# Assignment 1: CS 754


| Krushnakant Bhattad | Devansh Jain |
|---|---|
| 190100036 | 190100044 |


February 9, 2021

## Contents

# Question 1

**(1) It appears that the upper bound...**:
It only " appears" that the upper bound is reduced as s increases. We are actually missing the overall picture.

First of all, it can be easily seen from the definition that, $\delta_{2s}$ is an increasing function of s. (Since aLL $2s-$sparse signals are also $2(s+1)-$sparse signals)

Next, the constants $C_1$ and $C_2$ in the error term are, in fact, increasing functions of just $\delta_{2s}$; and thus, increasing functions of s.

Moreover, while $1/\sqrt{s}$ does decrease, its decreases at a very slow rate.

Thus, if we look at the overall picture, the apparent discrepancy about reduction of upper bound does not arise.

**(2) Error bound independent of m...**:
The error function, while on the surface looks like it is independent of $m$, we can see that it does depend on $s$.

The condition for this $s$-dependent error bound to increase is that the matrix $A$ obey RIP of order $2s$, with $\delta_{2s} < \sqrt{2} - 1$.

Satisfiability of this RIP is subject to the matrix $A$, and hence also to its dimensions- $m \times n$.

Thus, though not directly, the error bound does have a connection with $m$- it is not really independent f $m$.

**(3) Theorem 3A: Change in condition-** $\delta_{2s} < 0.3$
From the definition, $\delta_{2s}$ is the smallest constant so that for all $s$-sparse signals $\theta$, we should have:

$$(1 - \delta_{2s})||\theta||^2 \leq ||\mathbf{A}\theta||^2 \leq (1 + \delta_{2s})||\theta||^2$$

We can see that having $\delta_{2s} < 0.3$ is a more difficult condition than having $\delta_{2s} < 0.41$. In fact, if $\delta_{2s} < 0.3$, anyway $\delta_{2s} < 0.41$.

So, every matrix $\mathbf{A}$ that satisfies conditions of Theorem 3A, it will always satisfy conditions of Theorem 3. Vice-versa is not the case.

Thus, Theorem 3 is more useful.

**(4) Set $\epsilon = 0$ in BP...**
As clearly given in the statement, $\epsilon$ is an upper bound on the magnitude of the noise vector $\eta$.
Setting the upper bound to 0 will actually mean that we are considering there is no noise, which is absurd because we actually know $\eta$ is non-zero.

To see this through another point of view, suppose that we go on and solve the BP problem with $\epsilon = 0$; which basically reduces to finding the solution $\theta^*$ to $\mathbf{y} = \mathbf{\Phi\Psi}\theta$.

However, we know that for the actual $\theta$; if we consider our corrupted $\mathbf{y}$ then $\mathbf{y} \neq \mathbf{\Phi\Psi}\theta$ but actually $\mathbf{y} \neq \mathbf{\Phi\Psi}\theta + \eta$. Thus, the problem will most definitely give a wrong solution (well we are solving for the wrong problem in the first place), which may as well be far away from the right answer.

If we account for $\epsilon$ however, we increase the probability of being closer to the real solution; since now we are solving for the correct problem.

# Question 2

### Instructions for running the code:

- After extracting submitted file, look for a directory named `q2`, and `cd` (change directory) to it.
- File `q2.m` contains the main code which uses the OMP function defined in file `opm.m`.
- Run the file `q2.m`. The results can be found in `./results/`

### Extracting the frames from the video

```
F = zeros(H,W,T,'double');
for i=1:T
    F(:,:,i) = rgb2gray(video.frames(i).cdata(x_min:x_max, y_min:y_max, :));
end
```

### Generating Random Code Pattern

```
C = randi([0, 1], H, W, T, 'double');
```

### Calculating the Coded snapshot and saving it

```
E = sum(C.*F, 3) + noise_std*randn(H,W);
figure;
imwrite(cast(E/T, 'uint8'), sprintf('results/%s_%i_coded_snapshot.jpg',name,T));
```

### Generating the Ψ - 3D DCT basis of $(u = 8, v = 8, w = T)$

```
D1 = dctmtx(8);
D2 = kron(D1, D1);
psi = kron(D2, dctmtx(T));
```

### Initializing reconstruction image and the count of patches per pixel

```
R = zeros(H, W, T, 'double');
avg_mat = zeros(H, W, 'double');
```

### Non-overlapping 8x8 patches

```
% for i=1:H/8
%     for j=1:W/8
%         y = reshape(E(8*(i-1)+1:8*i,8*(j-1)+1:8*j), [8*8 1]);
%         phi = zeros(8*8, 8*8*T, 'double');
%         for k=1:T
%             phi(:,8*8*(k-1)+1:8*8*k) = diag(reshape(C(i:i+7,j:j+7,k), [8*8 1]));
%         end
%         x = omp(phi*psi, y, noise_std^2);
%         R(8*(i-1)+1:8*i,8*(j-1)+1:8*j,:) = reshape(psi*x, [8 8 T]);
%     end
% end
```

