4-8-2013

An Adaptive Optimal Bandwidth Sensor for Video Imaging and Sparsifying Basis

Imama Noor

AN ADAPTIVE OPTIMAL BANDWIDTH SENSOR FOR VIDEO IMAGING AND SPARSIFYING BASIS

by

Imama Noor

A Dissertation

Submitted in Partial Fullfillment of the

Requirements for the Degree of

Doctor of Philosophy

Major: Engineering

The University of Memphis

May 2013

Acknowledgements

First of all, I would like to thank my research advisor and com- mittee chair Dr. Eddie L. Jacobs for his tireless and excellent advice and encouragement throughout my graduate studies. I would also like to express my sincere appreciation and gratitude to Dr. Bonny Banerjee, Dr. Srikant Chari and Dr. Aaron Robinson for serving on my committee. Last but not least, I would like to thank my mother (Mrs. Rubia Karim) for her consistent support throughout my studies, without her none of this would have been possible.

ABSTRACT

Noor, Imama. PhD. The University of Memphis. May, 2013. An Adaptive Optimal Bandwidth Sensor for Video Imagining and Sparsifying Basis. Major Professor: Dr. Eddie Jacobs

Many compressive sensing architectures have shown promise towards reducing the bandwidth for image acquisition significantly. In order to use these architectures for video acquisition we need a scheme that is able to effectively exploit temporal redundancies in a sequence. In this thesis we study a method to efficiently sample and reconstruct specific video sequences. The method is suitable for implementation using a single pixel detector along with a digital micromirror device (DMD) or other forms of spatial light modulators (SLMs). Conventional implementations of single pixel cameras are able to spatially compress the signal but the compressed measurements make it difficult to exploit temporal redundancies directly. Moreover a single pixel camera needs to make measurements in a sequential manner before the scene changes making it inefficient for video imaging. In this thesis we discuss a measurement scheme that exploits sparsity along the time axis for video imaging. After acquiring all measurements required for the first frame, measurements are only acquired from the areas which change in subsequent frames. We segment the first frame and detect magnitude and direction of change for each segment and acquire compressed measurements for the changing segments in the predicted direction. TV minimization is used to reconstruct the dynamic areas and PSNR variation is studied against different parameters of proposed scheme. We show the reconstruction results for a few test sequences commonly used for performance analysis and demonstrate the practical utility of the scheme. A comparison is made with existing algorithms to show the effectiveness of proposed method for specific video sequences. We also discuss use of customized transform to improve reconstruction of submil-

limeter wave single pixel imager. We use a sparseness inducing transformation on the measurements and optimize the result using l1 minimization algorithms. We demonstrate improvement in result of several images acquired and reconstructed using this technique.

Table Of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation and Roadmap

Since the inception of digital signal processing there is an ever increasing demand for faster and more robust devices. The amounts of data generated using classical acquisition techniques is increases the burden on analog to digital convertors and pushes them to limits. It is no longer possible to meet the growing demands while still adhering to the classical Shannon Nyquist sampling rate. Fortunately we are able to understand the nature of signal processing better then before. We no longer have to treat all sorts of signals in a similar manner. Instead we can adjust the sampling rate depending on the information content in any signal.

The majority of natural and man made signals have an inherent structure which is not used in classical methods for acquisition. These signals when transformed have their energy concentrated in a few significant basis vectors and most coefficients are negligible. It has be shown that sampling in the specific sparsity domain reduces the number of samples compared to classical Nyquist sampling technique. The marriage of transform knowledge and sparse sampling schemes has generated a new framework for signal acquisition called compressive sampling. A single pixel camera (SPC) is designed to make compressive measurements of a scene to render still images. The SPC is designed to acquire samples sequentially taking one sample at a time. This approach cannot be translated directly applied to video acquisition. In this thesis we propose an algorithm to acquire measurements compressivley for video acquisition.

For the remainder of the chapter we give an overview of Compressive Sensing and some associated concepts. In Chapter 2, we detail the prior work done on static and dynamic compressive sensing. In Chapter 3, we present an adaptive scheme for

dynamic scene acquisition. In Chapter 4, we analyse utility of a dictionary learning scheme.

1.2 Compressive Sensing

The compressed sensing framework for image acquisition exploits properties of a signal to reduce the number of samples required for reconstruction. It exploits the sparsity hidden in the signal and acquires measurements in the domain where the signal is sparse. The measurement matrices or projections are carefully designed to acquire maximum information in the signal. Random projections have been shown to have this property. Many other forms of measurement matrices have also been shown to yield good results.

The CS framework captures the higher frequency information while sampling below Nyquist rate. The signal and sample relationship can be expressed as

$$y = Ax + \omega \tag{1.1}$$

Here ω incorporates the detection and quantization noise and A is the measurement matrix. The variable y is an $m \times 1$ vector of measurements and x is the signal $n \times 1$ with $m < n$. The value of m depends on the sparsity of signal and the coherence between the measurement matrix and the sparsity basis. The relationship is mentioned in its mathematical form later in this chapter.

1.2.1 Signal and Sparse Representations

If x is not sparse in the spatial domain we can express x in a sparse basis ϕ such that z is a sparse vector.

$$x = \phi z$$

Therefore, the measurement model can be rewritten as

$$y = A\phi z + \omega \tag{1.2}$$

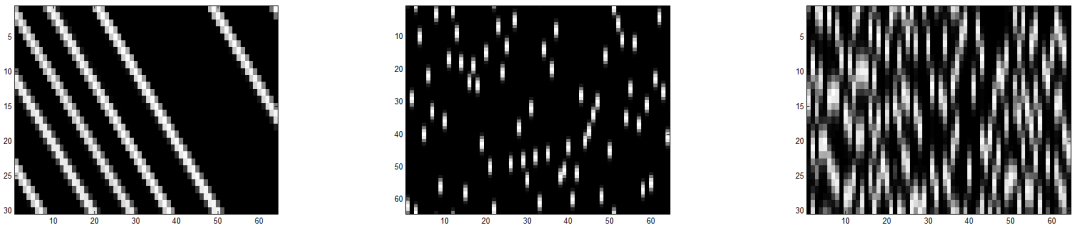
The knowledge of the sparse basis gives information about the structure of the signal to be reconstructed. The basis vectors are also known as atoms. If a signal is sparse in a specific basis, only a few atoms are required to represent the signal precisely. The sparse basis needs to be incoherent with the measurement matrix in order to reduce the number of samples required for reconstruction.

1.2.2 Dictionary Learning

CS theory requires the signal to be sparse in a specific domain for exact reconstruction. Many off-the-shelf sparse basis functions are available but they can only sparsely represent a subset of signals. In order to find a basis customized to a particular class of signals, we can pick key features in the set and form a dictionary. There are many ways to find a dictionary which can represent the signal as a linear combination of few atoms. I used an online learning approach for sparse coding. The algorithm considers a finite training set of signals and optimizes an empirical cost function

$$f_n(\phi) \triangleq \frac{1}{n} \sum l(x_i, \phi) \tag{1.3}$$

where ϕ is the dictionary in $R^{m \times n}$. Each column of ϕ is called an atom which are the sparse basis vectors. The variables x_i are the training signals in $R^{m \times k}$. The function l is the loss function which is small when ϕ represents x_i using a few sparse basis vectors or atoms. The number of training signals k is typically large and each signal is represented by only a few columns of ϕ . This approach is fast and guaranteed to converge.



(a) (b) (c)

Figure 1.1: a. 30x64 Matrix M b. A 64 x 64 Dictionary D c. Product of D and M

1.2.3 Mutual Coherence

A measurement matrix is formed based on a couple of rules. First, the measurement matrix should be incoherent with the basis matrix. A constant term used to determine the number of measurements depends on the coherence measure of the measurement matrix A and the sparsity basis ϕ. If a_i and ψ_j represent the columns of A and ϕ respectively then

$$\mu = max_{1 \leq i,j \leq n} |\langle a_i, \psi_j \rangle| \tag{1.4}$$

The matrices are incoherent if

$$n^{-1/2} \leq \mu \leq 1 \tag{1.5}$$

To be able to reconstruct from fewer samples we need to make measurements such that they are incoherent with the sparsity basis matrix. The eigenvalues of the gram matrix should be almost equal to one in the case where the two matrices are incoherent.

$$A\phi^T \approx I \tag{1.6}$$

1.2.4 Restricted Isometry Property(RIP)

In order to fully recover signal from the undersampled data, the measurement matrix needs to satisfy the restricted isometry property(RIP). The RIP implies that the measurement matrix conserves the energy in the signal, which further implies that the measurement matrix have a zero mean. This condition can be formally written as

$$(1-\delta_S)||z||_2^2 \leq ||A\phi z||_2^2 \leq (1+\delta_S)||z||_2^2 \tag{1.7}$$

where z is the sparse vector, δ_s is the RIP constant and lies between 0 and 1. Another way of stating the restricted isometry property is that it ensures that the transformation $A\phi$ preserves the distances between the nonzero planes of sparse vectors. This is equal to the requirement that the largest eigenvalue of $A\phi(A\phi)^T$ lies in between the interval $[1+\delta_s, 1-\delta_s]$. Let $Z, P \subseteq \mathbb{R}^{\mathbb{N}}$then the definition can be also be expressed as

$$(1-\delta_S)||z-p||_2^2 \leq ||A\phi z - A\phi p||_2^2 \leq (1+\delta_S)||z-p||_2^2 \tag{1.8}$$

for all $z \epsilon Z$ and $p \epsilon P$. In order to verify this property, all subsets of S columns taken from A are nearly orthogonal, and all pairwise distances between S-sparse signals can be well preserved in the measurements space. This makes verification of RIP a NP-hard problem. Random, iid Gaussian and Bernoulli matrices have proven to satisfy the RIP. The bounds on the number of measurements required to reconstruct a k-sparse signal have been established for these matrices as well. The figure shows transformation by an RIP verified matrix. The distances between k-sparse planes of the signal is conserved under the transformation by ϕ. If the entries of A are iid sampled from N(0, 1) Gaussian or U(-1, 1) Bernoulli than we

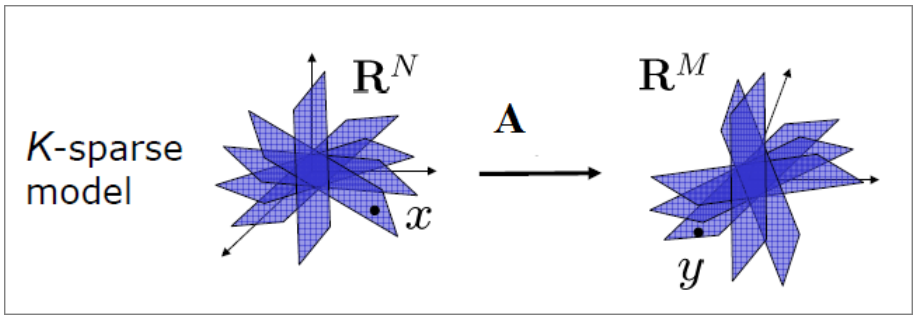


Figure 1.2: Distances between K-planes is conserved under transformation ϕ

can reconstruct the original signal with probability $1 - e^{-C_n}$ with $O(klog(C/k))$ complexity. Where A has a RIP of order (k, δ).

1.2.5 Null Space Property

A matrices capability to capture key information in the signal and allow accurate reconstruction can be verified by the Null-Space property(NSP). The Null-Space of a matrix is the solution space for a homogeneous linear system of equations. These set of equations are formed when there is no driving signal. Mathematically NSP for all vectors s in the null space of ϕ is evaluated based on

$$||s||_{k,1} \leq \alpha_k ||s||_1 \tag{1.9}$$

This holds for all vectors s in the null space of A (i.e Ax=0) where $||s||_{k,1}$ denotes the sum of the k largest absolute values of entries in s. Here $\alpha_k \epsilon$ [0, 1] and $||s||_1$is the l_1 norm of vector s. If a matrix satisfies the NSP of order $(2k, \alpha)$ for α belonging to [0,1] then it will satisfy the RIP of third order $(3k, \delta_k)$.

1.2.6 k-Neighbourliness Condition

Neighborliness can also be used as a measure to verify if the measurement matrix is appropriate for fully recovering a signal from under sampled data. If we

define $P = AL$, where L denotes cross polytopes in $\mathbb{R}^{\mathbb{N}}$and A is the measurement matrix then the convex polytope P should be centrally k-neighborly. This implies every subset of k vertices forms k-1 faces.[4] This property has been verified for random matrices.[4] A random matrix $n \times d$ with k =.089d and n = 2d is k-neighborly and a sparse solution is possible with far less computation if the solution exists.

1.2.7 Recovery Conditions

There are lower bounds on the number of samples required for reconstructing the signal accurately. It is directly proportional to the sparsity index of a signal and varies at log scale with the dimensionality of the signal.

Theorem 1(*sparse reconstruction*) : If we have an n-dimensional signal which is k-sparse in ϕ, then given m measurements such that

$$m \geq C \cdot \mu(A, \phi) \cdot k \cdot log\, n \tag{1.10}$$

the signal can be recovered exactly, with overwhelming probability. Here C is a constant, μ is the coherence measure between A and ϕ sand can be calculated using Eq. 1.4. The smaller the coherence between the two matrices, the fewer samples will be required.

1.2.8 Signal Reconstruction using Convex Optimization

The problem of recovering a sparse signal from undersampled data requires optimization of an under-determined set of linear equations. The most direct way is to minimize the l_0 norm over the solution space bounded by the linear equations. This method is NP hard and the complexity grows by $O(nm)$. Research has shown that if the objective function is replaced by the l_1 norm instead of the l_0 norm, the problem complexity can be reduced to $O(klog(n/k))$, provided that the measurement

matrix satisfies the RIP. The l_1 norm is convex and thus the problem can be solved using fewer computations. If ω is zero, we can formulate the problem as a basis pursuit optimization.

$$minimize\ ||x||_1\ subject\ to\ y = Ax \tag{1.11}$$

We can write $x = \phi z$ as a product of a sparse basis and a vector, to obtain

$$minimize\ ||z||_1\ subject\ to\ y = A\phi z \tag{1.12}$$

If we have bounded noise in the measurements such that $|\omega| < \varepsilon$, we can formulate the problem as a basis pursuit denoising (BPDN) problem.

$$minimize\ ||x||_1\ subject\ to\ ||y - Ax||_2 < \varepsilon \tag{1.13}$$

Or

$$minimize\ ||z||_1\ subject\ to\ ||y - A\phi z||_2 < \varepsilon \tag{1.14}$$

In addition to BPDN, LASSO is also a popular technique for solving this problem. It uses Lagrangian multipliers to make a combination of BP objective function and constraints. The Lagrangian determines the contribution of the BP objective functions to the solution at any point. For $\omega = 0$ it is formally expressed as

$$minimize\left(\frac{1}{2}||y - A\phi z||_2^2 + \lambda||z||_1\right) \tag{1.15}$$

The first term is a loss function and the second function induces sparsity. When closer to a solution, λ is manipulated such that the second term contributes

Algorithm 1.1 Bayesian learning for signal recovery

1: Input: Φ, y
2: Output: w, $\sum$, γ
3: Initialize all $\gamma_i = 0$, $\lambda = 0$
4: **while** convergence criterion not met do
5: Choose a γ_i(or equivalently choose a basis vector ϕ_i)
6: if $q_i^2 - s_i > \lambda$AND $\gamma_i = 0$ then
7: Add γ_i to the model
8: else if $q_i^2 - s_i > \lambda$ AND $\gamma_i > 0$ then
9: Re-estimate γ_i
10: else if $q_i^2 - s_i < \lambda$then
11: Prune from the model (set $\gamma_i = 0$)
12: end if
13: Update $\sum$ and μ
14: Update s_i, q_i
15: Update λ using $\frac{N-1+\frac{\nu}{2}}{\sum_i \gamma_i/2+\nu/2}$
16: Update ν using $log\frac{\nu}{2} + 1 - \psi(\frac{\nu}{2}) + log\lambda - \lambda = 0$
17: **end while**

more to the solution. Other optimization techniques mentioned below are also efficient in recovering the solution based on different objective functions.

1.2.8.1 Matching Pursuit(MP)/Greedy Algorithms

In greedy approaches, instead of an objective function, the algorithm steers to an optimum solution by some greedy rules. These algorithms are computationally less intensive than basis pursuit algorithms and the reconstruction accuracy is also comparable. The complexity of MP increases by $k^2O(n)$ in comparison to $O(n^3)$. [5, 6]

1.2.8.2 Bayesian Learning(BL)

The bayesian learning[7] approach has shown promise in recovering the signal provided some a-priori information is available. This class of algorithms uses the location of zeros in a signal as prior information and uses expectation maximization to optimize the solution.

1.2.8.3 Iterative Hard Thresholding (IHT)

A proxy is added in the first step to the current signal estimate $a_{s+1} = x_s + A^T(y - Ax^s)$, at s^{th} iteration. A predefined threshold is applied to select only the K largest elements of a_{s+1} using a predefined function. This step is optimizing the function $\frac{1}{2}||y - Ax||_2^2$.

$$minimize(\frac{1}{2}||y - Ax||_2^2) \qquad ||x||_0 = K \tag{1.16}$$

Other first order algorithms such as approximate message passing proceed in a similar manner

1.2.8.4 Proximal Point Methods

Proximal point methods are first order algorithms designed to solve problems where the objective function is not continuous over the whole solution space. The formal expression is

$$min\frac{1}{2}||z - (\hat{z} - \frac{1}{L} \bigtriangledown f(\hat{w}))||_2^2 + \frac{\lambda}{L}\Omega(z) \tag{1.17}$$

They have a closed form solution for most choices of regularization function Ω. For compressive sensing, Ω is chosen to be the l_1 norm and it operates as a soft threshold. L is the upper bound on the Lipschitz constant. The computation complexity of proximal point methods is $O(1/k)$ where k is the iteration number.

Objective functions that are convex over the solution space can be used according to requirements. Some are listed below

Algorithm 1.2 Proximal Structural Sparisty [3]

$u_1, z_1 \leftarrow$ arbitrary feasible values
for t=1,2,. . . **do**
    Compute a fixed point $\hat{v}^{(t)}$ by Picard-Opial
    $u_{t+1} \leftarrow z_{t-1} - \frac{1}{L}\nabla E(z_t) - \frac{c}{L}\Omega^\top \hat{v}^{(t)}$
    where $E(z) = \frac{1}{2}||A\phi z - y||_2^2$ and Ω is a regularizer
    $z_{t+1} \leftarrow \pi_{t+1}u_{t+1} - (\pi_{t+1} - 1)u_t$
end for

1.2.8.5 Lp Norms and Log Functions

Most commonly used Lp norms are $l_1 \, and \, l_2$ norms also known as Manhattan distance and euclidean distance respectively. These norms involve relatively fewer computations. If a matrix satisfies the RIP, then the $l_1 \, and \, l_2$ norms are convex in the solution space and converge to a solution in polynomial time. Generally the p-norm is defined as

$$||x||_p = (|x_1|^p + |x_2|^p + ... + |x_n|^p)^{\frac{1}{p}} \tag{1.18}$$

Minimizing the l_0 norm returns the sparest solution in a given solution space. If a matrix follows the RIP then the l_1 norm has proven to be a good approximation to the l_0 norm. The l_0 norm minimization is of combinatorial complexity and this approximation improves the time and computation required to reach a solution drastically. Figure 1.3 shows the combination of the $l_1 and l_2$ balls in a 2-dimensional space.

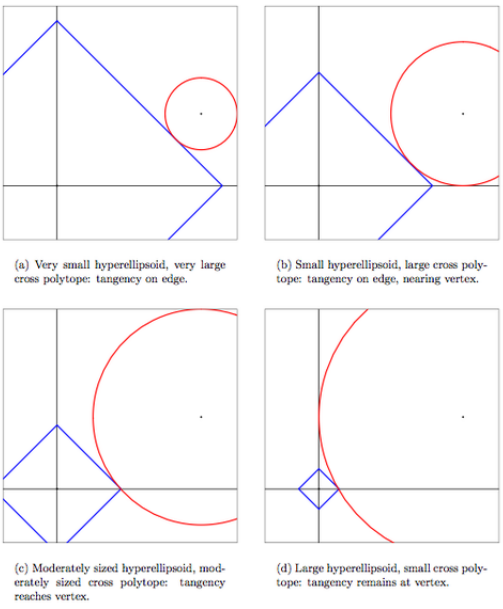


Figure 1.3: Variation of Lagrangian multipler with l_1, l_2 norms in LASSO

The sum of log convex functions is also convex and is widely used for optimizing a solution in polynomial time.

1.2.8.6 TV-Norm

Total variation minimization is useful for denoising corrupted images. It is calculated by the difference of the horizontal and vertical gradients. In mathematical form it is described as

$$x_{TV} = \sum \sqrt{(x_{i,j} - x_{i,j+1})^2 - (x_{i,j} - x_{i+1,j})^2} \tag{1.19}$$

The combined minimization problem can be expressed as

$$minimize \left(x_{TV} + \frac{1}{2\lambda} ||x - g||_2^2 \right) \tag{1.20}$$

It is computationally expensive due to the gradient calculation before calculating the norm but gives an accurate solution.

1.2.9 Noise Suppresion

Detection and quantization noise are significant in any optical system. Detection noise is produced at the detector and varies directly with integration time. Quantization noise arises at the analog to digital convertor due to limited dynamic range and discrete levels. One of the most attractive features of CS is the equal entropy in each sample acquired by using a RIP conforming matrix. In order to minimize quantization noise, it is possible to trade off between saturation and quantization while using compressive projections for measurements.[8] Since each compressive measurement carries the same information, by adjusting the quantization levels to saturate more measurements the quantization error can be greatly reduced on all the unsaturated measurements.

1.2.10 Feature Specific Imaging

One of the emerging techniques in the compressive sensing community is feature specific imaging. There are many applications where the point of interest is just some features of an object. It is a waste of resources to capture the whole scene in order to extract those features. Scientist have come up with measurement matrices that measure only specific features in a scene to train the system. This helps to reduce the computational load at the sampling stage.

1.2.11 Universal Coding Property of CS

CS is a universal encoder for it maps the source signal to codewords and the resulting average code length is bounded. Given an arbitrary source with nonzero entropy, a universal code achieves average code length which is at most a constant times the optimal possible for that source. An asymptotically optimal code is one for which average codeword length approaches entropy. A universal code with constant

=1 is asymptotically optimal.

1.3 Segmentation

1.3.1 Clustering Techniques

In this thesis we used segmentation as a tool to separate background and foreground objects. Segmentation is a key step for implementing this scheme. It separates the background from the objects which helps to track the direction of motion effectively. There are different techniques used for clustering. some are graph based Clustering, K-Means Clustering, Connectivity based Clustering etc.

