Compressive Sensing with Local Geometric Features

Rishi Gupta, Piotr Indyk, Eric Price
CSAIL, MIT
{rishig,indyk,ecprice}@mit.edu

Yaron Rachlin
Draper Laboratory
yaron.rachlin@alumni.cmu.edu

ABSTRACT

We propose a framework for compressive sensing of images with local geometric features. Specifically, let $x \in R^N$ be an N-pixel image, where each pixel p has value x_p. The image is acquired by computing the *measurement vector* Ax, where A is an $m \times N$ measurement matrix for some $m \ll N$. The goal is then to design the matrix A and recovery algorithm which, given Ax, returns an approximation to x.

In this paper we investigate this problem for the case where x consists of a small number (k) of "local geometric objects" (e.g., stars in an image of a sky), plus noise. We construct a matrix A and recovery algorithm with the following features: (i) the number of measurements m is $O(k \log_k N)$, which undercuts currently known schemes that achieve $m = O(k \log(N/k))$ (ii) the matrix A is ultra-sparse, which is important for hardware considerations (iii) the recovery algorithm is fast and runs in time sub-linear in N. We also present a comprehensive study of an application of our algorithm to a problem in satellite navigation.

Categories and Subject Descriptors: F.2 Analysis of Algorithms and Problem Complexity: General

General Terms: Theory, Algorithms

Keywords: Compressive sensing, Sparse matrices, Attitude determination, Star tracking

1. INTRODUCTION

In recent years, a new "linear" approach for acquiring digital images has been discovered [CRT06, Don06]. Traditional approaches to image acquisition first capture an entire N-pixel image and then process it for compression, transmission, or storage. In contrast, the new approach obtains a compressed representation directly, by acquiring a small number of nonadaptive linear measurements of the signal in hardware. Formally, for an image represented by a vector x, the representation is equal to Ax, where A is an $m \times N$ matrix. The advantage of this architecture is that it can use fewer sensors, and therefore can be cheaper and use less energy than a conventional camera [DDT$^+$08, FTF06, Rom09].

In order to reconstruct the image x from a lower-dimensional measurement vector (or sketch) Ax, one needs to assume that the image x is k-sparse for some k (i.e., it has at most k non-zero coordinates) or at least be "well-approximated" by a k-sparse vector. Then, given Ax one finds (an approximation to) x by performing *sparse recovery*. The latter problem is typically defined as follows: construct a matrix A such that, for any signal x, we can recover a vector $\hat{x}$ from Ax satisfying

$$\|x - \hat{x}\|_1 \leq C \cdot \mathrm{Err}_k^1(x) \tag{1}$$

where $\mathrm{Err}_k^1(x) = \min_{k\text{-sparse } x'} \|x - x'\|_1$ and C is the *approximation factor*. Note that if x is k-sparse, then $\mathrm{Err}_k^1(x) = 0$, and therefore $\hat{x} = x$. Although the main focus of this paper is signal acqusition, sparse recovery has applications to other areas such as data stream computing [Mut05, Ind07].

The problem of designing matrices A and corresponding recovery algorithms has been a subject of extensive study over the last few years, with the goal of designing schemes that enjoy good compression rate (i.e., low values of m) as well as good algorithmic properties (i.e., low encoding and recovery times). It is known by now that there exist binary matrices A and associated recovery algorithms that produce approximations $\hat{x}$ satisfying Equation (1) with constant approximation factor C and sketch length $m = O(k \log(N/k))$. In particular, a random Bernoulli matrix [CRT06] or a random binary matrix with column sparsity $O(\log(N/k))$ [BGI$^+$08] has this property with overwhelming probability. It is also known that this sketch length is asymptotically optimal [DIPW10, FPRU10]. See [GI10] for an overview.

In this paper we focus on sparse recovery with *ultra-low encoding complexity*, i.e., constant or "almost-constant" column sparsity. In addition, the matrices we construct are binary: all entries are either 0 or 1. Apart from general interest, our research is motivated by certain architectures for compressive imaging hardware that acquire all the compressive measurements concurrently by implementing the matrix multiply Ax. In particular, our "folding" measurements described in section 4 are readily implementable on systems that use *optically multiplexed imaging* [UGN$^+$09, TAN10]. For optical measurement architectures, due to the Poisson distributed shot noise affecting photon counting devices [HL07], splitting the signal in components of x can decrease the signal to noise ratio by as much as the square root of the column sparsity. Other potential advantages of ultra-low encoding complexity in electronic compressive im-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SCG'11, June 13–15, 2011, Paris, France.
Copyright 2011 ACM 978-1-4503-0682-9/11/06 ...$10.00.

agers include reduced interconnect complexity [Mei03], low memory requirements for storing the measurement matrix, and gain in image acquisition speed due to reduced operations.

Unfortunately, it is known [Nac10] that any *deterministic* scheme with guarantee as in Equation 1 requires column sparsity of $\Omega(\log(N/k))$. In the randomized case, where A is a random variable, and Equation 1 is required to hold only with constant probability over the choice of A, the same paper shows that any *binary* matrix A must have column sparsity as stated.

In this paper we show how to overcome the above limitations by employing a two-fold approach. First, we consider a class of images that possesses additional geometric structure (see next paragraph). Second, we relax the recovery guarantee, by requiring that only a constant fraction of the non-zero entries of the vector x are recovered correctly.

Model description

Our model for sparse images is motivated by astronomical imaging, where an image contains a small number of *distinguishable* objects (e.g., stars) plus some noise. We model each object as an image contained in a small $w \times w$ bounding box, for some constant w. The image is constructed by placing k objects in the image in an arbitrary fashion, subject to a *minimum separation constraint*. The image is then modified by adding *noise*. We formalize the notions of minimum separation constraint, distinguishability, and noise in the rest of this section.

Let x be an N-dimensional real vector, and assume $N = n^2$ for an integer n. We will treat x both as a vector and as an $n \times n$ matrix, with entries $x[i, j]$ for $i, j \in [n]$. An *object* o is a $w \times w$ real matrix. Let $O = \{o_1 \dots o_k\}$ be a sequence of k objects, and let $T = \{t_1 \dots t_k\}$ be a sequence of translations in x, i.e., elements from $[n - w]^2$. We say that T is *valid* if for any $i \neq j$ the translations t_i and t_j do not *collide*, i.e., we have $\|t_i - t_j\|_\infty \geq w'$ for some separation parameter $w' = \Omega(w)$. For $o \in O$ and $t = (t_x, t_y) \in T$, we define $t(o)$ to be a $w \times w$ matrix indexed by $\{t_x \dots t_x + w - 1\} \times \{t_y \dots t_y + w - 1\}$. The *ground truth image* is then defined as $x = \sum_i t(o_i)$.

