Blind Deconvolution

Xupeng Yu
xuy004@eng.ucsd.edu

Abstract

In this project, we will implement method from the paper *Blind Deconvolution using Convex Programming* [1] to solve the problem of blind deconvolution. Blind deconvolution is the problem of recovering two signals from the output of their convolution. To make this object a solvable convex problem, the paper adds some constraints on it. After that, we will implement another non-blind convex deconvolution method from paper *Rapid, Robust, and Reliable Blind Deconvolution via Nonconvex Optimization*. And we will compare the theoretical method and the experiment results of the two methods.

1 Background and Motivation

Consider two signals, w and x, we observe their convolution $y = w * x$, and we want to separate them.This challenging problem, commonly referred as blind deconvolution problem, arises in many areas of science and technology, including astronomy, medical imaging, optics, and communications engineering. There exist many method to solve this problem. *Blind Deconvolution using Convex Programming* regards this problem as a convex problem,*Rapid, Robust, and Reliable Blind Deconvolution via Nonconvex Optimization*[2] talks about non-convex solution. And *Fast High-Quality non-Blind Deconvolution Using Sparse Adaptive Priors* [3] use Gaussian deconvolution and other filters to deal with it.

2 Method

Consider the equation

$$y = w * x$$

Without any assumption, we cannot recover w and x from y because we don't know structural details of w and x. For example, if x and $y$lives in the same subspace($x = \alpha y$), they cannot be separated. However,if we give structural assumptions of w and x, we can usually find a unique solution.
First, we assume y has length L. Therefore, it lives in the known subspace of R^L, we suppose w,x has rank K,N. In other words,

$$w = Bh, h \in R^K, B \in R$$

$$x = Cm, m \in R^N$$

when k and N are much smaller than L, there is high possibility that $R(B) \neq R(C)$ are different. The more difference between $R(B)$ and $R(C)$, the more likely we can solve the problem.In the extreme case, if $R(B) \cap R(C)$ is nullspace, we can easily separate them.

$$\begin{aligned} \boldsymbol{y} &= m(1)\boldsymbol{w} * \boldsymbol{C}_1 + m(2)\boldsymbol{w} * \boldsymbol{C}_2 + \cdots + m(N)\boldsymbol{w} * \boldsymbol{C}_N \\ &= [\operatorname{circ}(\boldsymbol{C}_1) \operatorname{circ}(\boldsymbol{C}_2) \cdots \operatorname{circ}(\boldsymbol{C}_N)] \begin{bmatrix} m(1)\boldsymbol{w} \\ m(2)\boldsymbol{w} \\ \vdots \\ m(N)\boldsymbol{w} \end{bmatrix} \end{aligned}$$

2.1 Transformed Variable

It is more convenient solve convolution problem in the frequency domain. Therefore, we make use of the L points discrete Fourier matrix

$$\boldsymbol{F}(\omega,\ell)=\frac{1}{\sqrt{L}}\mathrm{e}^{-\mathrm{j}2\pi(\omega-1)(\ell-1)/L},\quad 1\leq\omega,\ell\leq L$$

to convert y into the frequency domain. That is

$$\hat{\boldsymbol{y}}=\boldsymbol{F}\boldsymbol{y}=\left[\begin{array}{cccc}\Delta_1\hat{\boldsymbol{B}} & \Delta_2\hat{\boldsymbol{B}} & \cdots & \Delta_N\hat{\boldsymbol{B}}\end{array}\right]\left[\begin{array}{c}m(1)\boldsymbol{h}\\ m(2)\boldsymbol{h}\\ \vdots\\ m(N)\boldsymbol{h}\end{array}\right]$$

Here $\hat{C}=FC$,$\hat{B}=FB$, $\operatorname{circ}(C_n)=F^*\Delta_nF$,$\Delta_n=\operatorname{diag}\left(\sqrt{L}\hat{C}_n\right)$
Therefore,while y is a nonlinear combination of the coefficients h and m, it is a linear combination of the entries of their outer product $\boldsymbol{X}_0=\boldsymbol{h}\boldsymbol{m}^*$.We can express $\hat{\boldsymbol{y}}=\mathcal{A}(\boldsymbol{X}_0)$, here $\mathcal{A}$ is a linear function which maps a $K\times N$ matrix to R^L.For $\mathcal{A}$ to be invertible over all matrices, we need at least as many observations as unknowns, $L\geq NK$. But since we know $\boldsymbol{X}_0$ has special structure, namely that its rank is 1, we will be able to recover it from $L\ll NK$ under certain conditions on A.

