Title
Fast and Robust Algorithms for Compressive Sensing and Other Applications

Author
Yang, Yi

Publication Date
2014

Peer reviewed|Thesis/dissertation

University of California

Los Angeles

Fast and Robust Algorithms for Compressive Sensing and Other Applications

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Mathematics

by

Yi Yang

2014

Abstract of the Dissertation

Fast and Robust Algorithms for Compressive Sensing and Other Applications

by

Yi Yang

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 2014

Professor Stanley Osher, Chair

Efficiency and robustness are often the main concerns in model design and algorithm development. Nowadays a lot of algorithms have been proposed with emphasis on one or the other. This thesis provides several algorithms together with their applications to address these two needs.

The first part of the thesis discusses the efficiency concern in video compression and reconstruction. With the increasing demand in real-time data transmission and storage, these two problems are attracting more and more attention. In terms of video compression, classic models often use a fixed temporal compression rate, while there are many potential gains in developing systems and procedures incorporating adaptive temporal compression rate. In Chapter 2, an algorithm based on local patches and polynomial fitting is proposed to adaptively predicts the temporal compression rate given the behavior of a few previous compressed frames. As for the inverse model, Chapter 3 presents a fast total variation based method for reconstructing video compressive sensing data. The regularization in the model is imposed on both the spatial and temporal components, which provides a more consistent approximation of the connection between neighboring frames with little to no increase in model complexity.

The second part of the thesis covers a new technique named adaptive outlier pursuit for handling sparsely corrupted data. In many real world applications, noise is often unavoidable during data acquisition and transmission. Some noise can damage part of the data seriously and make it contain no useful information at all. Algorithms robust to this type of noise are strongly needed. The technique adaptive outlier pursuit is introduced to deal with outliers in the acquired measurements. Instead of detecting and removing the outliers before applying classic algorithms, it alternates between the outlier detection and the signal reconstruction task, hence iteratively approaches the true signal in a more accurate way. It is applied to robust 1-bit compressive sensing and exact matrix completion in Chapter 5 and Chapter 6 respectively.

The dissertation of Yi Yang is approved.

Stefano Soatto

Joseph Teran

Lieven Vandenberghe

Wotao Yin

Stanley Osher, Committee Chair

University of California, Los Angeles

2014

To my friends and family

for their endless support and encouragement

throughout the entire doctorate program

Table of Contents

List of Figures

List of Tables

Acknowledgments

This dissertation would not have been possible without the constant help and support from so many people. I am extremely thankful to all of them, and here I would like to acknowledge a few of them who have strongly guided and enlightened me.

First and foremost, my deepest appreciation goes to my advisor, Professor Stanley Osher, for his professional advice, constant support and admirable wisdom. I would not have been able to enjoy the beauty of research and develop a disciplined mindset without his excellent guidance.

I would also like to thank all my doctoral committee members, Professors Stefano Soatto, Joseph Teran, Lieven Vandenberghe and Wotao Yin, for taking significant roles in my study and research during my years at UCLA. I would especially like to express my gratitude to Prof. Teran for serving as my temporal advisor when I first came to UCLA; to Prof. Vandenberghe for offering the fascinating optimization courses; and to Prof. Yin for the thought-provoking discussions as well as the tasteful coffee.

My thanks further goes to Professor Andrea Bertozzi and Luminita Vese, for offering wonderful applied math courses and seminars, as well as their kind advice in helping me decide research focuses. I would also like to thank all the people working in our department for their help and for making this department such a warm and fantastic place.

Besides, I am thankful to all my collaborators, Michael Moller, Ming Yan, Professor Jianwei Ma, Hayden Schaeffer and Jianing Shi. I would like to thank Michael Moller for teaching me how to write papers in a concise and rigorous way; thank Ming Yan for showing me how to become a real researcher; thank Prof. Ma for the valuable opportunity to visit Harbin Institute of Technology; and thank Hayden Schaeffer for all the help on math and English and for being

such a wonderful friend always. I would also like to show my appreciation to my friends Giang Tran, Ekaterina Merkurjev, Huiyi Hu and Hongjing Xia, for making my PhD life so colorful and joyful.

Finally, I would like to thank all my family members, especially my husband and my parents, for being so considerate and supportive during my entire graduate study.

The research presented in this dissertation was supported by NSF DMS 0835863, NSF DMS 0914561, ONR N00014-11-0749, ONR Grant N00014-11-1-0719, ONR Grant N00014-08-1-119, and an ARO MURI subcontract from Rice University.

Vita

2007	National Scholarship of China, University of Science and Technology of China (USTC).
2008	National Scholarship of China, USTC "Guo Moruo" Scholarship, USTC
2009	B.S. in Mathematics, USTC
2011	M.S. in Mathematics, University of California, Los Angeles (UCLA)
2009–present	Teaching Assistant, Department of Mathematics, UCLA Research Assistant, Department of Mathematics, UCLA
2013-2014	Dissertation Year Fellowship, UCLA

Publications

Yi Yang, Hayden Schaeffer, Wotao Yin and Stanley Osher, Mixing space-time derivatives for video compressive sensing, *Asilomar Conference on Signals, Systems, and Computers*, accepted, 2013.

Yi Yang, Jianwei Ma and Stanley Osher, Seismic data reconstruction via matrix completion, *Inverse Problems and Imaging*, 7(2013), 1379-1392.

Yi Yang, Michael Moller and Stanley Osher, A dual split Bregman method for

fast ℓ^1 minimization, *Mathematics of Computation*, 82(2013), 2061-2085.

Ming Yan, Yi Yang and Stanley Osher, Exact low-rank matrix completion from sparsely corrupted entries via adaptive outlier pursuit, *Journal of Scientific Computing*, 56(2013), 433-449.

Ming Yan, Yi Yang and Stanley Osher, Robust 1-bit compressive sensing using adaptive outlier pursuit, *IEEE Transactions on Signal Processing*, 60(2012), 3868-3875.

Part I

Real-Time Video Compression and Reconstruction

CHAPTER 1

Introduction

With the fast development in science and technology, nowadays people are looking for ways to analyze and process more data in shorter time. In terms of video processing, researchers are building optical devices that can handle data stream at 1 exapixel/second, and video processing is becoming one of the hottest "big data" problems. Although its size is often surprisingly large, the spatial and temporal redundancy of video data makes efficient storage and transmission applicable. How to efficiently compress and reconstruct the video data in real time is an important but challenging task. Here we propose two classes of models and algorithms for efficient video compression and reconstruction.

Chapter 2 discusses the forward model: adaptive temporal compression on video data. In video compressive sensing framework, the final stored data is defined as a linear combination of several consecutive spatially down-sampled frames. Here the spatial compression is typically accomplished with coded apertures. In terms of temporal compression, standard models often use a fixed compression rate. However, there are many potential gains in developing systems and procedures utilizing adaptive temporal compression rate. Our aim is to use a few previously compressed frames to predict a proper future temporal compression rate without any explicit knowledge of future frames. Through local patch-based study and adaptive polynomial fitting, we are able to achieve a restrictive estimate of the motion present in the most recent coded frame, hence a proper temporal compression rate can be obtained in an extremely fast way. Several numerical

tests are displayed to show the significance of our model.

Chapter 3 focuses on the inverse model: fast and robust video compressive sensing. According to the above description, the math model for the standard video compressive sensing can be formulated as $F = \sum_{i=1}^{n} A_i X_i$, where F is the compressed measurements, each A_i is a random binary matrix representing which pixels can store information, and X_i is the i^{th} original frame. Here our goal is to reconstruct X given A and F by leveraging certain desired sparsity properties on the original video. In order to achieve satisfying reconstruction results, proper regularization has to be imposed on both the spatial and temporal components. According to the connection between image and video data as well as the temporal consistency between adjacent frames, the total variation (TV) regularization is imposed on both the spatial and temporal terms. From numerical experiments, it is clear that our model outperforms other models greatly with several dB gain in PSNR values.

CHAPTER 2

Real-Time Adaptive Video Compression

2.1 Introduction

Adaptive temporal compression is at the frontier of application of compressive sensing (CS) [1], making it possible to acquire a large range of scenes using dynamic compression rates. Compressive systems focus on obtaining and storing the least amount of information while still maintaining a high level of recovery. For videos this means removing spatial and temporal redundancy, i.e. the high variation of physically observed motion, which appears over different time scales commonly found in video data. Simply stated, we wish to accelerate the acquisition process when the video is static and decelerate when the scene contains dynamic components – all in real-time.

In terms of hardware, current CS methods use physical techniques to code pixel data in order to compress spatial or spectral information. This is commonly done by coded apertures (typically consisting of mechanical gratings or variable materials) which block incoming light in a either patterned or random fashion, thereby subsampling the incoming signal. The idea of CS has had many applications to both hardware and data collection, which include but are not limited to the coded aperture snapshot spectral imaging (CASSI) [2, 3], single pixel camera [4, 5], cooperative analog and digital signal processing transform imager (CADSP) [6], random lens imaging [7], compressive structured light [8], compressive phase retrieval [9, 10], photodetector array camera and spectrometer [11], sparse mag-

netic resonance imaging (MRI) [12], and many more.

Mathematically speaking, the general forward model for CS can be formulated as follows: if the compression is encoded in a CS matrix A (normally non-invertible), then the relation between the encoded or compressed signal, vector X, and the "original" signal, vector F, is given by $X = AF$, where the assumption is that the data is obtained via linear measurements. For video compressive sensing (VCS) [13, 14, 15, 16, 17, 18], this F contains each frame of the true video, the X denotes the spatially and temporally compressed data, while the A contains the random frame by frame masks as well as the temporal compression via a linear combination of frames.

The idea of VCS is vastly different, both mathematically and philosophically, from the classical compression methods. In standard video compression algorithms, the incoming signal is sensed (acquired) in full. The acquired data is then processed, through various transformations and operations, until the data is represented in a sparse way (the compressed video). For example in MPEG-IV, the first frame is compressed in the wavelet basis and stored [13]. Then each incoming frame is stored by compressing the difference between the new frame and the first frame in the wavelet basis. Once the difference exceeds a specific tolerance, the process is reset. In some sense, this type of compression is *sensing then compressing*.

The problem we consider in this paper is that the incoming data exceeds the storage capacity, so the data must be compressed during the acquisition process. For this reason, we call this *video compressive sensing*.

Although there are many works in the literature focusing on the spatial compression of data, the field of variable temporal compression rates is fairly new. There are many potential gains in developing systems and procedures incorporating adaptive temporal compression rate. In terms of the memory, variable compression rates lead to optimized storage space without loss of quality as com-

pared to taking a moderate to high fixed rate. In terms of cost, the resource and energy savings outweigh the computational cost of predicting the frame rate, thereby increasing the efficiency of the system. VCS can be readily applied to many of the common big data sets, for example surveillance videos [19, 20] and traffic data.

In this work, we propose a simple and flexible real-time method to predict frame rates based on adaptive patchwise polynomial fitting. Our method is easy to implement and computationally inexpensive since it is based on temporal differences and polynomial fitting as well as robust to different applications and data conditions. The algorithm can be made parallel and can be extended to different imaging modalities. One current application of this method could be to coded aperture compressive temporal image (CACTI) systems [17].

This paper is organized as follows. Section 2.2 details the problem as well as our algorithm. That section also provides some connections between our model and the underlying physical behavior captured in the video. Some analytical remarks are provided in Section 2.3, with theoretical connections to classical methods. In Section 2.4, numerical simulations on real data are provided, which demonstrate the improvement in quality of the reconstructed video given our compression scheme. This section also discusses the robustness of our algorithm on the data acquisition process and the manner in which our algorithm adapts to the data. Lastly we conclude with some final remarks in Section 2.5.

2.2 Description of the Model

The main methodology of adaptive temporal compression is to give an estimate to the most restrictive motion present in the most recent coded frames, and to use this velocity to determine the potential frame rate. In an ideal case, the motion of objects in a video, i.e. the optical velocity V between frames, can be calculated

using the classical methods of optical flow [21, 22, 23]. From the optical velocity, it is clear that an optimal compression rate T can be determined by the relationship $T \sim \frac{1}{V}$. However, in VCS each compressed frame $X_j \in \mathbf{R}^{N\times M}$ is a coded linear combination of several true frames $F_{i|j} \in \mathbf{R}^{N\times M}$ with $0 < i \leq T_j$ (where $i|j$ is the i^{th} frame in the j^{th} sequence and where T_j is the frame rate for the j^{th} encoded sequence), i.e.

$$X_j := \sum_{i=1}^{T_j} A_{i|j} F_{i|j} \tag{2.1}$$

where $A_{i|j} \in \mathbf{R}^{N\times M}$ is a random binary spatial mask and we take the product above to be element-wise. In practice, either each mask $A_{i|j}$ is independently generated or only the first mask $A_{1|1}$ is randomly generated and each subsequential mask is a fixed translation of the previous one [17]. In this way, X_j has both missing data and motion blur. Due to the corruption, direct application of optical flow or block-matching techniques [16, 24, 25] is only possible after reconstructing each frame $F_{i|j}$. However, this drastically increases the computational cost, thus limiting its use in real-time video capturing. Parallel work [18] applies block matching directly on the raw data X_j to get a rough estimate of the fast moving blocks.

In this paper, we avoid the need to do local searches by analyzing the dynamics of each patch independently. First, each of the compressed frames X_j is processed by applying an averaging filter with small support (for simplicity we maintain the same notation for the smoothed frame). This removes the anomalies caused by missing data, while preserving the general structures. Then we divide each smoothed frame into non-overlapping patches of size $p_1 \times p_2$ in order to capture the local movements, where locality is related to the patch size. The non-overlapping nature breaks the computations down to a decoupled system of small subproblems of the patch sequences (in time), also making parallel computing possible. For

a given patch sequence, the mean of each element (denoted by $\mu^P(X)$ for each smoothed frame X and patch P) is calculated and the objective of our method is to detect the motion induced changes directly from the means. In essence, we use $\partial_t \mu^P(X(t))$ (we suppress the depends of X on the space variables for simplicity) as a motion estimator for the optical velocity V instead of directly obtaining V using classical methods. Since we are only able to view the running sum of compressed frames rather than each single frame in the original video, we can not derive V from the standard optical flow idea. However, each compressed frame comes from a short frame sequence, hence we can learn about the current motion changes through analyzing the difference between two adjacent compressed frames, which in some sense is closely related to the velocity at the current time. Note that the mean is used rather than other choices, for example median, since it is more sensitive to large changes while robust to Gaussian noise and small outliers.

As objects enter or leave a patch, the mean value of the patch changes by an amount related to their speed. In fact, we can show that the speed of the means of the patches is directly related to the optical velocity by the following relationship:

$$|\partial_t \mu^P(X(t))| \approx \frac{|V|}{|P|} \|X\|_{TV(P)} \tag{2.2}$$

where $|P|$ is the size of the patch and $||X||_{TV(P)} := \int_P |\nabla X|$ is the total variation (TV) semi-norm of the patch in space (note that this quantity is time dependent). The proof of this relationship is provided Section 2.3. Since many optical flow algorithms compare pixels or patches within a window, those algorithms can be thought of as a Lagrangian method, tracing out the motion path. Our model can be considered as an Eulerian based method, since the algorithm fixes the patch location and observes objects flowing through the patch. This distinction allows the model to have gains in speed and ease of implementation.

There are several important variables and functions, for a quick reference we

provide a list of them here:

- P is a rectangular patch of fixed size $p_1 \times p_2$.
- V is the velocity of the associated patch P.
- T is the temporal compression rate.
- $\mu^P(X(t))$ is the mean of a frame X in the patch P at time t, the spatial dependents of X is suppressed.
- $\mu_L^P(t)$ and $\mu_Q^P(t)$ are the linear and quadratic approximates (respectively) as a function of time associated with patch P.
- $\mu_{op}^P(t)$ is the optimal approximation of the mean in time for P.

2.2.1 Adaptive Polynomial Fitting

The main task of our algorithm is to calculate an estimation of the mean in the future frames and to use this information to determine the future compression rate. This is done in two stages. First, we use equation (2.2) to calculate the patch velocity and determine the extreme cases. If the approximated patch velocity V is very large (small), then the lowest (highest) compression rate is chosen. The approximation of V in equation (2.2) is a robust estimation of the large and small changes in the patch. Aside from theoretical reasons, we can also see from equation (2.2) that the TV term also helps to mitigate the influences of noisy patches. In the non-extremal cases, using equation (2.2) requires specific thresholds relating the optical velocity V to the compression rate T, which is usually much more difficult than deciding the extremal thresholds. In practice, relating V directly to a compression rate requires learning on training data [18]. Seeking a more self-contained estimator, we provide an adaptive approximation of $\mu^P(X(t))$ directly. Since a threshold on the maximum allowable tolerance of the changes in the mean

is related to the image intensity and the size of the patch, it provides a less sensitive measure than directly thresholding V. For example, while we can set the tolerance of the change to be a fraction of the maximum image intensity (known data), the tolerance on the velocity must be related to the range of object speeds present in the image (approximated or unknown data).

In the non-extremal cases, we use adaptive polynomial fitting to obtain a better prediction to the future patch mean. For the sake of simplicity, we will restrict the length of the patch sequence to be 4, although the following argument and methodology does not dependent on this value. The given data is now the patch means $\left\{\mu^P(X(t_{j-3})), ..., \mu^P(X(t_j))\right\}$ and their associated time points $\{t_{j-3}, ..., t_j\}$, which are the frame numbers in the true video data. The first three data points of the sequence act as the fitting data, where both a least squares linear fit $\mu_L^P(t)$ and quadratic interpolation $\mu_Q^P(t)$ are calculated. Then using the fourth data point the values $\mu_L^P(t_j)$ and $\mu_Q^P(t_j)$ are compared to the known value $\mu^P(X(t_j))$, providing us with an intrinsic way to learn which polynomial fit to use. Once a fit is chosen, that optimal polynomial $\mu_{op}^P(t)$ is used to estimate the maximum compression rate T such that $\left|\mu^P(t_j + T) - \mu_{op}^P(X_j)\right|$ is within our tolerance.

In application, the compression rate is usually restricted to a set of fixed values, for example, all the even numbers up to 16. Here we define T_{range} as an increasing sequence of length L storing all the compression rate candidates. We then arrive at the following algorithm for calculating the compression rate T at time point t_j.

Algorithm 1 Adaptive temporal compression

Input: $X(t_{j-3})$,...,$X(t_j)$, t_{j-3},...,t_j, p_1, p_2, T_{range}, $V_{\min}$, $V_{\max}$, threshold.
Initialization: (Optional:) Process each X with averaging filter.
Divide each frame into non-overlapping $p_1 \times p_2$ patches. $k = 1$.
while $k \leq \frac{MN}{p_1 p_2}$ **do**
Compute $\{\mu(X(t_{j-3})), ..., \mu(X(t_j))\}$ in the k^{th} patch sequence.
Determine the current V from (2.2) with $\mu(X(t_{j-1}))$, $\mu(X(t_j))$, p_1, p_2 and the most recent patch in this sequence.
if $V \leq V_{\min}$ **then**
$T_k = T_{range}(L)$. Break.
else if $V \geq V_{\max}$ **then**
$T_k = T_{range}(1)$. Break.
end if
Calculate $\mu_L^P(t)$ and $\mu_Q^P(t)$ with $\mu(X(t_{j-4}))$, ... , $\mu(X(t_{j-1}))$ and t_{j-4}, ... , t_{j-1}.
Decide the fit $\mu_{op}^P(X(t))$ by comparing the values of $\left|\mu_L^P(t_j) - \mu^P(X(t_j))\right|$ and $\left|\mu_Q^P(t_j) - \mu^P(X(t_j))\right|$.
$T_k = T_{range}(1)$. $i = 1$.
while $i < L$ **do**
if $\left|\mu_{op}^P(t_j + T_k) - \mu^P(X(t_j))\right|$ >threshold **then**
Break.
else
$T_k = T_{range}(i + 1)$. $i = i + 1$.
end if
end while
$k = k + 1$.
end while
return $T = \min_k T_k$.

A short visual description of the algorithm is detailed in Fig. 2.1. An example of a compressed frame using a frame rate of 4 is shown in Fig. 2.1(a) and its corresponding smoothed version is shown in Fig. 2.1(b). The smoothed version is blurry due to the temporal averaging (frame compression) and the spatial averaging filter. The predicted frame rates for each patch is given in Fig. 2.1(c). In Fig. 2.1(d), the region of predicted motion is highlighted, it contains the car and shadow as well as a few patches from its previous location.

Remarks:

- Depending on the data acquisition method, there are two types of data with which the algorithm can make a decision for the right polynomial. Besides testing the polynomial values at $X(t_j)$, we can also consider $A(t_j+1)F(t_j+1)$, the first frame in the uncompressed sequence with spatial mask. Since this method can be run in real-time, we can acquire this without calculating the next T. Hence the input data can also be chosen as

  $\{X(t_{j-2}),\ X(t_{j-1}),\ X(t_j), A(t_j+1)F(t_j+1)\}$ and $\{t_{j-2},\ t_{j-1},\ t_j,\ t_j+1\}$.

- Defining T as the minimum of all the T_k (the predicted compression rate from the k^{th} patch sequence) can be restrictive, allowing outlier values of T_k to dominate in the estimate of T. To avoid this issue, two possible methods can be used. The first is to sort the set $\{T_k\}_k$ and use the ordered data to determine the value T. For example, we could pick the smallest or an average of the pth smallest values. This can be costly, since it may encourage conservative values due to outliers, so instead we introduce another parameter T_{thresh} to relax this minimum. When the ratio of the number of minimum value over $\frac{MN}{p_1 p_2}$ (the cardinality of the T_k set) is smaller than T_{thresh}, we define T as the next compression level in T_{range}, essentially taking the minimum value over a more effective set. By doing so we remove outliers in the data, but we only move up one compression level in order to prevent over estimation. In general, this can be seen as an ordered weighted average (a weighted average on the sorted data set), in which the weights are determined adaptively based on the support of the smallest block-wise compression rate.

- Both the linear and quadratic fits are related to different types of physical motion present in video data. The linear polynomial gives an estimation of

the dynamics:

$$\mu_L^P(t) = \mu_0 + \partial_t \mu \; t \tag{2.3}$$

while the quadratic fit yields:

$$\mu_Q^P(t) = \mu_0 + \partial_t \mu \; t + \frac{\partial_t^2 \mu}{2} t^2 \tag{2.4}$$

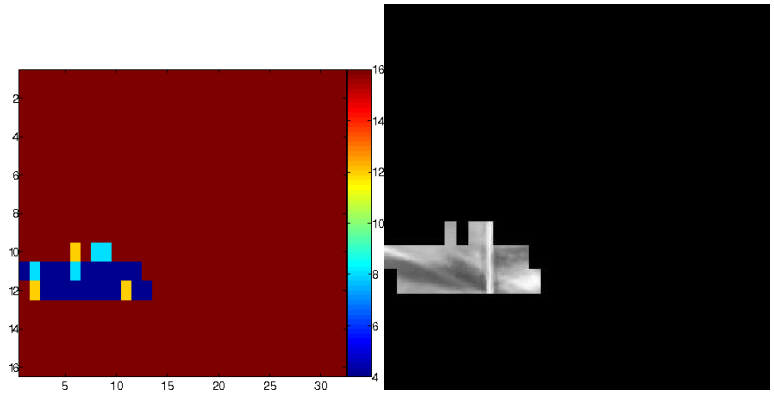
In the linear case, the polynomial estimates objects which move with velocity dominated motion, i.e. $|a| \ll |V|$ where a is the acceleration. The quadratic term in the polynomial fitting gives information on both the average tangential acceleration of objects in the patch as well as the twisting, stretching, and bending forces created by the velocity field. The additional knowledge can give a more appropriate approximation when the objects movement is governed by higher order effects. Therefore, the dynamics of the patch provide information on both the geometry and kinetics of the underlying video data. Furthermore, it can be shown that the second derivative of the mean is related to elastic forces and accelerations in the following way:

$$|\partial_t^2 \mu^P(t)| \approx \left| \left\langle V \cdot \nabla^2 f V \right\rangle_P + \left\langle \nabla f \cdot a \right\rangle_P \right| \tag{2.5}$$

where $\langle \cdot \rangle_P$ is the mean over the patch. Since the polynomials estimate dynamics only locally in time and space, the overall method can approximate a wide range of possible movements. The equations above will be made formal in Section 2.3.