During our algorithm, we impose a grid G on the image with cells of size $w' \times w'$. Let x_c be the image (i.e., an w'^2-dimensional vector) corresponding to cell c. We then use a projection F that maps each sub-image x_c into a *feature vector* $F(x_c)$. If $y \in x_c$ for some cell c, we use $F(y)$ to denote $F(x_c)$ after the entries of $x_c \setminus y$ are set to 0. If y is not a cell and not contained in a cell, we leave $F(y)$ undefined.

The distinguishability property we assume is that for any two distinct o, o' from the objects $O \cup \{\emptyset\}$, and for any two translations t and t', we have $\|F(t(o)) - F(t'(o'))\|_\Gamma > T$ (when it is defined) for some threshold $T > 0$ and some norm $\|\cdot\|_\Gamma$. In other words, different objects need to look different under F. For concreteness, the features we exploit in the experimental section are the *magnitude* (the sum of all pixels in the cell) and *centroid* (the sum of all pixels in the cell, weighted by pixel coordinates), since the magnitudes of stars follow a power law, and the centroid of a star can be resolved to .15 times the width of a pixel in each dimension. The distinguishability constraint is what ultimately allows us to undercut the usual lower bound by a factor of $\log k$.

The observed image x' is equal to $x + \mu$, where μ is a noise vector. The threshold T determines the total amount of noise that the algorithm tolerates. Specifically, let $\|\mu\|_F = \sum_c \|F(\mu_c)\|_\Gamma$, where μ_c is the noise corresponding to cell c. We assume that $\|\mu\|_F < \gamma k T$ for some constant $\gamma > 0$, and make no other assumptions about the noise.

Theoretical result

Our results assume sparsity parameter $k \geq C \log N$ for some constant C. We construct a distribution over random binary $m \times N$ matrices A, such that given Ax' for x' described above, we recover (with constant probability) a set D of k cells, such that at least $k/2$ of the cells fully containing an object are included in D[1]. The matrix has column sparsity $O(\log_k N)$. Note that if (say) $k = N^{1/2}$, then the column sparsity is *constant*. Moreover, the matrix A has only $m = O(k \log_k N)$ rows.

Empirical results

Our theoretical model is motivated by a task in satellite navigation formally known as *attitude* (i.e., orientation) *determination*. Many satellites compute their attitude by acquiring a picture of the night sky, extracting stars from the picture, and matching patterns of stars to an onboard database. We implement a standard attitude determination routine, with the picture aquisition and star extraction steps replaced with a simplified version of the algorithm presented in this paper. Our algorithm performs better recovery on small numbers of measurements and is orders of magnitude faster than comparable compressive sensing methods.

Our techniques

Our construction of the measurement matrix resembles those of other algorithms for sparse matrices, such as Count-Sketch [CCF02] or Count-Min [CM04]: we "hash" each cell c $s = O(\log_k N)$ times into s arrays of $q = O(k)$ "buckets", and sum all cells hashed to the same bucket. Each bucket defines one measurement of w'^2 pixels, which gives $m = O(k \log_k N)$. Hashing is done by using either the Chinese Reminder Theorem codes (i.e., "modulo prime" hashing) or Reed-Solomon codes. Both hash functions are standard in the literatue. It should be noted though that unlike in other scenarios, our use of CRT codes does not incur any additional polylogarithmic factors that typically lead to sub-optimal results.

The recovery process is based on the following novel approach. For simplicity, assume for now that the image contains no noise, and ignore the effect of two different objects being hashed to the same bucket. In this case, all buckets containing distinct objects are distinguishable from each other. Therefore, we can group non-empty buckets into k clusters of size s, with each cluster containing buckets with a single value. Since $q^s > N$, each cluster of buckets uniquely determines the cell in x containing the object in those buckets.

In order to make this approach practical, however, we need to make it robust to errors. The errors are due to distinct objects being hashed to the same bucket, the noise vector μ, and the grid cutting objects into pieces. Because of these issues, the clustering procedure aims to find clusters containing elements that are close to each other, rather than equal, and the procedure allows for some small fraction of outliers. For this purpose, we use the approximation algorithm for the k-center problem with outliers [CKMN01],

[1] A simple extension of the algorithm can then recover an approximation to x_c for any $c \in D$.

which correctly clusters "most" of the buckets. To handle the (small fraction of the) buckets that are grouped incorrectly, we construct our hash function using a constant rate error-correcting code.

2. HASHING

Our scheme works by "hashing" each cell c into s different arrays of size $O(k)$. We can think of this as a mapping f from $[N]$ to $[O(k)]^s$. As long as each character of the mapping is (approximately) pairwise independent, then in expectation most of the k objects will be alone in most of the array locations they map to. Our reconstruction algorithm will cluster the values in the cells, giving us a noisy version y' of the true codeword $y = f(c)$ with a constant fraction of errors. We then need to efficiently decode from y' to c.

Hence, we need an efficient error correcting code that is also approximately pairwise independent in each character. This section gives precise definitions of our requirements, then gives two codes achieving them.

2.1 Definitions

DEFINITION 2.1. *A hash family $\mathcal{H}$ of functions $h\colon A \to B$ is* pairwise-independent *if, for any $x_1, x_2 \in A$ and $y_1, y_2 \in B$ with $x_1 \neq x_2$, we have $\Pr_{h\in\mathcal{H}}[h(x_1) = y_1 \cap h(x_2) = y_2] = \frac{1}{|B|^2}$.*

In many of our applications the range B is the product of s "symbols" $B_1 \times \cdots \times B_s$. For a function $f\colon A \to B$ and $i \in [s]$, we use $f_i(x)$ to denote the ith coordinate of f.