### Reconstruction of all possible patches

```
for i=1:H-7
    for j=1:W-7
        y = reshape(E(i:i+7,j:j+7), [8*8 1]);

        phi = zeros(8*8, 8*8*T, 'double');
        for k=1:T
            phi(:,8*8*(k-1)+1:8*8*k) = diag(reshape(C(i:i+7,j:j+7,k), [8*8 1]));
        end

        x = omp(phi*psi, y, 9*8*8*noise_std^2);
        R(i:i+7,j:j+7,:) = R(i:i+7,j:j+7,:) + reshape(psi*x, [8 8 T]);
        avg_mat(i:i+7,j:j+7) = avg_mat(i:i+7,j:j+7) + ones(8,8);
        i, j % Prints the coordinates, to check for speed and debugging
    end
end
```

### Averaging out each pixel and saving

```
for i=1:T
    R(:,:,i) = R(:,:,i)./avg_mat(:,:);
    figure;
    imshow(cast([R(:,:,i), F(:,:,i)], 'uint8'));
    imwrite(cast([R(:,:,i), F(:,:,i)], 'uint8'), sprintf('results/%s_%i_%i.png',name,T,i));
    fprintf('RMSE for frame %i : %f\n',i,\
        norm(R(:,:,i)-F(:,:,i), 'fro')^2/norm(F(:,:,i), 'fro')^2);
end
fprintf('RMSE of video sequence : %f\n', \
    norm(reshape(R(:,:,:)-F(:,:,:), [H*W*T 1]))^2/norm(reshape(F(:,:,:), [H*W*T 1]))^2);
```

### Orthogonal Matching Pursuit Algorithm

```
function theta = omp(A, y, e)
    [N, K] = size(A); % N:dim of signal, K:#atoms in dictionary

    theta = zeros(K,1);      % coefficient (output)
    r = y;                   % residual of y
    T = [];                  % support set
    i = 0;                   % iteration
    A_omega = [];            % Sub-matrix of A containing columns which lie in the support set

    while(i < N && norm(r)^2 > e)
        i = i + 1;
        x_tmp = zeros(K,1);
        indices = setdiff(1:K, T); % iterate all columns except for the chosen ones
        for ind=indices
            x_tmp(ind) = A(:,ind)' * r / norm(A(:,ind)); % sol of min ||a'x-b||
```

```
        end
        [~,j] = max(abs(x_tmp)); % Choose the next column
        T = [T j];
        A_omega = [A_omega A(:,j)];
        theta_s = pinv(A_omega) * y; % Using pseudo-inverse of A_omega
        r = y - A_omega * theta_s;
    end

    for j=1:i
        theta(T(j)) = theta_s(j);
    end
end
```

## (c)

$E_u = \sum_{t=1}^{T} C_t \cdot F_t$ is the coded snapshot (measured signal).
This can be represented as $\boldsymbol{Ax} = \boldsymbol{b}$ where $\boldsymbol{b}$ is the vectorized form of $E_u$ of size (H*W)x1, $\boldsymbol{x}$ is the vectorize form of the video sequence $F_t$ from $t = 1$ to $t = T$ and $\boldsymbol{A}$ is the block diagonal matrix of size (H*W)x(H*W*T) formed by concatenating (H*W)x(H*W) diagonal matrices containing elements of $C_t$

## (d)

We follow similar pattern of $\boldsymbol{A}$ and $\boldsymbol{b}$ with H=8 and W=8.

$\boldsymbol{b} = \boldsymbol{Ax} + \boldsymbol{\eta}$, where $\boldsymbol{\eta}_i$ is Gaussian noise with mean 0 and standard deviation 2.
We find the OMP error bound in the similar way to which we found $\epsilon$ on slide 75 of CS_Theory slides (example related to Theorem 3).
If for each i=1 to m (here, H*W), $\boldsymbol{\eta}_i$ N(0,$\sigma^2$), with known $\sigma$ (here, 2).
Then, the squared magnitude of the vector $\boldsymbol{\eta}$ is a chi-square random variable.
Hence with very high probability, the magnitude of $\boldsymbol{\eta}$ will lie within 3 standard deviations from the mean, i.e. set $\epsilon \geq 9m\sigma^2$.
Thus, the required error bound dependent on the noise is $9 * (8 * 8) * (2^2)$.

