Proximity Algorithms and Its Applications in Image Processing

Lixin Shen

Department of Mathematics
Department of Electrical Engineering and Computer Science
Syracuse University, USA

E-mail: lshen03@syr.edu

Proximity Algorithms and Its Applications in Image Processing

Lixin Shen

Department of Mathematics

Department of Electrical Engineering and Computer Science

Syracuse University, USA

E-mail: lshen03@syr.edu

Proximity Algorithms and Its Applications in Image Processing

Lixin Shen

Department of Mathematics
Department of Electrical Engineering and Computer Science
Syracuse University, USA

E-mail: lshen03@syr.edu

Why Proximity Algorithms?

- Many inference tasks in data analysis are now routinely formulated as composite convex optimization problems.
- The composite functions often comprise non-smooth regularization functions that capture structural assumptions about the inference tasks and for which proximity operators are readily available.
- The resulting optimization problems are then amenable to modern proximal algorithms.

Why Proximity Algorithms?

- Many inference tasks in data analysis are now routinely formulated as composite convex optimization problems.
- The composite functions often comprise non-smooth regularization functions that capture structural assumptions about the inference tasks and for which proximity operators are readily available.
- The resulting optimization problems are then amenable to modern proximal algorithms.

Why Proximity Algorithms?

- Many inference tasks in data analysis are now routinely formulated as composite convex optimization problems.
- The composite functions often comprise non-smooth regularization functions that capture structural assumptions about the inference tasks and for which proximity operators are readily available.
- The resulting optimization problems are then amenable to modern proximal algorithms.

Incomplete Image Recovery

Incomplete Image Data

$\Downarrow$

Recovery/Reconstruction scheme $\Leftarrow$ A priori knowledge

$\Downarrow$

Processing algorithm $\Leftarrow$ Proximity algorithm

$\Downarrow$

Recovered/reconstructed image

Example: Incomplete image data

$$\begin{cases} \mathbf{P}_\Lambda \mathbf{f} = \mathbf{P}_\Lambda \mathbf{g}, & \Longleftarrow \text{available data in image domain} \\ \mathbf{P}_\Gamma \mathbf{A} \mathbf{f} = \mathbf{P}_\Gamma \mathbf{x}. & \Longleftarrow \text{available data in transform domain} \end{cases}$$

where

$\mathbf{f}$:the original image in $\mathbb{R}^N$

$\mathbf{g}$:the observed image in $\mathbb{R}^N$

$\mathbf{A}$:an $M \times N$ transform matrix satisfying $\mathbf{A}^T\mathbf{A} = \mathbf{I}$

$\mathbf{x}$:the transform of $\mathbf{f}$ under the transformation $\mathbf{A}$ in $\mathbb{R}^M$

$$\mathbf{P}_\Lambda[i,j] = \begin{cases} 1 & \text{if } i = j \in \Lambda \subset \mathcal{N} := \{1, \ldots, N\}, \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbf{P}_\Gamma[i,j] = \begin{cases} 1 & \text{if } i = j \in \Gamma \subset \mathcal{M} = \{1, \ldots, M\}, \\ 0 & \text{otherwise.} \end{cases}$$

Example: Incomplete image data

$$\begin{cases} \mathbf{P}_\Lambda \mathbf{f} = \mathbf{P}_\Lambda \mathbf{g}, & \Longleftarrow \text{available data in image domain} \\ \mathbf{P}_\Gamma \mathbf{A} \mathbf{f} = \mathbf{P}_\Gamma \mathbf{x}. & \Longleftarrow \text{available data in transform domain} \end{cases}$$

Motivated by the following identity

$$\mathbf{f} = (\mathbf{I} - \mathbf{P}_\Lambda)\mathbf{A}^T \big(\underbrace{\mathbf{P}_\Gamma \mathbf{x} + (\mathbf{I} - \mathbf{P}_\Gamma)\mathbf{A}\mathbf{f}} \big) + \mathbf{P}_\Lambda \mathbf{g},$$

we proposed the following iterative Algorithm:

$$\mathbf{f}^{(n+1)} = (\mathbf{I} - \mathbf{P}_\Lambda)\mathbf{A}^T \underbrace{\mathrm{Soft}_{\mathbf{u}}\big(\mathbf{P}_\Gamma \mathbf{x} + (\mathbf{I} - \mathbf{P}_\Gamma)\mathbf{A}\mathbf{f}^{(n)}\big)} + \mathbf{P}_\Lambda \mathbf{g}$$

References: Chan-Chan-Shen-Shen (SISC03, LAA03), Chan-Riemenschneider-Shen-Shen (ACHA04), Cai-Chan-Shen-Shen (SISC08, ACM 09, NM09)

Proximity operators

Definition (Moreau, 1962)

For a proper, lower semi-continuous, convex function f, the proximity operator of f at x is

$$\mathrm{prox}_f(x) := \arg\min \left\{ \frac{1}{2}\|u - x\|_2^2 + f(u) : u \in \mathbb{R}^d \right\}.$$

The Moreau envelope of f at x is

$$\mathrm{env}_f(x) = \frac{1}{2}\|\mathrm{prox}_f(x) - x\|_2^2 + f(\mathrm{prox}_f(x))$$

Definition

For a proper, lower semi-continuous, convex function f, the proximity operator of f w.r.t a given matrix $H \in S_+^d$ at x is

$$\mathrm{prox}_{f,H}(x) := \arg\min \left\{ \frac{1}{2}\|u - x\|_H^2 + f(u) : u \in \mathbb{R}^d \right\}.$$

Examples of proximity operators

Indicate function

If $C \subset \mathbb{R}^n$ is closed and convex, define

$$\iota_C(x) := \begin{cases} 0, & x \in C, \\ +\infty, & x \notin C, \end{cases}$$

then $\mathrm{prox}_{\iota_C}(x) = P_C(x)$, the projection on to C.