In this study we used a graph based technique and utilizing normalized cut to measure the goodness of the segmentation.[9] In the graph based technique, a set of points in the feature space are represented as a weighted graph. Nodes in the graph are points in the feature space and an edge is formed between each node. The weight on the edges represent the similarity between each node. A graph can be partitioned by removing the edges between two nodes based on a dissimilarity measure.The dissimilarity is calculated based on

$$cut(A, B) = \sum_{u \epsilon A, v \epsilon B} w(u, v) \tag{1.21}$$

where $w(u, v)$ is the weight associated with the removed edges. The normalized cut for two nodes for a given partition will be

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)} \tag{1.22}$$

here $assoc(A, V) = \sum_{u \epsilon A, t \epsilon V} w(u, t)$ is the sum of weights from node A to all nodes in the space.[9] The effectiveness of the technique varies depending on the contrast and texture in a video frame. The main goal here is to effectively separate the stationary background from objects that move.

1.3.2 Segmentation Quality Measures

Segmentation quality is determined by how effectively the technique separated the background from the foreground. An important parameter is the number of segments, which can be varied depending on the sparsity of a particular video sequence to achieve optimum segmentation. In order to determine the number of segments we maximize the inter-cluster distance and minimize the intra-cluster distances over a predefined range of the number of clusters. The contrast measure between clusters can be maximized to separate a cluster from neighboring clusters.[10] In order to decrease intra-cluster distances, variance can be minimized. We find the number of segments corresponding to the maximum value of contrast over a range of number of segments. This range is bounded below by the number of segments corresponding to the value of mean variance and bounded above by the maximum number of segments.

1.4 Motion Estimation

In order to sense a video efficiently we need to perform both spatial and temporal compression. If the signal is sparse in the spatial domain, random sampling can compress the signal effectively or the signal can be sensed randomly in a different sparse domain. In the time domain, change is sparse in most video sequences. In order to take measurements efficiently in the change domain we need a transformation that detects change. Most transformations require knowledge of past and future frames to calculate the change. This approach can compress the signal after acquisition or by using additional hardware to reach video rates. Another way to detect change is predictive modeling but due to the ephemeral nature of videos it is hard to model these changes with the exception of a few specific video sequences. In this thesis a simple optical flow technique is implemented, which uses the multiplex-

ing properties of spatial light modulators employed in single pixel cameras.[1] The scheme locates the dynamic and static segments in a frame and makes some strategic averages to access the direction in which the segments might be moving. The scheme predicts the direction of motion based on a novel feature of moving segments and allocates measurements in that direction for the next frame. We encircle each segment and note that the difference in averages outside the segment and inside the circle generates peaks at values of the angle θ which predicts the direction in which the segment is moving.[11] It is assumed that the motion is sparse along the time axis and only a few segments move between consecutive frames. A binary filter is formed based on these locations which filters out the static segments and enables measurements to be performed for only the dynamic pixels of moving objects. The binary pattern and the random sampling matrix are multiplied to acquire samples.

Chapter 2

Literature Review

2.1 Static Compressive Sensing

The methods to perform sparse reconstructions from under-determined linear model are older then the origin of compressive sensing. Penalty function using the l_1 norm were used to recover missing information from seismic data. However, it was not used for reconstruction with compressive sensing until the late 90's. The basic idea of CS is to take advantage of sparse information present in the signal in a specific domain. Measuring a signal in a sparse domain requires a transformation that is viable optically. Scientist have come up with a few imaging architectures which acquire measurements compressively in the sparse domain. They have employed spatial light modulators(SLMs) of different kinds to modulate the incoming light from the scene and direct the modulated light onto the focal plane of a single detector. The kinds of SLMs used are digital micromirror devices(DMD), coded apertures and CMOS programmable chips. They hold the measurement patterns and the incoming light is multiplied by these patterns one at a time. The resulting light is focused on a focal plane and summed by a single detector. An A/D convertor samples the signal from the detector after a complete integration period. The architectures designed to acquire compressed measurements can be divided into three categories; sequential, parallel and photon sharing. Most architectures designed to acquire compressive measurements use a single pixel detector to acquire measurements. The sample are acquired sequentially based on Eq. 1.1. This method is not suitable for applications involving fast changing scenes. Figure 2.2 shows one such camera designed at Rice University known as the Single Pixel Camera(SPC). Parallel architectures are expensive as they employ multiple modulators to acquire

measurements simultaneously. They can be implemented using photon sharing architectures which use beam splitters along with modulators to collect measurements.

2.1.1 Coded Aperture CS

This technique is probably the oldest among other spatial modulation techniques. It was employed for solving the problem of imaging light from weak and diffuse sources. By design it increases the SNR without compromising the spectral resolution. When a coded aperture is placed in the path of light, it allows light to pass through multiple holes distributed in space. Samples are taken for different hole patterns and the image is estimated by post processing the recorded measurements. The formation of the coded aperture is important in order to compressively sense the signal. These matrices have to follow some properties. Ideally a compressive measurement is random and involves negative coefficients as well. Due to the implementation constraints of optical systems a random matrix with negative values is not realizable. Some matrices with non zero mean have been able to satisfy the restricted isometry property. One of the popular options are block circulant matrices.[12] The structure of these matrices allows for faster reconstruction than simple random measurements. These techniques have been used for high resolution video imaging and super resolution imaging. Coded apertures do not offer diversity in measurement matrices which makes it a solution to only a few problems. However we can optimally make all measurements needed in one instance with this setting for the reconstruction of a frame by using a detector array.

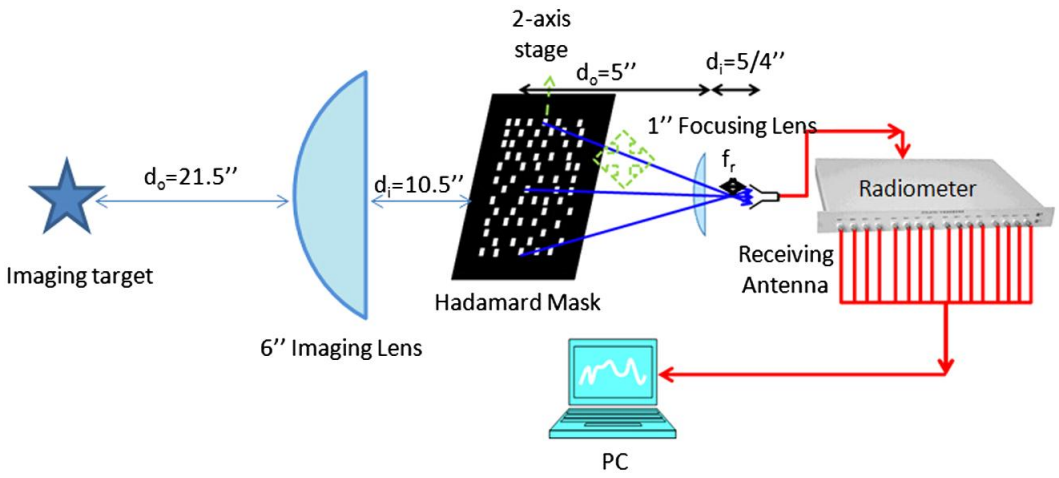


Figure 2.1: Hadamard transform coded-aperture submillimeter wave Imager

2.1.2 CMOS CS Imager

These devices are capable of performing convolution of measurement matrices with a signal in the analog domain. Due to much faster speeds of CMOS technology, these devices are able to compute inner products at a rate faster than the above mentioned techniques. The architecture is also very flexible and can implement all sorts of matrices in which the signal is thought to be sparse. In [13], the authors have used noiselets as a basis for making measurements. CMOS technology is cheap and energy efficient for implementation. Rather than making one measurement at a time they can make all measurement at a given instance by adding the modules accordingly. They require good signal to noise ratio for carrying out the measurement process.

2.1.3 Single Pixel Camera (SPC)

These devices were used for projecting small images onto large screens. They have tiny micromirrors arrays, which can be controlled electrically to change the reflected path of the incident light by one orientation or the other. Typically only two orientations are available. When the light falls on a 2 dimensional DMD array,

it is spatially modulated by the pattern of orientation of the mirrors. Recently, this technology has been used for compressively sensing a scene. The two positions of a micromirror are made to either focus the light on a single detector or disperse the light. Each pattern of the mirrors generates a projection vector. With only two degrees of freedom there is not much flexibility to generate a variety of matrices for different signals according to the sparse basis. Random Bernoulli matrices are used to set the mirror direction. In order to compressively sense the scene these matrices should follow the RIP property.[14] The measurement vectors should be independent of each other. The DMD is able to generate such matrices but when it comes to video imaging they don't have much to offer. A single detector can only measure one projection at a time, this makes it inefficient for video imaging. The DMD's are also a costly part of the optical system which effects its commercial feasibility at large scale.

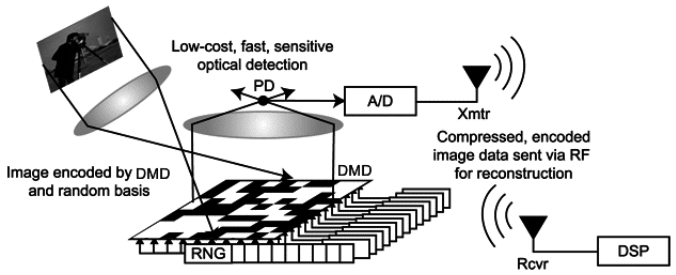


Figure 2.2: Single pixel Imager based on DMD

2.2 Dynamic Compressive Sensing

In this thesis, we discuss and analyse a technique to only acquire measurements corresponding to the dynamic areas in a scene changing over time.

2.2.1 3D-Wavelet CS

Wavelets are efficient at representing a natural signal with few significant coefficients. Many digital compression algorithms employ wavelet based compression for maintaining the entropy. In compressive sensing wavelets are used as a sparse basis in the spatial and temporal domain. Measurements are taken in the wavelet domain and after reconstruction, the actual signal is recovered from the wavelet vectors.

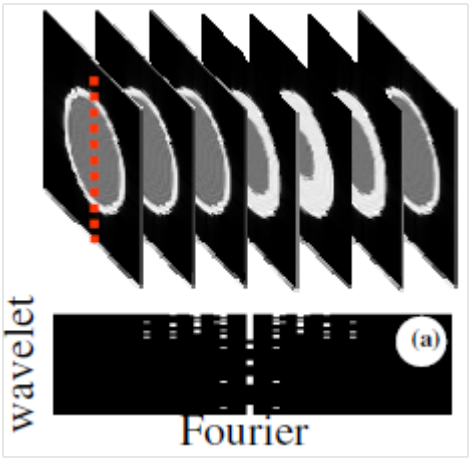


Figure 2.3

2.2.2 Linear Dynamical System(LDS) Based CS

Another approach is based on modeling specific video sequence evolution as a linear dynamical system.[15] It reduces the required samples for reconstruction considerably but this approach is restricted to videos possessing a LDS representation which is possible for only a few specific sequences. These sequences have slow motion content of largely textural nature such as flames, traffic and water. The model fixes some static parameters and the dynamic parameters are compressively measured.

Algorithm 2.1 Kalman Filtered Compressed Sensing

1. Initially support T is estimated by CS reconstruction from the measurements
2. Run Kalman filtering to estimate x at t give x at t-1

$$x_{t|t-1} = \hat{x_{t-1}}$$
$$(P_{t|t-1})_{T,T} = (P_{t-1})_{T,T} + \sigma^2_{sys} I$$
$$K_{t,T} = (P_{t|t-1})_{T,T} A'_T \sum_{ie,t} \sum_{ie,t} = A_T (P_{t|t-1})_{T,T} A'_T + \sigma^2_{obs} I$$
$$\hat{(x_t)_T} = (x_{t|t-1})_{T^c} = \hat{(x_{t-1})_{T^c}}$$
$$(P_t)_{T,T} = [I - K_{t,T} A_T](P_{t|t-1})_{T,T}$$

3. Where P is a posterior error covariance matrix, A is a random measurement matrix, K is optimal Kalman gain and y is measurements vector and
T is the support of x.
4. Compute the filtering error, if its greater then threshold we need to add some support.
5. Addition: Run CS on the observation, values greater than threshold gets added to support vector. In case of values smaller than CS ignore them.
6. Deletion: If some values are smaller then threshold delete them.
Assign T to $T_t \rightarrow x_t \rightarrow z_t$. Go to step two.

2.2.3 Motion Compensated(MC)/Motion Estimation(ME) CS

A lifting based invertible adaptive transform is used for sparsifying the video in the temporal direction and wavelets are used for 2D compression to achieve a 3D transform. This method improves the signal to noise ratio but it is also non-causal and reconstruction is performed after collecting measurements for a number of frames.

2.2.4 Kalman Filtered CS(KFCS)

The static solution to sparse reconstruction is $l_0 and\ l_1$ minimization. In KFCS, a Kalman filter is used for predicting the sparsity patterns in a video sequence. It is assumed that the sparsity patterns change slowly over time.[16] MRI machines measure the Fourier coefficients directly which are sparse and fMRI sequences change slowly over time. A basic algorithm is given below

This approach is real time, fast, and gives better reconstructions compared to other approaches. One assumption made by this technique is that change happen slowly. This does not fit the situation of object motion in natural images.

2.2.5 Block-Based CS(BCS)

Block based CS is an alternate way to solve CS problems. CS reconstructions for large images are complex and time consuming. By breaking the frame into a number of blocks, each block can be processed in parallel, thus making the process faster. BCS can be readily realized in hardware using a DMD. Figure 2.4 shows a matrix used for reconstruction after acquiring the measurements using

$$\Phi = \begin{bmatrix} \Phi_B & 0 & \dots & \dots & \dots & 0 \\ 0 & \Phi_B & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \ddots & \dots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \Phi_B \end{bmatrix}$$

Some works based on block based compressive sensing such as block based compressive sensing with smooth projection Landweber reconstruction(BCS-SPL), divide the frame into non overlapping blocks and process each block separately. The basic technique splits the processing into smaller blocks and combines the reconstruction for the final result [17]. This method does not take into account the temporal redundancies in a video. More advanced techniques based on BCS-SPL take into account motion estimation parameters to aid the reconstruction process. Motion estimation/motion compensation BCS (ME/MS -BCS) selects a group of pictures (GOP) for estimating motion vectors and reconstruct the GOP using this information. This improves the subrate performance but incurs an undesirable time delay in addition to increasing reconstruction complexity [18, 19, 20, 21].

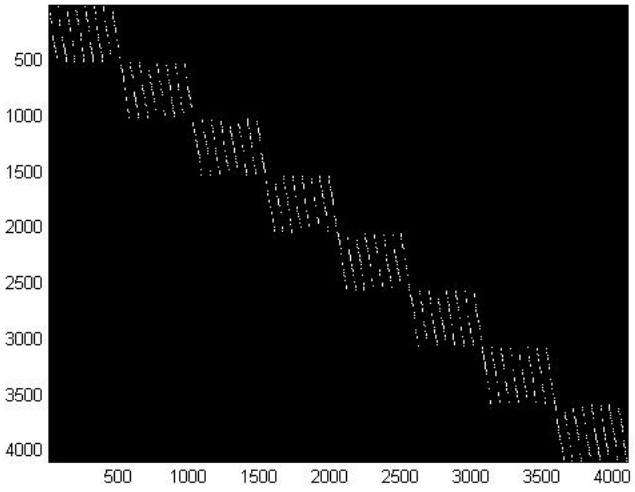


Figure 2.4: Measurement Matrix unwrapped

One variant of the block based approach is adaptive block based compressive sensing in which a frame is divided into a specific number of blocks and each block is assigned measurements based on changes and texture.[22] This approach accumulates residual error and gives block artifacts after recovery as with simple BCS. Moreover it is computationally expensive to optimize measurement allocation before each frame acquisition.

2.2.6 Distributed CS

Distributed CS is based on distributed video coding(DVC). In DVC, the frames are encoded separately at the encoder but processed jointly at the decoder. This is similar to the underlining concept of CS for static imaging. Distributed CS extend this concept to combine CS with DVC for video acquisition.[23] Each frame is individually compressivly sensed at the encoder and reconstructed together with other frames at the decoder. This approach reported improved reconstruction results compared to the direct approach. The number of samples satisfies Eq. 1.10 which can be further improved by designing sampling matrices based on the properties of the

video.

2.2.7 Interframe Difference CS

Considering the CS requirement for sparsity, change is sparse in most video sequences. Frame differencing has been used where the differences between consecutive frames are compressively measured and recovered using sparse reconstruction methods followed by addition to the previous frame.[24] This method not only accumulates residual error but the mean square error increase when the difference is not sparse as in the case of large changes.

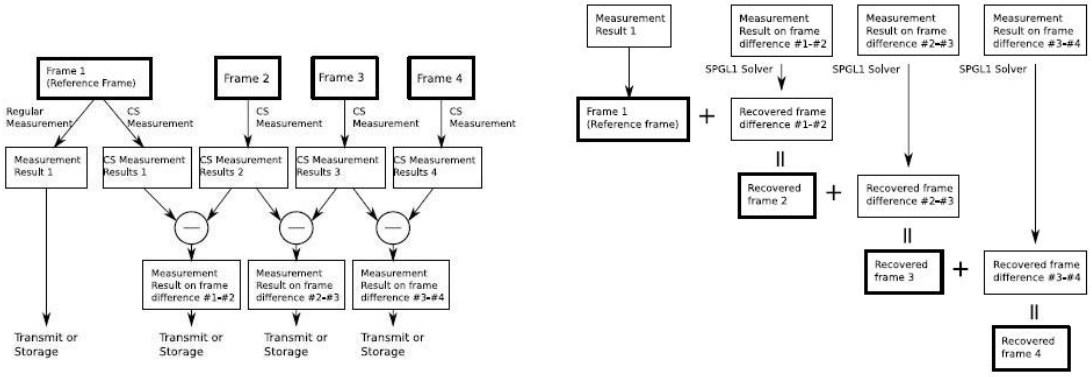


Figure 2.5: The Acquisition and the recovery using Difference Compressive Sensing

Chapter 3

Adaptive Compressive Sampling for Video

3.1 Parameter Estimation

Segmentation is a key step for implementing our proposed scheme. It separates the scene into segments which helps to identify and track object motion effectively. In this study we used normalized cut image segmentation [9]. The effectiveness of the technique varies depending on the contrast and texture in a video frame. The main goal here is to effectively separate portions of the image that may move in subsequent frames from static background portions of the scene. An important parameter is the number of segments produced during segmentation. This can be varied depending on the sparsity of a particular video sequence to achieve optimum segmentation. In order to determine the number of segments we maximize the inter-cluster distance and minimize the intra-cluster distances over a predefined range of number of clusters. The contrast measure between clusters can be maximized to separate a cluster from neighboring clusters [10]. In order to decrease intra-cluster distances, variance can be minimized. We find the number of segments corresponding to a maximum value of contrast by searching over a range of the number of segments. This range is bounded by the number of segments corresponding to the value of mean variance and a specified maximum number of segments. Variance for l clusters can be calculated as

$$\sigma = \frac{1}{l}\sum_{i=1}^{l}\sigma_i = \frac{1}{l}\sum_{i=1}^{l}\frac{1}{m_i}\sum_{k=1}^{m_i}|x_k - \mu_i| \tag{3.1}$$

where σ_i is the variance of pixels within the segment i and μ_i is the mean of ith segment, l is the total number of segments and m_i corresponds to the number of pixels within the segment i. Mean contrast for l clusters can be computed using the equation below,

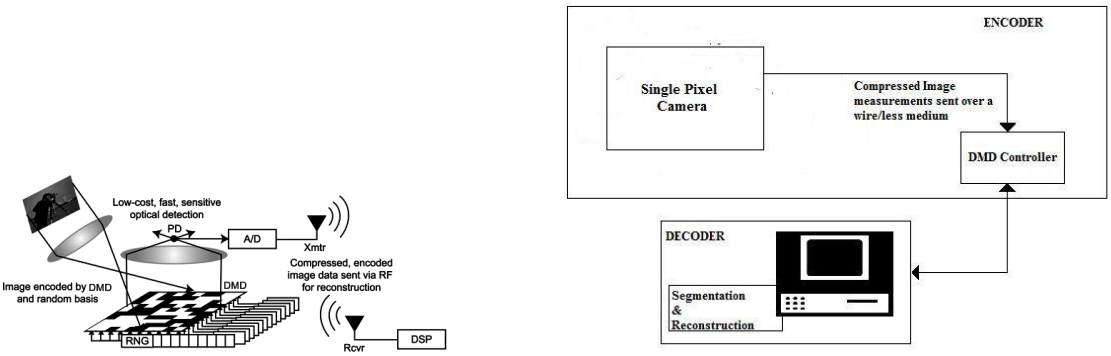
$$Ct = \frac{1}{l}\sum_{i=1}^{l} Ct_i = \frac{1}{l(l-1)}\sum_{i=1}^{l}\sum_{j=1}^{l}\sqrt{\frac{(\mu_i-\mu_j)^2+(\sigma_i-\sigma_j)^2}{(\mu_i+\mu_j)^2+(\sigma_i+\sigma_j)^2}} \qquad j \neq i \qquad (3.2)$$

where Ct is the mean contrast of all segments. The variation of the number of segments against maximum contrast is used to determine the number of segments used for all further processing.

We next classify segments as static or dynamic. The static and dynamic areas are found by looking at the temporal differences in the average pixel value over a segment. In the context of a single pixel DMD based sensor, averages over arbitrary shaped regions are easily calculated by simply directing the light associated with those pixels to the single detector and dividing the measured value by the number of pixels in the segment. These averages are compared with a threshold value. We assume only a few segments change significantly from one frame to the next. Therefore the histogram of differences between the segment averages forms a unimodal distribution with a mode at the first bin. We estimate the change detection threshold based on unimodal distribution to select segments that have significantly changed between consecutive frames. A straight line joining peak and last bin is drawn as shown in Figure 3.4a. The value corresponding to the point on the histogram with maximum divergence from the straight line gives an estimate for threshold [25].The area above threshold is classified as dynamic and below is taken as static. In the event that more segments change significantly, the histogram is inverted to calculate the threshold. All segments are selected in the case where no population is considerably larger than the other. The practical implementation of this procedure is described in section 3.

3.1.1 Motion Estimation

In order to sense a video efficiently we need to perform both spatial and temporal compression. Random sampling can compress the signal effectively if applied in the domain where the signal is sparse. In the time domain, changes are sparse in many video sequences. In order to take measurements efficiently in the change domain we need a transformation that detects change. Most transformations require knowledge of past and future frames to calculate the change. This approach can compress the signal **after** acquisition or by using additional hardware to reach video rates. Another way to detect change is predictive modeling but due to the ephemeral nature of videos it is hard to model these changes with the exception of a few specific video sequences. In this paper a simple optical flow technique is implemented, which uses the multiplexing properties of DMDs employed in single pixel cameras[1]. The scheme locates the dynamic and static segments in a frame and takes some strategic averages to access the direction in which the segments might be moving. The scheme predicts the direction of motion based on a novel feature of moving segments and allocates measurements in that direction for the next frame. The process begins by encircling each segment by a circle with a radius larger than the maximum size of the segment. The circle is divided into sectors. The difference in averages outside the segment and inside the circle generates peaks at values of the angle θ which predicts the direction in which the segment is moving[11]. The possible directions are quantized by the number of sectors used to divide the circle (eight in the current research). It is assumed that motion is sparse along the time axis and only a few segments move between consecutive frames. A binary filter is formed based on these locations which filters out the static segments and enables measurements to be performed for only the dynamic pixels of moving objects. The binary pattern and the random sampling matrix are multiplied to acquire samples.



