# Superiorized Inversion of the Radon Transform

Gabor T. Herman

Graduate Center, City University of New York

March 28, 2017

# The Radon Transform in 2D

- For a function $f$ of two real variables, a real number $t$ and an angle $\theta \in [0,\pi)$, we define $[\mathscr{R}f](t,\theta)$ as the line integral

$$[\mathscr{R}f](t,\theta) = \int_{\mathbb{R}^1} f(t\cos\theta - s\sin\theta, t\sin\theta + s\cos\theta)\, ds. \tag{1}$$

- $\mathscr{R}f$ is the *Radon transform* of $f$.
- In applications of this concept, $f$ represents the distribution of some physical parameter that is 0-valued outside in a bounded subset of the plane (the *support* of $f$) and there is some instrument that provides estimates $b_l$ of $[\mathscr{R}f](t_l,\theta_l)$ for a collection of $L$ pairs $(t_l,\theta_l)$, for $1 \le l \le L$. This set of estimates is referred to as the *projection data*.
- We wish to recover the distribution from the projection data. Mathematically speaking, we wish to *reconstruct* the function $f$ from a noisy and incomplete set of its line integrals.
- While there are closed-form formulas for the mathematical *inverse* of the Radon transform, we cannot in practice expect a perfect recovery of $f$ for two essential reasons:
  - $[\mathscr{R}f](t,\theta)$ is not (and, in practice, it cannot) be known to us for all $(t,\theta)$ and
  - the mathematical formula for the inverse transform involves (in practice unavoidably) some limit process(es) and, in any actual implementation, we cannot go all the way to the limit implied by the mathematics.

## Series Expansion: An Alternative to Approximating the Inverse Radon Transform

- In he *series expansion methods* it is assumed that the function $f$ can be usefully approximated by a linear combination of a finite number of fixed *basis functions* and so the algorithmic task becomes to estimate the coefficients of the basis functions in the linear combination.
- In this talk we restrict ourselves to series expansion methods for which the basis functions are defined as follows.
  - the support of the function $f$ is divided into $J = M \times M$ small squares (*pixels*),
  - each pixel gives rise to a basis function whose value is 1 in its interior and 0 in its exterior.
- We use $x_j$ to denote the coefficient in the expansion of the basis function associated with the $j$th of the $J$ pixels.
- Defining $a_{l,j}$ as the length of intersection of the $l$th line with the $j$th pixel:

$$b_l \approx \sum_{j=1}^{J} a_{l,j} x_j. \tag{2}$$

- The $b_l$ are provided by the physical measurements, the $a_{l,j}$ are known from the geometry of the measuring instrument and the $x_j$ are what we need to estimate based on the approximate equalities in (2).

# Constrained Optimization

- An alternative notation for (2) is $\boldsymbol{b} \approx \mathbf{A}\boldsymbol{x}$.
- For any nonnegative real number $\varepsilon$, we say that a $J$-dimensional vector $\boldsymbol{x}$ is $\varepsilon$*-compatible* (with the $L$-dimensional *measurement vector* $\boldsymbol{b}$ and the $L \times J$ *system matrix* $\mathbf{A}$) if $\|\boldsymbol{b} - \mathbf{A}\boldsymbol{x}\|_2^2 \leq \varepsilon$.
- An $\varepsilon$-compatible solution is not necessarily a good one (even for a small $\varepsilon$), since it does not take into consideration any prior knowledge about the nature of the object that is being imaged.
- One approach to overcoming this problem is by using a *secondary criterion* $\phi$, such that $\phi(\boldsymbol{x})$ is a measure of the prior undesirability of $\boldsymbol{x}$.
- For example, a secondary criterion that has been used to distinguish the "better" constraints-compatible solutions is TV (*total variation*), whose value is small for images $\boldsymbol{x}$ that are nearly piecewise homogeneous.
- The problem then becomes: **Find**

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \phi(\boldsymbol{x}), \text{ subject to } \|\boldsymbol{b} - \mathbf{A}\boldsymbol{x}\|_2^2 \leq \varepsilon. \tag{3}$$

- Superiorization is a recently-developed heuristic for constrained optimization. Heuristic approaches have been found useful in applications of optimization, mainly because they are often computationally much less expensive than their exact counterparts, but nevertheless provide solutions that are appropriate for the application.