The ℓ_1-norm

If $f(\cdot) := \lambda \| \cdot \|_1$, then

$$\mathrm{prox}_f(x) = \max\{0, |x| - \lambda\} \mathrm{sign}(x).$$

The closed-form formulation of the proximity operator is useful in developing efficient numerical algorithm.

Proximity Operators

Many common convex functions in data processing (image recovery, compressive sensing, machine learning, statistics) have explicit proximity operators:

- nuclear norm
- Huber's function
- elastic net regularizer
- hinge loss
- distance function
- Vapnik's ϵ-insensitive loss

Revisit the previous iterative algorithm

The iterative algorithm

$$\mathbf{f}^{(n+1)} = (\mathbf{I} - \mathbf{P}_\Lambda)\mathbf{A}^T \mathrm{Soft}_{\mathbf{u}}\left(\mathbf{P}_\Gamma \mathbf{x} + (\mathbf{I} - \mathbf{P}_\Gamma)\mathbf{A}\mathbf{f}^{(n)}\right) + \mathbf{P}_\Lambda \mathbf{g}$$

can be re-written as follows

$$\mathbf{f}^{(n+1)} = \mathrm{prox}_{\iota_{\mathcal{I}}}\left(\mathbf{f}^{(n)} - \nabla \mathrm{env}_\xi(\mathbf{A}\mathbf{f}^{(n)})\right)$$

where

- $\mathcal{I} := \left\{\mathbf{f} \in \mathbb{R}^N : \mathbf{P}_\Lambda \mathbf{f} = \mathbf{P}_\Lambda \mathbf{g}\right\}$
- $\mathcal{T} := \left\{\mathbf{y} \in \mathbb{R}^M : \mathbf{P}_\Gamma \mathbf{y} = \mathbf{P}_\Gamma \mathbf{T}_{\mathbf{u}} \mathbf{x}\right\}$, and
- $\xi = \|\mathrm{diag}(\mathbf{u}) \cdot \|_1 + \iota_{\mathcal{T}}$.

Thus our iterative algorithm is just the forward-backward algorithm for

$$\min_{\mathbf{f} \in \mathcal{I}} \mathrm{env}_\xi(\mathbf{A}\mathbf{f}) \quad = \quad \min_{\mathbf{f} \in \mathcal{I}} \left\{ \min_{\mathbf{y} \in \mathcal{T}} \left\{ \frac{1}{2}\|\mathbf{A}\mathbf{f} - \mathbf{y}\|_2^2 + \|\mathrm{diag}(\mathbf{u})\mathbf{y}\|_1 \right\} \right\}$$

Proximity operator is an effective tool

$$\boxed{\inf_u J(u)}$$

$$\Updownarrow$$

$$\boxed{0 \in \partial J(u^\star)}$$

$$\Updownarrow$$

$$\boxed{u^\star = \mathrm{prox}_{\lambda J}(u^\star)} \quad \equiv \quad \boxed{(I - \mathrm{prox}_{\lambda J})(u^\star) = 0}$$

Note: $\nabla \mathrm{env}_{\lambda J} = \frac{1}{\lambda}(I - \mathrm{prox}_{\lambda J})$. Mathematical optimization is about the strategies of splitting the monotone operator ∂J.

Convex problems

Two block convex problems

$$\min\{\varphi(x) + \psi(Bx) : x \in \mathbb{R}^n\}$$

where

- B is an $m \times n$ matrix
- φ and ψ are proper, l.s.c., and convex
- Difficulty: φ or ψ is non-differentiable

The characterization

Proximity characterization

If $x \in \mathbb{R}^n$ is a solution of the problem, then for any $P \in \mathbb{S}^n_+$ and $Q \in \mathbb{S}^m_+$ there exists a vector $y \in \mathbb{R}^m$ such that

$$\begin{aligned} x &= \operatorname{prox}_{\varphi,P}(x - P^{-1}B^\top y), \\ y &= \operatorname{prox}_{\psi^*,Q}(y + Q^{-1}Bx). \end{aligned}$$

Conversely, if there exist $P \in \mathbb{S}^n_+$, $Q \in \mathbb{S}^m_+$, $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ satisfying the above equations, then x is a solution of the problem

- $\psi^*(y) := \sup\{\langle y, z\rangle - \psi(z) : z \in \mathbb{R}^m\}$ conjugate function of ψ^*

Fixed-point characterization

$$\min\{\varphi(x) + \psi(Bx) : x \in \mathbb{R}^n\}$$

A fixed-point characterization

$$v = \mathcal{T} \circ E(v),$$

where

- $v := (x, y)$
- $\mathcal{T}(v) := (\mathrm{prox}_{\alpha\varphi}(x), \mathrm{prox}_{\beta\psi^*}(y))$
- $E := \begin{bmatrix} I & -\alpha B^\top \\ \beta B & I \end{bmatrix}$

Fixed-point iteration

It is naturally to perform fixed-point iteration

$$v^{k+1} = \mathcal{T} \circ E(v^k)$$

Convergence analysis:

- $\mathcal{T}$ is firmly nonexpansive
- $\|E\|_2 > 1$
- $\mathcal{T} \circ E$ is not necessarily nonexpansive

The sequence v^k may not converge

A simple observation

Finding fixed points of the operator $E : x \to -2x$.