$$\begin{aligned}\hat{y}(\ell)&=\hat{c}_\ell(1)m(1)\left\langle\boldsymbol{h},\hat{\boldsymbol{b}}_\ell\right\rangle+\hat{c}_\ell(2)m(2)\left\langle\boldsymbol{h},\hat{\boldsymbol{b}}_\ell\right\rangle+\cdots+\hat{c}_\ell(N)m(N)\left\langle\boldsymbol{h},\hat{\boldsymbol{b}}_\ell\right\rangle\\&=\langle\hat{\boldsymbol{c}}_\ell,\boldsymbol{m}\rangle\left\langle\boldsymbol{h},\hat{\boldsymbol{b}}_\ell\right\rangle\\&=\operatorname{trace}\left(\boldsymbol{A}_\ell^*\left(\boldsymbol{h}\boldsymbol{m}^*\right)\right),\quad\text{where}\quad\boldsymbol{A}_\ell=\hat{\boldsymbol{b}}_\ell\hat{\boldsymbol{c}}_\ell^*\end{aligned}$$

Here $\hat{\boldsymbol{b}}_\ell\in\mathbb{C}^K$ is the lth column of $\hat{B}^*$ and $\hat{\boldsymbol{c}}_\ell\in\mathbb{C}^N$ is the lth column of $\sqrt{L}\hat{C}$. So far, we have transfrom the deconvolution problem to a as a linear inverse problem over the (nonconvex) set of rank-1 matrices.And $\boldsymbol{A}$ here is the suitable transformed variable that makes the problem a linear function.However till now, the problem is still not a convex problem, to make the problem convex, the paper uses other relaxation called the nuclear norm minimization. Here nuclear norm is the sum of the singular values of a matrix, as a proxy for rank.

2.2 Convex Relaxation

Given $\hat{\boldsymbol{y}}\in\mathbb{C}^L$, our goal is to find $\boldsymbol{h}\in\mathbb{R}^K$ and $\boldsymbol{m}\in\mathbb{R}^N$. A natural way to choose h,m is using least-squares.

$$\min_{\boldsymbol{u},\boldsymbol{v}}\|\boldsymbol{u}\|_2^2+\|\boldsymbol{v}\|_2^2\text{ subject to }\quad\hat{\boldsymbol{y}}(\ell)=\langle\hat{\boldsymbol{c}}_\ell,\boldsymbol{u}\rangle\left\langle\boldsymbol{v},\hat{\boldsymbol{b}}_\ell\right\rangle,\quad\ell=1,\ldots,L$$

This is a non-convex quadratic optimization problem. The cost function is convex, but the quadratic equality constraints mean that the feasible set is non-convex. A good approach to solve constraint quadratic is to use duality, which is the semi-definite program (SDP).

$$\max_{\boldsymbol{\lambda}}\operatorname{Re}\langle\hat{\boldsymbol{y}},\boldsymbol{\lambda}\rangle\text{ subject to }\left[\begin{array}{cc}\boldsymbol{I} & \sum_{\ell=1}^L\lambda(\ell)\boldsymbol{A}_\ell\\ \sum_{\ell=1}^L\lambda(\ell)^*\boldsymbol{A}_\ell^* & \boldsymbol{I}\end{array}\right]\succeq 0$$

The dual SDP for the equation above is

$$\begin{array}{c}\min_{\boldsymbol{W}_1,\boldsymbol{W}_2,\boldsymbol{X}}\ \frac{1}{2}\operatorname{trace}(\boldsymbol{W}_1)+\frac{1}{2}\operatorname{trace}(\boldsymbol{W}_2)\\ \text{subject to}\\ \left[\begin{array}{cc}\boldsymbol{W}_1 & \boldsymbol{X}\\ \boldsymbol{X}^* & \boldsymbol{W}_2\end{array}\right]\succeq 0\\ \hat{\boldsymbol{y}}=\mathcal{A}(\boldsymbol{X})\end{array}$$

,which is equivalent to

$$\begin{array}{ll}\min & \|\boldsymbol{X}\|_*\\ \text{subject to} & \hat{\boldsymbol{y}}=\mathcal{A}(\boldsymbol{X})\end{array}$$

Therefore,the nuclear norm heuristic is the "dual-dual" relaxation of the intuitive but non-convex least-squares estimation problem.