### Issues with reconstruction result

Output seems to be clear enough to see motion between frames but it is not as good as the sample output provided.
We tried to debug the code but couldn't find the issue.
The implementation of OMP seems fine as even with using a thrid party OMP tool, similar results were produced.
One possible reason could be with $\boldsymbol{\Psi}$. We are unaware on how to create a 3D DCT basis matrix. We used `kron` and `dctmtx` but couldn't verify if it is correct or not.
On close observation, we saw that Relative Mean Squared Error of the last frame in all cases was higher than other frames and it was surprisingly dark (with lower intensity values), approximately 50-60% of the original intensity.
We think it should be a trivial error like possible shifting of indices somewhere or missing some normalization.

### Reporting Relative Mean Squared Error

- Video - `cars.avi`, T = 3
  RMSE = 0.038390
- Video - `cars.avi`, T = 5
  RMSE = 0.059257
- Video - `cars.avi`, T = 7
  RMSE = 0.084849
- Video - `flame.avi`, T = 5
  RMSE = 0.023120

# Question 3

All rows of $\mathbf{\Phi}$ are unit vectors in $R^n$ (given that it is unit normalized).
Take any row $\boldsymbol{g} = \mathbf{\Phi}_i$, it can be represented in terms of columns of $\mathbf{\Psi}$ (it is orthonormal basis matrix).

$\boldsymbol{g} = \sum_{k=0}^{n-1} a_k \mathbf{\Psi}_k \quad$ where $\sum_{k=0}^{n-1} |a_k|^2 = 1$

$\max_{k\in\{0,1,...,n-1\}} |\boldsymbol{g}^t \mathbf{\Psi_k}| = \max_{k\in\{0,1,...,n-1\}} |a_k|$

($\mathbf{\Psi}$ is orthonormal, so all $\mathbf{\Psi_k}$'s are orthogonal).

Let us assume that $\max_{k\in\{0,1,...,n-1\}} |a_k| < \frac{1}{\sqrt{n}}$

$|a_k| < \frac{1}{\sqrt{n}} \ \forall k \in \{0, 1, ..., n-1\}$

$|a_k|^2 < \frac{1}{n} \ \forall k \in \{0, 1, ..., n-1\}$

$\sum_{k=0}^{n-1} |a_k|^2 < 1 \implies ||\boldsymbol{g}||_2 < 1$

But this contradicts the fact that $\boldsymbol{g}$ is a unit vector.

Therefore, $\max_{k\in\{0,1,...,n-1\}} |a_k| \geq \frac{1}{\sqrt{n}}$

We can trivially see that $|a_k| \leq 1$ as $\sum_{k=0}^{n-1} |a_k|^2 = 1$

Thus, , $\max_{k\in\{0,1,...,n-1\}} |a_k| \leq 1$

For any arbitrary row of $\mathbf{\Phi}$ (say, $\boldsymbol{g}$), $\max_{k\in\{0,1,...,n-1\}} |\boldsymbol{g}^t \mathbf{\Psi_k}|$ lies in $\left[\frac{1}{\sqrt{n}}, 1\right]$.

Thus, $\max_{i\in\{0,1,...,m-1\}, j\in\{0,1,...,n-1\}} |\mathbf{\Phi^i}^t \mathbf{\Psi_j}|$ lies in $\left[\frac{1}{\sqrt{n}}, 1\right]$.

And, $\mu(\mathbf{\Phi}, \mathbf{\Psi}) = \sqrt{n} \cdot \max_{i\in\{0,1,...,m-1\}, j\in\{0,1,...,n-1\}} |\mathbf{\Phi^i}^t \mathbf{\Psi_j}|$ lies in $[1, \sqrt{n}]$.

Hence, the minimal value of coherence is 1.
To add to that - For coherence to be 1, the corresponding row of $\mathbf{\Phi}$ (say, $\boldsymbol{g} == \sum_{k=0}^{n-1} a_k \mathbf{\Psi}_k$) must have $\max_{k\in\{0,1,...,n-1\}} |a_k| = \frac{1}{\sqrt{n}}$ and as $\boldsymbol{g}$ is a unit vector, it must have all $|a_k| = \frac{1}{\sqrt{n}}$.
One such $\boldsymbol{g}$ is of the form $\frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} \mathbf{\Psi}_k$.

# Question 4

## 4(a) Only 1 non-zero element, m=1

### First Part

Solution may not be unique.
For example, $\Phi = [1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6]$ and $y = [8]$.
Then , $x_1 = [8\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]^T$ and $x_2 = [0\ \ 4\ \ 0\ \ 0\ \ 0\ \ 0]^T$ are at least 2 solutions.