Scheme 1: Consider the following implicit scheme

$$x_{n+1} = -2x_n.$$

Then $x_{n+1} = (-2)^n x_0$ and $|x_n| \to \infty$ as $n \to \infty$ if $x_0 \neq 0$.

Scheme 2: Consider the following implicit scheme

$$x_{n+1} = -x_{n+1} - x_n.$$

Then $x_{n+1} = (-1/2)^n x_0$ and $x_n \to 0$ as $n \to \infty$.

Scheme 3: We propose the following iterative scheme

$$x_{n+1} = -x_n - x_{n-1}.$$

Given $x_0 = x_1 = 1$, then $x_2 = -2$, $x_3 = 1$, $x_4 = 1$, $x_5 = -2$.

A simple observation

Finding fixed points of the operator $E : x \to -2x$.

Scheme 1: Consider the following implicit scheme

$$x_{n+1} = -2x_n.$$

Then $x_{n+1} = (-2)^n x_0$ and $|x_n| \to \infty$ as $n \to \infty$ if $x_0 \neq 0$.

Scheme 2: Consider the following implicit scheme

$$x_{n+1} = -x_{n+1} - x_n.$$

Then $x_{n+1} = (-1/2)^n x_0$ and $x_n \to 0$ as $n \to \infty$.

Scheme 3: We propose the following iterative scheme

$$x_{n+1} = -x_n - x_{n-1}.$$

Given $x_0 = x_1 = 1$, then $x_2 = -2$, $x_3 = 1$, $x_4 = 1$, $x_5 = -2$.

A simple observation

Finding fixed points of the operator $E : x \to -2x$.

Scheme 1: Consider the following implicit scheme

$$x_{n+1} = -2x_n.$$

Then $x_{n+1} = (-2)^n x_0$ and $|x_n| \to \infty$ as $n \to \infty$ if $x_0 \neq 0$.

Scheme 2: Consider the following implicit scheme

$$x_{n+1} = -x_{n+1} - x_n.$$

Then $x_{n+1} = (-1/2)^n x_0$ and $x_n \to 0$ as $n \to \infty$.

Scheme 3: We propose the following iterative scheme

$$x_{n+1} = -x_n - x_{n-1}.$$

Given $x_0 = x_1 = 1$, then $x_2 = -2$, $x_3 = 1$, $x_4 = 1$, $x_5 = -2$.

One-step proximity algorithms

Key idea: Choose an appropriate matrix M and decompose the matrix E as

$$E = (E - M) + M$$

We then have

$$v = \mathcal{T} \circ E(v)$$
$$\Updownarrow$$
$$v = \mathcal{T} \circ ((E - M)v + Mv)$$
$$\Downarrow$$
$$v^{k+1} = \mathcal{T} \circ ((E - M)v^{k+1} + Mv^{k})$$

When $E - M$ is a strictly block upper (or lower) triangular matrix, it leads to an explicit iterative scheme.

One-step proximity algorithms

Key idea: Choose an appropriate matrix M and decompose the matrix E as

$$E = (E - M) + M$$

We then have

$$v = \mathcal{T} \circ E(v)$$

$$\Updownarrow$$

$$v = \mathcal{T} \circ ((E - M)v + Mv)$$

$$\Downarrow$$

$$v^{k+1} = \mathcal{T} \circ ((E - M)v^{k+1} + Mv^{k})$$

When $E - M$ is a strictly block upper (or lower) triangular matrix, it leads to an explicit iterative scheme.

One-step proximity algorithms

Key idea: Choose an appropriate matrix M and decompose the matrix E as

$$E = (E - M) + M$$

We then have

$$v = \mathcal{T} \circ E(v)$$
$$\Updownarrow$$
$$v = \mathcal{T} \circ ((E - M)v + Mv)$$
$$\Downarrow$$
$$v^{k+1} = \mathcal{T} \circ ((E - M)v^{k+1} + Mv^k)$$

When $E - M$ is a strictly block upper (or lower) triangular matrix, it leads to an explicit iterative scheme.

One-step proximity algorithms

Key idea: Choose an appropriate matrix M and decompose the matrix E as

$$E = (E - M) + M$$

We then have

$$v = \mathcal{T} \circ E(v)$$
$$\Updownarrow$$
$$v = \mathcal{T} \circ ((E - M)v + Mv)$$
$$\Downarrow$$
$$v^{k+1} = \mathcal{T} \circ ((E - M)v^{k+1} + Mv^k)$$

When $E - M$ is a strictly block upper (or lower) triangular matrix, it leads to an explicit iterative scheme.

One-step proximity algorithms

Key idea: Choose an appropriate matrix M and decompose the matrix E as

$$E = (E - M) + M$$

We then have

$$v = \mathcal{T} \circ E(v)$$
$$\Updownarrow$$
$$v = \mathcal{T} \circ ((E - M)v + Mv)$$
$$\Downarrow$$
$$v^{k+1} = \mathcal{T} \circ ((E - M)v^{k+1} + Mv^k)$$

When $E - M$ is a strictly block upper (or lower) triangular matrix, it leads to an explicit iterative scheme.

Convergence analysis (Li-Shen-Xu-Zhang 14)

Let v^* be a fixed-point of $\mathcal{T} \circ E$, x^* be a solution of the convex optimization problem and

$$R := \mathrm{diag}\,(P, Q)\,.$$

Convergence Theorem

If RM is symmetric positive definite, then

- $v^k \to v^*$,
- $x^k \to x^*$.