(a) (b)

Figure 3.1: a. Single Pixel Camera [1] b. Adaptive compressive sensing imager block diagram

3.1.2 Hardware implementation

The proposed method can be efficiently realized in hardware. The basic operations to implement this scheme can be computed rapidly. To acquire measurements for dynamic regions, the mirrors of the DMD are turned off where the mask is zero and a random pattern is projected on the rest of the mirrors. The sampling requirements for the discussed algorithm depend upon the dynamic region in a video sequence and varies directly with the number of pixels that are changing. Our proposed technique adapts the sampling to these changing regions. In order to achieve real time video streaming, the sampling process on the encoder side would need to be fast enough to acquire dynamic measurements in less than 33 milliseconds. On the decoder end, the computations would need to be fast enough to optimize the dynamic area in less than 33 milliseconds. Under the assumption of slow change, the segmentation process can be performed in parallel and the newest segmentation can be used to form sampling masks instead of requiring the latest frame to be reconstructed. This saves significantly on reconstruction time. The discussed scheme for measurement allocation is capable of acquiring a number of frames without consuming as many resources for many types of video as compared to existing techniques.

The bandwidth gained by reducing samples required for a frame reconstruction can be used towards increasing frame rate or decreasing bandwidth.

3.2 Methodology

The setup described by Takhar [1] is used as an example for practical implementation (see Figure 3.1). We propose to acquire all the measurements for the reconstruction of the first frame by projecting different random patterns on the entire DMD. For subsequent frames, only temporally changing regions of the image are measured. To calculate magnitude and direction of change we adopt the scheme shown in the flow chart of Figure 3.2.

We segment the first frame after reconstruction into a number of segments. The segmentation algorithm used is based on normalized cut criterion which measures both dissimilarity between the groups and similarity within a group. It maximizes the normalized cut criterion for a given number of clusters. In order to estimate change due to each moving object between consecutive frames we try to separate each object from the background and estimate the minimum number of groups that can separate objects present in an image.

In order to optimize the number of segments we maximize the contrast between the segments and minimize the variance within segments in a frame. We start with a minimum of five segments and calculate variance and contrast between each frame. The number of segments is incremented in each iteration until a maximum contrast limit is reached. The contrast is estimated over a range of variance above the mean value as shown in Fig. 3.3. The number of segments corresponding to the maximum contrast is used for all subsequent calculations. The segment can be expressed formally as

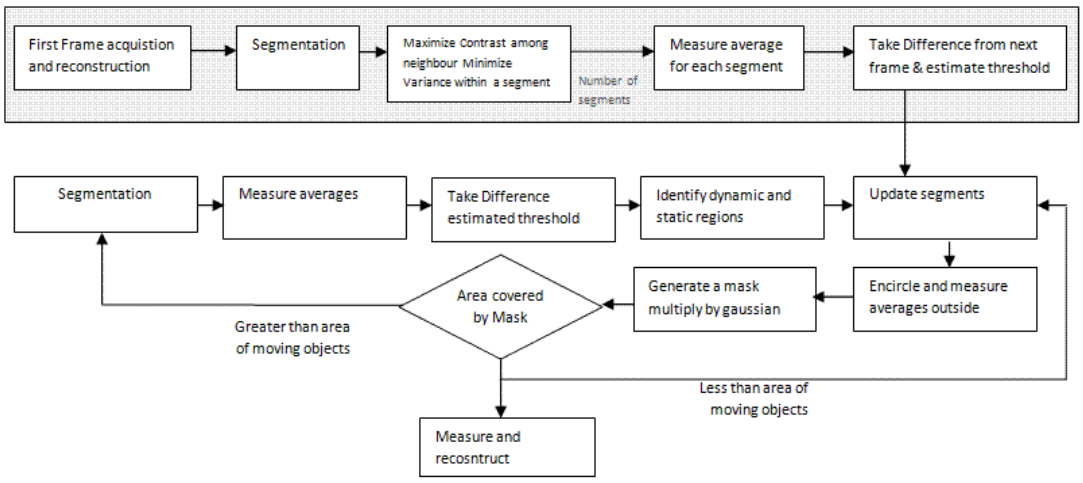


Figure 3.2: Flow Chart

$$y_p(x,y) = \left\{ \begin{array}{ll} 1 & x,y \epsilon p^{th}\ segment \\ 0 & otherwise \end{array} \right\} \tag{3.3}$$

The segmentation mask for each segment is used to drive the DMD of the single pixel camera to route light from each segment to the detector one by one. The output of the detector is the average of each segment.

Next we calculate the temporal differences in the averages over each segment to see which segment has changed significantly. We assume that change is sparse between two consecutive frames and only a few segments change significantly relative to the rest. In view of this assumption, unimodal thresholding can be used to estimate the level of significance for change detection [25]. We form the histogram of segment average differences and the threshold is the point of maximum divergence on the curve from the straight line joining the peak and the bin before the last empty bin as shown in Figure 3.4. The number of bins is kept equal to the number of segments. Increasing the number of bins does not affect the detection significantly. The probability to detect change accurately depends on the quality of segmentation

Algorithm 3.1 Adaptive Compressive Sensing for Video Acquisition Using SPC

1- Acquire the number of measurements for first frame according to Eq. 3.1

2-Divide the frame into a number of segments that maximize contrast and minimize variance within the segments and

calculate the vector $SegAvg_{y_p}(t) = \frac{1}{s_p} \sum_{i,j \in y_p} x_{i,j}$where y_p is set of pixel locations in a segment, p = 1,2,...,F, s_p is

number of pixels in y_p, and t is the frame number.

3- Measure $SegAvg_{y_p}(t+1) = \frac{1}{s_f} \sum_{i,j \in y_p} x_{i,j}$ and select p for which $|SegAvg_{y_p}(t) - SegAvg_{y_p}(t+1)| > \alpha$, where α is

the change threshold determined by unimodal thresholding.

4- Draw a circle $circ(cen_{y_p}, rad_p)$ where rad_p is greater than the distance between the center and the farthest pixel in

the segment.

5- Define sectors θ from $(k-1)\,\pi/4 \rightarrow k\pi/4$ where k = 1,2,3,...8. Measure each $PartAvg_{\theta_k}(y_p, t) = \frac{1}{s_p} \sum_{i,j>y_p}^{i,j<rad_{ymax_p}} x_{i,j}$

and find all ranges of θ for which $|PartAvg_{\theta_k}(y_p, t) - PartAvg_{\theta_k}(y_p, t+1)| > \beta$ where β is a fixed threshold and

update the motion vector for each y_p.

6- Update the segments location based on calculated magnitude and direction of significant motion vectors and form

a mask covering the dynamic area.

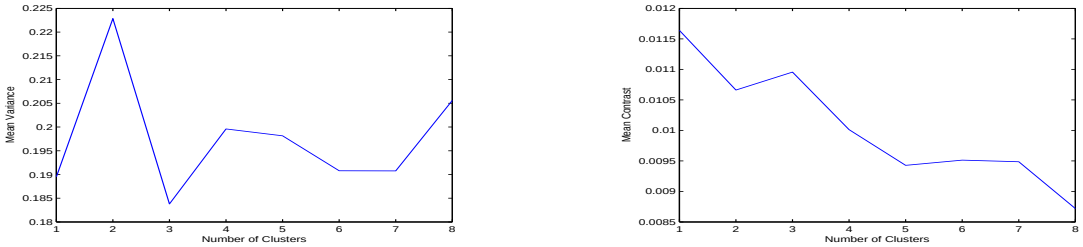
7- Calculate number of samples required for reconstruction using Eq. 3.1.

8- Form a measurement matrix as $A = Mask \times rand(m, n)$ and use the measurements for reconstruction.

9- Go to step 3 if area under mask is less than a predefined dynamic area. Start from Step 2 if area is greater than

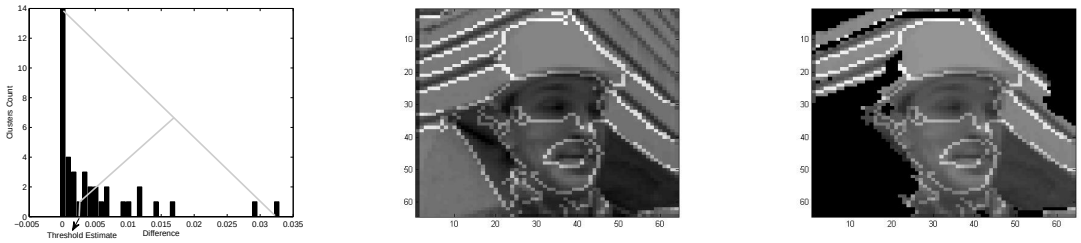
the dynamic area using same number of clusters estimated using first frame and same threshold value in step 3.

and the threshold estimation technique. In this approach, if the difference histogram is not unimodal, all segments are selected for further processing. We noticed in our simulations that increasing the number of segments makes the change detection more precise. Considering that the number of segments should be moderate in order for this scheme to be competitive with other methods we kept an upper bound of 40 segments with a negligible effect on the performance.



(a) (b)

Figure 3.3: a. Mean contrast vs number of clusters, b. Mean variance vs number of clusters



(a) (b) (c)

Figure 3.4: a. Threshold estimation, b. Segmented frame c. Selected segments with differences above threshold

The segments with changed average are selected and encircled with a radius exceeding the distance from the centroid to the farthest pixel in a segment by a fixed amount. For this work, this value was set to 4 pixels. Setting the radius beyond the farthest pixel defines the area within which a segment can move. In order to calculate the magnitude and direction of motion, we partition the circle into 8 equal sectors covering $0 \rightarrow 2\pi$ and representing eight degrees of freedom a segment can move. For each segment, the space outside the segment boundary and inside the circle is determined. This is projected by the DMD onto the detector to calculate the average in this boundary area. This average is compared with the previous frame average of this same area. We restrict the estimate of the direction of motion to the 8 central angles θ_k, $k = 1...8$ of the 8 sectors. We find all directions θ_k for which the

difference of the boundary averages *exceeds a predefined threshold. Mathematically, the average can be written as*

$$PartAvg_k(y_p, t) = \frac{1}{s_p} \sum h_k \cdot f_t \qquad for \qquad k = 1...8 \tag{3.4}$$

where f_t is the frame at time t and h_k is a mask defined by

$$h_k = (c_p \cup y_p) \cap s_k$$

Here c_p is the circle around the centroid of segment y_p .

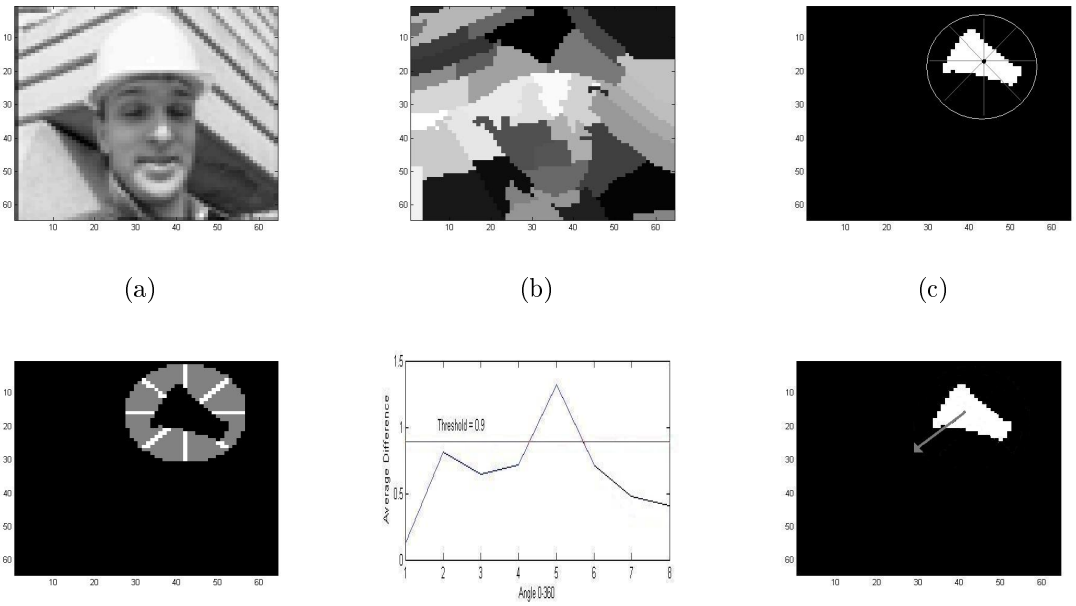
$$c_p(x, y) = \left\{ \begin{array}{cc} 1 & \sqrt{x^2 + y^2} < rad_p \\ 0 & otehrwise \end{array} \right\} \tag{3.5}$$

and s_k is the kth sector centered at rad_p.

$$s_k(x, y) = \left\{ \begin{array}{cc} 1 & \frac{\pi(k-1)}{4} < \theta_k < \frac{\pi k}{4} \\ 0 & otherwise \end{array} \right\} \tag{3.6}$$

We find all θ_k for which

$$|PartAvg_k(y_p, t) - PartAvg_k(y_p, t + 1)| > \beta \tag{3.7}$$



(a) (b) (c)

(d) (e) (f)

Figure 3.5: a. Foreman frame b. Segmented foreman frame c. One of the changed segment partitioned into eight parts d. Area outside the segment partitioned into eight parts e. $PartAvg_{\theta_k}(y_f, t) - PartAvg_{\theta_k}(y_f, t+1)$ for same x f. Direction of movement

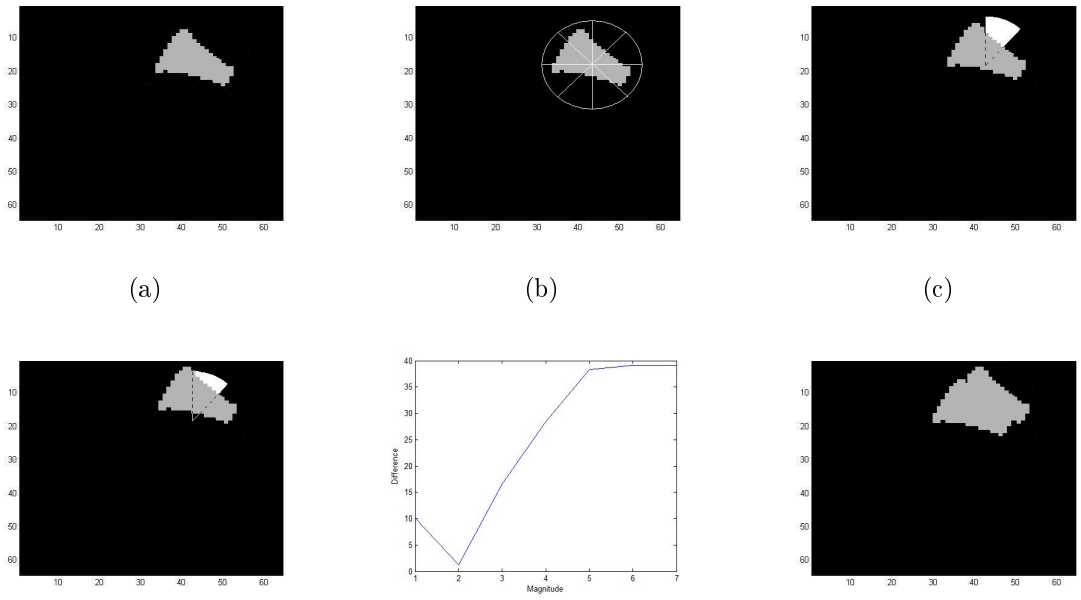


Figure 3.6: a. A dynamic segment of foreman video b. Initial segment position, $PartAvg_{\theta=45^o}(y_f, t)$ c. Segment position incremented in single direction and $PartAvg_{\theta=45^o}(y_f, t)$ for x_{i_1,j_1} d. Another segment position $PartAvg_{\theta=45^o}(y_f, t)$ for x_{i_2,j_2} e. $PartAvg_{\theta_k}(y_f, t) - PartAvg_{\theta_k}(y_f, t+1)$ for all θ f. Final dynamic area for measurements

In order to calculate the magnitude of change we move the segment in the estimated direction θ_k for all possible x and y inside the circle and measure the average outside the segment for each coordinate pair. For pixel locations y_p

$$y_p(x_i, y_i) = y_p(x + r_i cos\theta_k, y + r_i sin\theta_k)$$

where r_i is such that the distance of the farthest pixel in segment y_p remains less than the radius of the circle i.e. $d(y_p, cen_p) < rad_p$. Using Eq. 3.4 we calculate all averages outside the segment boundary, maximizing the difference between the

average calculated and the previous frame average over all pairs of coordinates. We then calculate the magnitude of the displacement m_{θ_k}in the θ_k direction.

$$m_{\theta_k} = max|PartAvg_{\theta_k}(y_p(x,y),t+1) - PartAvg_{\theta_k}(y_p(x_i,y_i),t)| \quad \forall\, i \tag{3.8}$$

The new coordinates for y_p in direction θ_k will be defined as

$$x' = x + m_{\theta_k} cos\theta_k$$

$$y' = y + m_{\theta_k} y sin\theta_k$$

We calculate the updated segment dynamic area by combining the translated segment in all significant θ_k directions. The updated segment dynamic area is found as

$$y_p(x,y) = y_p(x,y) \cup y_p(x+m_{\theta_1} cos\theta_1, y+m_{\theta_1} sin\theta_1) \cup ... \cup y_p(x+m_{\theta_k} cos\theta_k, y+m_{\theta_k} sin\theta_k) \tag{3.9}$$

A mask for all dynamic segment areas can be formally expressed as

$$M_t = y_1 \cup y_2 \cup ..... \cup y_p \tag{3.10}$$

The measurement matrix then can be written as a scalar product of a Gaussian rand matrix and the mask.

$$A = rand(m,n). \times M_t \tag{3.11}$$

A binary mask is created using the location information of the dynamic segments. By rewriting Eq. 3.1, the number of measurements M3 are assigned based on

$$M3 = log(n)/[\varepsilon/(C.S)]^{2p/(p-2)} \tag{3.12}$$

where n is sum of ones in the mask, S is the sparsity index of the last segmented frame dynamic area inside the mask, ε is the reconstruction error bound, C is a constant value dependent on the correlation between the sampling and basis matrix, and p is taken as 2/3. The number of measurements k is bounded below by $0.3n$ and bounded above by $0.6n$.

In order to form a measurement matrix we need to transmit the information about segments corresponding to each pixel. The location of each segment is shared with the encoder after resegmentation is performed at the decoder end. Therefore, the bandwidth required per frame depends on the resegmentation interval and estimated number of segments required using contrast and variance information from Eq. 3.1 and Eq. 3.2. We calculate the measurements required to transmit the information using the relationship shown below.

$$M1 = \frac{Total\ Number\ of\ pixels}{(bit\ depth - bits\ req\ to\ represent\ segments)Resegmentaion\ interval} \tag{3.13}$$

The number of measurements required for motion estimation is calculated using the following equation.

$$M2 = direction\ resolution \times Avg.\ dynamic\ segments/frame + No.\ of\ segments \tag{3.14}$$

These measurements are used at the encoder to generate a new mask and transmitted from encoder to decoder for updating the mask at the decoder for reconstruction. The total number of measurements can be expressed as

$$M = M1 + M2 + M3$$

After acquisition of the measurements from the dynamic area we use the total variation minimization algorithm for estimation in a manner similar to equation 1.11.

$$minimize\ ||x||_{TV}\ subject\ to\ y = Ax \tag{3.15}$$

All the dynamic pixels are replaced by new estimates and static pixel values are taken from the previous frame.

$$f_{t+1} = M_t^c \cdot f_t + M_t \cdot x$$

where M_t^c is the complement of M_t.

When the area under the mask is less than a predefined percentage of the whole frame κ, averages for each segment are measured and any segment other than the previously selected segment with changed average above threshold is included in the mask. Above threshold segments are processed further for motion detection and mask creation for the next frame. Re-segmentation is performed when the κ value jumps above a predefined percentage.

We have found this scheme efficient for surveillance videos with complex background and slow changes. Masking the static background using segmentation reduces the number of pixels to be estimated with greater precision thereby increasing the performance of the reconstruction algorithm.

3.3 Simulations and Analysis

In the experimental studies we first show the variation of subrate and PSNR for the proposed technique. Simulated and real videos were obtained for this study. Each video was created or downsampled to a size of 64x64 pixels per frame. The machine used for simulation has a 2.4 GHz processor and 4 GB RAM. The simulated videos are of a human shaped object moving linearly across a uniform background

at different speeds. The real videos are taken with a thermal infrared camera and show animals moving at different speeds under control of human handlers. These videos are representative of the types of scenes expected to be encountered in a practical implementation of our algorithm in a sensor. The quantization is assumed to be 16-bits for calculation of M1 and M2. The subrate is controlled by varying the multiplicative constant C_m from 0.1 to 0.8 in the following expression $M3 = C_m \times n$, where n is sum of ones in the mask. The κ threshold was set at 0.1 for simulated video and 0.5 for real video sequence. Here values of κ are chosen according to the changes in video. This also show the effect of κ on the number of motion estimation (ME) and mask measurements. All measurements are averaged over 30 frames. The results are shown in Table 3.1 for a simulated video and real video and plotted in Fig 3.7. The ME measurements for simulated video are less due to less complexity of scene. Mask measurements are high due to a smaller percentage of area threshold. In real video the mask measurements are less due to higher percentage of area threshold and ME measurements are high due to a greater number of segments selected to separate the foreground and background.

Table 3.1: The break up of subrate into measurement for mask transmission (M1), ME (M2), and scene measurements (M3) for a. Simulated video sequence b. Real video sequence

(a)

Sr.no.	M1	M2	M3	M	subrate	PSNR
1	82	10	149	241	0.05	27.8
2	82	10	176	268	0.06	28.9
3	82	10	201	293	0.07	35.5
4	82	10	247	339	0.08	42.7
5	82	10	327	419	0.102	44.5
6	82	10	340	432	0.105	49
7	82	10	355	447	0.109	50
8	82	10	399	491	0.119	52.5

(b)

Sr.no.	M1	M2	M3	M	subrate	PSNR
1	63	60	162	285	0.069	27.2
2	63	60	272	395	0.096	29.5
3	63	60	331	454	0.110	32.7
4	63	60	366	489	0.119	35.2
5	63	60	571	694	0.169	37.2
6	63	60	574	697	0.169	38.4
7	63	60	768	891	0.217	39.9
8	63	60	827	950	0.231	40

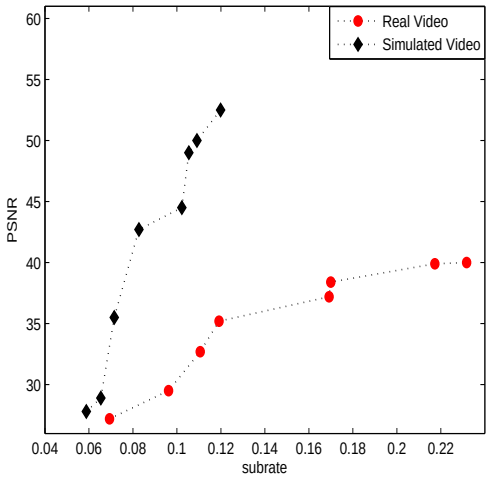


Figure 3.7: PSNR to subrate curve for simulated video sequence and real video sequence

3.3.1 Comparison study with existing techniques

In order to demonstrate the performance of the proposed technique, we performed a simulation using simulated and real video sequences and recorded the peak signal to noise ratio (PSNR) and the subrate used for each frame. In simulated videos, temporal changes are varied for eight video sequences over a minimum to a maximum range of temporal changes. The texture in each frame is kept minimum in order to minimize the number of segments required for separating foreground and background. A random 2D signal with 10^{-4} variance is added to each frame as well to simulate small changes. Change is calculated based on the expression below.

$$\Delta = MSE\left(f(t-1), f(t)\right) \tag{3.16}$$

The subrate and PSNR is recorded for each reconstructed video sequence and compared with intraframe TV, frame differencing, and BCS-SPL-CT methods. Real videos were recorded using a long wave infrared camera in a fixed position in a natural environment. The video is of an animal passing through the field of view at different speeds controlled by a human.