# The Idea of Superiorization

- In many applications there exist efficient iterative algorithms for producing *constraints-compatible* solutions (meaning: $\varepsilon$-compatible for a given $\varepsilon$).
- Often the algorithm is *perturbation resilient* in the sense that, even if certain kinds of changes are made at the end of each iterative step, the algorithm still produces a constraints-compatible solution.
- This property is exploited in *superiorization* by using such perturbations to steer the algorithm to an output that is as constraints-compatible as the output of the original algorithm, but it is superior to it according to a given secondary criterion.
- We present a totally automatic procedure that turns the iterative algorithm into such a superiorized version. The procedure is applicable to a very large class of iterative algorithms and secondary criteria.
- This can be significantly helpful in research, because it has the potential of saving a lot of time and effort for the researcher when the application of interest gives rise to a new constrained optimization problem.

# Constrained Optimization vs. Superiorization

- Superiorization has a world-view that is quite different from that of classical constrained optimization. Both in superiorization and in classical constrained optimization we assume the existence of domain $\Omega$ and a secondary criterion that is specified by a function $\phi$ that maps $\Omega$ into $\mathbb{R}$.
- In classical optimization it is assumed that there is a constraints set $C$ and the task is to find an $\boldsymbol{x} \in C$ for which $\phi(\boldsymbol{x})$ is minimal. Problems with this approach: (1) The constraints may not be consistent and so $C$ could be empty and the optimization task as stated would not have a solution. (2) Iterative methods of classical constrained optimization typically converge to a solution only in the limit. In practice some stopping rule is applied to terminate the process and the actual output at that time may not be in $C$ and, even if it is in $C$, it is most unlikely to be a minimizer of $\phi$ over $C$.
- Both problems are handled in the superiorization approach by replacing the constraints set $C$ by a nonnegative real-valued (*proximity*) function $\mathscr{P}r$ that serves as an indicator of how incompatible a given $\boldsymbol{x} \in \Omega$ is with the constraints. Then the merit of an actual output of an algorithm is given by the smallness of the two numbers $\mathscr{P}r(\boldsymbol{x})$ and $\phi(\boldsymbol{x})$. Roughly, if an iterative algorithm produces an output $\boldsymbol{x}$, then its superiorized version will produce an output $\boldsymbol{x}'$ for which $\mathscr{P}r(\boldsymbol{x}')$ is not larger than $\mathscr{P}r(\boldsymbol{x})$, but (in general) $\phi(\boldsymbol{x}')$ is much smaller than $\phi(\boldsymbol{x})$.

## Problem Structures

- We use $\Omega$ to denote a nonempty subset of $\mathbb{R}^J$. An example is

$$\Omega = \left\{ \boldsymbol{x} \in \mathbb{R}^J \mid 0 \leq \boldsymbol{x}_j \leq 1, \text{ for } 1 \leq j \leq J \right\}. \tag{4}$$

  This is reasonable if the $\boldsymbol{x}_j$ represent the x-ray linear attenuation coefficient in pixels, measured in cm$^{-1}$, to be obtained by CT scanning.
- In any application, there is a *problem set* $\mathbb{T}$; each problem $T \in \mathbb{T}$ is a description of the constraints in that case. For example, in CT a problem $T$ can be descried as $T = (\boldsymbol{A}, \boldsymbol{b})$, where $\boldsymbol{A}$ is the system matrix of the scanner and $\boldsymbol{b}$ is the vector of estimated line integrals obtained by the scanner.
- The notion of constraints-compatibility is formalized by a proximity function $\mathscr{P}r$ on $\mathbb{T}$ such that, for every $T \in \mathbb{T}$, $\mathscr{P}r_T : \Omega \to \mathbb{R}_+$. $\mathscr{P}r_T(\boldsymbol{x})$ is an indicator of how incompatible $\boldsymbol{x}$ is with the constraints of $T$. In CT, $\mathscr{P}r_T(\boldsymbol{x})$ should indicate by how much a proposed reconstruction $\boldsymbol{x}$ in $\Omega$ violates the constraints of the problem $T$ that are provided by the measurements taken by the scanner; a possible choice is the norm-distance

$$\mathscr{P}r_T(\boldsymbol{x}) = \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|. \tag{5}$$

