SID [32] and PCA-ID [33] are most prevalent algorithms. The EM algorithm treats this as maximum likelihood estimation of parameters that optimizes likelihood of observations. The N4SID algorithm identifies subspace that optimizes solution for model parameters. PCA-ID algorithm assumes that observation matrix and state transition matrix can be separately estimated. The model parameters can thus be computed efficiently by PCA. First, Space-filters provide estimates of the observation matrix and then results are used by time-filters to determine transition matrix [27]. Figure 2-2 presents an example of LDS and the models that define it. Few frames of six flashing LEDs from DynTex dataset [34], along with basis vectors are shown. Black pixels show non-negative whereas white denote positive values. Predictability is evident from smooth variations of state values.




(a) Video frames from DynTex '64ce310'

(c) The data matrix values of top 20 measurements

(b) Six-dimensional approximation; Basis vectors

(d) Associated six dimensional basis state sequence

**Figure 2-2** Example of LDS and model that defines it.

## 2.5 Linear Dynamical Systems

System of linear equations has an important role in modern day technology. These systems are immensely employed to model control systems, State estimations, observability, single input single output systems(SISO), Multiple input multiple output systems (MIMO). A system is said to be linearly dynamical if initial, current state or future states can be estimated from a linear relationship [35]. A system of one or more variables which evolve in time according to a given rule is called a dynamical system. represented by the following system of equations in discrete time;

$$X(t + \Delta t) = F(X(t)) \tag{2.12}$$

$$X_{n+1} = F(X_n) \tag{2.13}$$

A linear dynamical system is one in which the rule governing the time-evolution of the system involves a linear combination of all the variables. e.g.

$$\frac{dX}{dt} = AX + B \tag{2.14}$$

## 2.6 Summary

This chapter reviews fundamental definitions that help better understand the CS concepts. It also briefly sifts through the currently available literature that is relevant to this work. Various video CS techniques along with the limitations have been briefly mentioned. Similarly, Linear Dynamical Models and their applications in successfully modelling the videos have also be briefly touched upon.

# CHAPTER 3

# COMPRESSIVE SENSING VIDEO AS LINEAR DYNAMICAL SYSTEM

## 3.1. Compressive sensing video

This work models video as time-indexed series of images. If $\boldsymbol{a_t}$ is the still image of video at time *t* then $\boldsymbol{a_T}$ **=** $\boldsymbol{\{\, a_1, a_2, a_3, \ldots \ldots \ldots a_T \}}$ is the video from 1 to T. The $\boldsymbol{a_t}$ is also defined as 'video frame' at time *t.* The goal is to compressively sense $\boldsymbol{b_t = \Phi_t a_t}$ where $\boldsymbol{b_t}$ , $\boldsymbol{\Phi_t}$ and $\boldsymbol{a_t}$ are compressive measurements, sensing matrix and video frame at time *t.* With series of compressive measurements $\boldsymbol{b_T = \{\, b_1, b_2, b_3, \ldots \ldots \ldots b_T \}}$ it is tried to recover video $\boldsymbol{a_T = \{\, a_1, a_2, a_3, \ldots \ldots \ldots a_T \}}$. The focus will be SPC [26] since it exploits spatial multiplexing to the fullest but there is no temporal multiplexing. Assuming that scene varies slowly with time, SPC measures at each instant $\boldsymbol{b_t = \phi_t^T a_t}$ where $\boldsymbol{\phi_t}$ is a pseudo-random vector. Successive measurements are grouped as of same frame of a video. This supposition works well if either the motion in scene is slow or respective frame measurements are scant in quantity. SPC architecture is most suitable for re-designing cost effective sensors in higher spectrum like near and far infrared.

## 3.2. Video modelling as linear dynamical system

This work models video as time-indexed series of images. LDS model of video consists of two steps. As a first step video frames are considered laying close to *d*-dimensional subspace i.e. video frame can be represented as,

$$\boldsymbol{a_t \approx C z_t,} \tag{3.1}$$

at a time *t*; where $\boldsymbol{z_t}$ is the state vector at time *t* and ***C*** is subspace basis. Second step is modeled as linear evolution of the form,

$$\boldsymbol{z}_{t+1} \approx \boldsymbol{S}\, \boldsymbol{z}_t, \tag{3.2}$$

representing the predicted variations of route in *d*-dimensional subspace. Hence, the LDS model equations for video are defined as,

$$\boldsymbol{a}_t = \boldsymbol{C}\boldsymbol{z}_t + \boldsymbol{\eta}_t \qquad \boldsymbol{\eta}_t \approx \boldsymbol{N}(0, \boldsymbol{P}), \tag{3.3}$$

$$\boldsymbol{z}_{t+1} = \boldsymbol{S}\boldsymbol{z}_t + \boldsymbol{\eta}"_t \qquad \boldsymbol{\eta}"_t \approx \boldsymbol{N}(0, \boldsymbol{Q}). \tag{3.4}$$

Here, $\boldsymbol{z}_t \in \mathbb{R}^d$**,** in state-space dimension *d,* is the state vector at time *t***,** $\boldsymbol{C} \in \mathbb{R}^{N\times d}$ is the observation matrix, $\boldsymbol{S} \in \mathbb{R}^{d\times d}$ is the state transition matrix and $\boldsymbol{a}_t \in \mathbb{R}^N$ is the observed measurements vector. Here, $d \ll N$ for videos considered in this work, and $\boldsymbol{\eta}_t, \boldsymbol{\eta}"_t$ are Gaussian noise vectors with zero mean and ***P, Q*** covariance matrices belonging to $\mathbb{R}^{N\times N}\ and\ \mathbb{R}^{d\times d}$ respectively. Gaussian noise is assumed for simplicity and better results with dynamic textures [27].

The matrix pair (***C,S***) defines parametric model of video LDS. The unique choices of ***C*** and sate-sequence $\boldsymbol{z}_{1:T}$ in state-space are only possible in $d \times d$ linear transformation. Thus, any invertible matrix ***D*** defining LDS outlined by (***C,S***), of the order $d \times d$ with sate sequence $\boldsymbol{z}_{1:T}$ corresponds to LDS stated by (***CD, D$^{-1}$SD***) with state sequence

$$\boldsymbol{D^{-1}z}_{1:T} = \{\boldsymbol{D^{-1}z}_1,\ \boldsymbol{D^{-1}z}_2\ ,\ \boldsymbol{D^{-1}z}_3, \ldots\ldots,\ \boldsymbol{D^{-1}z}_T\}. \tag{3.5}$$

Most approaches fit an LDS model to videos by first estimating lower-dimensional embedded observations through principal component analyses (PCA) and then learning ***S*** by capturing time-varying dynamics of $\boldsymbol{a}_T$. Expectation-Maximization (EM), N4SID and PCA-ID are most prevalent algorithms. The EM algorithm treats this as maximum likelihood estimation of parameters that optimizes likelihood of observations. The N4SID algorithm identifies subspace that

optimizes solution for model parameters. PCA-ID algorithm assumes that observation matrix ***C*** and state transition matrix ***S*** can be separately estimated. The model parameters can thus be computed efficiently by PCA. First, Space-filters provide estimates of the observation matrix ***C*** and then results are used by time-filters to determine transition matrix ***S*** [27].

### 3.3. LDS-VCS Architecture

Linear Dynamical System-Video Compressive Sensing (LDS-VCS) protocol is proposed in this work. This protocol is presented here and it is implementable on single pixel camera (SPC) for videos modelled as LDS. It intends to capture the model parameters ***C*** and $\boldsymbol{z}_{1:T}$ subject to compressive measurements of the form

$$\boldsymbol{b}_t = \boldsymbol{\Phi}_t \boldsymbol{a}_t = \boldsymbol{\Phi}_t \boldsymbol{C}\, \mathbf{z}_t, \tag{3.6}$$

where ***C*** is static observation matrix of the LDS, $\boldsymbol{\Phi}_t$ is the sensing matrix, and $\boldsymbol{a}_t \ and \ \mathbf{z}_t$ are corresponding video frames and states at time *t.* The compressive measurements $\boldsymbol{b}_{1:T}$ are thus stated in bilinear terms of unknown parameters ***C*** and $\mathbf{z}_{1:T}$. Convex optimization techniques are typically unable to handle bilinear unknowns, hence a two-step sensing technique is proposed called LDS-VCS. The protocol is designed for compressively sensing and correspondingly recovering the LDS.

#### 3.3.1. Measurement Model

LDS-VCS model is summarized below: At time *t,* two sets of measurements are made:

$$\boldsymbol{b}_t = \begin{pmatrix} \widetilde{\boldsymbol{b}_t} \\ \widetilde{\boldsymbol{b}_t} \end{pmatrix} = \begin{bmatrix} \widetilde{\boldsymbol{\Phi}_t} \\ \widetilde{\boldsymbol{\Phi}_t} \end{bmatrix} \boldsymbol{a}_t = \boldsymbol{\Phi}_t \boldsymbol{a}_t. \tag{3.7}$$

Here $\breve{\boldsymbol{b}}_{\boldsymbol{t}} \in \mathbb{R}^{\breve{M}}$ and $\tilde{\boldsymbol{b}}_{\boldsymbol{t}} \in \mathbb{R}^{\tilde{M}}$ so that measurements of every frame are $M = \breve{M} + \tilde{M}$. In fact SPC takes only one measurement at $t$, but for slowly changing videos grouping of successive measurements is done to make multiple measurements. It holds true when sampling rate of SPC is higher compared to $M$. Two discreet parts; the time-invariant $\breve{\boldsymbol{\Phi}}_t$ and time variant $\tilde{\boldsymbol{\Phi}}_t$ together make up measurement matrix in (5). We represent *common* measurements as $\breve{\boldsymbol{b}}_t$ and *innovative* as $\tilde{\boldsymbol{b}}_t$ . A two-step approach is used to find LDS parameters. First using common measurements $\breve{\boldsymbol{b}}_{1:T}$ , state sequence will be estimated and then the sequence along with innovative measurements will be employed for recovery of observation matrix $\boldsymbol{C}$.