Algorithm 1: Blind deconvolution

Input: $\boldsymbol{y}, \boldsymbol{B}, \boldsymbol{c}$
Output: $\hat{w}, \hat{x}$
$\boldsymbol{A}_\ell = \hat{\boldsymbol{b}}_\ell \hat{\boldsymbol{c}}_\ell^*$
$\min \quad \|\boldsymbol{X}\|_*$
subject to $\quad \hat{\boldsymbol{y}} = \mathcal{A}(\boldsymbol{X})$
$[U, S, V] = svd(\boldsymbol{X})$
$\hat{w} = B * \hat{h}$
$\hat{x} = C * \hat{m}$

2.3 Recovering Constituent Signals

After that, according to $\boldsymbol{X} = \boldsymbol{h}\boldsymbol{m}^*$. We can use singular value decomposition to get $\hat{h}$ and $\hat{m}$ from $\boldsymbol{X}$,which is the first column of the U and V matrix.([U, S, V] = svd($\boldsymbol{X}$))

$$\hat{w} = B * \hat{h}$$
$$\hat{x} = C * \hat{m}$$

2.4 Theoretical Guarantees

We can guarantee the effectiveness of our convex relaxation for relatively large subspace dimensions K and N when B is incoherent in the Fourier domain, and when C is generic.And we will make several assumptions below.
First,the signal w is time-limited to Q, where $K \leq Q \leq L$. This means that the last $L - Q$ rows of B are zero.
Second,without additional loss of generality, that the columns of B are orthonormal, means $\boldsymbol{B}^*\boldsymbol{B} = \hat{\boldsymbol{B}}^*\hat{\boldsymbol{B}} = \sum_{\ell=1}^{L} \hat{\boldsymbol{b}}_\ell \hat{\boldsymbol{b}}_\ell^* = \boldsymbol{I}$.
And the effect of our result depends on several conditions.
First, our results will be most powerful when B is diffuse in the Fourier domain, meaning that the $\hat{\boldsymbol{b}}_\ell$ all have similar norms.
Second, our results will also depend on the minimum of these norms, we will always have $0 \leq \mu_{\min}^2 \leq 1$ and $\mu_{\min}^2 \leq \mu_{\max}^2$.

$$\mu_{\min}^2 = \frac{L}{K} \min_{1 \leq \ell \leq L} \left\| \hat{\boldsymbol{b}}_\ell \right\|_2^2$$

Third, Our analytic results also depend on how diffuse the particular signal we are trying to recover $\boldsymbol{w} = \boldsymbol{B}\boldsymbol{h}$ is in the Fourier domain.
The theoretical guarantee of the convex algorithm is:
Theorem . Fix $\alpha \geq 1$. Let $\boldsymbol{B}$ be a deterministic $L \times \boldsymbol{K}$ matrix satisfying (10) and whose last $L - Q$ rows are zero with[3]

$$Q \geq C_\alpha \cdot M \log(L) \log M, \quad M = \max\left(\mu_{\max}^2 K, \mu_h^2 N\right)$$

and L as an integer multiple of Q with $L/Q \geq \log\left(C_a' \sqrt{N \log L}\right) / \log 2$. Let C be an $L \times N$ Gaussian random matrix drawn as

$$\hat{c}_\ell \sim \begin{cases} \text{Normal}(0, \boldsymbol{I}) & \ell = 1 \\ \text{Normal}\left(0, 2^{-1/2}\boldsymbol{I}\right) + \text{jNormal}\left(0, 2^{-1/2}\boldsymbol{I}\right) & \ell = 2, \ldots, L/2 + 1 \end{cases}$$
$$\hat{\boldsymbol{c}}_\ell = \hat{\boldsymbol{c}}_{L-\ell+2}, \quad \text{for } \ell = L/2 + 2, \ldots, L$$

