Binary Tomography
SSIP 2008

**Vladimir Curic, Attila Kozma,
Tibor Lukic, Erik Wernersson**

**Technical University of Vienna
Vienna, 2008.**

1 Preface

Tomography is a technique to explore inside of objects without touching it at all. Normally electromagnetic radiation is emitted by an application, goes through the examined object and are measured on the opposite side how much radiation is absorbed. The result of the checking is a set of vectors, so called projection vectors, that contain the sum of the absorbed rays from different directions. The main goal is to reconstruate the original image from the projection vectors. We deal only with 2D binary images. This method is very useful in medicine, archaeology, geophysics, astrophysics and other sciences.

In the following we define the problem and give some possible solutions to it with examples.

2 Problem specification

A quadratical binary P image is given (e.g. a slice of a human). Afterwards, we make the projection vectors from 2, 4 or 6 directions. . Let D be a matrix where $d_{i,j}$ equals the length of a certain ray going through the $p_{i,j}$ pixel and let $\underline{c}$ be the projection vector. An example of how projections work can be seen in Figure 1.

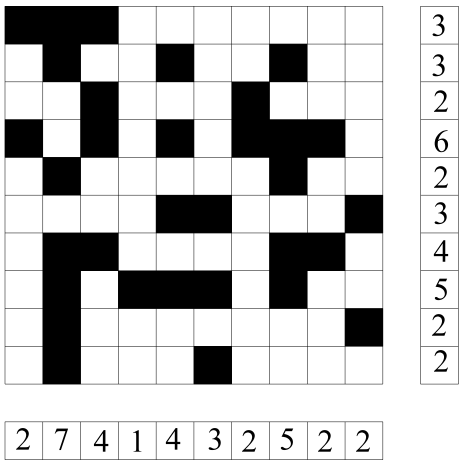

Figure 1: Calculation of projection vectors with horizontal and vertical projections.

For every emittor of every direction we can calculate the $\widehat{D}$ vector by concatenating the row vectors of D matrix. We concatenate the $\underline{c_k}$ projection vectors to a $\underline{b}$. The $\underline{x}$ variable is the vectorized form of the original binary image.

$$A = \begin{pmatrix} \widehat{D}_{1,1} \\ \widehat{D}_{1,2} \\ \vdots \\ \widehat{D}_{m,m} \end{pmatrix} \quad \underline{b} = \begin{pmatrix} \underline{c_1} \\ \underline{c_2} \\ \vdots \\ \underline{c_n} \end{pmatrix} \quad \underline{x} = \begin{pmatrix} x_{1,1} \\ x_{1,2} \\ \vdots \\ x_{m,m} \end{pmatrix} \tag{1}$$

Now we can write our system in a compact way:

$$A\underline{x} = \underline{b} \tag{2}$$

Consider that our system is a highly undetermined linear equation. For example if the input image has size 50×50 and we have only vertical and horizontal projections then we will have 100 equations and 2500 unknown variables. Because of the underdetermined system the one possible way is to use optimization techniques to obtain a global minium

3 Methods on Binary Tomography

In this section we discuss different optimization techniques to solve the previous binary programming problem. We give examples of how they work, what advantages and disadvantages they have.

3.1 Reconstruction with Simulated Annealing

Simulated annealing [6] is an algorithm, inspired by nature (metallurgy), that can be used to solve optimization problems of all kinds, i.e. not only convex or linear as most other methods. It can be proven to converge to the global optimum, which of course is a nice property. Sadly, the convergence can not be guaranteed in finite time so in practice you have to know something about your problem to decide weather simulated annealing is a good method or not.

There are a few things that can be varied in an implementation and the most important design choices will be discussed above.

Objective functions

What is going to be optimized is the goal or objective. The objective function in simulated annealing is the one that describes weather the current proposed solution is good or not. In our case, the natural objective must be to minimize the projection error

$$E_P(x) = ||Ax - b||, \tag{3}$$

which was defined above. This is the only criteria if nothing is known or can be assumed about the true solution.

```
  Set Initial Temperature, T=2
  Generate Initial Solution
WHILE T>0 DO
  1. Create A New Possible Solution
  2. Choose The Best Solution According To The Objective Function Or
       Choose The Worst With Probability ~exp(delta E / T)
  3. Lower The Energy According To Scheme
END
```

Figure 2: A pseudocode outline of the simulated annealing algorithm.

Since information is between uniformity and entropy, some homogeneity can usually be expected in the solution.