Let $\Phi_i$ denote the element in $i$th column of $\Phi$, and $x_i$ denote the element in $i$th row of $x$. Then, we'll have $y = \sum_{i=1}^{n} \Phi_i x_i$; And we know only one of the $x_i$s is non-zero.
We'll proceed by cases:
(a) $y \neq 0$ and for all $i, \Phi_i \neq 0$. In this case, any $x$ such that $x_j = y/\Phi_j$ for some particular $j$ and all others zero will give a solution. thus, solution is clearly not unique. However the solution set is finite.
(b) $y = 0$ and for all $i, \Phi_i \neq 0$. All the $x_i$s must be zero in this case. Thus, no solution
(c) There exists an $i$ for which $\Phi_i = 0$. We could make any change to the value of corresponding $x_i$ and RHS won't change. Hence, in this case, if one solution exists, there are infinitely many more. In any case thus, again solution cannot be unique.

### Next Part

Next, if we knew the index $i$ of the non-zero element, with $y \neq 0$ and for that $i$, $\Phi_i \neq 0$; the unique solution will be the vector $x$ with all zero entries, except $x_i$ which shall be equal to $y/\Phi_i$. In other cases, no solution.
This can be easily realised using the equality $y = \sum_{i=1}^{n} \Phi_i x_i$ and given conditions.

## 4(b) Only 1 non-zero element, m=2

Solution may not be unique.
For example, $\Phi = \begin{bmatrix} 1 & 2 & 4 & 5 & 6 & 9 \\ 2 & 4 & 8 & 6 & 9 & 11 \end{bmatrix}$ and $y = \begin{bmatrix} 8 \\ 16 \end{bmatrix}$
Then , $x_1 = [8\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]^T$ and $x_2 = [0\ \ 4\ \ 0\ \ 0\ \ 0\ \ 0]^T$ are at least 2 solutions.
Suppose that the two elements of $\mathbf{y}$ are $y_1$ and $y_2$; and correspondingly we have the equations:
$y_1 = \sum_{i=1}^{n} \Phi_{1i} x_i \quad$ and $\quad y_2 = \sum_{i=1}^{n} \Phi_{2i} x_i$
Algorithm to determine $x$:
For all $i$, check if $\frac{y_1}{\Phi_{1i}} = \frac{y_2}{\Phi_{2i}} = k$

Dealing with zeroes:

(a) If both $y_1$ and $y_2$ are zero: above will be true iff both denominators are zero too. If we get such a case, infinite solutions. Else, no solution.
(b) If any one of $y_1$ and $y_2$ is zero, above will be true iff the corresponding denominator is zero too. We will consider this as returning true for above ratio test.
(c) If both numerators are non-zero, then the above will be false if any one or both of the denominators is/are zero. We will consider this as returning false for above ratio test.

Further we do this:

(a) If there is no $i$ where the above happens, no solution.
(b) If there is exactly one $i$ where the above happens, unique solution with all entries zero except the particular $x_i$ which will be set equal to the corresponding k. unless there was a case with both denominators zero. If that happens, infinite solutions.
(c) Else, not unique solution.

### 4(c) Only 2 non-zero elements, m=3

The equations to solve are:

$y_1 = \sum_{i=1}^{n} \Phi_{1i} x_i \quad$ and $\quad y_2 = \sum_{i=1}^{n} \Phi_{2i} x_i \quad$ and $\quad y_3 = \sum_{i=1}^{n} \Phi_{3i} x_i$

Consider a pair of integers $(a, b)$ with $0 < a < b \leq n$. We do the following for all such pairs:
Solve the system of linear equations:
$\Phi_{1a} x_a + \Phi_{1b} x_b = y_1$ and
$\Phi_{2a} x_a + \Phi_{2b} x_b = y_2$ .
If there is no solution, proceed.
If there is unique solution check if it satisfies $\Phi_{3a} x_a + \Phi_{3b} x_b = y_3$ . If yes, we have found one solution. Store it and proceed. If not, just proceed.
If there are infinite solutions, solve the following system:
$\Phi_{1a} x_a + \Phi_{1b} x_b = y_1$ and
$\Phi_{3a} x_a + \Phi_{3b} x_b = y_3$
If yes, we have found one solution. Store it and proceed. If infinite solutions for this too, claim infinite solutions and halt. If not even that, just proceed.

Thus, finally we'll have a solution set. The algo or the size of this set and the structure of the set (i.e. whether some of the solutions involve the pair $(x_a, x_b) = (0, k)$ or $(k, 0)$ or $(0, 0)$ ) will tell us if there are no, unique, finitely many or infinite solutions.

