Image Reconstruction, Classification, and Tracking for Compressed Sensing Imaging and Video

by

Henry Braun

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved November 2016 by the
Graduate Supervisory Committee:

Pavan Turaga, Co-Chair
Andreas Spanias, Co-Chair
Cihan Tepedelenlioglu
Visar Berisha

ARIZONA STATE UNIVERSITY

December 2016

ABSTRACT

Compressed sensing (CS) is a novel approach to collecting and analyzing data of all types. By exploiting prior knowledge of the compressibility of many naturally-occurring signals, specially designed sensors can dramatically undersample the data of interest and still achieve high performance. However, the generated data are pseudorandomly mixed and must be processed before use. In this work, a model of a single-pixel compressive video camera is used to explore the problems of performing inference based on these undersampled measurements. Three broad types of inference from CS measurements are considered: recovery of video frames, target tracking, and object classification/detection. Potential applications include automated surveillance, autonomous navigation, and medical imaging and diagnosis.

Recovery of CS video frames is far more complex than still images, which are known to be (approximately) sparse in a linear basis such as the discrete cosine transform. By combining sparsity of individual frames with an optical flow-based model of inter-frame dependence, the perceptual quality and peak signal to noise ratio (PSNR) of reconstructed frames is improved. The efficacy of this approach is demonstrated for the cases of *a priori* known image motion and unknown but constant image-wide motion.

Although video sequences can be reconstructed from CS measurements, the process is computationally costly. In autonomous systems, this reconstruction step is unnecessary if higher-level conclusions can be drawn directly from the CS data. A tracking algorithm is described and evaluated which can hold target vehicles at very high levels of compression where reconstruction of video frames fails. The algorithm performs tracking by detection using a particle filter with likelihood given by a maximum average correlation height (MACH) target template model.

Motivated by possible improvements over the MACH filter-based likelihood estimation of the tracking algorithm, the application of deep learning models to detection and classification of compressively sensed images is explored. In tests, a Deep Boltzmann Machine trained on CS measurements outperforms a naive reconstruct-first approach.

Taken together, progress in these three areas of CS inference has the potential to lower system cost and improve performance, opening up new applications of CS video cameras.

I would like to thank Rachel Olzer for making me coffee most mornings for the past few months.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

In this dissertation, I present work at the intersection of compressed sensing (CS), image processing, and computer vision. Using the single-pixel camera [1] as a model, I develop and improve inference algorithms for CS data. Three broad categories of inference problems are considered: reconstruction of video sequences, video-based target tracking, and image-based detection/classification. I improve both the accuracy and precision of the inferred quantities and the computational cost of making the inference.

CS is a relatively new paradigm in sensor and signal processing which combines specially-designed sensors with *a priori* knowledge of a signal's compressibility. By exploiting this knowledge, a signal of interest can be recovered from an extremely underdetermined set of measurements [2]. The potential benefits are clear: compressive sensing technology, once commercialized, is likely to result in lower data rate requirements and less costly sensing hardware. However, these advantages come at the cost of higher computational requirements, as measurements from CS sensors must be processed to reconstruct the signal of interest.

This work focuses on the single-pixel camera as a specific case of CS. The single-pixel camera, in its simplest embodiment, displays a pseudo-random measurement pattern on a digital micromirror device. The image to be sensed is projected on this micromirror and the reflected light is captured by a photodiode, resulting in an analog implementation of the inner, or "dot", product operation. By rapidly switching through a known set of measurement patterns, a vector of these inner products with the image is assembled.

From a set of CS measurements, a reconstruction algorithm must be used to estimate the image. Natural images are known to be compressible - for instance using the discrete cosine or wavelet transforms. The transformed images contain most of the signal energy in a very small fraction of the components, with most of the components having near zero magnitude. The CS reconstruction procedure identifies and estimates these large components, allowing the image to be recovered from the single-pixel's camera's measurements. Several variations on this theme exist, but all rely on exploiting prior knowledge of a signal's *sparsity* in a transformed domain. The number of measurements required to accurately reconstruct the signal is larger than the number of nonzero components of the transformed signal; additional measurements are required to identify which indices are non-zero. The total number of measurements, though, is far less than the total length of the signal to be reconstructed.

1.1 Targeted Applications

This work targets automated systems which may benefit from the use of CS imaging. The problem of infrared is specifically targeted. Surveillance is one such application: by adopting computationally simple object recognition algorithms, the cost of IR video monitoring can be reduced. This reduction comes partially through lower computational demands, but also allows data transmission to be dramatically reduced. If the relevance of data can be determined at the sensor, only relevant data must be transmitted to a central location.

Another application is autonomous vehicles, especially unmanned aerial vehicles (UAVs). More effective video reconstruction algorithms for UAV applications allow higher performance or lower sensor cost. By reducing the computational, and therefore monetary, cost of a UAV's situational awareness, new applications for automomous UAVs become feasible.

1.2 Problem Statement

CS reconstruction of video sequences is less straightforward than for still images. Individual frames are compressible in a linear basis, but statistical dependence between frames is not as easily described. Even an relatively simple model of inter-frame image motion is bilinear rather than linear in the image intensity estimated motion vectors [3]. Non-CS video compression algorithms divide video sequences into Intra-coded *I frames* and predicted *P frames*, but extending this approach to CS is non-trivial since we have no access to the images from which predicted motion is estimated. Exploiting the dependence between frames improves video reconstruction performance, but finding the best possible method remains an active area of research.

Recovery of signals from CS measurements is computationally demanding. However, in some cases, such as automated surveillance or navigation, a system may not require a human to observe a video at all. In these cases it may be possible to perform target recognition and other computer vision tasks directly on the compressively sensed data, without reconstructing video frames. This is a hard problem, since the mixing effect of the compressed sensing operation forces the researcher to abandon principles, such as shift invariance, on which conventional algorithms rely heavily.

Consider the problem of tracking an object in a video sequence. This is already a difficult computer vision task without the addition of CS. Targets change scale as they get closer and further from the sensor. Light conditions change as they move in and out of shadows; reflections and glare cause sudden changes in the perceived color of the target's surface. Non-rigid objects change not just their projection into the image plane, but their three dimensional shape as well. None of these effects can be fully predicted from a single image of the target.

CS-based tracking adds to these difficulties, leaving a smaller set of tools. In-plane translation and rotation, normally relatively easily modeled, become much more difficult. Given a CS measurement vector of an image containing an object, we cannot even predict how the measurements would change if the object moves one pixel to the left. Doing so would require reconstructing the image, defeating our purpose of speeding up computation by avoiding reconstruction. A region of interest cannot be easily cropped out of an image either; every CS measurement encodes information about the whole image. On the other hand, some operations are still available for use. Vector lengths, Euclidean distances, and inner products are all preserved under the compressed sensing operation, albeit with increased noise. A reconstruction-free computer vision algorithm must use this reduced set of tools.

1.3 Contributions of This Work

This work focuses on the problems of reconstruction of CS video, reconstruction-free object tracking, and reconstruction-free image classification and detection. In video reconstruction, a method for incorporating frame-to-frame translational motion into reconstruction algorithms is studied. Moving to the tracking problem, an algorithm is developed and evaluated to track vehicles in surveillance video. This tracking algorithm depends heavily on a target likelihood estimation step, leading to the study of alternative approaches to classification and detection.

In cases where CS reconstruction must be performed, the problem of fully exploiting knowledge of signal statistics remains open. CS theory gives a method for reconstruction when the signal of interest has a sparse representation in some known linear basis, but video does not naturally fit this model. In Chapter 4, I examine the feasibility of an optical flow-based approach to multi-frame video reconstruction. A method for incorporating known frame-to-frame motion into a joint two-frame recon-

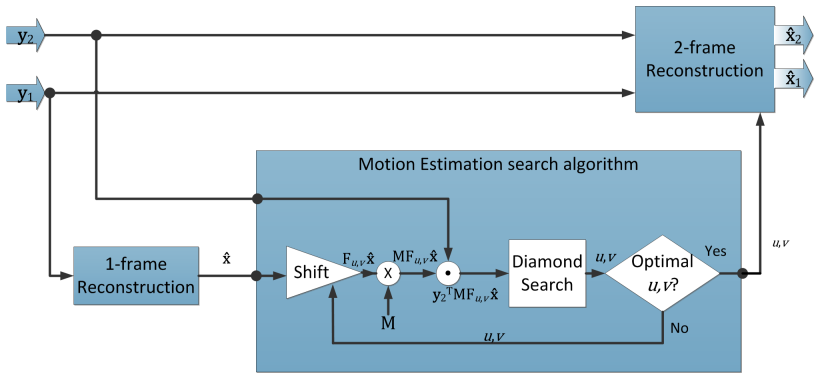


Figure 1.1: Reconstruction Algorithm Incorporating Optical Flow.

struction is proposed and evaluated on the Middlebury optical flow dataset [4]. This model is then extended to unknown but constant motion and evaluated on synthetic data from the PETS2000 dataset [5]. In both cases, the proposed method outperforms independent reconstruction of individual frames.

The reconstruction-free tracking algorithm of Chapter 5 [6] is based on a particle filter [7, 8] which maintains an estimate of the target's state and updates it with every frame of video as new measurements become available. This update is performed using a likelihood estimation step given by estimated cross-correlation using a maximum average correlation height (MACH) filter [9]. Correlation estimates are performed using the fast smashed filter approach [10], which allows cross-correlation to be estimated efficiently from CS measurements using the FFT [6]. Figure 1.2 gives a block diagram representation of this system. The algorithm is tested on moving vehicles from the PETS2000 and CDNET 2012 [11] datasets, and testing results are presented. When MACH filters with high peak height are possible, the algorithm can outperform a reconstruct-first approach at high levels of compression.

The tracking algorithm, although successful, relies on a very simple linear correlation filter for target detection. A more sophisticated approach taking cues from modern machine vision may be able to significantly outperform it. In Chapter 6 I

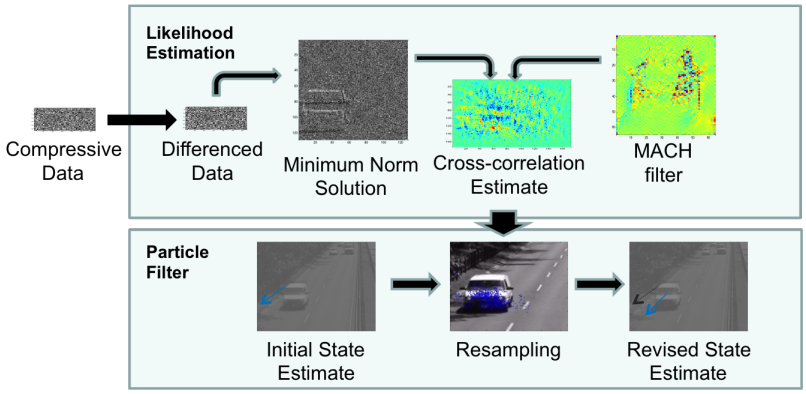


Figure 1.2: High-Level Block Diagram of Compressive Tracker

present one such case, a deep Boltzmann machine (DBM) algorithm. The DBM is a deep neural network architecture which attempts to model the probability distribution of a training dataset and use that model to classify new data. By training a base network on non-CS data and then fine-tuning it using compressive measurements of the same data, a network is developed which performs classification on CS measurements at computational cost similar to conventional image data. The performance of this CS-DBM network on the MNIST handwritten digit dataset is evaluated under varying levels of compression. Classification error rate increases with increasing compression, but the CS-DBM outperforms a procedure which reconstructs images before classification.

1.4 Organization

The thesis is organized as follows. First, Chapter 2 presents an overview of CS terminology and some important results on the accuracy of reconstruction algorithms. Hardware implementations of CS imaging sensors are also described, including the single-pixel camera considered in this work. An overview of several CS reconstruction algorithms is given, and their performance is compared. Challenges specific to video reconstruction are identified and works which attempt to address them are

discussed. Chapter 3 discusses existing work in image and video processing and computer vision. Compression by transforms and motion estimation and optical flow are described. Several computer vision algorithms for detection and classification of images are discussed. The problem of visual object tracking is described and some algorithms for tracking and state estimation are presented.

Chapters 4, 5, and 6 detail original contributions of this dissertation. Chapter 4 covers the video reconstruction algorithm. The optimization problem of video reconstruction incorporating optical flow is described. A diamond search procedure for estimating image motion from CS measurements is detailed. The algorithms are tested on the Middlebury and PETS2000 datasets. In Chapter 5 the tracking algorithm is discussed. Chapter 6 describes the CS-DBM classifier. Finally, the significance of the results is discussed in Chapter 7.

Chapter 2

BACKGROUND AND LITERATURE REVIEW OF COMPRESSED SENSING

The work presented in this thesis lies at the intersection of compressed sensing and computer vision. I attempt to separate these two areas by devoting this chapter to CS theory and literature and Chapter 3 to related work in image processing and computer vision. I begin by introducing the CS sensing matrix, ℓ_1 reconstruction, and other fundamental background information in Section 2.1. Next, Section 2.4 discusses existing CS imaging hardware in order to provide real-life motivation for the work. CS reconstruction algorithms are then discussed in Section 2.5 and bounds on CS sensing performance are covered in Section 2.7.

2.1 Foundations of Compressed Sensing

In this section, the background theory of compressive sensing is described. Terminology and symbols used throughout the work are introduced and their significance is explained. In essence, compressive sensing is the use of prior knowledge of a signal's statistics to reduce the number of measurements required to recover the signal. The use of this prior knowledge allows the Nyquist limit to be effectively "broken". CS reconstruction is concerned with the recovery of a compressible signal of interest $\mathbf{x} \in \mathbb{R}^{N\times 1}$ from a related measurement vector $\mathbf{y} \in \mathbb{R}^{M\times 1}$. $\mathbf{x}$ and $\mathbf{y}$ are related by a linear transformation

$$\mathbf{y} = \Phi\mathbf{x}. + \mathbf{w}. \tag{2.1}$$

where $\mathbf{w} \in \mathbb{R}^{M\times 1}$ is a noise vector and $\Phi \in \mathbb{R}^{M\times N}$ is known as the sensing matrix.

If $M < N$ (i.e. $\mathbf{y}$ is shorter than $\mathbf{x}$), the system in (2.1) is under-determined and x cannot be recovered with certainty from y even in the noiseless ($\mathbf{w} = \mathbf{0}$) case. However, let $\mathbf{x}$ be k-sparse (at most k non-zero components) with respect to some known basis with inverse transform matrix Ψ. That is,

$$\mathbf{y} = \Phi\mathbf{x} + \mathbf{w} = \Phi\Psi\boldsymbol{\theta} + \mathbf{w}, \tag{2.2}$$

$$\|\boldsymbol{\theta}\|_0 \leq k. \tag{2.3}$$

where $\boldsymbol{\theta}$ is the transformed representation of $\mathbf{x}$. Here the ℓ_0 norm denotes the number of non-zero elements. Strictly speaking, this is not a norm as it breaks the triangle inequality. However, this mild abuse of notation is common in the literature and will be used in the remainder of this document. The basis matrix Ψ should have full row rank ($\text{rank}(\Phi) = N$), but need not be square. Overcomplete bases are commonly used in several applications [12, 13]. Let $\mathrm{A} = \Phi\Psi$ be the combined sensing operator; this notation is used to simplify equations in which the two appear together.

If the conditions in (2.2) and (2.3) are met, CS theory dictates that it can be recovered with high probability as long as

$$(1 - \delta_k)\|\boldsymbol{\theta}\|_2 \leq \|\mathrm{A}\boldsymbol{\theta}\|_2 \leq (1 + \delta_k)\|\boldsymbol{\theta}\|_2 \tag{2.4}$$

for all k-sparse $\mathbf{x}$, with sufficiently small δ_k. (2.4) is known as the restricted isometry property (RIP) [14].

Several pseudorandom sensing matrices satisfy the RIP with high probability, including an i.i.d. Gaussian matrix [15]. Other matrices which satisfy the RIP include the Fourier and Hadamard ensemble [16] and the random orthoprojector [10]. The random orthoprojector is noteworthy as finding its pseudoinverse is computationally trivial ($\Phi^\dagger = \Phi^T(\Phi\Phi^T)^{-1} = \Phi^T$), a desirable property for some algorithms. In [17] an attempt is made to develop a deterministic sensing matrix, but the given approach does not fully satisfy the RIP.

Few naturally occurring signals are perfectly sparse, but many are known to be compressible; I here focus on natural images and video sequences. The JPEG image compression algorithm relies on the observation that the discrete cosine transform (DCT) of a natural image results in a transform-domain signal whose coefficients follow a power law distribution, with signal energy concentrated in a small number of components and the remainder near zero. The JPEG2000 standard similarly relies on the discrete wavelet transform for compression. In [18], discussed in Section 2.3.2, a bound is given for reconstruction of signals which are not perfectly sparse.

If the RIP is satisfied, $\boldsymbol{\theta}$, and therefore $\mathbf{x}$, can be recovered in the noiseless case from y by solving the ℓ_0 optimization problem

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\text{minimize}} \quad & \|\boldsymbol{\theta}\|_o \\ \text{subject to} \quad & \mathbf{y} - \mathrm{A}\boldsymbol{\theta} = 0. \end{aligned} \tag{2.5}$$

Unfortunately this optimization is nonconvex and cannot be solved in polynomial time. However, the convex relaxation of this problem is much more easily solved. It is given by

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\text{minimize}} \quad & \|\boldsymbol{\theta}\|_1 \\ \text{subject to} \quad & \mathbf{y} - \mathrm{A}\boldsymbol{\theta} = 0. \end{aligned} \tag{2.6}$$

(2.6) replaces the ℓ_0 norm with the ℓ_1 norm. This problem is known as basis pursuit (BP). It is convex and can be solved with conventional convex optimization techniques.

2.2 Related Problems

The basis pursuit reconstruction problem described in Section 2.1 is only the simplest of many related problems. In this section extensions and related problems are

described. Several other convex optimizations exist; a selection of these are covered in Section 2.3. Model-based CS, discussed in Section 2.3.2, improves performance by taking advantage of additional information not captured by a sparse prior. Blind CS, discussed in Section 2.3.3, treats the case where the sensing matrix is unknown or incompletely known.

2.3 Convex Optimizations

BP is valid in the noiseless case when $\mathbf{y} = \Phi\Psi\boldsymbol{\theta}$ must be satisfied exactly. Other related convex algorithms for noisy data include basis pursuit denoising (BPDN) and lasso. BPDN relaxes the equality constraint in (2.6), replacing it with an ℓ_2 term in the optimization which penalizes deviation from $\mathbf{y} = \Phi\Psi\boldsymbol{\theta}$. The BPDN problem is given by

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \left\|\mathbf{y} - \mathrm{A}\boldsymbol{\theta}\right\|_2 + \tau \left\|\boldsymbol{\theta}\right\|_1 \tag{2.7}$$

where τ is a tuning parameter reflecting the trade-off between encouraging sparsity and promoting a close match between the estimate and measured data. Intuitively, if measurements are noisy or there is a strong prior knowledge of sparsity of $\boldsymbol{\theta}$, a larger τ will be chosen. Conversely, if measurement noise is small or $\boldsymbol{\theta}$ is not known to be highly sparse, a smaller τ will be chosen. In an AWGN with known signal and noise power, τ may be optimized as in (2.17), discussed in Section 2.3.1. In practice, however, the solution is not highly sensitive to the choice of τ.

In the lasso algorithm, the objective function and constraint are reversed relative to BP. Where the BP algorithm seeks the sparsest possible $\boldsymbol{\theta}$ given the data, lasso seeks the closest possible match with the data subject to a sparsity constraint. Lasso

solves the optimization problem

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\text{minimize}} \quad & \|\mathbf{y} - \mathrm{A}\boldsymbol{\theta}\|_2 \\ \text{subject to} \quad & \|\boldsymbol{\theta}\|_1 \leq \epsilon_1 \end{aligned} \tag{2.8}$$

where ϵ_1 is a tuning parameter controlling the trade-off between sparsity of $\boldsymbol{\theta}$ and a close match to measurements $\mathbf{y}$.

With the appropriate choice of tuning parameters, all three problems become equivalent. In the limit of (2.7) as $\tau \to 0$, the solution of (2.6) is recovered and BPDN and BP are equivalent. Likewise, for every τ in (2.7), some ϵ_1 exists which results in the same solution to (2.8).

2.3.1 CS Reconstruction as Bayesian Inference

In Section 2.1, convex reconstruction based on the ℓ_1 norm was presented as a relaxation of a non-convex ℓ_0 problem. An alternative view of the problem [19] as maximum *a posteriori* (MAP) estimation under a Laplace prior is described here. This formulation will be useful in discussion of graphical reconstruction methods in Section 2.5.2. Let $\boldsymbol{\theta}$ be distributed i.i.d. Laplace with zero mean and scale parameter b. Its PDF is given as

$$p(\boldsymbol{\theta}) = \prod_{n=1}^{N} \frac{1}{2b} \exp\left(-\frac{|\theta_n|}{2b}\right) = \left(\frac{1}{2b}\right)^N \exp\left(-\frac{\|\boldsymbol{\theta}\|_1}{2b}\right). \tag{2.9}$$

The choice of a Laplace distribution for $p(\boldsymbol{\theta})$ is motivated by its history as a sparseness-inducing prior [20]; it is clearly a more informative prior than the Gaussian used in least squares.

$\mathbf{y}$ is then modeled as in (2.2) as a multiplication with the sensing matrix plus additive i.i.d. Gaussian noise of variance σ_w^2. The conditional PDF of $\mathbf{y}$ is given as

$$p(\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{\left(2\pi\sigma_w^2\right)^{\frac{M}{2}}} \exp\left(-\frac{1}{2\sigma_w^2} \|\mathbf{y} - \Phi\Psi\boldsymbol{\theta}\|_2\right). \tag{2.10}$$

The posterior PDF is then derived using Bayes' rule as follows:

$$p(\boldsymbol{\theta}|\mathbf{y}) = \frac{p(\mathbf{y}|\theta)p(\boldsymbol{\theta})}{p(\mathbf{y})} \tag{2.11}$$

$$= \frac{1}{p(\mathbf{y})}\left(\frac{1}{2b}\right)^N \frac{1}{(2\pi\sigma_w^2)^{\frac{M}{2}}} \exp\left(-\frac{1}{2\sigma_w^2}\left\|\mathbf{y}-\Phi\Psi\boldsymbol{\theta}\right\|_2 - \frac{\left\|\boldsymbol{\theta}\right\|_1}{2b}\right). \tag{2.12}$$

From this posterior PDF we derive an expression for the MAP estimate:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\theta} p(\boldsymbol{\theta}|\mathbf{y}) \tag{2.13}$$

$$= \arg\max_{\theta} \log\left(p(\boldsymbol{\theta}|\mathbf{y})\right) \tag{2.14}$$

$$= \arg\min_{\theta}\left(\frac{1}{2\sigma_w^2}\left\|\mathbf{y}-\Phi\Psi\boldsymbol{\theta}\right\|_2 + \frac{\left\|\boldsymbol{\theta}\right\|_1}{2b}\right) \tag{2.15}$$

$$= \arg\min_{\theta}\left(\left\|\mathbf{y}-\Phi\Psi\boldsymbol{\theta}\right\|_2 + \frac{\sigma_w^2}{b}\left\|\boldsymbol{\theta}\right\|_1\right). \tag{2.16}$$

Setting τ as

$$\tau = \frac{\sigma_w^2}{b}, \tag{2.17}$$

this is equal to the BPDN optimization problem of (2.7). This Bayesian interpretation of CS is heavily used in discussion of message passing algorithms, discussed in Section 2.5.2.

2.3.2 Model-based Compressed Sensing

In model-based CS, the constraint on number of non-zero components $k = \|\boldsymbol{\theta}\|_0$ is replaced by a more sophisticated model [18]. The most notable models are block-sparse and tree-sparse signals. A bound on the performance of ℓ_1 BP in reconstruction of approximately sparse data, distributed according to a power law, is also given.

The traditional CS approach constrains the set of possible signals, but still allows the signal $\boldsymbol{\theta}$ to lie on any of the $\binom{N}{k}$ possible k-sparse subspaces, with equal likelihood. For natural signals where this is too broad, Model-based CS further restricts

the allowed subspaces, allowing reconstruction from fewer measurements (i.e. smaller N). The RIP is replaced with a model-based RIP which defines isometry only for the allowed supports.