We measure homogeneity as the number of border elements in the image over the total number of elements. The number of border elements is calculated by subtraction of the eroded image from the original and then summation of the remaining elements.

$$E_H(Img) = \sum\sum(Img - Img \ominus strel) \tag{4}$$

Solution generator

The following methods were tried:

1. Move one element pixel randomly. This preserves the total number of picture elements.
2. Pick a pixel randomly and let it diffuse a few steps.
3. Pick one pixels randomly and flip its value.

It turned out that the last, and most simple alternative, was the best performing. Not in the number of steps but in computing time.

Temperature scheme and use

Since the log-scheme is theoretically the best, it was chosen. I.E.

$$T =\approx \log(2 - iter/NITER) \tag{5}$$

where $iter$ is the current iteration and $NITER$ is the total number of iterations.

To make it easy to plug in different objective functions we propose that the main algorithm should keep track of the maximal energy difference and use it to normalize the values of the objective function. This is implemented.

Files:

- bsm.m: The main skeleton. Here are all easy settings. This is to be run by the user.
- bsm_main.m: The main loop.
- bsm_newTemp.m: Contains the temperature scheme.
- bsm_proposeNewImage.m: Make the next guess.
- bsm_objectiveFunction.m: What it sounds like.

Results

Results can be seen in the Appendix A.

Discussion

The convergence is very slow at the end. This is because the probability that the right pixel is changed decreases as the number of pixels in the correct state increases. A solution to this problem could be to let the algorithm that proposes new solutions be more systematic as the temperature decreases. There are many things that can be optimized, the most relevant is probably to implement the alghoritm in a more efficient environment like c++.

3.2 Reconstruction with Branch and Bound

Branch and Bound algorithm is a widely used optimazation framework. It consists of two critical functions, called branching and boundary functions. Now we define these in order to solve the optimization problem in an efficient way.

The optimization problem can be interpreted as the following:

$$\begin{array}{rl} \max & \sum_{i=1}^{n}\sum_{j=1}^{n} x_{i,j} \\ \text{subject to} & A\underline{x} = b \\ & x_{i,j} \in 0,1 \end{array} \tag{6}$$

- **Branching** Let L be the set of feasible solutions of the original binary program. The branching function should divide L into two disjoint subsets. With this the original problem can be solved indirectly by solving two subproblems with smaller size. In this paticular case, we select a variable of vector X, let this be x_j and branch the root with L_0 and L_1 nodes. Here L_i contains vectors where $x_j = i$ stands. Afterwards branching is called recursively on the sub nodes.

- **Bounding** Of course we do not want to build the whole tree. For a 50 $\times$ 50 image there are 2^{2500} possible solutions, so we must use cuts in the Branch and Bound tree. Therefore in every selected node we solve the relaxtion of the actual problem. This means that variables are not binary but real. If L_i is the set of the feasible solutions of the actual problem and L_i^* is the set of the feasible solutions of the relaxation, then $L_i \subset L_i^*$ the following stands:
$$Z(\underline{x}^*) \geq Z(\underline{y}^*), \tag{7}$$
where $\underline{x}$ and $\underline{y}$ are the optimal solutions to the appropiate linear programs. In case a node has optimal solution $\underline{x}^*$ and with optimal value $Z(\underline{x}^*)$ it means that objective function value less than $Z(\underline{x}^*)$ is not possible in the whole subtree, thus if we have a binary solution from previous iterations better than x^*, the node shouldn't be extracted.

In Figure 2 you can see how the tree is built. The lower we go in the tree, the less dimensional LPs must be solved. If all the possible nodes are extracted we can read down the optimal solution by searching the leaf with the lowest objective function value.

Results

We solved the problem in Matlab with two projections. The following images were tested with vertical and horizontal projections without any noise. On the first image the cross has a vertical and a horizontal extension, that is why our approach is successful, the original image is almost perfect. On the second image the rotated cross has to be reconstructed, but with only two projections it is only similar to the original one. The third image is almost excellent except for some pixels. On the last image you can notice that for complex objects two projections is not enough and reconstruction is unsuccessful.

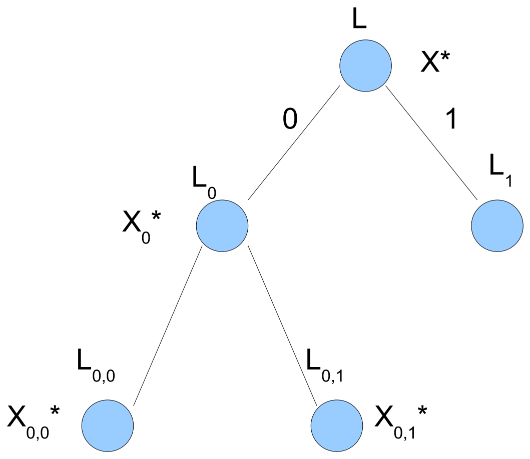

Figure 3: The branch and bound tree after 2 node extraction.