and set $\boldsymbol{w} = \boldsymbol{B}\boldsymbol{h}$ and $\boldsymbol{x} = \boldsymbol{C}\boldsymbol{m}$ for arbitrary basis coefficients $\boldsymbol{h} \in \mathbb{R}^K, \boldsymbol{m} \in \mathbb{R}^N$. Then there exists a constant $C_\alpha''' = O(\alpha)$ depending only on α such that if

$$\max\left(\mu_{\max}^2 K, \mu_h^2 N\right) \leq \frac{L}{C_\alpha'' \log^3 L}$$

then $\boldsymbol{X}_0 = \boldsymbol{h}\boldsymbol{m}^*$ is the unique solution to with probability $1 - O\left(L^{-\alpha+1}\right)$, and we can recover both $\boldsymbol{w}$ and $\boldsymbol{x}$ (within a scalar multiple) from $\boldsymbol{y} = \boldsymbol{w} * \boldsymbol{x}$.
This is a probabilistic guarantee.

3 results

The result of our experiment is shown below. To make the computation easier, we fix Length $L = 256$ and $5 \leq K \leq 100$, $5 \leq N \leq 100$.Here our $\boldsymbol{B}$ and $\boldsymbol{C}$ are chosen from the first K/N columns of a permutated identity matrix, $\boldsymbol{m}$ and $\boldsymbol{h}$ are random vectors with norm 1.

From the graph we can see the boundary between the success of deconvolution which is white in

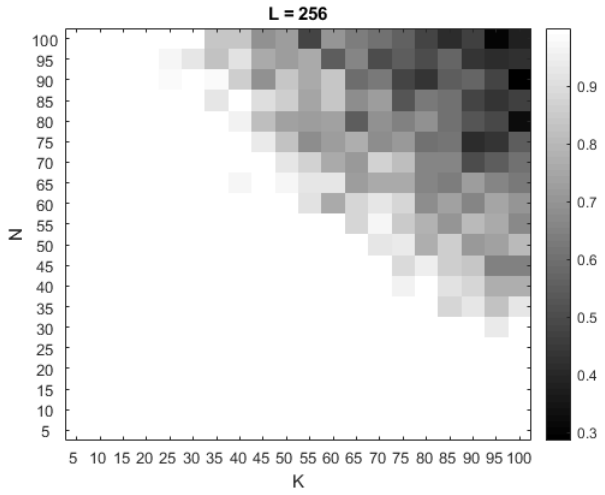

Figure 1: Convex deconvolution

the image and the failure of deconvolution which is black in the image. We can see that if we fix K/N, the probability of success decreases with the increase of N/K. And we are able to deconvolve when $L \gtrsim 2.1(K + N)$, which is similar to the result from the paper $L \gtrsim 2.7(K + N)$.

4 discussion

4.1 Non-blind and Non-convex approach

Here we use another non-blind deconvolution method from the paper *Rapid, Robust, and Reliable Blind Deconvolution via Nonconvex Optimization*. The setting of this paper is about the same as the convex optimization paper. In the paper, h and x can be complex vectors, but to compare with the convex one, we assume h and x are real vectors. Therefore,

$$\boldsymbol{y} = \boldsymbol{f} * \boldsymbol{g} + \boldsymbol{n}$$

and

$$f = Bh, h \in R^K, B \in R$$
$$g = Cx, x \in R^N$$

Also, it first turns the problem into frequency space using the Fourier matrix.

$$\sqrt{L}\boldsymbol{F}\boldsymbol{y} = \mathrm{diag}(\sqrt{L}\boldsymbol{F}\boldsymbol{f})(\sqrt{L}\boldsymbol{F}\boldsymbol{g}) + \sqrt{L}\boldsymbol{F}\boldsymbol{n}$$

Here $\boldsymbol{F}\boldsymbol{f} = \boldsymbol{B}\boldsymbol{h}$.Therefore,

$$\frac{1}{\sqrt{L}}\hat{y} = \mathrm{diag}(Bh)\overline{Ax} + \frac{1}{\sqrt{L}}Fn$$