### 3.3.2. State Sequence Estimation

State sequence $\breve{\boldsymbol{z}}_{1:T}$ is recovered through time-invariant part of the measurement matrix, $\breve{\boldsymbol{b}}_{1:T}$ . The basic preposition is that if $\boldsymbol{a}_{1:T}$ generates observations of LDS with system matrices [$\boldsymbol{C,S}$] then the measurement $\breve{\boldsymbol{b}}_{1:T}$ is the observation of LDS with system matrices [$\breve{\boldsymbol{\Phi}}\boldsymbol{C}, \boldsymbol{S}$ ] . State sequence estimation is only possible from observations of LDS if LDS is observable [36], hence LDS parameterized by [$\breve{\boldsymbol{\Phi}}\boldsymbol{C}, \boldsymbol{S}$] must also be observable. This simplifies the problem from state sequence estimation to system identification that can be resolved by Singular Value Decomposition, (SVD) method of a block Hankel matrix of the following from,

$$\boldsymbol{H}_{(\breve{\boldsymbol{b}}_{1:T,d})} = \begin{bmatrix} \breve{\boldsymbol{b}}_1 & \breve{\boldsymbol{b}}_2 & \cdots\cdots & \breve{\boldsymbol{b}}_{T-d+1} \\ \breve{\boldsymbol{b}}_2 & \iddots & \iddots & \breve{\boldsymbol{b}}_{T-d+2} \\ \vdots & \iddots & \iddots & \\ \breve{\boldsymbol{b}}_d & \cdots & \cdots & \breve{\boldsymbol{b}}_{T-1} \quad \breve{\boldsymbol{b}}_T \end{bmatrix}. \tag{3.8}$$

The state sequence estimates are calculated by $[\,\breve{\boldsymbol{z}}_{1:T}\,] = \boldsymbol{S_H V_H^T}$ given that SVD$\{\boldsymbol{H}_{(\breve{\boldsymbol{b}}_{1:T,d})}\} = \boldsymbol{U_H\, S_H\, V_H^T}$.

It is possible to estimate state sequence in all cases of number of measurements taken at time *t* within dimension *d* where $\breve{M} > d \; and \; \breve{M} < d$. It is true even when $\breve{M} < 1$ [37].

### 3.3.3. Observation Matrix estimates

The relationship between observation matrix ***C*** and innovation or time-variant measurements is linear i.e. $\widetilde{\boldsymbol{b}}_{\boldsymbol{t}} = \widetilde{\boldsymbol{\phi}}_{\boldsymbol{t}} \boldsymbol{C} \hat{\boldsymbol{z}}_{\boldsymbol{t}}$. With state sequence already estimated $\hat{\boldsymbol{z}}_{\boldsymbol{1:T}} \; and \; \boldsymbol{C}$ as time-invariant, innovative measurements can be accrued to recreate a stable ***C***. Innovative measurements $\widetilde{M}$ required for each frame are thus reduced substantially. This results in less error that is caused by motion blurring. The ***C*** can be recovered using state sequence $\hat{\boldsymbol{z}}_{\boldsymbol{1:T}}$ estimates using convex problem as follows

$$\boldsymbol{min} \sum_{i=1}^{\boldsymbol{d}} \| \boldsymbol{\Psi^T c}_i \|_1 \mid \forall t \, \| \boldsymbol{b}_t - \boldsymbol{\Phi}_t \boldsymbol{C} \hat{\boldsymbol{z}}_t \|_2 \leq \boldsymbol{e} \tag{3.9}$$

The i[th] column of ***C*** is represented as $\boldsymbol{c_i}$ and $\boldsymbol{\Psi}$ is sparsifying basis for columns of ***C***. The state sequence estimates generate structured sparsity pattern in support of ***C***. This structured sparsity is exploited in proposed recovery algorithm. All compressive measurements $\boldsymbol{b}_t$ of each frame of the video or image are used since innovative and common measurements are both linear. Matrix ***C*** is linear in compressive measurements and $l_1$ or $l_2$ optimization methods can recover ***C***. However, it is a matrix of order *N* x *d.* Thus, common measurements alone are not enough unless $\widetilde{M}$ is large. The block diagram of LDS-VCS is shown in figure 3-1.

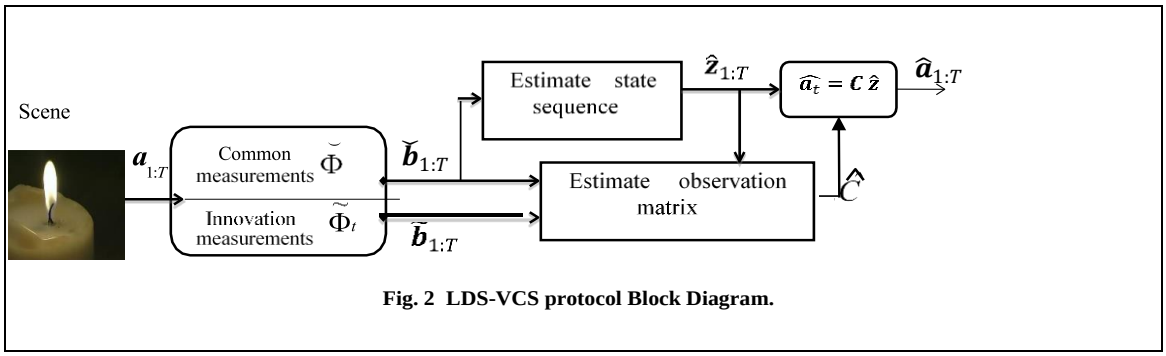



**Fig. 2 LDS-VCS protocol Block Diagram.**

**Figure 3-1** LDS-VCS protocol Block diagram

### 3.3.4. Structured Sparsity for *C*

Each video frame like video or image is sparse in some transform basis like Wavelet or Discrete Cosine Transform (DCT). Since ***C*** is basis of video frames, columns of ***C*** are compressible in similar transform basis. Additionally, the columns are also the principal components capturing the dominant motion pattern when spatially correlated. Therefore, it is assumed that columns of ***C*** are compressible in DCT /Wavelet domain. The ***C*** can be estimated by considering the convex problem of the type formed in equation (3.9).

There is possibility here that video is not sparse in transform basis. This can be handled using dictionary learning [38] provided training data is available. In case of non-availability of training data $l_2 - norm$ based approach can be used.

The vague LDS definition and use of SVD in estimating state sequence adds ambiguity and renders (3.9) convex problem ineffective to recover ***C***. It introduces ambiguity in the form $[\hat{\boldsymbol{z}}_{1:T}] \approx \boldsymbol{L}^{-1}[\boldsymbol{z}_{1:T}]$. Here ***L*** is an invertible matrix of the order *dxd*. Consequently, the LDS leads us to $\widehat{\boldsymbol{C}}$ = ***CL***, a linearly transformed ***C***. Now if

the columns of $\boldsymbol{C}$ are $K$-sparse in $\boldsymbol{\Psi}$ with support $\$_{\mathbf{k}}$, its columns are $dK$-sparse with similar support. The overall sparsity of $\widehat{\boldsymbol{C}}$ increases to $d^2K$. The apparent increase in sparsity is mitigated by the fact that columns of $\boldsymbol{C}$ have identical support. Hence that is exploitable for recovery of $\boldsymbol{C}$ [39]. With $\hat{\boldsymbol{z}}_{1:T}$ available we can calculate matrix $\boldsymbol{C}$ by solving $\boldsymbol{l_2 - l_1}$ mixed-norm optimization convex problem that enhances group sparsity as follows.

$$\boldsymbol{min} \sum_{i=1}^{N} \| \boldsymbol{w}_i \|_2 \mid \boldsymbol{C} = \boldsymbol{\Psi W}, \forall t, \| \boldsymbol{b}_t - \boldsymbol{\Phi}_t \boldsymbol{C} \hat{\boldsymbol{z}}_t \|_2 \leq \boldsymbol{e}. \tag{3.10}$$

The matrix $\boldsymbol{W} = \boldsymbol{\Psi}^T \boldsymbol{C}$ has $\boldsymbol{w}_i$ as the i$^{th}$ row of $\boldsymbol{W}$ and $\boldsymbol{\Psi}$ is the sparsifying basis of columns in $\boldsymbol{C}$. SPG-L1 [7] and model based CoSAMP [10] are some algorithms to solve $\boldsymbol{l_2 - l_1}$ problems efficiently. Algorithm-*LDS_VCS_*$\hat{C}$ provides pseudocode that uses union of sub-space model to group rows of $\boldsymbol{W} = \boldsymbol{\Psi^T C}$ into single sub-space to recover observation matrix $\boldsymbol{C}$ on the lines of (3.10).

A simple rule for recovery of $\boldsymbol{C}$ is that total compressive measurements should be

$$\widetilde{M} = 4\, dK\, log\, (\tfrac{N}{K}). \tag{3.11}$$

Estimation of $\boldsymbol{C}$ over period of $T$ instant implies that we have $\widetilde{M}T$ time-variant compressive measurements.

$$\therefore \widetilde{M}T = 4\, dK\, log\, \left(\frac{N}{K}\right) \tag{3.12}$$

$$\Rightarrow \widetilde{M} = 4\frac{dK}{T} log\, \left(\frac{N}{K}\right), \tag{3.13}$$

assuming $T = \tau f_\delta$ where $f_\delta$ is sampling rate of SPC and $\boldsymbol{\tau}$ is duration of video to be sensed then,

$$\widetilde{M} = 4\frac{dK}{\tau f_\delta} log\, \left(\frac{N}{K}\right), \tag{3.14}$$

i.e. number of measurements are inversely proportional to the sampling rate of the SPC for stable recovery of $\boldsymbol{C}$. LDS-VCS protocol thus has highly supportive conditions.

### 3.3.5. LDS with mean

Linear dynamical scenes are better modelled generally over static background in the form:

$$\boldsymbol{a}_t = \boldsymbol{C}\boldsymbol{z}_t + \boldsymbol{\mu}. \tag{3.15}$$

Algorithm LDS_VCS can be modified with two little changes to incorporate the mean. Firstly, the SVD on the Hankel matrix $\boldsymbol{H}_{(\check{\boldsymbol{b}}_{1:T,d_{guess}})}$ can be modified in such a way so that each row sums to zero to estimate state sequence $\hat{\boldsymbol{z}}_{1:T}$. Here the assumption is made that compressive measurement of $\boldsymbol{\mu}$, $\check{\boldsymbol{\Phi}}\boldsymbol{\mu}$ is sample mean of $\check{\boldsymbol{z}}_{1:T}$ . Secondly, given that $\boldsymbol{\mu}$ and $\boldsymbol{C}$ can have different support i.e. they may not necessarily have similar support, the optimization problem takes the following form;

$$min\ \|\boldsymbol{\Psi}^T\boldsymbol{\mu}\|_1 + \sum_{i=1}^{N}\|\boldsymbol{w}_i\|_2 \mid \boldsymbol{C} = \boldsymbol{\Psi}\boldsymbol{W}, \forall t, \|\tilde{\boldsymbol{b}}_t - \tilde{\boldsymbol{\Phi}}_t(\boldsymbol{\mu} + \boldsymbol{C}\hat{\boldsymbol{z}}_t)\|_2 \leq \boldsymbol{e}. \tag{3.16}$$

The model based CoSAMP algorithm defined in LDS-VCS can be modified to incorporate the $\boldsymbol{\mu}$ term, the mean. Additionally, the sparsity of mean is also required to be defined a priori as $K_{\boldsymbol{\mu}} = \|\boldsymbol{\Psi}^T\boldsymbol{\mu}\|_0$.