(a) Compressed Frame

(b) Average Filter applied to Compressed Frame



(c) Patchwise Predicted Compression Rates

(d) Region of Predicted Motion

Figure 2.1: Example of the motion detection element of our algorithm. The compressed frame (a) and the smoothed version (b) depict the input data seen by the algorithm. The region of non-trivial motion detected in the patches is shown in (c) accompanied by the patchwise compression rates in (d). We see that the car and its shadow are the fast moving elements, as expected.

2.3 Analytic Remarks

There are several techniques and assumptions which are made in order to produce our proposed model. In this section, we explain in detail the connection between the variables, parameters, and assumptions.

2.3.1 Polynomial Extrapolation

First, let us investigate the relationship between our variable of interest $\partial_t \mu$ and the classical optical velocity V. In the ideal case, we can define $\mu^P(t)$ to be the mean of frame f over patch P at a given time t (not the compressed frame). We assume that at a given frame, future frames can be locally approximated as smoothly generated displacements. Formally we have the following definition.

Definition 2.3.1. *We say that a set of frames are* ***temporally consistent*** *if they can be generate by a smooth displacement of the initial frame. More precisely, the sequence is generated by two functions* $(f(x,0), D(x,t)) \in C^1 \times C^1([0,T]; BV)$ *where future frames are related by* $f(x,t) = f(D(x,t),0)$.

This formulation is motivated mathematically and physically. The definition above is mainly used as a local approximation to the temporally behavior of the frames. In particular, if we start with a frame f (setting it to $f(x,0)$), the next frame, over some time dt, will be given by $f(x,dt) := f(D(x,dt),0)$, where $D(x,dt)$ is the deformation of the pixels between the two frames over the small time interval. The rest configuration of the deformation is assumed to be the identity. We will also assume $D \in BV$ and $f \in C^1$, which is true for our algorithm since we consider smooth frames with patchwise (possibly discontinuous) motion.

Definition 2.3.2. *A temporal displacement function* $D(x,t)$ *is* ***velocity dominated****, if its acceleration is smaller than the velocity, in particular* $||\partial_t^2 D(x,t)|| << ||\partial_t D(x,t)||$. *On the other hand if the first and second time derivatives are on the*

same order then we say $D(x,t)$ *is* ***accelerated driven*** *.*

We assume that typically observed motion is well-approximated by these two behaviors. From a mathematical perspective, these conditions reduce the local dynamics and provide sufficient conditions for polynomial approximations. From the physically perspective, the underlying assumption is that the observed motion is regular, which is common for people, cars, natural objects, etc. In the ideal case though, the velocity of the foreground is restricted to locally constant motion, which we make formal in the following definition.

Definition 2.3.3. *A temporal displacement function* $D(x,t)$ *is* ***piecewise rigid****, if for any* t*, we have* $\nabla\partial_t D(x,t) \equiv 0$ *over each patch. Specifically, we consider* $\partial_t D(x,-)$ *to be in the patchwise constant (a subset of* BV*).*

The definitions above are related to the standard assumptions in optical flow [21, 22, 23] as well as image registration [26, 27, 28, 29, 30]. In fact, we can show that in some limit, our model recovers the first-order optical flow equation:

$$\partial_t f - \nabla f \cdot V = 0$$

for the ideal sequence $f(x,t)$.

Theorem 2.3.1. *Let* $f(x,t)$ *be a temporally consistent sequence of frames generated by a velocity dominated displacement. Then the following hold:*

1. *If* $\theta(x)$ *is the angle between* ∇f *and* $\partial_t D(x,t)$ *at* $t = 0$ *in patch* P *and the angle is bounded by* $||\theta||_{L^\infty(P)} < \epsilon$*, then*

$$|\partial_t \mu^P(t)| = \frac{1}{|P|}\int_P |\nabla f(D(x,dt),0)|\ \ |V(x)|\,dx + \mathcal{O}(dt) + \mathcal{O}(\epsilon) \tag{2.6}$$

2. *If $D(x,t)$ is also patchwise rigid then*

$$|\partial_t \mu^P(t)| = \frac{|V(P)|}{|P|} \|f(x,dt)\|_{TV(P)} + \mathcal{O}(dt) + \mathcal{O}(\epsilon) \tag{2.7}$$

where $V(P)$ is the patch velocity.

3. *As $|P| \to 0$, we recover the first-order optical flow equation.*

Proof. To show 1. we differentiate the mean of the frame $f(x,t)$ at time dt. First, let $|P|$ be the area of the patch, then by [31, 32] we have

$$\begin{aligned}
\partial_t \mu^P(t) &= \frac{1}{|P|} \int_P \partial_t f(x,dt) dx \\
&= \frac{1}{|P|} \int_P \partial_t f(D(x,dt),0) dx \\
&= \frac{1}{|P|} \int_P \nabla f(D(x,dt),0) \cdot \partial_t D(x,dt) dx
\end{aligned}$$

evaluated at time dt. Next, we use the assumption that $D(x,t)$ is velocity dominated to expand the time derivative of the displacement $\partial_t D(x,dt) = \partial_t D(x,0) + \mathcal{O}(dt)$. Using this Taylor expansion and the fact that $f \in C^1$ (specifically, the fact that f has bounded derivatives) we have:

$$\partial_t \mu^P(t) = \frac{1}{|P|} \int_P \nabla f(D(x,dt),0) \cdot V(x) dx + \mathcal{O}(dt) \tag{2.8}$$

where we define $V(x) := \partial_t D(x,0)$ for simplicity. Next, from the assumption on

the angle between the image gradients and velocity, we have

$$\partial_t \mu^P(t) = \frac{1}{|P|} \int_P |\nabla f(D(x, dt), 0)| \ \ |V(x)| \cos(\theta(x)) dx + \mathcal{O}(dt) \tag{2.9}$$

Lastly, Equation (2.6) is achieved via the small angle approximation, $\cos(\theta(x)) = 1 - \mathcal{O}(\theta(x)^2)$.

For 2, we can easily see that if $D(x, t)$ is patchwise rigid, then

$$\begin{aligned} \partial_t \mu(f, P) &= \frac{1}{|P|} \int_P |\nabla f(D(x, dt), 0)| \ \ |V(x)| \, dx + \mathcal{O}(dt) + \mathcal{O}(\epsilon) \\ &= \frac{|V(P)|}{|P|} \, \|f(x, dt)\|_{TV(P)} + \mathcal{O}(dt) + \mathcal{O}(\epsilon) \end{aligned}$$

where $V(P)$ is the patch velocity.

And lastly, for 3, to show that the model recovers the optical flow equation recall

$$\frac{1}{|P|} \partial_t \int_P f(x, dt) dx = \frac{1}{|P|} \int_P \nabla f(D(x, dt), 0) \cdot V(x) dx + \mathcal{O}(dt) \tag{2.10}$$

At $dt = 0$, we can differentiate under the integral since f is smooth:

$$\frac{1}{|P|} \int_P \partial_t f(x, 0) dx = \frac{1}{|P|} \int_P \nabla f(x, 0) \cdot V(x) dx \tag{2.11}$$

therefore we have

$$\frac{1}{|P|}\int_P \left(\partial_t f(x,0) - \nabla f(x,0)\cdot V(x)\right) dx = 0 \tag{2.12}$$

By Lebegue differentiation theorem, as $|P| \to 0$ the integrand goes to zero, thus $\partial_t f(x,0) = \nabla f(x,0)\cdot V(x)$ a.e., which is the first-order optical flow equation at $t = 0$. □

Remark 2.3.1. *Note that the patchwise rigid restriction in Theorem 2.3.1 can be relaxed to having $||\nabla V(x,0)||_{L^p}$ small (by Poincaré-Wirtinger inequality).*

Theorem 2.3.1 provides the mathematical connection between the ideas presented in this work with the classical optical flow and block matching. The assumptions in the theorem are also related to the physical motion of objects in the frame. For example, the assumption on the angle $\theta(x)$ is equivalent to assuming that the velocity field is applied nearly parallel to the gradients of the dynamic objects in the video.

For the quadratic approximation, second order time derivatives of the patch mean must be consider. The following theorem provides the relationship between the $\partial_t^2 \mu^P(t)$ and image characteristics.

Proposition 2.3.1. *Let $f(x,t)$ be temporally consistent sequence of images generated by a acceleration driven displacement then*

$$\partial_t^2 \mu^P(t) = \frac{1}{|P|}\int_P V(x)\cdot\nabla^2 f(D(x,dt))V(x) + \nabla f(D(x,dt))\cdot a(x)\ dx + \mathcal{O}(dt)$$

Proof. The proof is very similar to Theorem 2.3.1, but instead of taking a first order approximation $D(x,dt)$ in time we take a second order approximation:

$D(x, dt) = x + \partial_t D(x, 0)dt + \partial_t^2 D(x, 0)\frac{dt^2}{2} + \mathcal{O}(dt^3)$. Once again, differentiating and expanding yields:

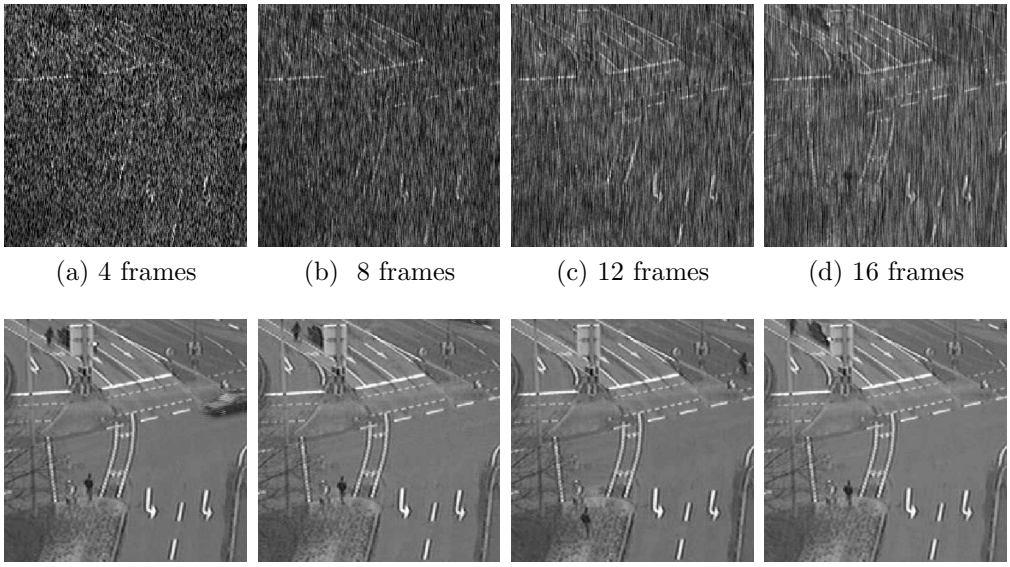
$$\begin{aligned}
\partial_t^2 \mu^P(t) &= \partial_t \frac{1}{|P|} \int_P \nabla f(D(x, dt)) \cdot \partial_t D(x, dt) dx \\
&= \frac{1}{|P|} \int_P \partial_t D(x, dt) \cdot \nabla^2 f(D(x, dt)) \partial_t D(x, dt) \\
&\qquad + \nabla f(D(x, dt)) \cdot \partial_{tt} D(x, dt)\ dx \\
&= \frac{1}{|P|} \int_P V(x) \cdot \nabla^2 f(D(x, dt)) V(x) \\
&\qquad + \nabla f(D(x, dt)) \cdot a(x)\ dx + \mathcal{O}(dt)
\end{aligned}$$

which provides another relationship between the image gradients, physical characteristics, and algorithmic terms. □

The physical interpretation of this approximation is that there is not a large amount of rotational motion, which is an appropriate assumption for surveillance, tracking, traffic, etc.

2.4 Experimental Results

In this section we use numerical experiments to demonstrate the robustness and efficiency of our algorithm. As shown in [17], the shifted masks will give comparable reconstruction results compared with completely random masks. Hence in most of our tests we first generate a random binary mask with 50% zeros, and keep shifting it in one direction to get subsequent masks. Other types of masks will also be considered in a later test. In our test, four different compression rates are considered, 4,8,12 and 16. Fig. 2.2 displays some selected compressed frames using different compression rates. In Fig. 2.2(a), since so few frames are averaged the effective spatial mask appears to be random, while in Fig. 2.2(b-d) as the frame rate increases so does the clarity. However, although the resolution



(a) 4 frames (b) 8 frames (c) 12 frames (d) 16 frames

(e) Frame from (a) (f) Frame from (b) (g) Frame from (c) (h) Frame from (d)

Figure 2.2: In (a-d), various compressed frames are shown. Each scene is compressed with a different rate and a frame from the scene is displayed in (e-h). The hierarchy shows that the scene with the car suddenly entering in (a) and (e) has the smallest compression rate while the one with pedestrian motion has the highest rate. The medium compression rates, as seen in (b)&(f) or (c)&(g), are related to the car entering in the top left quadrant and the movement of the second pedestrian, respectively.

increases with the frame rate, the trade-off is that the image becomes blurred. In essence, this is the balance in adaptive compressive video sensing.

Before getting into our algorithm, we would like to first show the importance of adaptiveness in video compression through some experiments. Let us first look at how T influences the reconstruction results of different types of video data. Here in each test T frames are compressed into one according to (2.1), then we apply TV-based video reconstruction algorithm on this compressed data to recover the original frame sequence, and the average PSNR (Peak signal-to-noise ratio) of the reconstructed sequence will be recorded.

Inspired by the reconstruction models from [33, 34], we adapt those previously proposed models to recover the compressed video. Our reconstruction model is as

follows.

$$\min_{F_i} \sum_{i=1}^{T} |DF_i| + \lambda \sum_{i=1}^{T-1} |F_{i+1} - F_i|, \ s.t. \ \sum_{i=1}^{T} A_i F_i = X \tag{2.13}$$

where D is the forward spatial derivatives. Both regularizers in this model are of L^1 type, therefore the minimization can be efficiently solved via the split Bregman method [35].

We first introduce two auxiliary variables G_i for $i = 1, \ldots, T$ and d_i for $i = 1, \ldots, T-1$ and the Bregman variables (constraint enforcing) X^k, B^k and b^k are the Bregman variables so that the Equation (2.13) becomes:

$$\begin{aligned} &\min_{F,G,d} \sum_{i=1}^{T} |G_i|_1 + \lambda \sum_{i=1}^{T-1} |d_i|_1, \\ &\text{s.t. } G_i = DF_i, \ d_i = F_{i+1} - F_i, \ \sum_{i=1}^{T} A_i F_i = X \end{aligned} \tag{2.14}$$

The constraints are incorporated into the energy as follows.

$$\begin{aligned} (F^k, G^k, d^k) = \arg\min F, G, d & \sum_{i=1}^{T} |G_i|_1 + \lambda \sum_{i=1}^{T-1} |d_i|_1 \\ & + \frac{\mu_1}{2} \| \sum_{i=1}^{T} A_i F_i - X + X^{k-1} \|^2 \\ & + \frac{\mu_2}{2} \sum_{i=1}^{T} \| G_i - DF_i + B_i^{k-1} \|^2 \\ & + \frac{\mu_3}{2} \sum_{i=1}^{T-1} \| d_i - F_{i+1} + F_i + b_i^{k-1} \|^2 \\ X^k = & \sum_{i=1}^{T} A_i F_i^k - X + X^{k-1} \\ B_i^k = & G_i^k - DF_i^k + B_i^{k-1} \\ b_i^k = & d_i^k - F_{i+1} + F_i + b_i^{k-1}. \end{aligned}$$

For the results in this work we take $\mu_2 = \mu_3$. The minimizers for G^k and d^k are explicit:

$$\begin{aligned} G_i^k =& \text{shrink}(DF_i^{k-1} - B_i^{k-1}, 1/\mu_2) \\ d_i^k =& \text{shrink}(F_{i+1}^{k-1} - F_i^{k-1} - b_i^{k-1}, \lambda/\mu_3) \end{aligned}$$

where the shrink function is defined for vectors by: $\text{shrink}(\cdot, \tau) := \max(\|\cdot\| - \tau, 0)\frac{\cdot}{\|\cdot\|}$.

For the F variable update, the minimizing equation is the following linear system

$$\begin{aligned} (\mu_1 A^T A + \mu_2 D^T D + \mu_3 D_3^T D_3)F =& \mu_1 A^*(X - X^{k-1}) + \mu_2 D^T(G^k + B^{k-1}) \quad (2.15) \\ &+ \mu_3 D_3^T(d^k + b^k) \quad (2.16) \end{aligned}$$

where D_3 is the forward difference with respect to the frame (not to be confused with the spatial differences).

Altogether, we alternate the shrinkage steps with a few iterations of conjugate gradient to solve Equation (2.15) in order to find F. The convergence of this algorithm to the correct minimizer is guaranteed, for example see [36].

Two types of video data are considered in the test, moving frames and frozen frames, where moving frames mean all the T frames are different from each other, while frozen frames stand for the case with T identical frames. The results are recorded in Table 2.1. We can see from the table that when we have stationary video data, a larger T value usually leads to better reconstruction results with higher PSNR values. On the other hand, when there are a lot of movements in the video, a smaller T is often more desirable.

Based on the above observation, we then use numerical tests to check the ad-

Table 2.1: Mean PSNR comparison for different types of video data

	Moving frames		Frozen frames	
	T = 4	T=16	T=4	T=16
Test 1	33.7849	28.2370	48.4005	75.3180
Test 2	37.7345	34.9292	45.4085	67.1824

vantage of adaptive compression over fixed rate compression. In the first setting, we generate a video of 32 frames, where the first 16 frames contain a lot of movements, while the rest are identical. We then manually compress the video into 5 frames, where $T_1 = \ldots = T_4 = 4$ and $T_5 = 16$. According to our assumption on T_{range}, this is the optimal way of adaptive compression for this particular video. We also define another compression by setting $T_1 = 8$ and $T_2 = \ldots = T_5 = 6$, and this is close to the fixed rate compression. For each case, the above reconstruction algorithm (2.13) is applied on these compressed frames to recover each original frame sequence separately, and the average PSNR of each sequence is recorded.

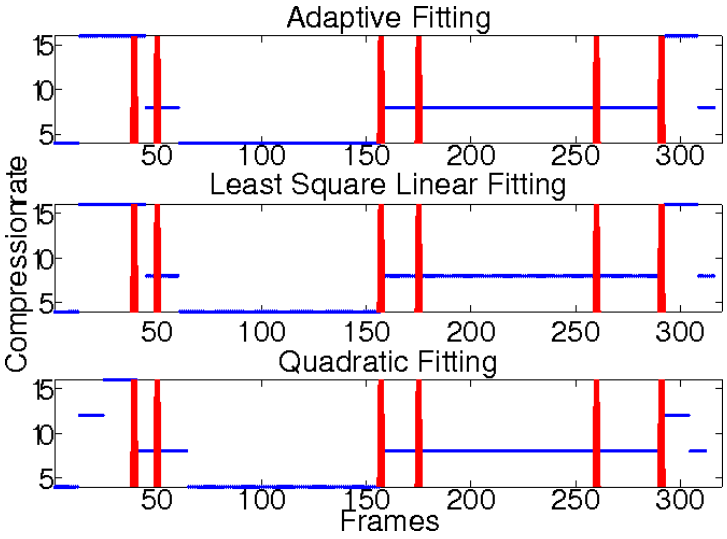
In the second setting, the same two compression strategies are used on the video with 32 moving frames. The mean PSNR are displayed in the following table. Similar tests are also conducted on videos where the first 16 frames are identical while the rest are moving frames.

We can see from the PSNR values in Setting 1 that adaptive compression leads to much better reconstruction results. The results in Setting 2 and 3 justify matching the compression with the motion, and not the choice of this particular partition, is responsible for the PSNR gain. In particular, we see that the results of the reconstruction algorithm can support the choice of compression rate as long as there is a significant gain in the PSNR.

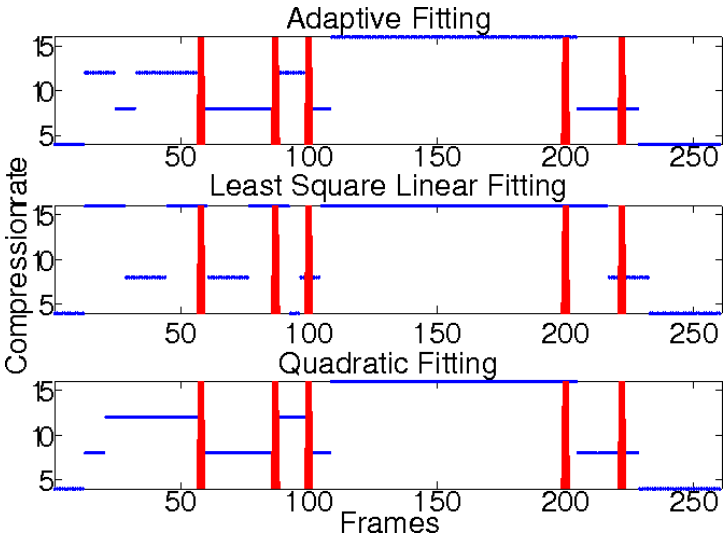
Table 2.2: Mean PSNR comparison between adaptive and fixed rate compression

	Setting 1		Setting 2		Setting 3	
	Adaptive	Fixed	Adaptive	Fixed	Adaptive	Fixed
Test 1	**53.1740**	40.6072	33.8991	**35.0905**	39.9775	**42.6081**
Test 2	**55.9060**	43.0610	36.5681	**37.2725**	39.7499	**43.2499**

2.4.1 Behavior of Adaptive Polynomial Fitting



(a) Linear Favored



(b) Combination

Figure 2.3: Comparison between adaptive polynomial fitting versus fixing the degree of the polynomial. Using different data (a) and (b) shows that the adaptive polynomial fit may favor one polynomial or use a combination of both.

In Figure 2.3, the temporal compression rates are plotted for each frame in two real data sets. The red markers indicate changes in the video sequence, for example an object entering or leaving the field of view, while the blue dots stand for the compression rate for each frame. To investigate the effect of our adaptive

polynomial fitting step in our algorithm, we compare our method to the case when we restrict the approximating polynomial to be either linear or quadratic only. In Figure 2.3 (a), the algorithm is applied to parking lot surveillance data, containing a static background with moving people and vehicles. The spatial compression rate is taken to be 50%. The first 12 frames are assigned a compression rate of 4 in order to generate input data to our algorithm. The initial computed compression rate is 16 since there is no movement, and decreases to 8 and 4 as the car enters (the first two red markers, where the car starts to enter at the first marker, and fully enters at the second marker). The second two markers are at the frame location when the car gradually stops and people enter the field of view. Since the dynamic component of the video is slowing down, the compression rate should increase, coinciding with our algorithm's performance. At the end of the sequence the moving people become obscure (effectively exiting the field of view) and reappear, which creates a jump in the compression rate. In this case, the adaptive fitting prefers the least squares linear fit, since most of the motion is locally constant.

In Figure 2.3 (b), the algorithm is applied to traffic data. Prior to the first marker, the main moving component consists of non-uniform pedestrian movement, therefore a medium-level compression rate is favored. This can be seen visually and agrees with our algorithm's performance. The first two markers shows the occurrence of a vehicle entering the field at various speeds with varying visibility. Therefore, we expect the compression rate to drop. This is exhibited by both the adaptive compression algorithm and the quadratic approximation. The next two markers bound the interval in which the frames are still. And the last signifies a fast moving car entering the video. In this case, the adaptive algorithm incorporates information from the quadratic fitting while also capturing information (near frame 35) that is overlooked using one polynomial exclusively.

In general, the linear fit favors consistent motion, since it approximates one

velocity over several compressed frames. The quadratic fit captures more subtle dynamics, shown through its ability to adjust to gradual changes; however, at times this results chooses the more prudent compression rate. Since the approximations are done patch by patch, one interesting observation to note is that the adaptive compression rate does not necessarily give you either the linear or the quadratic fitting result at any given frame. Instead, it balances the contributions from both approximations.