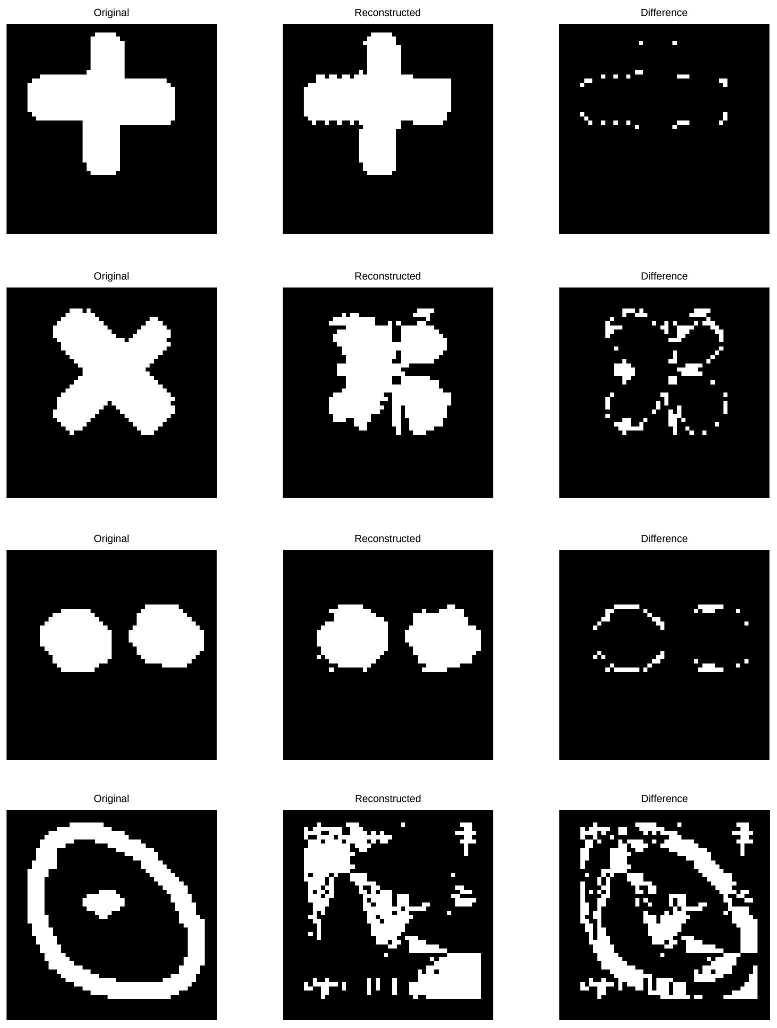

Figure 4: Results of branch and bound routine on different objects without any noise

3.3 SPG-based reconstruction

The *Spectral Projected Gradient* (SPG) algorithm is a deterministic, iterative optimization method, introduced by Birgin, Martínez and Raydan (2000) in [4] for solving the convex-constrained optimization problem

$$\min_{x\in\Omega} f(x)$$

where the feasible region Ω is a closed convex set in R^n. The requirements for application of SPG algorithm are: $i)$ f is defined and has continuous partial derivatives on an open set that contains Ω; $ii)$ the projection, P_Ω of an arbitrary point $x \in R^n$ onto a set Ω is defined. The algorithm uses the following parameters: integer $m \geq 1$; $0 < \alpha_{min} < \alpha_{max}$, $\gamma \in (0,1)$, $0 < \sigma_1 < \sigma_2 < 1$ and initially $\alpha_0 \in [\alpha_{min}, \alpha_{max}]$ (see [5] for details). Starting from an arbitrary configuration $x^0 \in \Omega$, the below computation is iterated until convergence.