The paper uses linear operator $\mathcal{A}(Z) = \{\boldsymbol{b}_l^* \boldsymbol{Z} \boldsymbol{a}_l\}_{l=1}^L$. The problem bocomes

$$\min_{(\boldsymbol{h},\boldsymbol{x})} F(\boldsymbol{h}, \boldsymbol{x})$$

,where

$$F(\boldsymbol{h}, \boldsymbol{x}) := \| \mathrm{diag}(\boldsymbol{B}\boldsymbol{h})\overline{\boldsymbol{A}\boldsymbol{x}} - \boldsymbol{y}\|^2 = \|\mathcal{A}\left(\boldsymbol{h}\boldsymbol{x}^* - \boldsymbol{h}_0\boldsymbol{x}_0^*\right) - \boldsymbol{e}\|^2$$

We assume there is no noise here. Then we define

$$F_0(\boldsymbol{h}, \boldsymbol{x}) := \|\mathcal{A}\left(\boldsymbol{h}\boldsymbol{x}^* - \boldsymbol{h}_0\boldsymbol{x}_0^*\right)\|_F^2$$

This is a nonlinear least square problem therefore it is a challenging optimization problem since it is highly nonconvex. The paper then uses a good gradient method which avoids the problem of local minima which is a common problem in such kind of solutions.
The paper makes the following assumption:

$$\mathcal{N}_{d_0} := \{(\boldsymbol{h}, \boldsymbol{x}) : \|\boldsymbol{h}\| \leq 2\sqrt{d_0}, \|\boldsymbol{x}\| \leq 2\sqrt{d_0}\}$$

and $d_0 = \|\boldsymbol{h}_0\| \|\boldsymbol{x}_0\|$

$$\mathcal{N}_\mu := \left\{\boldsymbol{h} : \sqrt{L}\|\boldsymbol{B}\boldsymbol{h}\|_\infty \leq 4\sqrt{d_0}\mu\right\}$$

$$\mathcal{N}_e := \{(\boldsymbol{h}, \boldsymbol{x}) : \|\boldsymbol{h}\boldsymbol{x}^* - \boldsymbol{h}_0\boldsymbol{x}_0^*\|_F \leq \varepsilon d_0\}$$

The gradient descent should remain in the area $\mathcal{N}_{\mathrm{d}_0} \cap \mathcal{N}_\mu \cap \mathcal{N}_\varepsilon$.The paper uses the regularizer

$$G(\boldsymbol{h}, \boldsymbol{x}) = \rho\left[G_0\left(\frac{\|\boldsymbol{h}\|^2}{2d}\right) + G_0\left(\frac{\|\boldsymbol{x}\|^2}{2d}\right) + \sum_{l=1}^{L} G_0\left(\frac{L\left|\boldsymbol{b}_l^*\boldsymbol{h}\right|^2}{8d\mu^2}\right)\right]$$

to make the gradient descent remain inside $\mathcal{N}_{\mathrm{d}_0} \cap \mathcal{N}_\mu$

$$\tilde{F}(\boldsymbol{h}, \boldsymbol{x}) = F(\boldsymbol{h}, \boldsymbol{x}) + G(\boldsymbol{h}, \boldsymbol{x})$$

$$\nabla\widetilde{F}_h = \nabla F_h + \nabla G_h, \quad \nabla\widetilde{F}_x = \nabla F_x + \nabla G_x$$

$$\nabla F_h = \mathcal{A}^*\left(\mathcal{A}\left(\boldsymbol{h}\boldsymbol{x}^*\right) - \boldsymbol{y}\right)\boldsymbol{x} = \mathcal{A}^*\left(\mathcal{A}\left(\boldsymbol{h}\boldsymbol{x}^* - \boldsymbol{h}_0\boldsymbol{x}_0^*\right) - \boldsymbol{e}\right)\boldsymbol{x}$$