- A *problem structure* is a pair $\langle \mathbb{T}, \mathscr{P}r \rangle$, where $\mathbb{T}$ is a nonempty problem set and $\mathscr{P}r$ is a proximity function on $\mathbb{T}$. For an $\boldsymbol{x} \in \Omega$, we say that $\boldsymbol{x}$ is $\varepsilon$-compatible with $T$ provided that $\mathscr{P}r_T(\boldsymbol{x}) \leq \varepsilon$.

## Algorithms

- We introduce the set $\Delta$, such that $\Omega \subseteq \Delta \subseteq \mathbb{R}^J$. Both $\Omega$ and $\Delta$ are assumed to be known and fixed for any particular problem structure $\langle \mathbb{T}, \mathscr{P}r \rangle$.
- An *algorithm* **P** for a problem structure $\langle \mathbb{T}, \mathscr{P}r \rangle$ assigns to each problem $T \in \mathbb{T}$ an operator $\mathbf{P}_T : \Delta \to \Omega$. For any initial point $\boldsymbol{x} \in \Omega$, **P** produces the infinite sequence $\left( (\mathbf{P}_T)^k \boldsymbol{x} \right)_{k=0}^{\infty}$ of points in $\Omega$.
- An example for the CT problem $T = (\boldsymbol{A}, \boldsymbol{b})$ is provided by the following iterative method (*ART*). Assume that $\boldsymbol{b}$ is an $I$-dimensional vector and that $\boldsymbol{a}^i \in \mathbb{R}^J$ is the transpose of the $i$th row of $\boldsymbol{A}$, for $1 \le i \le I$. We define $\mathbf{U}_i : \mathbb{R}^J \to \mathbb{R}^J$, $\mathbf{Q} : \mathbb{R}^J \to \Omega$ and the ART algorithmic operator $\mathbf{P}_T : \Omega \to \Omega$ by

$$\mathbf{U}_i(\boldsymbol{x}) = \boldsymbol{x} + \left( \left( \boldsymbol{b}_i - \left\langle \boldsymbol{a}^i, \boldsymbol{x} \right\rangle \right) / \left\| \boldsymbol{a}^i \right\|^2 \right) \boldsymbol{a}^i, \tag{6}$$

$$(\mathbf{Q}(\boldsymbol{x}))_j = \text{median value of} \left\{ 0, \boldsymbol{x}_j, 1 \right\}, \text{ for } 1 \le j \le J, \tag{7}$$

$$\mathbf{P}_T(\mathbf{x}) = \mathbf{Q}\mathbf{U}_I \cdots \mathbf{U}_2 \mathbf{U}_1(\mathbf{x}). \tag{8}$$

- Note that due to (7), $\mathbf{P}_T(\mathbf{x})$ is in the $\Omega$ of (4).

## Outputs

- For a problem structure $\langle \mathbb{T}, \mathscr{P}r \rangle$, a $T \in \mathbb{T}$, an $\varepsilon \in \mathbb{R}_+$ and a sequence $R = \left(\boldsymbol{x}^k\right)_{k=0}^{\infty}$ of points in $\Omega$, $O(T, \varepsilon, R)$ denotes the $\boldsymbol{x} \in \Omega$ with the following properties:
  - $\mathscr{P}r_T(\boldsymbol{x}) \leq \varepsilon$ and
  - there is a nonnegative integer $K$ such that
    - $\boldsymbol{x}^K = \boldsymbol{x}$ and
    - for all $k < K$, $\mathscr{P}r_T\left(\boldsymbol{x}^k\right) > \varepsilon$.
- If there is such an $\boldsymbol{x}$, then it is unique. If there is no such $\boldsymbol{x}$, then we say that $O(T, \varepsilon, R)$ is undefined.
- If $R$ is the (infinite) sequence of points that is produced by an algorithm, then $O(T, \varepsilon, R)$ is the output produced by that algorithm when we add to it instructions that make it terminate as soon as it reaches a point that is $\varepsilon$-compatible with $T$.