SPG iterative step [5].

```
Given $x^k$ and $\alpha_k$, the values $x^{k+1}$ and $\alpha_{k+1}$ are computed as follows:
$d^k = P_\Omega(x^k - \alpha_k \nabla f(x^k)) - x^k$;
$f_{max} = \max\{f(x^{k-j}) \mid 0 \leq j \leq \min\{k, m-1\}\}$;
$x^{k+1} = x^k + d^k$;  $\delta = \langle \nabla f(x^k), d^k \rangle$;  $\lambda_k = 1$;
while $f(x^{k+1}) > (f_{max} + \gamma \lambda_k \delta)$
    $\lambda_{temp} = -\frac{1}{2}\lambda_k^2 / (f(x^{k+1}) - f(x^k) - \lambda_k \delta)$;
    if $(\lambda_{temp} \geq \sigma_1 \wedge \lambda_{temp} \leq \sigma_2 \lambda)$ then $\lambda_k = \lambda_{temp}$ else $\lambda_k = \lambda_k/2$;
    $x^{k+1} = x^k + \lambda_k d^k$;
end while;
$s^k = x^{k+1} - x^k$;  $y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$;  $\beta_k = \langle s^k, y^k \rangle$;
if $\beta_k \leq 0$ then $\alpha_{k+1} = \alpha_{max}$ else $\alpha_{k+1} = \min\{\alpha_{max}, \max\{\alpha_{min}, \langle s^k, s^k \rangle \beta_k\}\}$
```

The SPG algorithm is particularly suited for the situations when the projection calculation is inexpensive, as in box-constrained problems, and its performance is shown to be very good in large-scale problems (see [5]). This motivates us to apply the SPG algorithm in defuzzification, which is a convex, box-constrained large-scale optimization problem.

3.3.1 SPG Based Algorithm for Binary Tomography

We consider the binary tomography problem (2) as a following binary optimization problem

$$\min_{x\in\{0,1\}^n} \frac{1}{2}\left(\|Ax - b\|^2 + \alpha \sum_i \sum_{j\in N(i)} (x_i - x_j)^2 \right) \tag{8}$$

where A is the projection matrix and b is the projection vector. A rule of the last term is to enforce the coherency of the solutions. The objective function in (8) can be reformulate in following way [3]

$$\Phi(x) = x^T Q\, x + q^T x,$$

where

$Q = A^T A + \alpha L,$

$x^T L\, x = \sum_i \sum_{j \in N(i)} (x_i - x_j)^2,$

$q = -A^T b.$

Since the matrix Q is positive definite, a function Φ is convex. Instead of problem (8) we consider a relaxed convex constrained problem

$$\min_{x \in [0,1]^n} \Phi(x) + \mu \cdot x^T (x - e)\,, \quad \mu > 0 \tag{9}$$

where $e = [1, 1, 1, .., 1]^n$, and μ is a binary enforcing term. Our strategy is to solve a sequence of optimization problems (9), with gradually increasing μ, which will lead, to a solution of the binary optimization problem (8). We define the projection P_Ω of an arbitrary vector $x \in R^N$ onto a set $\Omega = [0, 1]^N$ as

$$[P_\Omega(x)]_i = \begin{cases} 0, & x_i \le 0 \\ 1, & x_i \ge 1 \\ x_i, & \text{elsewhere} \end{cases}, \qquad \text{where } i = 1, \ldots, N\,. \tag{10}$$

P_Ω is a projection with respect to the Euclidean distance. Since the the objective function in (9) is differentiable and the projection onto a feasible set is given, a problem (9) can be solved by SPG algorithm. More precisely, we suggest the following optimization algorithm:

SPG based algorithm for binary tomography.

```
Parameters: $\epsilon_{in} > 0$; $\epsilon_{out} > 0$; $\mu_\Delta > 0$
$x^0$; $\mu = 0$; $k = 0$;
do
    do
        $x^{k+1}$ from $x^k$ by SPG iterative step; $k = k + 1$;
    while $\|x^k - x^{k-1}\|_\infty > \epsilon_{in}$
    $\mu = \mu + \mu_\Delta$;
while $\max_i\{\min\{x_i^k, 1 - x_i^k\}\} > \epsilon_{out}$.
```

The initial configuration is the original fuzzy set. In each iteration in the outer loop we solve, by using the SPG method, an optimization problem (9), for a fixed binary factor $\mu > 0$. By iteratively increasing the value of μ in the outer loop, binary solutions are enforced. The termination criterion for the outer loop, ϵ_{out}, regulates the tolerance for the finally accepted (almost) binary solution.

The termination criterion for the inner loop, ϵ_{in}, affects the number of iterations made for the specific binary enforcement setting μ. If ϵ_{in} is chosen too small, it may lead to an unnecessarily large number of inner iterations, and by that slow down the process, without any significant improvement on the final binary solution. Its value, however, should be small enough to ensure reasonably good output of the SPG algorithm.