The pseudocode for the algorithm LDS-VCS for estimation of $\widehat{\boldsymbol{C}}$ is given in figure 3-2:

$Algorithm\ \ LDS - VCS - \widehat{\boldsymbol{C}}$

$\widehat{\boldsymbol{C}}\ \{\boldsymbol{\Psi}, K, \boldsymbol{b}_t, \hat{\boldsymbol{z}}_t, \boldsymbol{\Phi}_t, t = 1 \dots \dots \dots T\}$

Notation:

*Support of K largest elements of vector = $(vec, K)*

*Submatrix of X with rows indexed by Λ and all columns =* $X_{|\Lambda,}$

*Submatrix of X with columns indexed by Λ and all rows =* $X_{|,\ \Lambda}$

*Initialization*

$\forall t,\ \boldsymbol{\beta}_t \leftarrow \boldsymbol{\Phi}_t \boldsymbol{\Psi}$

$\forall t, \boldsymbol{v}_t \leftarrow 0\ \in \mathbb{R}^M$

$\Lambda_{old} \leftarrow \boldsymbol{\phi}$

*While do (till stop conditions satisfied)*

$\boldsymbol{R} = \sum_t \boldsymbol{\beta}_t^T \boldsymbol{v}_t\, \hat{\boldsymbol{z}}_t^T$ *% Calculate signal proxy%*

$k \in [1, .., N],\ \boldsymbol{r}(k) = \sum_{i=1}^{d} \boldsymbol{R}^2\,(k, i)$ *%Calculate energy in each row%*

$\Lambda \ \leftarrow \Lambda_{old} \cup \$(\boldsymbol{r}; 2K)$ *% identify support and merge%*

*Find* $\boldsymbol{X} \in \mathbb{R}^{|\Lambda| \times d}$ *that maximize*

$\sum_t \| \boldsymbol{b}_t - (\boldsymbol{\beta}_t)_{|,\Lambda} \boldsymbol{X} \hat{\boldsymbol{z}}_t \|_2$ *% Estimate Least squares%*

$\boldsymbol{Y}_{|\Lambda,} \leftarrow \boldsymbol{X},\ \boldsymbol{Y}_{|\Lambda,} \leftarrow 0$

$k \in [1, .., N],\ \boldsymbol{y}(k) = \sum_{i=1}^{d} \boldsymbol{Y}^2\,(k, i)$ *% Find support %*

$\Lambda \leftarrow \$(\boldsymbol{y}; K), \boldsymbol{S}_{|\Lambda,} \leftarrow \boldsymbol{B}_{|\Lambda,}, \boldsymbol{S}_{|\Lambda,} \leftarrow 0$

$\widehat{\boldsymbol{C}} = \boldsymbol{\Psi S}$ *% re-form new estimates of C%*

$\forall t, \boldsymbol{v}_t \leftarrow \boldsymbol{b}_t \leftarrow \boldsymbol{\beta}_t \boldsymbol{S} \hat{\boldsymbol{z}}_t$ *% update residue%*

$\Lambda_{old} \leftarrow \Lambda$

*End*

**Figure 3-2** Pseudo-code for the proposed Algorithm

# CHAPTER 4

# SIMMULATION AND RESULTS

## 4.1. Simulation environment

The simulation results for modelling of videos as LDS and their recovery using the proposed algorithm as described in *LDS-VCS-*$\hat{C}$, is presented. Simulations on multiple aspects including compression ratio, recovery time and noise robustness (reconstruction SNR) were performed. Reduction in measurements rate viz-a-viz Nyquist rate was termed as compression ratio represented as $N/M$. Another parameter was the reconstruction SNR of videos. The SNR was defined on the basis of ground truth video $\boldsymbol{a_{1:T}}$ and recovered video $\widehat{\boldsymbol{a}}_{\boldsymbol{1:T}}$ in dBs as

$$10\ log_{10}\left(\frac{\sum_{t=1}^{T}\|\ \boldsymbol{a}_t\|_2^2}{\sum_{t=1}^{T}\|\ \boldsymbol{a}_t-\ \widehat{\boldsymbol{a}}_t\|_2^2}\right). \tag{4.1}$$

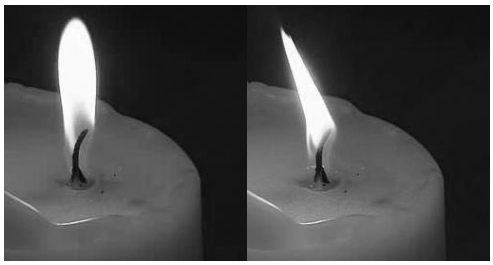
The test simulations were run on videos available in 'DynTex' dataset [34]. It is because dynamical systems are better modelled on the static background. All videos were LDS modeled as explained in chapter 3 for all simulations. Permuted noiselets [20] having fast scalable implementation were used for measurement matrices. The 2D DCT basis was used for sparsifying columns of ***C*** whereas 2D wavelets basis were employed as the sparsifying basis for the mean. Algorithm LDS_VCS was used for these results as sparsity of the columns of ***C*** was well controlled by it. Equation (3.11) was referred to for choices of other values. The protocol was simulated on laptop with a Core-i5 @ 2.5 GHz processor and 4 GB RAM. MATLAB version R2015A was employed with no paralleling. LDS_VCS is

compared with frame-by frame CS (FCS) that employs conventional CS methods to recover each frame of video separately.

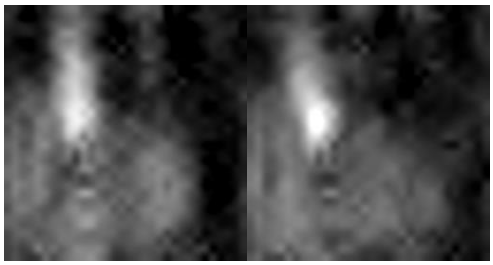
### 4.2. Resolution and Reconstruction time

Recovery of dynamic texture, ‘64cae10’ video of the DynTex [34] dataset is depicted in figure 4.1. The video comprises of 560 frames. The reconstruction is done with fixed compression ratio of x20 or $N/M = 20$ and with $d = 50$. The figure also reflects the SNR of reconstructed frames along with time taken to recover the video. It is evident from 4.1 (b) to (e) that reconstruction performance increases with increasing spatial resolution. As the compression ratio is kept fixed, improvement is due to increasing number of compressive measurements at higher resolution. Large number of measurements however, need an SPC with higher rate of sampling. Further, the time taken to compute the higher resolutions videos also increases with increasing spatial resolution

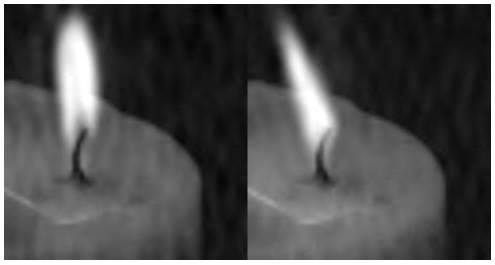
(a)



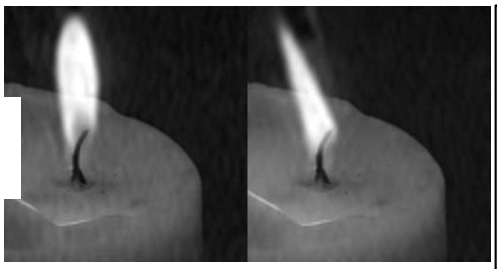
(b)



(c)



(d)



(e)

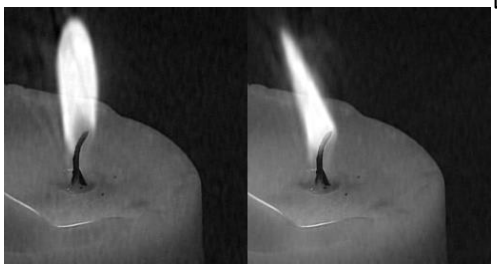


**Figure 4-1 (a-e)** Video ‘64cae10’ reconstructed at different resolutions with fixed compression ratio

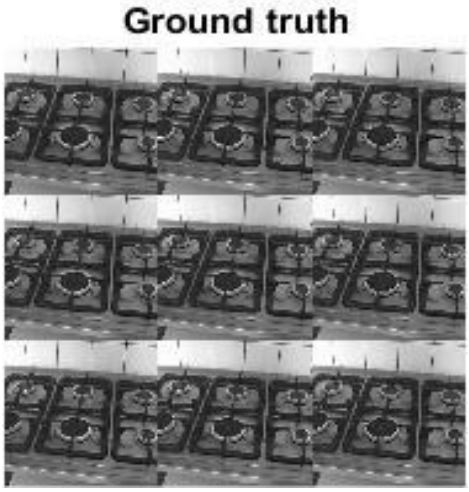
The figure 4-1 (a) shows two of the ground truth frames from the video '64cae10'. Subsequently 4-1 (b) to (e) represent reconstruction of the same frames with increasing spatial resolution. The resolution in 4-1 (b) is 32x32, whereas in (c) it is increased to 64x64, (d) is a frame of 128x128 and (e) depicts a reconstructed frame of resolution 256x256. The reconstruction SNR increases with increasing frame resolution from 10.11 dBs to 20 dBs to 22.06 dBs and finally 22.12 dBs from (b) to (e) respectively. The table 4-1 summarizes the information.

**Table 4-1** Summary of Resolution and Reconstruction Time

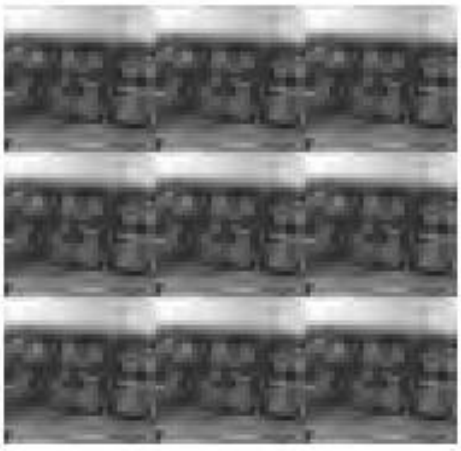
| **Figure Label** | **Resolution** | **Comp X N/M** | **Recon SNR (dBs)** | **Recon Time (Sec)** |
|---|---|---|---|---|
| 4-1 (a) | Ground Truth | | | |
| 4-1 (b) | 32 x 32 | X20 | 10.11 | 26.5 |
| 4-1 (c) | 64 x 64 | X20 | 20.00 | 51.8 |
| 4-1 (d) | 128 x 128 | X20 | 22.06 | 290 |
| 4-1 (e) | 256 x 256 | X20 | 22.12 | 1582 |

The reconstruction of another DynTex Dataset is reproduced in figure 4-2 along with the details of reconstruction parameters, the resulting reconstruction SNR and the reconstruction time taken by the simulation.