## Strong Perturbation Resilience

- An algorithm **P** for a problem structure $\langle \mathbb{T}, \mathscr{P}r \rangle$ is strongly perturbation resilient if, for all $T \in \mathbb{T}$,
  - there exists an $\varepsilon \in \mathbb{R}_+$ such that $O\left(T, \varepsilon, \left((\mathbf{P}_T)^k \boldsymbol{x}\right)_{k=0}^{\infty}\right)$ is defined for every $\boldsymbol{x} \in \Omega$;
  - for all $\varepsilon \in \mathbb{R}_+$ such that $O\left(T, \varepsilon, \left((\mathbf{P}_T)^k \boldsymbol{x}\right)_{k=0}^{\infty}\right)$ is defined for every $\boldsymbol{x} \in \Omega$, we have that $O(T, \varepsilon', R)$ is defined for every $\varepsilon' > \varepsilon$ and for every sequence $R = \left(\boldsymbol{x}^k\right)_{k=0}^{\infty}$ in $\Omega$ such that

$$\boldsymbol{x}^{k+1} = \mathbf{P}_T\left(\boldsymbol{x}^k + \beta_k \boldsymbol{v}^k\right), \text{ for all } k \geq 0, \tag{9}$$

    where the sequence $(\beta_k)_{k=0}^{\infty}$ of nonnegative real numbers is summable (that is, $\sum_{k=0}^{\infty} \beta_k < \infty$), the sequence $\left(\boldsymbol{v}^k\right)_{k=0}^{\infty}$ of vectors in $\mathbb{R}^J$ is bounded and, for all $k \geq 0$, $\boldsymbol{x}^k + \beta_k \boldsymbol{v}^k \in \Delta$.
- For a strongly perturbation resilient algorithm, if it is the case that for all initial points from $\Omega$ the infinite sequence produced by the algorithm contains an $\varepsilon$-compatible point, then it will also be the case that all perturbed sequences satisfying (9) contain an $\varepsilon'$-compatible point, for any $\varepsilon' > \varepsilon$.

## Conditions for Strong Perturbation Resilience

- Given an algorithm $\mathbf{P}$ for a problem structure $\langle \mathbb{T}, \mathscr{P}r \rangle$ and a $T \in \mathbb{T}$, we say that $\mathbf{P}$ is convergent for $T$ if, for every $\boldsymbol{x} \in \Omega$, there exists a $\boldsymbol{y}(\boldsymbol{x}) \in \Omega$ such that, $lim_{k \to \infty} (\mathbf{P}_T)^k \boldsymbol{x} = \boldsymbol{y}(\boldsymbol{x})$. If, in addition, there exists a $\gamma \in \mathbb{R}_+$ such that $\mathscr{P}r_T(\boldsymbol{y}(\boldsymbol{x})) \leq \gamma$, for every $\boldsymbol{x} \in \Omega$, then $\mathbf{P}$ is boundedly convergent for $T$.
- **Theorem 1**: *If $\mathbf{P}$ is an algorithm for a problem structure $\langle \mathbb{T}, \mathscr{P}r \rangle$ such that, for all $T \in \mathbb{T}$, $\mathbf{P}$ is boundedly convergent for $T$, $\mathscr{P}r : \Omega \to \mathbb{R}$ is uniformly continuous and $\mathbf{P}_T : \Delta \to \Omega$ is nonexpansive, then $\mathbf{P}$ is strongly perturbation resilient.*
- For example, it is clear that for the CT problem $T = (\boldsymbol{A}, \boldsymbol{b})$ the $\mathscr{P}r_T$ of (5) is uniformly continuous and the $\mathbf{P}_T$ in (8) is nonexpansive. By known properties of ART, bounded convergence of $\mathbf{P}$ is easy to prove for the problem set of consistent CT problems that result in the following set being nonempty:

$$C = \left\{ \boldsymbol{x} \in \mathbb{R}^J \mid 0 \leq \boldsymbol{x}_j \leq 1, \text{ for } 1 \leq j \leq J, \text{ and } \boldsymbol{Ax} = \boldsymbol{b} \right\}. \qquad (10)$$

- It follows from Theorem 1 that the algorithm $\mathbf{P}$ of (8) for the problem structure $\langle \mathbb{T}, \mathscr{P}r \rangle$, with $\mathbb{T}$ containing consistent CT problems and $\mathscr{P}r_T$ defined by (5), is strongly perturbation resilient.

## Nonascending Vectors

- Let $\phi : \Delta \to \mathbb{R}$. For an $\boldsymbol{x} \in \Delta$, a vector $\boldsymbol{d} \in \mathbb{R}^J$ is said to be *nonascending* for $\phi$ at $\boldsymbol{x}$ if $\|\boldsymbol{d}\| \leq 1$ and there is a $\delta > 0$ such that,

$$\text{for } \lambda \in [0,\delta]\,,\ (\boldsymbol{x}+\lambda\boldsymbol{d}) \in \Delta \text{ and } \phi\left(\boldsymbol{x}+\lambda\boldsymbol{d}\right) \leq \phi\left(\boldsymbol{x}\right). \tag{11}$$

- The zero vector is nonascending for any $\phi$ at any $\boldsymbol{x}$. This is useful for guaranteeing the behavior of proposed procedures but, in order to steer an algorithm toward a point at which the value of $\phi$ is small, we need a $\boldsymbol{d}$ such that $\phi(\boldsymbol{x}+\lambda\boldsymbol{d}) < \phi(\boldsymbol{x})$ rather than just $\phi(\boldsymbol{x}+\lambda\boldsymbol{d}) \leq \phi(\boldsymbol{x})$.
- **Theorem 2**: *Let $\phi : \mathbb{R}^J \to \mathbb{R}$ be a convex function and let $\boldsymbol{x} \in \mathbb{R}^J$. Let $\boldsymbol{g} \in \mathbb{R}^J$ satisfy the property: For $1 \leq j \leq J$, if the $j$th component $g_j$ of $\boldsymbol{g}$ is not zero, then the partial derivative $\frac{\partial\phi}{\partial x_j}(\boldsymbol{x})$ of $\phi$ at $\boldsymbol{x}$ exists and its value is $g_j$. Define $\boldsymbol{d}$ to be the zero vector if $\|\boldsymbol{g}\| = 0$ and to be $-\boldsymbol{g}/\|\boldsymbol{g}\|$ otherwise. Then $\boldsymbol{d}$ is a nonascending vector for $\phi$ at $\boldsymbol{x}$.*

## Superiorized Version of an Algorithm **P**

```
**set** $k = 0$
**set** $\boldsymbol{x}^k = \bar{\boldsymbol{x}}$
**set** $\ell = -1$
**repeat**
    **set** $n = 0$
    **set** $\boldsymbol{x}^{k,n} = \boldsymbol{x}^k$
    **while** $n < N$
        **set** $\boldsymbol{v}^{k,n}$ to be a nonascending vector for $\phi$ at $\boldsymbol{x}^{k,n}$
        **set** *loop=true*
        **while** *loop*
            **set** $\ell = \ell + 1$
            **set** $\beta_{k,n} = \gamma_\ell$        $\{(\gamma_\ell)_{\ell=0}^{\infty}$ is a summable sequence of positive real numbers$\}$
            **set** $\boldsymbol{z} = \boldsymbol{x}^{k,n} + \beta_{k,n}\boldsymbol{v}^{k,n}$
            **if** $\boldsymbol{z} \in \Delta$ **and** $\phi(\boldsymbol{z}) \leq \phi\left(\boldsymbol{x}^k\right)$ **then**
                **set** $n = n + 1$
                **set** $\boldsymbol{x}^{k,n} = \boldsymbol{z}$
                **set** *loop = false*
    **set** $\boldsymbol{x}^{k+1} = \mathbf{P}_T \boldsymbol{x}^{k,N}$
    **set** $k = k + 1$
```