The starting value of μ is 0, to ensure that the influence of the features observed in defuzzification is sufficiently high in the beginning of the process, compared to the influence of the binary enforcement term. The parameter μ_Δ regulates the speed of enforcement of binary solutions. Too fast binarization (high value of μ_Δ) usually leads to a poor result. On the other hand, too small μ_Δ step may lead to an unnecessary big number of outer iterations.

3.3.2 Discusion

Deterministic optimization algorithms, in contrary to non-deterministic ones, always converge to the same result with the same convergence speed, which makes such methods practically appealing. On the basis of our experiments the algorithm perform well. Its main advantages are: the deterministic nature, good reconstruction results in compare with other two consider methods (SA and B&B) and relatively low sensitivity for noise. However, the convergence in a case of noise data is low, which can be a challenge for further work.

3.3.3 Experiments

We implemented our method in Matlab. The algorithm are tested by using two, three and four projections. Also, in a case of four projections we add Gaussian noise to the projection vector with mean 0 and variance 0.001.

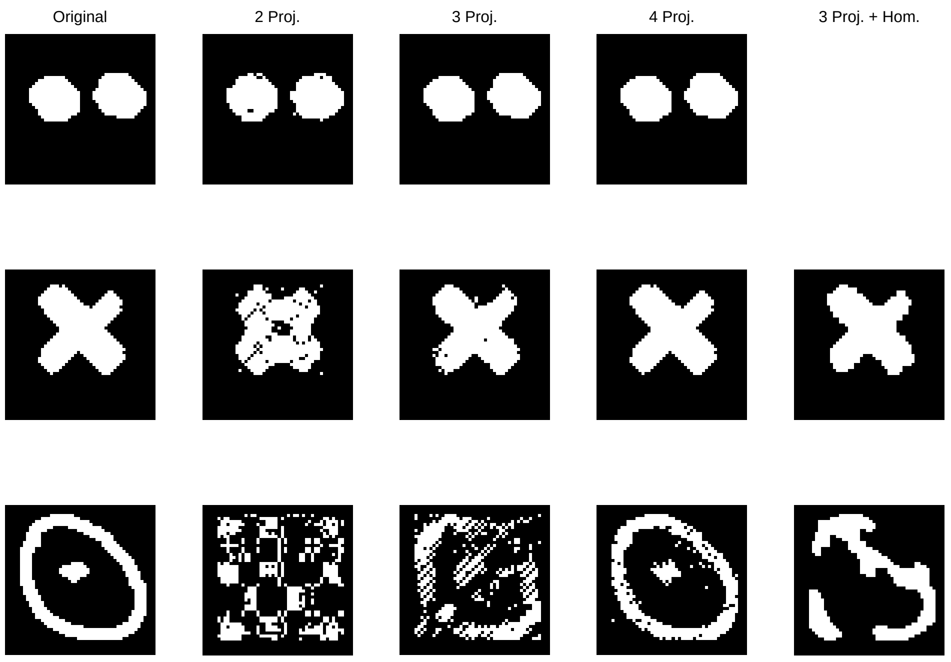

Figure 5: Reconstructions obtained by SPG based algorithm without any noise.