When B is a product space, we will sometimes settle for a weaker notion of pairwise independence. Rather than requiring pairwise independence for the whole range, we only require pairwise independence in each coordinate:

DEFINITION 2.2. *Let $B = B_1 \times \cdots \times B_s$. A hash family $\mathcal{H}$ of functions $h\colon A \to B$ is* coordinatewise C-pairwise-independent *if, for all $i \in [s]$, any $x_1 \neq x_2 \in A$, and all $y_1, y_2 \in B_i$, we have $\Pr_{h\in\mathcal{H}}[h_i(x_1) = y_1 \cap h_i(x_2) = y_2] \leq \frac{C}{|B_i|^2}$.*

DEFINITION 2.3. *Let $B = B_1 \times \cdots \times B_s$. A function $f\colon A \to B$ is C-*uniform *if, for all $i \in [s]$ and all $y \in B_i$, $\Pr_{x\in A}[f_i(x) = y] \leq \frac{C}{|B_i|}$.*

DEFINITION 2.4. *For any function $f\colon B \to D$ and family $\mathcal{H}$ of functions $h\colon A \to B$, $f\circ\mathcal{H}$ denotes the family of $A \to D$ functions $\{g(x) = f(h(x)) \mid h \in \mathcal{H}\}$.*

CLAIM 2.5. *If $\mathcal{H}$ is pairwise-independent and f is C-uniform, then $f \circ \mathcal{H}$ is coordinatewise C^2-pairwise-independent.*

PROOF. Let $\mathcal{H}$ be $A \to B$ and f be $B \to D = D_1 \times \cdots \times D_s$. Then for any $i \in [s]$, any $x_1 \neq x_2 \in A$, and all $y_1, y_2 \in D_i$ we have:

$$\begin{aligned}
&\Pr_{h\in\mathcal{H}}[f_i(h(x_1)) = y_1 \cap f_i(h(x_2)) = y_2] \\
&= \sum_{z_1,z_2\in B} \Pr_{h\in\mathcal{H}}[h(x_1) = z_1 \cap h(x_2) = z_2 \cap f_i(z_1) = y_1 \cap f_i(z_2) = y_2] \\
&= \sum_{z_1,z_2\in B} \frac{1}{|B|^2} \Pr[f_i(z_1) = y_1 \cap f_i(z_2) = y_2] \\
&= \Pr_{z_1,z_2\in B}[f_i(z_1) = y_1 \cap f_i(z_2) = y_2] \\
&= \Pr_{z_1\in B}[f_i(z_1) = y_1] \Pr_{z_2\in B}[f_i(z_2) = y_2] \\
&\leq \frac{C^2}{|B_i|^2}
\end{aligned}$$

as desired. □

DEFINITION 2.6. *We say that a function $f\colon A \to B$ for $B = B_1 \times \cdots \times B_s$ is an* error-correcting code of distance d *if, for any two distinct $x_1, x_2 \in A$, $f(x_1)$ and $f(x_2)$ differ in at least d coordinates.*

We say that f is efficiently decodable *if we have an algorithm f^{-1} running in $\log^{O(1)}|B|$ time with $f^{-1}(y) = x$ for any $x \in A$ and $y \in B$ such that $f(x)$ and y differ in fewer than $d/2$ coordinates.*

For any prime $P \geq N$, the function family $\mathcal{H}_P : ax + b \pmod P$ for $a, b \in [P]$ is pairwise independent when viewed as a set of functions from $[N]$ to $[P]$.

LEMMA 2.7. *If f is an efficiently decodable error-correcting code with distance d, then so is $f \circ h$ for every $h \in \mathcal{H}_P$ with $a \neq P$.*

PROOF. Since $a \neq P$, there exists an a^{-1} modulo P, and we can efficiently compute it. Hence h is injective, so $f \circ h$ is an error-correcting code of distance d. Furthermore, $(f \circ h)^{-1}(x) = a^{-1}(f^{-1}(x) - b) \pmod P$ is efficiently computable. □

DEFINITION 2.8. *We say that a family $\mathcal{G}$ of functions $g\colon A \to B_1 \times \cdots \times B_s$ is a $(C, N, s, d)_q$-independent-code if $\mathcal{G}$ is coordinatewise C-pairwise independent, $q \leq |B_i| < 2q$ for all $i \in [s]$, $|A| \geq N$, and with probability at least $1-1/N$ over $g \in \mathcal{G}$ we have that g is efficiently decodable with distance d.*

If $f\colon A' \to B_1 \times \cdots \times B_s$ is C-uniform and efficiently decodable with distance d and $|B_i| \geq q$ for all i, then $f \circ \mathcal{H}_P$ is a $(C^2, N, s, d)_q$-independent code.

2.2 Two code constructions

We explicitly give two $(4, N, s, s-r)_q$-independent codes. Both are achievable for any parameters with $2N < q^r$ and $s < q/\log q$ (and the first code allows any $s < q$). We let P be a prime in $\{\frac{1}{2}q^r, \ldots, q^r\}$.

Reed-Solomon code

Let $q \geq s$. The Reed-Solomon code $f_{RS}\colon [q^r] \to [q]^s$ is defined for $f(x)$ by (i) interpreting x as an element of $\mathbb{F}_q^r$, (ii) defining $\chi_x \in \mathbb{F}_q[\xi]$ to be the rth degree polynomial with coefficients corresponding to x, and (iii) outputting $f(x) = (\chi_x(1), \ldots, \chi_x(s))$. It is well known to have distance $s - r$ and to be efficiently decodable [Jus76].

CLAIM 2.9. *Let $f : [P] \to [q]^s$ be the restriction of f_{RS} to $[P]$. Then f is 2-uniform.*

PROOF. We know that f_{RS} is 1-uniform. Since $P \geq q^r/2$, f is 2-uniform. □

Hence $\mathcal{G}_{RS} = f \circ \mathcal{H}_P$ is a $(4, N, s, s-r)_q$-independent code.

Chinese Remainder Theorem (CRT) code

Let $p_1, \ldots, p_s \in [q, 2q]$ be distinct primes; note that the asymptotic distribution of prime numbers implies $q/\log q = \Omega(s)$. Hence for any $x \in [N]$, any r of the residues mod $p_1, \ldots, p_s$ uniquely identify x. The CRT code $f_{CRT} \colon [P] \to [p_1] \times \ldots \times [p_s]$ is defined by taking the residues modulo each prime. It has distance $s - r$ and is efficiently decodable [GRS99].

CLAIM 2.10. *The CRT code f_{CRT} is 2-uniform.*

PROOF. Let $i \in [s]$. The projection of f_{CRT} onto its ith coordinate is one-to-one over a domain that is any consecutive sequence of p_i integers. Hence over the domain $[P]$, the ratio between the likelihood of the most common and the least common values in the range is $\frac{\lceil P/p_i \rceil}{\lfloor P/p_i \rfloor} \leq 2$. □

Hence $\mathcal{G}_{CRT} = f_{CRT} \circ \mathcal{H}_P$ is a $(4, N, s, s-r)_q$-independent code.

2.3 Collisions

LEMMA 2.11. *Suppose $g \colon A \to B_1 \times \ldots \times B_s$ is drawn from a $(4, N, s, s-r)_q$-independent code. Let $S, S' \subset A$. Define the set of "colliding" symbols*

$$C = \{(a, i) \mid a \in S, i \in [s], \exists a' \in S' \text{ s.t. } g_i(a) = g_i(a'), a \neq a'\}$$

With probability at least 7/8, $|C| \leq 32\,|S|\,|S'|\,s/q$.