$$\nabla F_x = \left[\mathcal{A}^*\left(\mathcal{A}\left(\boldsymbol{h}\boldsymbol{x}^*\right) - \boldsymbol{y}\right)\right]^*\boldsymbol{h} = \left[\mathcal{A}^*\left(\mathcal{A}\left(\boldsymbol{h}\boldsymbol{x}^* - \boldsymbol{h}_0\boldsymbol{x}_0^*\right) - \boldsymbol{e}\right)\right]^*\boldsymbol{h}$$

$$\nabla G_h = \frac{\rho}{2d}\left[G_0'\left(\frac{\|\boldsymbol{h}\|^2}{2d}\right)\boldsymbol{h} + \frac{L}{4\mu^2}\sum_{l=1}^{L} G_0'\left(\frac{L\left|\boldsymbol{b}_l^*\boldsymbol{h}\right|^2}{8d\mu^2}\right) b_l b_l^*\boldsymbol{h}\right]$$

$$\nabla G_x = \frac{\rho}{2d} G_0'\left(\frac{\|\boldsymbol{x}\|^2}{2d}\right)\boldsymbol{x}$$

The whole process can be divided into two parts.
First, we initialize (u_0, v_0)

Algorithm 2: Initialization

Input: $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{y}$
Output: $(\boldsymbol{u}_0, \boldsymbol{v}_0)$
1. Compute $\mathcal{A}^*(\boldsymbol{y})$
2. Find the leading singular value, left and right singular vectors of $\mathcal{A}^*(\boldsymbol{y})$, denoted by $d\,\hat{h}_0$ and $\hat{x}_0$ respectively.
3: Solve the following optimization problem:

$$\boldsymbol{u}_0 := \mathrm{argmin}_{\boldsymbol{z}} \left\|\boldsymbol{z} - \sqrt{d}\boldsymbol{h}_0\right\|^2, \text{ subject to } \sqrt{L}\|\boldsymbol{B}z\|_\infty \leq 2\sqrt{d}\mu$$
$$\text{and } \boldsymbol{v}_0 = \sqrt{d}\hat{\boldsymbol{x}}_0$$

Then, we can perform gradient descent.

Algorithm 3: Gradient Descent

Input: $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{y}$
Output: $(\boldsymbol{u}_0, \boldsymbol{v}_0)$
1.Initialization: obtain $(\boldsymbol{u}_0, \boldsymbol{v}_0)$
for $t = 1, 2, \ldots,$ **do**
  2: $\boldsymbol{u}_t = \boldsymbol{u}_{t-1} - \eta\nabla\widetilde{F}_h\left(\boldsymbol{u}_{t-1}, \boldsymbol{v}_{t-1}\right)$
  3: $\boldsymbol{v}_t = \boldsymbol{v}_{t-1} - \eta\nabla\widetilde{F}_{\boldsymbol{x}}\left(\boldsymbol{u}_{t-1}, \boldsymbol{v}_{t-1}\right)$
end

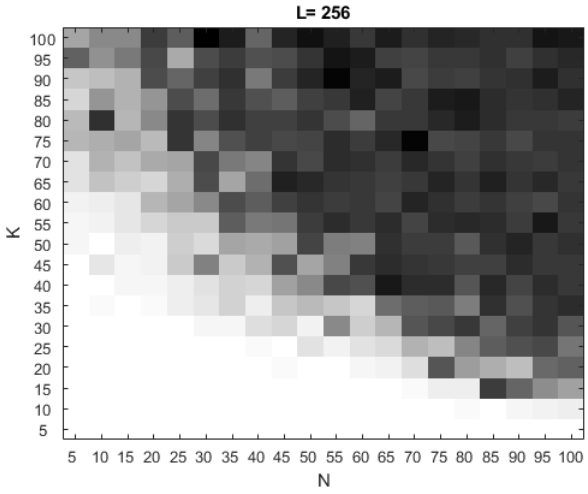

Figure 2: Nonconvex deconvolution

The result of the method is shown below.
Here we still fix Length $L = 256$ and $5 \leq K \leq 100$, $5 \leq N \leq 100$. From the result, we can find that the convex and nonconvex have the same trend that if we fix K/N, the probability of success decreases with the increase of N/K. However, if under the same N and K, the convex method will give a better result.
Also, there are ways to deal with non-blind deconvolution problem. *Fast High-Quality non-Blind Deconvolution Using Sparse Adaptive Priors* is a paper discussing using Gaussian deconvolution and other filters to deal with this problem.The model of the problem is

$$g = hf + n$$

. The difficulty of this problem is to deal with the noise n.