## Significance of the Superiorization Approach

- The above description produces automatically the superiorized version of any algorithm **P*.*** Due to repeated steering of the process by lines 7-17 toward a reduced value of $\phi$, we expect the output of the superiorized version to be superior to the output of the original algorithm. The superiorized version of a strongly perturbation resilient algorithm produces outputs that are essentially as constraints-compatible as those produced by the original version. Thus, superiorization can be significantly helpful in research, because it can save a lot of time of a researcher when the application of interest gives rise to a new constrained optimization problem.
- Another significant aspect of superiorization is the following. Constrained optimization problems that arise in applications are often huge. It can then happen that the traditional algorithms for constrained optimization require computational resources that are not easily available and, even if they are, the length of time needed to produce an acceptable output is too long to be practicable. We next illustrate that the computational requirements of a superiorized algorithm can be significantly less than that of a traditional algorithm, by reporting on a comparison of superiorization with the projected subgradient method (PSM), which is a standard method of classical optimization. We carry out this comparison for a consistent CT problem $T = (\boldsymbol{A}, \boldsymbol{b})$ for which the $C$ of (10) not empty.

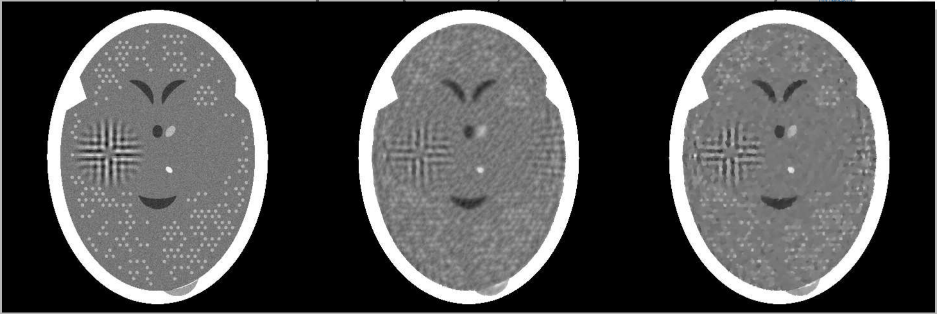
## PSM vs. Superiorized ART

- We (Y. Censor, R. Davidi, G.T. Herman, R.W. Schulte and L. Tetruashvili) used a version of PSM which is guaranteed to converge to a minimal value of $\phi$ over the set $C$, provided that $\phi$ is convex and Lipschitz continuous. These conditions are satisfied when $\phi$ is defined based on TV by obtaining a $G \times H$ array $\boldsymbol{X}$ from the vector $\boldsymbol{x}$ by $\boldsymbol{X}_{g,h} = \boldsymbol{x}_{(g-1)H+h}$, for $1 \le g \le G$ and $1 \le h \le H$, and setting

$$\phi(\boldsymbol{x}) = \mathrm{TV}(\boldsymbol{X}) = \sum_{g=1}^{G-1} \sum_{h=1}^{H-1} \sqrt{\left(\boldsymbol{X}_{g+1,h} - \boldsymbol{X}_{g,h}\right)^2 + \left(\boldsymbol{X}_{g,h+1} - \boldsymbol{X}_{g,h}\right)^2}. \qquad (12)$$

- The computational work reported here was done on a single machine, an Intel i5-3570K 3.4Ghz with 16GB RAM using SNARK09.
- The phantom is a $485 \times 485$ digitized image (thus $J = 235,225$) with pixel size $0.376 \times 0.376$ mm$^2$ with values in the range of [0, 0.6241]. It represents a cross-section of a human head.
- Data were collected by calculating line integrals through this phantom using 60 sets of equally rotated parallel lines, with lines in each set spaced at 0.752 mm from each other (resulting in $I = 18,524$).
- We first applied PSM and then superiorization, obtaining the following.