2.3.3 Blind Compressed Sensing

Blind CS [21] considers the case where both the (transformed) signal of interest $\boldsymbol{\theta}$ and sparsity basis Ψ are unknown. At first glance, this may seem an impossible task, and in fact in general it is. In spite of this, the sparse prior on $\boldsymbol{\theta}$ can be combined with additional constraints on Ψ to allow recovery of the signal $\mathbf{x}$ from an underdetermined set of measurements. This requires collecting multiple signals which are sparse under the same sparsity basis Ψ and jointly determining the basis for all of them. While blind CS achieves correct reconstructions with high probability, it requires higher sensing rates than the non-blind case.

Three constraints on the basis are considered in [21]. In the first, Ψ is known to be one of a finite and known set of commonly occurring bases, for instance one of several possible wavelet transforms. This case can be treated as a series of CS problems. In the second case, the rows of the basis are known to be a subset of a much larger dictionary matrix. This problem can be solved using standard CS methods, treating the larger dictionary as an overcomplete sparse basis. In the final case, Ψ is block-diagonal and the problem is solved by reformulating it as a dictionary learning problem.

2.4 Sensors for Compressive Image and Video Capture

This section describes two notable hardware implementations for capture of images and image-like data: the single-pixel camera and magnetic resonance imaging.

2.4.1 *Single-pixel camera*

Researchers at Rice University have developed a prototype single-pixel camera [1, 22] and complementary algorithms for processing of its output. The camera replaces the focal plane array of a conventional camera with a digital micromirror device (DMD) and a single light detector. The DMD displays a series of pseudorandom masks corresponding to the rows of the measurement matrix Φ, effectively performing a series of inner products on the incoming frame. The modulated light is then focused on a photodiode, which measures the magnitude of this inner product.

The single-pixel camera concept described above is being commercialized by InView corporation. InView's short-wave infrared (SWIR) cameras are capable of operating in a high-resolution/low-framerate and low-resolution/high-framerate mode. InView's products are not truly single-pixel: multiple measurements are sensed in parallel, reducing the time to acquire a single image [23].

2.4.2 *Medical Imaging and MRI*

Some of the most valuable contributions of CS are in the field of medical imaging, and especially in magnetic resonance imaging (MRI) [24]. Other applications include computed tomography (CT), positron emission tomography (PET), and Ultrasound imaging. The application of CS theory to MRI is briefly described in this section.

MRI exploits magnetic resonance, a process by which atoms in a magnetic field absorb and re-emit RF energy at a specific resonance frequency. In a typical MRI scanner, a very strong (typically 1-10 T) magnetic field is established in the scanner with a superconducting electromagnetic coil. Additional coils establish gradients in magnetic field strength over the field of view. A sequence of RF pulses is then

transmitted into the field of view and the response of the object to be imaged is recorded.

MRI is naturally amenable to CS techniques; the time and monetary cost per sample is high, creating an immediate need for techniques which reduce the number of samples needed. Sensing is also performed in k-space (i.e. Fourier domain) so reconstruction must be done whether or not CS is used. A conventional 3D MRI performs Cartesian sampling and outputs a "data cube" of samples evenly spaced on the three axes of k-space. An FFT is then needed to recover the 3-dimensional image for viewing. In parallel imaging implementations, more sophisticated reconstruction techniques are required to process the data from multiple receiver coils (sensors).

Designing the sensing matrix of an MRI is non-trivial. Sampling is limited by the maximum slew rate of the equipment used; successive samples must be near each other in k-space [25, 26]. In [27], the Cartesian data cube is simply undersampled. In [28, 29, 30, 31], a radial trajectory was used, and in [32, 33] a spiral trajectory was used. [34] advocates a pseudorandom poisson-disc sampling approach; this method generates good incoherence between the sparse basis and the samples, while maintaining relatively even coverage of k-space.

With the imaging speedup available from CS, applications such as 4D MRI, which adds video to the 3D MRI image, become feasible. In [35], a radial sensing trajectory was used to reconstruct a 4D MRI of the lungs during forced expiration. Acquisition time for each 3D frame was under 150 ms. "Bookend" image acquisition phases at the beginning and end of the expiration were used in which the patient is instructed to hold his or her breath while additional measurements are taken. The reconstruction algorithm minimizes spatial and temporal total variation and allows each frame to be expressed as a sum of the two bookend frames. This allows the full 4D signal to be reconstructed with only 3 measurements per 2D MRI "slice".

2.5 Reconstruction Algorithms

Many solvers have been developed for compressive reconstruction. I here discuss convex solvers, greedy solvers, and belief propagation approaches to the problem. Convex algorithms attempt to solve the (2.6) or a related problem more quickly than general interior-point methods. These algorithms include GPSR [36] and SPGL1 [37]. Although they are many times faster than general convex solvers, these algorithms typically are more computationally expensive than greedy and belief propagation based approaches, discussed in the following sections.

2.5.1 Greedy Solvers

The greedy algorithms follow an iterative procedure in which the signal support is iteratively built up. On each iteration, the elements of $\boldsymbol{\theta}$ which most improve the reconstruction are added to the support. Optionally, the least helpful components may also be dropped from the signal support. Three greedy algorithms are described here, although more exist.

Orthogonal Matching Pursuit

Orthogonal Matching Pursuit (OMP) and its related family of algorithms [38, 39, 40, 41] approximately solve (2.5) in a greedy manner. OMP runs in $O(MNk)$ time and is in practice much faster than convex solvers. This speedup is achieved using an iterative greedy procedure in which the support of the estimated solution $\hat{\boldsymbol{\theta}}$ is steadily increased until a k-sparse $\hat{\boldsymbol{\theta}}$ is achieved. At each iteration, the residual $\mathbf{r} = \mathbf{y} - \mathrm{A}\hat{\boldsymbol{\theta}}$ is calculated. The column $\mathrm{A}_{:,j}$ of A which maximizes $\langle \mathrm{A}_{:,j}, \mathbf{r} \rangle$ is added to the support of $\hat{\boldsymbol{\theta}}$. $\hat{\boldsymbol{\theta}}$ is then recalculated over the newly increased support. When $\|\boldsymbol{\theta}\|_0 = k$, the algorithm is halted and the current $\hat{\boldsymbol{\theta}}$ is output as the solution.

A theoretical guarantee exists for reconstruction by OMP, although it is weakened relative to the convex solvers. Specifically, for $\delta \in (0, 0.36)$ and $M \geq 4kln(N/\delta)$, OMP will identify the correct $\mathbf{x}$ with probability greater than $1 - 2\delta$ [39]. Note that this result is for the noiseless case ($\mathbf{w} = 0$).

CoSAMP

The CoSAMP reconstruction algorithm [42] is similar to OMP but adds the ability to discard unneeded elements from the support of $\hat{\mathbf{x}}$. On each iteration, some number of components are added to the support and least squares estimation is performed. The smallest components of the least-squares estimate are then discarded to maintain the desired sparsity level $\|\hat{\mathbf{x}}\|_0 = k$.

AdMIRA

The atomic decomposition for minimum rank approximation (ADMiRA) algorithm [43] applies an approach generalizing CoSAMP to the problem of reconstructing low-rank matrices rather than sparse vectors. This is useful for processing of video with lighting changes but no motion, since if each frame is assigned to a single column of a matrix, the resulting matrix will be low-rank.

2.5.2 Graphical Methods

The compressed sensing architecture can be represented as a Bayesian Network. $\boldsymbol{\theta}$ is drawn from a sparse prior distribution, and $\mathbf{x}$ and $\mathbf{y}$ are then calculated based on $\boldsymbol{\theta}$. The problem can thus be represented as a cyclic causal model and solved accordingly. Generalized approximate message passing (GAMP) [44] is an algorithm based on a quadratic approximation of a loopy belief propagation (BP) algorithm. In loopy BP, the measured vector $\mathbf{y}$ passes messages to the elements of $\mathbf{x}$ consisting of the condi-

tional probability of each possible value. Similarly, the sparse prior passes messages down to $\boldsymbol{\theta}$. GAMP may be viewed as a version of this message-passing algorithm in which only the means and variances of the messages are passed. In this section, the concept of Bayesian networks is first introduced. The belief propagation (BP) algorithm is described, followed by a review of its many extensions, generalizations, and approximations.

Bayesian Networks

A Bayesian network, or Bayes net, consists of a directed acyclic graph (DAG) in which nodes represent random variables and edges represent dependencies between nodes. Any node is conditionally independent from its non-neighboring nodes given its upstream and downstream neighbors. Although no directed cycles are allowed, *undirected* loops may be present. This constraint defines a clear path from causes (e.g. differing hypotheses) to effects (e.g. experimental measurements). Figure 2.1 shows a simple Bayesian network. In this example X_5 is conditionally independent from all other nodes given X_3. The goal of inference is to determine $p(X_i|E)$, the distribution of each unknown variable given a set of known evidence variables. Evidence may appear at any node in a network, but most typically is measured at the furthest downstream leaf nodes. Note also that the graph in Figure 2.1 has an undirected loop, shown in bold.

Performing inference on a Bayesian Network consists of estimating the marginal conditional probability distribution of a node or nodes (the unknowns), given values of a different set of nodes (the evidence). Performing exact inference on Bayesian networks has been shown to be in general NP-hard [45]. However, constrained versions of the problem have been solved exactly and good approximate solutions exist for many other cases. Belief propagation (BP), discussed below, generates exact solutions for

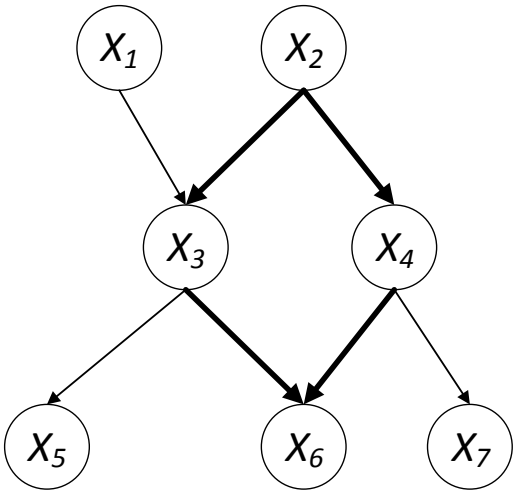


Figure 2.1: A Simple Bayesian Network.

tree and polytree graphs and leads to good approximations for many other problems. Further modifications of BP offer advantages in computational burden or stability.

Belief Propagation for Trees

Pearl's belief propagation (BP) algorithm [46] is a method for performing inference on Bayesian networks, Markov random fields, and factor graphs. In this section its operation on polytree-structured Bayesian networks is described. In graphs which are trees or polytrees, BP produces exact results. However, no such guarantee exists for graphs containing undirected loops.

Consider the fragment of a Bayesian network shown in Figure 2.2. This network consists of a node X whose value is to be estimated based on a set of downstream ("child") nodes $\{Y_1, Y_2, ...\}$ and upstream ("parent") nodes $\{U_1, U_2, ...\}$. The conditional probability distributions are known, and the upstream and downstream nodes are conditionally independent given X. The PDF of X can then be written in terms

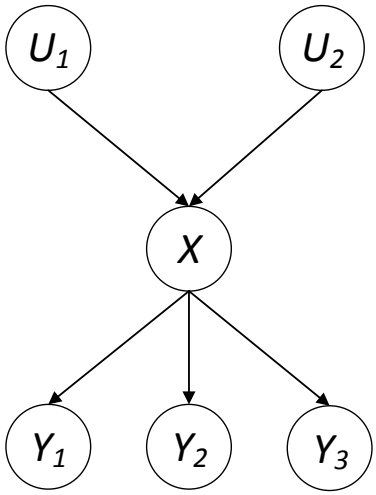


Figure 2.2: A Fragment of a Bayesian Network.

of the PDFs of its upstream and downstream neighbors as

$$p(x|u_1, u_2, ..., y_1, y_2, ...) = \frac{1}{\mathcal{Z}} \prod_i p(y_i|x) \int_{u_1,u_2,...} p(x|u_1, u_2, ...) \prod_j p(u_j)\ du_1 du_2 ... \tag{2.18}$$

where $\mathcal{Z}$ is a normalizing constant.

In BP, each node passes messages to its upstream and downstream neighbors. Each message is a measure but is not a probability distribution; it does not in general integrate to one. In the upstream direction, each node Y_i sends a message $\lambda_{Y_i X}(x)$to node X. Likewise, each upstream node U_i sends a message $\pi_{U_i X}(x)$. These λ and π messages take the place of the conditional probabilities in (2.18) to form a new expression for our belief about $p_X(x)$. Specifically,

$$BEL_X(x) = p(x|u_1, u_2, ..., y_1, y_2, ...) = \frac{1}{\mathcal{Z}} \lambda_X(x) \pi_X(x) \tag{2.19}$$

$$\lambda_X(x) = \prod_i \lambda_{Y_i X}(x) \tag{2.20}$$

$$\pi_X(x) = \int_{u_1,u_2,...} p(x|u_1, u_2, ...) \prod_j \pi_{U_j X}(x) du_1 du_2 ... \tag{2.21}$$

In contrast to (2.18), $\pi U_j X(x)$ and $\lambda_{Y_i X}(x)$ are not known exactly, but are based on the messages received at U_j and Y_i respectively. Note that $\lambda_{Y_i X}(x)$ denotes a measure passed as an upstream message from child to parent, while $\lambda_X(x)$ refers to a measure calculated at node X from received λ messages. Similarly, $\pi_X(x)$ is calculated from received $\pi_{U_i X}(x)$ messages. The subscript X will be dropped for brevity in the following discussion unless it is needed to avoid ambiguity. Normalization of beliefs is not necessary but may improve numerical stability and prevent underflow problems.

In a tree or polytree network, the assumption of conditional independence above is satisfied and (2.18) holds. Inference then proceeds as follows. Evidence nodes (i.e. nodes whose values have been measured) are initialized. That is, for each evidence node E_i with known value e_i,

$$\lambda_{E_i}(x) = \delta(x - e_i) \tag{2.22}$$

$$\pi_{E_i} = \delta(x - e_i) \tag{2.23}$$

$$BEL_{E_i}(x) = \delta(x - e_i). \tag{2.24}$$

Non-evidence nodes X_i with no parents are initialized using their Bayesian prior, as follows:

$$\pi_{X_i}(x) = p_{X_i}(x). \tag{2.25}$$

The downstream evidence $\lambda(x)$ of nodes with no children is initialized to 1. All other messages are unavailable at the start of the algorithm.

Once initialized, belief updates proceed by iterating the following steps until convergence:

- If all upstream messages $\lambda_{Y_i X}(x)$ are available at a node X, calculate $\lambda_X(x)$.
- If all downstream messages $\pi_{U_i}(x)$ are available at a node X, calculate $\pi_X(x)$.

- If all messages except $\pi_{U_iX}(x)$ are available at node X, calculate $\lambda_{XU_i}(x)$ and send to node U_i.

- If all messages except $\pi_{Y_iX}(x)$ are available at node X, calculate $\pi_{XY_i}(x)$ and send to node Y_i.

Stated more simply, each π and λ message is calculated as soon as the information on which it depends is available. This procedure results in messages first propagating inward from leaf nodes (and non-leaf evidence nodes), and then outward from the last node to be reached. Each message is calculated only once and retains its value until the end of the algorithm.

Approximate Loopy Belief Propagation

Many Bayesian inference problems are expressed in terms of graphs with loops or involve the computation of intractable integrals. This section covers several noteworthy approaches to applying BP when computing exact marginals is not possible.

Unlike tree-structured graphs, BP does not produce exact results on networks containing undirected loops. Intuitively speaking, evidence is passed around a loop, mistaken for new evidence, and "double-counted" [47]. In fact, nodes may not even converge toward a single value. Despite this disadvantage, "loopy BP" algorithms can perform very well. For instance, low-density parity check (LDPC) and turbo codes can achieve the maximum capacity of a channel using a decoding procedure which implements BP [48, 49].

In [49] an attempt is made to empirically study the limits of BP. This is done by running BP on several previously developed Bayesian networks containing multiple loops. BP is adapted to loopy graphs by simply updating every node in parallel on every iteration of the algorithm. The BP result is compared against that obtained by an

importance sampling algorithm. Four networks are used: PYRAMID, a hierarchical network of binary nodes; toyQMR, a simulated medical diagnosis predictor initialized with random probabilities; ALARM, a network for detecting emergencies in intensive care patients, and QMR-DT, a binary network mapping 600 diseases to 4000 findings (i.e. symptoms). In the cases of PYRAMID, toyQMR, and ALARM, BP converges to near the correct marginal probability. In the case of QMR-DT, however, BP does not converge and instead oscillates between two very different states. The authors are able to induce oscillation in the much smaller toyQMR network by making the prior probabilities of different diseases much lower, e.g. by choosing from the interval $[0, 0.1]$ rather than $[0, 1]$. Stability in the larger QMR-DT network can be induced by replacing the actual prior probabilities of disease by uniform random values on the interval $[0, 1]$. Adding a momentum term to the update reduces oscillation but does not necessarily improve the accuracy of the estimates for QMR-DT.

The success of turbo codes has prompted further analysis of BP, leading to additional performance guarantees. In [47] a belief revision procedure is described which partially corrects for the double-counting effect in networks consisting of a single loop of unknown nodes, each connected to a single evidence node. Under a set of easily satisfied restrictions, the belief revision procedure generates the correct maximum a posteriori (MAP) estimate. However, the distribution over the non-evidence nodes is not exact; double-counting of evidence leads to overconfident estimates.

In addition to loops, the problem of estimating marginal probability distributions is often intractable, or at least very computationally expensive. Approximate message passing algorithms attempt to solve this problem by instead estimating the distribution. Applied to CS, these algorithms are referred to as approximate message passing (AMP). Similar methods are known in other applications as "approximate BP" [50] or "relaxed BP" [51, 52]. In [53, 54] an i.i.d. Laplace (double-sided exponential) prior

is placed on $\mathbf{x}$, and messages are approximated as Gaussian distributions. Under this assumption only a mean and variance must be passed, rather than a measure on the real line. The authors argue that the central limit theorem causes messages to converge toward Gaussian distributions as the problem size becomes large. Furthermore, the variances of all the messages are approximately identical. Expressions are derived for the large-system limit of the means $\mu_{\pi_{YX}}$ and μ_λ and variance σ_λ^2 of the messages. In vector notation, these are given as

$$\boldsymbol{\mu}_{\boldsymbol{\pi}_{\mathbf{xy}}}^{t+1} = \eta(A^* \boldsymbol{\mu}_{\boldsymbol{\lambda}_{\mathbf{yx}}}^{t} + \boldsymbol{\mu}_{\boldsymbol{\pi}_{\mathbf{xy}}}^{t}; \sigma_\lambda^{2^t}), \tag{2.26}$$

$$\boldsymbol{\mu}_{\boldsymbol{\lambda}_{\mathbf{yx}}}^{t} = \mathbf{y} - A\boldsymbol{\mu}_{\boldsymbol{\pi}_{\mathbf{xy}}}^{t} + \frac{1}{\delta}\boldsymbol{\mu}_{\boldsymbol{\lambda}_{\mathbf{yx}}}^{t-1} \left\langle \eta'(A^* \boldsymbol{\mu}_{\boldsymbol{\lambda}_{\mathbf{yx}}}^{t-1} + \boldsymbol{\mu}_{\boldsymbol{\pi}_{\mathbf{xy}}}^{t-1}; \sigma_\lambda^{2^{t-1}}) \right\rangle \tag{2.27}$$

$$\sigma_\lambda^{2^t} = \frac{\sigma_\lambda^{2^{t-1}}}{\delta} \left\langle \eta'(A^* \boldsymbol{\mu}_{\boldsymbol{\lambda}_{\mathbf{yx}}}^{t-1} + \boldsymbol{\mu}_{\boldsymbol{\pi}_{\mathbf{xy}}}^{t}; \sigma_\lambda^{2^{t-1}}) \right\rangle \tag{2.28}$$

where $\langle \cdot \rangle$ denotes the average of a vector. $\eta(x; b)$ and $\eta'(x; b)$ are the soft threshold function and its derivative, given as

$$\eta(x; b) = \text{sign}(x)(|x| - b)_+ \tag{2.29}$$

$$\eta'(x; b) = \frac{\partial^2}{\partial x \partial b}(\eta(x; b)) \tag{2.30}$$

Since the variance σ_λ^2 is identical for all π and all σ messages, it is not passed; only one scalar, representing the mean, is passed in each direction along the graph edges. This algorithm displays comparable performance to Basis Pursuit (Section 2.1) at much lower complexity. A similar algorithm for Basis Pursuit Denoising in AWGN as in (2.2) is also derived.

The work of [53] is extended and generalized in [44]. Arbitrary prior distributions $p(\mathbf{x})$ are allowed, instead of a Laplace distribution. Arbitrary measurement noise models are also considered, instead of a noiseless (BP) or AWGN (BPDN) distribution on $p(\mathbf{y}|\mathbf{x})$. Finally, message updates for a nonlinear sensing operator A are allowed by the algorithm, although little analysis is given for the nonlinear case.

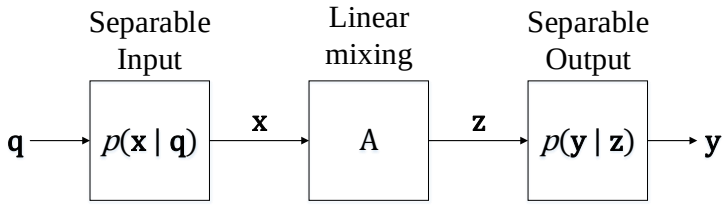


Figure 2.3: The GAMP System Model.

The measurement operation is divided into two separable but nonlinear measurement channels joined by a linear mixing operation.

Figure 2.3 shows a block diagram of the system model for GAMP. An unknown input vector $\mathbf{x} \in \mathbb{R}^N$ is generated based on a known $\mathbf{q} \in Q^N$. $\mathbf{x}$ is passed through a linear mixing channel with mixing operator A, generating the "noiseless" measurement matrix $\mathbf{z}$. Finally, $\mathbf{z}$ passes through an output channel, generating measurement vector $\mathbf{y}$. The input and output channels are separable, with mixing occuring only between $\mathbf{x}$ and $\mathbf{z}$. The conditional probability distributions are given by

$$p(\mathbf{x}|\mathbf{q}) = \prod_{n=1}^{N} p(x_n|q_n) \tag{2.31}$$

$$p(\mathbf{z}|\mathbf{x}) = \delta(\mathbf{z} - A\mathbf{x}) \tag{2.32}$$

$$p(\mathbf{y}|\mathbf{z}) = \prod_{m=1}^{M} p(y_m|z_m). \tag{2.33}$$

$\mathbf{q} \to \mathbf{x} \to \mathbf{z} \to \mathbf{y}$ forms a Markov chain.

The procedure for state updates in GAMP is given in Algorithm 1, while Figure 2.4 gives a graphical representation of the dependencies between the different variables. $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ are, as one would expect, the estimates of $\mathbf{x}$ and $\mathbf{z}$ at iteration t of the algorithm. The semantic meaning of the other variables, however, is less obvious. $\hat{\mathbf{r}}$ and $\hat{s}$ can be viewed as passing information "upstream", while $\hat{\mathbf{p}}$ passes it downstream. In MMSE GAMP, discussed below, the τ variables may be interpreted as variances of the corresponding variables.

Algorithm 1 GAMP Algorithm.

Given A $\in \mathbb{R}^{M\times N}$, $\mathbf{y} \in \mathbb{R}^{M\times 1}$, $\mathbf{q}in\mathbb{R}^{N\times 1}$ and functions $g_{in}(r, q, \tau^{(r)})$ and $g_{out}(p, y, \tau^(p))$, produce a series of estimates $\hat{\mathbf{x}}(t)$ and $\hat{\mathbf{z}}(t)$.