(a)




(b)




(c)

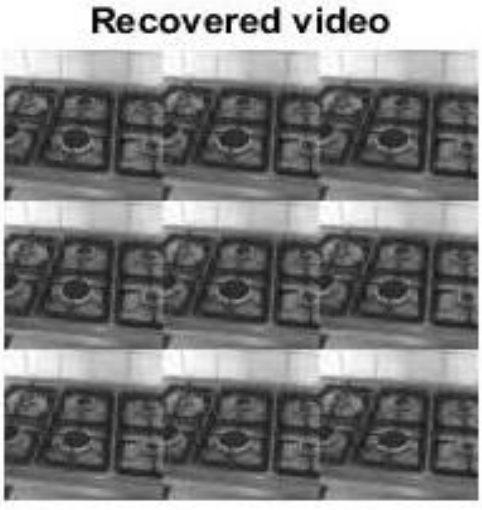



**Figure 4-2 (a-c)** Reconstructions with fixed compression ratio and varying resolution of ‘64ba910’

(d)


(e)

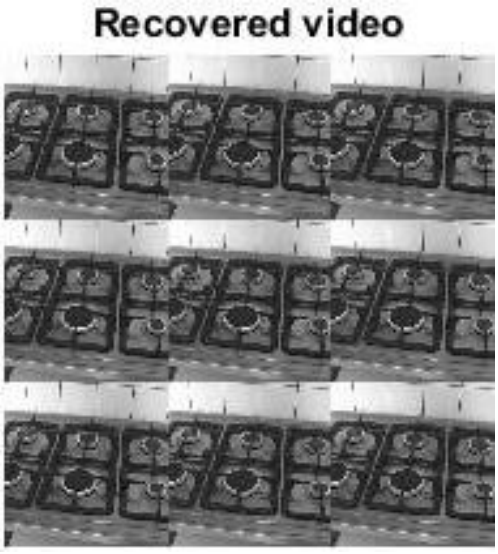



**Figure 4-2 (d-e)** Reconstructions with fixed compression ratio and increasing resolution for '64ba910'

Reconstruction of DynTex dataset '64ba910' with a fixed compression ratio are shown in figure 4-2 (a-e). The video has 250 frames and is reconstructed with a compression ratio of 20x for all the recovered videos as shown in figure 4-2. However, the resolution of each image varies from 32x32 to 256 x 256. The *d* was kept fixed at 30. The various reconstruction SNRs for increasing resolution are

listed in table 4-2. It is evident from the table 4-2 that the reconstruction performance SNR improves with enhancement in the resolution. This is because the increase in the number of compressive measurements. Hence, recovery of videos at higher resolutions are dependent on the faster sampling rate of the single pixel camera. It is also evident that the increasing spatial resolution increases the time taken to compute the reconstructed video.

**Table 4-2** Reconstruction SNR at different Compression Ratios for figure 4-2.

| Figure Label | Resolution | Comp X N/M | Recon SNR (dBs) | Recon Time (Sec) |
|---|---|---|---|---|
| 4-2 (a). | 256x256 | Ground Truth | | |
| 4-2 (b). | 256x256 | 20x | 31.544 | 99 |
| 4-2 (c). | 128x128 | 20x | 31.262 | 28 |
| 4-2 (d). | 64x64 | 20x | 25.006 | 19 |
| 4-2 (e) | 32x32 | 20x | 20.830 | 16 |

### 4.3. Reconstruction with different compression ratios

The reconstruction of a video, ‘64ce310’, 6-blinking LED lights is reproduced as figure 4-3. The referred number in commas is the name of the video as it appears in the DynTex dataset. Reconstruction results of the single dynamic textured video at different spatial resolutions and compression ratios are

enunciated. The *d* for the reconstruction was 7 and the $\breve{M}$ was 3*d*. Whereas the $\tilde{M}$ was chosen such that $N/\breve{M}+\tilde{M}$ preserves the desired compression ratio.

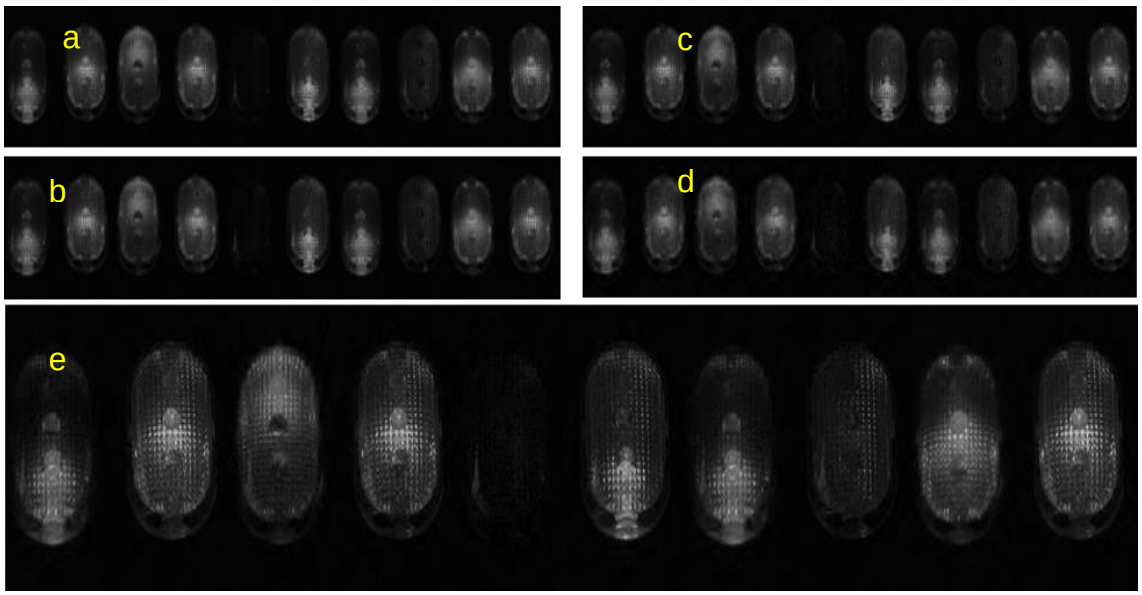



**Figure 4-3** Video reconstruction '64ce310' with different compression ratios

It is clearly evident that finer details are well preserved for resolution of 256x256 pixels despite compression ratio of 100x. The values of the compression ratio, spatial resolution and the reconstruction SNR are tabulated below in Table 4-3 for various labels in figure-4-3. Reconstruction SNR decreases with increasing compression ratio for same input resolution.

**Table 4-3** Reconstruction SNR at different resolutions for figure 4-3.

| Figure Label | Resolution | Comp X | Rec SNR |
|---|---|---|---|
| 4-2 (a). | 128x128 | Ground Truth | |
| 4-2 (b). | 128x128 | 20x | 22.7 |
| 4-2 (c). | 128x128 | 50x | 18.6 |
| 4-2 (d). | 128x128 | 100x | 16.3 |
| 4-2 (e). | 256x256 | 100x | 16.0 |

Another, result of video reconstruction with fixed resolution and increasing compression ratio demonstrates the effectiveness of the proposed algorithm. Figure 4-4 (a-e) reproduces the results of simulation for the reconstruction of DynTex data set ‘ 64cc610’.

(a)

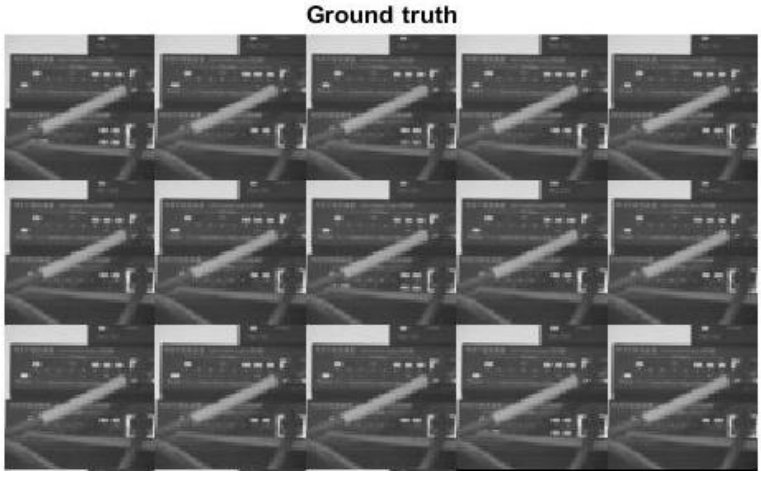
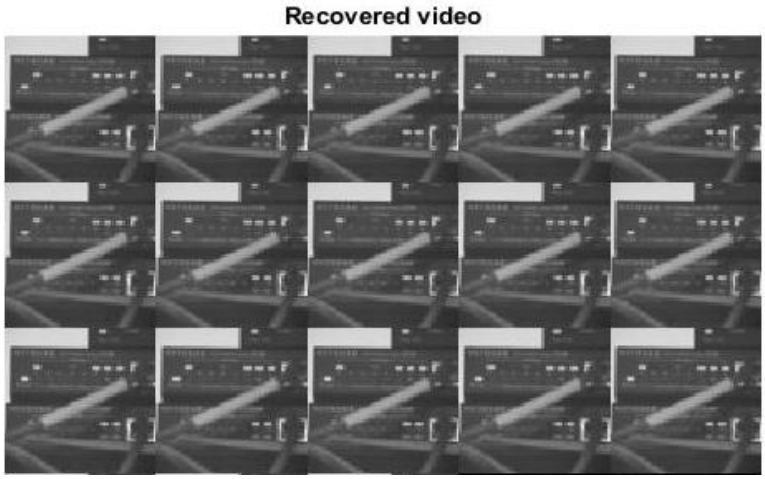



**Figure 4-4 (a)** Reconstruction SNR at different compression ratios of ‘64cc610’.

(b)




(c)

(d)