The proposed method reduces the computational complexity of the reconstruction algorithm and produces a frame in less time than the other methods when the change is below a threshold. It adapts the subrate according to the changes in a video. An average subrate over 30 frames for a particular video using our proposed method is used for reconstruction using the other methods. The time requirements and PSNR for intraframe TV and BCS-SPL-CT are irrespective of the changes in the video but depend on the subrate. As mentioned before the subrate of our adaptive method is passed to the other methods for reconstruction which changes the PSNR and time accordingly. Therefore we have used a ratio of PSNR to seconds per frame in order to demonstrate the performance comparison. As shown in Fig. 3.8

Table 3.2: Simulated Video sequences Comparisons with three other methods

	Δ	M1	M2	M3	M	subrate	Adaptive CS			Intraframe CS			Frame Differencing			Block CS		
							PSNR	t sec	ratio	PSNR	t sec	ratio	PSNR	t sec	ratio	PSNR	t sec	ratio
1	1.15	36	38	199	273	0.066	37.7	0.95	39.5	35.8	1.83	19.5	45.7	2.9	15.4	14.1	5.2	2.67
2	1.2	36	38	205	279	0.067	38.4	0.96	39.7	36.1	1.9	19	45.6	2.83	16.04	14.19	5	2.83
3	1.25	36	38	215	288	0.073	38.5	1.03	37.2	36.5	1.93	18.8	45.5	2.9	15.6	15.8	5.3	2.98
4	1.27	36	50	220	306	0.074	38.7	1.1	35.1	36.9	1.93	19.0	45.7	2.96	15.4	15.7	5	3.14
5	1.28	36	50	221	307	0.074	38.5	1.13	33.9	37.1	1.96	18.8	45.7	3	15.2	15.78	5.5	2.85
6	1.33	36	50	225	311	0.075	38.8	1.18	32.8	37.5	1.96	19.0	45.6	3.2	14.25	15.7	5.13	3.05
7	1.44	36	50	230	316	0.077	39.5	1.2	32.7	37.6	1.96	19.11	45.3	3.3	13.45	15.12	5.3	2.85
8	1.48	36	50	235	321	0.078	39	1.23	31.6	37.8	2.03	18.5	45.2	3.4	13.29	15.6	5.03	3.09

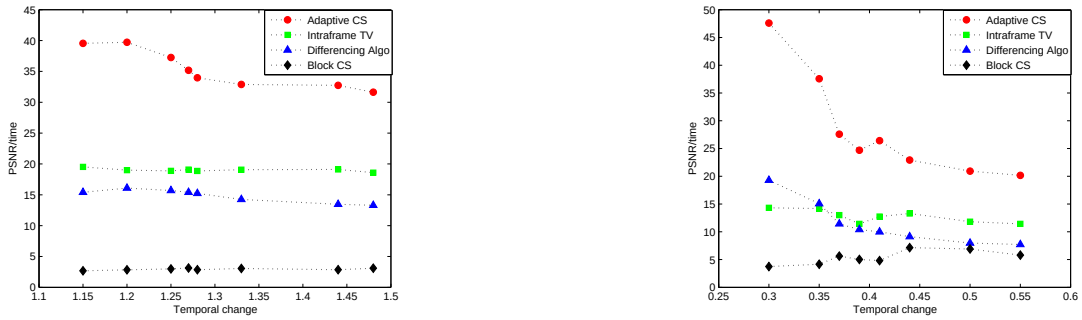
the ratio is maximum for least change for our proposed method and the differencing algorithm and drops as the change increases.

The PSNR is also compared to the existing methods and it is noted that results using the proposed method holds the PSNR value in the 2db range while other algorithms PSNR drops as the temporal changes are decreased. The reason for drop is non adaptive nature of other methods used for comparison. The proposed method adapts to less changes while the other algorithms, with the exception of differencing, reconstruct irrespective of changes in a scene. The change in subrate required by our proposed technique is not pronounced in the simulated video as compared to real videos. This affects the performance of the other algorithms curves in both cases. The differencing algorithm shows less steep downward trend compared to our proposed algorithm.

Fig. 3.9 shows five frames from four original and reconstructed simulated videos with Δ increasing from top down, following the increased speed of the object. Fig. 3.10 shows five frames from four original and reconstructed real videos, recorded at a trail using a long wave infrared camera. The κ threshold was kept at 0.3 and 0.5 for simulated and real video respectively for reconstruction. The parameter C was empirically chosen to be 1.5 and ε was taken to be 0.1. The parameter

Table 3.3: Real Video sequences Comparisons with three other methods

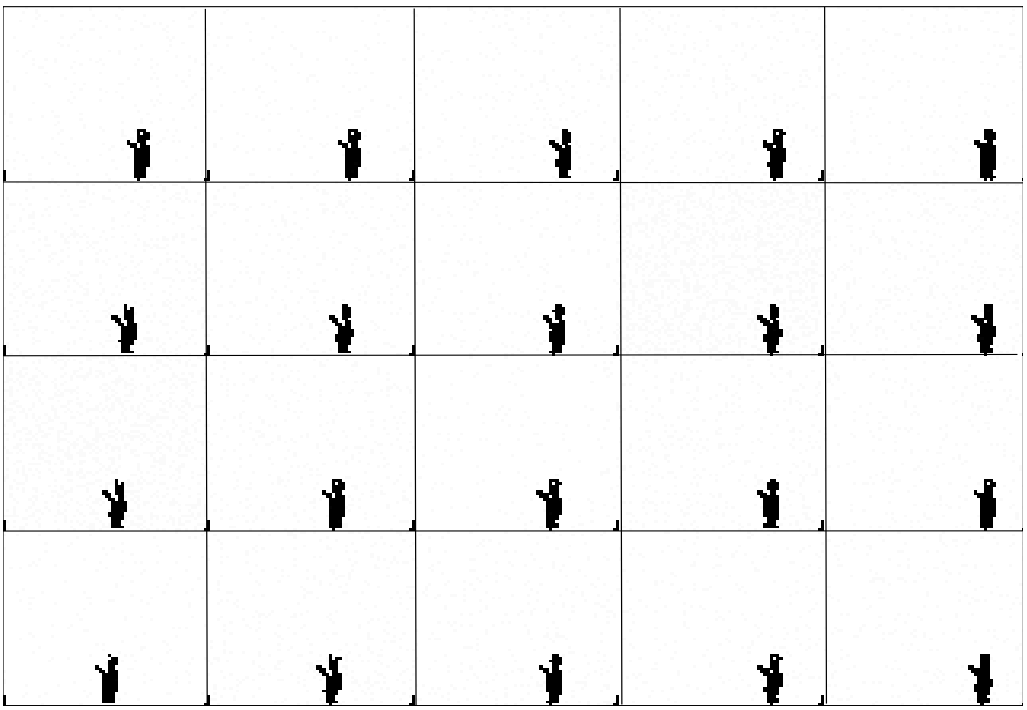
	Δ	M1	M2	M3	M	sub rate	Adaptive CS			Intraframe TV			Frame Differencing			Block CS		
							PSNR	t sec	ratio	PSNR	t sec	ratio	PSNR	t sec	ratio	PSNR	t sec	ratio
1	0.3	49	50	179	278	0.06	34.9	0.73	47.5	23.4	1.63	14.3	46.3	2.4	19.2	17.3	4.6	3.7
2	0.35	49	50	221	320	0.07	33.8	0.9	37.5	24.6	1.73	14.19	45.7	3.03	15.0	16.9	4.06	4.15
3	0.37	49	50	324	423	0.10	33.1	1.2	27.5	26	2	13	45.7	4	11.4	21.2	3.7	5.6
4	0.39	49	63	346	458	0.113	32.1	1.3	24.6	26.3	2.3	11.4	44.8	4.3	10.4	19.1	3.8	5.02
5	0.41	49	63	361	473	0.115	36.1	1.36	26.4	25.5	2	12.7	44.3	4.43	9.99	18.3	3.8	4.8
6	0.44	49	63	376	488	0.118	32.1	1.4	22.9	27.5	2.06	13.3	45.02	4.93	9.1	22.4	3.1	7.14
7	0.5	49	63	373	485	0.118	31.4	1.5	20.9	26.8	2.2	11.8	40.7	5.1	7.9	21.2	3.0	6.91
8	0.55	49	63	408	520	0.12	33.6	1.6	20.16	26.3	2.3	11.4	41.7	5.4	7.7	19.7	3.4	5.79



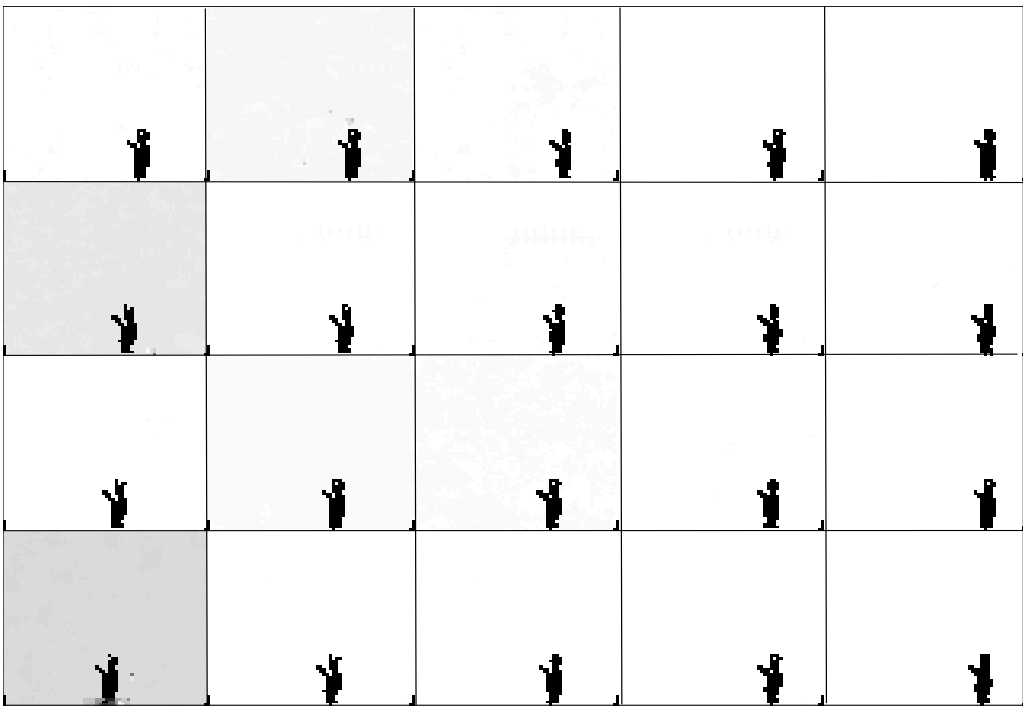
(a) (b)

Figure 3.8: PSNR/time verses temporal change curves for a. Simulated video sequence b. Real video sequence

S was calculated based on a threshold taken as 0.13 in Eq. 3.12. The first real surveillance video was reconstructed with about 6% of the measurements necessary for each frame on average compared to traditional raster scanning. The number of samples falls to only 1% for a few frames where most of the dynamic area over the whole frame is below threshold. These measurements are used to reconstruct only the dynamic area which is on average about 25% per frame for this video. This takes the computational load from the optimization algorithm hence improving the time required for reconstruction. This number can be further improved if we assign some parameters such as a threshold and number of clusters individually to each

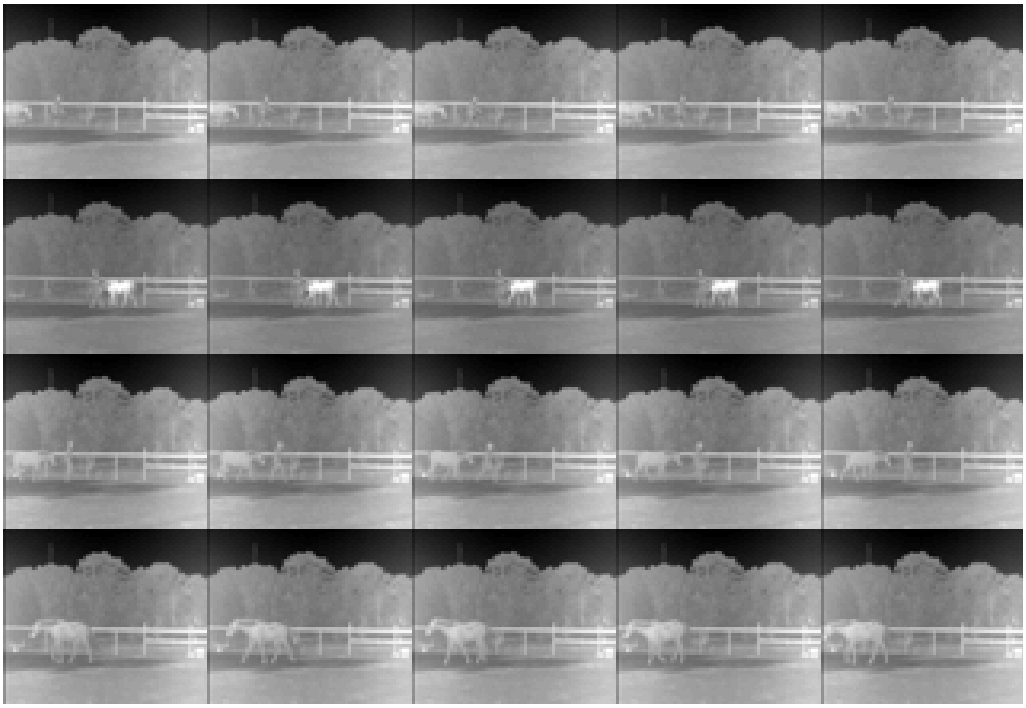


(a)

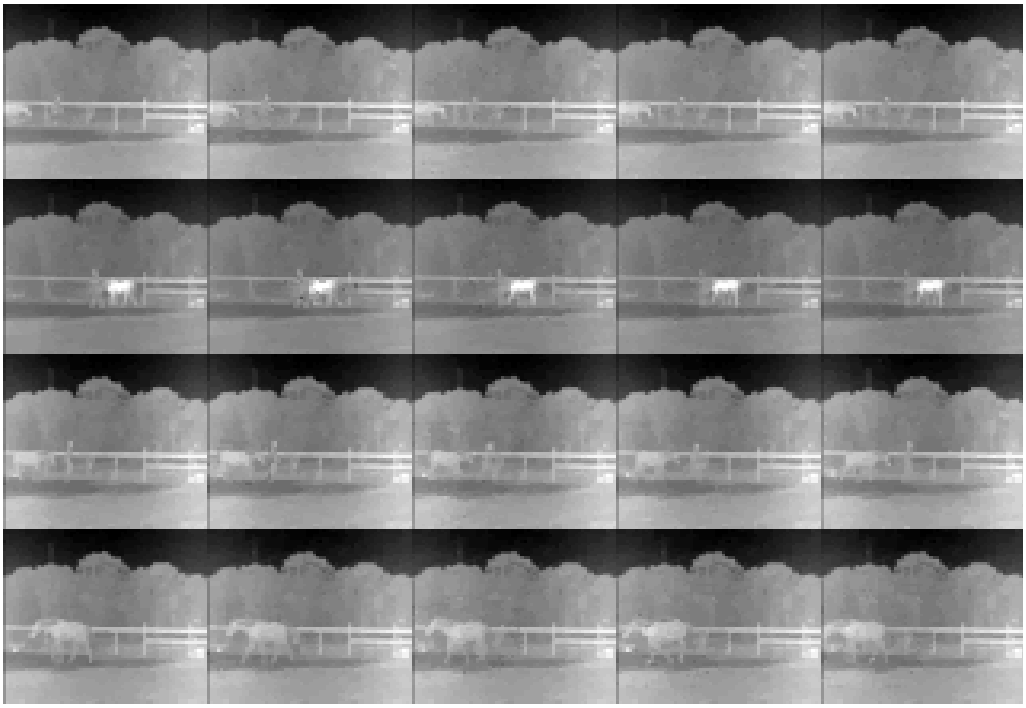


(b)

Figure 3.9: a. Simulated motion of person across a frame at different speeds increasing from top to bottom b. Reconstructed frames of simulated motion in 9a



(a)



(b)

Figure 3.10: a. Real videos of animals walking across a frame with increasing MSE top to bottom b. Reconstructed videos frames in 10a

sequence based on the characteristics of the sequence. Basically the scheme tracks an object once it is detected in motion. Prediction of the direction and magnitude of motion enables us to assign measurements strategically and improve reconstruction efficiency. Due to the shape and texture of segments, some static areas are picked up and some dynamic areas fall below threshold. The algorithm checks for change before collecting measurements and incorporates the new dynamic areas in the next frame so there is minimal residual error accumulation. The residual error may accumulate in areas classified as static. In all test cases in this study, the errors were removed within a few frames.

Table 3.4: Foreman Video sequence comparisons with three other methods

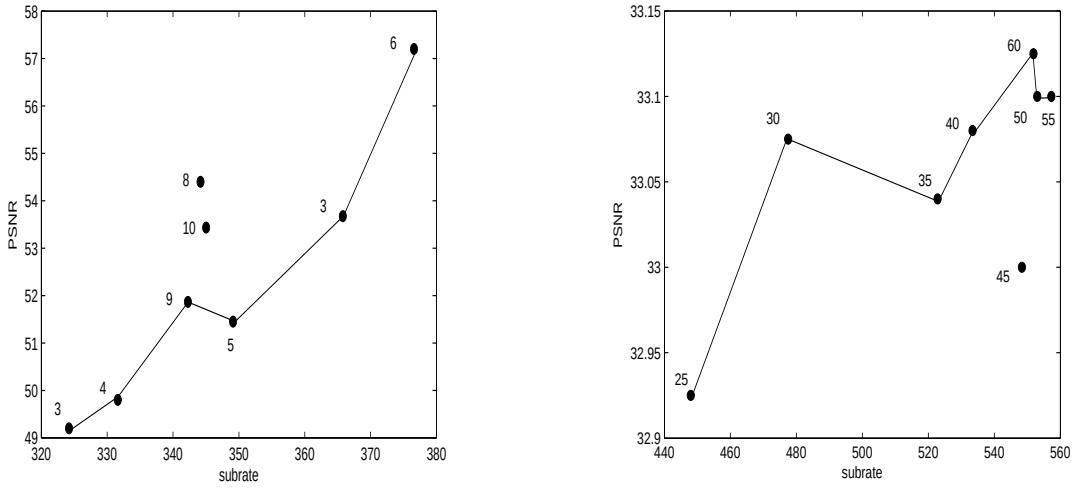
video	Δ	M1	M2	M3	M	subrate	Adaptive CS			Intraframe CS			Frame Differencing			Block CS		
							PSNR	t sec	ratio	PSNR	t sec	ratio	PSNR	t sec	ratio	PSNR	t sec	ratio
foreman	3.4	136	168	989	1294	0.32	27.2	3.4	7.8	25.4	3.3	7.6	33.5	12.1	2.7	23.5	0.9	25.1

Our interest in CS techniques is in applying them to problems related to surveillance videos. Most researchers applying CS to video are interested in a more general application of this technique to all types of video. The videos we have used represent the types of videos we expect to encounter in our applications. However, our restriction to these types of videos leaves open the question of how our technique would work against more traditional video sequences. To address this question, we show the results for the "Foreman" video sequence in Table 3.4. This video was downsampled to the same 64x64 pixel size as used for the other videos. This video is very different in character from the ones shown previously in this paper. The amount of change between frames as indicated by the Δ parameter is an order of magnitude bigger. As with most of the other videos, our technique has PSNR values

second only to the frame differencing method. However, the Block CS technique in this case outperforms our method significantly with respect to execution time. This is to be expected since our algorithm scales with the number and size of segments changing and there is a significant amount of change in this video. The Block CS method uses Contourlets as a sparsifying transform and seems to be more effective when the changes are larger. Since reconstruction in Block CS is done in the sparse domain and then transformed, this indicates that the "Foreman" video possess greater sparsity in this domain than the other videos used in this study. While this indicates that our algorithm loses performance for the conditions inherent in the "Foreman" video, the prior results presented in this paper indicate that Block CS loses performance when the change in videos is small. A full exploration of the reasons for this behavior is left for future study.

3.3.2 Parameter analysis

In order to see effects due to segmentation, the number of segments was varied keeping the parameter κ constant. PSNR was recorded against total subrate which is the sum of scene measurements, ME measurements, and mask transmission measurements and is plotted in Fig 3.11. The numbers above the points in Fig 3.11 represent the number of segments used. The PSNR does trend upward with an increase in subrate but is not strictly monotonic function of subrate. There is very little apparent relationship between the number of segments and PSNR. ME measurements and mask transmission measurements are directly related to the number of segments for simulated and real videos but they do not appear to directly relate to PSNR. Each video has an optimal number of segments for which PSNR is maximum. We believe that PSNR is primarily reflecting the characteristics of the segmentation algorithm used. As a result, further analysis was deemed outside the original scope of this paper and will be pursued in future research.

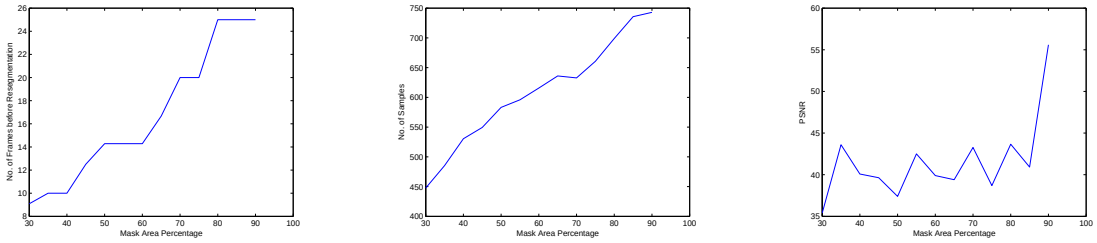


(a) (b)

Figure 3.11: PSNR Variation with subrate by changing Number of Segments a. Simulated Video b. Real Video, the number beside each data point shows number of segments used.