Set $t = 0$, $\hat{\mathbf{s}}(-1) = \mathbf{0}$

Set $\hat{\mathbf{x}}(0)$ and $\tau^{(x)}(0)$ to initial values

repeat

for all $m \in [1, M]$ **do**

$$\tau_m^{(p)}(t) = \sum_{n=1}^{N} |a_{mn}|^2 \tau_n^{(x)}(t)$$

$$\hat{p}_m(t) = \sum_{n=1}^{N} a_{mn}\hat{x}_n(t) - \tau_m^{(p)}(t)\hat{s}_m(t-1)$$

$$\hat{z}_m(t) = \sum_{n=1}^{N} a_{mn}\hat{x}_m(t)$$

$$\hat{s}_m(t) = \mathbf{g}_{out}(\hat{p}_m(t), y_m, \tau_m^{(p)}(t))$$

$$\tau_m^{(s)} = -\frac{\partial}{\partial \hat{p}}\mathbf{g}_{out}(\hat{p}_m(t), y_m, \tau_m^{(p)}(t))$$

end for

for all $n \in [1, N]$ **do**

$$\tau_n^{(r)}(t) = 1/\sum_{m=1}^{M} |a_{mn}|^2 \tau_m^{(s)}(t)$$

$$\hat{r}_n(t) = \hat{x}_n(t) + \tau_j^{(r)}(t) \sum_{m=1}^{M} a_{mn}\hat{s}_m(t)$$

$$\hat{x}_n(t+1) = g_{in}(\hat{r}_n(t), q_n, \tau_n^{(r)})$$

$$\tau_n^{(x)}(t+1) = \tau_n^{(r)}(t)\frac{\partial}{\partial \hat{r}} g_{in}(\hat{r}_n(t), q_n, \tau_n^{(r)})$$

end for

$t = t + 1$

until t reaches the maximum number of iterations

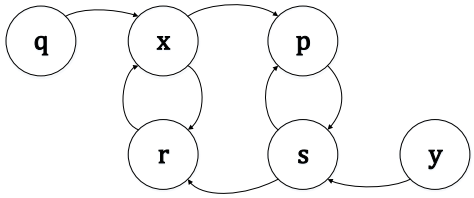


Figure 2.4: Dependencies of Messages in the GAMP Algorithm.
Arrows entering a node indicate variables on which that node depends.

The estimator functions g_{in} and g_{out} must also be chosen. [44] gives two methods for selecting these functions, based on MAP and minimum mean square error (MMSE) estimation. For MMSE estimation, the estimators are given by

$$g_{in}(\hat{r}, q, \tau_r) = \mathrm{E}\left(\hat{x}|\hat{r}, q, \tau^{(r)}\right). \tag{2.34}$$

$$g_{out}(\hat{p}, y, \tau^{(p)}) = \frac{1}{\tau^{(p)}}\left(\mathrm{E}\left(z|\hat{p}, y, \tau^{(p)}\right) - \hat{p}\right). \tag{2.35}$$

In the EMBGAMP algorithm [55], a Bernoulli-Gaussian prior is used, with prior PDF given by

$$p(\boldsymbol{\theta}) = \prod_{i=1}^{N}\left(q\delta(\theta_i) + (1-q)\frac{1}{\sqrt{2\pi\sigma_\theta^2}}\exp\left(\frac{\theta_i^2}{2\sigma_\theta^2}\right)\right). \tag{2.36}$$

. That is, elements of $\boldsymbol{\theta}$ are assumed to take non-zero values with probability $1-q$. The non-zero elements are then normally distributed with mean zero and variance σ_θ^2. α and σ_θ^2 unknown and are updated using expectation maximization on each iteration of the algorithm over the graph. In this way, the EMBGAMP algorithm is self-tuning and does not require the user to select any parameters to optimize its performance. It also is noteworthy in that it can outperform ℓ_1 basis pursuit for the noiseless case, discussed in Section 2.7.

In [56] the GAMP algorithm is extended to bilinear models, allowing the use of message passing as a solver for problems such as matrix completion, robust PCA, and dictionary learning. Under the bilinear model, both $\mathrm{A} \in \mathbb{R}^{M \times N}$ and $\mathrm{X} \in \mathbb{R}^{N \times L}$ are unknown and drawn independently from from their respective prior distributions. Note that X and Y are here allowed to be a matrix rather than a vector. The posterior PDF is given by

$$p(\mathrm{X}, \mathrm{A}, |\mathrm{Y}) = p(\mathrm{Y}|\mathrm{X}, \mathrm{A})p(\mathrm{X})p(\mathrm{A}) \tag{2.37}$$

$$= p(\mathrm{Y}|\ \mathrm{AX}\ )\ p(\mathrm{X})p(\mathrm{A}). \tag{2.38}$$

An optional expectation maximization step may be also be included as in EMBGAMP [55], leading to the EM-BiG-AMP algorithm. In Monte Carlo trials the EM-BiG-AMP algorithm shows performance near theoretical maximum for matrix completion problems.

2.6 Video Reconstruction Algorithms

Reconstruction of CS images is relatively straightforward, if computationally demanding. It is well-known that natural images exhibit excellent energy compaction under the 2-dimensional discrete cosine transform (DCT) or any of several discrete wavelet transforms (DWT); this compressibility is exploited by the JPEG and JPEG2000 standards, respectively. Video compression algorithms, however, exploit both compressibility of individual frames and correlations between nearby frames. There are several examples in the literature of video reconstruction algorithms which incorporate this inter-frame information.

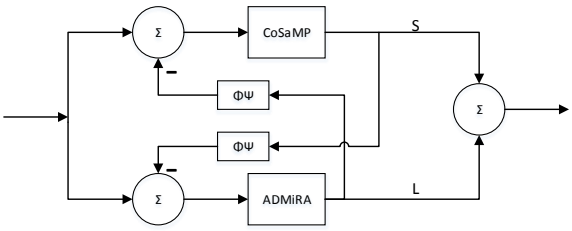


Figure 2.5: Conceptual Block Diagram of the SpaRCS Algorithm.

2.6.1 SPArse and low Rank decomposition via Compressive Sensing (SPArCS)

In [57], a quasi-static background with small moving targets is assumed. This assumption allows a video sequence to be represented as the sum of a low rank matrix, representing the background, and a sparse matrix, representing moving objects in the foreground. The SpaRCS algorithm, a greedy pursuit derived from the CoSAMP and ADMiRA algorithms [42], is then used to recover the video sequence. Figure 2.5 shows a conceptual block diagram of SpaRCS: On each iteration, CoSaMP and ADMiRA each attempt to solve for the other algorithm's residual. Although the SpaRCS algorithm performs well when the assumptions of static background and small motion are satisfied, it quickly fails when the background is not static or when targets are large. In addition, a relatively large number of frames (typically hundreds) are needed to accurately estimate the background. This is most likely a consequence of the algorithm's failure to take advantage of the known compressibility of the background image when performing reconstruction.

2.6.2 CS MUltiscale VIdeo recovery (CS-MUVI)

The CS-MUVI algorithm [58] most nearly achieves the goal of fully utilizing known information, including both optical flow and compressibility of individual frames. In

this algorithm, a special sensing matrix Φ is used which creates a well-conditioned matrix when combined with an upsampling operator U. The combined matrix ΦU is then used to reconstruct a low-resolution version of the video using least-squares estimation. Optical flow is then estimated using this low-resolution video and the resulting optical flow is used to improve reconstruction of the compressively sensed high-resolution video. This method, while highly effective, fails to reconstruct small objects since these objects are not visible in the low-resolution version of the video.

2.6.3 Background Subtraction

In [59] a static background with small targets is again assumed. In this case, however, the difference between two images is reconstructed. This is possible because as long as targets are small, the difference image is known to be spatially sparse. This algorithm achieves good results as long as all assumptions are met.

2.6.4 Other methods

In [60], global optical flow is estimated from compressive measurements of two video frames and then used to reconstruct both frames. While useful, this method requires the two input frames to be pure translations of one another. When optical flow is not constant across the entire image or is large (on the order of several pixels), the algorithm fails. This method also requires the use of a highly unorthodox sensing scheme in which compressively sensed pixels are compactly supported on the image. A sensing matrix of this type is likely to have undesirable properties.

In [61], dense optical flow is estimated from compressive measurements without performing any reconstruction. When the measurement rate is high, good estimates of optical flow are produced. However, an unconventional sensing scheme is again used, in which each measurement is compactly supported in one of the image's spatial

Table 2.1: Runtimes Required for Phase Transition Plot.

Algorithm	L1EQ	SPGL1	OMP	CoSaMP	GAMP	EMBGAMP
Runtime (hours)	51.5	9.0*	18.7	24.8	1.7*	2.2*

*SPGL1, GAMP, and EMBGAMP are partially or fully compiled C code. Others are interpreted MATLAB.

dimensions. In fact, each compressive measurement is restricted to a single row of the uncompressed scene.

Although not strictly a CS reconstruction algorithm, the work of [62] deserves mention. The authors solve an ℓ_1 problem similar to (2.6) to estimate the optical flow field between stereoscopic images. Sparsity in the derivative of the optical flow field is promoted; this assumption holds true when motion is caused by a small number of rigid objects undergoing translational motion. This work is discussed in more detail in Section 3.1.2.

2.7 Performance of CS reconstruction

Several approaches to the problem of CS reconstruction were presented above in Section 2.5. Applicable guarantees and bounds on the performance of these algorithms are discussed here, as well as a comparison of the computational cost of the different algorithms.

2.7.1 Computation Speed

Table 2.1 shows the time required to run each of the simulations in Figure 2.6. The comparison is imperfect, since the different algorithms are optimized to varying degrees, but trends are clear. Convex solvers are far more computationally demanding than greedy or message-passing algorithms.

2.7.2 Cramer-Rao Lower Bound

The Cramer-Rao lower bound (CRLB) is arguably the most well known bound on estimation problems. It is included here for completeness, though it is not very informative for the problem of CS. The CRLB gives a minimum bound on the variance of any unbiased estimator $\hat{\boldsymbol{\theta}}$ of the vector $\boldsymbol{\theta}$ given measurements $\mathbf{y}$. It is defined in terms of the Fisher information matrix $\mathrm{J}(\boldsymbol{\theta})$by the expression

$$\operatorname{var}\hat{\theta} \geq \mathrm{J}(\theta)^{-1} \tag{2.39}$$

$$\mathrm{J}_{ij}(\boldsymbol{\theta}) = -\operatorname*{E}_{\mathbf{x}}\left[\frac{\partial^2}{\partial\theta_i\partial\theta_j}\log(p(\mathbf{x}|\boldsymbol{\theta}))\right]. \tag{2.40}$$

2.40) holds only under certain easily satisfied regularity conditions and is not the most universal definition of Fisher information. (2.39) holds only for unbiased estimators, and biased estimators may achieve lower variance. (It also does not take the prior distribution of θ into account, since $\mathrm{J}(\theta)$ depends only on $p(\mathbf{y}|\theta)$. Allowing bias in estimation and defining a prior distribution $p(\boldsymbol{\theta})$ on the parameters to be estimated yields the Bayesian CRLB (BCRB), defined as [63]

$$\mathrm{E}((\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})^T) \geq \mathrm{J}_B^{-1}. \tag{2.41}$$

J_B is the Bayesian Fisher information matrix and is expressed in terms of the standard Fisher information matrix J and an additional term reflecting the prior:

$$\mathrm{J}_{ij} = \operatorname*{E}_{\mathbf{y},\theta}\left[-\frac{\partial^2\log p(\mathbf{y},\boldsymbol{\theta})}{\partial\theta_i\partial\theta_j}\right] \tag{2.42}$$

$$= \operatorname*{E}_{\boldsymbol{\theta}}(\mathrm{J}_{ij}) + \operatorname*{E}_{\boldsymbol{\theta}}\left[-\frac{\partial^2}{\partial\theta_i\partial\theta_j}\log p(\boldsymbol{\theta})\right] \tag{2.43}$$

$$= \mathrm{J}_D + \mathrm{J}_P \tag{2.44}$$

where J_D and J_p are defined as

$$\mathrm{J}_D = \underset{\boldsymbol{\theta}}{\mathrm{E}}(\mathrm{J}_{ij}) \tag{2.45}$$

$$\mathrm{J}_P = \underset{\boldsymbol{\theta}}{\mathrm{E}}\left[-\frac{\partial^2}{\partial\theta_i\partial\theta_j}\log p(\boldsymbol{\theta})\right] \tag{2.46}$$

The BCRB for CS reconstruction problems is derived in [64] for a typical Bayesian CS problem. The measurement matrix Φ is selected with zero-mean i.i.d. elements of variance σ_Φ^2. Noise is AWGN with variance $\sigma_{\mathbf{w}}^2$. $\boldsymbol{\theta}$ is distributed i.i.d. Bernoulli-Gaussian, with parameters q and σ_θ^2 and PDF $p(\theta_i) = q\delta(\theta_i) + (1-q)\frac{1}{\sqrt{2\pi\sigma_\theta^2}}\exp\left(\frac{\theta_i^2}{2\sigma_\theta^2}\right)$.

Under these assumptions, the BCRB is calculated as

$$\mathrm{J}_D = M\frac{\sigma_\Phi^2}{\sigma_{\mathbf{w}}^2}\Psi^T\Psi \tag{2.47}$$

$$\mathrm{J}_P = \frac{1-q}{\sigma_\theta^2}\mathrm{I} \tag{2.48}$$

$$\mathrm{J}_B = \mathrm{J}_D + \mathrm{J}_P = M\frac{\sigma_\Phi^2}{\sigma_{\mathbf{w}}^2}\Psi^T\Psi + \frac{1-q}{\sigma_\theta^2}\mathrm{I}. \tag{2.49}$$

Several problems with the bound in (2.49) are immediately evident. There is no choice of parameter values which creates a separation between regions where estimation is feasible or infeasible. Error actually increases as q increases (i.e. as the signal becomes more sparse). This is an artifact of the parametrization of $p(\boldsymbol{\theta})$, which leads to increasing signal energy as more non-zero components are added to the signal. The largest problem, though, is that there is no dependence on the number of parameters N, only on the number of measurements M. In the large system limit the J_D term dominates and drives the minimum error to zero. Likewise, in the noiseless ($\sigma_{\mathbf{w}}^2 = 0$) case, the J_D term is infinite. The BCRB therefore reduces to the trivial lower bound of $\mathrm{E}[(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})^T(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})] \geq 0$ for both the noiseless case and the large-system limit.

2.7.3 Phase Transition

A more productive alternative to the BCRB of Section 2.7.2 is clearly needed. One way to characterize the limits on reconstruction performance is to examine an algorithm's phase transition, loosely defined as the dividing line between regions where reconstruction is good versus poor. In this section, theoretical and empirical results on the phase transition regions of several CS reconstruction algorithms are presented and discussed.

Donoho and Tanner [65, 66] give a limit on the feasible region for ℓ_1 basis pursuit. The case of a random Gaussian measurement matrix $A \in \mathbb{R}^{M \times N}$ is considered. As the problem size approaches infinity, ℓ_1 basis pursuit generates a correct solution with probability approaching 1 as long as the condition

$$k < M \left(2e \log(N/M)\right)^{-1} \tag{2.50}$$

is met. As usual, $k = \|\theta\|_0$ is the sparsity level of the signal to be reconstructed. Empirical evidence also indicates that this limit holds for finite problems of reasonable size. Figure 2.6b shows the average reconstruction success rate over 10 trials, for a range of sparsity levels and sensing rates. Number of measurements $M = 1024$ was held constant and N and k were varied to achieve the desired ratios. Success rate is defined here as the percentage of trials with $\left\|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\right\|_\infty < 10^{-3}$. Although the phase transition of (2.50) is derived for i.i.d. Gaussian A, in [66] it is shown that many other choices of A show an identical phase transition, including Bernoulli, random Fourier, and random Hadamard matrices.

Additional refinements have been made to the Donoho-Tanner phase transition. It is known [67] that for exactly known sparsity level $\|\boldsymbol{\theta}\|_0 = k$, the maximum k grows proportionally to sensing rate r rather than $(1/r \log(1/r))^{-1}$ as in (2.50). In [67], however, (2.50) is shown to be tight for the ℓ_1 problem, in agreement with [65].

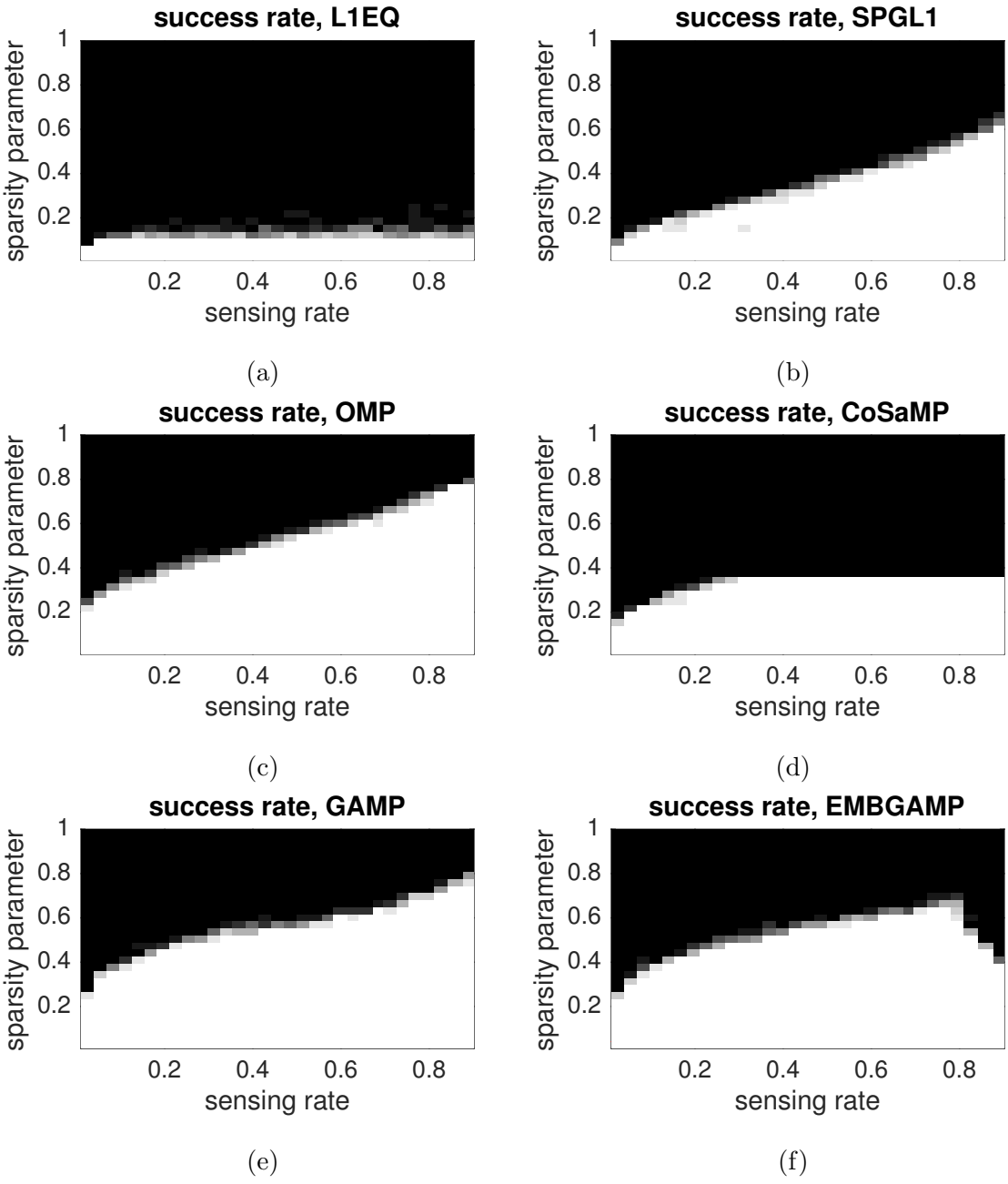


Figure 2.6: Phase Transition Regions for Different Reconstruction Algorithms. For each algorithm, success rate is plotted as a function of sparsity ratio k/M and sensing rate r. Algorithms shown are (a) convex ℓ_1 BP using the l1-magic library's l1eq command, (b) ℓ_1 BP using SPGL1 , (c) OMP, (d) CoSaMP, (e) GAMP with q, σ_θ^2 known, and (e) EMBGAMP.

This indicates a hard limit for the ℓ_1 basis pursuit problem; any improvements must come from other methods.

Different reconstruction algorithms may be compared by evaluating their phase transition region, as well as execution speed and error rates on finite-length data. Figure 2.6 shows empirically estimated phase transitions for two convex solvers, L1EQ from the l1-magic library [68] and SPGL1 [37]; two greedy algorithms, OMP [38] and CoSaMP [42]; and two message passing algorithms, GAMP[44] and EMBGAMP [55]. L1EQ, when run with the default parameters, showed very poor performance; this demonstrates the need for appropriate algorithm tuning. SPGL1, which nominally solves an equivalent problem, showed much better performance. OMP and CoSaMP show comparable performance to convex solvers at low sensing rates. The chosen CoSaMP implementation, though, requires $k < M/3$ for the successful solution of an internal least-squares problem. This leads to suboptimal performance at higher sensing rates. The message-passing algorithms both perform extremely well, with EMBGAMP slightly outperforming GAMP at low sensing rates but suffering from stability problems as more measurements are taken.

2.7.4 Sparse Recovery in Noise

All the phase transitions discussed above are for the case of noiseless measurement and perfectly sparse $\boldsymbol{\theta}$. A natural question is whether CS reconstruction algorithms are robust. How much noise can be added before the support of $\boldsymbol{\theta}$ can no longer be reliably determined? For the question of noisy CS reconstruction, [69] introduces the concept of the noise-sensitivity phase transition.

The noise-sensitivity phase transition is defined as follows. Let $\mathbf{y}$ be determined as in (2.2), as usual, under AWGN with $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_w^2 \mathrm{I})$. Let $MSE = \|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\|_2^2/N$ be the mean squared reconstruction error. The ratio MSE/σ_w^2 is defined as the

noise sensitivity. The authors then define a worst-case noise sensitivity for BPDN reconstruction as a function of sensing rate r and sparsity ratio k/M. This worst-case noise sensitivity is bounded over exactly the same region as the phase transition originally given in (2.50). This implies that BPDN is robust to noise and shows graceful degradation as noise increases.

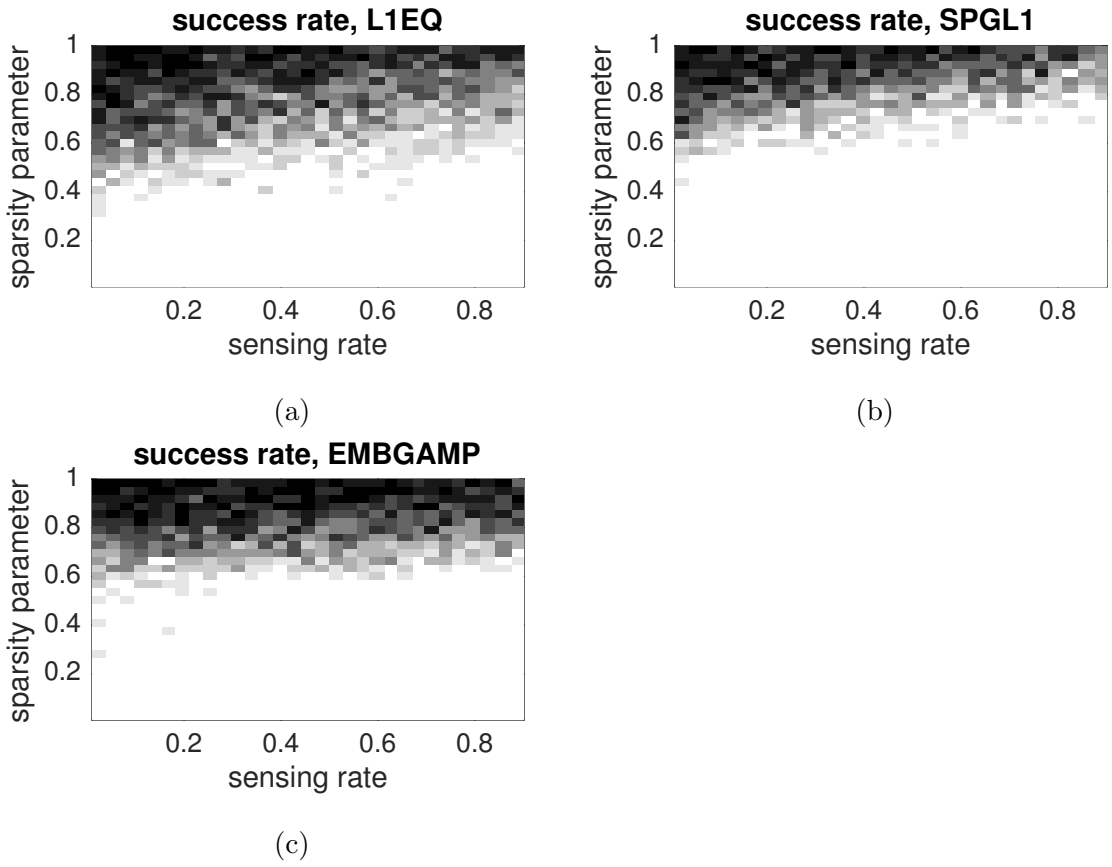
2.8 Recovery of Compressible Signals

Naturally-occurring signals are generally compressible but not perfectly sparse. For $\boldsymbol{\theta}$ following a power law (for instance), how well must energy be concentrated before data can be approximated as sparse? In this section, simulations are performed to characterize reconstruction for varying levels of signal compressibility, and the results are discussed. Approaches from the literature are described and another bound is put forward based on the noise sensitivity phase transition of [69].