2.4.2 Comparison of Patch Size

In Figure 2.4, we provide a comparison between different patch sizes. To measure optimality of the patch size, we look at the qualitative behavior of the compression results and the quantitative results (compression rate and PSNR). The optimal patch size for the parking lot data set is 16 by 8 (with an average PSNR of 45.45) and for this reason it is the patch configuration used in other sections. Figure 2.4 displays the result for patch configurations of 8 by 8 (average PSNR of 41.37), 8 by 16 (average PSNR of 44.96), and 16 by 16 (average PSNR of 45.15).

The patch size of 8 by 16 gives similar results; however, it neglects the motion of the pedestrians. The small square patch of size 8 by 8 is more sensitive to small changes between frames. The larger square patch of size 16 by 16 is less sensitive to small objects. The 16 by 16 patch size results has issues reconstructing the final part of the video data, where the small scale motion is dominant. The 8 by 8 patch size consistently gives too low of a compression rate, since it is sensitive to outliers, thus making it ineffective as a data reduction tool.

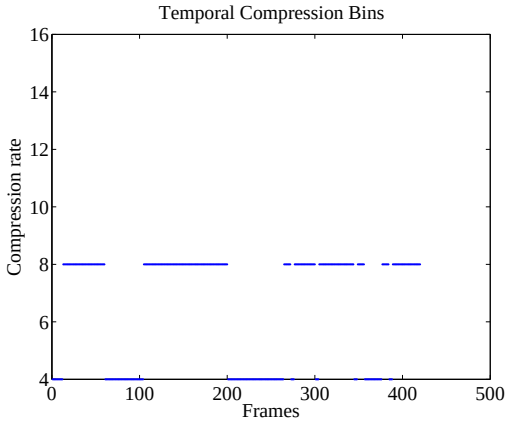
From this we can conclude that the patch size is determined by two factors: the scale of motion and its directionality. For consistent velocity V over a sampling time of Δt, one could argue that the diameter of the patch should satisfy the relationship $\text{diam}(P) = V\Delta t$ to capture the correct scale. To correctly resolve

directionality, the patch should be shorter in the direction of fast motion and long in the direction of slow motion. For the data set tested here, the patch size of 16 by 8 corresponds to the correct size given this analysis as well. This also shows that since 8 by 16 gives similar results while 8 by 8 gives worse results, scale plays a more important role then directionality.

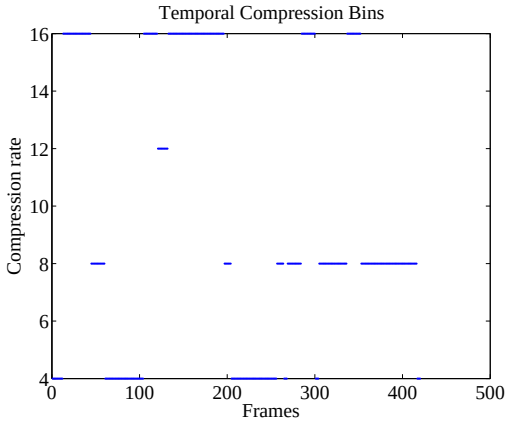
2.4.3 Comparison with Fixed Rate Compression

As seen in the tables, reconstruction algorithms for video compressive sensing provide satisfactory results when (and only when) many frames are averaged over low motion scenes or when few frames are averaged during high motion scenes. Therefore, since our algorithm takes advantage of this principle, we would expect that it should yield a better recovered video than using a fixed rate compression. In Fig. 2.5 and 2.6, the compression rate versus frame rate is plotted along with the PSNRs of each frame after applying the reconstruction algorithm. For the results displayed in Fig. 2.5, the mean PSNR is 45.43 dB, compared to a mean of 40.56 dB when applying a fixed rate compression with the same number of compressed frames (i.e. identical mean compression rates). The maximum and minimum PSNRs of our algorithm is 77.90 dB and 30.97 dB respectively, while the fixed rate compression yields 55.07 dB and 28.82 dB respectively. For the results in Fig. 2.6, our mean PSNR is 49.58 dB and the fixed rate compression has an average PSNR of 42.92 dB. The maximum and minimum PSNR of our method is 77.20 dB and 30.95 dB, while the fixed rate compression yields 52.67 dB and 28.31 dB, respectively. In both cases, our compression method provides significant gains in the average PSNR of the reconstructed image.

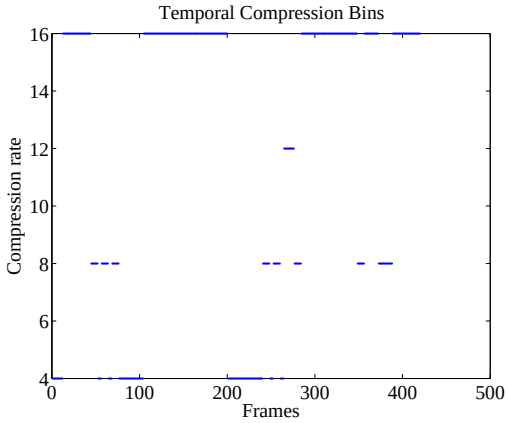
In Fig. 2.7, six reconstructed frames (Frame 93 to 98) from the video used in Fig. 2.5 are displayed. In Fig. 2.8, the same six frames are shown which are reconstructed from data compressed with a fixed rate. The results using our algorithm are of higher quality in the regions containing fast motion than that of a fixed



(a) Patch Size of 8 by 8



(b) Patch Size of 8 by 16



(c) Patch Size of 8 by 16

Figure 2.4: Comparison of compression results for various patch sizes using our algorithm.

rate. Also, since the temporal compression averages consecutive frames, motion elements in some frames can pollute both the compression and reconstruction of

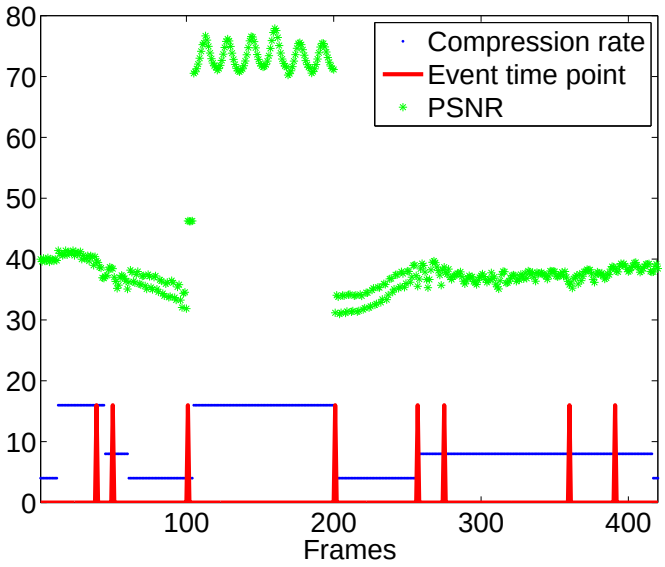


Figure 2.5: The compression rates (in blue) and the corresponding PSNRs (in green) of the reconstructed video method to the results generated by our algorithm to the surveillance data set. To increase the range of motion in our video data, we freeze the video at the 100^{th} frame and resume the original video at the 201^{st} frame.

neighboring frames, as seen by the errors incurred around the car in Fig. 2.6. In these figures, it is clear that the adaptive compression method yields less motion and compression artifacts than that of the fixed rate compression, thus resulting in better overall visual quality of the reconstructed video.

2.4.4 Robustness to Compressive Sensing Matrix

Lastly, we apply our algorithm to the case when the spatial compressive sensing mask is randomly generated for each frame (retaining 45% of the pixels for any given frame). In Fig. 2.9, the mean PSNR is 44.63 dB (39.97 dB for a fixed rate compression) with a maximum and minimum PSNR of 74.98 dB and 30.82 dB respectively (53.77 dB and 28.89 dB for a fixed rate compression). This is comparable with the results from Fig. 2.5, since the algorithm does not depend explicitly on the manner in which the mask is generated. Since the compression algorithm considers average patch information, a particular mask realization should not alter the patch means significantly. This shows the potential of incorporating our

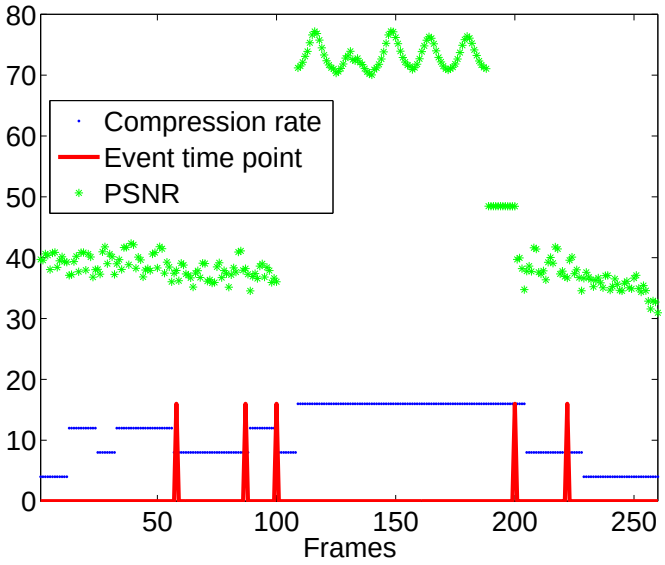


Figure 2.6: The compression rates (in blue) and the corresponding PSNRs (in green) of the reconstructed video method to the results generated by our algorithm to the traffic data set.

Figure 2.7: The reconstruction results of six consecutive frames compressed by our algorithm. The PSNRs for the first row starting from the left is: 33.7896 35.7451 36.1793 and the PSNRs for the second row starting from the left is: 33.7118 32.6931 34.7961.

Figure 2.8: The reconstruction results of six consecutive frames compressed using a fixed rate, chosen to match the average compression rate of our algorithm. The PSNRs for the first row starting from the left is: 32.5228 33.9217 34.5397 and the PSNRs for the second row starting from the left is: 33.9049 32.0122 29.9636.

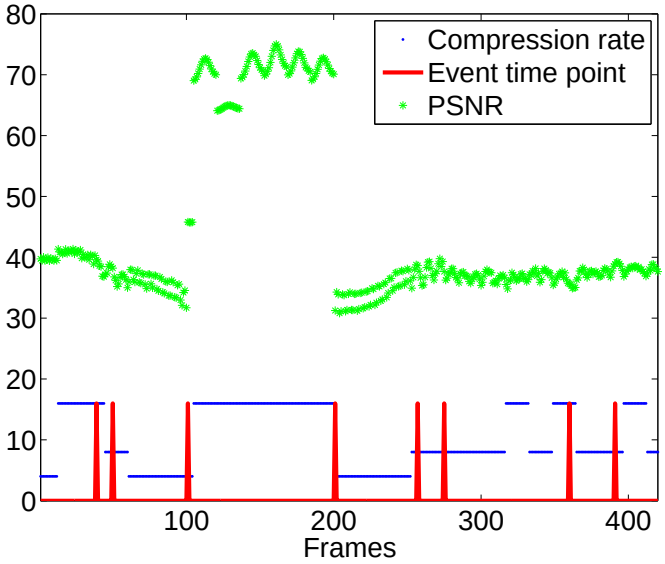


Figure 2.9: The compression rates (in blue) and the corresponding PSNRs (in green) of the reconstruction method applied to the results generated by our algorithm to the data that is acquired via a random CS mask. Notice the qualitative and quantitative similarity to Fig. 2.5.

algorithm in various video compression applications.

2.5 Conclusion

We present an adaptive algorithm for predicting compression rates in real-time. The underlying idea is simple: compress more when the video contains little motion and compress less during dynamic scenes. Using this idea, we build an efficient and compact way to estimate the motion of a sequence of compressed and subsampled frames using patch means. By considering the patch means, we reduce the size of the problem and decouple each task. Also, by using each individual patch's history to predict its own compression rate, we accelerate the time it takes to compute the predicted frame rate. Our adaptive model is shown to improve the PSNR of the reconstructed video by several dB as well as the visual quality of the images.

CHAPTER 3

Mixing Space-Time Derivatives for Video Compressive Sensing

Abstract

With the increasing use of compressive sensing techniques for better data acquisition and storage, the need for efficient, accurate, and robust reconstruction algorithms continues to be in demand. In this work we present a fast total variation based method for reconstructing video compressive sensing data. Video compressive sensing systems store video sequences by taking a linear combination of consecutive spatially compressed frames. In order to recover the original data, our method regularizes both the spatial and temporal components using a total variation semi-norm that mixes information between dimensions. This mixing provides a more consistent approximation of the connection between neighboring frames with little to no increase in complexity. The algorithm is easy to implement since each iteration contains two shrinkage steps and a few iterations of conjugate gradient. Numerical simulations on real data show large improvements in both the PSNR and visual quality of the reconstructed frame sequences using

our method.

3.1 Introduction

Compressive sensing (CS) techniques have a large range of applications from hardware development using 1 bit compressive sensing [38, 39] or coded apertures [2, 3] to methodologies using CS sampling methods [12]. Its popularity stems for the efficient way of handling large data sets - a growing issue in all fields and applications. For example in information science, large collections of images and videos are obtained at extremely high rates, thereby efficient transmission and storage accompanied by robust and accurate reconstruction is necessary. One recent device, the coded aperture compressive temporal imaging (CACTI) [17], shows promise on encoding optical data entering at rates approaching 1 exapixel/second. In fact, an emerging application of the CS paradigm, called video compressive sensing (VCS), has enabled high quality reconstruction of videos from very few observed frames.

In terms of the mechanics, VCS methods use physical techniques to code incoming optical data in order to compress either spatial or spectral information. For spatial compression, this is typically accomplished by using a mechanical grating (the coded aperture) to block the entering data stream, thus subsampling the incoming signal. There are several ways to compress the temporal data [13, 15], for this work we consider the methods employed in the compressive sensing framework. Here, we assume that the final stored measurement is defined as a linear combination of several spatially compressed frames, thereby removing both the spatial and temporal redundancy [18, 40].

The underlying forward model we consider is as follows: let X be the compressed data set, $F = [F_1, \ldots, F_T]$ be the vector formed by all the original frames, where T is the number of frames that are compressed into one. Define A as the

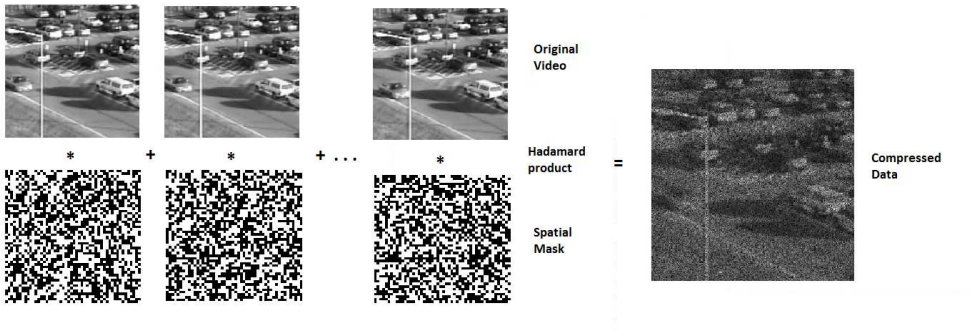


Figure 3.1: Illustration of the compression/encoding process of the video data.

CS matrix containing the random frame by frame masks as well as the temporal compression. In other words, A can be written as $[A_1, \ldots, A_T]$ with each A_i being a random binary matrix, representing which pixels can store information. X is acquired by taking the sum of $A_i F_i$ for all the i's with the product understood as the Hadamard product. For simplicity, here we denote this compression process as $X = AF$. The CS mask can be generated in two ways, depending on the application. The first is to independently generate A_i for each frame, a common technique in compressive sensing. The second way is to generate an initial mask A_1 for the first frame and shift it by one pixel (in a known direction) for each subsequent frame. In hardware, this amounts to translating the coded aperture between frames, which is easy to implement, requires less storage, and usually yields similar results as first way [17]. Fig. 3.1 illustrates the compression process for our problem.

Similar to other compressive sensing problems, the linear system $X = AF$ is underdetermined since there are much fewer measurements than unknown variables. In this way, we can greatly reduce the amount of information stored while still being able to recover the original signal with high accuracy. Our goal is to reconstruct F given A and X by leveraging certain desired sparsity properties on the original video. For image decompression from CS measurements, recent research supports the use of total variation (TV) regularization because of its ability

to preserve edge information. With the introduction of the Bregman iteration [41] and the split Bregman algorithm (related to the alternating direction method of multipliers (ADMM) algorithm) [35], the TV-based optimization problems can be solved in an extremely efficient way with sharper contrast in the recovered image. This is of particular importance to video reconstruction, since the temporal compression causes motion blur, adding to the overall difficultly of solving the inverse problem.

Although the application of TV to imaging problems is clear, the extra dimension in videos makes extensions more ambiguous. The most direct way is to incorporate the temporal dimension as a third coordinate, which leads to the following optimization problem (TV3):

$$\min_{F} \sum_{i=1}^{T} |F_i|_{TV^3}, \text{ s.t. } \sum_{i=1}^{T} A_i F_i = X \tag{3.1}$$

where the $3d$ TV semi-norm $|\cdot|_{TV^3}$ is defined as

$$|Y|_{TV^3} := |D_3 Y|_1 = \sum_{i,j,k} |D_3 Y_{(i,j,k)}| \tag{3.2}$$

with the difference operator defined as $D_3 := [D_x, D_y, D_t]$ and the vector norm $|D_3 Y_{(i,j,k)}| := \sqrt{(D_x Y_{(i,j,k)})^2 + (D_y Y_{(i,j,k)})^2 + (D_t Y_{(i,j,k)})^2}$. The TV3 model is used by TwIST (two-Step iterative shrinkage/thresholding) [33] and TVAL3 (TV minimization by augmented Lagrangian and alternating direction algorithm) [34]. The above regularizer places equal importance in all directions and treats the pixel values as piecewise constant in time. The underlying assumption is that the movement between successive frames is small (sparse). Though this model achieves reasonable performance in application (with respect to PSNR, i.e. Peak Signal-to-Noise Ratio), we can often greatly improve both the PSNR and visual quality of the reconstruction by carefully exploring the temporal component. More

examination of this model can be found in Section 3.3.

Rather than treating the difference between successive frames as sparse, we assume that the difference is piecewise constant. This assumption is consistent with the spatial regularizer, since the typical hypothesis is that each frame in the video can be approximated by a piecewise constant image, then the difference image between two successive frames should also belong to that category. As a matter of fact, this is consistent with the dynamics captured in images: a piecewise constant object travels with a certain speed across the field of view. Therefore, instead of considering the L^1 norm of the difference image, here we propose the TV norm on the frame difference. As seen in the experimental results presented in Section 3.3, our model provides better quality results regardless of the amount of motion present in the video.

This paper is organized as follows. Section 3.2 gives a detailed description of our model and the associated algorithm. In Section 3.3, numerical simulations on real data are provided, which demonstrate the improvement in both numerical and visual quality of the reconstructed video under various circumstances. We also compare our model with several total variation based methods. Lastly, we conclude with a short discussion in Section 3.4.

3.2 Model and Algorithm Description

The mathematical formulation of our model is as follows:

$$\min_{F} \sum_{i=1}^{T} |F_i|_{TV} + \lambda \sum_{i=1}^{T-1} |F_{i+1} - F_i|_{TV}, \text{ s.t. } \sum_{i=1}^{T} A_i F_i = X \tag{3.3}$$

where $|F_i|_{TV} = |DF_i|_1$ with $D = [D_x, D_y]$. The model essentially recovers the original data via deblurring the temporal component, reconstructing the compressed spatial component, as well as interpolating the missing frames.

The first term is the $2d$ TV semi-norm and the second term is the total variation of the difference between frames. The parameter λ balances the importance between the two regularizers. In practice λ has a direct relationship to T: when T is large (small), a larger (smaller) λ yields better results. This is related to the compression methodology, since large T is used when successive frames are similar, while small T is used for highly dynamic frames.

Since the model contains L^1 type regularizers, we can solve the minimization efficiently by applying the split Bregman technique. The convergence of this algorithm to the minimizer of (3.3) is guaranteed according to [36]. Introducing auxiliary variables P_i for $i = 1, \ldots, T$ and Q_i for $i = 1, \ldots, T-1$ to substitute for the spatial derivative and the mixed gradient respectively, the above problem is equivalent to

$$\begin{aligned} &\min_{F,P,Q} \sum_{i=1}^{T} |P_i|_1 + \lambda \sum_{i=1}^{T-1} |Q_i|_1, \\ &\text{s.t. } P_i = DF_i,\ Q_i = DF_{i+1} - DF_i,\ \sum_{i=1}^{T} A_i F_i = X \end{aligned} \tag{3.4}$$

Let us define $\widehat{D}F_i = DF_{i+1} - DF_i$, which will stand for the gradient of the time

differences. Next adding back the constraint yields:

$$(F^k, P^k, Q^k) = \underset{F,P,Q}{\operatorname{argmin}} \sum_{i=1}^{T} |P_i|_1 + \lambda \sum_{i=1}^{T-1} |Q_i|_1 \tag{3.5}$$

$$+ \frac{\mu_1}{2} \| \sum_{i=1}^{T} A_i F_i - X + X^{k-1} \|^2 \tag{3.6}$$

$$+ \frac{\mu_2}{2} \sum_{i=1}^{T} \| P_i - DF_i + B_i^{k-1} \|^2 \tag{3.7}$$

$$+ \frac{\mu_3}{2} \sum_{i=1}^{T-1} \| Q_i - \widehat{D} F_i + b_i^{k-1} \|^2 \tag{3.8}$$

$$X^k = \sum_{i=1}^{T} A_i F_i^k - X + X^{k-1} \tag{3.9}$$

$$B_i^k = P_i^k - DF_i^k + B_i^{k-1} \tag{3.10}$$

$$b_i^k = Q_i^k - \widehat{D} F_i^k + b_i^{k-1}. \tag{3.11}$$

where the variables X^k, B^k and b^k are the Bregman variables which are used to enforce the equality constraint. Although the choice of μ_j will not affect the final solution, their values relate to the convergence rate. Also, in practice we can simply set $\mu_2 = \mu_3$ since they both control the gradient terms. The update for the variables P, Q and F are done in an alternating fashion by the equations below:

$$P_i^k = \text{shrink}(DF_i^{k-1} - B_i^{k-1}, 1/\mu_2) \tag{3.12}$$

$$Q_i^k = \text{shrink}(\widehat{D} F_i^{k-1} - b_i^{k-1}, \lambda/\mu_3) \tag{3.13}$$

$$F^k = \underset{F}{\operatorname{argmin}} \quad \frac{\mu_1}{2} \| \sum_{i=1}^{T} A_i F_i - X + X^{k-1} \|^2 \tag{3.14}$$

$$+ \frac{\mu_2}{2} \sum_{i=1}^{T} \| P_i^k - DF_i + B_i^{k-1} \|^2 \tag{3.15}$$

$$+ \frac{\mu_3}{2} \sum_{i=1}^{T-1} \| Q_i^k - \widehat{D} F_i + b_i^{k-1} \|^2 \tag{3.16}$$

where the shrink function above is defined pointwise for a $2d$ vector Y as

$\text{shrink}(Y, \tau) := \max(\|Y\| - \tau, 0)\frac{Y}{\|Y\|}$. For the F variable update, since the associated subproblem is differentiable, taking the first derivative and rearranging terms leads to

$$(\mu_1 A^* A + \mu_2 D^* D + \mu_3 \widehat{D}^* \widehat{D})F = RHS \tag{3.17}$$

where $(\cdot)^*$ stands for the adjoint of the operator and $RHS = \mu_1 A^*(X - X^{k-1}) + \mu_2 D^*(P^k + B^{k-1}) + \mu_3 \widehat{D}^*(Q^k + b^k)$. In each iteration of split Bregman algorithm, (3.17) only needs to be approximately solved with a few steps of conjugate gradient. Faster convergence can be obtained by using a preconditioner.

Altogether the method reduces to two explicit shrinkage steps and solving a linear system for each iteration, making the algorithm fast and easy to implement.