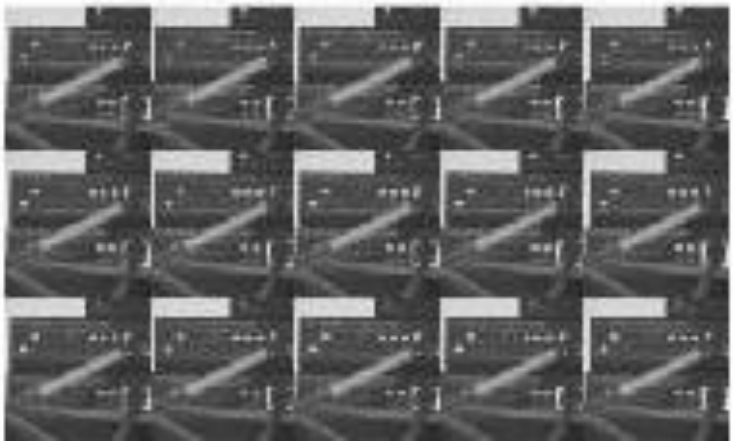


**Figure 4-4 (b-d)** Reconstruction SNR at different compression ratios of ‘64cc610’.

(e)

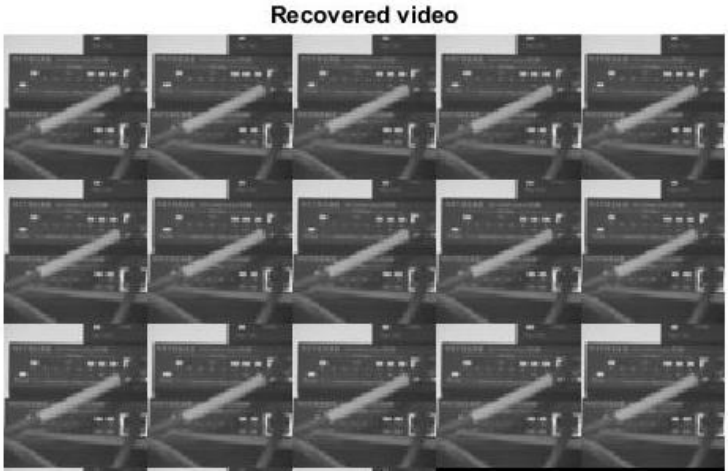



**Figure 4-4 (e)** Various compressed reconstructions of '64cc610''

The figure 4-4 reproduces various reconstructed frames of the DynTex '64cc610' video. The *d* for the reconstruction was 10 whereas the value of the $\breve{M}$ was selected as 3*d*. The value of the $\widetilde{M}$ was selected so that $N/\breve{M}+\widetilde{M}$ maintains the compression ratio as desired. The reconstructed results well preserve the details even at a compression ratio of 100x for the resolution of 256x256. A decrease in reconstruction SNR is observed with an increase in compression ratio while keeping the video resolution constant i.e. at 128x128. However, the increased resolution enhances the reconstruction SNR b almost 10 dBs when the resolution is enhanced from 128x128 to 256 x256. This is because of the fact that more sparse measurements are available with increasing resolution. The detailed data is tabulated in table 4-4.

**Table 4-4** Reconstruction SNR at different Compression Ratios for figure 4-4.

| Figure Label | Resolution | Comp X | Rec SNR |
|---|---|---|---|
| 4-4 (a). | 128x128 | Ground Truth | |
| 4-4 (b). | 256x256 | 100x | 29.346 |
| 4-4 (c). | 128x128 | 100x | 17.535 |
| 4-4 (d). | 128x128 | 50x | 30.644 |
| 4-4 (e) | 128x128 | 20x | 31.932 |

### 4.4. Robustness to noise and reconstruction SNR

It is important to evaluate the resilience of protocol to noise. The input SNR in dBs is defined here as;

$$\text{Input SNR}=10\, log_{10}\left[\frac{(\sum \|a_t\|_2^2)}{(T\sigma^2)}\right], \tag{4.2}$$

where $\sigma^2$ is the variance of the noise. State space estimates for various values of common measurements $\breve{M}$ and different SNRs were analyzed. For cases where $\breve{M} \geq 1$ with $d = 10$ and *T*= 500 frames the reconstruction SNR is high even for small values of $\breve{M}$ and low SNR. The system matrices and the state sequence were generated randomly for each Monte-Carlo run. Reconstruction SNR vs common measurements *M* per frame is plotted in figure-4.4. Each curve indicates different level of measurement noise against specific input SNR. Figure is shown on next page.

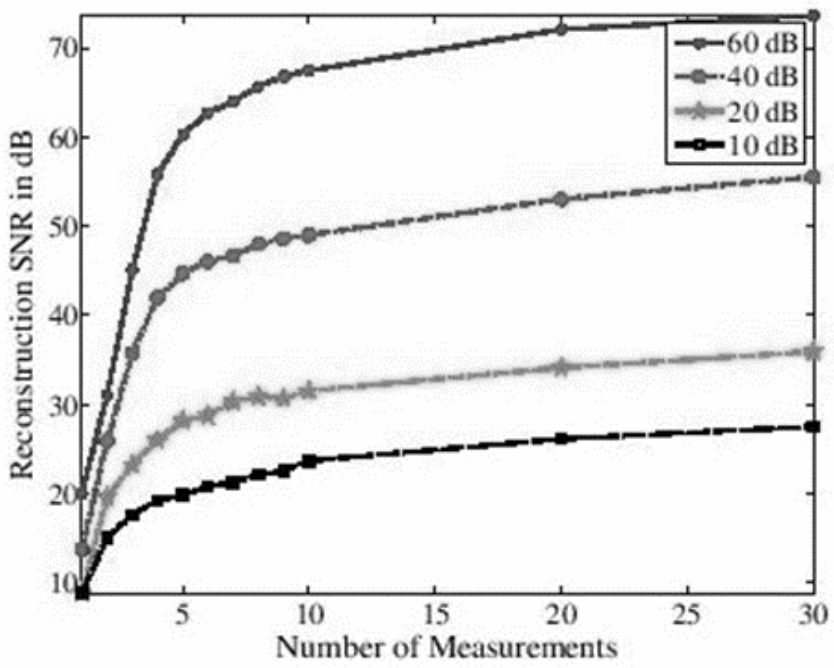



**Figure 4-5** Reconstruction SNR Vs Common Measurement, accuracy of state sequence estimates.

Similarly, for cases where $\breve{M} < 1$ the simulation results are encouraging as the Hankel matrix as envisaged in equation (3.8) is constructed for different missing measurements. Reconstruction SNR of Hankel matrix is depicted in figure 4-5 against number of missing measurements. The plot suggests that Hankel matrix can be reliably reconstructed even if 80 percent measurements are missing.

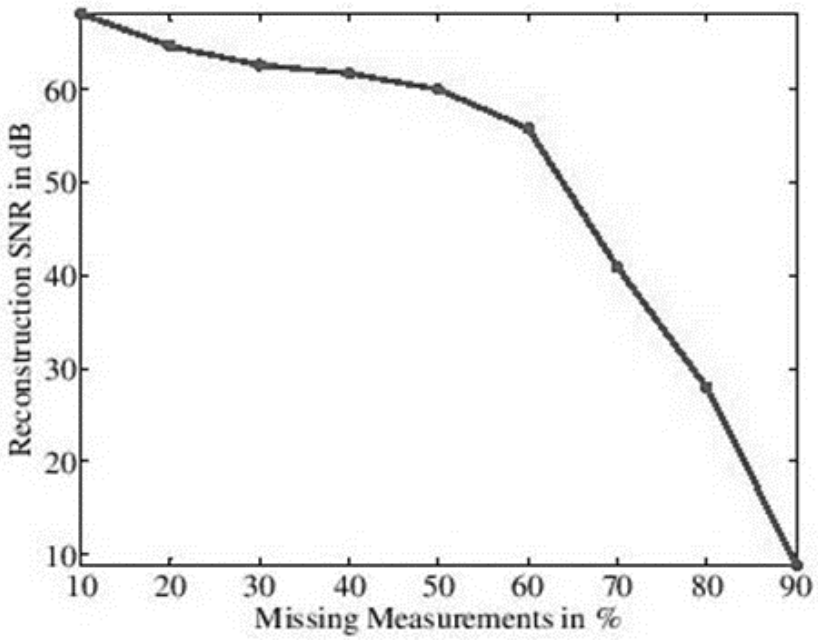



**Figure 4-6** Reconstruction accuracy of Hankel matrix.

### 4.5. Comparison with Frame to Frame CS (FCS) protocol

Another video reconstruction of a dynamic texture '6ammj00' from the DynTex dataset is shown in figure 4-6. The video is of 128X128 pixel resolution. It is a fire texture of length 250 frames. Here the compression is N/M =234. Frame-to-frame CS recovery (FCS) is completely infeasible at such high level of compression. However, the LDS-VCS protocol still effectively recovered the video. The comparatively smaller dynamic component of the scene: i.e. $d$ =20 allowed successful reconstruction of video even from limited measurements. LDS-VCS thus depicts a very high SNR of 22.08 dBs as compared to SNR of only 11.75 dBs for FCS for the recovered videos in figure 4-6. The reconstruction was performed with $d$ = 20 and $K$= 30. The figure shows sampling of frames of the (a) Ground

truth video, (b) LDS-VCS reconstruction, and (c) frame-to-frame (FCS) reconstruction.

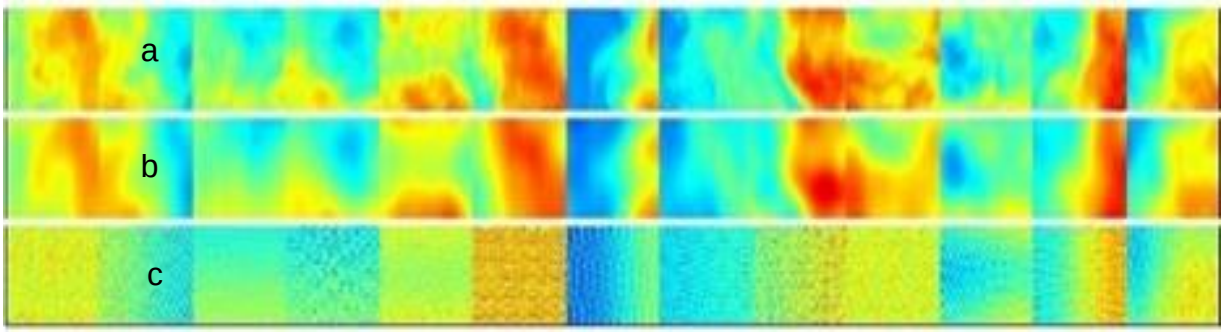



**Figure 4-7 (a-c)** Reconstruction of ‘’6ammj00’ DynTex video using LDS-VCS and FCS: Comparison. (a) Ground Truth (b) LDS-VCS reconstruction (c) FCS reconstruction

# CHAPTER 5

# CONCLUSION AND FUTURE WORK

## 5. Conclusion

The work evaluates a protocol for compressive video sensing of LDSs based models of dynamic textures. The effectiveness of protocol using predictive models is compared with frame by frame CS. It is a conclusion from above results that LDS-VCS performs better in terms of reconstruction SNR when compared with frame by frame CS. The protocol enables stable video reconstructions at very low measurement rates for videos modelled as LDS. It estimates state sequence of corresponding video even when number of common measurements are less than 1.