$$f = \arg\min \delta(\boldsymbol{f})$$
$$\delta(\boldsymbol{f}) = \|h\boldsymbol{f} - g\|_2^2 + \sum_{x=1}^{5} \lambda_s \|d_s \boldsymbol{f} - w_s\|_2^2$$

Here The matrix $d_n, s \in \{1, \ldots, 5\}$, represent the first and second-order-derivative filter operators: d_x, d_y, d_{xx}, d_{yy} and d_{xy}, respectively. Therefore, we use some prior knowledge about the derivatives of f.
By taking derivative of the minimum function, we can get

$$\left(\boldsymbol{h}^T\boldsymbol{h} + \sum_{s=1}^{5} \lambda_s d_s^T d_s\right) \hat{f} = \boldsymbol{h}^T\boldsymbol{g} + \sum_{s=1}^{5} \lambda_s d_s^T w_s$$

which is also

$$\boldsymbol{a}\hat{f} = \boldsymbol{b}$$

where

$$\boldsymbol{a} = \boldsymbol{h}^T\boldsymbol{h} + \sum_{s=1}^{5} \lambda_s d_s^T d_s$$
$$\boldsymbol{b} = \boldsymbol{h}^T\boldsymbol{g} + \sum_{s=1}^{5} \lambda_s d_s^T w_s$$

In the frequency domain,

$$\boldsymbol{A} \circ \widehat{\boldsymbol{F}} = \boldsymbol{B}$$

where

$$\boldsymbol{A} = \boldsymbol{H}^* \circ \boldsymbol{H} + \sum_{s=1}^{5} \lambda_s D_s^* \circ D_s$$
$$\boldsymbol{B} = \boldsymbol{H}^* \circ \boldsymbol{G} + \sum_{t=1}^{5} \lambda_s D_s^* \circ W_s$$
$$\hat{f} = \mathcal{F}^{-1}(\boldsymbol{B} \cdot / \boldsymbol{A})$$

The algorithm is described as:
First, we get the initial guess $\hat{f}^{(0)}$ by doconvolving using the using standard Tikhonov regularization.

This may cause some ringing effects.
Second, we get $\hat{f}^{(1)}$ using edge preserve filter to the initial guess to protect the edge.
Third, we get the first and second derivative of $\hat{f}^{(1)}$.
Finally, we compute the actual deconvolution of the image using the equations above.
It is also shown below.

Algorithm 4: Non-blind deconvolution

Input: g : captured image,h : blurring kernel, λ_s : regularization weights
Output: $\boldsymbol{f}$: deblurred image
1. Gaussian deconvolution
2. Evaluate $\hat{\boldsymbol{f}}^{(0)}$
3. Edge-preserving smoothing filter $\vec{f}^{(1)} + \mathcal{EPS}\left(\hat{f}^{(0)}\right)$; // EPS: edge-preserving filter
4. Evaluate priors Compute w_2 using $\vec{f}^{(1)}$ and Eq.(12)
5. Evaluate $\hat{f}$ using Eq. $(10a, 10b$ and $11)$ with $\boldsymbol{W}_n = \mathcal{F}\left(\boldsymbol{w}_n\right)$; $\quad //\mathcal{F}$: Fourier transform.

This algorithm does not use gradient descent or convex optimization method. Therefore it does not use a for loop and can be finished in a short time.

4.2 violating conditions

We tried to violate sparsity and low rank condition in the following experiment. We can find that although the result is a little worse than the original one, the result is still pretty good. Therefore, the non-blind algorithm is quite robust.