In order to explore the phase transition for power law distributed data, $\boldsymbol{\theta}$ was drawn as a symmetric α-stable distribution with stability parameter α_θ, scale parameter $\gamma_\theta = 1$, and center parameter $\mu_\theta = 0$. The PDF of θ is not analytically expressible, but its characteristic function $\phi_\theta(t)$ is given by

$$\phi_\theta(t) = \exp\left(jt\mu - |\gamma_\theta t|^{\alpha_\theta}\right). \tag{2.51}$$

The stability parameter $\alpha_\theta \in (0, 2]$ controls the tail behavior, and therefore the compressibility, of the signal. $\alpha_\theta = 2$ corresponds to the Gaussian distribution, while $\alpha_\theta = 1$ is the Cauchy distribution. Except for the Gaussian ($\alpha_\theta = 2$) case, all α-stable distributions have power-law tails with PDF proportional to $|x|^{-(1+\alpha)}$ for large $|x|$. θ has infinite variance, but was normalized in simulations so that $\|\boldsymbol{\theta}\|_2 = 1$ to avoid numerical overflow problems at small α.



(a)

(b)

(c)

Figure 2.7: Phase Transitions for Symmetric Stable Distributed $\boldsymbol{\theta}$.

Sparsity is controlled by stability parameter α.

Figure 2.7 explores the phase transition for power-law distributed $\boldsymbol{\theta}$. Unlike the k-sparse case, the phase transition region does not show a sharp boundary between successful and unsuccessful reconstructions. There also appears to be much less benefit from increased sensing rates. Instead, a steady decrease in success rate for increasing α is observed.

A natural question arises: what is occurring in the region of perfect (or near-perfect) success rates? Is this a true phase transition or does it instead depend on the error tolerance, set somewhat arbitrarily at $10^{-}3$? To answer this question, the Modulation error ratio (MER) of each method is displayed in Figure 2.8. MER is the

ratio of signal energy to error energy and is defined as

$$MER = \frac{\|\boldsymbol{\theta}\|_2^2}{\left\|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\right\|_2^2}. \tag{2.52}$$

Examining the MER, there is again find no clear line between success and failure of the reconstruction; instead there is a steady decrease in MER as α is increased (i.e. as the signal becomes less sparse). In order to compare the performance of different algorithms, Figure 2.9 shows the difference in MER between EMBGAMP and SPGL1. The plotted quantity is MER under EMBGAMP minus MER under SPGL1. A large ($>$ 25dB) advantage for EMBGAMP is seen for small α (very sparse $\boldsymbol{\theta}$). SPGL1, on the other hand, outperforms EMBGAMP for cases of larger α and higher sensing rates.

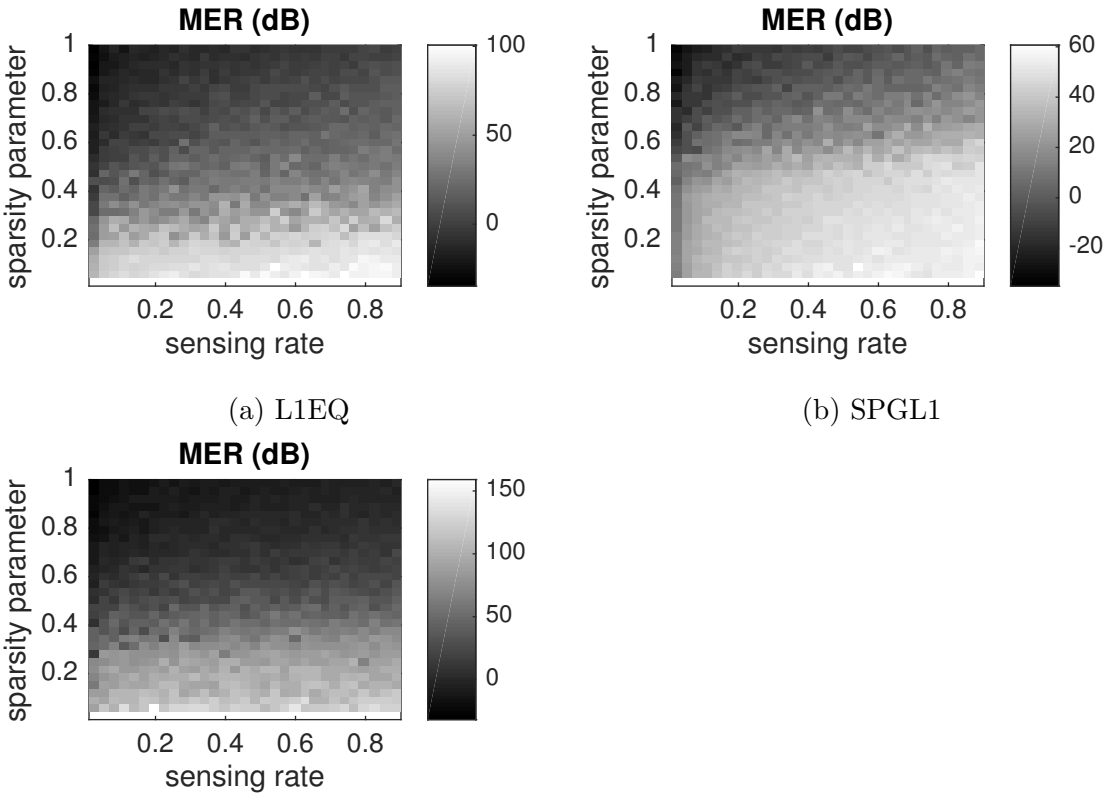
The disappearance of the phase transition in power-law data is worrying. If guarantees of reconstruction performance only apply to perfectly sparse data, they are of little use in most potential applications. Fortunately, there are bounds on reconstruction of imperfectly sparse data. The first approach is to treat all but the k largest components of $\boldsymbol{\theta}$ as noise; Using this approach, bounds have been derived for ℓ_1 reconstruction [14] and CoSaMP [42].

Let $\boldsymbol{\theta}$ be the (not necessarily sparse) signal to be reconstructed, and let $\boldsymbol{\theta}_k$ be the vector consisting of the k largest entries of $\boldsymbol{\theta}$, with zeros elsewhere. In [14] it is shown that the ℓ_1 BP solution $\hat{\boldsymbol{\theta}}$ satisfies

$$\left\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\right\|_1 \leq C_0 \left\|\boldsymbol{\theta} - \boldsymbol{\theta}_s\right\|_1 \tag{2.53}$$

as long as the restricted isometry property of (2.4) is satisfied with isometry constant $\delta_{2k} < \sqrt{2} - 1$. C_0 is a constant, given explicitly in [14].

Another possible approach is inspired by the noise-sensitivity phase transition [69]. Assume that there exists some function $g(r, \rho)$ which bounds the MSE of a



(a) L1EQ

(b) SPGL1

(c) EMBGAMP

Figure 2.8: Modulation Error Ratio for Symmetric Stable Distributed $\boldsymbol{\theta}$.

Sparsity is controlled by stability parameter α (plotted on y-axis).

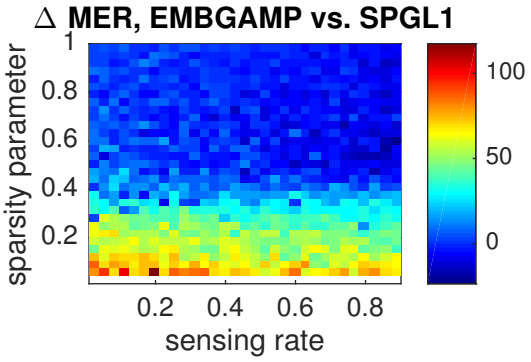


Figure 2.9: Comparison of MER of EMBGAMP versus SPGL1.

reconstruction on sparse data. That is,

$$\frac{\left\|\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}\right\|_2^2}{N} \leq \sigma_w^2 g(r,\rho), \quad , \|\boldsymbol{\theta}\|_0 \leq k \tag{2.54}$$

in the large-system limit as $M \to \infty$, $M/N \to r$, and $k/M \to \rho$. [69] gives strong evidence that such a $g(r,\rho)$ exists, and that it is finite over the same region as the Donoho-Tanner phase transition. Now, let $\boldsymbol{\theta}$ be non-sparse, but instead distributed i.i.d. symmetric, and centered on 0, with PDF $p_\theta(\theta)$ and CDF $F_\theta(\theta)$.

Proceed by dividing $\boldsymbol{\theta}$ into a "signal" component $\boldsymbol{\theta}_s$, which can be recovered with bounded error, and a "noise" component $\boldsymbol{\theta}_n$, which we abandon hope of recovering. Setting the threshold of signal and noise at $F_\theta^{-1}(\rho r/2)$, we have

$$\theta_{n,i} = \begin{cases} \theta_i, & |\theta_i| \leq -F_\theta^{-1}(\rho r/2) \\ 0, & \text{otherwise} \end{cases} \tag{2.55}$$

$$\boldsymbol{\theta}_s = \boldsymbol{\theta} - \boldsymbol{\theta}_n. \tag{2.56}$$

That is, $\boldsymbol{\theta}_n$ contains all of the signal components with whose magnitude is less than the top ρr quantile, with zeros elsewhere. Likewise, $\boldsymbol{\theta}_s$ contains all signal components with magnitude in at least the top ρr quantile. We can now write the CS sensing model as

$$\mathbf{y} = \mathrm{A}\boldsymbol{\theta} = \mathrm{A}\boldsymbol{\theta}_s + \mathrm{A}\boldsymbol{\theta}_n = \mathrm{A}\boldsymbol{\theta}_s + \tilde{\mathbf{w}}. \tag{2.57}$$

In the above, $\tilde{\mathbf{w}} = \mathrm{A}\boldsymbol{\theta}_n$ represents the effect of $\boldsymbol{\theta}_n$ on the measurement. By the central limit theorem,

$$\tilde{\mathbf{w}} \sim \mathcal{N}\left(\mathbf{0}, \sigma_{\theta_n}^2 \mathrm{A}\mathrm{A}^T\right), \tag{2.58}$$

where

$$\sigma_{\theta_n}(r,\rho) = \int_{F_\theta^{-1}(\rho r/2)}^{-F_\theta^{-1}(\rho r/2)} \theta^2 p_\theta(\theta) d\theta \tag{2.59}$$

is the variance of each element of $\boldsymbol{\theta}_n$. Most commonly used sensing matrices, including deterministic and random orthoprojectors and i.i.d. Gaussian matrices, satisfy $\mathrm{AA}^T \to \mathrm{I}$ in the large-system limit. In this case, the assumption of AWGN holds. We can now bound the MSE as

$$\frac{\left\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_s\right\|_2^2}{N} \leq \sigma_{\theta_n}^2(r,\rho)g(r,\rho). \tag{2.60}$$

Thus, the theory of compressed sensing is not only useful for perfectly sparse signals. It also bounds the error of CS reconstruction algorithms when operating on compressible signals which more closely model those found in nature.

2.9 Summay

In this chapter, the theory of compressed sensing was introduced and elaborated on. Two hardware implementations of CS sensors, the single-pixel camera and magnetic resonance imaging, were introduced. The convex, greedy, and message-passing reconstruction algorithm families were introduced, and several specific examples were elaborated on. Specialized reconstruction algorithms for image and video data were reviewed. Finally, the performance of CS signal recovery algorithms was discussed through both Monte Carlo simulations and analytically derived bounds.

Chapter 3

BACKGROUND AND LITERATURE REVIEW OF COMPUTER VISION AND IMAGE PROCESSING

My research depends heavily on existing work in the fields of traditional image and video processing, and on recent advances in machine learning and computer vision. In this chapter, I first discuss relevant work from image and video compression, enhancement, denoising and super-resolution imaging. I focus on the wavelet transform as an image compression method and optical flow in video compression. Next, several well-established problems in computer vision are presented and noteworthy solutions are described, with a focus on graph-based machine learning methods such as convolutional neural networks and deep Boltzmann machines.

3.1 Image Processing algorithms

Relevant work in image processing is covered in this section. The wavelet transform, a highly effective basis for compression of natural images, is first discussed. Next, algorithms for image motion estimation are described.

3.1.1 2-D Wavelet Transform

The two-dimensional wavelet transform is a highly efficient basis for compression of natural images. It achieves greater energy compaction than the discrete cosine transform (DCT) used in the JPEG image compression algorithm. Unlike the Fourier transform and DCT, the wavelet basis is inherently *local* in both time and frequency; it shares this property with the human visual system. A review of the discrete wavelet

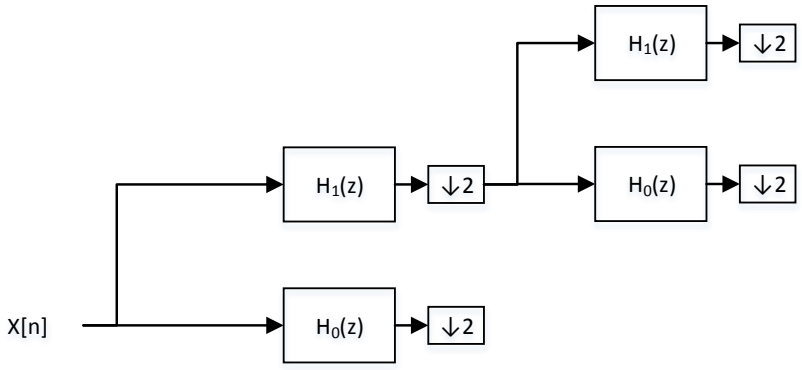


Figure 3.1: Block Diagram Representation of Wavelet Transform.

transform follows, beginning with the one dimensional algorithm and extending to two dimensions.

The discrete wavelet transform consists of a cascade of quadrature mirror filters (QMFs). [70], shown in Figure 3.1. At each stage, the original signal is split into approximation (lowpass) and detail (highpass) signals, each half the length of the original signal. In order to enable exact reconstruction of the signal, the lowpass filter $H_0(z)$ and highpass filter $H_1(z)$ must obey the constraints $H_1(z) = H_0(-z)$. Within these constraints, many different choices are possible. The Daubechies family of wavelets [71, 72] is the most well-known and commonly used. Unless otherwise specified, the Daubechies-4 (db4) wavelet will be used in the remainder of this work. Extending the wavelet transform to two dimensions is straightforward. at each stage, both dimensions of the signal are separated into approximation and detail coefficients, splitting the signal into four parts, each with half the resolution in each dimension. As before, the number of elements in the signal is unchanged.

3.1.2 Optical Flow and Image Registration

Modern video coding standards exploit optical flow to increase the efficiency of video compression algorithms. Stated simply, optical flow is the apparent motion of pixels in an image from one frame to the next. In video compression this enables a frame to be coded as a function of one or more nearby frames. In a visual tracking problem knowledge of optical flow can be used to estimate target motion. A related problem is image registration, in which two images or image patches are to be aligned with one another.

The image registration problem is formulated as follows. Given two image patches $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ we wish to determine a parameter vector $\mathbf{u}_t$ which gives the horizontal and vertical displacement from frame $t-1$ to t. As usual, 2-dimensional images are represented as one-dimensional column vectors, with columns of pixels "stacked" and vertically concatenated. The two video frames are related by the expression

$$\mathbf{x}_t = \mathrm{F}_{\mathbf{u}_t}\mathbf{x}_{t-1} + \boldsymbol{\eta} \tag{3.1}$$

where $\mathrm{F}_{\mathbf{u}_t}$ is a matrix implementing the interpolation operation mapping pixels in $\mathbf{x}_{t-1}$ onto $\mathbf{x}_t$. $\boldsymbol{\eta}$ is an error vector expressing details which are not captured by this matrix multiplication. These include unmodeled effects, for instance rotation, scaling, and warping in a translation-based optical flow model. $\boldsymbol{\eta}$ also encodes details of frame t which are out of frame, occluded, or otherwise not visible in frame $t-1$. The goal of optical flow estimation is to minimize some measure of the difference between $\mathbf{x}_{t-1}$ and $\mathrm{F}_{\mathbf{u}_t}\mathbf{x}_t$.

Lucas-Kanade Image Registration

The Lucas-Kanade algorithm [3] is arguably the most influential method for image registration and optical flow estimation. It relies on the assumption that images

are locally linear and that optical flow is locally constant among neighboring pixels. If the assumption of local linearity is not met, the input image can be smoothed with a low-pass filter before image registration is performed. The algorithm was developed to estimate translations but can be generalized to include other affine transformations[73]. This section briefly describes the procedure.

Assume a constant translational optical flow vector $\mathbf{h} = [h_1 \ h_2]^T$, giving the horizontal and vertical displacement from frame t to $t-1$. We wish to find the $\mathbf{h}$ which minimizes the cost function

$$\|\boldsymbol{\eta}\|_2^2 = \|\mathrm{F}_{u_t}\mathbf{x}_{t-1} - \mathbf{x}_t\|_2^2 \tag{3.2}$$

Let $f(\mathbf{u}, t)$, be the 3D continuous function from which the discrete images $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$ are sampled, where $\mathbf{u} = [u_1 \ u-2]^T$ gives spatial coordinates. From the assumption of local linearity it follows that, for small $\mathbf{h}$,

$$f(\mathbf{u}+\mathbf{h}, t) \approx f(w, h, t) + \mathbf{h}^T \frac{\partial}{\partial \mathbf{h}} f(\mathbf{u}, t). \tag{3.3}$$

where $\frac{\partial}{\partial \mathbf{h}} f(\mathbf{u}, t)$ is the spatial gradient. By iteratively solving (3.3) for $\mathbf{u}$ and linearizing around the new solution, the Lucas-Kanade algorithm converges to a solution for $\mathbf{u}$.

Block Matching

In video coding algorithms, the optical flow vectors of individual pixels are not calculated. Instead, the image is divided into rectangular blocks with constant flow vectors assigned to each block. Even with this simplification, an exhaustive search of all possible vectors impractical. Instead, search algorithms [74, 75, 76, 77, 78, 79] are used to efficiently estimate optical flow. Zhu and Ma's diamond search algorithm [79] is one such approach. In diamond search, two search steps, the large (LDSP) and small

(SDSP) diamond search patterns, are used. LDSP tests the current best estimate and all pixels with an ℓ_1 distance of exactly 2 from it. The LDSP step requires testing 9 possible optical flow vectors: the original guess and the eight neighbors of distance 2. The guess with the lowest MSE becomes the central point for the next iteration. LDSP steps are repeated until there is no change from the previous iteration; at this point a single SDSP step tests the 4 neighbors of distance 1, which have been previously untested. Diamond search may be viewed as a form of steepest descent, with a fixed ℓ_1 step size.

Sparsity-Exploiting Methods for Optical Flow Estimation

Sparse reconstruction methods may be used to estimate optical flow [62, 80]. In [62], a linear approximation is again used as with the Lucas-Kanade algorithm [3]. An ℓ_1 optimization problem resembling BP is solved, which enforces sparsity in both the registration residual $\boldsymbol{\eta}$ and in a transformed representation of the optical flow vectors.

Let $\mathbf{d}_x$, $\mathbf{d}_y$, and $\mathbf{d}_t$ be the partial derivatives of $\mathbf{x}_t$ with respect to the horizontal, vertical, and time parameters, respectively. From a linear approximation, we have for each pixel x_n with optical flow vector $[u_n \; v_n]^T$,

$$d_{x,n}u_n + d_{y,n}v_n + d_{t,n} \approx 0. \tag{3.4}$$

Combining the optical flow for each pixel gives the underdetermined system of equations

$$-\mathbf{d}_t \approx \begin{bmatrix} \mathbf{d}_x^T & \mathbf{d}_y^T \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \tag{3.5}$$

The underdetermined nature of this problem is traditionally solved by the block matching approach, which divides the image into regions of locally constant optical flow, thereby reducing the number of unknown variables. Shen and Wu replace this

ad-hoc approach with the observation that $\mathbf{u}$ and $\mathbf{v}$ both form compressible images with respect to a basis Ψ, such that

$$-\mathbf{d}_t \approx \begin{bmatrix} \mathbf{d}_x^T & \mathbf{d}_y^T \end{bmatrix} \begin{bmatrix} \Psi & 0 \\ 0 & \Psi \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta}_u \\ \boldsymbol{\theta}_v \end{bmatrix} \tag{3.6}$$

with $\boldsymbol{\theta}_u$ and $\boldsymbol{\theta}_v$ sparse. Several closely related approaches to this optimization are presented. The best performance is given by a BPDN-like procedure which simultaneously minimizes the ℓ_2 prediction error and promotes sparsity in both the Wavelet transform and gradient of $\mathbf{u}$ and $\mathbf{v}$. Let D_x and D_y be linear operators implementing first-order differentiation in the horizontal and vertical directions, respectively. To clarify the relationship with BPDN, temporarily let $\mathbf{y} = -\mathbf{d}_t$, $\Phi = \begin{bmatrix} \mathbf{d}_x^T & \mathbf{d}_y^T \end{bmatrix}$, $\mathrm{B} = \begin{bmatrix} \Psi & 0 \\ 0 & \Psi \end{bmatrix}$, and $\mathrm{D} = \begin{bmatrix} \mathrm{D}_x & \mathrm{D}_y \end{bmatrix}$. The optimization problem is then given as

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\| \mathbf{y} - \Phi \mathrm{B} \boldsymbol{\theta} \right\|_2 + \lambda_1 \left\| \boldsymbol{\theta} \right\|_1 \tag{3.7}$$

$$\hat{\mathbf{g}} = \underset{\mathbf{g}}{\operatorname{argmin}} \left\| \mathbf{y} - \Phi \mathrm{D}^\dagger \mathbf{g} \right\|_2 + \lambda_2 \left\| \mathbf{g} \right\|_1 \tag{3.8}$$

$$\hat{\mathbf{f}} = \underset{\mathbf{f}}{\operatorname{argmin}} \left\| \mathbf{f} - \mathrm{B} \hat{\theta} \right\|_2 + \lambda_3 \left\| \mathbf{f} - \mathrm{D}^\dagger \hat{\mathbf{g}} \right\|_2 . \tag{3.9}$$

λ_1, λ_2, and λ_3 are tuning parameters. This estimate is further refined by using the RANSAC robust estimation procedure [81] to eliminate outliers.

A very similar method is proposed in [80]. The optimization problem to be solved is given as

$$\hat{\mathbf{g}} = \underset{\mathbf{g}}{\operatorname{argmin}} \left\| \mathbf{y} - \Phi \mathrm{D}^\dagger \mathbf{g} \right\|_1 + \lambda_4 \left\| \mathbf{g} \right\|_1 \tag{3.10}$$

Only sparsity in the gradient of optical flow is promoted, encouraging a piecewise-constant optical flow field. Notably, ℓ_1 prediction error is minimized, effectively encouraging sparsity in $\boldsymbol{\eta}$.

In [80] both Shen and Wu's method [62] and Han's method [80] are tested on the Middlebury optical flow dataset [4]. The average angular error (AAE) and endpoint error (EPE) are evaluated for four different stereo image pairs; it is found that Han's method results in lower errors for all images. The RANSAC algorithm which forms the final step of Shen and Wu's method is not used in this comparison; it is uncertain which algorithm would produce lower error if all steps were performed.

3.2 Classification & Detection Algorithms

Classifying the content of images is a classic problem in computer vision and forms an essential part of many other problems. My primary interest is in the two-class problem of target detection and its application to tracking algorithms. This section covers a selection of classification algorithms, approximately ordered from the simplest and earliest to most recent and most complex.

3.2.1 MACH Filter

The maximum average correlation height (MACH) filter is a well-known approach to offline training of a correlation-based filter to optimally detect a single object in video[82, 83]. A MACH filter is trained using examples from the positive (target) and negative (noise or background) classes. In its most basic form the MACH filter is defined (in the frequency domain) by:

$$\mathbf{H} = \left(D_y + S_x\right)^{-1} \mu_x \tag{3.11}$$

where D_y is the power spectral density of the noise examples and S_x is the similarity matrix of the true examples. S_x and D_y are diagonal matrices. μ_x is the mean of the target examples. Choosing a filter $\mathbf{h}$ as in (3.11) maximizes the average correlation height of the detection peaks while minimizing the average similarity measure

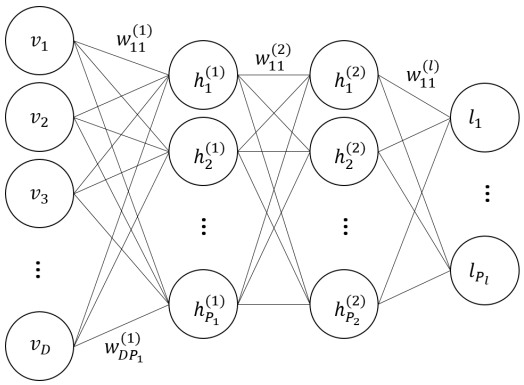


Figure 3.2: Deep Boltzmann Machine with Two Hidden Layers.