A Results of simulated annealing

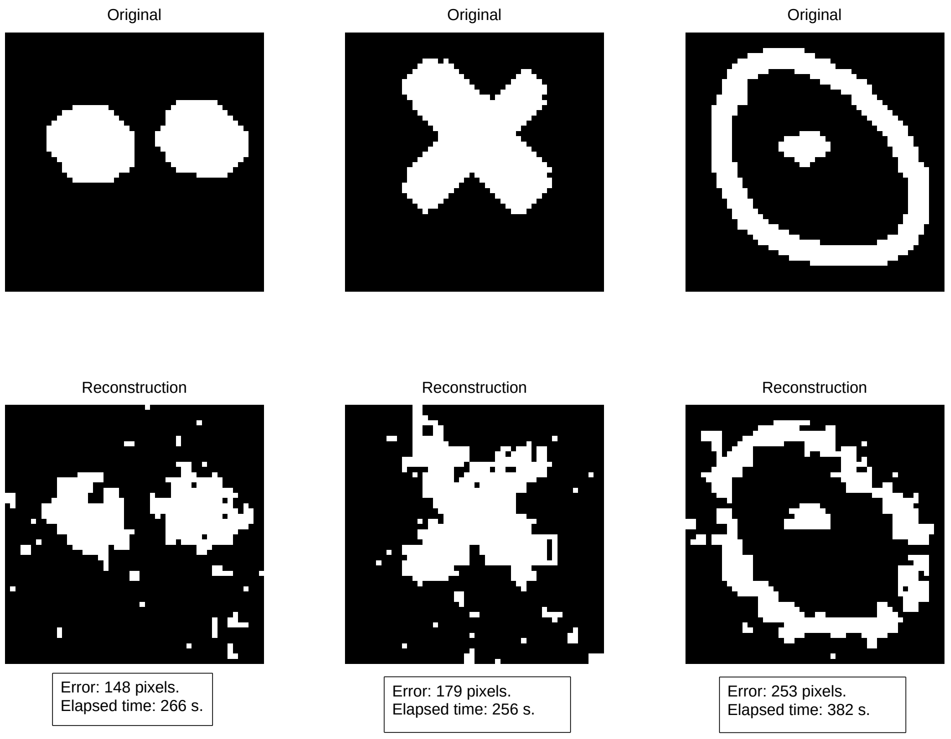

Figure 6: Reconstructions from projections obtained by SPG based algorithm with Gaussian noise. The homogenity prior term is included.

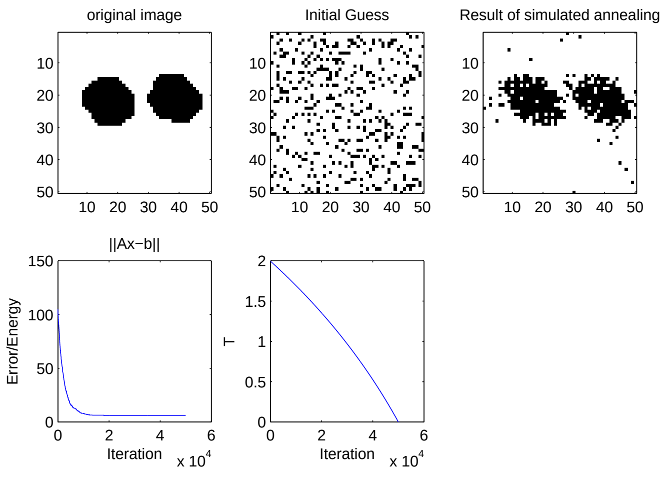

Figure 7: Simulated annealing. 50000 iterations. 2 Projections.

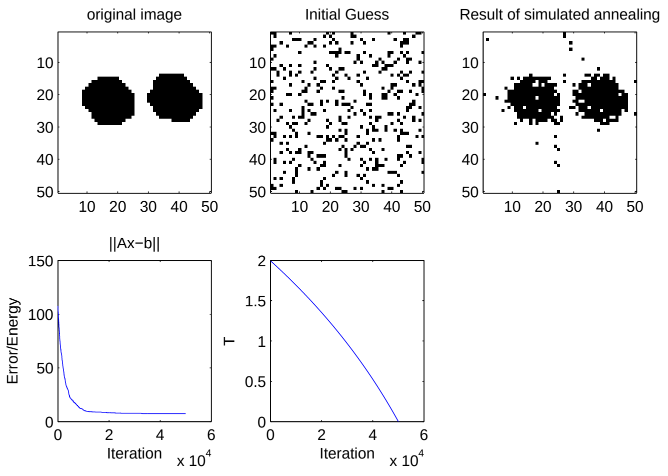

Figure 8: Simulated annealing. 50000 iterations. 3 Projections.

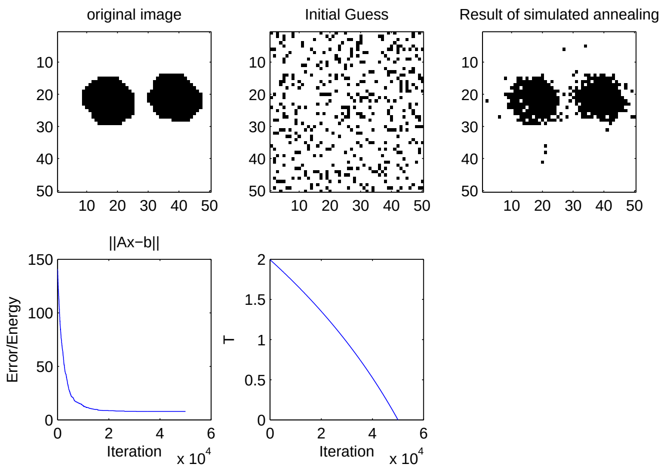

Figure 9: Simulated annealing. 50000 iterations. 4 Projections.