Special cases

The fixed-point proximity iteration

$$v^{k+1} = \mathcal{T}((E - M)v^{k+1} + Mv^k),$$

	matrix M	condition	RM
SBIA/ADMM	$\begin{bmatrix} \beta B^\top B & B^\top \\ \beta B & I_m \end{bmatrix}$	$\beta > 0$	S.P.S-D
Linearized-ADMM	$\begin{bmatrix} S + \alpha\beta B^\top B & \alpha B^\top \\ \beta B & I_m \end{bmatrix}$	S is S.P.D	S.P.D
PDEA	$\begin{bmatrix} I_n & \alpha B^\top \\ \beta B & I_m \end{bmatrix}$	$\alpha\beta < \frac{1}{\|B\|^2}$	S.P.D
Doubley ALM	$\begin{bmatrix} \gamma I_n + \alpha\beta B^\top B & \alpha B^\top \\ \beta B & (1+\gamma) I_m \end{bmatrix}$	$\gamma > 0$	S.P.D

SBIA: split Bregman iteration; PDEA: Primal-dual extrapolation algorithm; ALM: Augmented Lagrangian Method

These algorithms choose different M

Special cases

The fixed-point proximity iteration

$$v^{k+1} = \mathcal{T}((E - M)v^{k+1} + Mv^k),$$

	matrix M	condition	RM
SBIA/ADMM	$\begin{bmatrix} \beta B^\top B & B^\top \\ \beta B & I_m \end{bmatrix}$	$\beta > 0$	S.P.S-D
Linearized-ADMM	$\begin{bmatrix} S + \alpha\beta B^\top B & \alpha B^\top \\ \beta B & I_m \end{bmatrix}$	S is S.P.D	S.P.D
PDEA	$\begin{bmatrix} I_n & \alpha B^\top \\ \beta B & I_m \end{bmatrix}$	$\alpha\beta < \frac{1}{\|B\|^2}$	S.P.D
Doublely ALM	$\begin{bmatrix} \gamma I_n + \alpha\beta B^\top B & \alpha B^\top \\ \beta B & (1+\gamma) I_m \end{bmatrix}$	$\gamma > 0$	S.P.D

SBIA: split Bregman iteration; PDEA: Primal-dual extrapolation algorithm; ALM: Augmented Lagrangian Method

These algorithms choose different M

Two-step iteration

The two-step iteration scheme uses iterates of two previous steps to compute the current step.

$$v = \mathcal{T} \circ E(v)$$

$$\Updownarrow$$

$$v = \mathcal{T} \circ ((E - M_0)v + M_1 v + M_2 v)$$

$$\text{where } M_0 = M_1 + M_2$$

$$\Downarrow$$

$$v^{k+1} = \mathcal{T} \circ ((E - M_0)v^{k+1} + M_1 v^k + M_2 v^{k-1})$$

Two-step iteration

The two-step iteration scheme uses iterates of two previous steps to compute the current step.

$$v = \mathcal{T} \circ E(v)$$

$$\Updownarrow$$

$$v = \mathcal{T} \circ ((E - M_0)v + M_1 v + M_2 v)$$

$$\text{where } M_0 = M_1 + M_2$$

$$\Downarrow$$

$$v^{k+1} = \mathcal{T} \circ ((E - M_0)v^{k+1} + M_1 v^k + M_2 v^{k-1})$$

Two-step iteration

The two-step iteration scheme uses iterates of two previous steps to compute the current step.

$$v = \mathcal{T} \circ E(v)$$

$$\Updownarrow$$

$$v = \mathcal{T} \circ ((E - M_0)v + M_1 v + M_2 v)$$

$$\text{where } M_0 = M_1 + M_2$$

$$\Downarrow$$

$$v^{k+1} = \mathcal{T} \circ ((E - M_0)v^{k+1} + M_1 v^k + M_2 v^{k-1})$$

Convergence analysis

$$v^{k+1} = \mathcal{T} \circ ((E - M_0)v^{k+1} + M_1 v^k + M_2 v^{k-1})$$

Convergence Theorem

If the following conditions are satisfied:

- $M_0 = M_1 + M_2$
- $H := R(M_0 + M_2)$ is symmetric positive definite
- $\|H^{-1/2} R M_2 H^{-1/2}\|_2 < \frac{1}{2}$,

then the sequence $\{v^k : k \in \mathbb{N}\}$ generated by the two-step iteration converges to a fixed-point of the map $\mathcal{T} \circ E$.

New algorithms

In the two-step iteration scheme

$$v^{k+1} = \mathcal{T}((E - M_0)v^{k+1} + M_1 v^k + M_2 v^{k-1})$$

choosing

$$M_0 = \begin{bmatrix} \alpha I_n & -B^\top \\ -\theta B & \beta I_m \end{bmatrix}, \ M_1 = \begin{bmatrix} \alpha \mathcal{I}_n & (\theta - 2)B^\top \\ -\theta B & \beta I_m \end{bmatrix},$$

$$M_2 = \begin{bmatrix} 0 & (1-\theta)B^\top \\ 0 & 0 \end{bmatrix}$$

leads to

$$\begin{cases} x^{k+1} &= \operatorname{prox}_{\alpha\varphi}(x^k - \alpha B^\top \bar{y}^k), \\ \bar{x}^{k+1} &= x^{k+1} + \theta(x^{k+1} - x^k), \\ y^{k+1} &= \operatorname{prox}_{\beta\psi^*}(y^k + \beta B \bar{x}^{k+1})), \\ \bar{y}^{k+1} &= y^{k+1} + (1-\theta)(y^{k+1} - y^k). \end{cases}$$