The thesis presents results of simulations, however, in real world such results are a challenge to achieve. Here, only SPC imaging architecture was considered. Amount of motion in the video and sampling rate of SPC limit desirable compression and resolution results. The idea of 'frame-rate' is not applicable in real-life scenes. However, frame rate of SPC is important and is determined by amount of motion in video. The fast motion is not captured precisely and there is motion blur in fast changing images. The sampling rate of SPC at a desired compression ratio is dependent on the values of K, d and $\tau$ as in equation (3.14). The sampling rate will thus need to increase linearly with N to maintain similar compression level.

In existing technology, the sensing process is independent of recovery methods. Random matrix-based CS measurement techniques also exhibit similar capability.

Any development in recovery methods hence does not affect the sensing part i.e. camera. The protocol presented in this work however, deviates from this property as it translates bi-linear system model created through LDSs modelling to the two-step measurement process. The property can however be utilised to develop new cost-effective sensors in near and far infra-red range on the principle described in this work.

# APPENDIX

## APPENDIX

## MATLAB CODE FOR DEMO.m

```
clear all
close all

addpath('functions')
addpath('utility')
addpath('cosamp');

mycolon = @(x) x(:);


siz = [256 256]/2; %Spatial resolution
Comp = 40; %Compression
d = 20;  %d = LDS state dimension. Reduce this as Comp is increased.
solver = 1; %1 - Cosamp, 0 - Basis pursuit-group sparsity
hank_param = 1; % Used in forming hankel matrix. This is the number of blocks in the
hankel matrix


%This variable selects the sparsity basis
% (spSelect == 1)  %sparsity in a wavelet basis
% (spSelect == 2) %sparsity in a identity basis
% (spSelect == 3) %sparisty in a DCT basis
% (spSelect == 4) %sparisty of mean in wavelet, rest in DCT
% (spSelect == 5) %sparsity of mean in wavelet, rest in identity
spSelect = 4;

fname = 'dyntex/6amg500';
ydata = loadDyntexDataset(fname, siz);

[yrec, c0, Xhat, snr, psnr] = run_cslds(ydata, spSelect, Comp, d, hank_param, solver);

ydata = reshape(ydata, size(ydata, 1), size(ydata, 2), 1, size(ydata, 3));

figure(1)
subplot 121
montage(ydata(:,:,:,1:30:end));
title('Ground truth');
subplot 122
montage(yrec(:,:,:, 1:30:end));
title('Recovered video');

figure(2)
subplot 211
montage(reshape(c0*diag(1./(1e-10+max(c0)+max(-c0))), siz(1), siz(2),1,[]), [-1 1]/2);
colormap jet
axis image; axis off
title('Observation matrix');
subplot 212
```

```
plot(Xhat')
title('state transition')
```

```
Obtaining compressive menasurements
Estimating state sequence

****************************************
**  DWT Extension Mode: Periodization  **
****************************************

Iter: 0001. Err: 0.10325. Diff: 13364

lsqr_iter =

    40

Iter: 0002. Err: 0.04736. Diff: 8050
Iter: 0003. Err: 0.04386. Diff: 4156
Iter: 0004. Err: 0.04177. Diff: 2307
Iter: 0005. Err: 0.04176. Diff: 600
Iter: 0006. Err: 0.04219. Diff: 361
Iter: 0007. Err: 0.04207. Diff: 354
Iter: 0008. Err: 0.04207. Diff: 244
Iter: 0009. Err: 0.04186. Diff: 149
Iter: 0010. Err: 0.04181. Diff: 166
Final results.
 Compression = 40 x
 Reconstruction SNR = 26.433 dB
 Peak SNR =  33.762 dB
```