## Phantom and Outputs (PSM, Superiorization)



- PSM converges to an element of $C$ in the limit but, with a recommended stopping rule, it stopped at iteration 815 with $\mathscr{P}r_T\left(\boldsymbol{x}^{815}\right) = 0.0422$, having used 2,217 seconds of computer time. The output is in the **middle** above. Its TV value is 919, which is less than that of the phantom (on the **left** above) whose TV is 984.
- We used the superiorized version of ART (8) to generate a sequence $\left\{\boldsymbol{x}^k\right\}_{k=0}^{\infty}$ until it reached $O\left(T, 0.0422, \left\{\boldsymbol{x}^k\right\}_{k=0}^{\infty}\right)$. This output is on the **right** above. The computer time required was 102 seconds, which is over twenty times faster than what was needed by PSM. The TV of the superiorization output is 876, which is also less than that of the output of PSM.

## Positron Emission Tomography (PET)

- Problems in $\mathbb{T}$ are of the form $T = (\boldsymbol{A}, \boldsymbol{b})$, with the restrictions that the entries $\boldsymbol{A}_{i,j}$ of the matrix $\boldsymbol{A}$ are real numbers between 0 and 1 and both the row-sums and the column sums of $\boldsymbol{A}$ are strictly positive and that the components $\boldsymbol{b}_i$ of the vector $\boldsymbol{b}$ are nonnegative integers (the counts in the PET scanner data).
- In a proposed solution $\boldsymbol{x}$, the component $\boldsymbol{x}_j$ is the activity in the $j$th pixel and, so, for this problem set the appropriate choice is $\Omega = \left\{ \boldsymbol{x} \in \mathbb{R}^J \mid 0 \le \boldsymbol{x}_j, \text{ for } 1 \le j \le J \right\}$.
- Assuming that for the true (but unknown) $\boldsymbol{x}$, it is the case that, for all $1 \le i \le I$, $\boldsymbol{b}_i$ is an independent sample from a Poisson distribution whose mean is $\sum_{j=1}^{J} \boldsymbol{A}_{i,j}\boldsymbol{x}_j$, it is known that the $\boldsymbol{x}$ whose likelihood is maximal for the observed counts $\boldsymbol{b}$ is the same $\boldsymbol{x}$ that minimizes the Kullback-Leibler (KL) distance

$$\mathscr{P}r_T(\boldsymbol{x}) = \sum_{i=1}^{I} \left( \boldsymbol{b}_i \ln \left( \boldsymbol{b}_i / \sum_{j=1}^{J} \boldsymbol{A}_{i,j}\boldsymbol{x}_j \right) + \sum_{j=1}^{J} \boldsymbol{A}_{i,j}\boldsymbol{x}_j - \boldsymbol{b}_i \right). \tag{13}$$

## The Expectation Maximization (EM) Algorithm

- Using the notation that $(\mathbf{P}_T(\boldsymbol{x}))_j$ is the $j$th component of $\mathbf{P}_T(\boldsymbol{x})$, the EM algorithm is the iterative method specified by

$$(\mathbf{P}_T(\boldsymbol{x}))_j = \left(\boldsymbol{x}_j / \sum_{i=1}^{I} \boldsymbol{A}_{i,j}\right) \sum_{i=1}^{I} \left(\boldsymbol{A}_{i,j}\boldsymbol{b}_i / \sum_{n=1}^{J} \boldsymbol{A}_{i,n}\boldsymbol{x}_n\right), \text{ for } 1 \le j \le J. \tag{14}$$