New algorithms

Convergence analysis:

For $\theta \in [0,1]$, if one of the following statements hold

- $\alpha, \beta > 2(1-\theta)\|B\|_2$ and
  $\theta(\alpha - 2(1-\theta)\|B\|_2)(\beta - 2(1-\theta)\|B\|_2) < \frac{1}{\|B\|_2^2}$;
- $\alpha, \beta > 2\theta\|B\|_2$ and
  $((1-\theta)\alpha - 2(\theta)\|B\|_2)(\beta - 2\theta\|B\|_2) < \frac{1}{\|B\|_2^2}$;

then

- M_0, M_1, M_2 satisfy condition-M
- $\{x^k : k \in \mathbb{N}\}$ converges to a solution of the original problem

What is Image Inpainting?

- Image inpainting refers to the process of restoring missing or damaged areas in an image.
- Image inpainting is an ill-posed inverse problem that has no well-defined unique solution.

All methods for image impainting are guided by the assumption that

- pixels in the known and unknown parts of the image share the same statistical properties or geometric structures.

The goal of image inpainting: an inpainted image is as physically plausible and visually pleasing as possible.

Methods for Image Inpainting

- *Diffusion-based inpainting*: introducing smoothness priors via PDEs to propagate (or diffuse) local structures from the exterior to the interior of the regions to be inpainted. Suitable for completing lines, curves, and small regions, not for recovering textures of large areas.

- *Examplar-based methods*: refers to methods that synthesize entire patches by learning from patches in the known part of the image.

- *Sparse priors for inpainting*: The image or the patch is assumed to be sparse in a given basis. Known and unknown parts of the image are assumed to share the same sparse representation.

Methods for Image Inpainting

- *Diffusion-based inpainting*: introducing smoothness priors via PDEs to propagate (or diffuse) local structures from the exterior to the interior of the regions to be inpainted. Suitable for completing lines, curves, and small regions, not for recovering textures of large areas.

- *Examplar-based methods*: refers to methods that synthesize entire patches by learning from patches in the known part of the image.

- *Sparse priors for inpainting*: The image or the patch is assumed to be sparse in a given basis. Known and unknown parts of the image are assumed to share the same sparse representation.

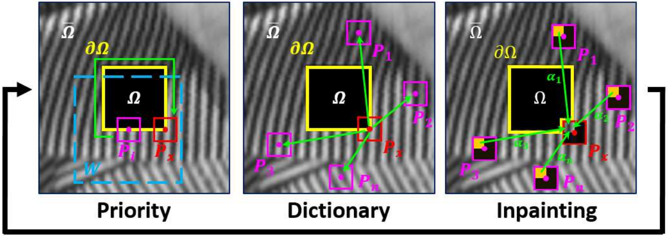
Methods for Image Inpainting

- *Diffusion-based inpainting*: introducing smoothness priors via PDEs to propagate (or diffuse) local structures from the exterior to the interior of the regions to be inpainted. Suitable for completing lines, curves, and small regions, not for recovering textures of large areas.

- *Examplar-based methods*: refers to methods that synthesize entire patches by learning from patches in the known part of the image.

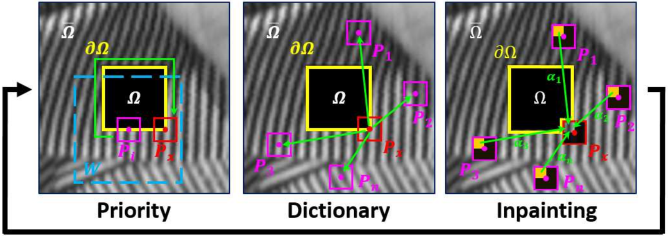
- *Sparse priors for inpainting*: The image or the patch is assumed to be sparse in a given basis. Known and unknown parts of the image are assumed to share the same sparse representation.

Framework of Patch-Based Inpainting



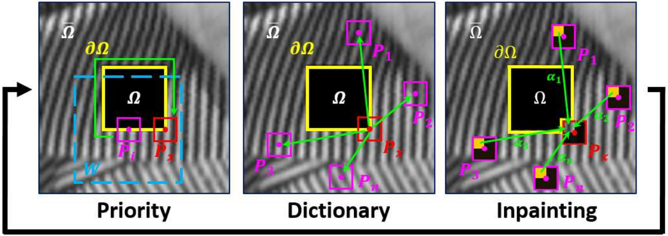
- Find a patch with the highest priority on the boundary
- Construct a dictionary from the selected similar patches
- Fill the unknown pixels through an optimization model

Framework of Patch-Based Inpainting



- Find a patch with the highest priority on the boundary
- Construct a dictionary from the selected similar patches
- Fill the unknown pixels through an optimization model

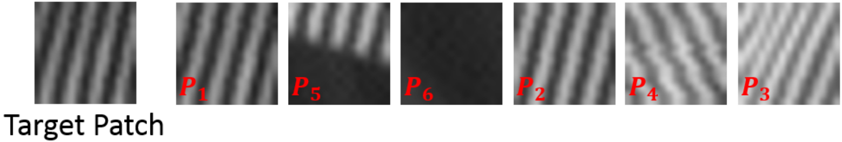
Framework of Patch-Based Inpainting



- Find a patch with the highest priority on the boundary
- Construct a dictionary from the selected similar patches
- Fill the unknown pixels through an optimization model

Proposed Scheme for Adaptive Dictionary

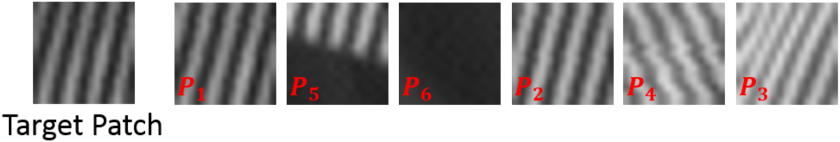
We use a metric to measure the similarity between the patch P_j and the target patch P_x. The sum of squared differences by the ℓ_2 norm (SSDL2), is defined as $SSDL2(P_x, P_j) = \|P_x - P_j\|_2$.