PROOF. We observe that

$$\begin{aligned}
\mathrm{E}[|C|] &= \sum_{i \in [s]} \sum_{a \in S} \Pr[(a, i) \in C] \\
&\leq \sum_{i \in [s]} \sum_{a \in S} \sum_{\substack{a' \in S' \\ a' \neq a}} \Pr[g_i(a) = g_i(a')] \\
&= \sum_{i \in [s]} \sum_{a \in S} \sum_{\substack{a' \in S' \\ a' \neq a}} \sum_{z \in B_i} \Pr[g_i(a) = z \cap g_i(a') = z] \\
&\leq \sum_{i \in [s]} \sum_{a \in S} \sum_{\substack{a' \in S' \\ a' \neq a}} \sum_{z \in B_i} \frac{4}{|B_i|^2} \\
&\leq s\,|S|\,\left|S'\right|\,4/q
\end{aligned}$$

Hence, by Markov's inequality, $|C| \leq 32\,|S|\,|S'|\,s/q$ with probability at least 7/8. □

3. RECOVERY

In this section we present the measurement matrix A and the recovery algorithm. A graphical representation of the algorithm is presented in Appendix B. Let $O = \{o_1 \ldots o_k\}$ be a sequence of k features, and let $T = \{t_1 \ldots t_k\}$ be a sequence of (non-colliding) translations in x. Also, let μ be the noise vector, and let x' be the noisy image. Finally, we introduce (small) constants $\alpha, \beta, \gamma, \delta, \eta > 0$, whose values will be determined in the course of the analysis.

At the beginning, we impose a square grid G with $w' \times w'$ cells on the image x', such that $w' = w/\alpha$. The grid is shifted by a vector v chosen uniformly at random from $[w']^2$. Let S' be the set of cells that intersect or contain some object $t_i(o_i)$, and $S \subset S'$ be the set of cells that fully contain some object $t_i(o_i)$. Observe that a fixed object is fully contained in some cell with probability $(1 - w/w')^2 > 1 - 2\alpha$, since each axis of the grid intersects the object with probability w/w'. This implies that the expected number of cells in $S' - S$ is at most $2\alpha k$, which implies by Markov that $|S' - S| \leq 16\alpha k$ with probability 7/8. From now on, we will assume the latter event holds. Let $k' = |S'|$. We choose $\alpha > 0$ such that $k' \leq 2k$.

Measurements

Our measurement matrix A is defined by the following linear mapping. Let G denote the set of cells. Let $g \colon G \to B = B_1 \times \cdots \times B_s$ be drawn from a $(4, N, s, 4(3\delta + \beta)s)_q$-independent code (such as either $\mathcal{G}_{RS}$ or $\mathcal{G}_{CRT}$). Moreover, we require that $k/q \leq \eta$; such a code is achievable per Section 2.2 with $s = \Theta(\log_k N)$ as long as $k > C \log N$ for some constant C (such that both $q^{(1-3(3\delta+\beta))s} > k^{s/2} > 2N$ and $s < \log N / \log k \leq q/\log q$). For each $i = 1 \ldots s$, we define a $|B_i|$-dimensional vector z^i whose entries are elements in $\mathbb{R}^{w'^2}$, such that for any j

$$z^i_j = \sum_{g_i(c) = j} x'_c$$

That is, we "hash" all cells into $|B_i| \geq q$ buckets, and sum all cells hashed to the same bucket. The measurement vector $z = Ax'$ is now equal to a concatenation of vectors $z^1 \ldots z^s$. Note that the dimension of z is equal to $m = w'^2 \sum |B_i| = O(qs) = O(k \log_k N)$.

Recovery algorithm

The recovery algorithm starts by identifying the buckets that likely contain the cells from S, and labels them consistently (i.e., two buckets containing cells from S should receive the same label), allowing for a small fraction of errors. We then use the labels to identify the cells.

The algorithm is as follows. For a set of pairs X, let $F(X)$ denote $\{F(z^i_j) : (i, j) \in X\}$.

1. Identify $R = \{(i, j) : \|F(z^i_j)\|_\Gamma \geq T/2\}$ (that is, R_i contains the "heavy cells" of the measurement vector z).
2. Partition R into sets $R', R^1, \ldots, R^k$ such that $|R'| \leq \delta s k$, and such that for each $1 \leq l \leq k$ the diameter of $F(R^l)$ is at most $T/2$.
3. For each label $l = 1 \ldots k$, create a vector $u^l \in B$ such that for each $i = 1 \ldots s$, $u^l_i = j$ if $(i, j) \in R^l$ (if there are many such j, ties are broken arbitrarily), or $u^l_i = \perp$ (an arbitrary erasure symbol) if no such j exists.
4. For each label $l = 1 \ldots k$ apply the decoding algorithm[2] for g to u^l, obtaining a (possibly invalid) decoded cell d^l.

We analyze the algorithm by keeping track of the errors at each step.

[2]Technically, we replace each $\perp$ in u^l with an arbitrary j before running the decoding algorithm, since the decoding algorithms don't know about $\perp$.

Step 1 For any cell $c \in S$ and $i = 1 \dots s$, we say that i *preserves* c if $\|F(z^i_{g_i(c)}) - F(x_c)\|_\Gamma \leq T/24$ and $g_i(c') \neq g_i(c)$ for all other $c' \in S$. That is, there is no collision from the hashing process, and the total amount of distortion due to the noise μ is small. Let $P = \{(i, g_i(c)) : i \text{ preserves } c\}$. Note that $P \subset R$. We show that P is large and that most of R is in P.

LEMMA 3.1. *With probability at least* 7/8,

$$|P| \geq (1 - \beta)sk.$$

PROOF. Consider any pair $(c, i) \in S \times \{1 \dots s\}$, and let $j = g_i(c)$. If i does not preserve c, it must be because either (i) there is another cell $c' \in S'$, $c' \neq c$ such that $g_i(c') = j$, or because (ii) the total noise affecting z^i_j, equal to $F(\mu^i_j) \leq \sum_{g_i(c)=j} F(\mu_c)$, has norm at least $T/24$.

By Lemma 2.11 with probability at least 7/8 the number of pairs affected by (i) is at most $32ks|S'|/q$. The event (ii) is determined by the noise vector μ. However, for each i, there are at most $\frac{\|\mu\|_F}{T/24} \leq 24\gamma k$ additional cells $c \in S$ that are not preserved under i due to this reason.