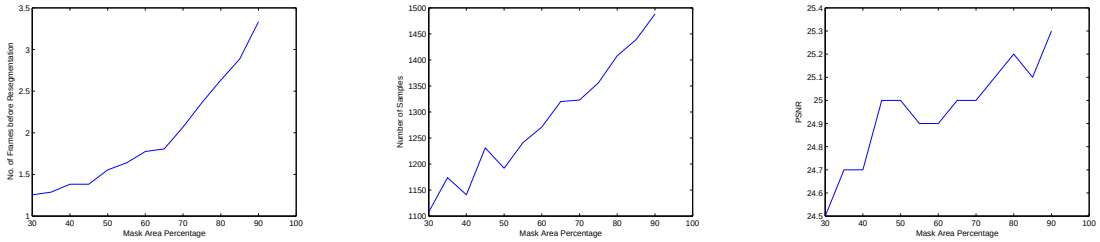
We varied the upper bound on the area of the mask that has to be reached before performing resegmentation and studied the impact on the number of samples, the interval for resegmentation, and PSNR of the reconstructed frames for the simulated moving letter sequence and a surveillance intelligent room video sequence. In the case of moving letter sequence, the increase on the upper bound on the mask area resulted in an increase in the required number of samples but the PSNR remained almost the same over the whole range of the upper bound as shown in Figure 3.12b and c. The resegmentation interval extends up to 33 frames for the maximum upper bound of 100% while for the minimum upper bound of 30% it reduced to 10 frames as shown in Figure 3.12a over a length of 90 frames. In the case of the Intelligent room video, when the upper bound on the mask area was increased, the number of samples required for reconstruction rose according to Eq. 3.12. This result was expected. The increase in PSNR is proportional to the rise

in the upper bound but does not exhibit a significant change and varies only up to 2db as shown in Figure 3.13c. The number of times resegmentation was performed during 300 frames of the intelligent room video decreased with the increase in upper bound. This reached up to a gap of 16 frames for a maximum upper bound of 100%. The number of samples reached 45.9% of the total number of pixels for the maximum upper bound. Conversely, for a minimum lower bound of 30%, which was kept above the percentage changes between two frames, the resegmentation was performed after every 1.3 frames and the corresponding number of samples required was 27.09% as shown in Figure 3.13 a and b. These experiments show that in order to get optimum performance, the value for the resegmentation interval should be adjusted according to the dynamic area. Some other test videos were used from online repository and results are shown in Fig. 3.14, 3.16, 3.17 and 3.18.



(a) (b) (c)

Figure 3.12: Variation of a. Segmentation interval b. No. of Samples c. PSNR of reconstructed frame with Mask Area upper bound for Simulated Moving Letter video sequence



(a) (b) (c)

Figure 3.13: Variation of a. Segmentation interval b. No. of Samples c. PSNR of reconstructed frame with Mask Area upper bound for Intelligent Room video sequence

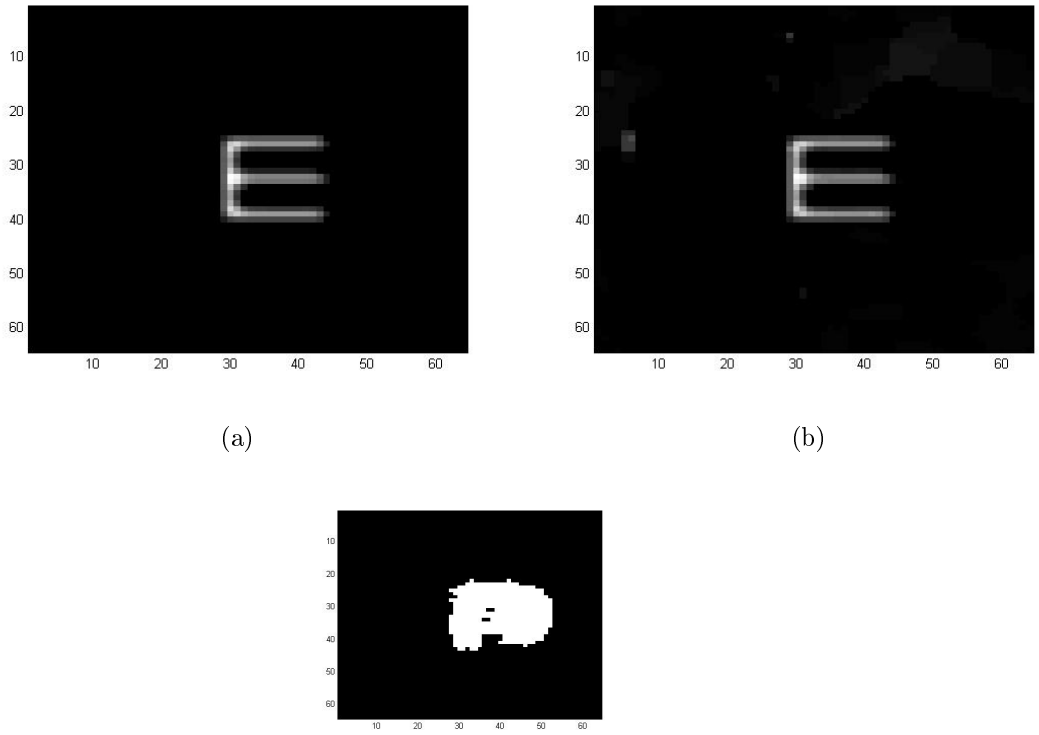


Figure 3.14: A single frame from Moving Letter video a. Original frame b. Reconstructed frame c. Measurement Mask Reconstructed using $0.05n^2$ with PSNR=33.8db

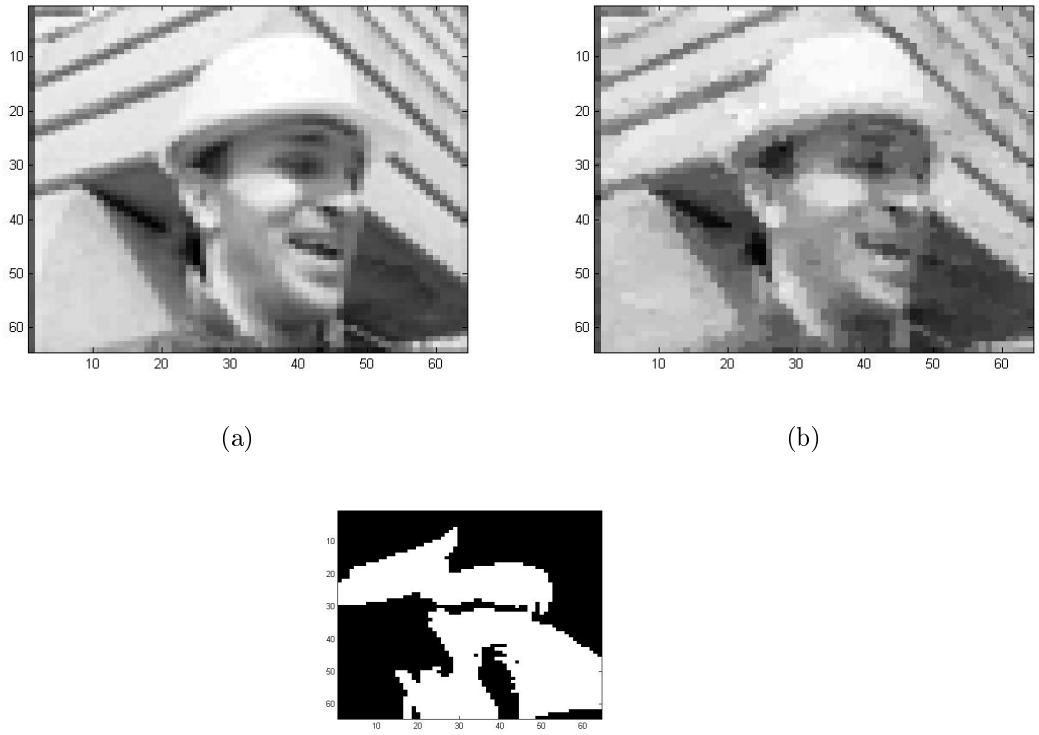


Figure 3.15: A single frame from foreman and surveillance video a. Original foreman b. Reconstructed foreman c. Measurement Mask Reconstructed using $0.17n^2$with PSNR=25.15db

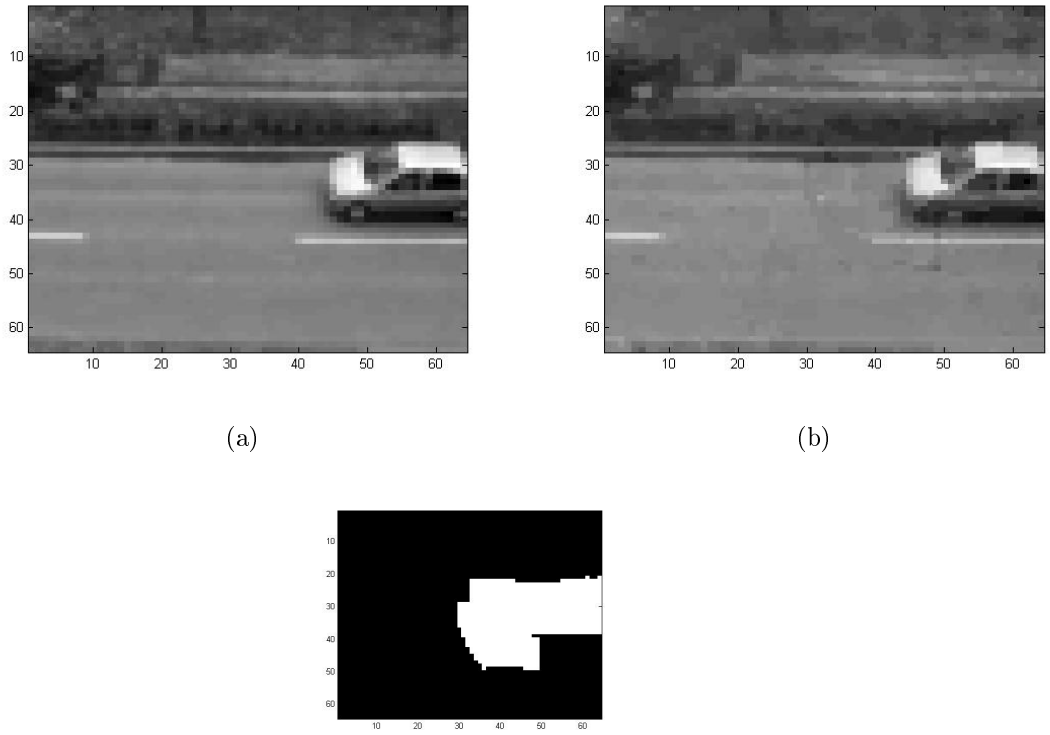
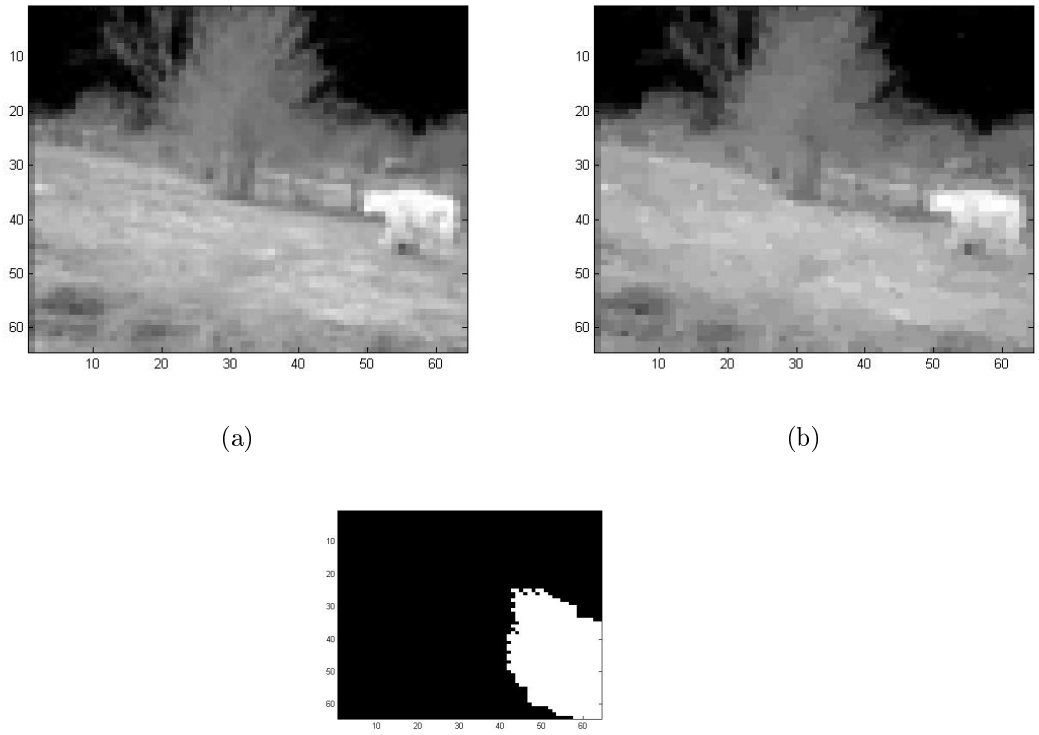


Figure 3.16: A single frame from Traffic Surveillance video a. Original b. Reconstructed c. Measurement Mask Reconstructed using $0.17n^2$ with PSNR=32.7db

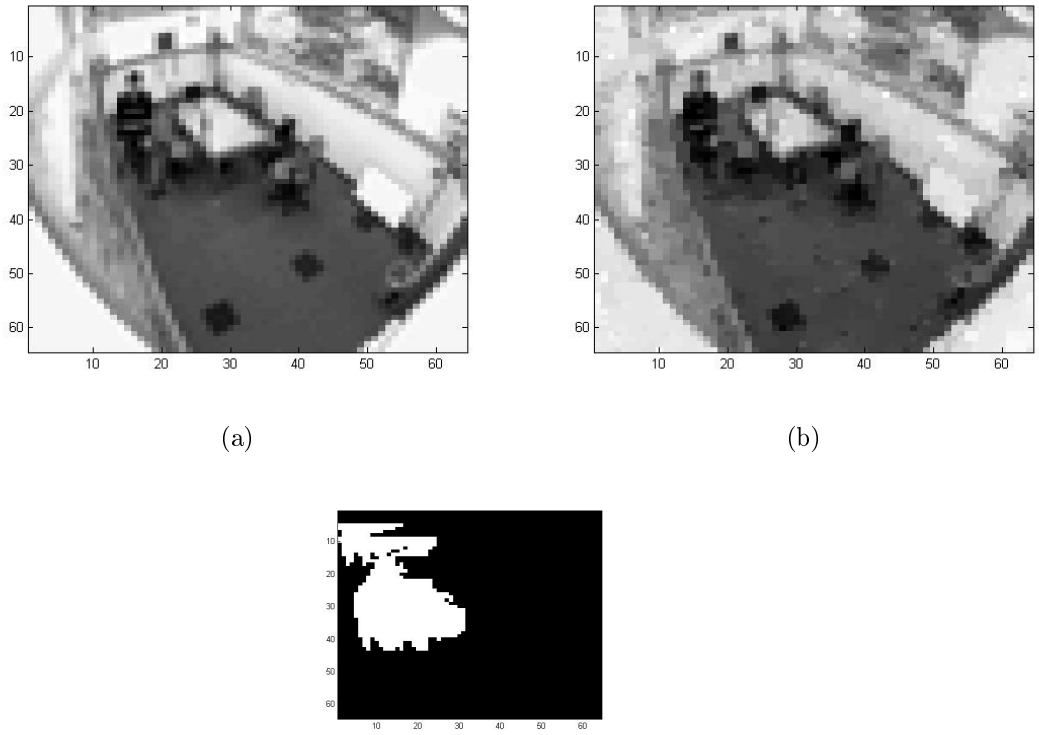


(a)

(b)

(c)

Figure 3.17: A single frame from Border Surveillance video a. Original b. Reconstructed c. Measurement Mask Reconstructed using $0.17n^2$ with PSNR=31.8db



(a) (b)

(c)

Figure 3.18: A single frame from Intelligent room video a. Original b. Reconstructed c. Measurement Mask Reconstructed using $0.17n^2$ with PSNR=25.8db

After mask transmission and until resegmentation, measurement allocation in our algorithm is performed at the sensor level with less complexity and higher speed than previous methods. The method is adaptive to the complexity of the scene. A change in average from a previous frame is calculated before sampling. At the decoder, only dynamic pixels are reconstructed reducing the complexity by the number of static pixels. Some methods based on motion estimation and motion compensation, such as ME/MC BCS-SPL and MH-BCS-SPL accumulate the measurements for a series of frames and perform reconstruction of all frames simultaneously [19, 18, 17]. The method proposed here reduces the complexity of the optimization process used in reconstruction not only by single frame reconstruc-

tion but also by reconstructing only the dynamic area of each frame. Scenes where the dynamic area is small and the motion is not complex can be potentially reconstructed in real-time. The resegmentation interval is adaptive to the complexity and the spread of motion in a video as shown in Table 3.2 and 3.3 , thereby reducing the computational requirements for segmentation. In addition, if the slow change assumption holds, segmentation of previous frames can be used for forming the mask and performing motion estimation which removes the constraint of generating new masks after reconstruction.

In a feature comparison to existing adaptive block based techniques which require optimization of measurement allocation before each frame on the decoder side, this technique can acquire a number of frames with far less computational time required on the encoder side depending on the dynamic areas spread in the scene[22]. We have observed that if the segmentation is of good quality then this method is efficient for surveillance videos in terms of computations and sampling efficiency.

In this study we have taken the last reconstructed frame for resegmentation but in order to avoid latency due to the segmentation process, we can also make it a parallel background process. Segmentation can be performed after each reconstruction and the last available segmented frame before the mask area upper bound is reached can be used to avoid any delays in making measurements. This scheme basically reduces computations by load sharing between two routines. It can be further improved by implementing better parameter estimation techniques and by optimizing the process of measuring averages for motion detection. The TV minimization package used for reconstruction in this paper had the parameters shown in Table 3.5.[26]

Table 3.5: Parameters used for TV minimization

Parameter	Value
opt.mu	2^8
opt.beta	2^5
opt.tol	1×10^{-3}
opt.maxit	300
opt.TVnorm	1
opt.nonneg	true

In order to give an overview of computational time requirements to realize this technique, we calculated some values based on the first real video results. The nominal flipping rates of 200 nsec for a micromirror array are taken from the advertised specifications of the device.[**?**] The sampling and transmission is assumed to be performed at the same rate. Total time for measurement acquisition, motion estimation, mask update and transmission latency is calculated and subtracted from the minimum time to render a frame for real-time streaming. This time bound can be used to complete reconstruction and segmentation in parallel. The results are listed in Table 3.6.

Table 3.6: Computational time bounds for real-time reconstruction

Task	Analytical Time	Time Requirement msec
Measurement Acquisition	$M3 \times sampling\,rate$	0.09
Motion Estimation	$M2 \times sampling\,rate + Subtraction\,time$	0.0009
Mask update	$transmission\,rate \times M1$	0.0002
Transmission latency	$(M1 + M2 + M3) \times sampling\,rate$	0.10
Reconstruction || Segmentation	$0.33 - Total\,time$	320
All		<0.33 sec

In order to summarize the important points, in the proposed technique, after mask transmission and until resegmentation, the measurement allocation is performed at the sensor level with less complexity and high speed. The method is adaptive to the complexity of the scene. A change in average from previous frame is calculated before sampling. At the decoder only dynamic pixels are reconstructed reducing the complexity and thereby increasing the efficiency of reconstruction algorithms.

3.4 Conclusion and Future Work

We have discussed in this paper a new scheme to acquire measurements for video reconstruction. We found this scheme useful for temporal compression in

videos having static background and slow foreground changes over time. Depending on the video, this scheme is able to decrease reconstruction time and computations compared to some existing sampling techniques. Furthermore, the motion estimation is very efficient from a hardware implementation perspective. However, while we believe we have made a good case that the computational burden of this algorithm is actually less than most other compressive sensing techniques for video, it should not be compared with traditional video compression. Given the fact that memory is inexpensive, visible band cameras are inexpensive, and hardware coding/decoding is fairly inexpensive for traditional video devices, the desirability of any compressive sensing technique is limited for visible band sensors. However, our scheme can prove useful at wavelengths where an array of sensors is expensive and single pixel detection is the most cost efficient method for producing video. Examples would include terahertz sensors[27] and perhaps infrared [12]. The gains from this scheme can be utilized towards reducing reconstruction time and computational requirements or increasing frame rates for video imaging at wavelengths where sensor arrays are expensive. In order to improve the results shown here, use of spatial sparsity transform domain knowledge incorporated with the sampling matrix could prove fruitful. Further studies of the impact of parameters such as the radius of the circle and shape to encircle should also be performed. The motion estimation scheme can be optimized to use least averages for predicting the direction. Further, other robust methods can be investigated for parameter estimation. Noise should be taken into consideration to study the effects on performance and a denoising schemes designed and applied for reducing the artifacts due to quantization and detection noise. We are constructing a hardware simulator of a single pixel camera device for testing and further development of this algorithm.

Chapter 4

Sparse Transform and Sensor Selection

4.1 Introduction

Submillimeter wave are capable of traveling through non-conducting materials such as cloth, paper, cardboard etc. without much attenuation. Many applications such as security scanning and tumor detection require submillimeter wave imaging. In this thesis, we consider the problem of estimating an undersampled signal formed by sub-millimeter waves(SMMW). Earlier methods solved an inverse problem for signal reconstruction which requires as many or more measurements than the signal dimension. We address the issue of reconstruction from less measurements than required by a least square or inverse solution and show that the number of measurements required for optimum reconstruction is less than the dimensions of the signal for a specific error bound. If the measurement matrix satisfies certain required conditions, then we can use CS methods for reconstruction using far fewer measurements. To reconstruct an undersampled signal using CS principles, we need knowledge of the sparsifying basis for SMMW images. For this purpose we used an online dictionary learning algorithm to find a sparsity basis in which the signal can be represented by fewer atoms and is also incoherent with the measurement matrix.

The measurements obtained along with the product of the measurement matrix and sparsity basis is passed to the reconstruction algorithm. We cast the problem as a basis pursuit optimization. It estimates a solution to a set of underdetermined linear equations. The objective function l_1 norm minimization guarantees finding the sparsest solution consistent with the measurements if RIP is satisfied.[14] This problem falls into the convex optimization category, ensuring the solution is unique and tractable. As discussed above there are a lot of libraries available to solve convex optimization problems.[28] We used the disciplined convex optimiza-

tion CVX modeling system used for implementing BP optimization. It uses a base library of convex functions and sets. The solution obtained is an estimate of the coefficients in the sparse basis. The actual signal is obtained by transforming the coefficients back to the spatial domain. We demonstrate this method by reconstructing images from simulations and experimental measurements.