- From left to right the similarities measured by SSDL2 are 5.0×10^2, 8.1×10^2, 8.9×10^2, 1.1×10^3, 1.2×10^3, and 1.3×10^3, respectively;
- By the SSDL2 index, the uniform patch P_6 is more similar to the target patch than the P_2 patch, which is not consistent with our human visual system.

Proposed Scheme for Adaptive Dictionary

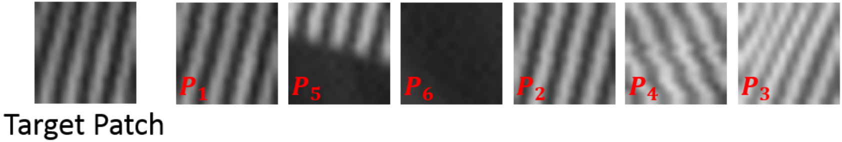
We use a metric to measure the similarity between the patch P_j and the target patch P_x. The sum of squared differences by the ℓ_2 norm (SSDL2), is defined as $SSDL2(P_x, P_j) = \|P_x - P_j\|_2$.



- From left to right the similarities measured by SSDL2 are 5.0×10^2, 8.1×10^2, 8.9×10^2, 1.1×10^3, 1.2×10^3, and 1.3×10^3, respectively;
- By the SSDL2 index, the uniform patch P_6 is more similar to the target patch than the P_2 patch, which is not consistent with our human visual system.

Proposed Scheme for Adaptive Dictionary

We use a metric to measure the similarity between the patch P_j and the target patch P_x. The sum of squared differences by the ℓ_2 norm (SSDL2), is defined as $SSDL2(P_x, P_j) = \|P_x - P_j\|_2$.



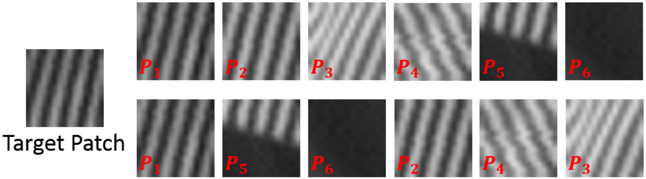
- From left to right the similarities measured by SSDL2 are 5.0×10^2, 8.1×10^2, 8.9×10^2, 1.1×10^3, 1.2×10^3, and 1.3×10^3, respectively;
- By the SSDL2 index, the uniform patch P_6 is more similar to the target patch than the P_2 patch, which is not consistent with our human visual system.

Patch in DCT-Haar Tight Frame System

A similarity metric between target patch P_x and the similar patch P_j is proposed to be the ℓ_2 difference of Laplace probability distributions (LPDL2) of the patches in DCT-Haar domain as

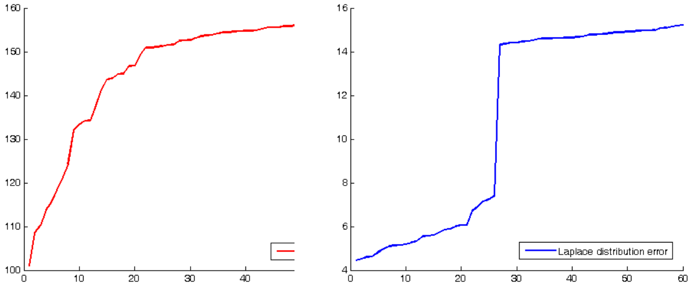
$$\mathrm{LPDL2}(P_x, P_j) = \|\tilde{P}_x - \tilde{P}_j\|_2, \tag{1}$$

where $\tilde{P}_x$ and $\tilde{P}_j$ are probability distributions of P_x and P_j, respectively.



- Similar patches sorted by two different metrics. In the first row, the similarities measured in DCT-Haar domain by LPDL2; In the second row, the similarities measured by SSDL2.

Dictionary by DCT-Haar Domain



- The patches found by SSDL2 in the left image can be efficiently separated into two groups by LDPL2 in the right image.

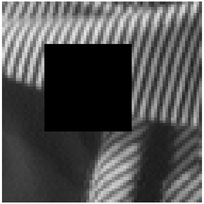
Patch Inpainting Model

For a dictionary based on patches $D \in R^{m \times n}$, we need to get sparse coefficients $\alpha \in R^n$ to approximate the observed patch $x \in R^m$ by the following optimization model

$$\min_{\alpha} \left\{ \frac{1}{2}\|D\alpha - x\|_2^2 + \lambda(\|\alpha\|_1 + \frac{\gamma}{2}\|\alpha\|_2^2), \text{ s.t. } a^\top \alpha = 1 \right\} \quad (2)$$

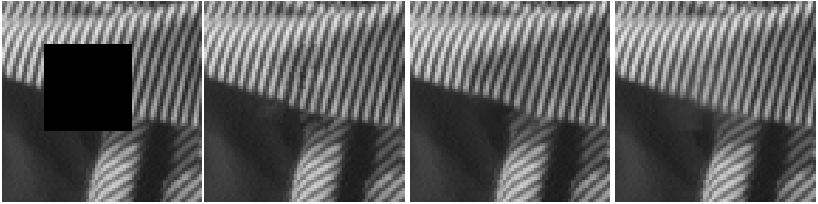
where λ is a nonnegative parameter and $a = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$.