(ASM) between the targets and noise [84]. Several modifications and variations on the MACH filter exist, including, the optimal trade-off MACH (OT-MACH) filter [85, 86], which also optimizes output noise variance and average correlation energy.

3.2.2 Deep Boltzmann Machines

The deep Boltzmann machine (DBM) is a network-based bio-inspired computational model. Unlike traditional neural networks, Boltzmann machines are undirected networks and can be run in reverse to generate example data. They are also stochastic: a node's inputs do not define its output but rather a probability distribution over its possible outputs. The DBM expands on the Boltzmann machine concept by adding an arbitrarily large number of layers to the network and defining an unsupervised layer-by-layer pre-training step. Figure 3.2 gives a visual representation of a DBM. A brief sketch of DBM evaluation and training is given here; I follow the conventions and definitions of [87] and invite readers to explore this work for a more complete treatment of the subject.

A standard single-layer restricted Boltzmann machine (RBM) consists of a visible layer $\mathbf{v} \in \{0,1\}^{D\times 1}$ and a hidden layer $\mathbf{h} \in \{0,1\}^{P\times 1}$ of nodes. The values of the nodes in each layer are conditionally independent given the value of the other layer,

with conditional probabilities given by

$$p(h_j = 1|\mathbf{v}, \mathbf{h}_{-j}) = \sigma\left(\sum_i^D w_{ij} v_i\right) \tag{3.12}$$

$$p(v_i = 1|\mathbf{v}, \mathbf{h}_{-i}) = \sigma\left(\sum_j^P w_{ij} h_j\right). \tag{3.13}$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the logistic function. This conditional independence makes evaluation of the RBM computationally tractable. The RBM seeks to model a probability distribution over $\mathbf{v}$ using the weights W and the values of the hidden layers $\mathbf{h}$.

Exact maximum likelihood training of the RBM is not feasible, but contrastive divergence (CD) learning builds a model of the training dataset by minimizing the difference between a training sample v_i and its reconstruction $\tilde{v}_i$ [88].

The RBM concept above is extended to multiple layers by greedy layer-by-layer pretraining. The first layer weight matrix $\mathrm{W}^{(1)}$ is trained using CD. Then, the hidden-layer $\mathbf{h}_1$ corresponding to the first layer of the network is used as training input to determine the second-layer weights $\mathrm{W}^{(2)}$. This may be continued for an arbitrary number of hidden layers. After the pre-training, backpropagation algorithms may be used to fine-tune the network weights.

3.2.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) have risen to prominence in computer vision after showing competitive or best-in-class performance on a wide variety of difficult problems. The CNN may be viewed as a feedforward neural network which uses several tricks to exploit prior knowledge of the behavior of many naturally occurring signals, including digital images. In this section, a basic overview of the structure of CNNs is given and their training is discussed.

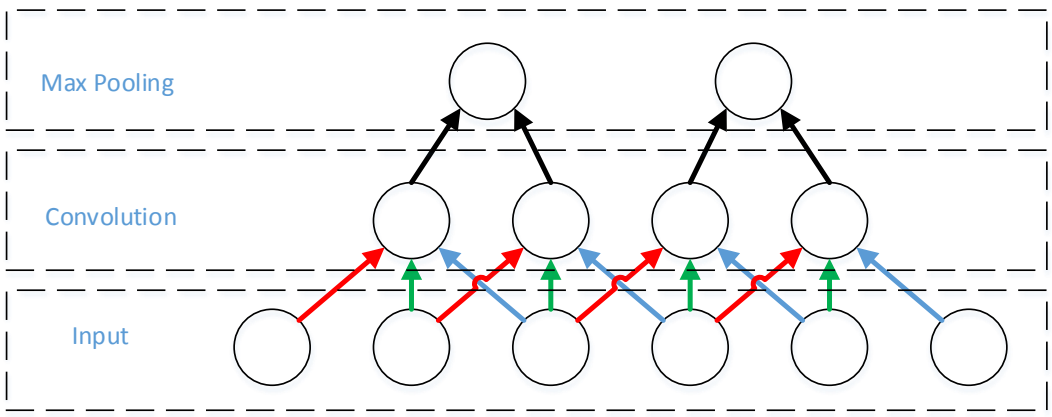


Figure 3.3: Convolutional Neural Network Architecture.

CNNs mitigate the problem of overfitting by weight-sharing and max-pooling, both of which reduce the number of parameters in the computational model while still modeling complex input-output relationships. Figure 3.3 gives a one-dimensional example of a small linear convolutional layer. Each hidden-layer neuron is connected to only a small neighborhood of the input layer, in this case 3 samples wide. Already, the number of weights has been reduced from DP_1 to $3P_1$. The weights are further shared between neurons, so that every node performs the same inner product on its neighborhood; this weight sharing is shown in Figure 3.3 by the colors of the arrows. The number of model parameters is now only 3. In addition to the massive reduction in model parameters, the P_1 inner products, each requiring D multiplications, are replaced by a single convolution operation. With the use of the FFT, the complexity of this operation has been reduced to $D \log D$. Convolution is, of course, a linear operation; nonlinearity must be added to the network. In a typical CNN this is accomplished via max-pooling, again shown in Figure 3.3. In max-pooling, an input image's pixels are simply binned into a grid, and the maximum of each bin is

passed to the next layer. In this way, max-pooling functions as a sort of nonlinear downsampling.

3.2.4 Deep Learning for Compressed Sensing Reconstruction

Machine learning, and particularly deep learning, may be used to reconstruct compressively sensed images with high performance. In [89], a restricted Boltzmann machine is used as a prior for CS reconstruction. The case of sparse $\mathbf{x}$ (i.e. $\Psi = \mathrm{I}$) is considered, but results can easily be extended to the case of dense $\mathbf{x}$ and sparse $\boldsymbol{\theta}$. This reconstruction method is tested on the MNIST digit data and shows performance near that of a "support oracle" algorithm which is given the support of the signal. This is possible because the RBM is generative and attempts to learn the probability distribution of the training data. A 2-layer RBM consisting of binary nodes is joined to a Bernoulli-Gaussian prior on $\mathbf{x}$, which is then reconstructed with AMP. In this way, the algorithm can be viewed as a more sophisticated form of re-weighted CS reconstruction algorithms [90] or model-based CS [18].

The system model consists of a Markov chain $\mathbf{h} \to \mathbf{v} \to \mathbf{x} \to \mathbf{y}$ with conditional probabilities given as

$$\mathbf{h}|\mathbf{v} \sim \text{Bern}\left(\sigma\left(\mathrm{W}^{(1)^T}\mathbf{v}\right)\right) \tag{3.14}$$

$$\mathbf{v}|\mathbf{h} \sim \text{Bern}\left(\sigma\left(\mathrm{W}^{(1)}\mathbf{h}\right)\right) \tag{3.15}$$

$$p(x_i|v_i) = \begin{cases} \delta(x_i), & v_i = 0 \\ \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp\left[\frac{x_i^2}{2\sigma_x^2}\right], & v_i = 1 \end{cases} \tag{3.16}$$

$$\mathbf{y}|\mathbf{x} \sim \mathcal{N}\left(\Phi\mathbf{x}, \sigma_w^2\mathrm{I}\right). \tag{3.17}$$

That is, the hidden layer $\mathbf{h}$ and visible layer $\mathbf{v}$ are related as in the binary RBM described by (3.12), while the unknown data vector $\mathbf{x}$ and measurement vector $\mathbf{y}$ are

related by the standard CS sensing operation in AWGN. $\mathbf{x}$ is independently but not identically distributed according to a BG prior, taking non-zero values with element-wise probability given by $\Pr(x_i \neq 0) = \Pr(v_i = 1)$.

3.2.5 Previous Work in Reconstruction-Free Classification

Relatively little work exists in the field of direct CS inference; most effort is concentrated on reconstruction algorithms. Researchers at Rice University have developed a correlation-based technique which takes advantage of preservation of inner products under the RIP [10]. This approach, known as the smashed filter, is analogous to the non-compressive matched filter technique. I show in Section 5.1.1 that in a visual tracking scenario, the 2-D cross correlations used by the smashed filter can be performed efficiently using the FFT [6]. I call this approach the fast smashed filter, a name which highlights its compressive nature, computational efficiency, and close relationship to the matched filter. Using maximum average correlation height (MACH) filters trained on multiple targets, the algorithm is able to track vehicles in stationary surveillance video. In [91], a spatio-temporal MACH-based smashed filter is used for reconstruction-free activity recognition. In [92], another template learning approach, the Maximum Margin Correlation Filter, is used to perform face recognition tasks. This approach outperforms a reconstruct-first paradigm at low sensing rates, where reconstruction algorithms fail.

In [93] a sensing matrix Φ is chosen using secant projections of training data. This ensures that isometry is preserved on a training dataset, rather than on the set of all k-sparse vectors. The secant projections approach outperforms the smashed filter with random Φ when noise is present.

Much of the most relevant published research comes from the closely related field of random projections. In [94], It is shown that random projections outperform principal

component analysis (PCA) in preserving Euclidean distances between 50×50 image patches. Distances are well-preserved at only 10 random projections, equivalent to a sensing rate of 0.4%. This result is attributed to the Johnson-Lindenstrauss lemma, which is closely related to the RIP given in (2.4). In [95], random projections are used to perform dimensionality reduction before data are input to an artificial neural network. This system is used for malware detection rather than image classification and shows a minimum two-class error rate at a sensing rate of 2.2%. These results are encouraging and help to motivate our approach. My application differs in that random projections are assumed to be performed directly by a compressive camera. That is, I target applications in which the algorithm only has access to $\mathbf{y}$, with $\mathbf{x}$ available only through a computationally expensive reconstruction step.

The approach of classification by sparse representation also deserves mention [96, 97, 98, 13]. All of these algorithms classify a measurement by expressing it as a combination of training vectors. In this way, the training dataset forms a (possibly over- or undercomplete) basis for the vector space in which the measurement lies. Classification can then be performed based on this new representation. Although it is mathematically related to CS, classification by sparse representations requires access to the uncompressed representation of a signal to be classified.

3.3 Tracking Algorithms

The problem of detecting and tracking known objects in images and video is a classic and well-studied problem in computer vision. However, adapting state-of-the-art techniques to the CS domain is non-trivial and may not be feasible for many techniques. This section focuses on tracking work which is most readily adaptable to the CS domain.

A typical visual tracking system follows a pattern in which input video frames are first preprocessed to increase image quality or enhance characteristics of the target. The image, or one or more regions of interest within the image, are then processed by a feature extraction algorithm, which converts the image into a lower-dimensional feature vector. This feature vector forms the input for the detection stage, which outputs an estimate of the confidence that a target is present at a certain location within the image. Finally, likelihood estimates across frames are fused, forming an overall estimate of the location of the target. The direct CS tracking paradigm covered in Chapter 5 presents problems mainly in the preprocessing and feature extraction stages, since unless reconstruction is performed, access to individual pixels is lost. Once features are extracted, tracking is able to proceed normally.

Several approaches in the literature apply sparsity or ℓ_1 based methods to the visual tracking problem. In [99] an OMP algorithm is used to extract sparse representations for visual tracking. In [100] a sparse sensing matrix is used to extract features from image regions in an approach similar to the famous Viola-Jones paradigm [101]. However, neither of these approaches operates on compressive data. Rather, they start from conventional images and apply compressive techniques for feature extraction.

3.3.1 State-Space Model of the Tracking Problem

This section introduces the system model used to describe the Kalman and particle filter algorithms. A system's time-varying state vector $\mathbf{q}_t$ is estimated from a set of measurements $\mathbf{y}_t$. $\mathbf{q}$ typically contains an estimate of target location and may also track parameters such as target speed, acceleration, scale, or pose. Optionally, the model may include a system input $\mathbf{u}_t$. In a tracking setting, this input vector could, for instance, account for known sensor platform motion. The state evolves according

to the hidden Markov model (HMM)

$$\mathbf{q}_t \sim p_{\mathbf{q}_t|\mathbf{q}_{t-1},\mathbf{u}_t}(\mathbf{q}_t|\mathbf{q}_{t-1},\mathbf{u}_t) \tag{3.18}$$

$$\mathbf{y}_t \sim p_{\mathbf{y}_t|\mathbf{q}_t}(\mathbf{y}_t|\mathbf{q}_t). \tag{3.19}$$

where $p_{\mathbf{p}_i|\mathbf{p}_{i-1}}$ and $p_{\mathbf{y}_i|\mathbf{p}_i}$ are the probability density functions of the state and observed measurements, respectively. The goal of the tracking algorithm is to estimate a sequence of states $\mathbf{q}_0, \mathbf{q}_1, ...$ as a function of the corresponding measurements $\mathbf{y}_1, \mathbf{y}_2, ...$ Because of the structure of the HMM, the conditional probability of states given measurements at time t is given by the recursive relation

$$p(\mathbf{q}_1, ..., \mathbf{q}_t|\mathbf{y}_1, ..., \mathbf{y}_t) = \frac{p(\mathbf{q}_t|\mathbf{q}_{t-1})p(\mathbf{y}_t|\mathbf{q}_t)}{p(\mathbf{y}_n|\mathbf{y}_1, ..., \mathbf{y}_{t-1})} p(\mathbf{q}_1, ..., \mathbf{q}_{t-1}|\mathbf{y}_1, ..., \mathbf{y}_{t-1}) \tag{3.20}$$

$$\propto p(\mathbf{q}_t|\mathbf{q}_{t-1})p(\mathbf{y}_t|\mathbf{q}_t)p(\mathbf{q}_1, ..., \mathbf{q}_{t-1}|\mathbf{y}_1, ..., \mathbf{y}_{t-1}). \tag{3.21}$$

In (3.20), $p(\cdot)$ denotes the probability density function; subscripts have been omitted for clarity and concision. This structure separates the estimate of current state $\mathbf{q}_t$ from all past measurements. Only the current measurement $\mathbf{y}$ and statistics of $\mathbf{q}_{t-1}$ are needed to calculate it.

3.3.2 Kalman Filter

The Kalman filter [102] gives an optimal estimate of $p(\mathbf{q}_0, \mathbf{q}_1, ...|\mathbf{y}_1, \mathbf{y}_2, ...)$ for the specific case of linear state updates and measurements in additive Gaussian noise. The general state update equation of (3.18) can be simplified to a matrix form given by

$$\mathbf{q}_t = \mathrm{F}\mathbf{q}_{t-1} + \mathrm{B}\mathbf{u}_t + \mathbf{w}_t \tag{3.22}$$

$$\mathbf{y}_t = \mathrm{H}\mathbf{q}_t + \mathbf{v}_t \tag{3.23}$$

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\mathbf{w}}) \tag{3.24}$$

$$\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\mathbf{v}}). \tag{3.25}$$

Under these conditions, the *a priori* state estimate $p(\mathbf{q}_k|q_{k-1})$ and the posterior probability $\mathbf{p}(\mathbf{q}_k|\mathbf{q}_{k-1}, \mathbf{y}_{k-1})$ are also Gaussian. The prior mean and covariance are given by

$$\hat{\mathbf{q}}_{t|t-1} = \mathrm{F}\hat{\mathbf{q}}_{t-1|t-1} + \mathrm{B}\mathbf{u}_t \tag{3.26}$$

$$\Sigma_{\mathbf{q}_{t|t-1}} = \mathrm{F}_t \Sigma_{\mathbf{q}_{t-1|t-1}} \mathrm{F}_t^T + \Sigma_{\mathbf{w}}. \tag{3.27}$$

From these quantities and the new measurement $\mathbf{y}_t$ the posterior mean and covariance are calculated. This is given in terms of the measurement residual $\tilde{\mathbf{y}}_t = \mathbf{y}_t - \mathrm{H}\mathbf{q}_{t|t-1}$ and *Kalman gain* K as

$$\hat{\mathbf{q}}_{t|t} = \hat{\mathbf{q}}_{t|t-1} + \mathrm{K}\tilde{\mathbf{y}}_t \tag{3.28}$$

$$\Sigma_{\mathbf{q}_{t|t}} = (\mathrm{I} - \mathrm{KH})\, \Sigma_{\mathbf{q}_{t|t-1}}. \tag{3.29}$$

The Kalman gain is chosen to minimize the mean square error of $\hat{\mathbf{q}}_{t|t}$; (3.29) is valid only for this optimal K. The optimal Kalman gain is given by

$$\mathrm{K} = \Sigma_{\mathbf{q}_{t|t-1}} \mathrm{H}^T \Sigma_{\tilde{\mathbf{y}}_k}^{-1} \tag{3.30}$$

where $\Sigma_{\tilde{\mathbf{y}}_k} = \mathrm{H}\Sigma_{\mathbf{q}_{t|t-1}}\mathrm{H}^T + \Sigma_{\mathbf{v}}$ is the covariance matrix of the measurement residual $\tilde{\mathbf{y}}$.

All discussion above has assumed time-invariance of the system. The algorithm may be extended by simply replacing F, B, H, $\Sigma_{\mathbf{w}}$, and $\Sigma_{\mathbf{v}}$ by their values at time t. The Kalman filter can be used even in non-Gaussian noise. However, in this case it will not produce exact marginal probabilities.

3.3.3 Particle Filter

Particle filters are a family of algorithms for estimation of posterior probability densities. Unlike the Kalman filter and related algorithms, the particle filter is a

Monte Carlo method and is able to handle arbitrary distributions. This section describes the basic vocabulary of a particle filter and details its use in visual tracking.

The particle filter estimates a time-varying state $\mathbf{p}$ given noisy measurements $\mathbf{y}$. The state $\mathbf{p}$ evolves according to the Markov model given in (3.18). When these PDFs are non-Gaussian or state transition equations are nonlinear, there is in general no closed-form solution for the posterior distribution of $\mathbf{p}$. However, the particle filter gives a method to sample from the distribution. q particles are generated i.i.d according to an initial density $p_{\mathbf{p}_0}(\mathbf{p})$. At each time step, a new set of states is predicted from the previous frame, the measurement $\mathbf{y}_i$ is observed and each particle $\mathbf{p}_j$ is assigned a weight $w_{j,i}$ based on the particle's likelihood,

$$w_{j,i} = p(\mathbf{y}_i|\mathbf{x}_i)w_{j,i-1}. \tag{3.31}$$

In this way, a particle's weight grows (relative to its neighbors) during frames when its likelihood is large and shrinks during frames when its likelihood is small.

Optionally, particles may be resampled during some or all frames. This is accomplished by normalizing all the weights so that they sum to one, and forming a categorical distribution over the current particles based on these weights. A new set of particles is then drawn from this categorical distribution. Resampling is a strategy to avoid degeneracy in the particle filter, where the weights of most particles quickly approach 0 and contribute nothing to the estimation. On the other hand, when noise is small, resampling too often can lead to the estimation being dominated by one or a small number of possible states.

3.4 Summary

This chapter covered several relevant problems and solutions from the image processing and computer vision literature. Compression of images, through the wavelet

transform, and video, through motion estimation, was discussed. Approaches to detection and classification of images were then covered, from relatively simple template models to more sophisticated deep learning algorithms. Finally, the video tracking problem was introduced, and a selection of tracking algorithms was presented. These algorithms are a starting point for approaches to higher-level inference from CS images.

Chapter 4

COMPRESSIVE VIDEO RECONSTRUCTION INCORPORATING INTER-FRAME CORRELATIONS

Reconstruction of compressively sensed video remains an open problem. Progress has been made in reconstruction of static images [22], but to my knowledge no currently published algorithm fully utilizes prior knowledge of frame-to-frame correlations and compressibility to improve reconstruction. Previous efforts have made restrictive assumptions such as a static background [57, 59] or constant optical flow across the entire image [60]. Clearly, a reconstruction algorithm which fully utilizes dense optical flow information to perform multi-frame video reconstruction would be a valuable tool and would bring compressive sensing video hardware closer to commercialization.

In this section, a method of reconstructing compressively sensed video frames in the presence of known optical flow is presented in Section 4.1 and applied to an existing dataset in Section 4.3.1. This approach is extended to the case of unknown but constant motion in Section 4.2, with results presented in Section 4.3.2. In both cases, significant performance improvements are achieved by incorporating optical flow information in reconstruction. It is hoped that this work represents a step toward fast and reliable estimation of dense optical flow and wide adoption of compressive sensing video technology.

Two image vectors, $\mathbf{x}_1$ and $\mathbf{x}_2$, are known to be compressible. These images are measured with sensing matrix Φ, resulting in measurements $\mathbf{y}_1$ and $\mathbf{y}_2$. That is,

$$\mathbf{y}_1 = \Phi\mathbf{x}_1 = \Phi\Psi\boldsymbol{\theta}_1 \tag{4.1}$$

$$\mathbf{y}_2 = \Phi\mathbf{x}_2 = \Phi\Psi\boldsymbol{\theta}_2, \tag{4.2}$$

where $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are k-sparse vectors forming compressed representations of $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively.

In the case of compressively sensed video, it is also known that adjacent frames $\mathbf{x}_1$ and $\mathbf{x}_2$ are related by some linear optical flow operator $\mathrm{F}_{\mathbf{u},\mathbf{v}}$, where $\mathbf{u}$ and $\mathbf{v}$ are optical flow vectors in the x- and y-directions, respectively. That is,

$$\mathbf{x}_2 = \mathrm{F}_{\mathbf{u},\mathbf{v}}\mathbf{x}_1 + \boldsymbol{\eta}, \tag{4.3}$$

where $\boldsymbol{\eta}$ is an error term representing the part of $\mathbf{x}_2$ which is independent of $\mathbf{x}_1$. In natural images, $\boldsymbol{\eta}$ is sparse, a fact which is exploited by modern video codecs.

4.1 Reconstruction with Known Optical Flow

In the special case where optical flow vectors $\mathbf{u}$ and $\mathbf{v}$ are known *a priori*, reconstruction is relatively straightforward and the benefits of incorporating optical flow in a reconstruction algorithm are clear. The case of reconstruction from two frames using forward prediction is presented here, although this method is easily extended to include larger numbers of frames and backward prediction. In this case, the underdetermined system to be solved may be jointly given in matrix form by combining (4.1), (4.2), and (4.3), and is described by

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} \Phi\Psi & 0 \\ \Phi\mathrm{F}_{\mathbf{u},\mathbf{v}}\Psi & \Phi \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta} \\ \boldsymbol{\eta} \end{bmatrix}. \tag{4.4}$$

$\boldsymbol{\theta}_1$ can then be recovered from $\mathbf{y}_1$, $\mathbf{y}_2$, $\mathbf{u}$, and $\mathbf{v}$ by solving the convex optimization problem

$$\underset{\boldsymbol{\theta},\boldsymbol{\eta}}{\text{minimize}} \left\| \begin{bmatrix} \Phi\Psi & 0 \\ \Phi \mathrm{F}_{\mathbf{u},\mathbf{v}} \Psi & \Phi \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta} \\ \boldsymbol{\eta} \end{bmatrix} - \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} \right\|_2 + \tau \left\| \begin{bmatrix} \boldsymbol{\theta} \\ \boldsymbol{\eta} \end{bmatrix} \right\|_1 . \tag{4.5}$$

Equation (4.5) differs from the optimization problem used in [58] in several important ways. First, optical flow appears as part of the objective function rather than as a constraint, allowing only $\boldsymbol{\theta}_1$ to be estimated and reducing the size of the estimation problem. Second, the inclusion of $\|\boldsymbol{\eta}\|_1$ in the objective function creates sparsity in the optical flow error η. This should allow the more accurate reconstruction of small objects and edges in the final image. The algorithm also differs from that of [60] in that no additional special properties of the sensing matrix are required to perform motion estimation.

The case of $\boldsymbol{\eta} \approx \mathbf{0}$ will also be examined; this is a reasonable assumption for several useful applications of compressive sensing. If $\boldsymbol{\eta} \approx \mathbf{0}$, (4.5) becomes

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \left\| \begin{bmatrix} \Phi\Psi \\ \Phi \mathrm{F}_{\mathbf{u},\mathbf{v}} \Psi \end{bmatrix} \boldsymbol{\theta} - \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} \right\|_2 + \tau \|\boldsymbol{\theta}\|_1 . \tag{4.6}$$

4.2 Reconstruction with Constant but Unknown Optical Flow

An interesting special case arises when optical flow is known to be constant across an entire image; that is, when $\mathbf{u}$ and $\mathbf{v}$ are constant vectors. This is generally the case for aerial surveillance video, an important potential application of compressive sensing video technology. In this case, only a single frame must be reconstructed in order to estimate flow. This is possible because of the ability to perform correlations in the compressive domain [103].