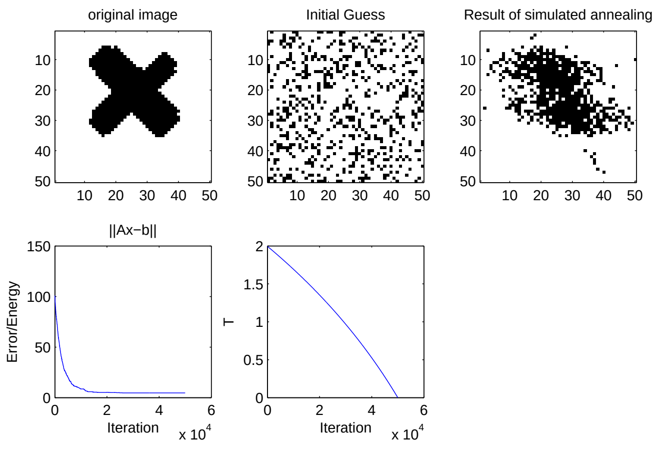

Figure 10: Simulated annealing. 50000 iterations. 2 Projections. seconds.

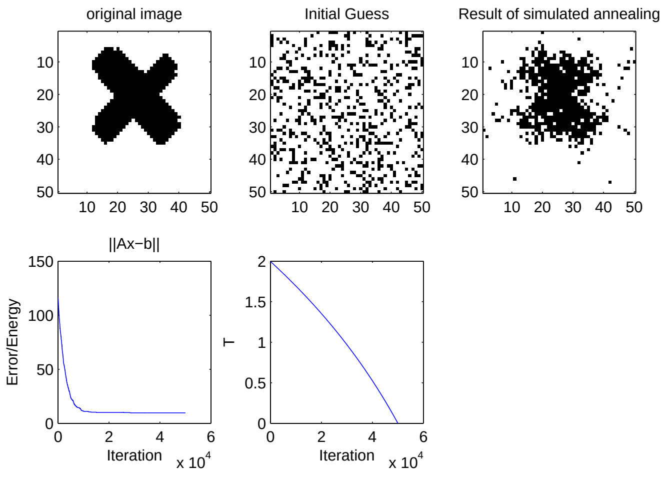

Figure 11: Simulated annealing. 50000 iterations. 3 Projections. seconds.

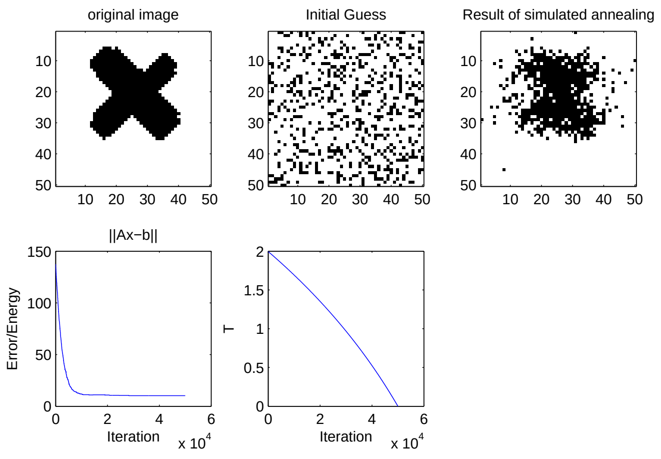

Figure 12: Simulated annealing. 50000 iterations. 4 Projections. seconds.

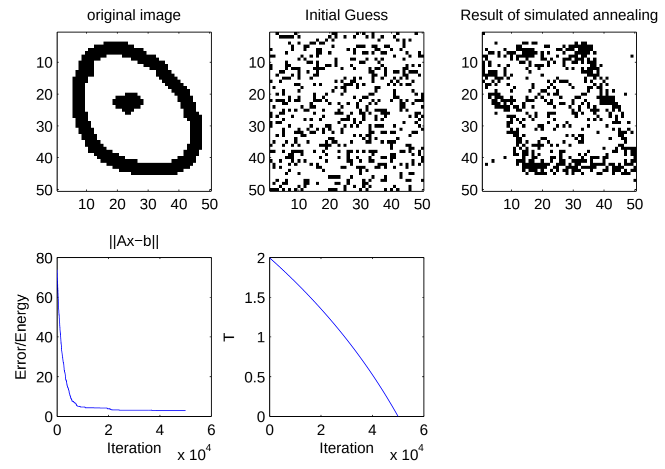

Figure 13: Simulated annealing. 50000 iterations. 2 Projections. seconds.