- The reason why we desire to superiorize the EM algorithm of (14) is that it was observed that images deteriorate with a large number of iterations, in the sense that they present high local variations; it is expected that we can get rid of this problem by superiorizing with an appropriately chosen secondary criterion $\phi$.
- An example of such a $\phi$ is (with $M$ the set of indices $m$ of pixels not on the border and $M_m$, for $m \in M$, the set of indices of their eight neighbors)

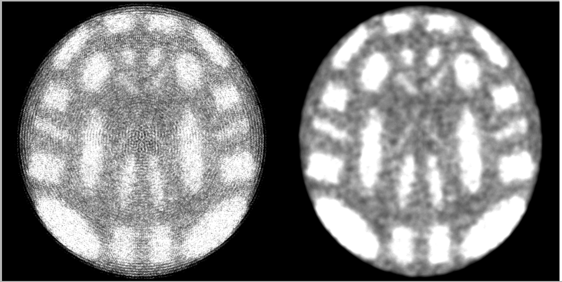
$$\phi(\boldsymbol{x}) = \sum_{m \in M} \left(\boldsymbol{x}_m - \frac{1}{8} \sum_{j \in M_m} \boldsymbol{x}_j\right)^2. \tag{15}$$

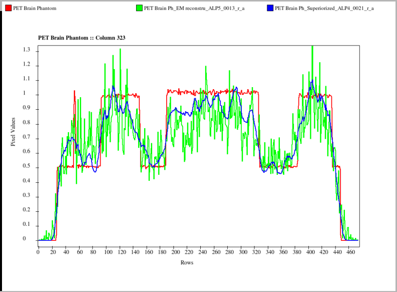
## Superiorization of the EM Algorithm

- Currently we do not know whether or not the EM algorithm of (14) is strongly perturbation resilient. A useful aspect of the superiorization methodology is that its pseudocode can be applied to any algorithm **P**, it does not require that **P** has any mathematical properties beyond those implied by the definition of an algorithm for a problem structure. The production of a superiorized version is quite automatic and, if we have computer code for the $\mathbf{P}_T$ in line 18 of the pseudocode, then the production of computer code for the superiorized version is trivial. So it requires little effort to produce the computer programs needed for experimental evaluation of a superiorized version of an algorithm.
- E. Garduño and I demonstrated this idea on the superiorized version of the EM algorithm of (14) with the secondary criterion $\phi$ of (15). We used SNARK09 to simulate the positron activity in a cross-section of the human brain and the noisy data collection by a PET scanner with a ring of 300 detectors. For that data, the KL distance defined by (13) of the phantom is 14,481.41. We ran both the EM algorithm and its superiorized version until the first time when the value of KL dropped below 14481.41, at that time the value of $\phi$ of (15) was 1,845.81 for EM and 12.94 for its superiorized version. Thus superiorization was clearly demonstrated to work.

# Illustration of Superiorization of EM

| | |
|---|---|
| Left: | EM without superiorization. |
| Middle: | EM with superiorization. |
| Right: | Plots of values along the 323rd column. |





## Summary

- Practical inversion of the Radon Transform often uses constrained optimization, with the constraints arising from the desire to produce a solution that is constraints-compatible. It is typically the case that a large number of solutions would be considered good enough from the point of view of being constraints-compatible. In such a case, an secondary criterion is introduced that helps us to distinguish the “better” constraints-compatible solutions.
- The superiorization methodology is a recently-developed heuristic approach to constrained optimization. The underlying idea is that in many applications there exist computationally-efficient iterative algorithms that produce constraints-compatible solutions. Often the algorithm is perturbation resilient in the sense that, even if certain kinds of changes are made at the end of each iterative step, the algorithm still produces a constraints-compatible solution. This property is exploited by using such perturbations to steer the algorithm to a solution that is not only constraints-compatible, but is also desirable according to a specified secondary criterion. The approach is very general, it is applicable to many iterative procedures and secondary criteria.
- Most importantly, superiorization is a totally automatic procedure that turns an iterative algorithm into its superiorized version.