Let $\hat{\mathbf{x}}_1$ be an estimate of $\mathbf{x}_1$ acquired by solving (2.6). $\mathbf{u} = u\mathbf{1}$ and $\mathbf{v} = v\mathbf{1}$ may then be estimated maximizing the estimated correlation with a matched filter:

$$\underset{u,v}{\text{maximize}} \ \mathbf{y}_2^\top \left(\Phi\Phi^\top\right)^{-1} \Psi \mathrm{F}_{u\mathbf{1},v\mathbf{1}} \hat{\mathbf{x}}_1. \tag{4.7}$$

Once $\mathbf{u}$ and $\mathbf{v}$ have been determined, two-frame reconstruction may be carried out as in (4.6) or (4.5). Although (4.7) appears straightforward, several practical considerations arise when attempting to apply it. Equation (4.7) is, in general, a non-convex problem and is relatively expensive to compute. However, if u and v are quantized, an exhaustive search is often feasible, for instance if the image size is small or the motion vector magnitude $\|[u, v]\|_2$ is bounded. Several efficient search algorithms also exist to approximately determine optical flow; Zhu and Ma's diamond search algorithm [79] is used in this work . The correlation is also heavily affected by the way the flow operator handles the edges of the image. In order to avoid this concern, $\hat{\mathbf{x}}_1$ is first windowed before applying the optical flow transform. It was also empirically determined that reconstruction was superior when optical flow was non-zero, even if the ground truth optical flow was zero. Because of this, u and v are forced to take non-zero values: if a zero value is found to be optimal, the next-best value is chosen instead.

4.3 Results

4.3.1 Known Optical Flow

The algorithm in Section 4.1 was tested using the Middlebury optical flow dataset's "Venus" image, a stereo image for which known ground truth motion vectors are provided [4]. The image was preprocessed by shrinking by a factor of 2 and cropping to size 128×128. Sensing was performed using an IID Gaussian sensing matrix

using $n = 4915$ measurements, for a sensing rate of 0.3. Figure 4.1 shows the results of reconstruction of a single frame, along with simultaneous reconstruction of both frames using known optical flow information. The PSNR was increased from 21.12 dB to 23.80 dB by the use of optical flow information. An improvement in the perceptual quality of the reconstruction is also visible. Figure 4.2 shows PSNR vs. number of sensors for single-frame and 2-frame reconstruction. The benefits are clear, particularly at low sensing rates.

(a) ground truth (b) single-frame (c) two-frame

Figure 4.1: Reconstruction with Known Optical Flow for 128×128 Image with Sensing Rate $r = 0.212$.

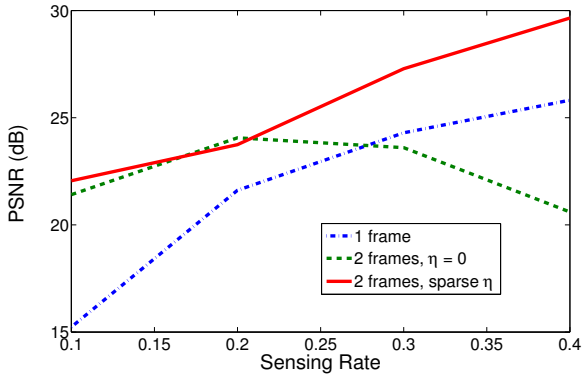


Figure 4.2: PSNR of Reconstructed Signal as a Function of Sensing Rate.

(a) ground truth (b) single-frame (c) two-frame

Figure 4.3: Comparison of Single-Frame and Two-Frame Reconstruction for Constant Optical Flow.

4.3.2 Constant Optical Flow

The problem of reconstruction under constant but unknown optical flow was evaluated using the PETS2000 dataset. A segment of the video with no activity was selected and used to generate a pair of 128×128 test frames, represented in vector form as $\mathbf{x}_1$ and $\mathbf{x}_2$, in which $\mathbf{x}_2$ experienced a small shift relative to $\mathbf{x}_1$. $\mathbf{y}_1$ and $\mathbf{y}_2$ were generated using $n = 4915$ sensors, and the algorithm described in Section 4.2 was used to reconstruct the image. Figure 4.3 compares single-frame and two-frame optical flow based reconstruction algorithms. The two-frame reconstruction is clearly perceptually superior and has a PSNR of 27.21 dB, a 2.81 dB improvement over the single-frame reconstruction. Reconstruction was performed both with a term $\boldsymbol{\eta}$ for imperfect optical flow (Equation 4.5) and without it (Equation 4.6). The $\boldsymbol{\eta}$ term was found to be necessary in order to achieve improved PSNR relative to the single-frame method; only results from this method are presented.

4.4 Significance of results

Optical flow is a key element of modern video codecs, but has not been fully used in reconstruction algorithms for compressively sensed video sequences. This is mainly due to the difficulty of estimating optical flow without an existing high-quality reconstruction of the image. However, once optical flow is known, clear benefits are available from utilizing optical flow information. This was shown in Section 4.3.1, where the problem of reconstruction under known optical flow was presented as a convex optimization.

A potential work-around to the problem of optical flow estimation without a high-quality image was described in Section 4.2. In the special case considered, optical flow is known to be constant across the entire image.This allows optical flow in subsequent frames to be estimated without reconstruction from a single reconstructed frame. I believe that this method is likely to be applicable to the case of small moving targets in a moving frame, as the optical flow error parameter $\boldsymbol{\eta}$ should be able to accommodate them. However, this claim is speculative.

It is hoped that this work will serve to advance the development of compressive sensing video cameras by integrating existing ideas of optical flow based reconstruction from conventional video codecs. A robust reconstruction algorithm with high performance could dramatically reduce the cost and data rate requirements of video sensing systems at a wide range of light wavelengths.

Chapter 5

RECONSTRUCTION-FREE TRACKING FROM CS IMAGERS

The advantages of compressive sensing come at the cost of higher computational requirements at the receiver, where compressively sensed video data must be mathematically reconstructed in order to display video sequences in a human-readable form. However, in some cases, such as automated surveillance or navigation, a system may not require a human to observe a video at all. In these cases it may be possible to perform target recognition and other computer vision tasks directly on the compressively sensed data, without reconstructing video frames. In this section, a method for target detection and tracking without reconstruction is presented. The algorithm is tested on moving vehicles from the PETS2000 and CDNET 2012 datasets, and testing results are presented.

Section 5.1 gives a detailed description of the direct tracking algorithm. In Section 5.2 the algorithm is tested on the PETS2000 and CDNET 2012 datasets and testing results and tracker performance are described. Finally, Section 5.3 discusses the significance of the results.

5.1 Tracking Algorithm

The prototype compressive tracker is a particle filter with importance sampling updates handled by correlations with a MACH filter. Figure 5.1 shows a high-level block diagram showing the key components of the system and their interaction. The algorithm implements tracking-by-detection using by using a particle filter to maintain an estimate of both the target's likelihood and location, based only on data from a compressive sensing camera. Correlations are estimated directly from the measured

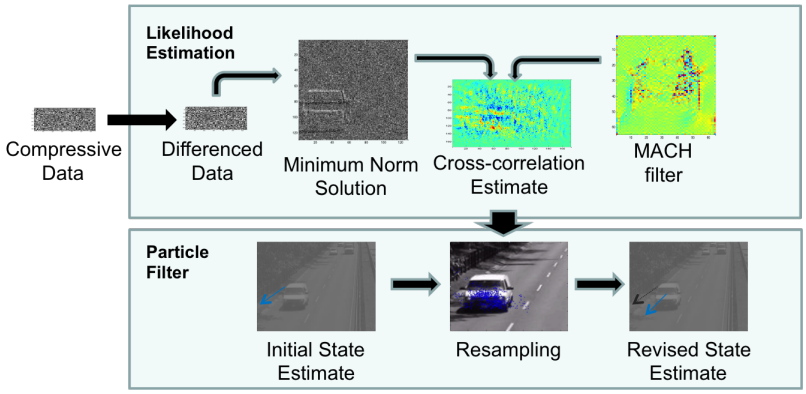


Figure 5.1: High-Level Block Diagram of Compressive Tracker

frame data $\mathbf{y}_i$ using a fast smashed filter, discussed in Section 5.1.1. The smashed filter produces noisy correlation estimates, reducing the effective Signal-to-noise ratio (SNR) of the detector. In order to mitigate these effects, the track-before-detect paradigm is implemented in the particle filter; the details of this approach are covered in Section 5.1.2.

5.1.1 Compressive-Domain filtering: The Fast Smashed Filter

The smashed filter [10] is the compressive-domain analog of the well-known matched filter. Like the matched filter, it is the maximum likelihood classifier for the case when the observed data for each class come from a known manifold (e.g. shifted versions of a template for the case of the matched filter).

Target classification is to be performed based on compressive measurements $\mathbf{y} = \Phi(\mathbf{x} + w)$ of the spatial domain image $\mathbf{x}$. Let each element of target class $\mathcal{C}_j$ lie on a manifold defined by the function f_j and parametrized by ρ:

$$\mathbf{x} = f_j(\rho_j). \tag{5.1}$$

This leads to the following formulation of the Generalized Maximum Likelihood Classifier (GMLC):

$$\hat{j} = \underset{j}{\operatorname{argmax}}\, p(\mathbf{y}|\hat{\theta}_j, \mathcal{H}_i) \tag{5.2}$$

Where $\hat{\rho}$ is the maximum likelihood estimate of the parameter vector ρ under hypothesis $\mathcal{H}_j$:

$$\hat{\rho}_j = \underset{\rho}{\operatorname{argmin}} \|y - \Phi f_j(\rho)\|_2^2 \tag{5.3}$$

Next, consider a simplistic description of the problem of visual classification. Images to be classified are shifted noisy versions of some template $\mathbf{s}_j$, and the compressive measurement y is given by

$$\mathbf{y} = \Phi(\mathbf{x} + \mathbf{w}) = \Phi(\mathbf{s}_{j,u,v} + \mathbf{w}) \tag{5.4}$$

where u and v are the x- and y-locations of the target, respectively.

When the $f_j(\theta_j)$ are shifted versions of each other as above, the smashed filter reduces to the following simple test statistic:

$$t = \mathbf{y}^T \Phi^\dagger \mathbf{s}_{j,u,v} = \mathbf{y}^T \left(\Phi\Phi^T\right)^{-1} \Phi \mathbf{s}_{j,u,v} \tag{5.5}$$

The smashed filter is typically viewed as projecting the template into the compressed domain and then performing an inner product with the signal to be classified. Indeed, when many images are to be processed with the same set of templates, this approach will typically be fastest since the compressed representation of each filter can be precomputed and stored. However, (5.5) may just as easily be interpreted as bringing the compressive measurement $\mathbf{y}$ into the spatial domain using least-squares estimation and then performing the same inner product. When $\mathbf{s}_{j,u,v}$ are shifted versions of each other, these inner products are a cross-correlation with the template, and computation may be dramatically sped up using the FFT. I refer to this formulation as the "fast smashed filter" to emphasize this fact.

As intuition would suggest, the correlation estimates provided by the fast smashed filter are noisy relative to those produced by the matched filter on the original image. They are also severely degraded relative to the matched filter operating on a reconstructed image. This performance degradation is quantified in Section 5.2.1.

5.1.2 Track-Before-Detect Particle Filter

After particles are resampled, the state of each particle is updated before the next time step in accordance with the chosen Markov model of state evolution. Figure 2 gives a high-level pseudocode description of the algorithm.

State evolution and likelihood model

I have described the general operation of a particle filter; In this section I will discuss the state evolution and likelihood update steps used for implementation of the tracker. In this implementation, a first-order 2-dimensional motion model with scaling is used. This is defined by a linear update step with additive noise, given as

$$\mathbf{p}_i = \mathrm{A}\mathbf{p}_{i-1} + \mathbf{w}_i \tag{5.6}$$

The state vector $\mathbf{p}$ is given by

$$\mathbf{p} = [x\ \dot{x}\ y\ \dot{y}\ c]^T \tag{5.7}$$

where (x, y) is the target location, $(\dot{x}, \dot{y})$ is the target velocity, and $\mathbf{c}$ is the logarithm of the target's scale. The update matrix A is given by

Algorithm 2 Particle Filter State Update Procedure.

Input: compressively sensed frames $\mathbf{y}_1, ..., \mathbf{y}_N$, initial target state $\mathbf{p}_0 = [x_0 \ y_0 \ \dot{x}_0 \dot{y}_0]^T$

Output: Estimated states $\mathbf{p}_i^{(j)}$, $i \in [1, N]$, $j \in [1, q]$

Initialize particle state $\mathbf{p}_1^{(1)} ... \mathbf{p}_1^{(q)} \sim p(\mathbf{p}|\mathbf{p}_0)$

Load target template $\mathbf{s}_{0,0}$

for $i = 1$ to N {for each frame} **do**

calculate smashed filter cross-correlation estimate C := $\text{IFFT}_2 \left(\text{FFT}_2 \left(\Phi^\dagger \mathbf{y}_i \right) \text{FFT}_2^* \left(\mathbf{s}_{0,0} \right) \right)$

for $j = 1$ to q {for each particle} **do**

Calculate likelihood $l_j := |C_{y_j, x_j}|$

end for

Resample to draw new particles $\mathbf{p}_{i+1}^{(1)} ... \mathbf{p}_{i+1}^{(q)} \sim \Pr(\mathbf{p}_{i+1}^{(j)} = \mathbf{p}_i^{(k)}) = \frac{l_k}{\sum_m l_m}$

for each particle $\mathbf{p}_i^{(j)}$ **do**

Update state for next frame $\mathbf{p}_{i+1}^{(j)} = \text{A}\mathbf{p}_{i+1}^{(j)} + \mathbf{w}$

end for

end for

$$\mathrm{A} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.8}$$

Likelihood updates are accomplished by cross-correlation with a template $\mathbf{s}$, estimated using the fast smashed filter described in Section 5.1.1. Let $\mathbf{s}_{x,y}$ denote the image consisting of the template inserted at location (x, y), with zeros elsewhere. Likelihood is then modeled by

$$p(\mathbf{p}|\mathbf{y}) \propto \left|\mathbf{y}^T \Phi^\dagger \mathbf{s}_{x,y}\right|. \tag{5.9}$$

Note that this is something of an abuse of the term "likelihood," since (5.9) does not correspond to any probability distribution on $\mathbf{y}$. This approach, however, is common in the visual tracking community.

Track-before-detect Implementation

Track-before-detect is a paradigm in detection wherein estimates of target state and presence/absence are maintained concurrently. This contrasts with the conventional tracking paradigm, in which a detection is declared at the same time as a target location estimate is first given. Track-before-detect may be viewed as a soft decision based approach to this problem. In this work, I adapt the track-before-detect approach given in [7].

In the particle filter, track-before-detect is implemented as follows. State is defined using a first-order motion model as in (5.7), with the addition of a binary "alive" state a. The state vector and update model then proceeds as follows.

$$\mathbf{p} = [\mathbf{p}_0^T \ a]^T \tag{5.10}$$

$$A = \begin{bmatrix} A_0 & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \tag{5.11}$$

$$\mathbf{p}_{i+1} = \begin{cases} A\mathbf{p}_i + \mathbf{w} & , a_i = 1 \\ [\mathcal{U}(\mathbf{0}, [w \ h]) \ \mathcal{N}(\mathbf{0}, \sigma_v^2 I) \ a_{i+1}]^T & , a_i = 0 \end{cases} \tag{5.12}$$

$$a_{i+1} = \begin{cases} \text{Bern}(P_b) & , a_i = 0 \\ \text{Bern}(1 - P_d) & , a_i = 1, \end{cases} \tag{5.13}$$

where $\mathbf{p}_0$ and A_0 are the state vector and update matrix given in (5.7) and (5.8), respectively. In other words, when $a = 1$ the particle is alive (i.e. target is present) and updates proceed as in (5.6); when $a = 0$, the particle is dead and updates sample the space of possible states. x- and y- positions are uniformly distributed over the entire input image, and velocities are normally distributed with variance σ_v^2. a follows a Markov model in which a dead particle is "born" with probability P_b and a living particle "dies" with probability P_d.

Likelihood estimation is also defined piecewise, based on whether a particle is alive or dead. The likelihood is given by

$$p(\mathbf{y}|\mathbf{p}) = \begin{cases} \left|\mathbf{y}^T \Phi^\dagger \mathbf{s}_{x,y}\right| & , a = 1 \\ \gamma & , a = 0 \end{cases} \tag{5.14}$$

Likelihood is calculated as normal for living particles and assigned a constant value γ for dead particles. γ is selected to be less than the correlation peak height for a true detection but greater than the peak height for a false detection; changing γ allows some control over the trade-off between detection rate and false alarm rate.

5.2 Experimental Results

The algorithm described in Section 5.1 was tested using simulations on several relatively simple visual datasets. Datasets were deliberately chosen for high MACH filter peak heights using a small number of filters; this allows the effect of direct CS domain processing to be separated from the merits of the MACH/Particle filter tracking approach. Two natural video sequences from the literature were chosen for testing; the PETS2000 [5] and CDNET 2012 "Highway" [11] sequences. Both video clips show vehicles moving against a static background under relatively constant lighting conditions, allowing the use of frame differencing. The choice of vehicles as a target also minimizes the number of templates needed, since vehicles are very nearly rigid objects.

5.2.1 Quantifying the Performance of the Smashed Filter

As mentioned in Section 5.1.1, the smashed filter suffers from increased noise relative to its non-compressive equivalent. For a given sensing rate r, SNR will be reduced by a factor of $\sqrt{r}$ [10], i.e. a 10 dB/decade SNR penalty is imposed due to compression. In this section, I briefly compare this theoretical prediction with results from a dataset of natural images.

In order to quantify the increased noise in the smashed filter test statistic, I use the CIFAR-100 image training database, which consists of 50,000 32x32 RGB images. Each image was converted to grayscale and normalized so that its mean was zero and its ℓ_2 norm was equal to 1. Each normalized frame $\mathbf{x}_i$ was then sensed using a pseudo-random orthonormal sensing matrix Φ, generating compressive measurement $t = \mathbf{y}_i = \Phi\mathbf{x}_i$. The estimated inner product $\mathbf{y}_i^T\Phi^\dagger\mathbf{x}_i$ was then calculated, as used

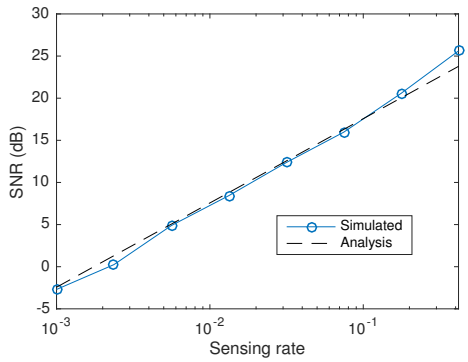


Figure 5.2: Average SNR of Smashed Filter for Varying Sensing Rate, CIFAR-100 Dataset

in the fast smashed filter test statistic. This was repeated for 10 different random drawings of Φ, and the variance over all images and all trials was calculated.

Fig. 5.2 shows the effect of sensing rate r on the effective SNR of the test statistic t_{smash}, defined as

$$SNR = \frac{\mathrm{E}^2\left[t_{smash}\right]}{\mathrm{E}\left[t_{smash}^2\right]}. \tag{5.15}$$

As expected, noise variance increases as sensing rate decreases, with a 10-fold increase in sensing rate r leading to a 10 dB improvement in SNR. Note that at high sensing rates, the simulated results appear to outperform the analysis in [10]. I believe this is because the $\mathbf{x}_i$ are merely compressible and not perfectly sparse. As a result, they contain a non-sparse component which appears as noise at low sensing rates but is more accurately represented at high sensing rates. Work is underway to more accurately quantify the noise characteristics of the smashed filter for compressible signals.

5.2.2 PETS2000

The algorithm was first tested on the PETS2000 surveillance video dataset[5]. The purpose of this test was to bring the detection and tracking system to the proof

Figure 5.3: PETS2000 Dataset. (a) Example Frame and (b) Difference Frame.

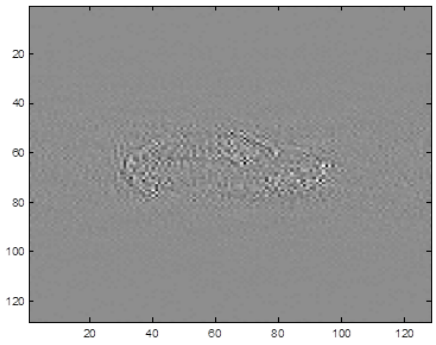


Figure 5.4: Trained MACH Filter for PETS2000 Dataset.

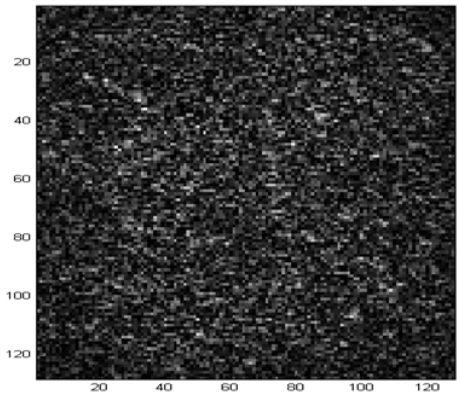


Figure 5.5: Increased Noise in Cross-Correlation Output due to Smashed Filter. Note that no peak from the target is visible.

Figure 5.6: Sample Frames from PETS2000 Dataset.

of concept stage of development by achieving detection and tracking at sensing rates comparable to those required to reconstruct an image at reasonable perceptual quality. This was accomplished, with the algorithm detecting the target at a 0.3 sensing rate.

The PETS2000 dataset was developed as a benchmark for outdoor person and vehicle tracking. In the dataset, several different cars and pedestrians cross a parking lot located in the camera's field of view. Because it includes a static background, it is a reasonable candidate for testing our algorithm. Vehicles are also seen only from a few limited angles and scales, making the detection problem more tractable. To further simplify matters, the same 201 video frames (frames 2700-2900 of the training sequence) were used for filter training and determining detector performance. I argue that reducing mismatch between the filter and the target in this way is valid, since our goal is to study the effect of the smashed filter relative to equivalent non-compressive tracker.

A single 128×128 pixel MACH filter was used to estimate target likelihood for the particle filter importance sampling step. The MACH filter was trained as follows: frames 2700-2900 of the differenced video sequence were cropped to a 64×64 pixel window containing only the target, centered as well as possible by hand. These cropped target regions were then zero-padded to 128×128 pixels and used as input to the MACH filter generator described in (3.11). Fig. 5.4 shows the spatial domain

representation of the trained MACH filter, which is identifiable as the outline of a vehicle.

Simulations were run on this dataset using the track-before-detect particle filter implementation described in Section 5.1.2.

The tracker successfully detected the target at a sensing rate of 0.3 (4915 compressive measurements for 16384 pixels). Figure 5.7 shows the tracker's RMSE over time for several different sensing rates. While the tracker was able to run on the PETS2000 dataset, the nature of the video sequence also served to highlight some limitations of the algorithm. Most importantly, the limitations of the MACH filter are evident, as it is only possible to cover a small range of the target's rotation with a single filter. Because of this, peak heights, and therefore SNR, for the detector are quite low, even in the non-compressive domain. One possible approach to mitigating this effect is to employ multiple filters, each covering a small range of target rotations. This leads to a trade-off in the number of filters, as increased performance in individual filters is balanced against increased false alarm rate across all filters.

Another limitation of the algorithm is its reliance on frame differencing. This limits the tracker to detecting only moving targets, as is evident in Figure 5.7. When the target vehicle stops and reverses direction, the tracker temporarily loses it and begins to "wander" over the image until the vehicle begins moving again. One possible approach to mitigating this effect is to maintain an estimate of the background and subtract this rather than subtracting the previous frame. This approach is currently being considered.