The fact is that if we want unique solutions, a sufficient condition would be every 4 columns of the matrix must be linearly independent. In that case, if there exists one solution, then it is unique.
Otherwise, we can find an example so that the solution will not be unique. For example, consider:

$$\Phi = \begin{bmatrix} 1 & 2 & 4 & 5 & 6 & 9 \\ 2 & 4 & 8 & 6 & 9 & 11 \\ 4 & 8 & 16 & 9 & 11 & 1 \end{bmatrix} \text{ and } y = \begin{bmatrix} 7 \\ 14 \\ 28 \end{bmatrix}$$

This clearly has at least two 2-sparse solutions: $x_1 = [1 \;\; 3 \;\; 0 \;\; 0 \;\; 0 \;\; 0]^T$ and $x_2 = [1 \;\; 0 \;\; 1.5 \;\; 0 \;\; 0 \;\; 0]^T$
In fact, we can prove that if
(a) There is at least one solution $\mathbf{x}$ to $\mathbf{y} = \mathbf{\Phi x}$ ; and (b) Every 4 columns of the matrix $\mathbf{\Phi}$ are linearly independent.
Then there is a unique solution.
(This was actually proved by contradiction in class as a theorem: any 2S columns of an $m \times n$ matrix $\Phi$ are linearly independent. Then any S-sparse signal f can be uniquely reconstructed from measurements $y = \Phi f$. )

### 4(d) Only 2 non-zero elements, m=4

Again here, I can give an example, for whic answer is not unique:
consider:

$$\Phi = \begin{bmatrix} 1 & 2 & 4 & 5 & 6 & 9 \\ 2 & 4 & 8 & 6 & 9 & 11 \\ 4 & 8 & 16 & 9 & 11 & 1 \\ 8 & 16 & 32 & 11 & 1 & 25 \end{bmatrix} \text{ and } y = \begin{bmatrix} 7 \\ 14 \\ 28 \\ 56 \end{bmatrix}$$

This clearly has at least two 2-sparse solutions: $x_1 = [1\ \ 3\ \ 0\ \ 0\ \ 0\ \ 0]^T$ and $x_2 = [1\ \ 0\ \ 1.5\ \ 0\ \ 0\ \ 0]^T$
In fact, we can prove that if
(a) There is at least one solution $\mathbf{x}$ to $\mathbf{y} = \mathbf{\Phi x}$ ; and (b) Every 4 columns of the matrix $\mathbf{\Phi}$ are linearly independent.
Then there is a unique solution.
(This was actually proved by contradiction in class as a theorem: any 2S columns of an $m \times n$ matrix $\Phi$ are linearly independent. Then any S-sparse signal f can be uniquely reconstructed from measurements $y = \Phi f$. )

# Question 5

## (a) Explain the similarities and differences...

Here, we'll explain the similarities and differences between the architecture given in the linked research paper and the video compressed sensing architecture using coded snapshots (Hitomi et al, ICCV 2011) that we studied in class.
The figures 1 and 2 show the experimental hardware setup wrt both the cases.

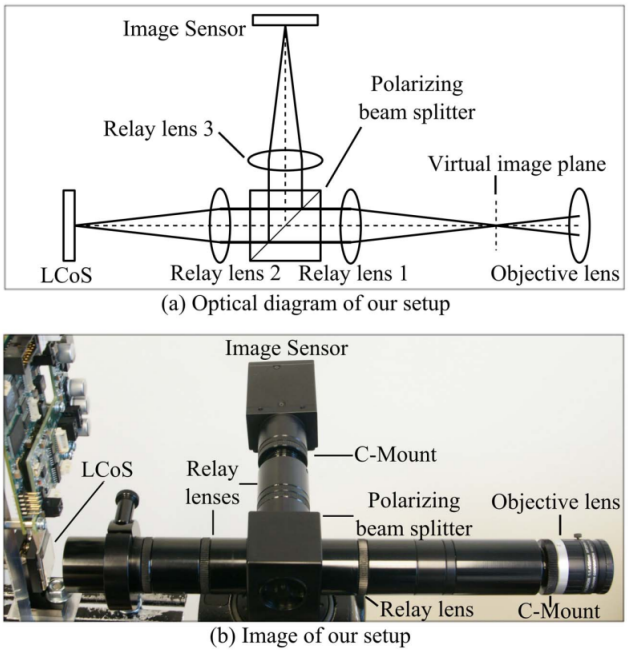


Figure 8: **Our hardware setup:** Optical diagram (top) and image (bottom) of our setup. Our system emulates fast per-pixel shutter using a liquid crystal on silicon device.

Figure 1: Hitomi et al paper' setup

**Similarities:**

1. In both the set-ups, the scene is first imaged through a simple objective lens before beginning with any complex processing.
2. Also, relay lens is used in both the cases for performing imaging of the modulated scene on to the image sensor (e.g. a CCD Camera).
3. Structural sparsity of the sub-frames in each frame is supposed to be useful to reduce the measurements in both the set-ups.
4. Hence, both the architectures are, in the end, supposed to save only the coded snapshots of each frame; which is later to be used for reconstruction.