3.3 Numerical Experiments and Discussion

In this section we use various numerical experiments to illustrate how our model and the associated algorithm can improve the reconstruction results compared with TV3. In order to show the importance of temporal consistency in video reconstruction, we also include the following model (TV2) in our comparison:

$$\min_F \sum_{i=1}^{T} |F_i|_{TV}, \text{ s.t. } \sum_{i=1}^{T} A_i F_i = X \tag{3.18}$$

where the regularizer is decoupled. The TV2 model is able to reconstruct a reasonable result since the CS mask is random. Numerically, (3.18) is solved via the split Bregman method, similar to the algorithm described in Section 3.2. Our method converges in under a minute for all the experiments described below.

As discussed in [17], shifting the CS masks usually leads to reconstruction results that are comparable with generating a random masks for each frame. Hence

we simulate this type of mask generation during our experiments. This is done by first generate a random binary mask with a certain amount of zero elements, and translating the mask in a fixed direction after each frame. We also provide an experiment using random mask, which are generated for each frame independently. In all cases, the parameters are chosen to maximize the PSNR values.

In the first experiment (Setting 1), 4 frames are compressed into one – each with 50% spatial downsampling. According to the first row of results displayed in Table 3.1, we can see that our model achieves a much higher PSNR value compared with the results obtained from TV2 and TV3. The model without temporal regularizers provides the worst results among all the candidates with respect to the PSNR values. This justifies the importance of leveraging the relationship between neighboring frames. Note that all the methods yield a relatively good PSNR. Since the region of motion is relatively small compared to the overall size of the image, one could poorly recover the dynamic object and still have a good PSNR.

Next, in Setting 2 we keep the same conditions as in Setting 1 but input a more dynamic video (see Fig. 3.2). From Table 3.1, we see that our model yields higher PSNRs than the other models. Although all models give a relatively high PSNR, as seen in Fig. 3.2 there are clear qualitative differences. Our reconstruction has less point structures commonly associated with poor CS reconstruction. It is also able to recover the details of the background and capture the moving structures. The TV2 model provides a piecewise constant reconstruction of each frame, but it fails to recover the fine scale details found in the background. The TV3 model is able to reconstruct the background well and some of the textures, but is unable to resolve the moving object. From this example, it is clear that the temporal regularizer helps promote texture reconstruction.

In Setting 3, we increase the T value to 8 while keep all other conditions the same. The PSNR gap between our model and the other two grows, showing that

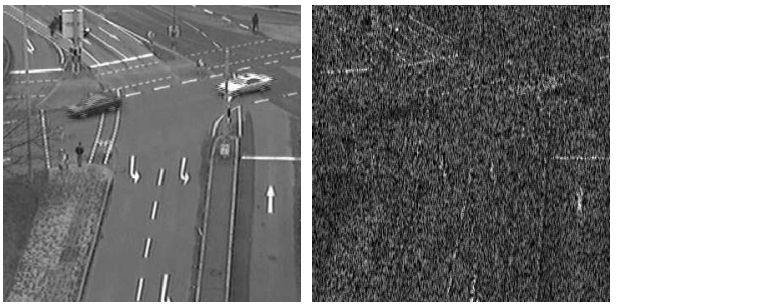
Table 3.1: Mean and maximum PSNR comparison between different video reconstruction models

	Our model		TV2		TV3	
	mean	max	mean	max	mean	max
Setting 1	**38.6716**	**39.3424**	32.3084	32.5162	36.9144	37.6050
Setting 2	**34.4394**	**34.9082**	30.8161	31.1694	31.7932	32.3711
Setting 3	**36.6830**	**37.4991**	30.0898	30.4783	34.1394	35.3124
Setting 4	**33.4960**	**34.1152**	28.8421	29.1447	31.0489	31.6782
Setting 5	**33.1679**	**33.6097**	27.8260	28.0761	31.0260	31.6416

our model is more robust to larger compression rates. Fig. 3.3 provides a visual comparison between our model, TV2 and TV3. Fig. 3.3 (a) shows an original frame from the video sequence and (b) shows the compressed frame X. Comparing the reconstructions (c-e), the zoomed in images (f-h) and difference images (i-k), we can easily see that our model gives a smoother reconstruction with sharper contrast. The loss of contrast in (d) is primarily due to the lack of temporal consistency. The point structure effect in (e) is caused by the inaccuracy of L^1 regularization for large motion. In the difference images (i-k), it is clear that our model produces the least amount of errors in the background component while recovers a piecewise smooth dynamic foreground. The PSNR gain of our model for this particular frame is more than 3.

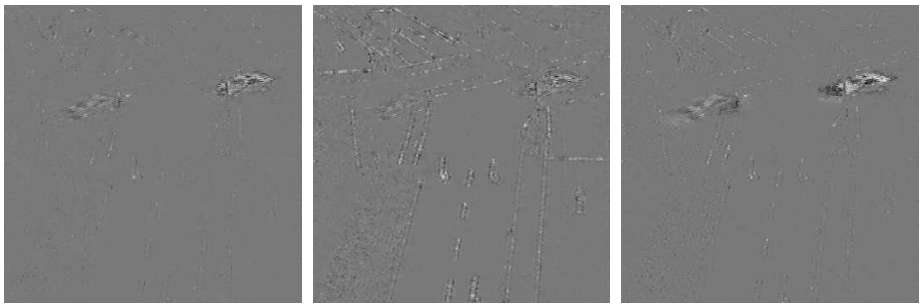
To examine the effect of both the random mask simulation and the sampling ratio, in the forth setting we generate an independent random mask for each frame and lowered the sampling ratio to 45%. The temporal compression rate, T, is set at 8 for this case. We can see that all the PSNRs have decreased as expected, while the values from our model are much higher than the others. The PSNRs for TV2 are robust in the sense that they do not change dramatically between the settings; however, they are consistently lower than the ones from other algorithms. Lastly, when $T = 12$ and the spatial compression ratio returns to 50%, again our model provides the best PSNR values among all the models.

In general, adding two regularizers for the same variable does not necessarily increase the quality of the reconstruction. In our model, since the regularizers



(a) Original (b) Compressed frame

(c) Our reconstruction (d) TV2 reconstruction (e) TV3 reconstruction



(f) Our difference (g) TV2 difference (h) TV3 difference

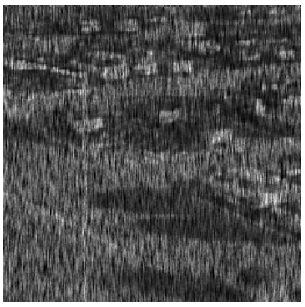
Figure 3.2: Figure illustration for Setting 2. The PSNR of this particular reconstruction with our model is 34.9082, while TV2 gives 31.1694 and TV3 gives 32.3697.

act on the different coordinates, both are necessary. In Fig. 3.4, we compare our model to the temporal-only version of our model:

$$\min_F |F_1|_{TV} + \sum_{i=1}^{T-1} |F_{i+1} - F_i|_{TV} + |F_T|_{TV}, \tag{3.19}$$
$$\text{s.t. } \sum_{i=1}^{T} A_i F_i = X$$

Note that the first and last term are also time difference with zero boundary

(a) Original



(b) Compressed frame

(c) Our reconstruction

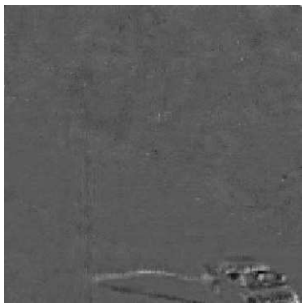
(d) TV2 reconstruction

(e) TV3 reconstruction

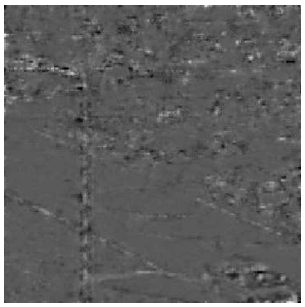
(f) Our zoomed in

(g) TV2 zoomed in

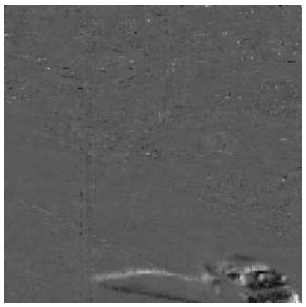
(h) TV3 zoomed in



(i) Our difference

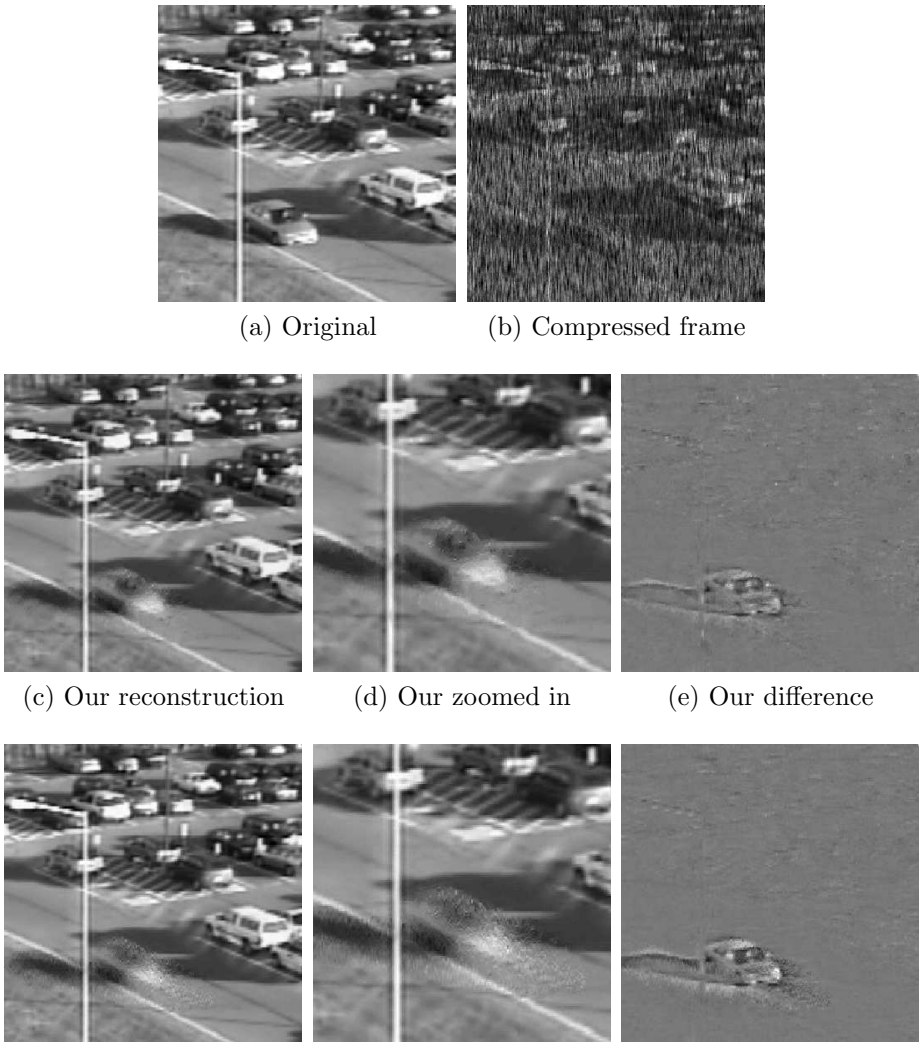


(j) TV2 difference



(k) TV3 difference

Figure 3.3: Figure illustration for Setting 3. The PSNR of this particular reconstruction with our model is 35.4635, compared with 29.7760 for TV2 and 32.4518 for TV3.



(a) Original (b) Compressed frame

(c) Our reconstruction (d) Our zoomed in (e) Our difference

(f) Temporal-only model (g) Temporal-only zoomed in (h) Temporal-only difference

Figure 3.4: The comparison between our model and the model with only temporal regularizer. The PSNR of this particular reconstruction for our model is 34.3988, and temporal-only regularized version of our model gives 32.2018.

conditions. In this form, the recovered videos is (in some sense) a piecewise constant interpolation between the two recovered endpoints. In Fig. 3.4, a motion blur artifact remains in the temporal-only model (g), while our model better localizes the temporal information (d). This also appears in the difference images as a region of error around the cars. In practice, we see that the spatial regularizer stabilizes each frame.

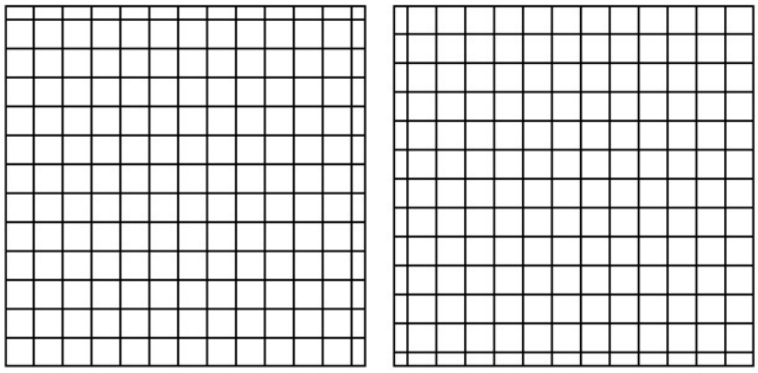


Figure 3.5: Two examples of the non-overlapping patch system. The difference in the patch edges ensures the improvement in the averaging result.

In real applications, the dimension of F is often quite large, hence directly solving (3.3) can take a long time. In order to meet the need of real-time reconstruction, a commonly used technique is to break the original problem into a bunch of subproblems and use parallel computing to solve each small problem simultaneously. One possible way of realizing that is to cut X into several non-overlapping patches $\{X^j\}$, and for each X^j recover the associated video data cube F^j by solving (3.3) with a reduced problem size. Then by putting all the cubes together we are able to obtain the original F. However, solely applying this idea often bears the grid artifact on patch edges.

In order to alleviate the side effect, we can build the non-overlapping patch system in different ways and then take the average of all the computed results. This idea was applied to image recovery in [42]. Figure 3.5 are two examples of the non-overlapping patch system.

To significantly remove the grid artifact, usually at least 3 different systems have to be built, and this on the other hand greatly increases the computational cost. Here we propose a new way to build the parallel system, whose construction process is illustrated in Figure 3.6. As before, the first step is to divide the original measurement matrix X into several non-overlapping patches. For each patch (e.g. the black square in Figure 3.6), we extend its edges by several pixels and construct

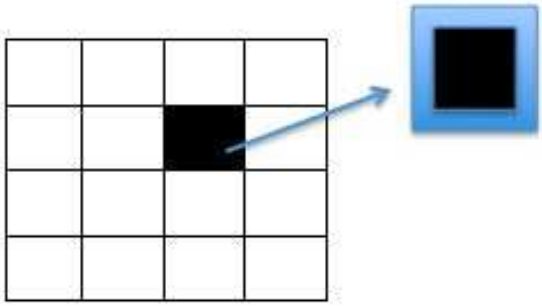


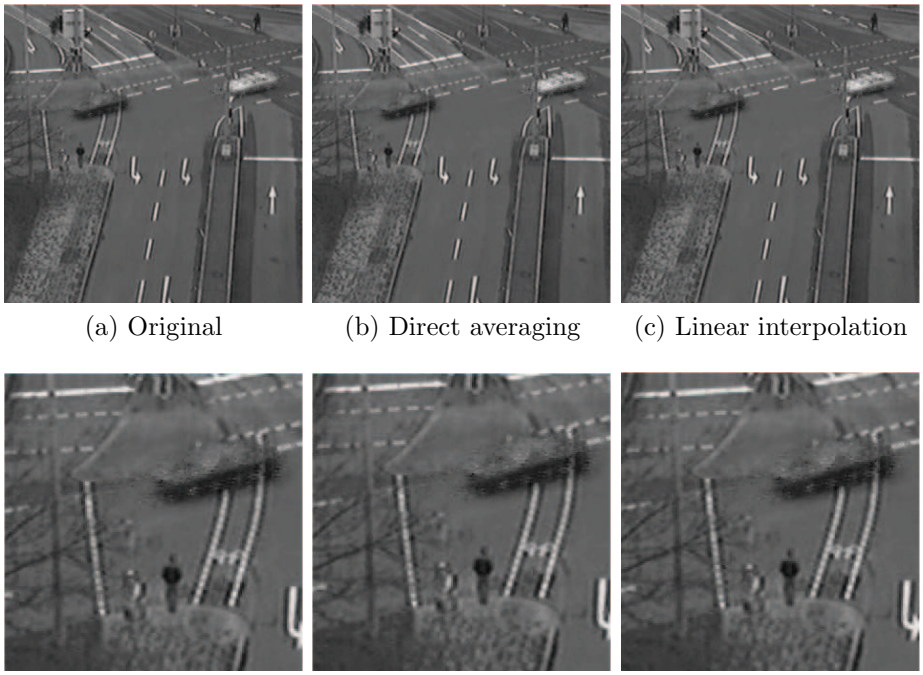
Figure 3.6: Illustration of our parallel system.

Table 3.2: Comparison of results from solving the original problem and the ones from solving the parallel system with different averaging functions

	The original problem	The parallel system with direct averaging	The parallel system with linear interpolation
Setting 1	34.5321	34.3030	34.4960
Setting 2	34.5321	34.3921	34.5094
Setting 3	37.1156	37.0263	37.0497
Setting 4	37.1112	36.5263	36.5862

a new patch with slightly larger size (the blue one). Through solving (3.3) on the enlarged patches, each reconstructed data cube now have some overlap with its neighbors, and we can either use direct averaging or linear interpolation on the overlapping areas to remove the grid artifact.

Table 3.2 displays the comparison of mean PSNRs between results from solving the original problem (3.3) and the ones from our patch system in different settings. In Setting 1 the original F has size $350 \times 350 \times 4$. For the parallel system, we use patch size 35×35 with 5 pixel overlap. i.e. the blue path in Figure 3.6 has size 45×45. In the second test, F is unchanged while the patch size (the black one) is increased to 50×50 with 5 pixel overlap. Figure 3.7 compares the second reconstructed frame from the three models in this senario. In setting 3, we fix the patch size and increase the temporal compression rate to 8. In the last test, X is the compressed result from 12 frames and the patch size is still 50×50. It is easily seen from the results that by utilizing the new patch system we can achieve



(a) Original (b) Direct averaging (c) Linear interpolation

(d) Original zoomed in (e) Direct averaging zoomed in (f) Linear interpolation zoomed in

Figure 3.7: Illustration of a reconstructed frame from Setting 2 in Table 3.2. From left to right: result from solving the original problem (PSNR 34.8539); result from the parallel system with direct averaging (PSNR 34.6945); result from the parallel system with linear interpolation (PSNR 34.8241). The grid artifact from our patch system is negligible.

comparable results with much less time and little loss in PSNRs.

3.4 Conclusion

Our model shows dramatic improvement using a more consistent regularizer. The main function of our temporal regularizer is to better capture both fast moving objects and temporal relationships. Numerical simulations show large improvements in both the PSNR and visual quality of the reconstructions using our method, regardless of the type of dynamics present in the video. In this work, the problem we are considering derives from VCS; however, the method presented here can easily be applied to other compression types.

Part II

Adaptive Outlier Pursuit

CHAPTER 4

Introduction

In many real world applications, such as signal and image processing, the acquired measurements are often corrupted by different kinds of noise during data acquisition and transmission. Therefore, developing algorithms that are robust to noise is another important topic besides the efficiency concern.

A widely studied noise type is the additive Gaussian noise, whose norm is often relatively small compared with the original data, hence most of the information in the data can be retained in this case. However, sometimes the noise can damage part of the data completely, leaving it with no useful information at all. In this scenario, using the damaged data for signal reconstruction is useless and can even worsen the performance of the reconstruction methods. Hence algorithms robust to outliers are strongly needed. In applications like image processing, several techniques have been proposed to deal with outliers, such as adaptive median filter(AMF) for salt-and-pepper impulse noise removal, while in many other applications there is no useful technique for effective outlier detection. Here we propose a technique named adaptive outlier pursuit (AOP) and apply it to robust 1-bit compressive sensing (CS) and exact matrix reconstruction.

Instead of preprocessing the data before applying standard algorithms, the AOP technique detects error location and recovers the signal in an alternating fashion. Here a binary variable Λ is introduced to label the all the measurements. Λ_i equals 1 if the i^{th} measurement is considered to be "correct", and 0 otherwise. In each iteration, only the "correct" measurements are used to reconstruct the

original data, then Λ is updated through minimizing the energy function with the newly recovered signal.

Chapter 5 focuses on robust 1-bit compressive sensing with AOP. The field of compressive sensing and techniques using sparsity have recently gained a lot of attention in various areas of research, such as image processing and machine learning. In the standard compressive sensing framework, the original signal is recovered by finding the sparsest solution to an underdetermined linear system. 1-bit CS was introduced and studied in 2008 by Boufounos and Baraniuk [38], which examines the efficient reconstruction of a sparse or compressible signal from its linear measurements followed by the most severe quantization: retaining only the sign information, i.e. 1 bit per measurement. It enjoys many advantages in practice. A lot of models and algorithms have been proposed to solve this problem. However, they all perform poorly when the measurements contain a lot of sign flips. With adaptive outlier pursuit, we are able to detect the sign flips and reconstruct the signals with very high accuracy. Numerical experiments show our algorithms outperform other algorithms greatly in all the displayed tests.

Chapter 6 covers the application of AOP on exact matrix completion. The problem of matrix completion aims at finding a low-rank matrix from incomplete samples of its entries. It enjoys wide applications in many areas, such as model reduction and the famous Netflix problem. This is a well studied problem for the clean data case. However when some of the given entries are contaminated by sparse random-valued noise, most of the current algorithms will not be able to reconstruct the original matrix. Similarly, with the introduction of AOP to the existing model, we are able to achieve a remarkable improvement in reducing the relative error and finding all the correct error locations compared with other algorithms.

CHAPTER 5

Robust 1-bit Compressive Sensing using Adaptive Outlier Pursuit

Abstract

In compressive sensing (CS), the goal is to recover signals at reduced sample rate compared to the classic Shannon-Nyquist rate. However, the classic CS theory assumes the measurements to be real-valued and have infinite bit precision. The quantization of CS measurements has been studied recently and it has been shown that accurate and stable signal acquisition is possible even when each measurement is quantized to only one single bit. There are many algorithms proposed for 1-bit compressive sensing and they work well when there is no noise in the measurements, e.g., there are no sign flips, while the performance is worsened when there are a lot of sign flips in the measurements. In this paper, we propose a robust method for recovering signals from 1-bit measurements using adaptive outlier pursuit. This method will detect the positions where sign flips happen and recover the signals using correct measurements. Numerical experiments show the accuracy of sign flips detection and high performance of signal recovery for our algorithms compared with other algorithms.

5.1 Introduction

The theory of compressive sensing (CS) enables reconstruction of sparse or compressible signals from a small number of linear measurements relative to the dimension of the signal space [44, 45, 46, 47, 48]. In this setting, we have

$$y = \Phi x, \tag{5.1}$$

where $x \in \mathbf{R}^N$ is the signal, $\Phi \in \mathbf{R}^{M \times N}$ with $M < N$ is an underdetermined measurement system, and $y \in \mathbf{R}^M$ is the set of linear measurements. It was demonstrated that K-sparse signals, i.e., $x \in \Sigma_K$ where $\Sigma_K := \{x \in \mathbf{R}^N : \|x\|_0 := |\text{supp}(x)| \leq K\}$, can be reconstructed exactly if Φ satisfies the restricted isometry property (RIP) [49]. It was also shown that random matrices can satisfy RIP with high probability if the entries are chosen according to independent and identically distributed (i.i.d.) Gaussian distribution.

Classic compressive sensing assumes that the measurements are real valued and have infinite bit precision. However, in practice, CS measurements must be quantized, i.e., each measurement has to be mapped from a real value (over a potentially infinite range) to a discrete value over some finite range, which will induce error on the measurements. The quantization of CS measurements has been studied recently and several new algorithms are proposed [50, 51, 52, 53, 54, 55].