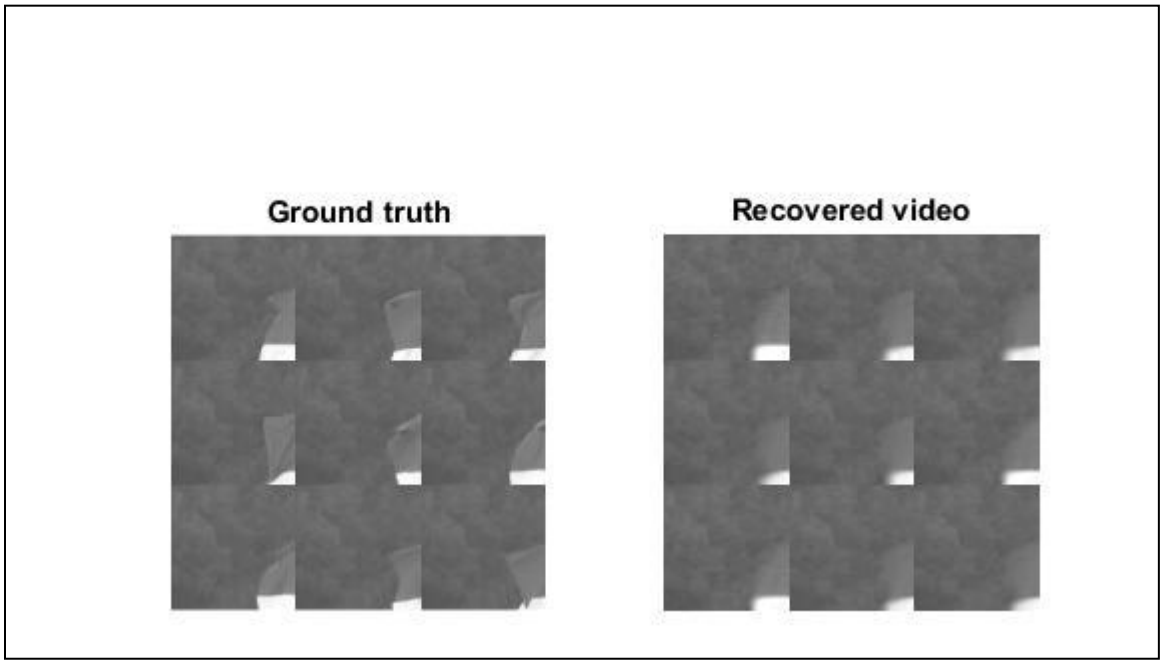



**Figure A-1 .** Reconstructed result for Dyntex ‘6amg500’

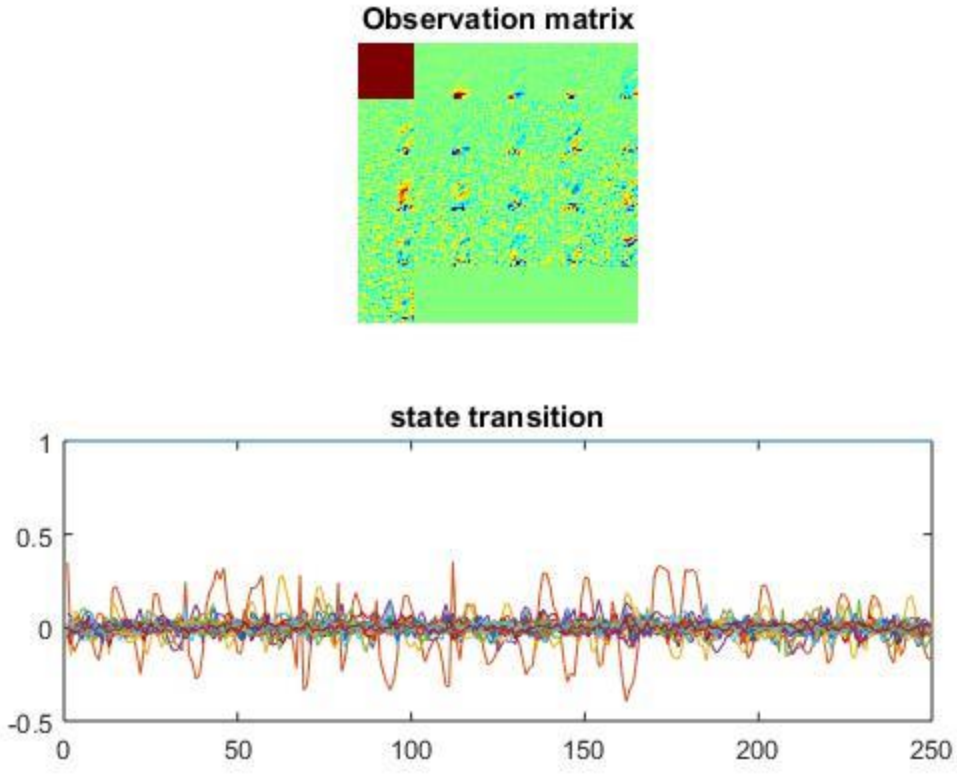



**Figure A-2** Correspondix Observation and State transition Matrices for ‘6amg500’

## LDS-VCS MATLAB CODE

```
function [yrec, c0, Xhat, snr, psnr] = run_cslds(ydata, spSelect, Comp, d, hank_param,
solver)
%%%%%Simulate LDS-VCS on a video
%%%%% Look at "Compressive Acquisition of Linear Dynamical Systems" Thesis
%%%%% submitted to MCS, NUST
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input
%    ydata --- 3D video cube.
%    spSelect --- Sparsity basis for the observation matrix
%                 (spSelect == 1)  %sparsity in a wavelet basis
%                 (spSelect == 2) %sparsity in a identity basis
%                 (spSelect == 3) %sparisty in a DCT basis
%                 (spSelect == 4) %sparisty of mean in wavelet, rest in DCT
%                 (spSelect == 5) %sparsity of mean in wavelet, rest in identity
%    Comp  --- Compression factor.
%    d     --- LDS state space dimension
%    solver --- 1 or 0
%              1 -- uses cosamp code in 'cosamp' folder
%              0 -- uses spg_bpdn (needs spgl1 package)
%
% Output
%    yrec ---- reconstructed video
%    c0    --- estimated observation matrix
%    Xhat  --- Estiamted state sequence
%    snr, psnr --- SNR and Peak SNR
%
%

addpath('functions')
addpath('utility')
addpath('cosamp');

mycolon = @(x) x(:);

%loading dataset
siz = [ size(ydata, 1) size(ydata, 2)];
N = prod(siz);
T = size(ydata, 3);



M = floor(N/Comp);
Mhat = 3*d;    %Feel free to change this
Mtilde = M - Mhat;
if (Mtilde < 0)
    Mhat = floor(M/2);
    Mtilde = M - Mhat;
end

%%Get Compressive measurements
%I am using noiselets for speed
```

```
%Noiselt code is in "utility" folder
idx2 = randperm(N); %column permutation
idx = zeros(M, T);
for kk=1:T
    tmp = randperm(N);
    idx(:, kk) = tmp(1:M);
end

%this next step ensures that we obtain "common" measurements for each frame
idx(1:Mhat, :) = idx(1:Mhat,1)*ones(1, T);

%Obtain compressive measurements
disp('Obtaining compressive menasurements');
z = zeros(M, T);
for kk=1:T
    z(:, kk) = Aoperator_noiselet( mycolon(ydata(:, :, kk)), idx(:, kk), idx2);
end


%%%%%%%
% CSLDS Starts here
%%%%%%%%%%%%%%%%%%%%%%%
%GET STATE SEQUENCE
disp('Estimating state sequence');
%We use a simple version of the Hankel matrix here (just the top block).
zhat = z(1:Mhat, :);

if (spSelect >= 4)
    zhat = zhat - mean(zhat, 2)*ones(1, T); %Subtracting mean
end

hankmat = formHankelMatrix( zhat, hank_param);
[Uz, Sz, Vz] = svd(hankmat);
dhat = d; %Can use heuristics here
Xhat = (Sz(1:dhat, 1:dhat))*Vz(:, 1:dhat)';
Xhat = [ Xhat Xhat(:, end)*ones(1, hank_param-1) ];

if (spSelect >= 4)
    Xhat = [ ones(1, T); Xhat]; %The ones(1, T) incorporates the mean term
    dhat = dhat + 1;
end

%GET OBSERVATION MATRIX
wave.name = 'db4';
wave.level = 5;
dwtmode('per');
[tmp, wave.Cbook] = wavedec2(randn(siz), wave.level, wave.name);

%Function handles for the nosielet operator
funMeas = @(Cvar) ANoiselet_Cmat(Cvar, idx, idx2, Xhat, N, dhat);
funMeasTr = @(zVar)  ATNoiselet_Cmat(zVar, idx, idx2, Xhat, N, dhat);

if (spSelect == 1)  %sparsity in a wavelet basis
    funSparse = @(Svar) COperator_wavelet(Svar, wave, N, dhat, siz);
```

```
    funSparseTr = @(Cvar) CTOperator_wavelet(Cvar, wave, N, dhat, siz);
end
if (spSelect == 2) %sparsity in a identity basis
    funSparse = @(x) x(:);
    funSparseTr = @(x) x(:);
end
if (spSelect == 3) %sparisty in a DCT basis
    funSparse = @(Svar) COperator_dct(Svar, N, dhat, siz);
    funSparseTr = @(Cvar) CTOperator_dct(Cvar, N, dhat, siz);
end
if (spSelect == 4) %sparisty of mean in wavelet, rest in DCT
    funSparse = @(Svar) COperator_wavelet_dct(Svar, wave, N, dhat, siz);
    funSparseTr = @(Cvar) CTOperator_wavelet_dct(Cvar, wave, N, dhat, siz);
end
if (spSelect == 5) %sparsity of mean in wavelet, rest in identity
    funSparse = @(Svar) COperator_wavelet_identity(Svar, wave, N, dhat, siz);
    funSparseTr = @(Cvar) CTOperator_wavelet_identity(Cvar, wave, N, dhat, siz);
end

%form compound measurement operator
A = @(x) funMeas(funSparse(x));
At = @(x) funSparseTr(funMeasTr(x));

Kmax = floor((M*T - Mhat*T)/(5*d)); %Feel free to change this
K1 = min(2*Kmax, prod(siz)); %Sparsity of mean
K2 = min(prod(siz), floor(Kmax*2/3)); %sparsity of the rest

MaxIter = 10; %A small value is usally good enough
tol = 1e-3;

if (solver)
    if (spSelect >= 4)
        grp = [N dhat-1]; %this tells the cosamp code on the grouping pattern of the
structured sparisty
        s0 = Cosamp_mean_cslds(z(:), A, At, K1, K2, grp, MaxIter, tol, 'cgs');
    else
        grp = [N dhat]; %this tells the cosamp code on the grouping pattern of the
structured sparisty
        s0 = Cosamp_groupsparsity(z(:), A, At, Kmax, grp, MaxIter, tol, 'cgs');
    end
else
    fSPG = @(x, mode) spg_wrapper(x, mode, A, At);
    opt = spgSetParms; opt.iterations = 300;
    opt.verbosity = 1;
    groups = [ (1:prod(siz))' (prod(siz)+(1:prod(siz))')*ones(1, d)];
    [s0,r,g,info] = spg_group( fSPG, z(:), groups, norm(z(:))/100, opt );
end

c0 = funSparse(s0);
c0 = reshape(c0, [N dhat]);

yrec = c0*Xhat;

snr = -20*log10( norm(yrec(:) - ydata(:))/norm(ydata(:)));
```

```
psnr = -20*log10( norm(yrec(:) - ydata(:))/(sqrt(length(yrec(:)))*max(abs(ydata(:)))));


disp(sprintf('Final results. \n Compression = %d x \n Reconstruction SNR = %3.3f dB ',
Comp, -20*log10( norm(yrec(:) - ydata(:))/norm(ydata(:)))))
disp(sprintf(' Peak SNR = % 3.3f dB',-20*log10( norm(yrec(:) -
ydata(:))/(sqrt(length(yrec(:)))*max(abs(ydata(:)))))))


ydata = reshape(ydata, siz(1), siz(2), 1,  []);
yrec = reshape(yrec, siz(2), siz(2), 1, []);
```

## GROUP SPARSITY BASED CoSAMP

```
function sCosamp = Cosamp_groupsparsity(b, A, At, K, grp, MaxIter, tol, method)
%%%%%
% solves b = A(sCosamp) using a model-based cosamp algorithm
%
% b - observation vector
% A, At - forward/adjoint in a functional form. Includes sparsifying operator
% K1 - sparsity of the mean
% K2 - sparsity of the remainder
% grp - [N d] where N = number of pixels, d = LDS state dim (doesnt include
%        mean term)
% MaxIter - max number of cosamp iterations
% tol - stopping criterion on normalized residue
% method - for solving least sqyares problem below. use 'cgs' for conj
%          gradient, 'lsqr' for least sqyare

M = length(b);
N = length(At(b));

y = b;
iter = 0;
S_old = [];
S_tilde_old = [];
lsqr_iter = 100;


while ((iter < MaxIter) & (norm(y)/norm(b) > tol))
    iter = iter + 1;

    %Support discovery
    r = At(y);
    r_model = signalToModel(r, grp);
    [tmp, idx_m] = sort(r_model, 'descend');
    idx_m = idx_m(1:K);

    idx = modelToSupp(idx_m, grp);

    S_tilde = [S_old; idx(:) ];


    %Least Squares over S_tilde
    if (strcmp(method, 'lsqr'))
        funcA = @(x,t) AHandle(x, t, A, At, S_tilde, N);
        [a, flag] = lsqr(funcA, b, 1e-3, lsqr_iter);
    end
    if (strcmp(method, 'cgs'))
        funcA = @(x,t) AHandle(x, t, A, At, S_tilde, N);
        funcB = @(x) funcA(funcA(x, 'notransp'), 'transp');
        [a , flag] = cgs(funcB, funcA(b, 'transp'), 1e-3, lsqr_iter);
    end
    if (flag == 1)
        lsqr_iter = lsqr_iter*2
```

```
    end
    stmp = zeros(N, 1); stmp(S_tilde) = a;
    r_model = signalToModel(stmp, grp);

    [tmp, idx_m] = sort(r_model, 'descend');
    idx_m = idx_m(1:K);

    idx = modelToSupp(idx_m, grp);

    s0 = zeros(N, 1); s0(idx) = stmp(idx);
    y = b - A(s0);

    err = norm(y)/norm(b);
    disp(sprintf('Iter: %04d. Err: %2.5f. Diff: %d', iter, err, length(setdiff(idx,
S_old))));

    if length(setdiff(idx, S_old)) < length(S_old)/50
        disp('Converged');
        break;
    end



    S_old = idx;
    S_tilde_old = S_tilde;


end
sCosamp = s0;

function y = AHandle(x, t, A, At, Supp, N)
if strcmp(t,'transp')
    s = At(x);
    y = s(Supp);
elseif strcmp(t,'notransp')
    s = zeros(N, 1);
    s(Supp) = x;
    y = A(s);
end


function y = signalToModel(x, grp)
    x = x.^2;
    x = reshape(x, grp);
    y = sum(x, 2);

function y = modelToSupp(x, grp)

    x1 = x(:)*ones(1, grp(2));
    x2 = ones(length(x(:)),1)*(0:grp(2)-1);
    y = x1+grp(1)*x2;
    y = y(:);
```