5.2.3 CDNET 2012 "Highway" Sequence

The CDNET 2012 dataset [11] is a benchmark dataset for testing motion detection and tracking algorithms. It consists of several short video sequences including both

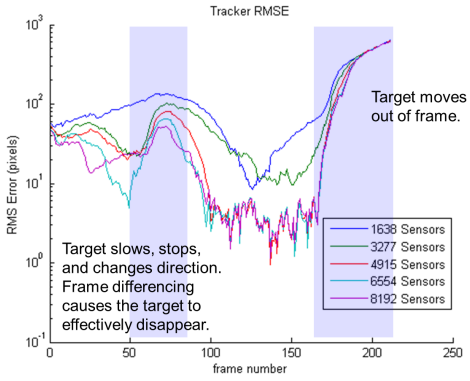
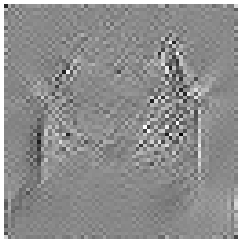


Figure 5.7: Tracker root mean square error on PETS2000 dataset.

Figure 5.8: Sample frames from CDNET 2012 highway video sequence

indoor and outdoor scenes, but I will concentrate on a single sequence, "Highway", for tracker testing. The highway sequence (Fig. 5.8) consists of two lanes of cars approaching the camera on a highway. 1700 RGB frames, each of size 320×240 pixels, are available; pixel-level annotations are provided for frames 470-1700. This sequence was chosen because it allows the use of a single MACH filter with excellent peak height; this presents a best-case scenario for tracking with a single template.

Because the video sequence contains multiple vehicle targets in most frames, no attempt was made to perform detection. Multi-target detection and tracking is an active area of research in and of itself and I expect multi-target tracking to be adaptable to the compressive domain with few modifications. Although detection was not formally tested, it was observed that once a target moved out of frame, the tracker quickly converged to a different target and began tracking it instead. This obser-



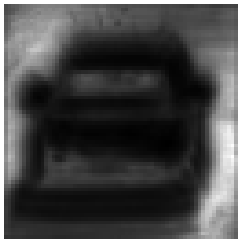


Figure 5.9: (a) MACH Filter and (b) SVM Filter Trained for CDNET 2012 Highway Dataset.

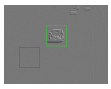


Figure 5.10: CDNET 2012 Highway Examples. (a) Example Frame, (b), Pixel-Level Ground Truth Annotation, (c) Difference Frame with Positive and Negative Windows Selected for Training.

vation, although not conclusive, is encouraging for the potential performance of a detection algorithm on this dataset.

Two different methods, MACH and linear SVM, were used to train a filter for cross-correlation. In both cases, frames 471-1700 of the video sequence were used for template training and frames 1-470 were used for testing. The CDNET 2012 dataset labels targets at the pixel level; targets were identified using blob analysis on the supplied annotation. A true example was generated by padding the target's bounding box by a factor of 1.5, cropping the resulting window, and resizing to 64×64 pixels. A false example of the same size was then generated by uniformly sampling the image windows which did not contain any pixels belonging to any target. This process is illustrated in Fig. 5.10.

Process noise $\mathbf{w}$ was normally distributed with covariance matrix

$$\Sigma_{\mathbf{w}} = \mathrm{diag}\left([5\ 5\ 1\ 1\ .05]\right). \tag{5.16}$$

Table 5.1: Reconstruction Parameters for GAMP Algorithm.

Output probability	$\mathcal{N}\left(\mathbf{y}, \sigma_y^2 \mathrm{I}\right)$
Output Variance σ_y^2	13.01
Input probability	Bernoulli-Gaussian
Input Bernoulli probability	0.005
Input Gaussian variance	500520

The CS tracker described above was compared with two other algorithms. First, a baseline non-compressive tracker was used. This approach applied the same tracking system directly to spatial domain image data, allowing virtually perfect and noise-free reconstructions relative to the compressive methods. The baseline method is thus a best-case scenario for the templates and tracking system used. Second, the reconstruct-then-track approach was examined. This method first reconstructed difference frames using the generalized approximate message passing (GAMP) algorithm [44] before performing cross-correlation with the template. Table 5.1 gives the parameters used for the GAMP reconstruction.

Figure 5.11 shows the target's ground truth path, a sample path from the tracker operating at $r = 0.1$ sensing rate, and the range of variation in 10 trials of the algorithm. Figure 5.12 shows tracker mean absolute error on the CDNET 2012 dataset as a function of sensing rate, and compares this against the baseline and GAMP reconstruction approaches. The GAMP tracker breaks down suddenly at a sensing rate of $r = 0.01$, while the smashed filter tracker with MACH filter shows much more graceful degradation. The smashed filter tracker with SVM filter underperforms the MACH tracker. Likewise, the Hadamard sensing matrix underperforms the Gaussian case. The state-of-the-art STRUCK tracker [104] achieves an MAE of 3.6 pixels

operating on non-compressive data. It slightly outperforms our algorithm, which achieves an MAE of 4.6 pixels at a sensing rate $r = 0.2$ and 6.2 pixels at $r = 0.005$.

Another metric for the perfomance of a tracker is the success rate. Success rate R_s, defined in [100], is the percentage of frames with overlap $o > 0.5$ between the ground truth and estimated bounding boxes. Overlap is in turn defined as the ratio of the intersection and the union between the ground truth and estimated bounding boxes:

$$o = \frac{\text{area}(B \cap \hat{B})}{\text{area}(B \cup \hat{B})}. \tag{5.17}$$

Success rate is chosen as a metric because it has the effect of normalizing over changes in scale while remaining simple to compute. Figure 5.13 shows the success rate of the three tracking algorithms (baseline/non-compressive, GAMP reconstruct-then-track, smashed filter) as a function of sensing rate. Again, the GAMP tracker no longer holds the target at sensing rates $r \leq 0.01$ while the smashed filter tracker continues functioning with success rate $R_s > 0.9$ at sensing rates as low as $r = 0.005$. Once again, the linear SVM underperforms the MACH filter based tracker. Because STRUCK tracks a constant-sized window, the success rate metric vastly understates its performance under scale changes and is omitted from the plot.

The tracking algorithm's runtime was compared with that of the GAMP reconstruct-then-track approach. Table 5.2 summarizes the speed results for these simulations. At a sensing rate of 0.01, frame rate is increased by a factor of 10 when employing the smashed filter rather than the GAMP solver.

5.3 Conclusion

I have described an algorithm for visual tracking in the compressive domain which does not require a computationally costly image reconstruction step. The algorithm is

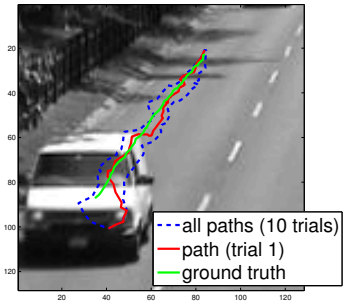


Figure 5.11: Tracker Paths and Ground Truth for CDNET Data, $r = 0.1$.

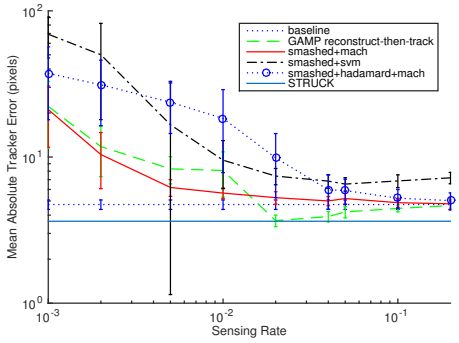


Figure 5.12: Mean Absolute Tracker Error for Highway Dataset, Frames 80-135.

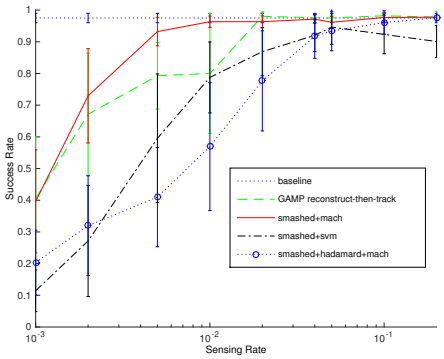


Figure 5.13: Tracker Success Rate for Highway Dataset, Frames 80-135.

Table 5.2: Comparison of Algorithm Framerates.

Algorithm	Framerate (frames/s)
Baseline	1.18
GAMP	0.112
Smashed	1.12

able to successfully track moving vehicles in grayscale videos from stationary cameras. I believe this algorithm represents a promising approach to the problem of detection and tracking from compressed sensing cameras in a resource-constrained setting.

Beyond the clear computational advantages, the fast smashed filter + MACH approach described here shows graceful degradation. When the sensing rate crosses the phase transition point for the GAMP algorithm, the reconstruct-then-track approach abruptly fails along with the reconstruction algorithm - between a 0.01 and 0.02 sensing rate, in this case. The smashed filter tracker continues to function at lower sensing rates, albeit with increased tracker error.

Chapter 6

RECONSTRUCTION-FREE CS IMAGE CLASSIFICATION

In this section I present an alternate approach to direct inference, adapting the existing Deep Boltzmann Machine (DBM) method to compressive data. A DBM is first pre-trained on non-compressive data and then adapted to CS data. Section 6.1 details modifications to the DBM training procedure to adapt it for use on CS data. Results for the MNIST handwritten digit classification task are given in Section 6.2 and the significance of the results is discussed in Section 6.3.

6.1 Training Approach

A simple two-stage approach is used to adapt the DBM for use on CS data. The DBM is first trained on non-compressive data $\mathbf{x} \in \mathbb{R}^{N\times 1}$ to generate initial network weights W_0. This is done in order to most effectively learn the structure of the training data, before it has been obfuscated by the compressive sensing operation. Next, a sensing matrix Φ with orthonormal rows is chosen. An initial guess for layer-1 weights $\mathrm{W}_{CS}^{(1)}$ is chosen according to the expression

$$\mathrm{W}_{CS}^{(1)} = \Phi \mathrm{W}^{(1)} \tag{6.1}$$

where $\mathrm{W}^{(1)} \in \mathbb{R}^{N\times P_1}$ is the weight matrix containing the first-layer weights $w_{ij}^{(1)}$. This provides an initial guess for a network which performs the same classification task given the $M \times 1$ input vector. The RIP implies preservation of inner products [10], making this a plausible approach. However, this guess does not perform significantly better than chance, and further training is needed. This stage 2 training is accomplished by running a backpropagation algorithm on a compressively sensed version of

the same training dataset. Backpropagation has little effect on the weights of inner layers but these weights have already been optimized by the stage 1 training.

6.2 Simulations

The training and classification approach described above was tested on the MNIST handwritten digit dataset. The dataset consists of 28x28 grayscale images of the digits 0-9. These digits are divided into a training set of 50,000 images and a testing set of 10,000 images. MNIST was chosen as an initial dataset for testing because it presents a well-defined problem with adequate training data, and a DBM-based system has already shown high performance on this classification task [87]. This allows the effects of adding compressive sensing to the system to be isolated.

The initial training used a DBM with 2 hidden layers (4 layers total). The number of nodes was chosen as in [87] (Figure 3.2) with $D = 784$, $P_1 = 500$, $P_2 = 1000$, and $P_l = 10$. A sensing matrix Φ was generated as a random orthoprojector and a DBM was trained as described in 6.1. This was repeated with r swept from 0.01 to 0.4. Figure 6.1 shows the progress of stage 2 training, in which backpropagation is used to learn a set of weights specific to the chosen sensing matrix. The confusion matrix for $r = 0.4$ is given in Table 6.1.

Figure 6.2 shows the results of this simulation. The network achieves a 1.21% error rate at a sensing rate $r = 0.4$. As expected, reduced sensing rates lead to higher error rates. In addition, the error appears to vary linearly with the compression ratio N/M.

The algorithm can be compared against a naive reconstruct-first approach, again shown in Figure 6.2. In this approach, the EMBGAMP algorithm [55] with a 2-level Daubechies-4 wavelet basis was used to reconstruct the characters. EMBGAMP was chosen because it achieves good reconstruction performance and does not require the

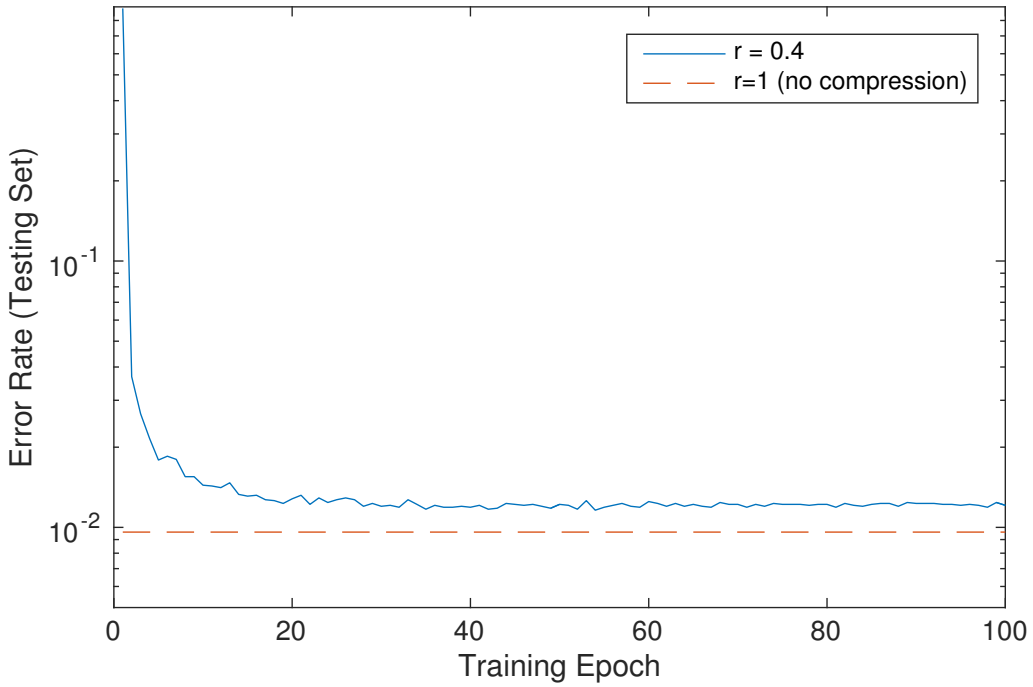


Figure 6.1: Compressive DBM Error During Training, for Sensing Rate $r = 0.4$. The CS-DBM achieves a 1.21% error rate. This is in contrast to the 0.99% error rate of the standard non-compressive DBM, shown for comparison.

user to set any parameters before use. The reconstructions were then classified by the standard (uncompressed) DBM. The algorithm performed poorly; I believe this is because the DBM did not encounter any CS reconstruction errors during training and therefore did not learn to account for them.

6.3 Discussion of Results

The training method and simulations described above show that the DBM approach can be used to perform classification directly on CS data. The MNIST handwritten digit dataset was used as a test case, and showed steadily increasing error as measurements were further compressed. This is in contrast to reconstruction algo-

Table 6.1: Compressive DBM Confusion Matrix, $r = 0.4$

		Predicted Digit (%)									
True Digit		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
	0	99.6	0	0.1	0	0	0	0.2	0.1	0	0
	1	0.1	99.7	0.1	0.1	0	0	0	0	0.1	0
	2	0	0.4	98.9	0	0	0	0.1	0.5	0.1	0
	3	0	0	0.1	99.1	0	0.3	0	0.1	0.3	0.1
	4	0	0	0.1	0	98.8	0	0.4	0	0.1	0.6
	5	0	0	0	0.6	0	99.1	0.3	0	0	0
	6	0.3	0.2	0	0	0.1	0.4	98.6	0	0.3	0
	7	0	0.2	0.6	0	0	0	0	98.9	0	0.3
	8	0.3	0	0.2	0.2	0.1	0.2	0.1	0.2	98.4	0.3
	9	0.2	0	0	0	0.6	0.3	0	0.5	0.4	98.0

rithms, which perform well up to a certain "phase transition" point and then break down. The behavior also differs from that observed in [95], where a minimum in neural network error rate was observed, giving an optimal level of dimensionality reduction. This is not surprising. The dataset (MNIST) and classification method (DBM) were well suited to each other and showed good performance. With no need to mitigate over-fitting, the dimensionality reduction process could only hurt. Despite this, the reconstruction-free algorithm outperformed the naive reconstruct-first approach at all sensing rates.

The approach described in this section kept the number of hidden-layer nodes constant as the sensing rate was reduced. I conjecture that an approach which instead maintains a constant number of weights may show improved performance. If correct,

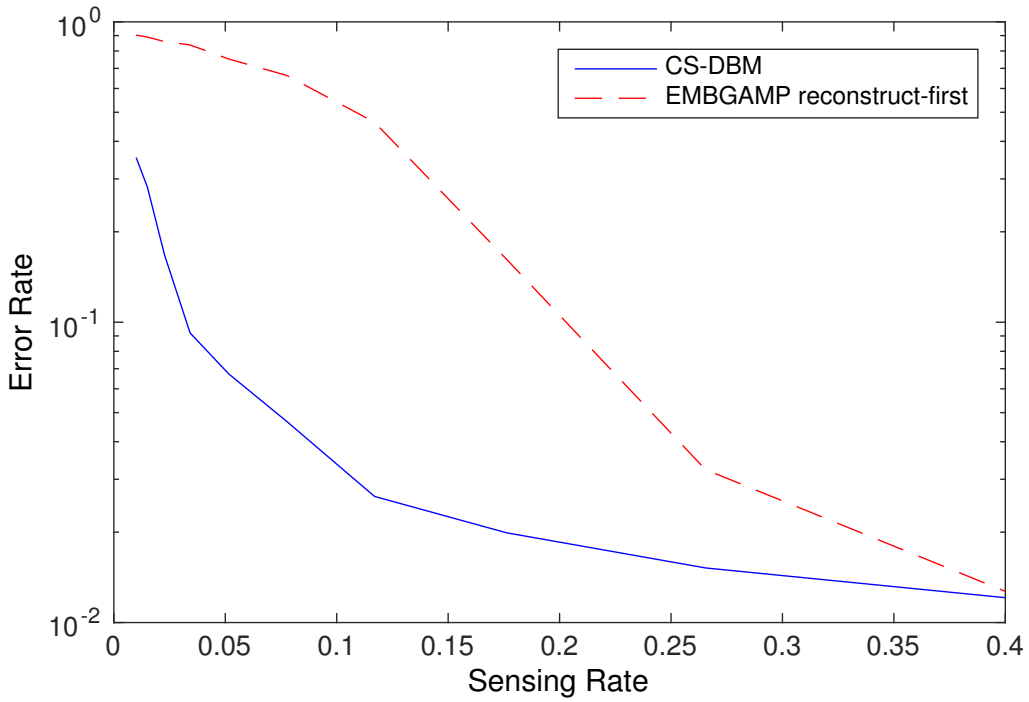


Figure 6.2: CS-DBM Performance Versus Sensing Rate.

Error rate steadily increases as the number of measurements is reduced. However, there is no "breakdown" point at which detection abruptly fails. The EMBGAMP reconstruct-first approach is shown for comparison. It is not robust to CS reconstruction errors and shows poor performance, eventually reaching an error rate comparable to random chance at low sensing rates.

there is an opportunity to improve error rate in the above-breakdown region of the phase transition map. In the region where reconstruction fails, I believe misclassification to be related to more fundamental limits on data compression and do not expect large improvements from optimizing network parameters.

It was hoped that the initial weights given in (6.1) would show good performance without additional training. This proved not to be the case; results were approximately equal to chance. Since the RIP guarantees the preservation of inner products,

this failure must be explained. I believe the observed performance is caused by the influence of noise, since compressive sensing will reduce SNR of the inner products by a factor of $\sqrt{r}$ [10].

Work is currently underway to move toward a network which does not need to be trained for use with a specific Φ. I hope to accomplish this by training a larger network on many randomized sensing matrices. In this way, I hope that the network will learn to be robust to the increased noise of CS-based inner products.

Chapter 7

CONCLUSION AND FUTURE WORK

In this work I have explored several aspects of performing inference on compressively sensed data. My aim has been two-fold: first to improve the accuracy of these inferences, and second to reduce the computational burden of making them. I have focused on the problem of a sensor similar to the single-pixel camera [1], with the goal of targeting applications in short-wave infrared. Despite this focus, this work should apply to a variety of imaging modalities, including magnetic resonance and hyperspectral imaging.

In Chapter 4, I worked to improve reconstruction accuracy in CS videos without regard for computational cost. My approach was to exploit statistical dependence between adjacent video frames, in a paradigm that mirrors the block matching algorithms employed by video codecs. I formulated the problem of reconstructing two frames of CS video as an ℓ_1 basis pursuit problem. A sequence of two frames was modeled as a single frame (the *I frame*), a displacement field predicting the second frame (the *P frame*), and a residual encoding changes which cannot be modeled by a translation-based prediction algorithm. Wavelet sparsity in both the *I frame* and spatial sparsity in the prediction residual was promoted through the convex optimization's ℓ_1 term.

In simulations of multi-frame reconstruction, two cases were considered. In the first case, optical flow was assumed already known; the Middlebury dataset includes ground truth optical flow and was used for these simulations. As expected, the two-frame optimization outperformed its single-frame equivalent, as measured by PSNR. In the second case, a constant, but unknown, translation was assumed over the entire

image. To generate two frames meeting this assumption, the PETS2000 dataset was used. An arbitrary frame was chosen as the *I frame*, and the *P frame* was synthesized as a function of the *I frame* and a randomly generated displacement vector. The reconstruction algorithm estimated the motion vector using a diamond search and performed reconstruction as described above. Again, PSNR of the reconstructed frames was improved.

This algorithm and the associated simulations show the importance of exploiting temporal dependence in CS video reconstruction algorithms. The importance of explicitly modeling the distribution of prediction residuals is also shown. In some simulations, the prediction residual was constrained to 0. Without sparsity in the prediction residual, the reconstruction algorithm showed reduced performance, and even showed decreasing PSNR as sensing rates increased. As with conventional video coding, motion estimation alone is not enough.

Although Chapter 4 targeted CS video applications, it is relevant to still imaging as well. The single-pixel camera architecture is not compressive in time and suffers from effects analogous to motion blur in a conventional camera. At a measurement rate of 32 kHz, a single-pixel camera still requires 7.4 seconds to acquire a 1024x768 pixel XGA image at a sensing rate of 0.3. InView Corporation mitigates this problem by collecting multiple measurements in parallel–a low-pixel rather than single-pixel approach. By applying motion estimation algorithms, the effective shutter speed of CS cameras could be dramatically increased. Magnetic resonance imaging also suffers from motion artifacts, with patients required to stay still over long scan times. Motion estimation and correction in MRI is an active area of research and the nature of MRI sensing presents the same chicken-and-egg problem as the single-pixel camera. A solution to one problem is likely to be useful in the other. Another potential application lies in "rolling shutter" correction for conventional CMOS cameras.

In Chapter 5, I describe an automated vehicle tracking algorithm which follows targets in CS video at computational speeds comparable to an equivalent non-CS algorithm. This algorithm successfully holds targets at sensing rates below that required for successful single-frame image reconstruction. The tracker operates in the tracking by detection paradigm and employs a particle filter to maintain and update estimates of target state. In each frame, detection is performed with the fast smashed filter, which generates noisy estimates of image correlation with a target template. By performing background subtraction and training templates based on difference frames, clutter was dramatically reduced, at the cost of limiting the algorithm's application to moving targets with stationary backgrounds.

The tracking algorithm was tested on the CDNET2012 "highway" video sequence. Two-dimensional target position and velocity were tracked, as well as target scale. In tests, the tracker successfully held a target vehicle with 93% success rate at a sensing rate of $r = 0.005$. In contrast, an equivalent reconstruct-first tracker achieved 79% success at $r = 0.005$ and required a rate of $r \geq 0.01$ to achieve success rates over 90%.

The tracker was also used as a detector in the PETS200 dataset. By introducing a binary alive/dead variable to the particle filter's state model, the algorithm concurrently estimated target location and presence/absence. Starting from a uniform prior distribution on target location, the algorithm was able to correctly identify and track the target at a sensing rate of 0.3. However, a single target template was not able to accurately track all target poses seen in the video well enough to outperform a reconstruct-first tracker. Achieving this goal would require either a prohibitively large number of filters, or a more sophisticated detection approach than the MACH filter.

Chapter 6 considers deep Boltzmann machines as one possible way to translate sophisticated machine learning algorithms to the reconstruction-free CS classification

and detection problem. A DBM model was trained on non-compressive data, and its network weights were projected to an equivalent CS model. This equivalent model was then fine-tuned by further training on compressively sensed versions of the same training data, generating a CS-DBM network trained specifically for a single sensing matrix. This fine-tuning step was found to greatly reduce classification error.

The CS-DBM network was tested on the MNIST handwritten digit dataset. As expected, error rate increased with decreasing sensing rate. In this case, error rate was approximately proportional to $1/r$. At a sensing rate of $r = 0.4$, a 1.21% error rate was observed, compared with 0.99% for a non-compressive DBM. A conventional DBM operating on CS data reconstructed using EMBGAMP, however, only achieved a 1.27% error rate at $r = 0.4$, with much worse performance as sensing rate increased. These results clearly demonstrate the inadequacy of the naive reconstruct-first approach to classification and the necessity of training networks specifically for use with CS data.