Furthermore, for some real world problems, severe quantization may be inherent or preferred. For example, in analog-to-digital conversion (ADC), the acquisition of 1-bit measurements of an analog signal only requires a comparator to zero, which is an inexpensive and fast piece of hardware that is robust to amplification of the signal and other errors, as long as they preserve the signs of the measurements, see [38, 56]. In this chapter, we will focus on the CS problem with 1-bit quantization.

The 1-bit compressive sensing framework [38] is as follows. Measurements of a signal $x \in \mathbf{R}^N$ are computed via

$$y = A(x) := \text{sign}(\Phi x). \tag{5.2}$$

Therefore, the measurement operator $A(\cdot)$ is a mapping from $\mathbf{R}^N$ to the Boolean cube [1] $\mathcal{B}^M := \{-1, 1\}^M$. We have to recover signal $x \in \sum_K^* := \{x \in S^{N-1} : \|x\|_0 \leq K\}$ where $S^{N-1} := \{x \in \mathbf{R}^N : \|x\|_2 = 1\}$ is the unit hyper-sphere of dimension N. Since the scale of the signal is lost during the quantization process, we can restrict the sparse signals to be on the unit hyper-sphere. Jacques et al. provided two flavors of results for the 1-bit CS framework [57]: 1) a lower bound is provided on the best achievable performance of this 1-bit CS framework, and if the elements of Φ are drawn randomly from i.i.d. Gaussian distribution or its rows are drawn uniformly from the unit sphere, then the solution will have bounded error on the order of the optimal lower bound. 2) A condition on the mapping A, binary ϵ-stable embedding (BϵSE), that enables stable reconstruction is given to characterize the reconstruction performance even when some of the measurement signs have changed (e.g., due to noise in the measurements).

Since this problem was introduced and studied by Boufounos and Baraniuk in 2008 [38], it has been studied by many people and several algorithms have been developed [38, 57, 58, 59, 60, 61, 62]. Binary iterative hard thresholding (BIHT) [57] is shown to perform better than other algorithms such as matching sign pursuit (MSP) [58] and restricted-step shrinkage (RSS) [60] in terms of decreasing reconstruction error as well as keeping signal consistency, see [57] for more details. The experiment in [57] shows that the one-sided ℓ_1 objective (BIHT) performs better when there are only a few errors, and the one-sided ℓ_2 objective (BIHT-ℓ_2) performs better when there are significantly more errors, which implies that BIHT-ℓ_2

[1] Generally, the M-dimensional Boolean cube is defined as $\{0, 1\}^M$. Without loss of generality, we use $\{-1, 1\}^M$ instead.

is useful when the measurements contain significant noise that might cause a large number of sign flips.

In practice, there will always be noise in the measurements during data acquisition and transmission, therefore, a robust algorithm for 1-bit compressive sensing when the measurements contain sing flips. One possible way to build this robust algorithm is to introduce an outlier detection technique.

There are many applications where accurate detection of outliers is needed. For example, when an image is corrupted by random-valued impulse noise, the corrupted pixels are useless in image denoising. There are some methods (e.g., adaptive center-weighted median filter (ACWMF) [63]) for detecting the damaged pixels. But these methods will miss quit a lot of real noise and false-hit some noise-free pixels when the noise level is high. In [64], we proposed a method to adaptively detect the noise pixels and restore the image with ℓ_0 minimization. Instead of detecting the damaged pixels before recovering the image, we iteratively restore the image and detect the damaged pixels. This idea works really well for impulse noise removal. In this 1-bit compressive sensing framework, when there is a sign flip in one measurement, this measurement will worsen the reconstruction performance. If we can detect all the measurements with sign flips, then we can change the signs for these measurements and improve the reconstruction performance a lot. However, it is much more difficult than detecting impulse noise and there is no method for detecting sign flips, but we can still utilize the idea in [64] to adaptively find the sign flips. In this chapter, we will introduce a method for robust 1-bit compressive sensing which can detect the sign flips and reconstruct the signals with very high accuracy even when there are a large number of sign flips.

This chapter is organized as follows. We will introduce several algorithms for reconstructing the signal and detecting the sign flips in section 5.2. Section 5.3 studies the case when the noise information is not given. The performance of these

algorithms is illustrated in section 5.4 with comparison to BIHT and BIHT-ℓ_2. We will end this work by a short conclusion.

5.2 Model and Algorithm

Binary iterative hard thresholding (BIHT or BIHT-ℓ_2) in [57] is the algorithm for solving

$$\begin{aligned} \min_{x} \quad & \sum_{i=1}^{M} \phi(y_i, (\Phi x)_i) \\ \text{subject to:} \quad & \|x\|_2 = 1, \quad \|x\|_0 \leq K, \end{aligned} \tag{5.3}$$

where ϕ is the one-sided ℓ_1 (or ℓ_2) objective:

$$\phi(x, y) = \begin{cases} 0, & \text{if } x \cdot y > 0, \\ |x \cdot y| \text{ (or } |x \cdot y|^2/2), & \text{otherwise.} \end{cases} \tag{5.4}$$

The high performance of BIHT is demonstrated when all the measurements are noise-free. However when there are a lot of sign flips, the performance of BIHT and BIHT-ℓ_2 is worsened by the noisy measurements. There is no method to detect the sign flips in the measurements, but adaptively finding the sign flips and reconstructing the signals can be combined together as in [64] to obtain better performance.

Let us assume firstly that the noise level (the ratio of the number of sign flips over the number of measurements for 1-bit compressive sensing) is provided. Based on this information, we can choose a proper integer L such that at most L elements of the total measurements are wrongly detected (having sign flips). For measurements $y \in \{-1, 1\}^M$, $\Lambda \in \mathbf{R}^M$ is a binary vector denoting the "correct"

data:

$$\Lambda_i = \begin{cases} 1, & \text{if } y_i \text{ is "correct"}, \\ 0, & \text{otherwise.} \end{cases} \tag{5.5}$$

According to the assumption, we have $\sum_{i=1}^{M}(1-\Lambda_i) \leq L$.

Introducing Λ into the old problem solved by BIHT, we have the following new problem with unknown variable x and Λ:

$$\begin{aligned} \min_{x,\Lambda} \quad & \sum_{i=1}^{M} \Lambda_i \phi(y_i, (\Phi x)_i) \\ \text{s.t.} \quad & \sum_{i=1}^{M}(1-\Lambda_i) \leq L, \\ & \Lambda_i \in \{0,1\} \quad i = 1, 2, \cdots, M, \\ & \|x\|_2 = 1, \quad \|x\|_0 \leq K. \end{aligned} \tag{5.6}$$

The above model can also be interpreted in the following way. Let us consider the noisy measurement y as the sign of Φx with additive unknown noise n, i.e. $y = \text{sign}(\Phi x + n)$. Though the binary measurement is robust to noise as long as the sign does not change, there exist some n_i's such that the corresponding measurements change. In our problem, only a few measurements are corrupted, and only these corresponding n_i's are important. Therefore, n can be considered as sparse noise with nonzero entries at these locations, and we have to recover the signal x from sparse corrupted measurements [65, 66], even when the measurements is acquired by taking the sign of $\Phi x + n$. This equivalent problem is

$$\begin{aligned} \min_{x,n} \quad & \sum_{i=1}^{M} \phi(y_i, (\Phi x)_i + n_i) \\ \text{s.t.} \quad & \|n\|_0 \leq L, \\ & \|x\|_2 = 1, \quad \|x\|_0 \leq K. \end{aligned} \tag{5.7}$$

The equivalence is described in the appendix.

The problem defined in (5.6) is non-convex and has both continuous and discrete variables. It is difficult to solve it together, thus we use alternating minimization method, which separates the energy minimization over x and Λ into two steps:

- Fix Λ and solve for x:

$$\begin{aligned} \min_{x} \quad & \sum_{i=1}^{M} \Lambda_i \phi(y_i, (\Phi x)_i) \\ \text{s.t.} \quad & \|x\|_2 = 1, \quad \|x\|_0 \leq K. \end{aligned} \tag{5.8}$$

  This is the same as (5.3) with revised Φ and y. We only need to use the ith rows of Φ and y where $\Lambda_i = 1$.

- Fix x and update Λ:

$$\begin{aligned} \min_{\Lambda} \quad & \sum_{i=1}^{M} \Lambda_i \phi(y_i, (\Phi x)_i) \\ \text{s.t.} \quad & \sum_{i=1}^{M} (1 - \Lambda_i) \leq L, \\ & \Lambda_i \in \{0, 1\} \quad i = 1, 2, \cdots, M. \end{aligned} \tag{5.9}$$

  This problem is to choose $M - L$ elements with least sum from M elements $\{\phi(y_i, (\Phi x)_i)\}_{i=1}^{M}$. Given an x estimated from (5.8), we can update Λ in one step:

$$\Lambda_i = \begin{cases} 0, & \text{if } \phi(y_i, (\Phi x)_i) \geq \tau, \\ 1, & \text{otherwise.} \end{cases} \tag{5.10}$$

  where τ is the L^{th} largest term of $\{\phi(y_i, (\Phi x)_i)\}_{i=1}^{M}$. If the L^{th} and $(L+1)^{th}$ largest terms are equal, then we can choose any Λ such that $\sum_{i=1}^{M} \Lambda_i = M - L$ and

$$\min_{i, \Lambda_i = 0} \phi(y_i, (\Phi x)_i) \geq \max_{i, \Lambda_i = 1} \phi(y_i, (\Phi x)_i).$$

Since for each step, the updated Λ identifies the outliers, this method is named as adaptive outlier pursuit (AOP). When $L = 0$, this is exactly the BIHT proposed in [57]. Our algorithm is as follows:

Algorithm 2 AOP

Input: $\Phi \in \mathbf{R}^{M\times N}$, $y \in \{-1, 1\}^M$, $K > 0$, $L \geq 0$, $\alpha > 0$, Miter > 0
Initialization: $x^0 = \Phi^T y/\|\Phi^T y\|$, $k = 0$, $\Lambda = \mathbf{1} \in \mathbf{R}^M$, Loc $= 1 : M$, tol $= \inf$, TOL $= \inf$.
while $k \leq$ Miter and $L \leq$ tol **do**
  Compute $\beta^{k+1} = x^k + \alpha\Phi(\text{Loc}, :)^T(y(\text{Loc}) - \text{sign}(\Phi(\text{Loc}, :)x^k))$.
  Update $x^{k+1} = \eta_K(\beta^{k+1})$,
  Set tol $= \|y - A(x^{k+1})\|_0$.
  if tol $\leq$TOL **then**,
    Compute Λ with (5.10).
    Update Loc to be the location of 1-entries of Λ.
    Set TOL = tol.
  end if
  $k = k + 1$.
end while
return $x^k/\|x^k\|$.

$\eta_K(v)$ computes the best K-term approximation of v by thresholding. Since $y_i \in \{-1,\ 1\}$, once we find the locations of the errors, instead of deleting these data, we can also "correct" them by flipping their signs. Hence x can also be updated with Φ and this new measurements. This algorithm with changing signs is called AOP with flips.

Remark: Similar to BIHT-ℓ_2, we can also choose the one-sided ℓ_2 objective instead of ℓ_1 objective and obtain two other algorithms.

5.3 The Case with L Unknown

In previous section, we assume that L, the number of corrupted measurements, is known in advance. However in real world applications there are cases when no pre-knowledge about the noise is given. If L is chosen smaller or larger than the true

value, the performance of these algorithms will get worse. As shown numerically in section 5.4, when L is less than the true value, even if the L detections are completely correct, some sign flips still stay in the measurements. On the other hand, some correct measurements will be lost if L is too large, and the problem will have more solutions if the number of total measurements is not large enough, which will affect the accuracy of the algorithm. Therefore, in this scenario we have to apply an L detection skill to find an L which is not far from the true value.

When no noise information is given, the following procedure can be applied to predict L. The first-phase preparation is to do extensive experiments on simulated data with known L and record the Hamming distances between $A(x)$ and noisy y of BIHT-ℓ_2 and AOP. Here we can simply use the results in our first experiment in section 5.4. The average of the results describes nicely the behavior of these two algorithms at different noise levels. Hence a formula can be derived to predict the Hamming distance of AOP based on the results obtained by BIHT-ℓ_2. This could be a fair initial guess for the noise level, and we can derive an L based on the result, labeled as L_0. Then we calculate $L_t = \|A(x) - y\|_0$ using the result x gained by AOP with L_0 as the input for L. If L_t is greater than L_0, which means that L_0 is too small while L_t is too large, we set L_t as the upper bound $L_{\max}$ and L_0 as the lower bound $L_{\min}$. Otherwise, if L_t is smaller than or equal to L_0, which means L_0 may be too large, we use μL_0 $(0 < \mu < 1)$ as the new L_0 to look for new L_t. We will keep doing this until L_t is greater than L_0. Then the previous L_0 is defined as the upper bound $L_{\max}$ and the new L_0 is defined as the lower bound $L_{\min}$. This is just one method for finding lower and upper bounds for L, and there are certainly other possible ways to decide the bounds. Then we use the bisection method to find a better L. The mean of $L_{\max}$ and $L_{\min}$ (L_{mean}) is then used as input to derive L_t with AOP. If L_t is greater than L_{mean}, we update $L_{\min}$ with L_{mean}. Otherwise, L_{mean} is set as $L_{\max}$. This bisection method is applied to

update these two bounds until $L_{\max} - L_{\min} \leq 1$. The final $L_{\min}$ is our input L.

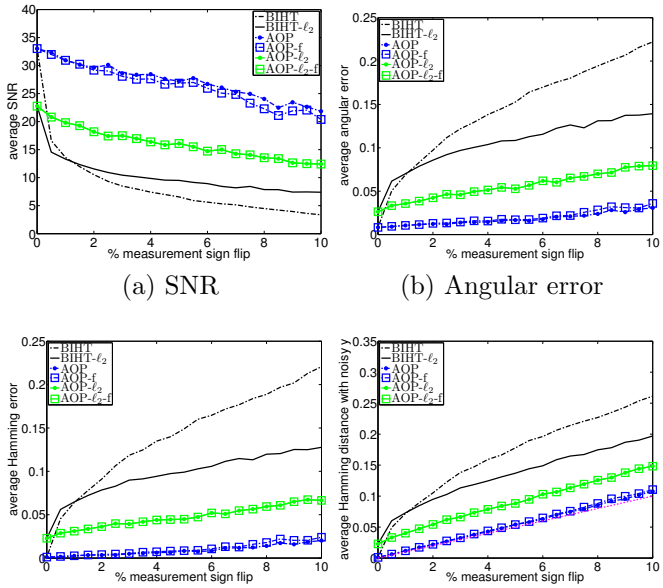
5.4 Numerical Results

In this section we use several numerical experiments to demonstrate the effectiveness of AOP algorithms. Here AOP is implemented in the following four ways: 1) AOP with one-sided ℓ_1 objective (AOP); 2) AOP with flips and one-sided ℓ_1 objective (AOP-f); 3) AOP with one-sided ℓ_2 objective (AOP-ℓ_2); 4) AOP with flips and one-sided ℓ_2 objective (AOP-ℓ_2-f). The four algorithms, together with BIHT and BIHT-ℓ_2, are studied and compared in the following experiments.

The setup for our experiments is as follows. We first generate a matrix $\Phi \in \mathbf{R}^{M\times N}$ whose elements follow i.i.d. Gaussian distribution. Then we generate the original K-sparse signal $x^* \in \mathbf{R}^N$. Its non-zero entries are drawn from standard Gaussian distribution and then normalized to have norm 1. $y^* \in \{-1, 1\}^M$ is computed by $A(x^*)$.

5.4.1 Noise Levels Test

In our first experiment, we set $M = N = 1000$, $K = 10$, and examine the performance of these algorithms on data with different noise levels. Here in each test, we choose a few measurements at random and flip their signs. The noise level is between 0% and 10% and we assume it is known in advance. For each level, we perform 100 trials and record the average signal-to-noise ratio (SNR), average reconstruction angular error for each reconstructed signal x with respect to x^*, average Hamming error between $A(x)$ and $A(x^*)$ and average Hamming distance between $A(x)$ and the noisy measurements y. Here SNR is denoted by $10\log_{10}(\|x\|^2/\|x - x^*\|^2)$, angular error is defined as $\arccos\langle x, x^*\rangle/\pi$, Hamming error stands for $\|A(x) - A(x^*)\|_0/M$ and the Hamming distance between $A(x)$ and y, defined as $\|A(x) - y\|_0/M$, is used to measure the difference between $A(x)$



(a) SNR (b) Angular error

(c) Hamming error between $A(x)$ and $A(x^*)$ (d) Hamming distance between $A(x)$ and y

Figure 5.1: Algorithm comparison on corrupted data with different noise levels. (a) average SNR vs noise level, (b) average angular error vs noise level, (c) average Hamming error between $A(x)$ and $A(x^*)$ vs noise level, (d) average Hamming distance between $A(x)$ and noisy measurements y vs noise level. AOP proves to be more robust to measurement sign flips compared with BIHT.

and the noisy measurements y. The results are depicted in Figure 5.1. The plots demonstrate that in these comparisons four AOP algorithms outperform BIHT and BIHT-ℓ_2 for all noise levels, significantly so when more than 2% of the measurements are corrupted. Compared with BIHT, BIHT-ℓ_2 tends to give worse results when there are only a few sign flips in y and better results if we have high noise level. This has been shown and studied in [57]. Of all the AOP series, AOP and AOP-f give better results compared with AOP-ℓ_2 and AOP-ℓ_2-f. We can also see that there is a lot of overlap between the results obtained by AOP and the ones acquired by AOP with flips, especially when one-sided ℓ_2 objective is used, the results are almost the same. Figure 5.1(d) compares the average Hamming distances between $A(x)$ and the noisy measurements y for all algorithms. If the sign flips can be found correctly, then the Hamming distance between $A(x)$ and y should be equal to noise level. The result shows that average Hamming distances

for AOP and AOP-f are slightly above the noise levels, which means that AOP with one-sided ℓ_1 objective performs better in consistency than other algorithms in noisy cases.

In order to show that our algorithms can find the positions of sign flips with high accuracy, we measure the probabilities of correct detections of sign flips in the noisy measurements for different noise levels from 0.5% to 10% in Figure 5.2 ($M = N = 1000$, $K = 10$). The exact number of sign flips is used as L in the algorithms and we compare the exact locations of sign flips in measurements y with those detected from the algorithms for all 100 trials, then the average probabilities of correct detections are shown for different algorithms at different noise levels. From this figure, we can see that all four algorithms have high accuracy in detecting the sign flips. When the noise level is low ($\leq$4%), the accuracy of AOP and AOP-f can be as high as 95%, even when the noise level is high (e.g., 10%), the accuracy of AOP and AOP-f is still above 90%. Comparing to algorithms with one-sided ℓ_1 objective, algorithms with one-sided ℓ_2 objective have lower accuracy. The accuracy for AOP-ℓ_2 and AOP-ℓ_2-f is around 80%.

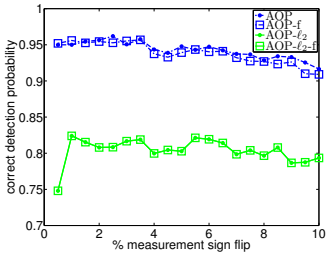
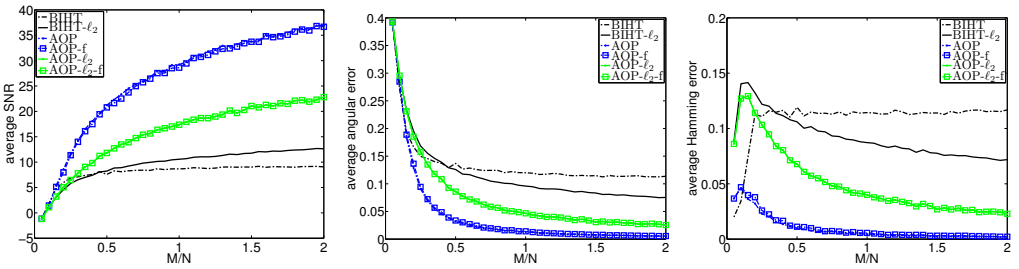


Figure 5.2: The probabilities of correct detections of sign flips for different noise levels ranging from 0.5% to 10%. AOP and AOP-f have very high accuracy (great than 90%) in detecting the sign flips, while AOP-ℓ_2 and AOP-ℓ_2-f have relatively lower accuracy (around 80%).

5.4.2 M/N Test

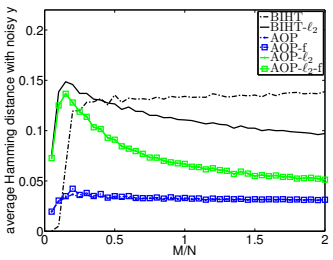
In the second experiment, $N = 1000$, $K = 10$ and the noise level 3% are fixed, and we change M/N within the range $(0, 2]$. 40 different M/N are considered and we perform 300 tests for each value. The results are displayed in five different ways:



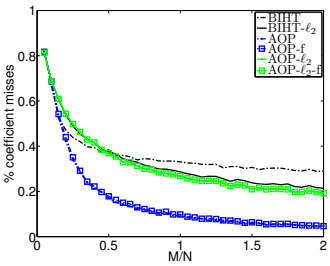
(a) SNR

(b) Angular error

(c) Hamming distance between $A(x)$ and $A(x^*)$



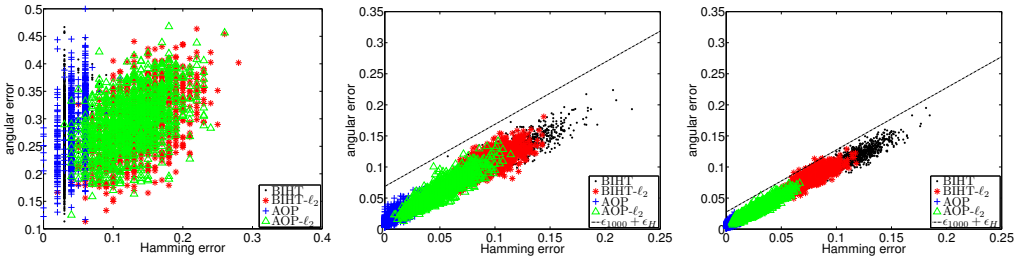
(d) Hamming error between $A(x)$ and y



(e) % coefficient misses

Figure 5.3: Algorithm comparison on corrupted data with different M/N. (a) average SNR vs M/N, (b) average angular error vs M/N, (c) average Hamming error between $A(x)$ and $A(x^*)$ vs M/N, (d) average Hamming distance between $A(x)$ and y vs M/N, (e) average percentage of coefficient misses vs M/N. AOP yields a remarkable improvement in reducing the Hamming and angular error and achieving higher SNR.

the average SNR, average angular error, average Hamming error between $A(x)$ and $A(x^*)$, average Hamming distance between $A(x)$ and y and average percentage of coefficient "misses". Here "misses" stands for the coefficients where $x_i^* \neq 0$ while $x_i = 0$. According to Figure 5.3, although all the algorithms show the same trend as M/N increases, AOP and AOP-f always obtain a much smaller angular error (higher SNR) than BIHT and BIHT-ℓ_2. There are also fewer coefficient misses in the results acquired by AOP series. Furthermore, we see that even when 3% of the measurements are corrupted, AOP can still recover a signal with SNR greater than 20 using less than 0.5 bits per coefficient of x^*. In Hamming error comparison, AOP and AOP-f beat other algorithms significantly when $M/N > 0.15$. Moreover, we see that the average Hamming error of AOP and AOP-f is extremely close to zero when $M/N > 0.5$. When $M/N < 0.15$, the seemingly failure of AOP and



(a) M=0.1N (b) M=0.7N (c) M=1.5N

Figure 5.4: Hamming error vs angular error with different M. AOP gives the most consistent results for $M = 0.7N$ and $M = 1.5N$. In these two cases we can see a linear relationship $\epsilon_{\text{sim}} \approx C + \epsilon_H$ between the average angular error ϵ_{sim} and average Hamming error ϵ_H, where C is constant. For really small M ($M = 0.1N$) BIHT returns almost the same results as AOP since AOP may fail to find the exact sign flips in the noisy measurements. The dashed line $\epsilon_{1000} + \epsilon_H$ is a upper bound for 1000 trials.