4.2 Imaging Device Architecture

The SMMW imager consist of heterodyne source and receiver pair, image forming optics, a spatially selective mask, and data acquisition and post-processing hardware and software. The receiver detects the intensity coming from the object, filtered by the spatially selective mask. An analog to digital converter outputs a digital signal after receiving an analog output from receiver.[29] The hardware outputs linear measurements and an algorithm using CS methods is used to reconstruct the signal. After digitizing the measurements we feed the measurements to the reconstruction algorithm.

The waves guided by an optical setup are made to fall on the mask through an imaging window.[2] The mask consist of a disk with holes along a constant radius of same size at random locations. The size and spacing of holes dictates the quality of the reconstructed image. we will analyze errors for a range of permissible sizes and spacings to find the optimal parameters.

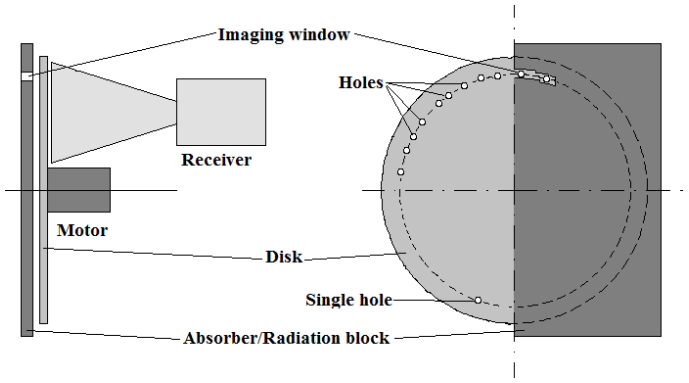


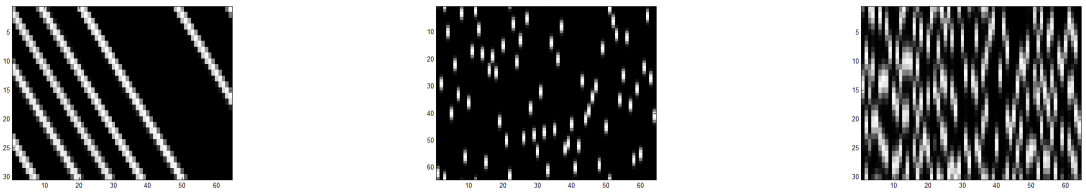
Figure 4.1: Conceptual diagram of spatial mask and receiver[2]

4.3 Solution Optimization

The measurement process of a spatially selective mask can be modeled by a matrix. The measurement matrix used for calculations is shown in Fig. 4.2. There are many factors which can be optimized in construction of the mask. In this paper we talk about two parameters: hole diameter and spacing. The thickness of the diagonals signifies the diameter of holes and the distance between the diagonals is related to the spacing between holes. We optimized the spacing and diameter of holes over a range of permissible values by minimizing the mean square error. The optimum value of diameter and spacing was used for further computations and implementation. The measurements are obtained using obtained values.

Submillimeter wave images can be well represented by a sum of Gaussians due to their inherent blur. As we know a linear combination from a set of basis functions spanning the entire space can be used to represent any signal. We acquired the image of a point source and translated it over the whole spatial range. The set of signals obtained are fed to a dictionary learning algorithm to obtain a transformation by training it with images.[30] The parameters for the algorithm were chosen empirically. The dictionary learning algorithm enforced minimum $l_1$norm to obtain a sparse representation.[30] We use the learned dictionary along with the measure-

ment matrix for reconstruction. For reconstruction of the signal from an optimal number of measurements, the two matrices are tested for their degree of incoherence. The sum of columns of the gram matrix as in equation1.6 for both matrices was found to be in the range of equation 1.4, which implies the incoherence of these two matrices.



(a) (b) (c)

Figure 4.2: a. 30x64 Matrix M b. A 64 x 64 Dictionary D c. Product of D and M

Now that we have written our problem as an underdetermined system of linear equations. We have cast the problem as a BP and use the SeDuMi solver for optimization. It was chosen due to its lower computation time and consistent performance compared to other available algorithms.

For noiseless reconstruction we solve using equality constraints. In case of noisy reconstruction we first estimate the value of an error bound ε. The value of the error bound is kept greater than the noise variance. For a fixed number of measurements we estimated optimum ε which minimized the MSE of the reconstructed signal. The optimum value of error bound for a specific number of measurements was used for noisy reconstruction.

4.4 Mask Position Selection

The pattern of holes at an instance forms the projection vector. For a disk of radius specified, a fixed number of hole patterns can be used for making measure-

ments. In this study we used 1000 different patterns for acquiring measurements. In order to select the measurement vector combination that produce least error we used convex optimization to select an optimum subset of the sensor measurements to reconstruct the image.[31] From the measurement model a single measurement can be written as

$$y_i = a_i^T x + v_i$$

where a is the measurement vector, x is the image and y denotes the measurements. The maximum likelihood estimate of the solution can be expressed as

$$\hat{x} = \left(\sum_{i=1}^{m} a_i a_i^T \right)^{-1} \sum_{i=1}^{m} y_i a_i$$

The log volume of the confidence ellipsoid which contains the solution can be expressed as

$$log\ vol(\varepsilon_\alpha) = \beta - \left(\frac{1}{2}\right) log\ det \left(\sum_{i=1}^{m} a_i a_i^T \right)$$

The log volume measures the information in the subset of measurements. we maximize the the log function to incorporate maximum information in the selected subset S.

$$maximize \quad log\ det \left(\sum_{i \epsilon S} a_i a_i^T \right)$$

$$subject\ to \qquad |S| = k$$

where S is a subset of {1,2, ... , m}. and |S| is the cardinality of S. Here k is the number of selected measurements.

4.5 Results and Discussion

We simulated 90 signals of length 64x1 with a 64x64 measurement matrix. A measurement vector of size 64x90 was calculated. A 64x64 dictionary was obtain by training 64 samples of point spread functions spanning the whole space shown in Fig 4.2. A line image was created spanning the spatial space of length 64x1. The signal was reconstructed using 30 noiseless measurements for each sample. The optimum hole size and spacing was found by minimizing MSE over a range of permissible values of hole sizes and maximum spacing. Fig. 4.3 shows the variation of MSE with hole diameter and maximum spacing.

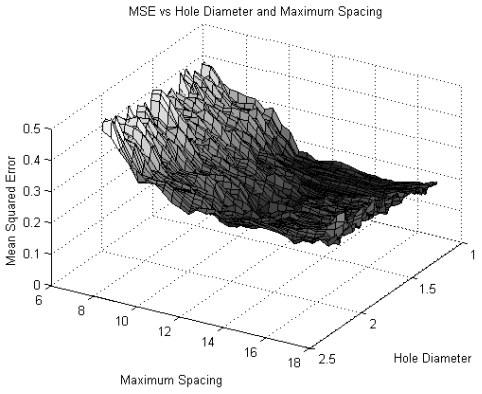
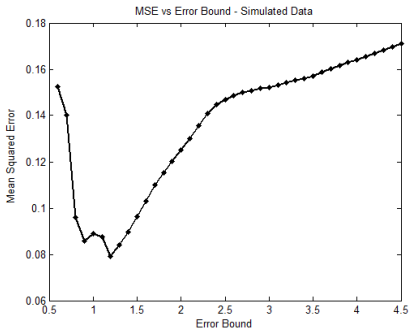
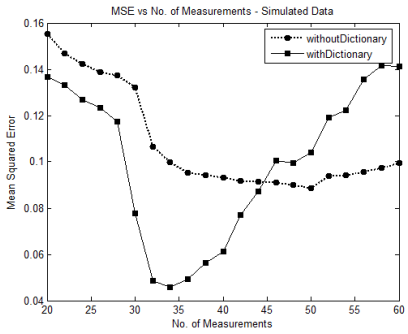


Figure 4.3: MSE vs Hole Diameter and Maximum spacing in mm

We simulated noisy measurements with $\sigma = 0.2$. Using the above results for the optimized hole size and spacing, we minimized MSE over a range of error bound ε starting from the minimum $\varepsilon = \sigma$. The error bound against least MSE was found to be 1.2 which is used in further analysis. Fig. 4.6a shows the MSE plotted against a range of error bound for a specific number of measurements.

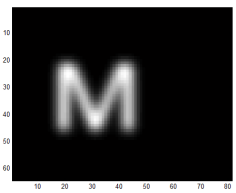


(a)

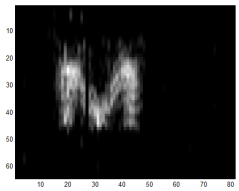


(b)

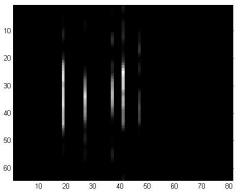
Figure 4.4: Reconstruction optimization and analysis a. MSE vs Error bound b. MSE vs Measurements



(a)



(b)



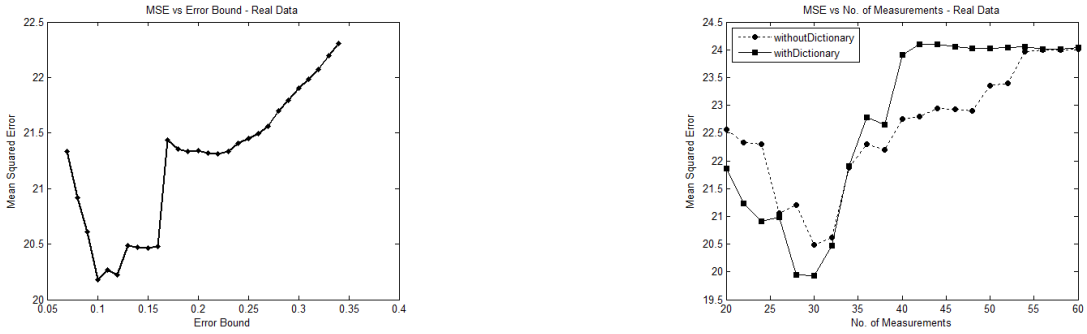
(c)

Figure 4.5: Simulated signal 90 (64x1) vectors a. Object M b. Reconstruction using m = 30 c. Reconstruction using m = 60

Fig. 4.6b shows the mean square error plotted against the number of measurements for a fixed $\varepsilon = 1.2$. We also compare the MSE with and without the aid of a dictionary for reconstruction. The error using the dictionary is lower than without the dictionary when smaller numbers of measurements have been used. After the specified number of measurements is passed the error starts to increase. The specified number of measurements in this case were 30 which shows optimum reconstruction can be achieved by less than half the number of measurements required by a conventional algorithm for a given error bound. Moreover this optimum re-

construction is achieved earlier with the aid of a dictionary in the reconstruction process.

We gathered real data using the imaging device for an object representing the character "M" and shown in Fig. 4.7. The value of optimum error bound ε is calculated by minimizing MSE over a range of ε for a specific number of measurements as shown in Fig 4.6a.



(a) (b)

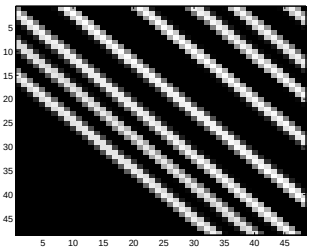
Figure 4.6: Reconstruction, optimization and analysis of real data a. MSE vs Error bound b. MSE vs no. of Measurements

(a) (b)

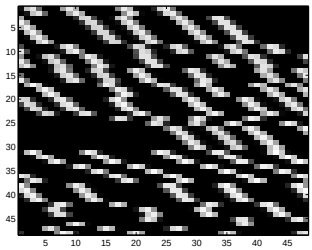
Figure 4.7: Real Data 191 (64x1) vectors a. Object M b. Reconstruction using m = 30 c. Reconstruction using m = 60

The mean square error increases after exceeding the specific number of measurements as shown in Fig. 4.6b. The error is lower initially when the dictionary is used in the reconstruction process before the specified number of measurements. Fig 4.7b shows the reconstructed object M shown in Fig. 4.7a from 30 measurements for each vector . The results were consistent with simulations.

The selection of measurements using convex optimization was simulated for the letter M. MSE was recorded for the measurement matrix built using the selection algorithm. Fig. 4.9 shows the results for consecutive measurement and selected measurements reconstructions using same number of measurements. Fig. 4.8 a and b shows the measurement matrix formed using consecutive and selective measurements respectively.

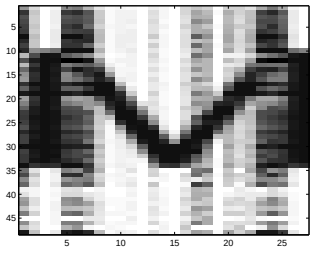


(a)

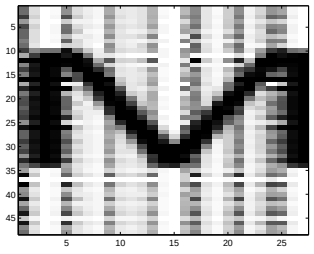


(b)

Figure 4.8: a. Measurement Matrix formed using consecutive measurements b. Measurement Matrix formed using selected measurements using convex optimization



(a)



(b)

Figure 4.9: a. Reconstruction using 35 consecutive measurements MSE=11.02 b. Reconstruction using 35 selected measurements MSE=10.32

4.5.1 Conclusion

The SMMW imager under analysis requires an object to be stationary until it makes same number of measurements as the resolution of the device. Since fewer measurements are required by compressive sensing reconstruction methods we can let the object move at a faster speed before inducing motion blur. To further improve the RIP property of measurement matrix we will also work on adding another degree of freedom to the distance of holes from center.

On the processing side the quality can be enhanced by thresholding the coefficients before transforming them back to spatial domain. Some methods will be explored to find the optimum threshold level. Also there is room to improve the role of the sparsity basis, so that its more robust to noise in the signal and other variations. Sparsity in the third dimension (time) can be explored to further reduce the number of required samples.

Bibliography

[1] R. G. B. Dharmpal Takhar, Jason N. Laska, Michael B. Wakin, Marco F. Duarte, Dror Baron Shriram Sarvotham, Kevin F. Kelly, "A new compressive imaging camera architecture using optical-domain compression," *Proceedings of SPIE* **6065**(January), pp. 606509–606509–10, 2006.

[2] O. Furxhi and E. L. Jacobs, "A sub-millimeter wave line imaging device," *Image (Rochester, N.Y.)* , pp. 76700L–76700L–12, 2010.

[3] L. Baldassarre, J. Morales, A. Argyriou, and M. Pontil, "A General Framework for Structured Sparsity via Proximal Optimization," pp. 82–90, 2012.

[4] D. L. Donoho, , and J. Tanner, "Neighborliness of randomly projected simplices in high dimensions.," *Proceedings of the National Academy of Sciences of the United States of America* **102**, pp. 9452–7, July 2005.

[5] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing,* **41**(12), pp. 3397–3415, 1993.

[6] S. S. Chen, D. L. Donoho and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing* **20**(1), pp. 33–61, 1998.

[7] S. D. Babacan, R. Molina, and A. K. Katsaggelos, "Bayesian Compressive Sensing Using Laplace Priors," in *IEEE transactions on image processing*, **19**(1), pp. 53–63, 2010.

[8] J. N. Laska, P. T. Boufounos, M. a. Davenport, and R. G. Baraniuk, "Democracy in action: Quantization, saturation, and compressive sensing," *Applied and Computational Harmonic Analysis* **31**, pp. 429–443, Feb. 2011.

[9] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)* **22**(8), p. 731, 1997.

[10] P. L. Correia and F. Pereira, "Objective evaluation of video segmentation quality.," *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* **12**, pp. 186–200, Jan. 2003.

[11] I. Noor and E. L. Jacobs, "Adaptive Compressive Sensing Algorithm for Video Acquisition Using a Single Pixel Camera," in *Compressive Sensing, Proceedings Vol. 8036*, SPIE, 2012.

[12] R. M. Willett, R. F. Marcia, and J. M. Nichols, "Compressed sensing for practical optical imaging systems: a tutorial," *Optical Engineering* **50**(7), p. 072601, 2011.

[13] R. Robucci, L. Chiu, J. Gray, J. Romberg, P. Hasler, and D. Anderson, "Compressive sensing on a CMOS separable transform image sensor," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP),*, 2009.

[14] E. J. Candès, "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique* **346**, pp. 589–592, May 2008.

[15] A. C. Sankaranarayanan, P. K. Turaga, R. Chellappa, and R. G. Baraniuk, "Compressive acquisition of dynamical scenes," in *ECCV'10 Proceedings of the 11th European conference*, pp. 129–142, 2010.

[16] N. Vaswani and W. Lu, "Modified-CS : Modifying Compressive Sensing for Problems with Partially Known Support," *Compare A Journal Of Comparative Education* , pp. 1–12.

[17] J. E. Fowler, S. Mun, and W. Tramel, "Block-Based Compressed Sensing of Images and Video," *Foundations and Trends in Signal Processing* , pp. 1–123, 2012.

[18] E. W. Tramel and J. E. Fowler, "Video Compressed Sensing with Multihypothesis," (March), 2011.

[19] H. Jung, K. Sung, K. S. Nayak, E. Y. Kim, and J. C. Ye, "k-t FOCUSS : a general compressed sensing framework for high resolution dynamic MRI," *Magnetic Resonance in Medicine,* **61**(1), pp. 103–116, 2009.

[20] J. E. Fowler, S. Mun, and E.W. Tramel, "Multiscale block compressed sensing with smoother projected Landweber reconstruction," *Proceedings of the European Signal Processing Conference,* , pp. 564–568, 2011.

[21] S. Mun, M. Starkville, and J. Fowler, "Residual reconstruction for block-based compressed sensing of video," in *Proceedings of the IEEE Data Compression Conference,*, pp. 183–192, 2011.

[22] Zhaorui Liu and A. Y. Elezzabi and H. Vicky Zhao, "Maximum Frame Rate Video Acquisition Using Adaptive Compressed Sensing," *IEEE Transactions, Circuits and Systems for Video Technology,* **21**(11), pp. 1704–1718, 2011.

[23] L.-W. Kang and C.-S. Lu, "Distributed compressive video sensing," *2009 IEEE International Conference on Acoustics, Speech and Signal Processing* , pp. 1169–1172, Apr. 2009.

[24] J. Zheng and E. L. Jacobs, "Video compressive sensing using spatial domain sparsity," *Optical Engineering* **48**(8), p. 087006, 2009.

[25] P. L. Rosin, "Unimodal Thresholding," *Science Direct* **34**(11), pp. 2083–2096, 2001.

[26] Li Chengbo, Yin Wotao, and Zhang Yin, "TV Minmization http://www.caam.rice.edu/optimization/L1/TVAL3/," 2010.

[27] W. L. Chan, K. Charan, D. Takhar, K. F. Kelly, R. G. Baraniuk, and D. M. Mittleman, "A single-pixel terahertz imaging system based on compressed sensing," *Applied Physics Letters* **93**(12), pp. 12110–5, 2008.

[28] M. Grant and S. Boyd, "CVX Users Guide," 2010.

[29] O. Furxhi and E. L. Jacobs, "A Sub-Millimeter Wave Line Scanning Imager,"

[30] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09* , pp. 1–8, 2009.

[31] S. Joshi and S. Boyd, "Sensor Selection via Convex Optimization," *Design* **57**(2), pp. 451–462, 2009.

[32] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A Simple Proof of the Restricted Isometry Property for Random Matrices," *Constructive Approximation* **28**, pp. 253–263, Jan. 2008.

[33] P. Blomgren and T. F. Chan, "Color TV: total variation methods for restoration of vector-valued images.," *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* **7**, pp. 304–9, Jan. 1998.

[34] E. J. Candès, "A Probabilistic and RIPless Theory of Compressed Sensing," *Information Theory, IEEE Transactions* **57**, pp. 7235 – 7254, 2011.

[35] E. J. Candès and J. Romberg, "Sparsity and Incoherence in Compressive Sampling," *Inverse Prob* **23**(3), pp. 969–985, 2007.

[36] E. J. Candès, J. Romberg, and T. Tao, "Stable Signal Recovery from Incomplete and Inaccurate Measurements," *IEEE Transactions on Information Theory* **52**(12), pp. 5406–5425, 2006.

[37] E. J. Candès and T. Tao, "Near Optimal Signal Recovery From Random Projections : Universal Encoding Strategies ?," *IEEE Transactions on Information Theory* **52**(12), pp. 5406–5425, 2006.

[38] E. J. Candès and M. Wakin, "An Introduction To Compressive Sampling," *IEEE Signal Processing Magazine* **25**, pp. 21–30, Mar. 2008.

[39] Cour Timothee, Yu Stella, and Shi Jianbo, "Normalized Cut Segmentation," 2004.

[40] I. Daubechies, C. D. E. Mol, and U. L. D. Bruxelles, "An Iterative Thresholding Algorithm for Linear Inverse Problems with a Sparsity Constraint," *Communications on Pure and Applied Mathematics,* **57**(2004), pp. 1413–1457, 2004.

[41] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, "Gradient Projection for Sparse Reconstruction : Application to Compressed Sensing and Other Inverse Problems," *IEEE Journal on Selected Areas in Communications* **1**(4), pp. 586–597, 2007.

[42] L. Gan, C. Ling, T. T. Do, and T. D. Tran, "Analysis of the Statistical Restricted Isometry Property for Deterministic Sensing Matrices Using Steins Method," 2009.

[43] R. F. Marcia and R. M. Willett, "COMPRESSIVE CODED APERTURE VIDEO RECONSTRUCTION," in *Proc. EUSIPCO*, (1), 2008.

[44] D. Needell and J. A. Tropp, "Cosamp: iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis* **26**(3), pp. 301–321, 2008.

[45] I. Noor, O. Furxhi, and E. L. Jacobs, "Compressive Sensing for a Sub-millimeter Wave Single Pixel Imager," in *Passive Millimeter-Wave Imaging Technology XIV, Proceedings Vol. 8022*, SPIE, 2011.

[46] H. Nyquist, "Certain topics in telegraph transmission theory," *Proceedings of the IEEE* **90**(2), pp. 280–305, 2002.

[47] J. Y. Park, A. Arbor, and M. B. Wakin, "A MULTISCALE FRAMEWORK FOR COMPRESSIVE SENSING OF VIDEO," *Proceedings of the Picture Coding Symposium* , pp. 1–4, 2009.

[48] R. Robucci, J. Gray, L. K. Chiu, J. Romberg, and P. Hasler, "Compressive Sensing on a CMOS Separable-Transform Image Sensor," in *Proc. IEEE*, pp. 1089–1101, 2010.

[49] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory,* **53**(12), pp. 4655–4666, 2007.

[50] R. Vezzani and R. Cucchiara, "Video Surveillance Online Repository (ViSOR): an integrated framework," 2010.

[51] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo, "Sparse Reconstruction by Separable Approximation," *IEEE Transactions on Signal Processing* **57**(7), pp. 2479–2493, 2009.

[52] W. Xu, "Compressive sensing for sparse approximations: constructions, algorithms and analysis," *CaltechThesis* , 2010.

[53] H. Yu, G. Cao, L. Burk, Y. Lee, J. Lu, P. Santago, O. Zhou, and G. Wang, "Compressive sampling based interior reconstruction for dynamic carbon nanotube micro-CT.," *Journal of X-ray science and technology* **17**, pp. 295–303, Jan. 2009.