Altogether, the total number of pairs (c, i) such that c is not preserved by i is at most

$$32sk\left|S'\right|/q + 24\gamma sk = [32\eta(1 + 16\alpha) + 24\gamma]sk = \beta sk$$

as desired. □

LEMMA 3.2. *With probability at least* 3/4,

$$|R \setminus P| \leq \delta sk.$$

PROOF. Any element (i, j) of $R \setminus P$ ("heavy but not preserved") must belong to one of the following three categories:

1. $j = g_i(c)$ for $c \in S$ such that c is not preserved by i. By the previous lemma, there are at most βsk such pairs (c, i) with probability at least 7/8.
2. $j = g_i(c)$ for some cell $c \in S' \setminus S$. There are at most $16\alpha sk$ such pairs (c, i), with probability at least 7/8.
3. The vector $F(\mu^i_j) = \sum_{g_i(c)=j} F(\mu_c)$ has norm at least $T/2$. There are at most $2\gamma sk$ such pairs (i, j).

This implies that with probability at least 3/4 the total number of pairs $(i, j) \in R \setminus P$ is at most

$$(\beta + 16\alpha + 2\gamma)sk = \delta sk$$

as desired. □

Step 2 Observe that the elements of $F(P)$ can be clustered into k clusters of diameter $T/12$. Thus, by the previous lemma, there is a k-clustering of all but δsk elements of $F(R)$ such that the diameter of each cluster is at most $T/12$. We now apply a 6-approximation algorithm for this problem, finding a k-clustering of $F(R)$ such that the diameter of each cluster is at most $T/2$. Such an approximation algorithm follows immediately from the 3-approximation algorithm for k-center with outliers given in [CKMN01].

Step 3 Consider cells $c, c' \in S$ such that c is preserved by i and c' is preserved by i'. If $F(z^i_{g_i(c)})$ and $F(z^{i'}_{g_i(c')})$ belong to the same cluster, then it must be the case that $c = c'$, since otherwise the distance between them would be at least $T - 2T/24 > T/2$. In other words, for each l, if $u^l \subset P \cap R^l$ contains at least one element of P, then all the elements of u^l are "derived" from the same cell.

LEMMA 3.3. *With probability at least 3/4,* $u^1 \dots u^k$ *contain a total of at most* $2\delta sk$ *errors and* $(\delta + \beta)sk$ *erasures* $(i, l$ *such that* $u^l_i = \perp)$.

PROOF. Let $R'' = R \setminus R' = R^1 \cup \dots \cup R^k$. Let $P' = P \cap R'$, and $P'' = P \cap R''$.

Note that $|P'| \leq |R'| \leq \delta sk$. Each error in $u^1 \dots u^k$ corresponds to a unique element of $R'' \setminus P''$, and we have

$$|R'' \setminus P''| \leq |R'' \setminus P| + |P \setminus P''| \leq |R \setminus P| + |P'| \leq \delta sk + \delta sk = 2\delta sk.$$

Additionally, $u^1 \dots u^k$ contain at least P'' elements total, and so the number of erasures is at most $sk - |P''| = sk - |P| + |P'| \leq \beta sk + \delta sk$, where we use $|P| \geq (1 - \beta)sk$ from Lemma 3.1. □

Step 4 We can replace erasures by errors, and conclude that $u^1 \dots u^k$ have a total of at most $(3\delta + \beta)sk$ errors. It follows that at least $k/2$ of them have at most $2(3\delta + \beta)s$ errors, and therefore can be decoded. Therefore, the set $D = \{d^1 \dots d^k\}$ contains at least $k/2$ elements of S.

THEOREM 3.4. *Assume* $k \geq C \log N$ *for some constant* C*, a signal* x *with* k *objects, and a noise vector* μ*, all subject to the constraints delineated in the Model description of Section 1. There is a distribution over random binary* $m \times N$ *matrices* A*,* $m = O(k \log_k N)$*, and an associated recovery algorithm with the following property. Suppose that the algorithm is given* Ax' *for* $x' = x + \mu$*. Then the algorithm recovers (with probability at least* 3/4*) a set* D *of* k *cells, such that at least* $k/2$ *of the cells fully containing an object are included in* D*. Moreover, the matrix has column sparsity* $O(\log_k N)$.

4. EXPERIMENTAL RESULTS

Our theoretical model is motivated by a task in satellite navigation formally known as *attitude determination*. An attitude is a triple (roll, pitch, yaw) that describes an orientation relative to the fixed night sky. Satellites compute their attitude by taking a picture of the night sky, extracting stars, and matching patterns of stars to an onboard database. We implement our algorithm[3], referred to as *Star Recovery with Geometric Features*, or SRGF, to perform star extraction in very small space and time, conforming to the physical constraints of computing on a satellite.

One of the main energy costs in conventional imaging is incurred by analog to digital conversion. Several (non-satellite) compressive imagers have been built that use compressive sensing to reduce the number of measurements that need to be digitized [RGL+10, MJS+10]. However, these compressive imagers use dense matrices, which are impractical when the the signal is weak compared to the noise.

4.1 Star Cameras

In modern satellites, the entire task of attitude determination is encapsulated in a *star tracker* or *star camera*. [Lie02] has an excellent tutorial on star camera fundamentals. To acquire attitude, such cameras run the following sequence of steps:

1. Acquire the incoming light as an n-by-n pixel analog signal, which we call the *preimage*.

[3] Source code for the experiments is available from http://web.mit.edu/rishig/papers/local-geo/

2. Digitize each of the n^2 pixels, to obtain the *digital image*.
3. Locate a set S of star like objects in the digital image.
4. Match patterns formed by 3-5 element subsets of S to an onboard database. This step is commonly known as *star identification*.
5. Recover spacecraft attitude by using the database coordinates of the identified stars.

Step 2 consumes a significant amount of power due to the number of analog to digital conversions, and Step 3 requires significant processing and memory resources due to the large number of measurements that must be processed. Our compressive sensing solution fits in after Step 1, by compressing the n^2 measurements of the preimage into $m \ll n^2$ measurements that then get digitized, reducing the cost of Steps 2 and 3.

4.2 Implementation Details

We fix $n = 800$ ($N = 640000$) for all experiments. We expose the camera to a .08 radian by .08 radian (4.6 by 4.6 degree) patch of the sky, which means that a typical image will have 3-10 objects (bright stars, 10%-ile to 90%-ile) and 50 to 150 stars that act as background noise. To generate test images, we use the Smithsonian Astrophysical Observatory (SAO) Star Catalog [Smi], randomly select patches from the sky (subject to the constraints of Appendix A), and convolve the stars with a gaussian of radius .5 pixels.[4] These are typical values for modern star cameras.