AOP-f compared with BIHT is due to the fact that there are usually more than one solution to (5.6) for really small M, and with high probability our method will return one solution with L sign flips, which may not be the actual ones. Hence we may not be able to detect the actual errors in the measurements.

We also try to explore the relationship between the Hamming error between $A(x)$ and $A(x^*)$ and the reconstruction angular error. With $N = 1000$, $K = 10$ and the noise level 3% fixed, we plot the Hamming error vs angular error for three different M in Figure 5.4. Since AOP and AOP with flips tend to return almost the same results if we use the same objective (one-sided ℓ_1 or one-sided ℓ_2) for x update, we only compare the results acquired by BIHT, BIHT-ℓ_2, AOP and AOP-ℓ_2. We can see clearly that almost all the blue (+) points stay in the lower left part of the graph for $M = 0.7N$ and $M = 1.5N$, which proves that AOP gives more consistent results compared with other three algorithms. For these two M, the average angular error is close to a linear function of average Hamming error, which is predicted by BϵSE property in [57]. We also plot an empirical upper bound for AOP of $\epsilon_{1000}+\epsilon_H$ defined in [57], where ϵ_{1000} is the largest angular error of AOP and ϵ_H is the Hamming distance. For especially "under-sampled" case like $M = 0.1N$, none of these algorithms is able to return consistent reconstructions,

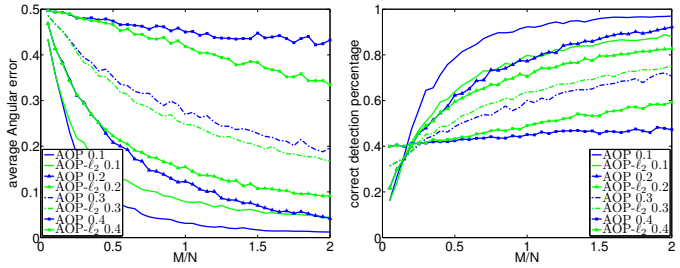
as we can see the points scatter almost randomly over the domain. In this case the results obtained by BIHT stay really close to those gained by AOP. As mentioned above, this is because AOP may not be able to detect the exact sign flips in the noisy measurements when M is too small.

5.4.3 High Noise Levels

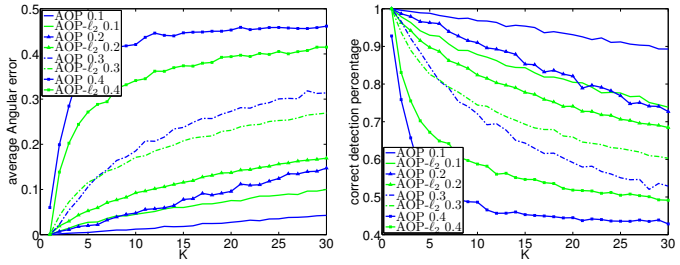
In this subsection, we study the performance of AOP and AOP-ℓ_2 when a large number of measurements are corrupted. Two settings are considered. In the first experiment, we fix $N = 1000$, $K = 10$, and change the M/N ratio between 0.05 and 2. Four different noise levels are considered from 0.1 to 0.4 and we record the average angular error and correct detection probability from 100 tests. In the second setting, we fix $M = 2000$, $N = 1000$ and change K from 1 to 30. Still, four noise levels are considered and the mean results from 100 tests are recorded. From Figure 5.5 (a) and (b), we can see similar trend for the behavior of angular error and correct detection probability as we have discovered in Figure 5.3. According to (c), (d), for all the noise levels the performance of these two algorithms tends to get worse as K increases. We also have another interesting discovery that when the noise level is greater than 0.2, AOP-ℓ_2 turns out to be a better choice than AOP. This is because when the noise level is extremely high, even with outlier detection technique, lots of sign flips remain in the recovered measurements, and this new "noise level" is still relatively high. According to [57], BIHT-ℓ_2 outperforms BIHT when the measurements contain lots of sign flips. Therefore, when the noise level is high enough, AOP-ℓ_2 is considered as a better choice compared with AOP.

5.4.4 L Mismatch

In Figure 5.6, we analyze the influence of incorrect selection of L on AOP. Here we choose $M = N = 1000$, $K = 10$, noise level 5%, and change the input value



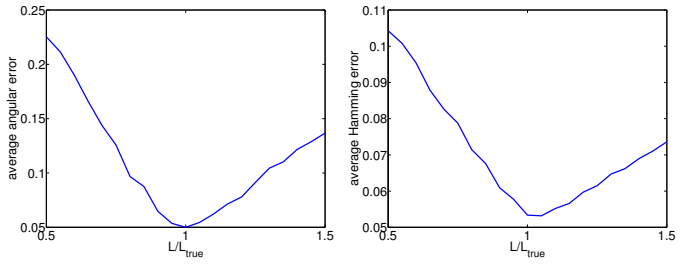
(a) Angular Error vs M/N (b) Correct Detection vs M/N



(c) Angular Error vs K (d) Correct Detection vs K

Figure 5.5: AOP and AOP-ℓ_2 performance under different noise levels. (a) average angular error vs M/N with different noise levels, (b) correct detection percentage vs M/N with different noise levels, (c) average angular error vs K with different noise levels, (d) correct detection percentage vs K with different noise levels. The performance gets better when we increase M/N or decrease K.

from $0.5L$ to $1.5L$. 100 tests are conducted and the mean results are recorded. It is easily seen that the error will become larger when the input L digresses from its true value. According to this plot, we know that in order to obtain good performance for our method, we should choose a proper L as input.

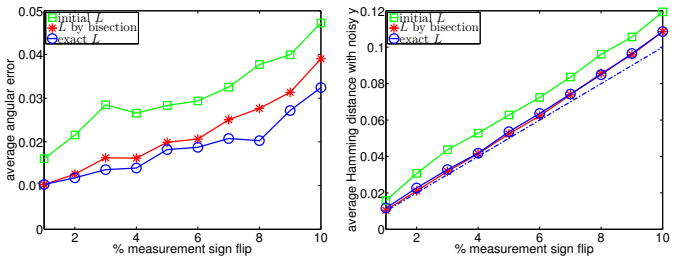


(a) Angular Error (b) Hamming Error

Figure 5.6: AOP performance with different L. L has to stay close to its true value in order to get good performance

5.4.5 Unknown L

To show that our method works even when L is not given, we use the method described in Section III to find an approximation of L, and compare the results of AOP with different L. Here $M = N = 1000$, $K = 10$ are fixed, and 10 different noise levels (from 1% to 10%) are tested. Three inputs for L: the initial L_0 predicted from the result of BIHT-ℓ_2, L obtained from bisection method, exact L, are used in AOP to obtain the results. The following figure 5.7 is depicted with the average results from 100 trials. Even with the initial L_0, the results are comparable to those with exact L, and bisection method can provide a better approximation for L with longer time for predicting L.



(a) Angular error (b) Hamming distance with noisy y

Figure 5.7: Comparison of results by different L at different noise levels from 1% to 10%. (a) average angular error vs noise level, (b) average Hamming distance between $A(x)$ and noisy y vs noise level. By choosing appropriate L as the input, we can still obtain the results comparable to those with exact L.

5.5 Conclusion

In this paper, we propose a method based on adaptive outlier pursuit for robust 1-bit compressive sensing. By iteratively detecting the sign flips in measurements and recovering the signals from "correct" measurements, this method can obtain better results in both finding the noisy measurements and recovering the signals, even when there are a lot of sign flips in the measurements. Four algorithms

(AOP, AOP-f, AOP-ℓ_2 and AOP-ℓ_2-f) are given based on this method, and the performances of these four algorithms are shown in the numerical experiments. The algorithms based on one-sided ℓ_1 objective (AOP and AOP-f) have better performance compared to the other two algorithm (AOP-ℓ_2 and AOP-ℓ_2-f), which are based on one-sided ℓ_2 objective when the noise level is not high (less than 20%), when the noise level is extremely high, AOP-ℓ_2 is a better choice compared with AOP. In addition, we proposed a simple method to find a candidate for the number of sign flips L when L is unknown and the numerical experiments show that the performance of AOP with this inexact input L is comparable with that of exact L.

Appendix

In this appendix, we show the equivalence of problem (5.6) and (5.7). If (x, n) satisfies the constraints of problem (5.7), we can define

$$\Lambda_i = \begin{cases} 1, & \text{if } n_i = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{5.11}$$

then we have $\phi(y_i, (\Phi x)_i + n_i) = 0$ if $\Lambda_i = 0$, since we can always find n_i such that $\phi(y_i, (\Phi x)_i + n_i) = 0$ for fixed x. If $\Lambda_i = 1$, we have $n_i = 0$, thus $\phi(y_i, (\Phi x)_i + n_i) = \phi(y_i, (\Phi x)_i)$. Therefore, problem (5.7) is equivalent to

$$\begin{aligned} \min_{x,n} \quad & \sum_{i=1}^{M} \Lambda_i \phi(y_i, (\Phi x)_i) \\ \text{s.t.} \quad & \|n\|_0 \le L, \\ & \|x\|_2 = 1, \quad \|x\|_0 \le K. \end{aligned} \tag{5.12}$$

From the relation of Λ and n in (5.11), we know the constraint $\|n\|_0 \le L$ in the above problem can be replaced with the constraints on Λ defined in (5.6).

Therefore, problem (5.6) and (5.7) are equivalent.

CHAPTER 6

Exact Low-Rank Matrix Completion from Sparsely Corrupted Entries via Adaptive Outlier Pursuit

This chapter is reprinted from Springer and Journal of Scientific Computing, volume 56, Ming Yan, Yi Yang and Stanley Osher, "Exact low-rank matrix completion from sparsely corrupted entries via adaptive outlier pursuit", pages 433-449, 2013, with kind permission from Springer Science and Business Media [39].

Abstract

Recovering a low-rank matrix from some of its linear measurements is a popular problem in many areas of science and engineering. One special case of it is the matrix completion problem, where we need to reconstruct a low-rank matrix from incomplete samples of its entries. A lot of efficient algorithms have been proposed to solve this problem and they perform well when Gaussian noise with a small variance is added to the given data. But they can not deal with the sparse random-valued noise in the measurements. In this paper, we propose a robust method for recovering the low-rank matrix with adaptive outlier pursuit when part of the measurements are damaged by outliers. This method will detect the positions where the data is completely ruined and recover the matrix using correct measurements. Numerical experiments show the accuracy of noise detection and high performance of matrix completion for our algorithms compared with other

algorithms.

6.1 Introduction

Nowadays, a lot of real world models can be categorized as matrix completion (MC) problems, such as video denoising [67], data mining and pattern recognitions [68], model reduction [69], low-dimensional embedding [70] etc. The general form of the MC problem is:

$$\underset{X\in\mathbf{R}^{m\times n}}{\text{minimize}} \quad \text{rank}(X), \text{ s.t. } X_{i,j} = M_{i,j} \ \forall (i,j) \in \Omega, \tag{6.1}$$

where we are given some entries of a matrix X with index set Ω and we want to recover it with its rank as low as possible. $\text{rank}(X)$ is defined as the number of nonzero singular values of X. However, solving (6.1) is often numerically expensive. Hence people tend to consider its relaxation:

$$\underset{X\in\mathbf{R}^{m\times n}}{\text{minimize}} \quad \|X\|_*, \text{ s.t. } X_{i,j} = M_{i,j} \ \forall (i,j) \in \Omega. \tag{6.2}$$

Here $\|X\|_*$ stands for the nuclear norm of X, which is the L_1 norm of the singular values $\sigma_i(X)$, i.e. $\|X\|_* = \sum_{i=1}^{r} \sigma_i(X)$ where $r = \text{rank}(X)$. It has been shown in [71, 72, 73] that, under certain reasonable conditions, (6.2) and (6.1) share the same solution. [73] also did further study about the recovery for general linear operator $\mathcal{A}: \ \mathbf{R}^{m\times n} \to \mathbf{R}^p$.

$$\underset{X\in\mathbf{R}^{m\times n}}{\text{minimize}} \quad \|X\|_*, \text{ s.t. } \mathcal{A}(X) = y. \tag{6.3}$$

Different types of algorithms have been proposed to solve (6.2), such as linearized Bregman method [74], fixed point and Bregman iterative methods [75] and accelerated proximal gradient algorithm [76]. [76] solved an unconstrained

version of (6.2):

$$\underset{X\in\mathbf{R}^{m\times n}}{\text{minimize}} \quad \mu\|X\|_* + \frac{1}{2}\|\mathcal{P}_\Omega(X) - \mathcal{P}_\Omega(M)\|_F^2. \tag{6.4}$$

Here μ is a properly tuned parameter. $\mathcal{P}_\Omega$ stands for the projection onto the subspace of matrices with nonzeros restricted to the index subset Ω, and $\|\cdot\|_F$, the Frobenius norm, is defined as

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|A_{i,j}|^2}. \tag{6.5}$$

for any matrix $A = (A_{i,j})_{m\times n}$. Most of the existing MC algorithms require singular value decomposition (SVD) in each iteration, which is the main time cost in these algorithms. In order to get rid of SVD and accelerate the algorithm, the authors in [77] proposed a new method LMaFit (low-rank matrix fitting) which solves a slightly modified version of (6.2):

$$\underset{U,W,Z}{\text{minimize}} \quad \|UW - Z\|_F^2 \text{ s.t. } \; Z_{i,j} = M_{i,j}, \; \forall (i,j) \in \Omega. \tag{6.6}$$

Here $U \in \mathbf{R}^{m\times k}$, $W \in \mathbf{R}^{k\times n}$, and $Z \in \mathbf{R}^{m\times n}$, where k is a predicted rank bound. With an appropriate k, it could give us the same result as (6.2). U, W, Z can be updated in an alternating fashion. Following the idea of nonlinear successive over relaxation (SOR) technique, [77] used weighted average between this update and the previous iterate and achieved a faster convergence. Recently, people have optimized this model and derived other efficient algorithms, such as RTRMC (Riemannian trust-region for matrix completion) [78]. Other approaches include [79, 80, 81]. We refer the readers to these references for more details.

In practice, there will always be noise in the measurements during data acquisition, therefore, a robust method for solving MC problem is strongly needed. Almost all the existing algorithms can deal with additive Gaussian noise with a

relatively small variance, but they can not perform stably when the given data is corrupted by outliers [82], another type of noise which often appears in application. For example, the problem of anticipating people's preference is gaining more and more attention nowadays. We are often asked to rate various kinds of products, such as movies, books, games, or even jokes. This problem is to use incomplete rankings provided by users on some of the products to predict their preference on other unrated products. It is typically treated as a low-rank MC problem. However, as the data collection process often lacks control and sometimes a few people may not be willing to provide their true opinions, the acquired data may contain some outliers. Therefore, applying the regular MC algorithm on this corrupted data may not lead to satisfactory result.

In order to deal with this case, we propose a method using adaptive outlier pursuit (AOP) which adaptively detects the damaged data location with high accuracy. Without the effect of wrong measurements, the reconstruction performance can be improved a lot. This AOP technique has been applied to image denoising in [64] and robust 1-bit compressive sensing in [43] and performed remarkably well. Combining this technique with the existing MC algorithm, our method is able to reconstruct the exact matrix even from sparsely corrupted entries. Here we also want to mention that besides AOP, people have proposed several methods to deal with outliers in the given data, such as [83, 84, 85].

This paper is organized as follows. We will describe our algorithm together with other popular methods for robust matrix completion in section 6.2. Section 6.3 focuses on the connection between our problem and another robust low-rank matrix approximation model. We also provide extensive study in section 6.4 on the case when we only have limited information about the noise. The performance of the algorithms is shown in section 6.5. We will end this work by a short conclusion.

6.2 Algorithm Description

From now on, let us assume that the rank r is given in advance, i.e. the rank estimate k is set to be r. According to massive experiments, the model (6.6) proves to be a quite efficient way to deal with MC problems when some information about the rank is known in advance. One drawback about this formulation is that the solution (U, W) is not unique. As a matter of fact, for any $r \times r$ invertible matrix A, $(UA, A^{-1}W)$ is another pair of solution. Many people have devoted to improve this model, such as [86, 87, 88, 79, 89, 90, 91].

The author in [78] combined the ideas in these work and proposed the following model and the associated algorithm RTRMC:

$$\underset{\mathcal{U}\in\mathcal{G}(m,r),W\in\mathbf{R}^{r\times n}}{\text{minimize}} \frac{1}{2}\sum_{(i,j)\in\Omega} C_{i,j}^2((UW)_{i,j} - M_{i,j})^2 + \frac{\lambda^2}{2}\sum_{(i,j)\notin\Omega}(UW)_{i,j}^2. \tag{6.7}$$

Here r is the given rank, $U \in \mathbf{R}^{m\times r}$ is any matrix such that its column space $\mathcal{U}$ belongs to the Grassmann manifold $\mathcal{G}(m, r)$. The confidence index $C_{i,j} > 0$ is introduced for each observation, and λ is a weighted parameter. A Riemannian trust-region method, GenRTR [92] was used to solve the above optimization problem on the Grassmannian. According to the numerical experiments, RTRMC outperforms other state-of-the-art algorithms on a wide range of problem instances. It is especially efficient for rectangular matrices and achieves a much smaller relative error.

However, its performance will be ruined when sparse random-valued noise is introduced to the measurements. In order to obtain better result, adaptively finding the error locations and reconstructing the matrix can be combined together as in [64, 43]. Here we will plant this idea into the existing model.

We define K as the number of error terms in the given data and derive the

following revised model:

$$\begin{cases} \underset{\mathcal{U},W,\Lambda}{\text{minimize}} & \frac{1}{2}\sum_{(i,j)\in\Omega} C_{i,j}^2\Lambda_{i,j}((UW)_{i,j}-M_{i,j})^2+\frac{\lambda^2}{2}\sum_{(i,j)\notin\widetilde{\Omega}}(UW)_{i,j}^2 \\ \text{s.t.} & \sum_{(i,j)\in\Omega}(1-\Lambda_{i,j})\leq K,\ \Lambda_{i,j}\in\{0,1\}. \end{cases} \tag{6.8}$$

Here $\Lambda \in \mathbf{R}^{m\times n}$ is a binary matrix denoting the "correct" data:

$$\Lambda_{i,j} = \begin{cases} 1, & \text{if } (i,j)\in\Omega,\ M_{i,j} \text{ is "correct"}, \\ 0, & \text{otherwise.} \end{cases} \tag{6.9}$$

and $\widetilde{\Omega}$ is a subspace of Ω such that $\Lambda_{i,j}=1$ for all the indices in $\widetilde{\Omega}$. In this work, we only consider the case with sparsely corrupted measurements, and the other measurements are assumed to be correct. The parameter λ is set to be 10^{-8}, i.e., the term $\sum_{(i,j)\notin\widetilde{\Omega}}(UW)_{i,j}^2$ can be neglected. All the entries of the confidence matrix C are chosen to be 1. Hence the model can be simplified into:

$$\begin{cases} \underset{\mathcal{U},W,\Lambda}{\text{minimize}} & \sum_{(i,j)\in\Omega}\Lambda_{i,j}((UW)_{i,j}-M_{i,j})^2, \\ \text{s.t.} & \sum_{(i,j)\in\Omega}(1-\Lambda_{i,j})\leq K,\ \Lambda_{i,j}\in\{0,1\}. \end{cases} \tag{6.10}$$

In order to solve this non-convex problem, we use alternating minimization method, which splits the energy minimization over Λ and U, W into two steps:

- Fix Λ and update U, W. We need to solve the following sub-problem:

$$\underset{\mathcal{U},W}{\text{minimize}} \sum_{(i,j)\in\widetilde{\Omega}}((UW)_{i,j}-M_{i,j})^2. \tag{6.11}$$

  This can be solved with RTRMC.

- Fix U, W and update Λ. This time we are solving:

$$\begin{cases} \underset{\Lambda}{\text{minimize}} & \sum_{(i,j)\in\Omega} \Lambda_{i,j}((UW)_{i,j} - M_{i,j})^2, \\ \text{s.t.} & \sum_{(i,j)\in\Omega} (1-\Lambda_{i,j}) \leq K,\ \Lambda_{i,j} \in \{0,1\}. \end{cases} \tag{6.12}$$

  This problem is to choose $|\Omega| - K$ elements with least sum from $\{((UW)_{i,j} - M_{i,j})^2,\ (i,j) \in \Omega\}$. Here $|\Omega|$ stands for the number of elements in set Ω. Defining τ as the value of the K^{th} largest term in that set, Λ can then be calculated by

$$(\Lambda)_{i,j} = \begin{cases} 1, & \text{if } (i,j) \in \Omega,\ ((UW)_{i,j} - M_{i,j})^2 < \tau, \\ 0, & \text{otherwise.} \end{cases} \tag{6.13}$$

  If the K^{th} and $(K+1)^{th}$ larges terms have the same value, then we can choose any Λ such that $\sum_{(i,j)\in\Omega}(1-\Lambda_{i,j}) = K$ and

$$\min_{(i,j)\notin\widetilde{\Omega}} ((UW)_{i,j} - M_{i,j})^2 \geq \max_{(i,j)\in\widetilde{\Omega}} ((UW)_{i,j} - M_{i,j})^2. \tag{6.14}$$

In each iteration, we use Λ to identify the location of outliers based on the newly constructed U and W. This outlier detection technique, defined as adaptive outlier pursuit (AOP), was firstly used in [64, 43]. Our algorithm is as follows:

Algorithm 3 RTRMC with AOP

Input: Ω, $\mathcal{P}_\Omega(M)$, Miter > 0, $r > 0$, $K \geq 0$, C, λ.
Initialization: $k = 0$, $\Lambda_{i,j} = 1$ for $(i,j) \in \Omega$ and 0 otherwise, $\widetilde{\Omega} = \Omega$.
while $k \leq$ Miter **do**
  Replace Ω in (6.7) with $\widetilde{\Omega}$ and update U^k and W^k with RTRMC.
  Update Λ^k with (6.13).
  Update $\widetilde{\Omega}$ to be the indices in Ω where $\Lambda^k_{i,j} = 1$.
  If this new $\widetilde{\Omega}$ is the same as the old $\widetilde{\Omega}$, break.
  $k = k + 1$.
end while
return $U^k W^k$.

This algorithm, together with other two methods, SpaRCS (sparse and low-rank decomposition via compressive sensing) [93] and GRASTA (Grassmannian robust adaptive subspace tracking algorithm) [94], will be studied and compared with extensive numerical experiments. SpaRCS is a recently proposed algorithm which aims at recovering low rank and sparse matrices from compressive measurements with the following model:

$$\underset{L,S}{\text{minimize}} \quad \|y - \mathcal{A}(L+S)\|_2, \text{ s.t. } \text{rank}(L) \leq r, \|S\|_0 \leq K. \tag{6.15}$$

Here $\|S\|_0$ stands for the number of nonzero entries of the matrix S. In our test this linear transformation $\mathcal{A}(L+S)$ is defined as the vector formed by the entries of $(L+S)$ in Ω. This model can be applied to solve MC problem with sparsely corrupted entries. We can form the vector y with the given noisy data. S can be treated as the matrix recording outliers, and L is the low-rank matrix we want to recover. GRASTA, a robust subspace tracking algorithm, is designed to tackle the following model:

$$\underset{S,W,\mathcal{U}}{\text{minimize}} \quad |\mathcal{P}_\Omega(S)|_1, \text{ s.t. } \mathcal{P}_\Omega(UW+S) = \mathcal{P}_\Omega(M), \ \mathcal{U} \in \mathcal{G}(m,r). \tag{6.16}$$

It alternates between estimating the subspace $\mathcal{U}$ with Grassmannian and finding the optimal W, S with augmented Lagrangian function. According to the numerical experiments in that paper, it can efficiently recover a low-rank matrix from partial measurements, even if the partially observed entries are corrupted by gross outliers.