[54] Z. Zhang and B. D. Rao, "Iterative reweighted algorithms for sparse signal recovery with temporally correlated source vectors," *ICASSP* , pp. 0–3, 2011.

Appendix A

Matlab Listings

A.1 Main Function

```
clc
clear all
pathname = pwd;
filename = strcat(pathname,'/foreman.avi');
BndryThres = .23;
global RowThres nbSegments sizeIm
RowThres = .01;
segm = 1;
dm = 1;
sizeIm = 64;
SegMov = zeros(nbSegments,1);
MaskforMeas = zeros(sizeIm);
H = fspecial('gaussian',[2 2],1);
%[movdr nFrames vidHeight vidwidth] =
readvideo(filename,sizeIm);
load 'foremanframes207';
%load('Cow3Video310Frames')
%load ('MovingLetters')
%load ('intelligentroom300frames')
%load('visor_TrafficSurvillance')
%load('visor_ManWalkingOcludded')
%load('Camera1_SchoolParking')
% for i=1:nFrames
```

```
%movdr(i).cdata=movdr(i).cdata./max(max(movdr(i).cdata));
%end
%load('dscr8')
%load('dsch1')
%load('ird1')
%load('dscsp7n1')
%load('irdp07')
nFrames = 30;
%%%%%%%%%%Incldue for Infrared Videos%%%%%
%nFrames = size(mov,2);
%% movdr(1:nFrames) = ...
%%struct('cdata', zeros(sizeIm, sizeIm, 1, 'double'),...
%%                    'colormap', []);
%% k = 1;
%% for i=25:nFrames
% start at 250 for infrared video
%%      movdr(k).cdata = mov(i).cdata./255;
%%      k = k + 1;
%% end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Fix_Seq(1:6) =
struct('data',zeros(sizeIm, sizeIm, 1, 'uint8'));
for i=1:6
        Fix_Seq(i).data = sqrt(6).* randn(sizeIm,sizeIm);
end
ReconsIm(1:nFrames) =
struct('cdata', zeros(sizeIm, sizeIm, 1,
```

```
'double'),'colormap', []);
MeasMatrix(1:nFrames)=
struct('cdata', zeros(sizeIm, sizeIm, 1,
'uint8'),'meas',zeros(1,1,1,'double'), ...
'colormap', []);
CorrMotn(1:nFrames-200)=
struct('Self',zeros(sizeIm,sizeIm,1,'double'),
'Crx',zeros(sizeIm,sizeIm,1,'double'));
movd_seg(1:nFrames) =
struct('cdata',zeros(sizeIm,sizeIm,1,'double'));
p1=0;
k=1;%1;
Re_seg = 1;
DetChangeSeg = 0;
DB_ratio = .1;
VarA = [];
ContA = [];
nbSegments = 5;
b=1;
NoiseFloor = 0;
SelectSeg(1:8)=
struct('cdata',zeros(sizeIm,sizeIm,'double'));
ReconsIm(k).cdata=movdr(k).cdata;
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%Firstframereconstruction%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
[r,c] = find(movdr(k).cdata > 0.22);
NoMeas = floor(log10(sizeIm*sizeIm)
      *size(r,1)*1/.8);%10*percen
NoMeas = .6*4096;%size(r,1);
NewMeasMat_ber = zeros(NoMeas,sizeIm*sizeIm);
NewMeasMat = ones(64,64);
for j = 1:NoMeas
         Ber = sqrt(6)*randn(sizeIm);
    NewMeasMat_ber(j,:) = (NewMeasMat(:).* Ber(:))';
end
NewMeasVector = NewMeasMat_ber *
                movdr(k).cdata(:);
opts.mu = 2^8;
opts.beta = 2^5;
opts.tol = 1E-3;
opts.maxit = 300;
opts.TVnorm = 1;
opts.nonneg = true;
    [U, out] = TVAL3(NewMeasMat_ber,NewMeasVector
             ,sizeIm,sizeIm,opts);
ReconsIm(k).cdata = reshape(U,sizeIm,sizeIm);
MeasMatrix(k).cdata = NewMeasMat;
MeasMatrix(k).meas = NoMeas;
clear NoMeas r c
sumt = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
```

```
%Segment the previous frame and run inside
loop for all segments
while (k <= 30) %140 for simulated videos
        DiffBoundary=sum(abs(movdr(k).cdata(1,:)
         - movdr(k+1).cdata(1,:)))
         +sum(abs(movdr(k).cdata(:,1)
                 - movdr(k+1).cdata(:,1)))
              +sum(abs(movdr(k).cdata(sizeIm,:)
                 - movdr(k+1).cdata(sizeIm,:)))
        +sum(abs(movdr(k).cdata(:,sizeIm)
        - movdr(k+1).cdata(:,sizeIm)));
        if (Re_seg == 1)
                 p1 = p1+1;
                 b = 1;
                 VarA = [];
                 ContA = [];
                 nbSegments = 40;
%30 for infrared video , 2 for simulated videos
[movd_seg(p1).cdata,NcutDiscrete,NcutEigenvectors,
NcutEigenvalues,W,imageEdges]=
                    NcutImage(ReconsIm(k).cdata,nbSegments);
[dm MaskSeg dmV thresh] =
   MovedSegmentManual(movd_seg,movdr,movdr, k,
     sizeIm,nbSegments,p1);
%%*added ReconsIm
dm = dm - 1;
        end
```

```
    if ( DiffBoundary < BndryThres)
   [dm MaskSeg] =
          MovedSegmentManual(movd_seg,movdr,segm,
   Seg,k,sizeIm,nbSegments);
% Calculates the Segment that moved.
% and output is MaskSeg structure having all
%Segements that moved,
% dm is the count
%dm = dm - 1;
c = 0;
dmMov = [];
  if (DetChangeSeg == 1)
    [dc MaskSegCh dmCh threshCh] =
    MovedSegmentManual(movd_seg,movdr,movdr, k,
sizeIm,nbSegments,p1);
     for i=1:size(dmCh,2)
                  if(sum(dmCh(1,i)~=dmV)==size(dmV,2))
                        c = c + 1;
                        dmMov(1,c) = dmCh(1,i);
                  end
          end
end
CombineMask (1:dm+c) =
struct('ses',zeros(sizeIm,sizeIm,'double'), ...
'x',zeros(sizeIm,1,'double'),'y',zeros(sizeIm,1,'double'));
          if (c~=0)
     for i=1:c
```

```
                imx = find(dmMov(1,i) == dmCh);
        CombineMask(1,i).ses = MaskSegCh(1,imx).ses;
            CombineMask(1,i).x = MaskSegCh(1,imx).x;
            CombineMask(1,i).y = MaskSegCh(1,imx).y;
          end
end
    for i=c+1:dm+c
        CombineMask(i).ses = MaskSeg(i-c).ses;
            CombineMask(i).x = MaskSeg(i-c).x;
            CombineMask(i).y = MaskSeg(i-c).y;
          end
dmV = [dmMov dmV];
dm = size(dmV,2);
[MeasMaskAvg MeasOutAvg MaskSegAvg]=
MaskOutnIn(CombineMask,dm,sizeIm);
%Calculates In and outside mask for each segment
MeasAvgSeg = MeasMaskAvg * ReconsIm(k).cdata(:);
MeasAvgSeg1 = MeasMaskAvg * movdr(k+1).cdata(:);
MeasOutSeg = MeasOutAvg * ReconsIm(k).cdata(:);
MeasOutSeg1 = MeasOutAvg * movdr(k+1).cdata(:);
DiffOut = abs(MeasOutSeg1 - MeasOutSeg);
k1=1;
for u = 0:8:(dm*8)-8
DiffOut(k1:k1+7,1)=
DiffOut(k1:k1+7,1)./sum(DiffOut(k1:k1+7,1));
k1 = k1 + 8;
end
```

```
MotionMat = CalcMotionMatrix ( DiffOut, dm );
% Calculate the motion matrix
%%%%%%%%%%%%%%%NEW MEASUREMENT MASK%%%%%%%%%%%%%%%%%%%%%%%%
[NewMeasMat MaskSeg] =
DispMeasureMaskNew( MotionMat, CombineMask, MaskSegAvg,
movdr,sizeIm,dm,MeasOutSeg1,k);
          else if ( DiffBoundary > BndryThres)
           [dmB BorderMask]=MovedBorderSeg(movdr, movdr,
          movd_seg,sizeIm,nbSegments,k,RowThres,p1,thresh);
%%*added ReconsIm
% calculate the motion vector of these segments
% Calcualte the measurement matrix for these segments
% Make measurements for this area on the outer side
% Segements that moved,
% dm is the count
   dmbS = size(dmB,2)-1;
   c1=0;
   dmBor = [];
   for i=1:size(dmB,2)
       if(sum(dmB(1,i)~=dmV)==size(dmV,2))
       c1 = c1 + 1;
      dmBor(1,c1) = dmB(1,i);
   end
end
    dm = c1 + size(dmV,2);
    dmV = [dmBor dmV];
    c2 = 0;
```

```
    dmMov = [];
  if (DetChangeSeg == 1)
   [dc MaskSegCh dmCh threshCh] =
   MovedSegmentManual(movd_seg,movdr,movdr, k,
                                                    sizeIm,nbSegments,p1);
    for i=1:size(dmCh,2)
        if(sum(dmCh(1,i)~=dmV)==size(dmV,2))
          c2 = c2 + 1;
          dmMov(1,c2) = dmCh(1,i);
      end
    end
end
dmV = [dmMov dmV];
dm = size(dmV,2);
CombineMask (1:dm) =
struct('ses',zeros(sizeIm,sizeIm,'double'),
'x',zeros(sizeIm,1,'double'),'y',
zeros(sizeIm,1,'double'));
    if (c2~=0)
    for i=1:c2
     imx = find(dmMov(1,i) == dmCh);
     CombineMask(1,i).ses = MaskSegCh(1,imx).ses;
     CombineMask(1,i).x = MaskSegCh(1,imx).x;
     CombineMask(1,i).y = MaskSegCh(1,imx).y;
    end
   end
   if(c1~=0)
```

```
    for i=1+c2:c1+c2
        ibx = find(dmB==dmBor(1,i-c2));
     CombineMask(i).ses = BorderMask(ibx).ses;
     CombineMask(i).x = BorderMask(ibx).x;
     CombineMask(i).y = BorderMask(ibx).y;
    end
   end
    for i=c1+c2+1:dm
     CombineMask(i).ses = MaskSeg(i-(c1+c2)).ses;
     CombineMask(i).x = MaskSeg(i-(c1+c2)).x;
     CombineMask(i).y = MaskSeg(i-(c1+c2)).y;
    end
%%
[MeasMaskAvg MeasOutAvg MaskSegAvg] =
          MaskOutnIn(CombineMask,dm,sizeIm);
%Calculates In and outside mask for each segment
MeasAvgSeg = MeasMaskAvg * ReconsIm(k).cdata(:);
MeasAvgSeg1 = MeasMaskAvg * movdr(k+1).cdata(:);
MeasOutSeg = MeasOutAvg * ReconsIm(k).cdata(:);
MeasOutSeg1 = MeasOutAvg * movdr(k+1).cdata(:);

DiffOut = abs(MeasOutSeg1 - MeasOutSeg);
k1=1;
for u = 0:8:(dm*8)-8
     DiffOut(k1:k1+7,1)=DiffOut(k1:k1+7,1)
     ./sum(DiffOut(k1:k1+7,1));
    k1 = k1 + 8;
```

```
end
MotionMat = CalcMotionMatrix ( DiffOut, dm );
% Calculate the motion matrix
%%
%%%%%%%%%%%%%%%%%%%%NEW MEASUREMENT MASK %%%%%%%%%%%%%%%%
[NewMeasMat MaskSeg] =
DispMeasureMaskNew( MotionMat, CombineMask, MaskSegAvg,
ReconsIm, sizeIm,dm,MeasOutSeg1,k);
end
end
%       figure(1)
%       colormap(gray)
%       title 'Reconstructed Image using 36% Measurements'
%       colorbar
%       imagesc(NewMeasMat)
%
%% if(sum(sum(isnan(NewMeasMat)==0)))
     U = zeros(sizeIm,sizeIm);
    NoMeas = 0;
%%%%%%%%%%%MAKE MEASUREMENTS %%%%%%%%%%
 if(sum(sum(NewMeasMat)~=0))
   N = log10(sum(sum(NewMeasMat)));
  MaxIm = max(max(ReconsIm(k).cdata));
  deltaVal = 0.22;
  [r c] = find(ReconsIm(k).cdata.*NewMeasMat
     > (MaxIm - deltaVal));
  sparsty = size(r,1);
```

```
 C = 1.5;
 Err = .8;
 Nt = sum(sum(NewMeasMat));
 NoMeas = .4 * Nt;
%NoMeas = (N*C*sparsty)/Err;
upBound = 0.5;
LwBound = 0.3;
if (NoMeas > upBound*sum(sum(NewMeasMat)))
    NoMeas = upBound*sum(sum(NewMeasMat));
end
if (NoMeas < LwBound*sum(sum(NewMeasMat)))
     NoMeas = LwBound*sum(sum(NewMeasMat));
end
%NoMeas = floor(.70*size(r,1));%10*percen
NewMeasMat_ber = zeros(NoMeas,sizeIm*sizeIm);
for j = 1:NoMeas
     Ber = sqrt(6)*randn(sizeIm);
     NewMeasMat_ber(j,:) = (NewMeasMat(:).* Ber(:))';
end
Noise_var = .01;
regp = 4.5;
s2 = sizeIm*sizeIm;
NewMeasVector = NewMeasMat_ber * movdr(k+1).cdata(:);
% + Noise_var.*randn(NoMeas,1);
%reducedImage=
uint8((single(grayImage)/256)*2^integerValue);
%%%%%%%%%%%%%%%%%%%%%%%%RECONSTRUCTION%%%%%%%%%%%%%%%%%%%%%%%%
```

```
opts.mu = 2^8;
opts.beta = 2^5;
opts.tol = 1E-3;
opts.maxit = 300;
opts.TVnorm = 1;
opts.nonneg = true;
tic
[U, out] =
    TVAL3(NewMeasMat_ber,NewMeasVector,sizeIm,sizeIm,opts);
%%      cvx_begin
%%      variable x(s2);
%       minimize( norm(x,1) );
%       subject to
%       norm ((NewMeasMat_ber*x - NewMeasVector),2) <= regp;
%       cvx_end
% ReconsX = reshape(x,64,64);
%%%%%%%%%%%%%%%%%RECONSTRUCTION%%%%%%%%%%%%%%%%%%
t = toc;
sumt = t + sumt;
U = U.*NewMeasMat;
%U = U./max(max(U));
end
%if(sum(sum(NewMeasMat))==0)
%k
%end
temp =  ReconsIm(k).cdata;
[r,c] = find(NewMeasMat ~=0 );
```

```
for i=1:size(r,1)
  temp(r(i,1),c(i,1)) = U(r(i,1),c(i,1));
end
NoiseLevel =
sum(sum(sqrt(abs(temp.^2 - movdr(k+1).cdata.^2))))
      ./(sizeIm*sizeIm);
%Diff_Seq = zeros(1,6);
%for p=1:6
%  Diff_Seq(1,p) =
sum(sum(temp .* abs(Fix_Seq(p).data))) -
sum(sum(movdr(k+1).cdata .* abs(Fix_Seq(p).data)));
%end
%
%sum_diff = abs(mean(abs(Diff_Seq./(sizeIm*sizeIm))));
%ErrorThres = .01;
%if (sum_diff < ErrorThres )
%DetChangeSeg = 0;
%else
%DetChangeSeg = 1;
%end
DetChangeSeg = 1;
PercenCoverThres = .7;
% .1 for simulations
if (sum(sum(NewMeasMat)) < PercenCoverThres*sizeIm*sizeIm)
   Re_seg = 0;
else
  Re_seg = 1;
```

```
% DetChangeSeg = 0;
end
ReconsIm(k+1).cdata=temp;
MeasMatrix(k+1).cdata = NewMeasMat;
MeasMatrix(k+1).meas = NoMeas;
k = k + 1;
display(k)
figure(2)
colormap(gray)
title 'Reconstructed Image using 36% Measurements'
colorbar
imagesc(temp)
%pause
close all
clear NewMeasMat U out temp NewMeasMat_ber
      NewMeasVector Ber MotionMat DiffOut
      MeasAvgSeg MeasAvgSeg1 MeasOutSeg
      MeasOutSeg1 MeasMaskAvg MeasOutAvg
      MaskSegAvg BorderMask CombineMask
      VarA ContA Cont Var NoMeas
end
end
%%
%pause
% comparison(1:nFrames) = ...
% struct('cdata', zeros(sizeIm, 2*sizeIm, 1,
'double'),
```

```
'colormap', []);
% Video(1:nFrames)=struct('cdata', zeros(sizeIm,
2*sizeIm, 3, 'uint8'),'colormap',[]);
% Videocmp(1:nFrames)=struct('cdata', zeros(sizeIm,
sizeIm, 3,'uint8'),'colormap',[]);
%
% for i=1:nFrames
%Video(i).cdata(:,:,1)=ReconsIm(i).cdata.*255;
%Video(i).cdata(:,:,2)=ReconsIm(i).cdata.*255;
%Video(i).cdata(:,:,3)=ReconsIm(i).cdata.*255;
% end
%
% movie2avi(ReconsIm,'InsertFramesVid1','compression','None')
%
% for i=1:nFrames
%     comparison(i).cdata(:,1:64)= FullVid(i).cdata(:,:);
%     comparison(i).cdata(:,64+1:2*64)= movdr(i+1).cdata;
% end
%
% for i=1:nFrames
% Videocmp(i).cdata(:,:,1)=comparison(i).cdata.*255;
%Videocmp(i).cdata(:,:,2)=comparison(i).cdata.*255;
%Videocmp(i).cdata(:,:,3)=comparison(i).cdata.*255;
% end
% movie2avi(Video,'InsertFramesVid1','compression','None')
```

A.2 Function for Moved Segments identification

```
function [dm MaskSeg dmV Est_thresh]
        = MovedSegmentManual(movd_seg,movdr,
                             ReconsIm, k,sizeIm,nbSegments,p1)
dm = 1;
dmV = [];
MaskforMeas = zeros(sizeIm);
Est_thresh = 1;
Seg(1) = struct('Mean',zeros(nbSegments,1,1,'double'),
                'Var', zeros(nbSegments,1,'double'));
MaskSeg(1:nbSegments)=
struct('seg',zeros(sizeIm,sizeIm,'double'),'ses',
zeros(sizeIm,sizeIm,'double'),'x',zeros(sizeIm,1,'double'),
'y',zeros(sizeIm,1,'double'),'Rad',zeros(1,3,'double'));
diff = zeros(nbSegments,1); MaxHist = nbSegments;
for i=1:nbSegments
        [r,c] = find(movd_seg(1,p1).cdata == i);
        SegFirstFrameMean =
    mean(diag((ReconsIm(k).cdata(r,c)),0));
        %movdr changed to ReconsIm
        SegSecondframeMean =
    mean(diag((movdr(k+1).cdata(r,c)),0));
        %Averages from net frame
        diff(i,1) =
    abs(SegFirstFrameMean - SegSecondframeMean);
end

Est_thresh = .0030;   %Threshold for Foreman video = .0030
```

```
%Threshold for pony video is .0013
% Threshold for horse video is .0009
%Threshold for simulated video of puppet was .0009
%Threshold for donkey video is .001


  for j=1:nbSegments
%Calculates the Segment that moved. and output
%is MaskSeg structure having all Segements that
%moved, dm is the count
[r,c] = find(movd_seg(p1).cdata == j);
if ( size(r,1)> 5)
Seg(k).Mean(j,1) = mean(diag((ReconsIm(k).cdata(r,c)),0));
Seg(k).Var(j,1) = var(diag((ReconsIm(k).cdata(r,c)),0));
Seg(k+1).Mean(j,1) = mean(diag((movdr(k+1).cdata(r,c)),0));
Seg(k+1).Var(j,1)  = mean(diag((movdr(k+1).cdata(r,c)),0));
Maskdum = zeros(sizeIm,sizeIm);
if (abs(Seg(k).Mean(j,1)-Seg(k+1).Mean(j,1))
   > Est_thresh)
   for p=1:size(r)
       MaskforMeas(r(p,1),c(p,1)) = j./j;
       Maskdum(r(p,1),c(p,1)) = j./j;
   end
        end
   if (sum(sum(Maskdum)) ~= 0)
     MaskSeg(dm).seg = edge(Maskdum,0.01);
          MaskSeg(dm).ses = Maskdum;
     MaskSeg(dm).x = r;
```

```
        MaskSeg(dm).y = c;
            %MaskSeg(dm).x = r
        dm = dm + 1;
            %MaskSeg(dm).y = c
            fixed at later in the program after line 307
            dmV = [dmV j];
    end
      clear r c;
    end
  end
end
```