Given the physical constraints of the proposed architecture, we set the column sparsity s to the smallest value our algorithm will tolerate: 2. We set the object size w to 3, as the vast majority of a star's mass falls within a 3-by-3 box. The rest of this section follows the structure of Section 3.

Measurements

We assume the stars are randomly (rather than adversarially) distributed within the image x', and use that to simplify construction of the measurement vector $z = Ax'$. In particular, we no longer need to compose our error correcting code with a hash family such as that from Definition 2.1. If p_1 and p_2 are the primes used to construct the CRT code in Section 2.2, we find primes[5] p'_i such that $p'_i \approx \sqrt{p_i}$. Notating x' as an n-by-n image, we define z^i to be a p'_i-by-p'_i image with

$$z^i[j_1, j_2] = \sum_{\substack{c_1 \equiv j_1 \pmod{p'_i} \\ c_2 \equiv j_2 \pmod{p'_i}}} x'[c_1, c_2].$$

Since the stars were assumed to be randomly distributed in x', they are randomly distributed within each z^i as well. We define the "measurement vector" z to be a 1-dimensional representation of the pair (z^1, z^2), and construct our measurement matrix A accordingly. However, from here on we will think of each z^i as a 2-dimensional image.

[4]Stars are point sources of light. Stars "cover" more than one pixel in the preimage only because the lens of the camera convolves the image with a function approximated by a gaussian. See [Lie02] for more details.

[5]Note that we don't literally need p'_i and p'_2 to be prime, as long as they are relatively prime, since the Chinese Remainder Theorem from Step 4 of the Recovery applies in this case as well. We will use the word *prime* to refer to p'_1 and p'_2 even when they are just relatively prime.

Recovery Algorithm

We follow the presentation of the corresponding paragraph in Section 3. The recovery algorithm is:

1. In each z^i, we identify ten 3-by-3 cells with high mass such that no two cells collide in more than 4 pixels. We allow the 3-by-3 cells to wrap around the edges of z^i.

2. In place of the k-centering algorithm, we greedily choose up to 8 pairs of cells (c^1, c^2) from (z^1, z^2) such that the feature vectors $F(c^1)$ and $F(c^2)$ are close. In our case, $F(c)$ is a triple of values: the mass of c, along with two coordinates for the centroid of c.

3. No longer needed. Each pair from Step 2 corresponds to one of the u^l.

4. In place of the error correcting code, we simply apply the Chinese Remainder Theorem in each dimension to recover a 3-by-3 region of the original image for each pair from Step 2.

Star Identification

We now have an image x'' with up to 8 decoded regions, or "star candidates". We input x'' into a simplified version of the star identification algorithm presented in [MSBJ04]. The algorithm works as follows: We first extract a subset SAO$'$ of the Star Catalog SAO, by taking the 10 largest stars in every ball of radius 4.6 degrees. SAO$'$ has 17100 stars, compared to 259000 stars in SAO. We then build a data structure DS that can find a triangle of stars in SAO$'$ given three edge lengths and an error tolerance. We query DS with subsets of three stars from x'' until we find a match. This portion of the algorithm can be replaced by any regular star identification algorithm that has some tolerance for incorrect data, such as [MA08], or several in the survey [SM09].

4.3 Results and Observations

There are many sources of noise in real star cameras; we mimicked the major ones by having photons hit the camera in a random poisson process, and by adding gaussian noise to Ax'. We report our results as a function of the standard deviation of the added gaussian noise.

We ran SRGF as well as Sequential Sparse Matching Pursuit (SSMP) on 159 randomly generated images of the sky (subject to the discussion in Appendix A). SSMP is a near-linear (in N) time procedure with recovery quality on par with other known methods [BI09]. We used the same measurement matrix A for both algorithms. Both SRGF and SSMP only attempt to recover portions of the signal x'; we then ran the same star identification algorithm of Section 4.2 on the outputs of both of them. We note that SSMP is not designed for this level of sparsity, and works better for $d > 2$. The results of the experiments are in Figure 1.

The first observation we make is that SRGF works very well down to an almost minimal number of measurements. The product $p'_1 p'_2$ has to be greater than 800, and the minimal set of primes is 26 and 31. As the number of measurements increases, SSMP catches up and surpasses SRGF, but we note that running SSMP (implemented in C) takes 2.4 seconds per trial on a 2.3 GHz laptop, while SRGF (implemented in Octave/Matlab) takes .03 seconds per trial. Computation power on a satellite is substantially lower than

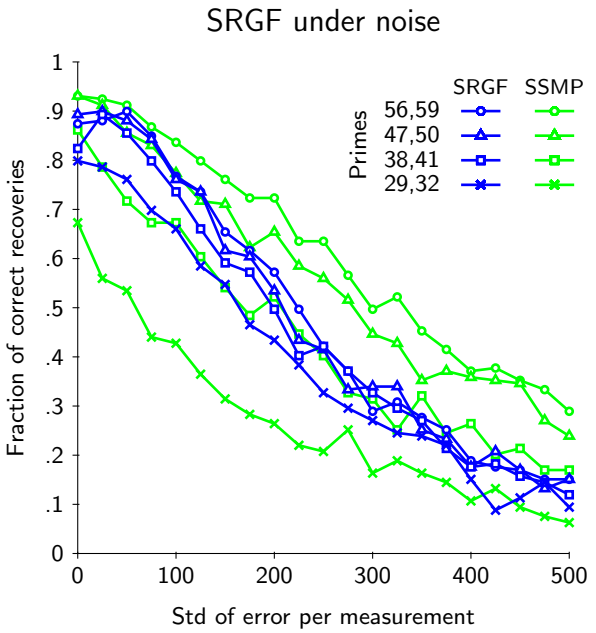


Figure 1: Experimental results. Each point on the figure is computed using the same 159 underlying images.

that of a low end laptop, and given that the entire acquisition has to happen in .1 to 2 seconds, it seems unlikely that any algorithm linear or near linear in N is going to be practical. Finally, we note that all plot lines in the results could be improved by a more thorough implementation of a star identification algorithm.

Acknowledgements

The authors would like to thank Tye Brady and Ben Lane from Draper Lab for numerous conversations and help with the data, and the anonymous reviewers for helping clarify the presentation.

This research has been supported in part by a David and Lucille Packard Fellowship, MADALGO (Center for Massive Data Algorithmics, funded by the Danish National Research Association) and NSF grant CCF-0728645. R. Gupta has been supported in part by a Draper Laboratory Fellowship. E. Price has been supported in part by an NSF Graduate Research Fellowship.

5. REFERENCES

[BGI+08] R. Berinde, A. Gilbert, P. Indyk, H. Karloff, and M. Strauss. Combining geometry and combinatorics: a unified approach to sparse signal recovery. *Allerton*, 2008.