6.3 Connection between (6.10) and (6.15)

In this section, we will show the connection between our problem and (6.15) with specially defined $\mathcal{A}(\cdot)$, i.e.

$$\underset{L,S\in\mathbf{R}^{m\times n}}{\text{minimize}} \quad \|\mathcal{P}_\Omega(M-L-S)\|_F, \text{ s.t. } \text{rank}(L)\leq r, \|S\|_0\leq K. \tag{6.17}$$

We can change $\|\cdot\|_F$ in the above problem to $\|\cdot\|_F^2$ while still getting the same solution. Basically, for (6.17) we are given partial entries of a matrix ($M_{i,j}$ with $(i,j)\in\Omega$) and we want to represent this M by the sum of a low-rank matrix L and a sparse matrix S.

If a matrix pair (L,S) satisfies the constraints of problem (6.17), we can define

$$\Lambda_{i,j}=\begin{cases}1, & \text{if } (i,j)\in\Omega,\ S_{i,j}=0,\\ 0, & \text{otherwise.}\end{cases} \tag{6.18}$$

Hence for any $(i,j)\in\Omega$, we have $M_{i,j}=L_{i,j}+S_{i,j}$ if $\Lambda_{i,j}=0$, since we can simply set $S_{i,j}=M_{i,j}-L_{i,j}$ for fixed L without violating the constraint on S. If $\Lambda_{i,j}=1$, we have $S_{i,j}=0$, thus $M_{i,j}-(L_{i,j}+S_{i,j})=M_{i,j}-L_{i,j}$. Therefore, (6.17) is equivalent to

$$\begin{cases}\underset{L,S,\Lambda}{\text{minimize}} & \sum\limits_{(i,j)\in\Omega}\Lambda_{i,j}(M_{i,j}-L_{i,j})^2\\ \text{s.t. } \text{rank}(L)\leq r, \|S\|_0\leq K,\ \Lambda \text{ satisfies (6.18).}\end{cases} \tag{6.19}$$

From the relation of Λ and S in (6.18) and the constraint $\|S\|_0\leq K$, we know the constraints for S and Λ in the above equation can be replaced by the constraint on Λ in (6.10). On the other hand, we know any matrix L with size $m\times n$ and $\text{rank}(L)\leq r$ can be written as the product of two matrices U and W, where $U\in\mathbf{R}^{m\times r}$ and $W\in\mathbf{R}^{r\times n}$. The global optimal solution of one problem is equivalent to the global optimal solution of the other problem. Since these two problems

are non-convex problems, a local optimal solution of one problem may not be equivalent to a local optimal solution of the other problem. However, the problem of minimizing the function of L only by eliminating S and Λ for both problems are the same.

6.4 The K Study

In practice, the exact number of corrupted entries K may be unknown. If K is underestimated, some damaged entries will still be used to solve the matrix completion problem, which will induce error for the reconstructed matrix. On the other hand, if K is overestimated, too many entries are removed and the reconstructed matrix may not be unique if the "new" sampling set is not large enough. We first focus on the case when K is overestimated.

When K is overestimated, we can always find more than one solution of problem (6.10) with the objective function value being zero. If K is overestimated by a small number, the product of U and W will be the same for all the solutions and we are able to recover the original low-rank matrix. When the difference is greater than a certain number, UW may not be unique. The following theorem provides a sufficient condition for the non-uniqueness of the matrix UW.

Theorem 6.4.1. *Suppose we are given p entries of an $m \times n$ matrix M, where the locations of these data are chosen randomly. We know in advance that K of the given entries are corrupted. Define the difference between our overestimated K value and the real K value as ΔK. If ΔK satisfies $\Delta K > (p-K)/\max(m,n) - r > 0$, then the reconstructed matrix will be non-unique.*

The above theorem provides a necessary condition for K estimate in order to have uniqueness of the problem. In practice, what we care about is how much we can overestimate K without sacrificing the accuracy of our algorithm. Since K is closely related to the location of the known data, how the given entries are dis-

tributed over the matrix will be important for our study. In the following theorem, we assume that whether the value at each entry is given or not is independent and identically distributed. Let us define k_i^r as the number of given entries in the i^{th} row and k_j^c as the number of given entries in the j^{th} column. $k_{\min}$ denotes the minimum of all the k_i^r and k_j^c. Through extensive numerical tests, we notice that the distribution of $k_{\min}$ is similar to the conditional distribution of $k_{\min}$ given the total number of known entries. For simplicity we use the former one to replace the conditional distribution and arrive at the following theorem.

Theorem 6.4.2. *Suppose that the probability of each entry being given is* $q = (p - K)/(mn)$*, and whether the entry is known or not is independent of other entries. For any small number* $P_0 \in (0, 1)$*, let us define*

$$K_1 = \min \left(nq - \sqrt{\frac{-n}{2} \log(1 - (1 - P_0/2)^{1/m})}, \right.$$
$$\left. mq - \sqrt{\frac{-m}{2} \log(1 - (1 - P_0/2)^{1/n})} \right) \tag{6.20}$$
$$K_2 = \min \left(nq - \sqrt{-2nq \log(1 - (1 - P_0/2)^{1/m})}, \right.$$
$$\left. mq - \sqrt{-2mq \log(1 - (1 - P_0/2)^{1/n})} \right). \tag{6.21}$$

Then with at most P_0 *probability there exists one row or column in this matrix with at most* $\max(K_1, K_2)$ *given entries.*

The proof of the above two theorems can be found in the appendix. As we know, the uniqueness of MC problem with outliers depends on a lot of subtle factors. Here we want to derive an empirical upper bound for ΔK such that when ΔK is bounded by this value, with high probability our algorithm can recover the exact matrix. Considering the revised sampling set (the set with $(p-K)$ entries), in order to study the relationship between this upper bound and the number of given entries in each row and column, we design the following experiment. We

first fix the matrix size 512×512. For each rank r, the sample ratio sr, defined as $p/(mn)$, is chosen as the smallest number which could guarantee the exact matrix recovery when the real K value is used as input. For more details about the sr value, we refer the readers to the phase transition charts in Section 6.5. Labeling the minimum of all the k_i^r and k_j^c from the revised sampling set as $k_{\min}$, we randomly choose 10 positive integers bounded above by $(k_{\min} - r)$ and treat them as ΔK. For each input $(K + \Delta K)$, the error of the recovered matrix M_r is calculated with the following expression:

$$\text{equ:MC-Err} = \max_{i,j} \ |M_{i,j} - (M_r)_{i,j}|. \tag{6.22}$$

If the error is less than 10^{-4}, we say the recovery is successful. Otherwise, we label it as a failure. The results displayed in Table 6.1 come from the average of 20 different tests for each setting.

Through massive experiments, we can see that if ΔK is smaller than $(k_{\min} - r)$, our algorithm can find the correct matrix with extremely high probability. Hence we come up with the following conclusion: for any small number P_0, we can find two values K_1 and K_2 according to Theorem 2 such that with at most P_0 probability there exists one row or column with at most $\max(K_1, K_2)$ given entries. Then, if ΔK is less than $(\max(K_1, K_2) - r)$, with high probability our algorithm will return the exact matrix with this overestimated K input.

In application, when K is overestimated, according to the above conclusion we just need to construct a strategy to update it such that ΔK can be bounded by $(k_{\min} - r)$. Let us define $\widetilde{K}$ as the estimated K value. One intuitive idea is to check the value of the $\widetilde{K}^{th}$ largest term in set $S_\Omega = \{((UW)_{i,j} - M_{i,j})^2, \ (i,j) \in \Omega\}$ in each iteration. If this value is less than the tolerance K_{tol}, it is possible that some of the deleted data are not outliers, and we can update $\widetilde{K}$ to be the number of terms in this set which are greater than K_{tol}. When our algorithm reaches

rank & sample rate	avg K/p	avg $k_{\min}$	success percentage
rank=5, sr=0.15	0.01	50.75	100%
	0.03	49.15	100%
	0.05	49.00	100%
	0.07	47.05	100%
	0.09	45.35	100%
rank=10, sr=0.20	0.01	72.25	100%
	0.03	70.70	100%
	0.05	68.40	100%
	0.07	67.65	100%
	0.09	66.30	100%
rank=15, sr=0.25	0.01	95.25	100%
	0.03	93.20	100%
	0.05	92.75	100%
	0.07	89.25	100%
	0.09	85.40	100%
rank=20, sr=0.30	0.01	119.55	100%
	0.03	116.70	100%
	0.05	114.50	100%
	0.07	111.30	100%
	0.09	107.25	100%
rank=25, sr=0.35	0.01	143.30	100%
	0.03	139.20	100%
	0.05	136.15	100%
	0.07	134.10	100%
	0.09	129.30	100%

Table 6.1: The K study. For each matrix 10 different K inputs are chosen and 20 trials are conducted for each matrix setting.

a certain stage, we can calculate the minimum number of entries in one row or column from the sampling set with $(p - \widetilde{K})$ elements (label as $\widetilde{k}_{\min}$). If it is less than r, we update $\widetilde{K} = \widetilde{K} + \widetilde{k}_{\min} - r$. Here we just choose the smallest decrease in $\widetilde{K}$. In fact, we may pick a larger decrease in order to reduce the number of outer iterations. On the other hand, when K is underestimated, we need to increase its value in order to remove all the outliers. As we know, when there are outliers in the given data, it is quite possible that we are not able to find a low-rank matrix with entries equalling the given data at these locations. Based on the

fast convergence of our algorithm, if at certain iteration the difference between the entries of the recovered matrix at those $(p - \widetilde{K})$ locations and the associated input data are greater than tolerance ϵ, we can update $\widetilde{K}$ to be $\rho_1 \widetilde{K}$ with $\rho_1 > 1$. ρ_1 should be chosen properly. When it is too small, we need a lot of steps to make $\widetilde{K}$ larger than the exact K, however, when ρ_1 is too large, $\widetilde{K}$ may go far above the exact K, and more iterations are required to decrease its value. Therefore, we arrive at the following algorithm.

Algorithm 4 RTRMC-AOP with K update

Input: Ω, $\mathcal{P}_\Omega(M)$, InnerMiter, OuterMiter > 0, $r > 0$, $\widetilde{K} \geq 0$, C, λ, K_{tol}, ρ_1, ϵ .
Initialization: $k, l = 0$, $\Lambda_{i,j} = 1$ for $(i,j) \in \Omega$ and 0 otherwise, $\widetilde{\Omega} = \Omega$.
while $l \leq$ OuterMiter **do**
  while $k \leq$ InnerMiter **do**
    Replace Ω in (6.7) with $\widetilde{\Omega}$ and update U^k and W^k with RTRMC.
    Find the $\widetilde{K}^{th}$ largest term in set S_Ω.
    If it is less than K_{tol}, update $\widetilde{K}$ to be the number of elements in S_Ω that are greater than K_{tol}.
    Update Λ^k with (6.13).
    Update $\widetilde{\Omega}$ to be the indices in Ω where $\Lambda^k_{i,j} = 1$.
    If this new $\widetilde{\Omega}$ is the same as the old $\widetilde{\Omega}$, break.
    $k = k + 1$.
  end while
  Calculate $\widetilde{k}_{\min}$ with the updated $\widetilde{K}$ value.
  If $\widetilde{k}_{\min} < r$, $\widetilde{K} = \widetilde{K} + \widetilde{k}_{\min} - r$.
  If $\widetilde{k}_{\min} \geq r$ and the function value of (6.11) is less than ϵ, break.
  If $\widetilde{k}_{\min} \geq r$ and the function value is greater than ϵ, $\widetilde{K} = \rho_1 \widetilde{K}$.
  $k = 0$, $l = l + 1$.
end while
return $U^k W^k$, $\widetilde{K}$.

6.5 Numerical Results

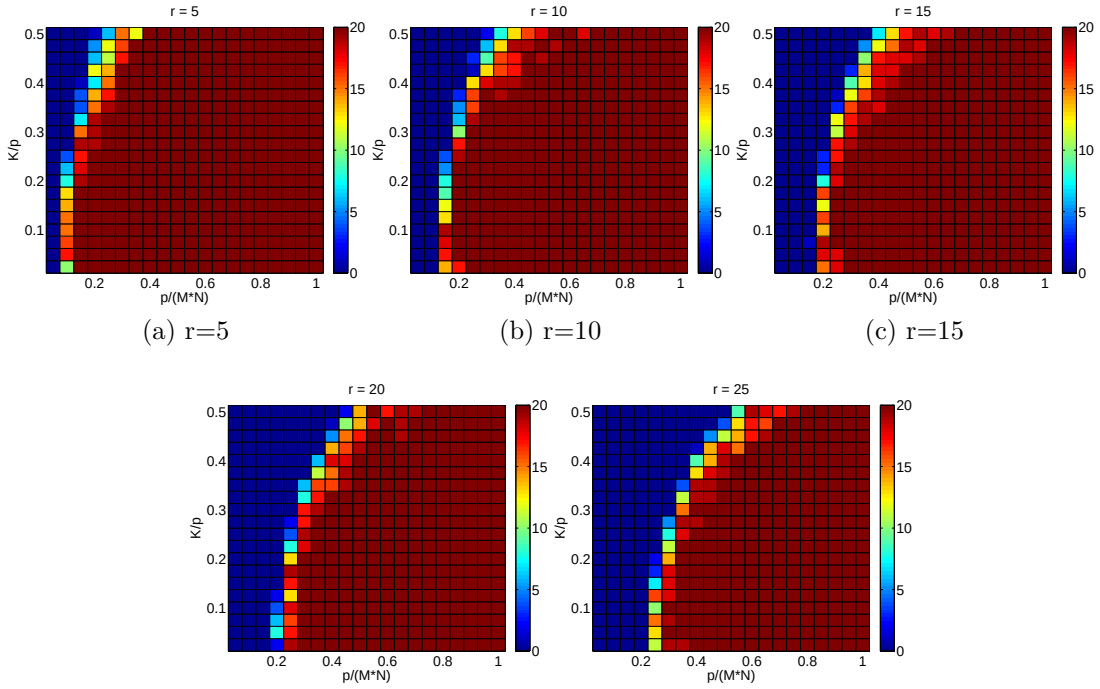
In this section we use some numerical experiments to demonstrate the effectiveness of our algorithm (AOP for short). AOP, together with SpaRCS and GRASTA are studied and compared.

In each experiment, the original matrix M is generated by the product of $U \in \mathbf{R}^{m\times r}$ and $W \in \mathbf{R}^{r\times n}$, whose entries follow independent and identically distributed (i.i.d.) Gaussian distribution with variance 1. We denote the largest and smallest value of M as m_L and m_S. p entries are chosen randomly from M and their locations are recorded in Ω. We then pick K locations randomly from Ω and replace the values at these locations by a random number from $[m_S, m_L]$. The corrupted p entries and Ω are used as input in our code.

In the first experiment, we investigated the empirical performance of Algorithm 1 by testing it under different circumstances. Here the size of the matrix is chosen to be $m = n = 512$ and different values of r, p and K are considered. For each small rectangle in Figure 6.1, we fix the value of r, p and K, and applied AOP to recover the low-rank matrix. If the relative error, i.e. $\|M_r - M\|_F/\|M\|_F$, is smaller than 10^{-3}, we denote the test as "successful". 20 different tests are conducted for each setting and the probability of successful recovery are recorded. Here red indicates recovery success and blue indicates failure. As expected, the performance gets worse when we increase r or K or decrease p.

In the following two tests, the results of SpaRCS and GRASTA are also shown in the figures. Let us assume the K value is known in advance, i.e. Algorithm 1 is used in the comparison. We will compare these three items (since GRASTA solves a different problem, only the relative error is compared):
1) distance between $P_{\widetilde{\Omega}}(M_r)$ and $P_{\widetilde{\Omega}}(M)$, i.e. $\|P_{\widetilde{\Omega}}(M_r) - P_{\widetilde{\Omega}}(M)\|_F$;
2) relative error;
3) the probability of correct detections of corrupted data in the noisy measurements.

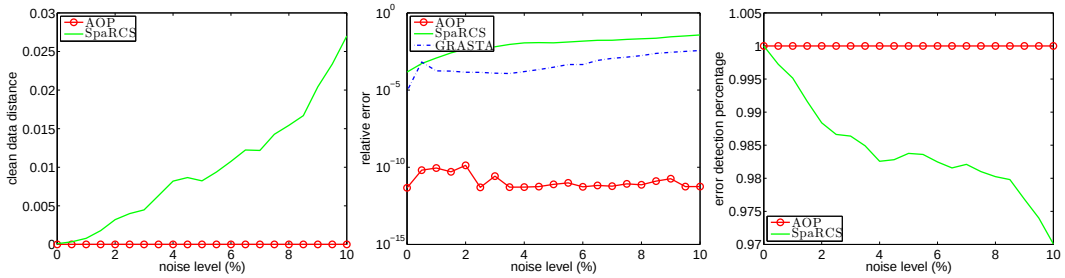
In our second experiment, we set $m = n = 500$, $r = 10$, and examine the performance of these algorithms on data with different noise levels. Here the noise level is defined as K/p. 21 noise levels are chosen by $5 \cdot i \cdot 10^{-3}$ with i ranging from 0 to 20. p is calculated by $6r(m+n-r)$. 20 trials are performed for



(a) r=5 (b) r=10 (c) r=15

(d) r=20 (e) r=25

Figure 6.1: Phase transitions for a recovery problem of size 512×512. Aggregate results are shown over 20 Monte-Carlo runs at each specification of r, K and p. Red indicates recovery success and blue indicates failure.

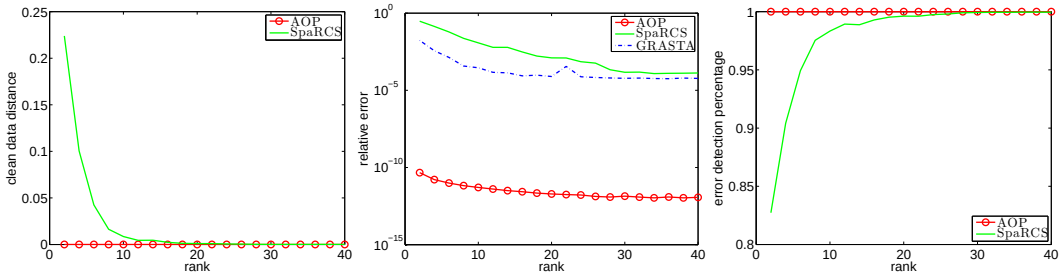


(a) Distance between clean data (b) Relative error (c) Error detection

Figure 6.2: Algorithm comparison on corrupted data with different noise levels. (a) Distance between $P_{\widetilde{\Omega}}(M_r)$ and $P_{\widetilde{\Omega}}(M)$ vs noise level, (b) Relative error vs noise level, (c) Error location detection vs noise level. AOP proves to be more robust to contaminated data compared with others.

each noise level and the mean of the above three items are recorded in Figure 6.2. The plots demonstrate that in these comparisons AOP outperforms the other two greatly for all noise levels. According to the relative error plot, the result from our

method is always around 10^{-10} while the result gained by the other two algorithms is bounded below by 10^{-4}. Compared with SpaRCS, GRASTA is slightly more stable when the given data is ruined by gross outliers. In plot (c), we record the probabilities of correct error locations detection. From the graph we can see that AOP algorithms can find all the positions of corrupted data with probability 1, while in comparison the performance of SpaRCS detection is not very satisfying.



(a) Distance between clean data

(b) Relative error

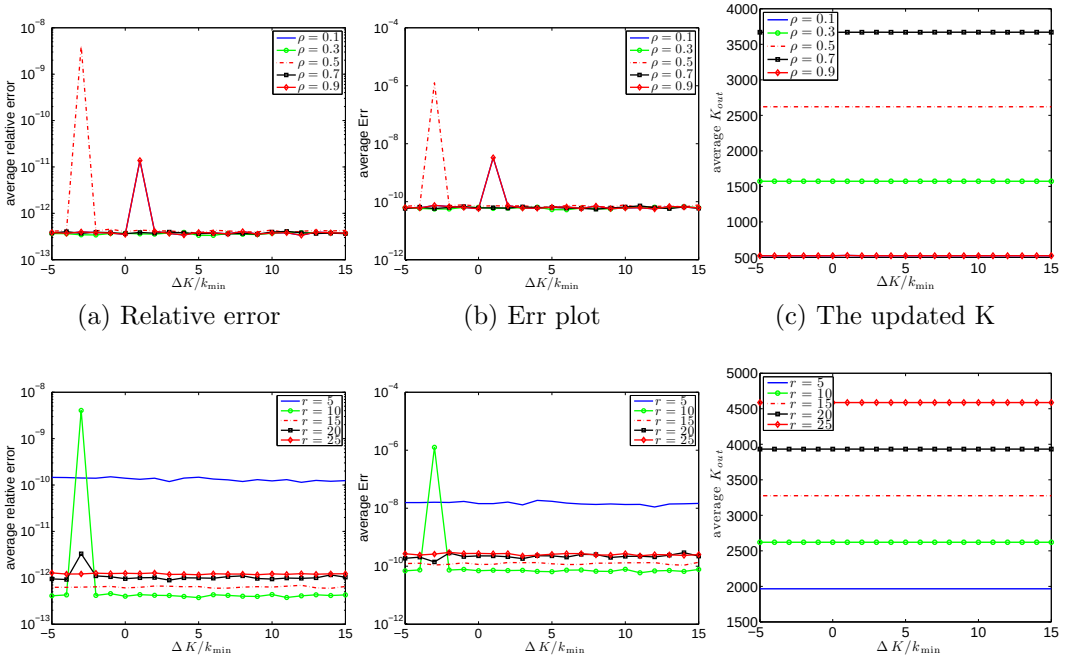
(c) Error detection

Figure 6.3: Algorithm comparison on corrupted data with different ranks. (a) Distance between $P_{\widetilde{\Omega}}(M_r)$ and $P_{\widetilde{\Omega}}(M)$ vs rank, (b) Relative error vs rank, (c) Error location detection vs rank. AOP yields a remarkable improvement in reducing the relative error and finding the correct error locations compared with others.

Next we fix $m = n = 500$ and the noise level 5% and change r between 2 and 40. p is still calculated by $6r(m+n-r)$. The results in Figure 6.3 come from the average of 20 tests. We can see that the distance between $P_{\widetilde{\Omega}}(M_r)$ and $P_{\widetilde{\Omega}}(M)$ and the relative error tend to decrease as the rank of M increases. Although all the algorithms show the same trend, AOP series always return a much better result with relative error staying around 10^{-11}, and it detects the true error locations with probability 1 in all the tests. The relative error from GRASTA is bounded below by 10^{-5} and when the rank is extremely low, the relative error could be as high as 10^{-2}.

In the previous experiments, we assume the exact K value is always given. Now let us study the case when the input K is just an estimate of the actual number of errors. This time we fix $m = n = 512$, and examine the performance

of Algorithm 2 under different settings. The relative error, Err defined by (6.22) and the updated K value will be displayed here. In Figure (a)-(c), we set $r = 10$ and pick 5 different noise levels between 0.01 and 0.09. For each setting, we calculate $k_{\min}$, and choose 21 different ΔK between $-5k_{\min}$ and $15k_{\min}$. Then each $(K + \Delta K)$ is used as the input K value. In Figure (d)-(f), we fix the noise level to be 0.05 and vary r from 5 to 25. Still, 21 different ΔK values are selected. All the p values in these tests are chosen the same as Table 6.1. The following results come from the average of 20 different trials. We can see that in all the cases our AOP with K update algorithm can detect the correct number of outliers with high probability even when the input K differs a lot from its real value. The relative error plot and Err plot suggest that this method always recovers the matrices with extremely high accuracy.



(a) Relative error (b) Err plot (c) The updated K

(d) Relative error (e) Err plot (f) The updated K

Figure 6.4: The K study. For (a)-(c) we fix the rank and vary the noise level. (a) relative error vs different K input. (b) Err vs different K input. (c) K output vs different K input. For (d)-(f) we fix the noise level and vary the rank of the matrix. (d) relative error vs different K input. (e) Err vs different K input. (f) K output vs different K input.