A.3 Function for Creating Mask

```
function [NewMeasMat MaskSegMoved] =
DispMeasureMaskNew( MotionMat, MaskSeg,MaskSegAvg, ReconsIm,
                                                sizeIm,dm,MeasOutSeg1,k)

NewMask(1:dm*8) = struct('seg', zeros(sizeIm,sizeIm,'double'));
NewAvgMeas = zeros(7,1);
diffAvgNew = zeros(7,1);
NewMeasMat = zeros(sizeIm);
NewMaskAvgseg = zeros(sizeIm);
NewMaskAvgout = zeros(sizeIm);
change = zeros(dm,1);
MaskSegMoved(1:dm)=struct('ses',zeros(sizeIm,sizeIm),
'x',zeros(sizeIm,1,'double'),'y',zeros(sizeIm,1,'double'));
%    NewTempMask(j).seg = zeros(sizeIm,sizeIm);
```

```
multx = 0;
multy = 0;
Temp1x = 0;
Temp1y = 0;
Temp2x = 0;
Temp2y = 0;
NewTempMask(1:dm) = struct('seg',zeros(sizeIm,sizeIm,'double'));
ns = 0;
TempM = zeros(sizeIm);
for j = 1:dm
    for d = 1:8
     %ns = ns + 1;
     nsc = int2str(d);
     temx = strcat('xd',nsc);
     temxm = strcat('xm',nsc);
     temy = strcat('yd',nsc);
     temym = strcat('ym',nsc);
     Temp1y = getfield(MotionMat,{1 j},temy); %#ok<GFLD>
    Temp2y = ceil(1*(getfield(MotionMat,{1 j},temym)));
    Temp1x = ceil(1*(getfield(MotionMat,{1 j},temx)));
    Temp2x = ceil(1*(getfield(MotionMat,{1 j},temxm)));
    if ( sum(Temp1y) ~= 0 || sum(Temp1x ~= 0))
        if ( d==1)
            IncR = 1;
            IncC = 0;
        end
        if ( d==2 )
```

```
        IncR = 1;
        IncC = 1;
    end
    if ( d == 3)
        IncR = 0;
        IncC = 1;
  end
  if(d == 4)
     IncR = -1;
     IncC = 1;
  end
  if(d == 5)
     IncR = -1;
     IncC = 0;
  end
  if(d == 6)
     IncR = -1;
     IncC = -1;
end
if( d== 7)
   IncR = 0;
   IncC = -1;
end
if (d == 8)
IncR = 1;
IncC = -1;
end
```

```
r = MaskSeg(j).x;
c = MaskSeg(j).y;
if ( IncR <=1 || IncC <=1)
    inc = 1;
    while ( inc <= 2)
      for i = 1:size(r,1)
          x = ceil(r(i,1)+ IncR*inc*Temp2x);
          y = ceil(c(i,1)+ IncC*inc*Temp2y);
          if (x>sizeIm)
              x=sizeIm;
         end
          if (y>sizeIm)      %Boundary Conditions
              y=sizeIm;
          end
          if (x <= 0)
              x=1;
          end
          if(y <= 0)
           y=1;
          end
     NewTempMask(j).seg(x,y) = MaskSeg(j).ses(r(i,1),c(i,1));
     end
     NewTempMask(j).seg = NewTempMask(j).seg(1:sizeIm,1:sizeIm);
          NewMaskAvgout = MaskSegAvg(8.*(j-1)+d).mask .*
          (1-NewTempMask(j).seg./max(max(NewTempMask(j).seg)));
    NewAvgMeas(inc,1) = sum(sum(NewMaskAvgout
    .* ReconsIm(k).cdata));
```

```
%%*movdr changed to ReconsIm
    diffAvgNew(inc,1) = abs(NewAvgMeas(inc,1)
    - MeasOutSeg1(8.*(j-1)+d));
  %figure;
    imagesc(30.*NewTempMask(j).seg+20.*MaskSegAvg(8.*(j-1)+d).mask)
    inc = inc + 1;
    NewMaskAvgout = zeros(sizeIm);
    NewTempMask(j).seg = zeros(sizeIm);
end
diffAvgNew==min(diffAvgNew)));
for i = 1:size(r,1)
     x = ceil(r(i,1)+ IncR*incF);
     y = ceil(c(i,1)+ IncC*incF);
     if (x>sizeIm)
        x=sizeIm;
     end
     if (y>sizeIm)
        y=sizeIm;
     end
     if (x <= 0)
        x=1;
     end
     if(y <= 0)
      y=1;
     end
     TempM(x,y) = MaskSeg(j).ses(r(i,1),c(i,1));
  end
```

```
  TempM = TempM(1:sizeIm,1:sizeIm);
 NewMask(j).seg = NewMask(j).seg + TempM;
 NewMask(j).seg = ceil(NewMask(j).seg./max(max(NewMask(j).seg)));
 TempM = zeros(sizeIm);
 NewTempMask(j).seg = zeros(sizeIm,sizeIm);
 diffAvgNew = zeros(7,1);
 multx = 0;
 multy = 0;
 Temp1x = 0;
 Temp1y = 0;
 Temp2x = 0;
 Temp2y = 0;
end
 change(j,1) = 1;
end
end
NewMask(j).seg(isnan(NewMask(j).seg)) = 0;
%NewMask(j).seg =  NewMask(j).seg./max(max( NewMask(j).seg));
Temp = NewMask(j).seg(1:sizeIm,1:sizeIm);
 if (k == 1)
    Lg1= logical(MaskSeg(j).ses);
   Lg2= logical(Temp);
   outL = Lg1 | Lg2;
   MaskSegMoved(j).ses = double(outL);
 else
   MaskSegMoved(j).ses = Temp;
 end
```

```
[r1,c1] = find(MaskSegMoved(j).ses == 1 );
MaskSegMoved(j).x = r1;
MaskSegMoved(j).y =c1;
NewMeasMat = NewMeasMat + Temp;
NewMeasMat = ceil(NewMeasMat./max(max(NewMeasMat)));
NewMeasMat(isnan(NewMeasMat))=0;
Temp = zeros(sizeIm);
end
for j=1:dm
    if(change(j,1) ~= 1)
     NewMeasMat = NewMeasMat + MaskSeg(j).ses;
     NewMeasMat = ceil(NewMeasMat./max(max(NewMeasMat)));
     MaskSegMoved(j).ses = MaskSeg(j).ses;
     [r1,c1] = find(MaskSegMoved(j).ses == 1 );
     MaskSegMoved(j).x = r1;
     MaskSegMoved(j).y =c1;
    end
end
```

A.4 Function for Calculating required averages

```
function [MeasMaskAvg MeasOutAvg MaskSegAvg] =

dem = 1;
segm = 1;
skip = 1;
MaskSegAvg(1:8*dm)=struct('seg',zeros(sizeIm,sizeIm,'double'),
'out',zeros(sizeIm,sizeIm,'double'),'mask',
```

```
zeros(sizeIm,sizeIm,'double'));
NewMaskAvg(1:8*dm)=struct('seg',zeros(sizeIm,sizeIm,'double'),
                          'out',zeros(sizeIm,sizeIm,'double'));
l = 0;
MeasMaskAvg = zeros(dm*8,sizeIm.^2);
MeasOutAvg = zeros(dm*8,sizeIm.^2);
for j = 1:dm              %Calculates the masks of directions
                      %Maks each segments into 8 separte
                  %directions
%
if (nargin == 4)
%if(size(find( j == dmB),2) ~= 0 )
%zero when not already done = true for condition
%skip = 0;
%one when already done
%end
%end
%
%if (skip ~= 0 || nargin == 3)
% Donot skip if skip is 1
c = MaskSeg(j).y;
r = MaskSeg(j).x;
if (size(c,1)~=0 && size(r,1)~=0)
    min_Xy = [c(min(find(r==min(r)))) min(r)];
    min_Yx = [min(c) r(min(find(c==min(c))))];
    max_Xy = [ c(max(find(r==max(r)))) max(r)];
% Finding Max and min for Segment i
```

```
 max_Yx = [ max(c) r(max(find(c==max(c))))];
 avgy=0;
 avgx=0;
point=0;
segm = max(max(MaskSeg(j).ses));
for ig=1:sizeIm
    for ih=1:sizeIm
        if (MaskSeg(j).ses(ig,ih) == segm)
            avgx = avgx + ig;
            avgy = avgy + ih;
            point = point + 1;
        end
     end
 end
 Cen(1,1) = avgx./point;
 Cen(1,2) = avgy./point;
 Radius =
[sqrt((Cen(1,1)-min_Xy(1,2)).^2+(Cen(1,2)-min_Xy(1,1)).^2)...
sqrt((Cen(1,1)-max_Xy(1,2)).^2+(Cen(1,2)-max_Xy(1,1)).^2)...
sqrt((Cen(1,1)-min_Yx(1,2)).^2+(Cen(1,2)-min_Yx(1,1)).^2)...
sqrt((Cen(1,1)-min_Yx(1,2)).^2+(Cen(1,2)-min_Yx(1,1)).^2)];
MaxRad = max(Radius);
MaskSeg(j).Rad(1,1) = MaxRad;
MaskSeg(j).Rad(1,2) = Cen(1,1);
MaskSeg(j).Rad(1,3) = Cen(1,2);
th = 0:pi/50:2*pi;
xunit = MaxRad * cos(th) + Cen(1,1);
```

```
yunit = MaxRad * sin(th) + Cen(1,2);
HorLinex1 = MaxRad * cos(0) + Cen(1,1);
HorLiney1 = MaxRad * sin(0) + Cen(1,2);
HorLinex2 = MaxRad * cos(th(1,50)) + Cen(1,1);
HorLiney2 = MaxRad * sin(th(1,50)) + Cen(1,2);
VerLinex1 = MaxRad * cos(th(1,25)) + Cen(1,1);
VerLiney1 = MaxRad * sin(th(1,25)) + Cen(1,2);
VerLinex2 = MaxRad * cos(th(1,75)) + Cen(1,1);
VerLiney2 = MaxRad * sin(th(1,75)) + Cen(1,2);
TiltLine1x1 = MaxRad * cos(th(1,14)) + Cen(1,1);
TiltLine1y1 = MaxRad * sin(th(1,14)) + Cen(1,2);
TiltLine2x1 = MaxRad * cos(th(1,39)) + Cen(1,1);
TiltLine2y1 = MaxRad * sin(th(1,39)) + Cen(1,2);
TiltLine1x2 = MaxRad * cos(th(1,64)) + Cen(1,1);
TiltLine1y2 = MaxRad * sin(th(1,64)) + Cen(1,2);
TiltLine2x2 = MaxRad * cos(th(1,89)) + Cen(1,1);
TiltLine2y2 = MaxRad * sin(th(1,89)) + Cen(1,2);
masks(1:8) = struct ( 'seg',zeros(sizeIm,sizeIm,'double'));
in=0;
fi = pi/4;
for f=1:8
    for th = in:pi/100:fi
        for ra = 0:.1:MaxRad+2
            x = floor(ra * cos(th) + MaskSeg(j).Rad(1,2));
           y = floor(ra * sin(th) + MaskSeg(j).Rad(1,3));
           if (x>sizeIm)
              x=sizeIm;
```

```
            end
            if (y>sizeIm)
                y=sizeIm;
            end
           if (x <= 0)
              x=1;
          end
          if(y <= 0)
           y=1;
          end
          masks(f).seg(x,y) = 1; %#ok<AGROW>
      end
    end
   in = fi;
   fi = in + pi/4;
%             masks(f).seg && MaskSeg(f).Seg
 MaskSegAvg(f+1).seg = MaskSeg(j).ses.* masks(f).seg;
 MaskSegAvg(f+1).out = masks(f).seg.*
(1-MaskSeg(j).ses./max(max(MaskSeg(j).ses)));
 MaskSegAvg(f+1).mask = masks(f).seg;
 end
 mh=1;
 for ma = 1:8
     if (sum(sum(MaskSegAvg(mh).seg)) ~= 0)
      MeasMaskAvg(mh+1,:) = MaskSegAvg(mh+1).seg(:);
      MeasOutAvg(mh+1,:) = MaskSegAvg(mh+1).out(:);
      mh = mh + 1;
```

```
    end
 end
mh=1;
l = l + 8;
dem = dem + 1;
%skip = 1;
end
end
```

A.5 Function for Border averages

```
function [dmB BorderMask] =
MovedBorderSeg(movdr, ReconsIm, movd_seg,sizeIm,nbSegments,
                                    k,RowThres,p1,threh)
dmB = [];
MaskforMeas = zeros(sizeIm);
BorderMask(1:nbSegments) =
        struct('ses',zeros(sizeIm,sizeIm,'double'), ...
'x',zeros(sizeIm,1,'double'),'y',zeros(sizeIm,1,'double'));

SegI(1:2) = ...
    struct('Mean',zeros(nbSegments,1,1,'double'));
Diff_FirstCol = sum(abs(ReconsIm(k).cdata(:,1)
                                    - movdr(k+1).cdata(:,1)));
Diff_LastCol  = sum(abs(ReconsIm(k).cdata(:,sizeIm)
                                    - movdr(k+1).cdata(:,sizeIm)));
Diff_FirstRow = sum(abs(ReconsIm(k).cdata(1,:)
                                    - movdr(k+1).cdata(1,:)));
```

```
Diff_LastRow  = sum(abs(ReconsIm(k).cdata(sizeIm,:)
                                    - movdr(k+1).cdata(sizeIm,:)));
mm = 1;
for d = 1:nbSegments
    [r,c] = find(movd_seg(p1).cdata == d);
  if(size(r,1)>5)
   SegI(1).Mean(d,1) = mean(diag((ReconsIm(k).cdata(r,c)),0));
  SegI(2).Mean(d,1) = mean(diag((movdr(k+1).cdata(r,c)),0));
  %Calcaulating averages of next frame
  tempF_col = find(c==1);
  tempF_row = find(r==1);
  tempL_col = find(c==sizeIm);
  tempL_row = find(r==sizeIm);
  if (size(tempF_col,1) == 0)
     C_condf = false;
  else
     C_condf = true;
  end
  if (size(tempF_row,1) == 0)
     R_condf = false;
  else
     R_condf = true;
  end
  if (size(tempL_col,1) == 0)
     C_condl = false;
  else
     C_condl = true;
```

```
    end
    if (size(tempL_row,1) == 0)
      R_condl = false;
    else
      R_condl = true;
    end
  if( Diff_FirstCol > RowThres && C_condf &&
  abs(SegI(1).Mean(d,1)-SegI(2).Mean(d,1)) > threh)
    for p=1:size(r)
      BorderMask(mm).ses(r(p,1),c(p,1)) =
      movd_seg(p1).cdata(r(p,1),c(p,1))./d;
  %Orignal Segmentation
    end
    BorderMask(mm).x = r;
    BorderMask(mm).y = c;
    mm = mm + 1;
    dmB = [dmB d];
   else if ( Diff_LastCol > RowThres && C_condl
   && abs(SegI(1).Mean(d,1)-SegI(2).Mean(d,1)) > threh)
   %enable all segments along this Col
      for p=1:size(r)
       BorderMask(mm).ses(r(p,1),c(p,1)) =
            movd_seg(p1).cdata(r(p,1),c(p,1))./d;
  %Orignal Segmentation
      end
      BorderMask(mm).x = r;
      BorderMask(mm).y = c;
```

```
    mm = mm + 1;
    dmB = [dmB d];
  else if ( Diff_FirstRow > RowThres && R_condf
 && abs(SegI(1).Mean(d,1)-SegI(2).Mean(d,1)) > threh)
%enable all segments along this row
    for p=1:size(r)
   BorderMask(mm).ses(r(p,1),c(p,1)) =
                                                movd_seg(p1).cdata(r(p,1
%Orignal Segmentation
end
BorderMask(mm).x = r;
BorderMask(mm).y = c;
mm = mm + 1;
dmB = [dmB d];
 else if ( Diff_LastRow > RowThres &&
   R_condl && abs(SegI(1).Mean(d,1)-SegI(2).Mean(d,1)) > threh)
%enable all segments along this row
 for p=1:size(r)
  BorderMask(mm).ses(r(p,1),c(p,1)) =
                          movd_seg(p1).cdata(r(p,1),c(p,1))./d;
%Original Segmentation
 end
 BorderMask(mm).x = r;
 BorderMask(mm).y = c;
mm = mm + 1;
dmB = [dmB d];
 end
```

```
end
end
end
end
end
% Calculate the average of consolidated segment Mask
% Keep segments above the threshold
```

A.6 Function for optimal segment number

```
%intercluster and intracluster ratio [FinalVar FinalCon]
function [Cont FinalVar] =
InterandIntra(ReconsIm, movd_seg, nbSegments, sizeIm,p1,k)
%similarity measure is Euclidean distance between
%average of the cluster and each pixel
Maskdum = zeros(sizeIm,sizeIm);
MaskSeg(1:nbSegments)=     struct('ses',
      zeros(sizeIm,sizeIm,'double'),'seg',
      zeros(sizeIm,sizeIm,'double'),'x',
      zeros(sizeIm,1,'double'),'y',
      zeros(sizeIm,1,'double'),'AvgDis',
      zeros(sizeIm,1,'double'),'contrast',
      zeros(1,1,'double'),'var',
      zeros(1,1,'double'),'variance',
      zeros(1,1,'double'),'mean',
      zeros(1,1,'double'));

NeighSeg(1:nbSegments)=
```

```
struct('cdata',zeros(sizeIm,sizeIm,'double'),
'NoSeg',zeros(sizeIm,1,'double'),distoutseg',
zeros(1,1,'double'),'Len',zeros(1,1,'double'));
segm = 1;
for j=1:nbSegments
      % Calculates the nonzero indices of each segment
      % Segements that moved, dm is the count
     [r,c] = find(movd_seg(p1).cdata == j);
     if ( sum(r)~=0)
    AvgofSeg = mean(diag((ReconsIm(k).cdata(r,c)),0));
    distAvgInClus = zeros(size(r,1),1);
   for p=1:size(r,1)
    Maskdum(r(p,1),c(p,1)) = ReconsIm(k).cdata(r(p,1),c(p,1));
   MaskSeg(j).seg(r(p,1),c(p,1)) = segm;
   distAvgInClus(p,1) = sqrt(abs(Maskdum(r(p,1),c(p,1)).^2-
   AvgofSeg.^2));
 end
   MaskSeg(j).ses = Maskdum;
   MaskSeg(j).seg = edge(MaskSeg(j).seg,.01);
   MaskSeg(j).x = r;
   MaskSeg(j).y = c;
   MaskSeg(j).AvgDis = distAvgInClus;
   segm = segm + 1;
   Maskdum = zeros(sizeIm,sizeIm);
   end
end
 for j=1:nbSegments
```

```
      u = 1;
      [r,c] = find(MaskSeg(j).seg == 1);
      for l=1:nbSegments
       InSec =(MaskSeg(l).seg + MaskSeg(j).seg);
      if( j~=l && logical(sum(find(InSec == 2))))
       NeighSeg(j).cdata = NeighSeg(j).cdata
                                    + MaskSeg(l).ses;
       NeighSeg(j).NoSeg(u,1) = l;
       u = u + 1;
    end
    NeighSeg(j).Len = u-1;
%NeighSeg(j).NoSeg = NeighSeg(j).NoSeg(1:u,1);
   InSec = zeros(sizeIm,sizeIm);
 end
end
%Ratio of intercluster and intracluster similarity measure
for j=1:nbSegments
    l = NeighSeg(j).Len;
% Selecting the main segment
    r = MaskSeg(j).x;
    c = MaskSeg(j).y;
   if(sum(r)~=0)
   RatioInOut = zeros(1,l);
   distAvgInClus = MaskSeg(j).AvgDis;
   [m n] = find(MaskSeg(j).seg == 1);
%       for d=1:size(m,1)
%           pm = m(d,1);
```

```
%           pn = m(d,1)-1;
%
%            if (pm > sizeIm)
%                pm = sizeIm;
%            end
%            if (pn>sizeIm)
%Boundary Conditions
%                pn = sizeIm;
%            end
%            if (pm <= 0)
%               pm = 1;
%            end
%            if(pn <= 0)
%               pn = 1;
%            end
%
%
%MaskSeg(j).contrast = MaskSeg(j).contrast
          + abs(movdr(1).cdata(pm,n(d,1))
      -movdr(1).cdata(pn,n(d,1)));
%end
%
%       if size(m,1)~=0
%       MaskSeg(j).contrast = MaskSeg(j).contrast./size(m,1);
%       end
%       MaskSeg(j).var = mean(distAvgInClus);
MaskSeg(j).variance = var(diag((ReconsIm(k).cdata(r,c)),0));
```

```
MaskSeg(j).mean = mean(diag((ReconsIm(k).cdata(r,c)),0));
      if(l~=0)
        for p=1:l
          Contrast = 0;
          idx = NeighSeg(j).NoSeg(p,1);
          %Selecting the 1st Neighbour and so on
          v = MaskSeg(idx).x;
          w = MaskSeg(idx).y;
MeanOutClus = mean(diag((ReconsIm(k).cdata(v,w)),0));
VarOutClus = var(diag(ReconsIm(k).cdata(v,w),0));
Num = sqrt((MaskSeg(j).mean - MeanOutClus).^2+
                  (MaskSeg(j).variance - VarOutClus).^2);
den  = sqrt((MaskSeg(j).mean + MeanOutClus).^2+
                  (MaskSeg(j).variance + VarOutClus).^2);
if (den ~= 0)
   Contrast = Num./den;
end
RatioInOut(1,p) = Contrast;
%choose average if max is too much precise
end
%       Ratio between var of current and neighbour to the
%       differnce between averages
%                 NeighSeg(j).distoutseg = max(RatioInOut);
end
end
end
```

```
SortedVar = zeros(nbSegments,1);
%% SortedCon = zeros(nbSegments,1);
%%  for i=1:nbSegments
    SortedVar(i,1) = MaskSeg(i).variance;
%%      SortedCon(i,1) = MaskSeg(i).contrast
end
%%
%% SortedVar = sort(SortedVar,'ascend');
%% SortedCon = sort(SortedCon,'descend');
%%  %% percenV = floor(.7*nbSegments);
FinalVar = sum(SortedVar)./nbSegments;
%% percenC = floor(.5*nbSegments);
%% FinalCon = sum(SortedCon(1:percenC))./percenC;
% maxR = NeighSeg(1).distoutseg;
%% for j=1:nbSegments
%%
%%    if (maxR < NeighSeg(j).distoutseg)
%%          maxR = NeighSeg(j).distoutseg;
%%      end
%% end
meanRatio = 0;
for i =1:nbSegments
    meanRatio = meanRatio + (NeighSeg(i).distoutseg);
end
Cont = meanRatio./nbSegments;
```

A.7 Function for Results

```
MSE = zeros(k,1);
NonZeroMeas = zeros(k,1);
spatMeas = zeros(k,1);
temporalMeas = 0;
temporalComp = zeros(k,1);
spatComp = zeros(k,1);
PSNRim=0;
FramePSNR = zeros(k,1);
        for i = 1:k
        NonZeroMeas(i,1) = sum(sum(MeasMatrix(i).cdata));
        spatMeas(i,1) = MeasMatrix(i).meas;
        if (NonZeroMeas(i,1)~=0)
        spatComp(i,1) = (
    NonZeroMeas(i,1)-spatMeas(i,1))./NonZeroMeas(i,1);
        else
        spatComp(i,1) = 0;
        end
        temporalComp(i,1) = (4096-NonZeroMeas(i,1))./4096;
        temporalMeas = NonZeroMeas(i,1) + temporalMeas;
        MSE(i,1) =
    mean(mean((255.*movdr(i).cdata - 255.*
                                ReconsIm(i).cdata).^2));
        FramePSNR(i,1)=
    10*log10((255.*max(max(ReconsIm(i).cdata))).^2
                                                /MSE(i,1));
        PSNRim =
    10*log10((255.*max(max(ReconsIm(i).cdata))).^2
```

```
                                        /MSE(i,1))+PSNRim;
        end
AvgTemporalComp = sum(temporalComp)./k;
temporalMeas = temporalMeas./k;
spatMeasAvg = sum(spatMeas)./k;
TempCompAvg = sum(temporalComp)./k;
MSEAvg = sum(MSE)./k;
PSNRim = PSNRim/k spatCompAvg = sum(spatComp)./k
load('movingletterANDintelligentroomRESULT')
for i=1:15
        NoSamp(i,1)=IntRoom(1,i).NoSamp;
 PerCov(i,1)=IntRoom(1,i).PerCov;
ReSeg(i,1)=IntRoom(1,i).ReSeg;
PSNR(i,1)=IntRoom(1,i).psnr;
end
plot(100.*PerCov(1:13),300./ReSeg(1:13))
plot(100.*PerCov(1:13),PSNR(1:13))
plot(100.*PerCov(1:13),NoSamp(1:13))
for i=1:14
 NoSamp(i,1)=Result(1,i).NoSamp;
 PerCov(i,1)=Result(1,i).PerCov;
 ReSeg(i,1)=Result(1,i).ReSeg;
 PSNR(i,1)=Result(1,i).psnr;
end
plot(100.*PerCov(1:13),NoSamp(1:13))
plot(100.*PerCov(1:13),PSNR(1:13))
plot(100.*PerCov(1:13),100./ReSeg(1:13))
```