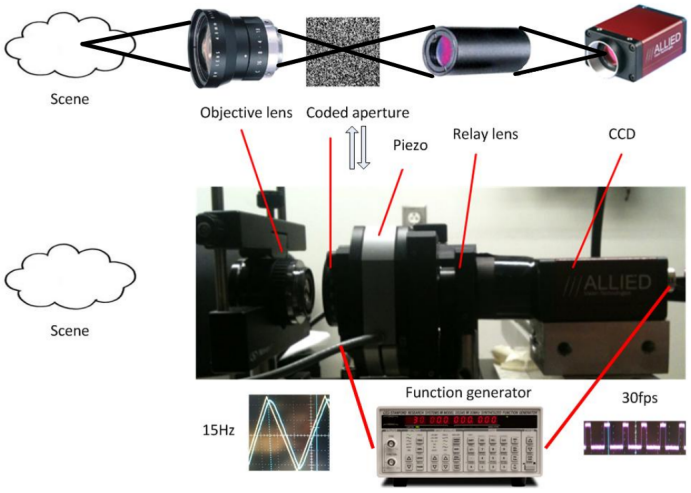


Fig. 6. CACTI Prototype hardware setup. The coded aperture is 5.06mm × 4.91mm and spans 248 × 256 detector pixels. The function generator moves the coded aperture and triggers camera acquisition with signals from its function and SYNC outputs, respectively.

Figure 2: Duke University's paper's setup

5. The "coding function" in both the architectures (i.e. called the coded aperture's transmission function in CACTI and known as the "sampling function" in the Hitomi et al paper) uses random binary sampling.

**Differences:**

1. Compressive measurement for visible imaging has been implemented using Liquid Crystal on Silicon (LCoS) devices to code pixel values incident on a single detector, in the Hitomi et al paper. LCoS is not used in CACTI.
2. Instead of LCoS, CACTI proposes mechanical translation of a passive coded aperture for the space-time compressive measurement.
3. The LCoS strategy increases, rather than decreases, operating power and bandwidth. CACTI does low power space-time compressive measurement.
4. Once imaged through an objective lens, the scene is first imaged on a virtual image plane in Hitomi et al paper's setup. In the other one, the continuous scene f is directly imaged onto the piezo-positioned mask through an objective lens.
5. The LCoS was operated at 1000 Hz. Such high frequencies are unheard of in the other setup.
6. Single bump exposure was essential in the Hitomi et al paper's setup. The coded aperture pattern in the other one didn't follow single bump exposure.

## (b) What cost function is the...

Here, we'll explain the cost function used in the unconstrained optimization problem pertaining to the Two-step Iterative-Shrinkage Thresholding (TwIST) Algorithm.
TwIST solves the unconstrained optimization problem:

$$\mathbf{f_e} = \arg\min_{\mathbf{f}} \mathcal{C}(\mathbf{f})$$

Where, the cost function $\mathcal{C}(\mathbf{f})$ is:

$$\mathcal{C}(\mathbf{f}) = ||\mathbf{g} - \mathbf{H}\mathbf{f}||^2 + \lambda\Omega(\mathbf{f})$$

Note:
(i) $N$ is the total number of pixels in one frame. We suppose $\sqrt{N}$ pixels per row and per column.
(ii) $N_F$ is the total number of frames in one coded snapshot.

Now we explain the meaning and dimensions of the terms in $\mathcal{C}(\mathbf{f})$: $\mathbf{f}, \mathbf{g}, \mathbf{H}, \Omega(\mathbf{f})$, and $\lambda$ respectively.

1. $\mathbf{f}$: This denotes the three-dimensional (2 spatial and 1 temporal) scene that we're trying to capture. $\mathbf{f}$ was originally the scene's discretized form, with $\mathbf{f} \in \mathbb{R}^{\sqrt{N}\times\sqrt{N}\times N_F}$. In the cost function, this $\mathbf{f}$ is flattened so that $\mathbf{f} \in \mathbb{R}^{NN_F\times 1}$.

   As we'll see later, in $\Omega(\mathbf{f})$, $f_{i,j,k}$ denotes the pixel value wrt the scene $\mathbf{f} \in \mathbb{R}^{\sqrt{N}\times\sqrt{N}\times N_F}$. $(i, j)$ are spatial indices and $k$ is the temporal index.

2. $\mathbf{g}$: This is the detector image $\mathbf{g} \in \mathbb{R}^{\sqrt{N}\times\sqrt{N}}$, obtained by integrating the $N_F$ temporal channels of $\mathbf{f}$, through a time-varying spatial transmission pattern. In the cost function, this $\mathbf{g}$ is flattened so that $\mathbf{g} \in \mathbb{R}^{N\times 1}$.