[BI09] R. Berinde and P. Indyk. Sequential sparse matching pursuit. *Allerton*, 2009.

[CCF02] M. Charikar, K. Chen, and M. FarachColton. Finding frequent items in data streams. *ICALP*, 2002.

[CKMN01] M. Charikar, S. Khuller, D.M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. *SODA*, 2001.

[CM04] G. Cormode and S. Muthukrishnan. Improved data stream summaries: The count-min sketch and its applications. *Latin*, 2004.

[CRT06] E. J. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Comm. Pure Appl. Math.*, 59(8):1208–1223, 2006.

[DDT+08] M. Duarte, M. Davenport, D. Takhar, J. Laska, T. Sun, K. Kelly, and R. Baraniuk. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 2008.

[DIPW10] K. Do Ba, P. Indyk, E. Price, and D. Woodruff. Lower bounds for sparse recovery. *SODA*, 2010.

[Don06] D. L. Donoho. Compressed Sensing. *IEEE Trans. Info. Theory*, 52(4):1289–1306, 2006.

[FPRU10] S. Foucart, A. Pajor, H. Rauhut, and T. Ullrich. The gelfand widths of lp-balls for $0 < p \leq 1$. *preprint*, 2010.

[FTF06] R. Fergus, A. Torralba, and W. T. Freeman. Random lens imaging. *MIT CSAIL-TR-2006-058*, 2006.

[GI10] A. Gilbert and P. Indyk. Sparse recovery using sparse matrices. *Proceedings of IEEE*, 2010.

[GRS99] O. Goldreich, D. Ron, and M. Sudan. Chinese remaindering with errors. *STOC*, pages 225–234, 1999.

[HL07] G. C. Holst and T. S. Lomheim. *CMOS/CCD Sensors and Camera Systems*. JCD Publishing and SPIE Press, 2007.

[Ind07] P. Indyk. Sketching, streaming and sublinear-space algorithms. *Graduate course notes, available at http://stellar.mit.edu/S/course/6/fa07/6.895/*, 2007.

[Jus76] J. Justesen. On the complexity of decoding Reed-Solomon codes (Corresp.). *Information Theory, IEEE Transactions on*, 22(2):237–238, 1976.

[Lie02] C. C. Liebe. Accuracy performance of star trackers — a tutorial. *IEEE Transactions On Aerospace and Electronic Systems*, 38:587–599, 2002.

[MA08] W. Makowiecki and W. Alda. New sky pattern recognition algorithm. *ICCS*, Part 1, LNCS 5101:749–758, 2008.

[Mei03] James D. Meindl. Beyond moore's law: The interconnect era. *Computing in Science and Engineering*, 5:20–24, 2003.

[MJS+10] V Majidzadeh, L Jacques, A Schmid, P Vandergheynst, and Y Leblebici. A (256x256) Pixel 76.7mW CMOS Imager/Compressor Based on Real-Time In-Pixel Compressive Sensing. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010.

[MSBJ04] D. Mortari, M. Samaan, C. Bruccoleri, and J. Junkins. The pyramid star identification technique. *Navigation*, 51(3):171–183, 2004.

[Mut05] S. Muthukrishnan. Data streams: Algorithms and applications). *Foundations and Trends in Theoretical Computer Science*, 2005.

[Nac10] M. Nachin. Lower bounds on the column sparsity of sparse recovery matrices. *6.UAP: MIT Undergraduate Thesis*, 2010.

[RGL+10] R. Robucci, J. Gray, L. K. Liu, J. Romberg, and P. Hasler. Compressive sensing on a cmos separable-transform image sensor. *Proceedings of the IEEE*, 2010.

[Rom09] J. Romberg. Compressive sampling by random convolution. *SIAM Journal on Imaging Science*, 2009.

[SM09] B. B. Spratling IV and D. Mortari. A survey on star identification algorithms. *Algorithms*, 2:93–107, 2009.

[Smi] Smithsonian astrophysical observatory star catalog. *Available at http://heasarc.gsfc.nasa.gov/W3Browse/star-catalog/sao.html*.

[TAN10] V. Treeaporn, A. Ashok, and M. A. Neifeld. Increased field of view through optical multiplexing. *Optics Express*, 18(21), 2010.

[UGN+09] S. Uttam, A. Goodman, M. A. Neifeld, C. Kim, R. John, J. Kim, and D. Brady. Optically multiplexed imaging with superposition space tracking. *Optics Express*, 17(3), 2009.

APPENDIX

A. CELESTIAL INTUITIONS

Spatial distribution of stars

We assume everywhere that stars live on a rectangular interval, rather than a sphere. We index the rectangle by right ascension (longitude) $\alpha \in [-\pi, \pi]$ and declination (latitude) $\delta \in [-\pi/2, \pi/2]$; in particular, $\delta = 0$ is the equator, and $\delta = \pi/2$ is the north pole. So that the approximation makes sense, we ignore the portion of the sky where $|\delta| > \pi/2 - \pi/8$ (the dashed blue lines in Figure 2). We also assume without loss of generality that the camera is axis-aligned.

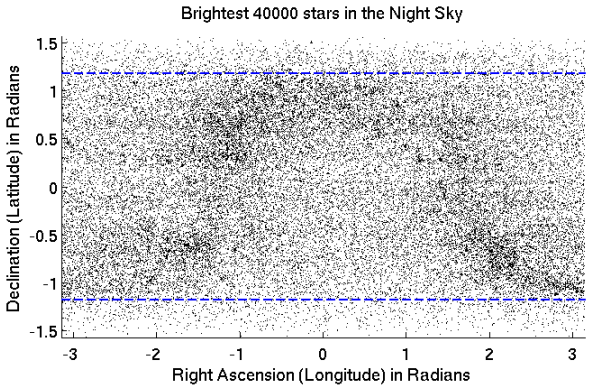


Figure 2: The night sky, as seen from Earth's orbit. We test our algorithm on the areas between the dashed blue lines, where the Mercator projection does not skew the distribution of stars too much.

Figure 2 shows the distribution of stars across the night sky. The dense omega-shaped region is the central disk of our galaxy. In our pictures, if the median total star mass is 1, the 10%-ile mass is .6, and the 90%-ile mass is 2.25.

Mass distribution of stars

We define the *mass* of a star to be the number of photons from the star that hit our camera. The mass of the j^{th} brightest star in the sky is $\Theta(j^{-1.17})$. Figure 3 gives some intuition for the mass of the biggest stars in our pictures relative to the amounts of noise. There are usually 50 to 150 total stars in a given preimage.