## CoSAMP MODEL BASED ALGORITHM-2

```
function sCosamp = Cosamp_mean_cslds(b, A, At, K1, K2, grp, MaxIter, tol, method)
%%%%%%
% solves b = A(sCosamp) using a model-based cosamp algorithm
%
% b - observation vector
% A, At - forward/adjoint in a functional form. Includes sparsifying operator
% K1 - sparsity of the mean
% K2 - sparsity of the remainder
% grp - [N d] where N = number of pixels, d = LDS state dim (doesnt include
%        mean term)
% MaxIter - max number of cosamp iterations
% tol - stopping criterion on normalized residue
% method - for solving least sqyares problem below. use 'cgs' for conj
%          gradient, 'lsqr' for least sqyare


M = length(b);
N = length(At(b));


y = b;
iter = 0;
S_old = [];
S_tilde_old = [];
lsqr_iter = 20;

while ((iter < MaxIter) & (norm(y)/norm(b) > tol))
    iter = iter + 1;

    %Support discovery
    r = At(y);
    r1 = r(1:grp(1)); r2 = r(grp(1)+1:end);

    [tmp, idx1] = sort(abs(r1), 'descend');
    idx1 = idx1(1:K1);

    r_model = signalToModel(r2, grp);
    [tmp, idx_m] = sort(r_model, 'descend');
    idx_m = idx_m(1:K2);
    idx2 = modelToSupp(idx_m, grp);

    idx = [idx1(:); idx2(:)];
    S_tilde = [S_old; idx(:) ];


    %Least Squares over S_tilde
    if (strcmp(method, 'lsqr'))
        funcA = @(x,t) AHandle(x, t, A, At, S_tilde, N);
        [a, flag] = lsqr(funcA, b, 1e-3, lsqr_iter);
    end
    if (strcmp(method, 'cgs'))
```

```
        funcA = @(x,t) AHandle(x, t, A, At, S_tilde, N);
        funcB = @(x) funcA(funcA(x, 'notransp'), 'transp');
        [a , flag] = cgs(funcB, funcA(b, 'transp'), 1e-3, lsqr_iter);
    end
    if (flag == 1)
        lsqr_iter = lsqr_iter*2
    end

    stmp = zeros(N, 1); stmp(S_tilde) = a;

    stmp1 = stmp(1:grp(1)); stmp2 = stmp(grp(1)+1:end);

    [tmp, idx1] = sort(abs(stmp1), 'descend');
    idx1 = idx1(1:K1);

    r_model = signalToModel(stmp2, grp);

    [tmp, idx_m] = sort(r_model, 'descend');
    idx_m = idx_m(1:K2);
    idx2 = modelToSupp(idx_m, grp);

    idx = [idx1(:); idx2(:)];
    s0 = zeros(N, 1); s0(idx) = stmp(idx);
    y = b - A(s0);

    err = norm(y)/norm(b);
    %disp(sprintf('Iter: %04d. Err: %2.5f. Diff: %d', iter, err, length(setdiff(idx,
S_old))));
    disp(sprintf('Iter: %04d. Err: %2.5f. Diff: %d', iter, err, length(setdiff(idx,
S_old))));
    if length(setdiff(idx, S_old)) < length(S_old)/100
        disp('Converged');
        break;
    end


    S_old = idx;
    S_tilde_old = S_tilde;


end
sCosamp = s0;

function y = AHandle(x, t, A, At, Supp, N)
if strcmp(t,'transp')
    s = At(x);
    y = s(Supp);
elseif strcmp(t,'notransp')
    s = zeros(N, 1);
    s(Supp) = x;
    y = A(s);
end
```

```
function y = signalToModel(x, grp)
    x = x.^2;
    x = reshape(x, grp(1), []);
    y = sum(x, 2);

function y = modelToSupp(x, grp)

    x1 = x(:)*ones(1, grp(2));
    x2 = ones(length(x(:)),1)*(1:grp(2));
    y = x1+grp(1)*x2;
    y = y(:);
```

## LOAD DYNTEX DATA FROM FILE

```
function ydata = loadDyntexDataset(fname, siz)


fdir = dir([ fname '/*.jpg']);

ydata = zeros(siz(1), siz(2), length(fdir));
for kk=1:length(fdir)
    img = imread( [fname '/' fdir(kk).name ]);
    img = double(img)/255;
    img = mean(img, 3);

    img = img(15+(1:256), 50+(1:256));
    img = imresize(img, siz, 'bilinear');

    ydata(:,:, kk) = img;
end
```

Published with MATLAB® R2015a

# REFERENCES

# REFERENCES


[1] E. J. Candes, J. Romberg, and T. Tao, 'Robust uncertainty principles: Exact signal reconstruction from highly in-complete frequency information,' IEEE Trans. Inf. Theory, vol. 52, pp. 489–509, Feb. 2006.

[2] D. L. Donoho, 'Compressed sensing,' IEEE Trans. Inf. Theory, vol. 52, pp. 1289–1306, Apr. 2006.

[3] M.A. Davenport, et al., *Introduction to Compressed Sensing.* Preprint 93.1 (2011): 2 Available: http://www.dfg-spp1324.de/download/preprints/preprint093.pdf

[4] D. E. Lazich, (2014, 23 April). *Compressed Sensing- Efficient Information Acquisition*. Available: http://www.uni-ulm.de/in/nt/teaching/lectures/summer-2014/nt-cs.html

[5] R. G. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, 'A simple proof of the restricted isometry property for random matrices', Constr. Approx., 28 (2008), pp. 253–263.

[6] J. Haupt and R. Nowak, 'Signal reconstruction from noisy random projections', IEEE Trans. Inf. Theory, 52 (2006), pp. 4036–4048.

[7] E. van den Berg and M. P. Friedlander, 'Probing the pareto frontier for basis pursuit solutions', SIAM J. Scientific Comp., 31 (2008), pp. 890–912.

[8] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, 'Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition', in Asilomar Conf. Signals Sys. Comp., Nov. 1993.

[9] D. Needell and J. A. Tropp, *Cosamp:* 'Iterative signal recovery from incomplete and inaccurate samples', Appl. Comp. Harm. Anal., 26 (2009), pp. 301–321.

[10] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde, 'Model-based compressive sensing', IEEE Trans. Inf. Theory, 56 (2010), pp. 1982–2001.

[11] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, “Single-pixel imaging via compressive sampling,” IEEE Signal Process. Mag., vol. 25, pp. 83–91, Mar. 2008.

[12] M. Gupta, A. Agrawal, A. Veeraraghavan, and S. Narasimhan, “Flexible voxels for motion-aware videography,” in Euro. Conf. Comp. Vision, (Crete, Greece), Sep. 2010.

[13] D. Reddy, A. Veeraraghavan, and R. Chellappa, “P2C2: Programmable pixel compressive camera for high speed imaging,” in IEEE Conf. Comp. Vision and Pattern Recog, (Colorado Springs, CO, USA), June 2011.

[14] R. G. Baraniuk, T. Goldstein, A. C. Sankaranarayanan, C. Studer, A. Veeraraghavan, and M. B. Wakin, “Compressive video sensing: Algorithms, architectures, and applications” IEEE Signal Process. Mag., vol. 34, pp. 52–66, Jan. 2017.

[15] J. Y. Park and M. B. Wakin, “A multiscale framework for compressive sensing of video,” in Pict. Coding Symp., (Chicago, IL, USA), May 2009.

[16] A. C. Sankaranarayanan, P. Turaga, R. Baraniuk, and R. Chellappa, “Compressive acquisition of dynamic scenes,” in Euro. Conf. Comp. Vision, (Crete, Greece), Sep. 2010.

[17] N. Vaswani, “Kalman filtered compressed sensing,” in IEEE Conf. Image Process., (San Diego, CA, USA), Oct. 2008.

[18] M. B. Wakin, J. N. Laska, M. F. Duarte, D. Baron, S. Sarvotham, D. Takhar, K. F. Kelly, and R. G. Baraniuk, “Compressive imaging for video representation and coding,” in Pict. Coding Symp., (Beijing, China), Apr. 2006.

[19] S. Mun and J. E. Fowler, “Residual reconstruction for block- based compressed sensing of video,” in Data Comp. Conf., (Snowbird, UT, USA), Apr. 2011.

[20] A. Veeraraghavan, D. Reddy, and R. Raskar, “Coded strobing photography: Compressive sensing of high speed periodic events,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, pp. 671–686, Apr. 2011.

[21] A. C. Sankaranarayanan, C. Studer, R. G. Baraniuk, “CS-MUVI: Video compressive sensing for spatial multiplexing cameras,” in proceedings of IEEE Intl. Conf. Comp. Photography, (Seattle, WA, USA) ICCP-2012, DOI:10.1109/ICCPhot.2012.6215212

[22] Y. Hitomi, J. Gu, M. Gupta, T. Mitsunaga, and S. K. Nayar, “Video from a single coded exposure photograph using a learned over-complete dictionary,” in IEEE Intl. Conf. Comp. Vision, (Barcelona, Spain), Nov. 2011.

[23] D. Mahajan, F. C. Huang, W. Matusik, R. Ramamoorthi, and P. Belhumeur, “Moving gradients: A path-based method for plausible image interpolation,” ACM Trans. Graph., vol. 28, pp. 1–42, Aug. 2009

[24] A. Mousavi, A. Patel, and R. G. Baraniuk, “A deep learning approach to structured signal recovery,” in Proc. 53rd Annu. Allerton Conf. Communication, Control, and Computing, Monticello, IL, 2015.

[25] A. C. Sankaranarayanan, L. Xu, C. Studer, Y. Li, K. F. Kelly, and R. G. Baraniuk, ‘Video compressive sensing for spatial multiplexing cameras using motion-flow models’, SIAM J. Imag. Sci., vol. 8, no. 3, pp. 1489–1518, 2015

[26] A. B. Chan and N. Vasconcelos, ‘Probabilistic kernels for the classification of auto-regressive visual processes’, in IEEE Conf. Comp. Vision and Pattern Recog, June 2005.

[27] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto, ‘Dynamic textures’, Intl. J. Comp. Vision, 51 (2003), pp. 91–109.

[28] P. Saisan, G. Doretto, Y. Wu, and S. Soatto, ‘Dynamic texture recognition’, in IEEE Conf. Comp. Vision and Pattern Recog, Dec. 2001.

[29] T. Ding, M. Sznaier, and O. I. Camps, ‘A rank minimization approach to video inpainting’, in IEEE Intl. Conf. Comp. Vision, 2007

[30] M. Ayazoglu, B. Li, C. Dicle, M. Sznaier, and O. I. Camps, ‘Dynamic subspace-based coordinated multicamera tracking’, in IEEE Intl. Conf. Comp. Vision, 2011

[31] M. Sznaier, ‘Compressive information extraction: A dynamical systems approach, in System Identification’, vol. 16, 2012, pp. 1559–1568

[32] P. Van Overschee and B. De Moor, ‘N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems’, Automatica, 30 (1994), pp. 75–93.

[33] S. Soatto, G. Doretto, and Y. N. Wu, ‘Dynamic textures’, in IEEE Intl. Conf. Comp. Vision, July 2001

[34] R. Peteri, S. Fazekas, and M.J. Huiskes, ‘DynTex: A comprehensive database of dynamic textures’, Pattern Recog. Letters, 31 (2010), pp. 1627–1632.

[35] K. Shah, (2017, May). *Introduction to linear Dynamical Systems.* Available: http://ee263.stanford.edu/lectures.html

[36] R. W. Brockett, ‘Finite Dimensional Linear Systems’, Wiley, 1970.

[37] A. C. Sankaranarayanan, P. K. Turaga, R. Chellappa, and R. G. Baraniuk, ‘Compressive Acquisition of Linear Dynamical Systems*’,* SIAM J. Imaging Sci. 6-4 (2013), pp. 2109-2133.

[38] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T. W. Lee, and T. J. Sejnowski, ‘Dictionary learning algorithms for sparse representation’, Neural Comp., 15 (2003), pp. 349–396

[39] M. F. Duarte, M. B. Wakin, D. Baron, S. Sarvotham, and R. G. Baraniuk, ‘Measurement bounds for sparse signal ensembles via graphical models’, IEEE Trans. Inf. Theory, 59 (2013), pp. 4280–4289