   We can actually write, $\mathbf{g} = \mathbf{H}\mathbf{f} + \mathbf{n}$, where $\mathbf{H}$ is as explained below, and $\mathbf{n} \in \mathbb{R}^{n\times 1}$ represents the flattened noise vector, representing the imaging noise.

3. $\mathbf{H}$: This is the system's discrete forward matrix. It accounts for sampling factors including the optical impulse response, pixel sampling function, and time-varying transmission function.

   It is a 2-D representation of the 3-D time-varying spatial transmission pattern $\mathbf{T} \in \mathbb{R}^{\sqrt{N}\times\sqrt{N}\times N_F}$.

   This uniquely codes each of the $N_F$ temporal channels of $\mathbf{f}$.

   Let $\mathbf{H_k} := \text{diag}[T_{1,1,k} \;\; T_{2,1,k} \;\; \cdots \;\; T_{\sqrt{N},\sqrt{N},k}]$ where $k = 1, 2, \ldots, N_F$.

   Each $\mathbf{H_k} \in \mathbb{R}^{N\times N}$

   Then, $\mathbf{H}$ is a concatenation of all $\mathbf{H_k}, k \in \{1, 2, \cdots, N_F\}$.

   $$\mathbf{H} := [\mathbf{H_1} \;\; \mathbf{H_2} \;\; \cdots \;\; \mathbf{H_{N_F}}]$$

   Thus, $\mathbf{H} \in \mathbb{R}^{N\times NN_F}$.

4. $\Omega(\mathbf{f})$: This is a regularizer function, included in the cost function so as to penalise characteristics of the estimated $\mathbf{f}$ that would result in poor reconstructions.

   Here, we use the Total Variation (TV) regularizer function, defined as:

$$\Omega(\mathbf{f}) = \sum_{k}^{N_F} \sum_{i,j}^{N} \sqrt{(f_{i+1,j,k} - f_{i,j,k})^2 + (f_{i,j,k} - f_{i,j+1,k})^2}$$

   Since many natural scenes are well-described by sparse gradients, this function was chosen which penalizes especially those estimates, which have sharp spatial gradients as desired.

5. $\lambda : \lambda \in \mathbb{R}$ is a scalar constant, known as regularization weight.

   (a) It basically *quantifies* how much importance should be given to the regularizer function over other terms in the cost function.

   (b) The regularization weight was chosen via experimental optimization over several test values $\lambda \in [0.3, 2]$.

   (c) It was observed that a weight of $\lambda = 1$ yielded in the clearest reconstructions.

# Question 6

## (a)

Title of paper - "Compressed Sensing Microscopy with Scanning Line Probes"
Published on 26th Sept 2019 at arXiv.
Link to the paper - `https://arxiv.org/pdf/1909.12342.pdf`.

## (b)

Image reconstruction from scanning chemical microscopic imaging (SECM) data measured by continuous line probe (CLP).
A CLP-SECM scan is setup by putting the sample on a rotational stage and locate the line probe at one side of sample. A single scan line is generated as follows:
The microscope firstly rotate its stage by preassigned angle, then followed by step-wise moving CLP to the other end of sample with constant step-size.
At each steps, CLP measures the current generated by electro-chemical reaction between sample and probe end.
A line scan is the collection of all measurements at all steps with sample rotation fixed at some given angle. Combining single line scans under different sample rotation angles gives the complete CLP-SECM scan.

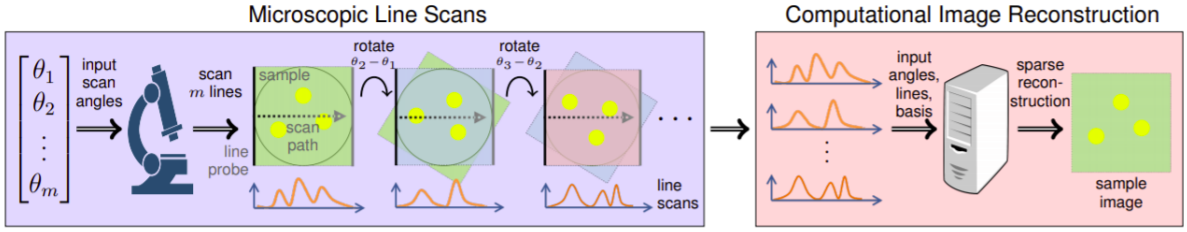


Fig. 2. Scanning procedure of SECM with continuous line electrode probe. The user begins with mounting the sample on a rotational stage of microscope and chooses $m$ scanning angles. The microscope then carries on sweeping the line probe across the sample, and measures the accumulated current generated between the interreaction of probe and the sample at equispaced intervals of moving distance. After a sweep ends, the sample is rotated to another scanning angle and the scanning sweep procedure repeats, until all $m$ line scans are finished. Collecting all scan lines, and providing the information of the scanning angles, the microscope system parameters (such as the point spread function) and the sparse representing basis of image, the final sample image is produced via computation with sparse reconstruction algorithm.