4.2.1 sparsity

Previously, $\boldsymbol{B}$ and $\boldsymbol{C}$ are sparse matrix, which means most of the elements in the two matrices are zero. We do this by letting B and C formed by a subset of the identity matrix.To violate sparsity, we choose $\boldsymbol{B}$ and $\boldsymbol{C}$ to be random matrices, then we normalize each column vector of the matrix.The experiment result is shown below. From the plot, we can see that under the same $\boldsymbol{B}$ and $\boldsymbol{C}$, the

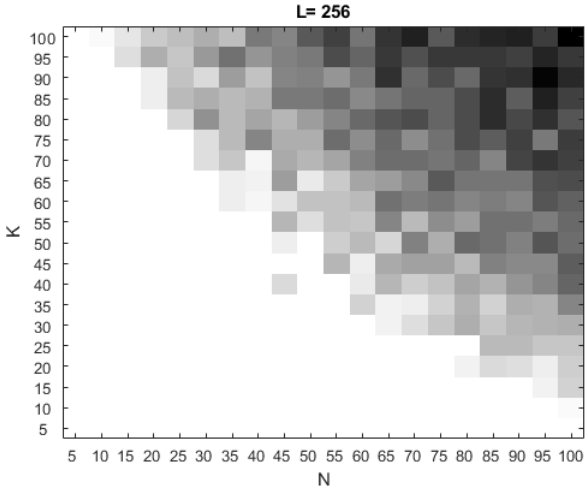

Figure 3: Dense $\boldsymbol{B}$ and $\boldsymbol{C}$

dense matrix gives a worse result than the sparse matrix.

4.2.2 low rank

From our phase transition plots we can find when N and K is smaller, which means we have low rank B and C, our probability of successful recovery is high.However when N and K becomes large, which means we are more likely to get high rank B and C, our probability of successful recovery decreases. Therefore, violating low rank condition will hurt our result.

5 2D blind deconvolution

The 2D blind deconvolution problem is similar to 1D problem but require some extra operations.
First, we need to find the most significant wavelet coefficients for the blur kernel and the target object.
To find the components of the kernel, we first reshape the kernel into a line vector, then we find the non-zero places in the vector as the B for the kernel.
We find the wavelet coefficients of the target from the blurred image. We need to do haar transformation to the blurred image,and select the indices of the N largest wavelet coefficients as a proxy for the support of the significant coefficients of the original image. The largest N wavelet coefficients capture most of the energy in the blurred image.
The theoretical recovery result of the paper is shown below.However, I do not produce the exact result as the paper shows. The result of my recovery is shown below. Also I have to resize the image for hardware limitation.

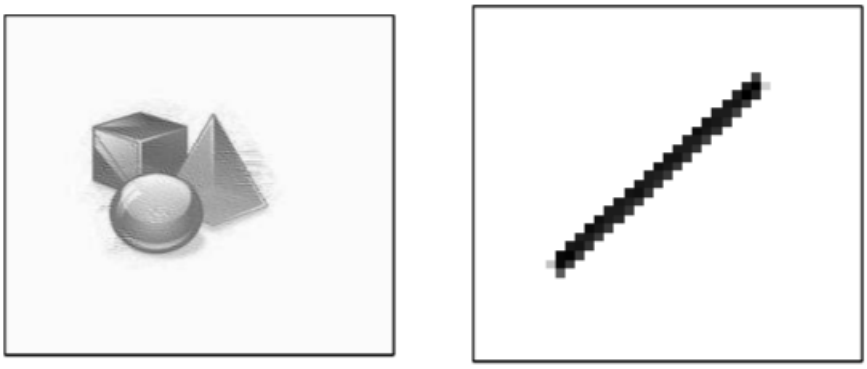

Figure 4: 2D blind deconvolution result from the paper

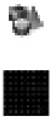

Figure 5: My bad 2D blind deconvolution result

6 dataset

Our 1D experiment does not uses any dataset. And our 2D experiment uses the image the same as the original paper uses to compare the result.

References

[1] Ali Ahmed, Benjamin Recht, and Justin Romberg. Blind deconvolution using convex programming, 2012.

Figure 6: Picture used for 2D blind deconvolution

[2] Xiaodong Li, Shuyang Ling, Thomas Strohmer, and Ke Wei. Rapid, robust, and reliable blind deconvolution via nonconvex optimization, 2016.

[3] Horacio E. Fortunato and Manuel M. Oliveira. Fast high-quality non-blind deconvolution using sparse adaptive priors. *The Visual Computer*, 30(6-8):661–671, 2014.