6.6 Conclusion

In this paper, we propose a method for exact low-rank matrix completion from sparsely corrupted data via adaptive outlier pursuit. By iteratively detecting the damaged measurements and recovering the matrix from "correct" measurements, this method can obtain better results in both finding the noisy measurements and recovering the exact matrix when random-valued noise is introduced in the measurements. Our algorithm is implemented and compared with SpaRCS and GRASTA in the numerical experiments. It has better performance in many aspects compared to the other two, especially in detecting all the outlier locations. When the exact value of the number of outliers is not provided, the AOP with K update algorithm can always detect the correct number of outliers and recover the exact matrix in all the cases with high probability. Our next step is to study the case when the given data is corrupted by Gaussian noise as well as sparse random-valued noise.

Appendix

This appendix provides the mathematical proofs of the theoretical results in Section 6.4.

Proof of Theorem 6.4.1

Proof. When we overestimate K, the K outliers will be found to make the objective function 0. Therefore, we only need to consider the $(p - K)$ correct entries. In the mean time, some non-outliers (the overestimated ΔK entries) are also considered as outliers and will not be used for matrix completion. As we know, if the number of given entries in one row (or column) is less than r, the reconstructed matrix is not unique. Since ΔK of the $(p - K)$ known entries will not be used

in reconstructing the matrix, when ΔK is large enough to make the number of known entries in one row (or column) less than r, we will have more than one solution. It is easily seen that the smallest number of known entries in one row (or column) of the matrix is less than or equal to $\lfloor (p-K)/m \rfloor$ (or $\lfloor (p-K)/n \rfloor$), here $\lfloor x \rfloor$ is the largest integer that does not exceed x. Without loss of generality, let us assume column j has the smallest number of known entries. It is obvious that this number is bounded by $\min(\lfloor (p-K)/m \rfloor, \lfloor (p-K)/n \rfloor)$. To make the number of known entries in this column no less than r, the smallest number of entries to be deleted should not exceed $\min(\lfloor (p-K)/m \rfloor, \lfloor (p-K)/n \rfloor) - r + 1$. Thus if ΔK is greater than $\min(\lfloor (p-K)/m \rfloor, \lfloor (p-K)/n \rfloor) - r$, the reconstructed matrix will not be unique. □

Proof of Theorem 6.4.2

Proof. Since the probability that a certain location is chosen is fixed and equals $q = (p-K)/(mn)$. In addition, whether one entry is chosen or not is independent of other entries. The number of known entries in each row (or column) of this matrix follows binomial distribution. For each row, the cumulative distribution can be expressed as

$$F(x, n, q) = P(X \leq x) = \sum_{i=0}^{\lfloor x \rfloor} \binom{n}{i} q^i (1-q)^{n-i}, \tag{6.23}$$

where X is the number of known entries in this row.

From the Hoeffding's inequality, we have

$$F(k, n, q) \leq \exp\left(-2\frac{(nq-k)^2}{n}\right), \tag{6.24}$$

for any integer $k \leq nq$. Since the distribution of the number of known entries in each row is independent, we can find the upper bound for the probability that

there exists one row with at most k given entries:

$$P(\min(k_1^r, k_2^r, \cdots, k_m^r) \leq k) \leq 1 - \left(1 - \exp\left(-2\frac{(nq-k)^2}{n}\right)\right)^m := P_1. \quad (6.25)$$

Here k_i^r stands for the number of given entries in the i^{th} row. Similarly the upper bound for the probability that there exists one column with at most k given entries can be expressed as follows:

$$P(\min(k_1^c, k_2^c, \cdots, k_n^c) \leq k) \leq 1 - \left(1 - \exp(-2\frac{(mq-k)^2}{m})\right)^n := P_2. \quad (6.26)$$

where k_j^c is defined as the number of given entries in the j^{th} column. Combing these two together, we have

$$P(\min(k_1^r, k_2^r, \cdots, k_m^r, k_1^c, k_2^c, \cdots, k_n^c) \leq k) \leq P_1 + P_2 \leq 2\max(P_1, P_2), \quad (6.27)$$

which means the probability that there exists one row or column with at most k given entries can be bounded by $2\max(P_1, P_2)$.

Let us first assume $P_1 > P_2$. Defining

$$P_0 = 2\left(1 - \left(1 - \exp\left(-2\frac{(nq-k)^2}{n}\right)\right)^m\right), \quad (6.28)$$

we then have

$$\exp\left(-2\frac{(nq-k)^2}{n}\right) = 1 - \left(1 - \frac{P_0}{2}\right)^{1/m} \quad (6.29)$$

$$\Longrightarrow k = nq - \sqrt{\frac{-n}{2}\log(1 - (1 - P_0/2)^{1/m})}. \quad (6.30)$$

When $P_1 < P_2$, we have

$$k = mq - \sqrt{\frac{-m}{2}\log(1 - (1 - P_0/2)^{1/n})}. \quad (6.31)$$

We define K_1 as the minimal of these two values. Hence given P_0, the probability of having one row or column with at most K_1 entries is less than P_0.

Besides the Hoeffding's inequality, we also have Chernoff's inequality,

$$F(k,n,q) \leq \exp\left(-\frac{1}{2q}\frac{(nq-k)^2}{n}\right). \tag{6.32}$$

In this case

$$P_1 = 1 - \left(1 - \exp\left(-\frac{1}{2p}\frac{(nq-k)^2}{n}\right)\right)^m \tag{6.33}$$

$$P_2 = 1 - \left(1 - \exp\left(-\frac{1}{2p}\frac{(mq-k)^2}{m}\right)\right)^n. \tag{6.34}$$

After similar calculation, we have

$$k = nq - \sqrt{-2nq\log(1-(1-P_0/2)^{1/m})} \tag{6.35}$$

for $P_1 > P_2$, and

$$k = mq - \sqrt{-2mq\log(1-(1-P_0/2)^{1/n})} \tag{6.36}$$

when $P_1 < P_2$. Similarly we define K_2 to be the smaller one of these two values, and the probability of having one row or column with at most K_2 entries is less than P_0. Combining the results from two inequalities together, we know that with at most P_0 probability there exists one row or column with at most $\max(K_1, K_2)$ given entries. □

Acknowledgments

This work is supported by NSF Grant DMS-0714945, Center for Domain-Specific Computing (CDSC) under the NSF Expeditions in Computing Award CCF-

0926127, ONR Grant N00014-11-1-0719, N00014-08-1-119 and an ARO MURI subcontract from Rice University.

References

[1] R. G. Baraniuk, "Compressive sensing [lecture notes]", *Signal Processing Magazine, IEEE*, vol. 24, no. 4, pp. 118–121, 2007. 4

[2] A. Wagadarikar, R. John, R. Willett, and D. Brady, "Single disperser design for coded aperture snapshot spectral imaging", *Applied optics*, vol. 47, no. 10, pp. B44–B51, 2008. 4, 35

[3] A. A. Wagadarikar, N. P. Pitsianis, X. Sun, D. J. Brady, et al., "Video rate spectral imaging using a coded aperture snapshot spectral imager", *Opt. Express*, vol. 17, no. 8, pp. 6368–6388, 2009. 4, 35

[4] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling", *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 83–91, 2008. 4

[5] M. Wakin, J. Laska, M. Duarte, D. Baron, S. Sarvotham, D. Takhar, K. F. Kelly, and R. G. Baraniuk, "Compressive imaging for video representation and coding", in *Picture Coding Symposium*, 2006. 4

[6] P. Hasler and D. V. Anderson, "Cooperative analog-digital signal processing", in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*. IEEE, 2002, vol. 4, pp. IV–3972. 4

[7] R. Fergus, A. Torralba, and W. T. Freeman, "Random lens imaging", 2006. 4

[8] J. Gu, S. Nayar, E. Grinspun, P. Belhumeur, and R. Ramamoorthi, "Compressive structured light for recovering inhomogeneous participating media", *Computer Vision–ECCV 2008*, pp. 845–858, 2008. 4

[9] W. L. Chan, M. L. Moravec, R. G. Baraniuk, and D. M. Mittleman, "Terahertz imaging with compressed sensing and phase retrieval", *Optics letters*, vol. 33, no. 9, pp. 974–976, 2008. 4

[10] S. Marchesini, "Ab initio compressive phase retrieval", *arXiv preprint arXiv:0809.2006*, 2008. 4

[11] A. Poglitsch, C. Waelkens, N. Geis, H. Feuchtgruber, B. Vandenbussche, L. Rodriguez, O. Krause, E. Renotte, C. Van Hoof, P. Saraceno, et al., "The photodetector array camera and spectrometer (pacs) on the herschel space observatory", *Astronomy and Astrophysics*, vol. 518, 2010. 4

[12] M. Lustig, D. Donoho, and J. 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. 5, 35

[13] D. Le Gall, "Mpeg: A video compression standard for multimedia applications", *Communications of the ACM*, vol. 34, no. 4, pp. 46–58, 1991. 5, 35

[14] J. Zheng and E. L. Jacobs, "Video compressive sensing using spatial domain sparsity", *Optical Engineering*, vol. 48, no. 8, pp. 087006–087006, 2009. 5

[15] Y. Hitomi, J. Gu, M. Gupta, T. Mitsunaga, and S. K. Nayar, "Video from a single coded exposure photograph using a learned over-complete dictionary", in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 287–294. 5, 35

[16] D. J. L. Gall, "The mpeg video compression algorithm", *Signal Processing: Image Communication*, vol. 4, no. 2, pp. 129 – 140, 1992. 5, 7

[17] P. Llull, X. Liao, X. Yuan, J. Yang, D. Kittle, L. Carin, G. Sapiro, and D. J. Brady, "Coded aperture compressive temporal imaging", *Optics express*, vol. 21, no. 9, pp. 10526–10545, 2013. 5, 6, 7, 20, 35, 36, 41

[18] X. Yuan, J. Yang, P. Llull, X. Liao, G. Sapiro, D. J. Brady, and L. Carin, "Adaptive temporal compressive sensing for video", *algorithms*, vol. 10, no. 11, pp. 12–13, 2013. 5, 7, 9, 35

[19] R. V. Babu and A. Makur, "Object-based surveillance video compression using foreground motion compensation", in *Control, Automation, Robotics and Vision, 2006. ICARCV'06. 9th International Conference on*. IEEE, 2006, pp. 1–6. 6

[20] L. Liu, Z. Li, and E. J. Delp, "Efficient and low-complexity surveillance video compression using backward-channel aware wyner-ziv video coding", *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 4, pp. 453–465, 2009. 6

[21] B. K. Horn and B. G. Schunck, "Determining optical flow", *Artificial intelligence*, vol. 17, no. 1, pp. 185–203, 1981. 7, 16

[22] J. L. Barron, D. J. Fleet, and S. Beauchemin, "Performance of optical flow techniques", *International journal of computer vision*, vol. 12, no. 1, pp. 43–77, 1994. 7, 16

[23] G. Adiv, "Determining three-dimensional motion and structure from optical flow generated by several moving objects", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, , no. 4, pp. 384–401, 1985. 7, 16

[24] C.-H. Hsieh and T.-P. Lin, "Vlsi architecture for block-matching motion estimation algorithm", *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 2, no. 2, pp. 169–175, 1992. 7

[25] M. Ezhilarasan and P. Thambidurai, "Simplified block matching algorithm for fast motion estimation in video compression", *Journal of Computer Science*, vol. 4, no. 4, pp. 282–289, 2008. 7

[26] B. D. Lucas, T. Kanade, et al., "An iterative image registration technique with an application to stereo vision", in *Proceedings of the 7th international joint conference on Artificial intelligence*, 1981. 16

[27] L. G. Brown, "A survey of image registration techniques", *ACM computing surveys (CSUR)*, vol. 24, no. 4, pp. 325–376, 1992. 16

[28] C. Le Guyader and L. A. Vese, "A combined segmentation and registration framework with a nonlinear elasticity smoother", in *Scale Space and Variational Methods in Computer Vision*, pp. 600–611. Springer, 2009. 16

[29] I. Yanovsky, C. Le Guyader, A. Leow, P. Thompson, and L. Vese, "Nonlinear elastic registration with unbiased regularization in three dimensions", in *Computational Biomechanics for Medicine III, MICCAI 2008 Workshop*, 2008. 16

[30] T. Lin, E.-F. Lee, I. Dinov, C. Le Guyader, P. Thompson, A. Toga, and L. A. Vese, "A landmark-based nonlinear elasticity model for mouse atlas registration", in *Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE International Symposium on*. IEEE, 2008, pp. 788–791. 16

[31] L. Ambrosio and G. D. Maso, "On the relaxation in bv $(\omega; \mathbb{R}^m)$ of quasi-convex integrals", *Journal of functional analysis*, vol. 109, no. 1, pp. 76–97, 1992. 17

[32] L. Ambrosio and G. Dal Maso, "A general chain rule for distributional derivatives", *Proceedings of the American Mathematical Society*, vol. 108, no. 3, pp. 691–702, 1990. 17

[33] J. M. Bioucas-Dias and M. A. Figueiredo, "A new twist: two-step iterative shrinkage/thresholding algorithms for image restoration", *Image Processing, IEEE Transactions on*, vol. 16, no. 12, pp. 2992–3004, 2007. 21, 37

[34] C. Li, W. Yin, H. Jiang, and Y. Zhang, "An efficient augmented Lagrangian method with applications to total variation minimization", *Computational Optimization and Applications*, vol. 56, no. 3, pp. 507–530, 2013. 21, 37

[35] T. Goldstein and S. Osher, "The split bregman method for l1-regularized problems", *SIAM Journal on Imaging Sciences*, vol. 2, no. 2, pp. 323–343, 2009. 22, 37

[36] W. Deng and W. Yin, "On the global and linear convergence of the generalized alternating direction method of multipliers", unpublished, 2012. 23, 39

[37] Y. Yang, H. Schaeffer, W. Yin, and S. Osher, "Mixing space-time derivatives for video compressive sensing", in *Asilomar Conference on Signals, Systems, and Computers*, 2013. 34

[38] P. T. Boufounos and R. G. Baraniuk, "One-bit compressive sensing", in *Conference on Information Sciences and Systems (CISS)*, Princeton, March 2008. 35, 52, 54, 55

[39] M. Yan, Y. Yang, and S. Osher, "Robust 1-bit compressive sensing using adaptive outlier pursuit", *IEEE Trans. Signal Process.*, vol. 6, pp. 3868–3875, 2012. 35, 72

[40] H. Schaeffer, Y. Yang, and S. Osher, "Real-time adaptive video compressive sensing", unpublished, 2013. 35

[41] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, "An iterative regularization method for total variation-based image restoration", *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 460–489, 2005. 37

[42] Y. Xu and W. Yin, "A fast patch-dictionary method for whole image recovery", 2013. 47

[43] M. Yan, Y. Yang, and S. Osher, "Exact low-rank matrix completion from sparsely corrupted entries via adaptive outlier pursuit", *Journal of Scientific Computing*, vol. 56, pp. 433–449, 2013. 53, 75, 76, 78

[44] E. Candès, "Compressive sampling", in *Int. Congress of Mathematics*, Madrid, Spain, 2006, vol. 3, pp. 1433–1452. 54

[45] E. J. Candès, J. K. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information", *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006. 54

[46] E. J. Candès, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements", *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006. 54

[47] E. J. Candès and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?", *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406–5425, 2006. 54

[48] D. L. Donoho, "Compressed sensing", *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006. 54

[49] E. Candès, "The restricted isometry property and its implications for compressed sensing", *Comptes Rendus Mathematique*, vol. 346, no. 9-10, pp. 589–592, 2008. 54

[50] J. Z. Sun and V. K. Goyal, "Quantization for Compressed Sensing Reconstruction", in *SAMPTA'09, International Conference on Sampling Theory and Applications*, L. Fesquet and B. Torrésani, Eds., Marseille, France, 2009, p. Special session on sampling and quantization. 54

[51] W. Dai, H. V. Pham, and O. Milenkovic, "Distortion-rate functions for quantized compressive sensing", in *IEEE Information Theory Workshop on Networking and Information Theory*, 2009, pp. 171–175. 54

[52] A. Zymnis, S. Boyd, and E. Candes, "Compressed sensing with quantized measurements", *IEEE Signal Processing Letters*, vol. 17, no. 2, pp. 149–152, 2010. 54

[53] J. N. Laska, P. T. Boufounos, M. A. Davenport, and R. G. Baraniuk, "Democracy in action: Quantization, saturation, and compressive sensing", *Applied and Computational Harmonic Analysis*, vol. 31, no. 3, pp. 429–443, 2011. 54

[54] L. Jacques, D. K. Hammond, and M.-J. Fadili, "Dequantizing compressed sensing: When oversampling and non-gaussian constraints combine.", *IEEE Transactions on Information Theory*, pp. 559–571, 2011. 54

[55] Y. Plan and R. Vershynin, "Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach", *Information Theory, IEEE Transactions on*, vol. 59, no. 1, pp. 482–494, 2013. 54

[56] J. N. Laska and R. G. Baraniuk, "Regime change: Bit-depth versus measurement-rate in compressive sensing", *Signal Processing, IEEE Transactions on*, vol. 60, no. 7, pp. 3496–3505, 2012. 54

[57] L. Jacques, J. N. Laska, P. T. Boufounos, and R. G. Baraniuk, "Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors", *IEEE transactions on information theory*, vol. 59, no. 4, pp. 2082–2102, 2013. 55, 57, 60, 63, 66, 67

[58] P. T. Boufounos, "Greedy sparse signal reconstruction from sign measurements", in *Proceedings of the 43rd Asilomar conference on Signals, systems and computers*, Piscataway, NJ, USA, 2009, Asilomar'09, pp. 1305–1309, IEEE Press. 55

[59] A. Gupta, R. Nowak, and B. Recht, "Sample complexity for 1-bit compressed sensing and sparse classification", in *Proceedings of the IEEE International Symposium on Information Theory*, 2010, pp. 1553–1557. 55

[60] J. N. Laska, Z. Wen, W. Yin, and R. G. Baraniuk, "Trust, but verify: Fast and accurate signal recovery from 1-bit compressive measurements", *IEEE Transactions on Signal Processing*, vol. 59, no. 11, pp. 5289–5301, 2011. 55

[61] Y. Plan and R. Vershynin, "One-bit compressed sensing by linear programming", *Communications on Pure and Applied Mathematics*, vol. 66, no. 8, pp. 1275–1297, 2013. 55

[62] T. Zhou and D. Tao, "Hamming compressed sensing", *ArXiv preprint arXiv:1110.0073*, 2011. 55

[63] T. Chen and H. R. Wu, "Adaptive impulse dectection using center-weighted median filters", *IEEE Signal Processing Letters*, vol. 8, no. 1, pp. 1–3, 2001. 56

[64] M. Yan, "Restoration of images corrupted by impulse noise and mixed gaussian impulse noise using blind inpainting", *SIAM Journal on Imaging Sciences*, vol. 6, no. 3, pp. 1227–1245, 2013. 56, 57, 75, 76, 78

[65] J. N. Laska, M. A. Davenport, and R. G. Baraniuk, "Exact signal recovery from sparsely corrupted measurements through the pursuit of justice", in *Proceedings of the 43rd Asilomar conference on Signals, systems and computers*, Piscataway, NJ, USA, 2009, Asilomar'09, pp. 1556–1560, IEEE Press. 58

[66] C. Studer, P. Kuppinger, G. Pope, and H. Bölcskei, "Recovery of sparsely corrupted signals", *IEEE Transactions on Information Theory*, to appear. 58

[67] H. Ji, C. Liu, Z. Shen, and Y. Xu, "Robust video denoising using low rank matrix completion", in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference*, June 2010. 73

[68] L. Elden, *Matrix Methods in Data Mining and Pattern Recognition*, Society for Industrial and Applied Mathematics, Philadephia, PA, 2007. 73

[69] M. Fazel, H. Hindi, and S. P. Boyd, "A rank minimization heuristic with application to minimum order system approximation", *Proceedings of the 2001 American Control Conference Cat No01CH37148*, vol. 6, no. 2, pp. 4734–4739, 2001. 73

[70] N. Linial, E. London, and Y. Rabinovich, "The geometry of graphs and some of its algorithmic applications", *Combinatorica*, vol. 15, pp. 215–245, 1995. 73

[71] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization", *Foundations of Computational Mathematics*, vol. 9, pp. 717–772, 2008. 73

[72] E. J. Candès and T. Tao, "The power of convex relaxation: Near-optimal matrix completion", *Information Theory, IEEE Transactions on*, vol. 56, no. 5, pp. 2053 –2080, May 2010. 73

[73] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization", *SIAM Review*, vol. 52, no. 3, August 2010. 73

[74] J. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion", *SIAM J. on Optimization*, vol. 20, pp. 1956–1982, March 2010. 73

[75] S. Ma, D. Goldfarb, and L. Chen, "Fixed point and Bregman iterative methods for matrix rank minimization", *Mathematical Programming*, vol. 128, no. 1-2, 2011. 73

[76] K.-c. Toh and S. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems", *Engineering*, vol. 117543, no. 3, pp. 1–31, 2009. 73

[77] Z. Wen, W. Yin, and Y. Zhang, "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm", *Mathematical Programming Computation*, vol. 4, no. 4, pp. 333–361, 2012. 74

[78] N. Boumal and P.-A. Absil, "RTRMC: A Riemannian trust-region method for matrix completion", in *Twenty-Fifth Annual Conference on Neural Information Processing Systems*, 2011. 74, 76

[79] R. Keshavan, A. Montanari, and S. Oh, "Matrix completion from a few entries", *Information Theory, IEEE Transactions on*, vol. 56, no. 6, pp. 2980 –2998, 2010. 74, 76

[80] P. Jain, R. Meka, and I. Dhillon, "Guaranteed rank minimization via singular value projection", in *Advances in Neural Information Processing Systems 23*, J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., pp. 937–945. 2010. 74

[81] J. Cai and S. Osher, "Fast singular value thresholding without singular value decomposition", 2010. 74

[82] R. H. Keshavan, A. Montanari, and S. Oh, "Low-rank matrix completion with noisy observations: a quantitative comparison", in *Proceedings of the 47th annual Allerton conference on Communication, control, and computing*. 2009, Allerton'09, pp. 1216–1222, IEEE Press. 75

[83] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?", *Journal of ACM*, vol. 58, no. 1, 2009. 75

[84] B. Dong, H. Ji, J. Li, Z. Shen, and Y. Xu, "Wavelet frame based blind image inpainting", *Applied and Computational Harmonic Analysis*, vol. 32, no. 2, 2012. 75

[85] H. Ji, S. Huang, Z. Shen, and Y. Xu, "Robust video restoration by joint sparse and low rank matrix approximation", *SIAM Journal on Imaging Sciences*, vol. 4, no. 4, 2012. 75

[86] W. Dai, E. Kerman, and O. Milenkovic, "A geometric approach to low-rank matrix completion", *Information Theory, IEEE Transactions on*, vol. 58, no. 1, pp. 237 –247, Jan. 2012. 76

[87] W. Dai, O. Milenkovic, and E. Kerman, "Subspace evolution and transfer (SET) for low-rank matrix completion", *Signal Processing, IEEE Transactions on*, vol. 59, no. 7, pp. 3120 –3132, July 2011. 76

[88] R. Keshavan and A. Montanari, "Regularization for matrix completion", in *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, June 2010, pp. 1503 –1507. 76

[89] G. Meyer, S. Bonnabel, and R. Sepulchre, "Linear regression under fixed-rank constraints: a Riemannian approach.", in *28th International Conference on Machine Learning (ICML)*, 2011. 76

[90] L. Balzano, R. Nowak, and B. Recht, "Online identification and tracking of subspaces from highly incomplete information", in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, 2010, pp. 704 –711. 76

[91] B. Vandereycken, "Low-rank matrix completion by Riemannian optimization", *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1214–1236, 2013. 76

[92] P. A. Absil, C. G. Baker, and K. A. Gallivan, "Trust-region methods on Riemannian manifolds", *Foundations of Computational Mathematics*, vol. 7, no. 3, pp. 303–330, 2006. 76

[93] A. E. Waters, A. C. Sankaranarayanan, and R. G. Baraniuk, "Sparcs: Recovering low-rank and sparse matrices from compressive measurements", in *Neural Information Processing Systems (NIPS)*, Granada, Spain, Dec. 2011. 79

[94] J. He, L. Balzano, and J. C. S. Lui, "Online robust subspace tracking from partial information", *CoRR*, vol. abs/1109.3827, 2011. 79