Figure 3: Page 2 of the paper - describes the basic flow of data

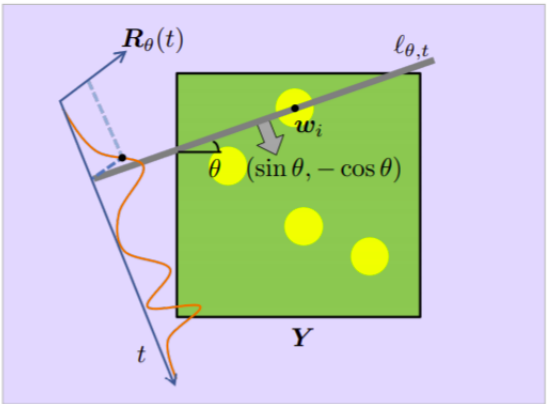


Fig. 3. Mathematical expression of a single measurement from the line probe. When the stage rotate by $\theta$ clockwise, the relative rotation of probe to sample is counterclockwise by $\theta$. The grey line in the figure represents the rotated line probe, orienting in direction $\boldsymbol{u}_\theta = (\cos\theta, \sin\theta)$, and is sweeping in direction $\boldsymbol{u}_\theta^\perp = (\sin\theta, -\cos\theta)$. When it comes across the point $\boldsymbol{w}_i$ where $t = \langle \boldsymbol{u}_\theta^\perp, \boldsymbol{w}_i \rangle$, it integrates over the contact region $\ell_{\theta,t}$ between the probe and substate and produces a measurement $\boldsymbol{R}_\theta(t)$.

Figure 4: Page 3 of the paper - describes how the measurement of $R_\theta(t)$ is made

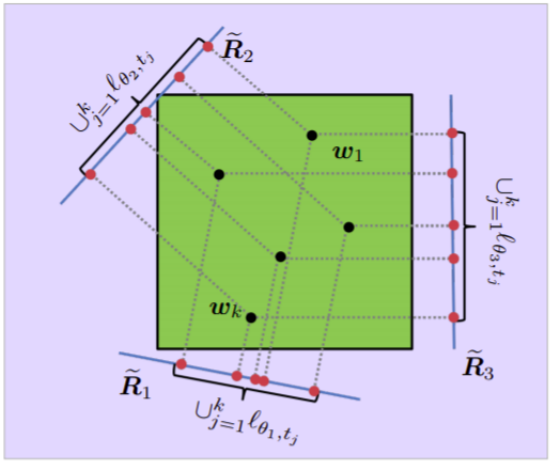


Fig. 4. Proof sketch for sufficiency of image recovery from three line projections. Given a sample with separated tiny discs $\boldsymbol{w}_1, \ldots \boldsymbol{w}_k$ (black dots), randomly choosing three lines projection forms lines $\widetilde{\boldsymbol{R}}_1, \widetilde{\boldsymbol{R}}_2, \widetilde{\boldsymbol{R}}_3$, in which all the discs after line projection (red dots) are well-separated. From each of these lines, we construct the dual $\widetilde{\boldsymbol{Q}}$ as center of red dots, and a back projection image form the dual (dash lines), forming the set $\cup_{j=1}^k \ell_{\theta_i,t_j}$. Intersection of three such line sets is exactly the set of ground truth disc centers.

Figure 5: Page 5 of the paper - three line projections shown

## (c)

In the experiments performed, the interested sample—chemical reactive species attached on substrate is "highly structured". More specifically, the shape reactive species are known, are much smaller in comparison with the sample size, and also are sparsely populated on the substrate.
A sample Y can therefore being modeled as 2D-convolution between the reactive species $D$ and a sparse activation map $X_0$, denote as $Y = D * X_0$. We adopt **compressed sensing methodology**, to reconstruct the SECM sample image by finding the exact locations map $X_0$. Define CLP line scan $L$ takes the sample $Y$ and CLP properties $p$ - scanning angles, point-spread function of CLP, etc., and generates measured current lines $R$ where $R = L[D * X_0, p]$.

The reconstruction algorithm gathers CLP-SECM scan lines $R$ to perform image reconstruction using numerical optimization procedure. The algorithm formulate the task of finding the location of the reactive species as a variation of **LASSO problem**. Namely, the algorithm solves the following problem

$$\min_{X>0,p} C||X||_1 + ||L[D * X - R, p]||^2$$

where $C$ is some positive constant, $||X||_1$ denotes total magnitude of $X$, and $||R||^2$ represents energy of lines $R$. We claim that the sample image reconstruction is successful, if the location (non-zero entries) of $X$ is identical to $X_0$.