7.1 Future work

Several open-ended questions remain in CS video reconstruction. First, there is the relatively simple and mechanical work of extending the algorithm to more than two video frames and quantifying the trade-offs involved. There is also the possibility of placing more sophisticated priors on the prediction residual than simple spatial sparsity. Beyond this, the reconstruction algorithm of Chapter 4 does not completely solve the CS reconstruction problem for dense optical flow - the "chicken and egg" problem of estimating movement without a fully reconstructed image remains. A natural first step is to alternate stages of motion estimation with image reconstruction. However, the ideal system would maintain and concurrently update estimates of both the reconstructed images and the corresponding optical flow fields. It would optimize

an objective function that simultaneously sought compressibility of all video frames, the optical flow fields between them, and the residual from the optical flow prediction. The highly general graphical models of the message passing algorithms provide a potential solution. In particular, both optical flow estimation [62] and CS reconstruction have been formulated as ℓ_1 problems. The BiGAMP algorithm provides a potential method for combining the two estimation problems in a single bilinear model.

The tracking algorithm of Chapter 5 also shows clear opportunities for improvement. The particle filtering paradigm provides a powerful and modular framework for continued improvement and optimization. Many improvements to the state evolution model are possible; the track-before-detect particle filter used on the PETS2000 data is one example. The addition of target scaling required for reliable tracking of CDNET2012 data is another. I believe, however, that the most important improvements will come through new methods of estimating target likelihood. The current likelihood model relies on estimates of correlation with a template image (the smashed filter approach). It further improves performance by performing background subtraction and using a filter trained on difference images of the target. This approach is powerful and allows tracking at very low sensing rates, but is only capable of identifying moving targets in video sequences with stationary backgrounds. In the "Tracking by detection" paradigm implied by the use of the particle filter, more sophisticated classifiers and detectors are the drivers of tracking performance.

The DBM-based CS classifier of Chapter 6 represents an attempt to move beyond correlation with a target template and apply recent developments in machine learning to CS problems. While we consider a standard classification problem, the goal of this research is to develop a fast and highly accurate likelihood estimation method for use in tracking algorithms. We demonstrate the inadequacy of the naive reconstruct-first

approach to classification, but the networks trained in Chapter 6 are only one of many possible approaches to the problem.

A best-case reconstruct-first network must be developed for comparison with reconstruction-free methods. This tracker must be as robust as possible to reconstruction error. An obvious first step is to train the network on CS reconstructions of training data, but there are unanswered questions. We can generate an arbitrarily large volume of training data by sensing and then reconstructing the same sample with different sensing matrices. How much must the training data be expanded in this way for the network to learn invariance to noise? Does the number of nodes need to be increased, and if so by how much?

Work must also be done to optimize the direct classification network's size. The CS-DBM must, in a sense, perform both reconstruction and tracking, a fact that would suggest a larger network is necessary. In simulations, however, larger networks suffered from overfitting and showed reduced performance, even when training data were expanded by using multiple sensing matrices.

The CS-DBM was evaluated using Monte Carlo methods, specifically a variation on Gibbs sampling. Applying belief propagation to the evaluation of DBMs is a natural fit, and was partially attempted in [89]. Stability and convergence guarantees for such a method could also be developed.

Beyond the DBM, other neural network architectures should be examined, most notably convolutional neural networks. The DBM was selected because it is a generative model and it was hoped that it would learn a strong prior distribution for the training dataset, but it is not the only possible choice of generative model. For instance, the deep dream network achieved popular notoriety by employing a CNN in a generative way.

7.2 Final Remarks

This work is an attempt to broaden the applications of compressed sensing, particularly in areas where computational power is limited. Applications in video-based surveillance and autonomous navigation were targeted. A form of model-based CS is also discussed: videos have a compressible structure which does not translate into sparsity in a straightforward way. By formulating a reconstruction problem based on optical flow, CS sensing performance is improved. By developing algorithms that perform inference on CS data at computational costs comparable to computer vision algorithms operating on non-CS, new applications of CS sensors become feasible. My hope is that this work leads to the widespread adoption of cost-effective high-performance compressed sensing systems in a diverse range of imaging applications.

REFERENCES

[1] Dharmpal Takhar, Jason N. Laska, Michael B. Wakin, Marco F. Duarte, Dror Baron, Shriram Sarvotham, Kevin F. Kelly, and Richard G. Baraniuk, "A new compressive imaging camera architecture using optical-domain compression," in *Proc. SPIE*, 2006, vol. 6065, pp. 606509–606509–10.

[2] D.L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[3] B.D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th international joint conference on Artificial intelligence*, 1981.

[4] Simon Baker, Daniel Scharstein, J.P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski, "A database and evaluation methodology for optical flow," *International Journal of Computer Vision*, vol. 92, no. 1, pp. 1–31, 2011.

[5] Vision@Reading, "PETS: Performance evaluation of tracking and surveillance," 2007.

[6] H. Braun, P. Turaga, and A. Spanias, "Direct tracking from compressive imagers: A proof of concept," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, May 2014, pp. 8139–8142.

[7] D. Salmond and H. Birch, "A particle filter for track-before-detect," in *Proceedings of the American Control Conference*, Jun 2001, pp. 3755–3760.

[8] Jun S Liu and Rong Chen, "Sequential monte carlo methods for dynamic systems," *Journal of the American statistical association*, vol. 93, no. 443, pp. 1032–1044, 1998.

[9] A. Mahalanobis, BVK Vijaya Kumar, S. Song, SRF Sims, and JF Epperson, "Unconstrained correlation filters," *Applied Optics*, vol. 33, no. 17, pp. 3751–3759, 1994.

[10] Mark A. Davenport, Marco F. Duarte, Michael B. Wakin, Jason N. Laska, Dharmpal Takhar, Kevin F. Kelly, and Richard G. Baraniuk, "The smashed filter for compressive classification and target recognition," *Proc. SPIE*, vol. 6498, pp. 64980H–64980H–12, 2007.

[11] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "Changedetection.net: A new change detection benchmark dataset," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, June 2012, pp. 1–8.

[12] Bruno A Olshausen and David J Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?," *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[13] Enrique G Ortiz and Brian C Becker, "Face recognition for web-scale datasets," *Computer Vision and Image Understanding*, vol. 118, pp. 153–170, 2014.

[14] Emmanuel J. Candes, "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique*, vol. 346, no. 910, pp. 589 – 592, Apr. 2008.

[15] E.J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?," *Information Theory, IEEE Transactions on*, vol. 52, no. 12, pp. 5406 –5425, Dec. 2006.

[16] Lu Gan, Cong Ling, Thong T Do, and Trac D Tran, "Analysis of the statistical restricted isometry property for deterministic sensing matrices using Stein's method," *Preprint*, vol. 190, 2009.

[17] R. Calderbank, S. Howard, and S. Jafarpour, "Construction of a large class of deterministic sensing matrices that satisfy a statistical isometry property," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 358–374, April 2010.

[18] R.G. Baraniuk, V. Cevher, M.F. Duarte, and C. Hegde, "Model-based compressive sensing," *Information Theory, IEEE Transactions on*, vol. 56, no. 4, pp. 1982–2001, 2010.

[19] Shihao Ji, Ya Xue, and Lawrence Carin, "Bayesian compressive sensing," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2346–2356, Jun. 2008.

[20] Mário Figueiredo, "Adaptive sparseness using jeffreys prior," in *Advances in neural information processing systems*, 2001, pp. 697–704.

[21] S. Gleichman and Y. C. Eldar, "Blind compressed sensing," *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6958–6975, Oct 2011.

[22] M.B. Wakin, J.N. Laska, M.F. Duarte, D. Baron, S. Sarvotham, D. Takhar, K.F. Kelly, and R.G. Baraniuk, "An architecture for compressive imaging," in *Image Processing, 2006 IEEE International Conference on*, Atlanta, GA, 2006, pp. 1273–1276.

[23] M.A. Herman, D.E. Hewitt, T.H. Weston, and L. McMackin, "Overlap patterns and image stitching for multiple-detector compressive-sensing camera," Mar. 3 2015, US Patent 8,970,740.

[24] Ge Wang, Yoram Bresler, and Vasilis Ntziachristos, "Guest editorial: compressive sensing for biomedical imaging," *IEEE transactions on medical imaging*, vol. 5, no. 30, pp. 1013–1016, 2011.

[25] Gary H. Glover, "Simple analytic spiral k-space algorithm," *Magn. Reson. Med*, pp. 412–415, 1999.

[26] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, "Compressed sensing mri," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 72–82, March 2008.

[27] Michael Lustig, David Donoho, and John M Pauly, "Sparse mri: The application of compressed sensing for rapid mr imaging," *Magnetic resonance in medicine*, vol. 58, no. 6, pp. 1182–1195, 2007.

[28] Ti-Chiun Chang, Lin He, and Tong Fang, "Mr image reconstruction from sparse radial samples using bregman iteration," in *Proceedings of the 13th Annual Meeting of ISMRM, Seattle*, 2006, vol. 696.

[29] Jong Chul Ye, Sungho Tak, Yeji Han, and Hyun Wook Park, "Projection reconstruction mr imaging using focuss," *Magnetic Resonance in Medicine*, vol. 57, no. 4, pp. 764–775, 2007.

[30] Kai Tobias Block, Martin Uecker, and Jens Frahm, "Undersampled radial mri with multiple coils. iterative image reconstruction using a total variation constraint," *Magnetic resonance in medicine*, vol. 57, no. 6, pp. 1086–1098, 2007.

[31] Sean B Fain, Walter Block, Andrew Barger, and Charles A Mistretta, "Correction for artifacts in 3d angularly undersampled mr projection reconstruction," in *9th Annual Meeting of ISMRM, Glasgow*. Citeseer, 2001, p. 759.

[32] Juan M Santos, Charles H Cunningham, Michael Lustig, Brian A Hargreaves, Bob S Hu, Dwight G Nishimura, and John M Pauly, "Single breath-hold whole-heart mra using variable-density spirals at 3t," *Magnetic resonance in medicine*, vol. 55, no. 2, pp. 371–379, 2006.

[33] Michael Lustig, Jin Hyung Lee, David L Donoho, and John M Pauly, "Faster imaging with randomly perturbed, under-sampled spirals and l1 reconstruction," in *Proceedings of the 13th annual meeting of ISMRM, Miami Beach*, 2005, p. 685.

[34] S.S. Vasanawala, M.J. Murphy, M.T. Alley, P. Lai, K. Keutzer, J.M. Pauly, and M. Lustig, "Practical parallel imaging compressed sensing mri: Summary of two years of experience in accelerating body mri of pediatric patients," in *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on*, March 2011, pp. 1039–1043.

[35] Benjamin P Berman, Abhishek Pandey, Zhitao Li, Lindsie Jeffries, Theodore P Trouard, Isabel Oliva, Felipe Cortopassi, Diego R Martin, Maria I Altbach, and Ali Bilgin, "Volumetric mri of the lungs during forced expiration," *Magnetic resonance in medicine*, 2015.

[36] M.A.T. Figueiredo, R.D. Nowak, and S.J. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 1, no. 4, pp. 586–597, 2007.

[37] E. van den Berg and M. P. Friedlander, "SPGL1: A solver for large-scale sparse reconstruction," June 2007, http://www.cs.ubc.ca/labs/scl/spgl1.

[38] Y.C. Pati, R. Rezaiifar, and PS Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, Pacific Grove, CA, 1993, pp. 40–44.

[39] Joel Tropp and Anna Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, pp. 4655–4666, Dec. 2007.

[40] Deanna Needell and Roman Vershynin, "Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit," *Foundations of computational mathematics*, vol. 9, no. 3, pp. 317–334, Jun. 2009.

[41] D. L. Donoho, Y. Tsaig, I. Drori, and J. L. Starck, "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 58, no. 2, pp. 1094–1121, Feb 2012.

[42] D. Needell and J.A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 301–321, May 2009.

[43] K. Lee and Y. Bresler, "ADMiRA: Atomic decomposition for minimum rank approximation," *Information Theory, IEEE Transactions on*, vol. 56, no. 9, pp. 4402–4416, Sep. 2010.

[44] Sundeep Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, St. Petersburg, Russia, 2011, IEEE, pp. 2168–2172.

[45] Gregory F Cooper, "The computational complexity of probabilistic inference using bayesian belief networks," *Artificial intelligence*, vol. 42, no. 2-3, pp. 393–405, 1990.

[46] Judea Pearl, "Reverend bayes on inference engines: A distributed hierarchical approach," in *AAAI*, 1982, pp. 133–136.

[47] Y. Weiss, "Correctness of local probability propagation in graphical models with loops," *Neural Computation*, vol. 12, no. 1, pp. 1–41, Jan 2000.

[48] R. J. McEliece, D. J. C. MacKay, and Jung-Fu Cheng, "Turbo decoding as an instance of pearl's ldquo;belief propagation rdquo; algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 140–152, Feb 1998.

[49] Kevin P Murphy, Yair Weiss, and Michael I Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1999, pp. 467–475.

[50] Andrea Montanari, Balaji Prabhakar, and David Tse, "Belief propagation based multi–user detection," in *Proceedings of the 43rd Allerton Conference on Communication, Control and Computing*, Sep. 2005.

[51] S. Rangan, "Estimation with random linear mixing, belief propagation and compressed sensing," in *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*, Princeton, NJ, March 2010, pp. 1–6.

[52] Dongning Guo and Chih-Chun Wang, "Asymptotic mean-square optimality of belief propagation for sparse linear systems," in *2006 IEEE Information Theory Workshop-ITW'06 Chengdu*. IEEE, 2006, pp. 194–198.

[53] David L Donoho, Arian Maleki, and Andrea Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 18914–18919, 2009.

[54] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing: I. motivation and construction," in *Information Theory (ITW 2010, Cairo), 2010 IEEE Information Theory Workshop on*, Jan 2010, pp. 1–5.

[55] J.P. Vila and P. Schniter, "Expectation-maximization Gaussian-mixture approximate message passing," *Signal Processing, IEEE Transactions on*, vol. 61, no. 19, pp. 4658–4672, Oct 2013.

[56] Jason T. Parker, Philip Schniter, and Volkan Cevher, "Bilinear generalized approximate message passing," *CoRR*, vol. abs/1310.2632, 2013.

[57] A.E. Waters, A.C. Sankaranarayanan, and R.G. Baraniuk, "SpaRCS: Recovering low-rank and sparse matrices from compressive measurements," *Proc. Adv. in Neural Processing Systems (NIPS), Granada, Spain*, 2011.

[58] A.C. Sankaranarayanan, C. Studer, and R.G. Baraniuk, "CS-MUVI: Video compressive sensing for spatial-multiplexing cameras," in *Computational Photography (ICCP), 2012 IEEE International Conference on*, Apr. 2012, pp. 1 –10.

[59] V. Cevher, A. Sankaranarayanan, M. Duarte, D. Reddy, R. Baraniuk, and R. Chellappa, "Compressive sensing for background subtraction," in *European Conference on Computer Vision*, Marseille, France, 2008, pp. 155–168, Springer.

[60] N. Jacobs, S. Schuh, and R. Pless, "Compressive sensing and differential image-motion estimation," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 718–721.

[61] Vijayaraghavan Thirumalai and Pascal Frossard, "Correlation estimation from compressed images," *ArXiv*, 2011.

[62] Xiaohui Shen and Ying Wu, "Sparsity model for robust optical flow estimation at motion discontinuities," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, june 2010, pp. 2456 –2463.

[63] Petr Tichavsky, Carlos H Muravchik, and Arye Nehorai, "Posterior cramér-rao bounds for discrete-time nonlinear filtering," *IEEE Transactions on signal processing*, vol. 46, no. 5, pp. 1386–1396, 1998.

[64] Hadi Zayyani, Massoud Babaie-Zadeh, and Christian Jutten, "Bayesian cramer-rao bound for noisy non-blind and blind compressed sensing," *arXiv preprint arXiv:1005.4316*, 2010.

[65] David Donoho and Jared Tanner, "Counting faces of randomly projected polytopes when the projection radically lowers dimension," *Journal of the American Mathematical Society*, vol. 22, no. 1, pp. 1–53, 2009.

[66] David Donoho and Jared Tanner, "Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing," *Philosophical Transactions A: Mathematical, Physical and Engineering Sciences*, vol. 367, no. 1906, pp. 4273–4293, Nov. 2009.

[67] Khanh Do Ba, Piotr Indyk, Eric Price, and David P Woodruff, "Lower bounds for sparse recovery.," in *SODA*. SIAM, 2010, vol. 10, pp. 1190–1197.

[68] Emmanuel Candes and Justin Romberg, "l1-magic: Recovery of sparse signals via convex programming," 2005.

[69] D. L. Donoho, A. Maleki, and A. Montanari, "The noise-sensitivity phase transition in compressed sensing," *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6920–6941, Oct. 2011.

[70] A. S. Lewis and G. Knowles, "Image compression using the 2-d wavelet transform," *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 244–250, Apr 1992.

[71] Ingrid Daubechies, "The wavelet transform, time-frequency localization and signal analysis," *IEEE Transactions on Information Theory*, vol. 36, no. 5, pp. 961–1005, 1990.

[72] Ingrid Daubechies, *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics, 1992.

[73] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[74] Jaswant Jain and Anil Jain, "Displacement measurement and its application in interframe image coding," *IEEE Transactions on communications*, vol. 29, no. 12, pp. 1799–1808, 1981.

[75] Ram Srinivasan and K Rao, "Predictive coding based on efficient motion estimation," *IEEE Transactions on Communications*, vol. 33, no. 8, pp. 888–896, 1985.

[76] M Ghanbari, "The cross-search algorithm for motion estimation [image coding]," *IEEE Transactions on Communications*, vol. 38, no. 7, pp. 950–953, 1990.

[77] Liang-Wei Lee, Jing-Fa Wang, Jau-Yien Lee, and J-D Shie, "Dynamic search-window adjustment and interlaced search for block-matching algorithm," *IEEE Transactions on circuits and systems for video technology*, vol. 3, no. 1, pp. 85–87, 1993.

[78] Reoxiang Li, Bing Zeng, and Ming L Liou, "A new three-step search algorithm for block motion estimation," *IEEE transactions on circuits and systems for video technology*, vol. 4, no. 4, pp. 438–442, 1994.

[79] Shan Zhu and Kai-Kuang Ma, "A new diamond search algorithm for fast block matching motion estimation," in *Information, Communications and Signal Processing, 1997. ICICS., Proceedings of 1997 International Conference on*, sep 1997, vol. 1, pp. 292 –296 vol.1.

[80] Junyu Han, Fei Qi, and Guangming Shi, "Gradient sparsity for piecewise continuous optical flow estimation," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, sept. 2011, pp. 2341 –2344.

[81] Martin A Fischler and Robert C Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[82] Hanying Zhou, Casey Hughlett, Jay C. Hanan, Thomas Lu, and Tien-Hsin Chao, "Development of streamlined OT-MACH-based ATR algorithm for grayscale optical correlator," in *Proc. SPIE*, 2005, pp. 78–83.

[83] Mikel D. Rodriguez, Javed Ahmed, and Mubarak Shah, "Action MACH: a spatio-temporal maximum average correlation height filter for action recognition," in *In Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[84] Abhijit Mahalanobis and B. V. K. Vijaya Kumar, "Optimality of the maximum average correlation height filter for detection of targets in noise," *Optical Engineering*, vol. 36, no. 10, pp. 2642–2648, 1997.

[85] Oliver C Johnson, Weston Edens, Thomas T Lu, and Tien-Hsin Chao, "Optimization of OT-MACH filter generation for target recognition," in *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics, 2009, pp. 734008–734008.

[86] Abhijit Mahalanobis, B. V. K. Vijaya Kumar, Sewoong Song, S. R. F. Sims, and J. F. Epperson, "Unconstrained correlation filters," *Appl. Opt.*, vol. 33, no. 17, pp. 3751–3759, Jun 1994.

[87] Ruslan Salakhutdinov and Geoffrey Hinton, "Deep Boltzmann machines," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009, vol. 5, pp. 448–455.

[88] Geoffrey E Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[89] Eric W Tramel, Angélique Drémeau, and Florent Krzakala, "Approximate message passing with restricted boltzmann machine priors," *arXiv preprint arXiv:1502.06470*, 2015.

[90] Rick Chartrand and Wotao Yin, "Iteratively reweighted algorithms for compressive sensing," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2008, pp. 3869–3872.

[91] K. Kulkarni and P. Turaga, "Reconstruction-free action inference from compressive imagers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 4, pp. 772–784, Apr. 2016.

[92] Suhas Lohit, Kuldeep Kulkarni, Pavan Turaga, Jian Wang, and Aswin C Sankaranarayanan, "Reconstruction-free inference on compressive measurements," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Boston, MA, 2015, pp. 16–24.

[93] Yun Li, Chinmay Hegde, Aswin C Sankaranarayanan, Richard Baraniuk, and Kevin F Kelly, "Compressive image acquisition and classification via secant projections," *Journal of Optics*, vol. 17, no. 6, pp. 065701, May 2015.

[94] Ella Bingham and Heikki Mannila, "Random projection in dimensionality reduction: applications to image and text data," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 245–250.

[95] George E Dahl, Jack W Stokes, Li Deng, and Dong Yu, "Large-scale malware classification using random projections and neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3422–3426.

[96] Jarvis Haupt, Rui Castro, Robert Nowak, Gerald Fudge, and Alex Yeh, "Compressive sampling for signal classification," in *Signals, Systems and Computers, 2006. ACSSC'06. Fortieth Asilomar Conference on*. IEEE, 2006, pp. 1430–1434.

[97] Junbin Gao, Qinfeng Shi, and Tibério S Caetano, "Dimensionality reduction via compressive sensing," *Pattern Recognition Letters*, vol. 33, no. 9, pp. 1163–1170, 2012.

[98] John Wright, Allen Y Yang, Arvind Ganesh, Shankar S Sastry, and Yi Ma, "Robust face recognition via sparse representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 2, pp. 210–227, 2009.

[99] Hanxi Li, Chunhua Shen, and Qinfeng Shi, "Real-time visual tracking using compressive sensing," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, Colorado Springs, CO, Jun. 2011, pp. 1305–1312.

[100] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang, "Real-time compressive tracking," in *European Conference on Computer Vision*, Florence, Italy, 2012, pp. 864–877.

[101] Paul Viola and Michael Jones, "Rapid object detection using a boosted cascade of simple features," in *Conference on Computer Vision and Pattern Recognition*, Kauai, HI, 2001.

[102] Rudolph Emil Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[103] Mark A. Davenport, Michael B. Wakin, and Richard G. Baraniuk, "The compressive matched filter," Tech. Rep., Rice University, 2006.

[104] Sam Hare, Amir Saffari, and Philip HS Torr, "Struck: Structured output tracking with kernels," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 263–270.

APPENDIX A

NOTES ON NOMENCLATURE

Vectors, Scalars, and Matrices

Scalars and scalar-valued functions will be italicized (e.g. x, y, α). Vectors will be written in bold, (e.g. $\mathbf{x}$, $\mathbf{y}$, $\boldsymbol{\theta}$). Individual elements of a vector will be denoted with a subscript, as in the example $\mathbf{x} = [x_1\ x_2\ x_3\ ]^T$. Unless otherwise specified, column vectors are used. Matrices are neither bold or italic (e.g. A, Φ). Where possible, matrices are denoted by capital letters.

Random Variables

When possible, random variables are denoted by capital letters (e.g. $\mathbf{X}$) and specific realizations of that variable are lowercase (e.g. $\mathbf{x}$). Probability density functions (PDFs) are denoted by the letter p, with subscripts to distinguish the densities of different variables (e.g. $p_{\mathbf{X}}(\mathbf{x})$). The subscript is omitted when it is clear from context. Estimates of a random variable or unknown value are denoted with a circumflex, or "hat" (e.g. $\hat{\mathbf{x}}$).

Complex numbers

The symbol i is reserved for the imaginary unit. $\mathbb{R}$, $\mathbb{I}$, and $\mathbb{C}$ are the sets of real, imaginary, and complex numbers, respectively.

Iteration

When convenient, j, k, and l are used as indices for iterative operations. Other symbols may be used when they are more meaningful. For instance m will typically denote iteration over elements of the measurement vector $\mathbf{y} \in \mathbb{R}^{M\times 1}$. The letter t is used to refer to both continuous and discrete time.