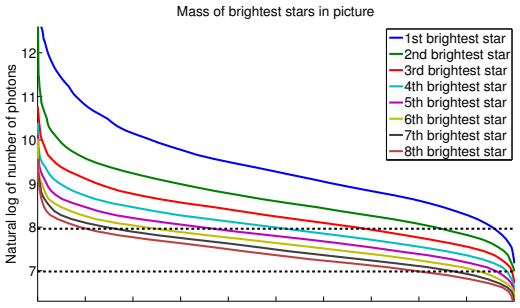
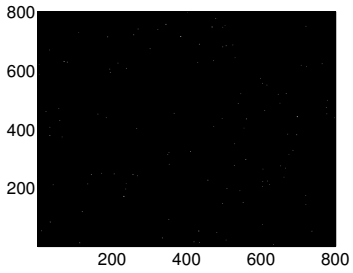
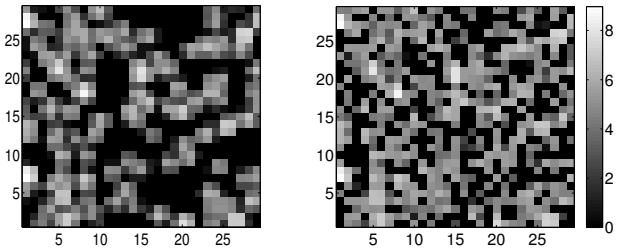


Figure 3: The mass of the brightest stars in our pictures. The x-axis ranges from 100 to 0, and for instance if (x, y) falls on the green line, it means that in x% of the pictures, the ln(mass) of the 2nd biggest star is greater than y. The two dashed lines are the expected gaussian L_1 noise over a given star when there is a standard deviation of 150 and 400 photons of noise per pixel.

Sample Images



(a) Underlying signal x (zoom in).



(b) $p' = 29$, No Noise. (c) $p' = 29$, Noise = 150

Figure 4: log(mass) in sample images from one representative section of the sky. The legend on the right applies to all three images. All images have poisson noise applied, though (a) and (b) have no additional gaussian noise. We cut off pixels with values below 0 before taking the log in (c).

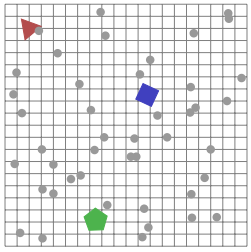
Figure 4 has a few pictures to help build intuition about star density and what stars look like. It is generally possible to see where the biggest stars are located, though some fraction of them get occluded by small stars or noise.

B. THE ALGORITHM IN PICTURES

Measurements

We first compute Ax' from the received signal x'. An element of the $(4, N, s, s-r)_q$-independent code $\mathcal{G}_{CRT} = f_{CRT} \circ \mathcal{H}_P$ is depicted below.

Received signal $x' = x + \mu$



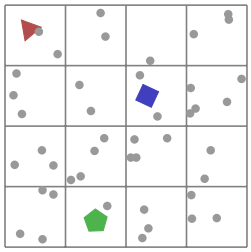
n

n pixels

There are a total of $N = n^2$ pixels.
The goal is to recover the k objects (colored polygons).
Each object fits in a $w \times w$ pixel box.

w

w



n/w'

n/w' cells

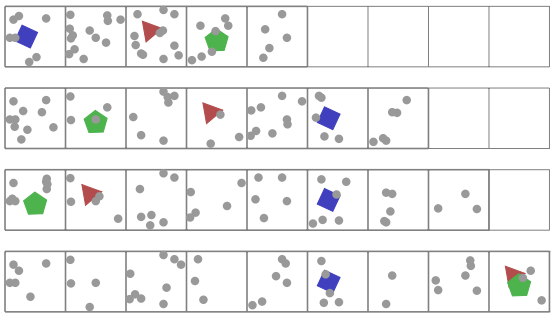
Impose a (randomly shifted) grid G of cells of width $w' = w/\alpha$.
For clarity we no longer draw the pixel grid.

Apply a pairwise independent hash function such as
$\mathcal{H}_P : x \to ax + b \pmod P$ to a numbering of the cells.

1

$P > (n/w')^2$ cells

Measured signal $z = Ax'$



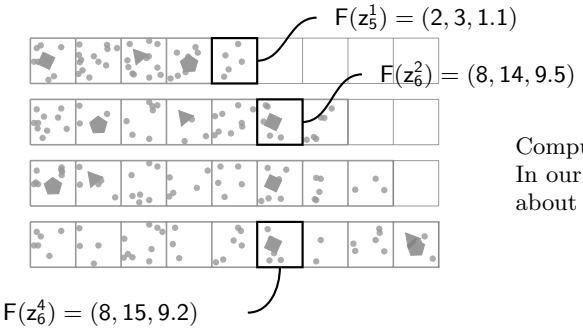
s

$|B_i| \geq q$ buckets in row z^i

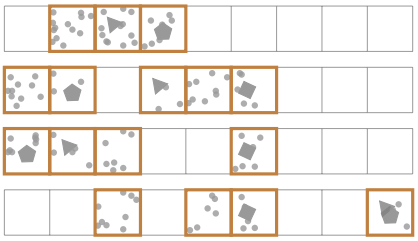
Apply an error correcting code f that maps each cell onto exactly one bucket in every row. Sum the cells mapping onto each bucket. The code shown to the left is f_{CRT}, where each cell is mapped to its index modulo various (relatively) prime numbers.

Recovery

We now recover from the measurements Ax'.



Compute the feature vector $F(z_j^i)$ of each bucket.
In our experiments, the feature vector contains information about mass and centroid.

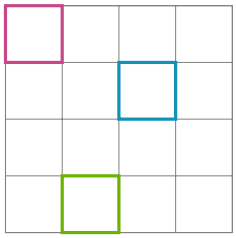


Set $R = \{(i, j) : \|F(z_j^i)\|_\Gamma \text{ is large}\}$. Discard buckets not in R.

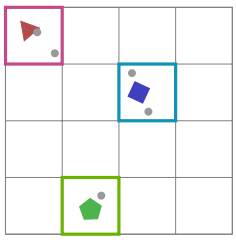
$R =$ 

Cluster $F(R) = \{F(z_j^i) : (i, j) \in R\}$ into k clusters (with outliers). This induces a partition $R', R^1 \ldots R^k$ of R, with $F(R^l)$ equal to the l-th cluster.

$R^1 =$  $R^2 =$  $R^3 =$ 



Decode each R^l to obtain a cell d^l in the original image.



Though we don't elaborate in the text, a simple min or median process can be used to obtain an approximation for the contents of each d^l.