Inpainting Result



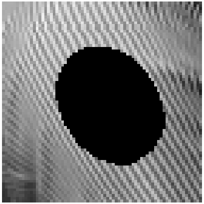
- Observed image with a 31x31 missing block;
- The results from left to right are by Xu's Alg., Photoshop and Our Alg., respectively.

Inpainting Result



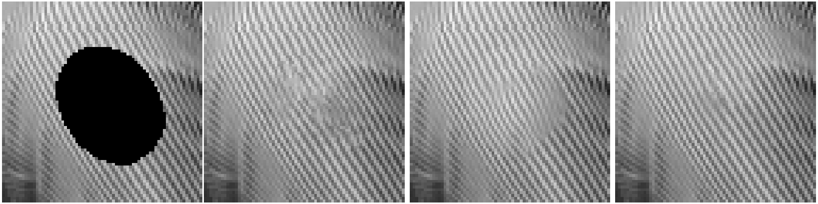
- Observed image with a 31x31 missing block;
- The results from left to right are by Xu's Alg., Photoshop and Our Alg., respectively.

Inpainting Result



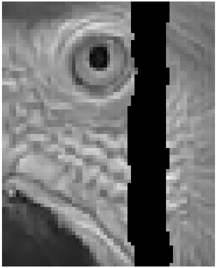
- Observed image with 1224 missing pixels;
- The results from left to right are by Xu's Alg., Photoshop and Our Alg., respectively.

Inpainting Result



- Observed image with 1224 missing pixels;
- The results from left to right are by Xu's Alg., Photoshop and Our Alg., respectively.

Inpainting Result



- Observed image;
- The results from left to right are by Photoshop and Our Alg., respectively.

Inpainting Result

- Observed image;
- The results from left to right are by Photoshop and Our Alg., respectively.

pMRI Problem

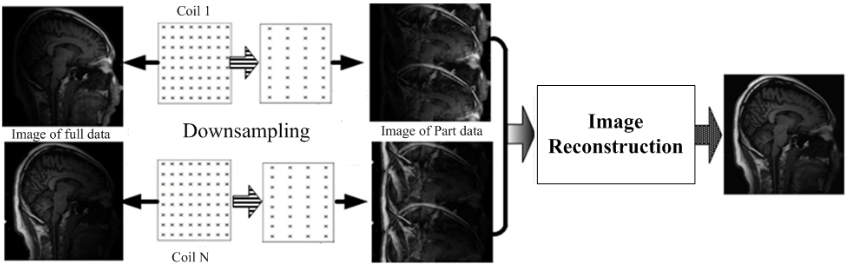


Figure: Multi-coil Parallel Magnetic Resonance Imaging (pMRI) Problem by subsampling parts of k-space date to accelerate imaging.

1.2 Observation Model

The coil image from the ℓ-th coil is modeled as follows:

$$g_\ell = \mathcal{F}^{-1}\mathcal{P}\mathcal{F}\mathcal{S}_\ell u + \eta_\ell, \tag{3}$$

where

- u is the desired image
- η_ℓ is the additive noise
- $\mathcal{F}$ is the discrete Fourier transform matrix
- $\mathcal{P}$, called sampling matrix, is a diagonal matrix with 0 and 1
- $\mathcal{S}_\ell$ are sensitivies and have to be pre-estimated accurately in real applications.

Combining equations from all p coils, the set of the above equations can be transformed into matrix form:

$$g = Mu + \eta, \tag{4}$$

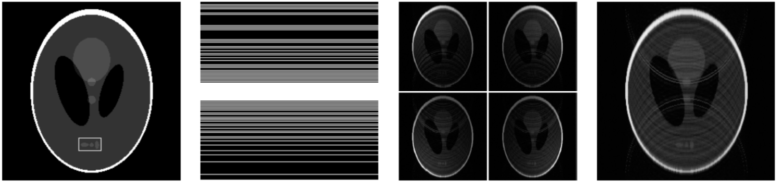
The Proposed Model

The proposed model is

$$\min_u \left\{ \frac{1}{2}\|Mu - g\|_2^2 + \|\Gamma W u\|_1 \right\},$$

where W is formed from the directional Haar framelet (Li-Chan-Shen-Hsu-Tseng SIIS16).

Experiments on Simulated Data



(a) (b) (c) (d)

Figure: (a) The Shepp-Logan phantom (the rectangle is for later zoom-in comparison); (b) Sampling model of the 33% k-space; (c) Four coil images with Gaussian noise $\sigma = 0.01$; (d) The SoS image from the 33% k-space.

Experiments on Simulated Data (4 Coils)

Table: The NMSE indexes and the CPU usage for the simulation data.

pMRI Alg.	TV-based Alg.	$\text{FA}\mathbb{C}\text{TF}_6\text{A}$	FAHaarA	FADHFA
NMSE	8.2×10^{-4}	2.6×10^{-3}	3.6×10^{-4}	2.19×10^{-4}
CPU Time	12.3s	48.2s	9.1s	9.0s

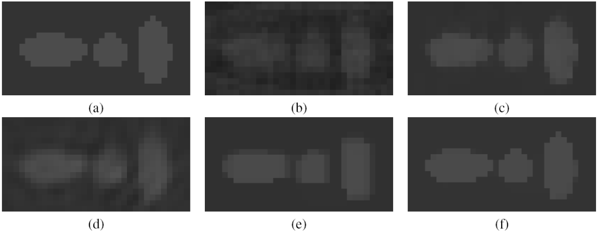


Figure: (a) Original; (b) SoS image from the 33% k-space; (c) the TV alg. with λ =0.005; (d) $\text{FA}\mathbb{C}\text{TF}_6\text{A}$; (e) FAHaarA; and (f) FADHFA.

Experiments on MRI Phantoms (4 Coils)

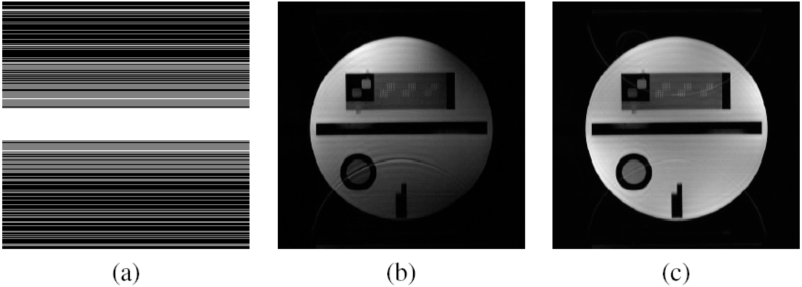
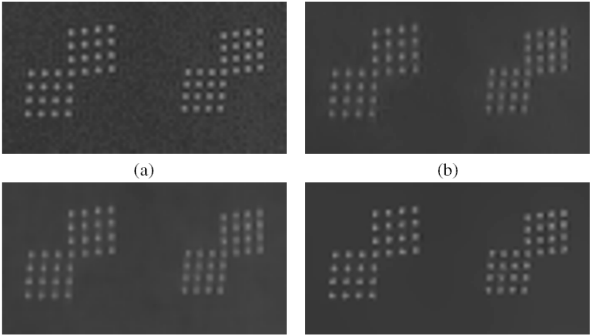


Figure: (a) Sampling model of 33% k-space; (b) One coil image from the 33% k-space; (c) SoS image from the 33% k-space.

Experiments on MRI Phantoms (4 Coils)

Table: First number in parentheses: the CNR (contrast-to-noise ratio) values for the regions marked;Second number in parentheses: the CPU time in seconds.

% of k-space	ℓ_1-ESPIRiT	TV-based Alg.	Our FADHFA
33%	(65.0, 255s)	(73.3, 65s)	(124.4, 28s)



(a) (b) (c) (d)

Figure: (a) SoS image of the full k-space; (b) ℓ_1-ESPIRiT with parameter

Experiments on In-vivo Data (12 Coils)

Table: The CNR values at four different square regions marked and CPU time (in seconds) by pMRI algorithms.

pMRI Alg.	Region **1**	Region **2**	Region **3**	Region **4**	Time
ℓ_1-ESPIRiT	39.21	45.19	34.61	40.39	293s
TV. Alg.	59.03	68.82	66.99	84.51	108s
Our FADHFA	192.17	223.06	217.45	272.70	50s

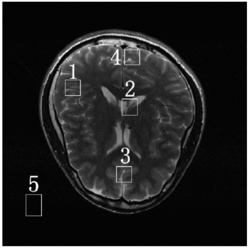


Figure: (a) SoS image of the full k-space with zoom-in parts

Experiments on In-vivo Data (12 Coils)

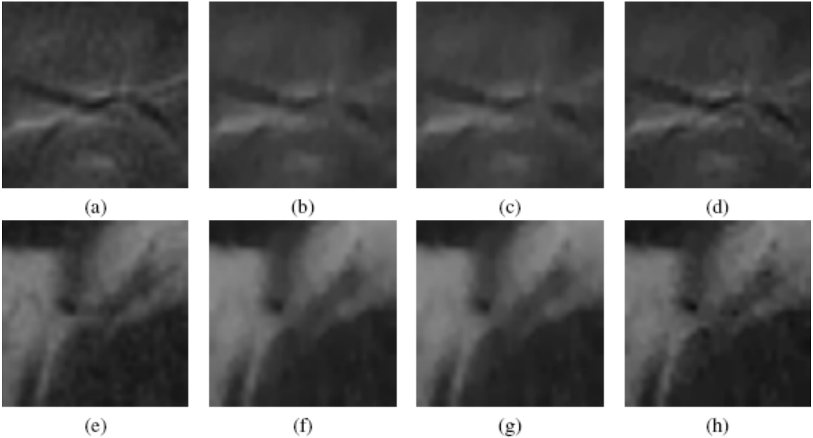


Figure: The first column: Reference SoS image of the full k-space. The reconstructed images of the second, third and forth columns by ℓ_1-ESPIRiT , TV regularization algorithm and our FADHFA on 33% full k-space data, respectively.

Singular value decomposotion

- The singular value decomposition (SVD) of a matrix A is the factorization of A into the product of three matrices $A = UDV^T$ where the columns of U and V are orthonormal and the matrix D is diagonal with nonnegative real entries.
- The SVD of a matrix can be formulated as solving a sequential optimization problems. The corresponding objective function has a bilinear from and therefore is nonconvex.
- Noncovex: bilinear and quasiconvex

Thank You !

Lixin Shen

E-mail:lshen03@syr.edu

Syracuse University, USA