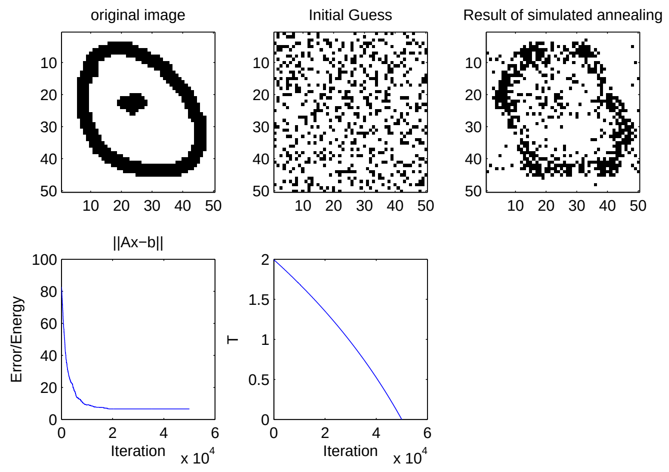

Figure 14: Simulated annealing. 50000 iterations. 3 Projections. seconds.

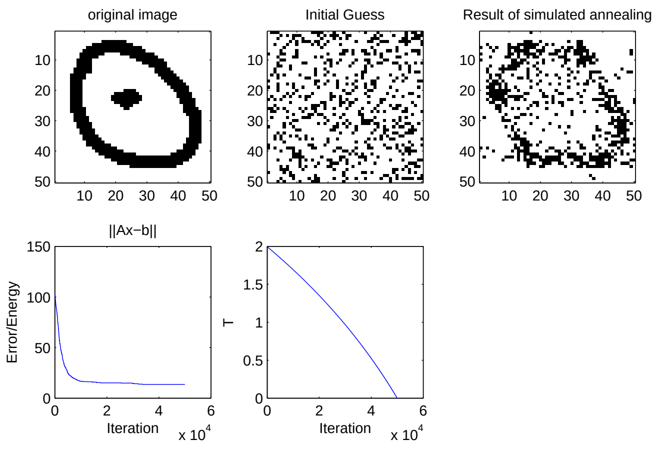

Figure 15: Simulated annealing. 50000 iterations. 4 Projections. seconds.

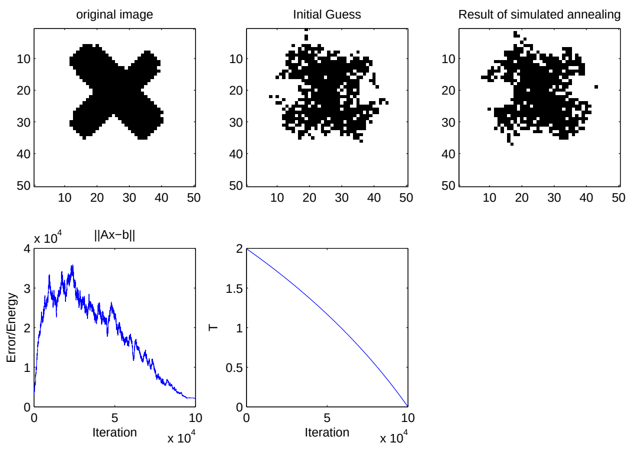

Figure 16: Simulated annealing. Three projections. 200000 iterations. Reannealing after 100000 iterations. The noise in each projection was $\in N(0, 0.01)$

References

[1] E. L. Lawler and D. E. Wood: Operations Research, Vol. 14, No. 4 (Jul. - Aug., 1966), pp. 699-719

[2] Gabor T. Herman, Attila Kuba: Discrete Tomography: Foundations, Algorithms, and Applications 1999. Springer

[3] T. Schüle, C. Schnörr, S. Weber, J. Hornegger : Discrete tomography by convex-cocncave regularization and D.C. programming. SIAM J. on Optimization 10, (2000), 1196–1211.

[4] Birgin, E.G., Martínez, J.M., Raydan, M.: Nonmonotone spectral projected gradient methods on convex sets. SIAM J. on Optimization 10, (2000), 1196–1211.

[5] Birgin, E.G., Martínez, J.M., Raydan, M.: Algorithm: 813: SPG - Software for convex-constrained optimization. ACM Transactions on Mathematical Software 27, (2001), 340–349.

[6] S. Kirkpatrick and C. D. Gelatt and M. P. Vecchi, Optimization by Simulated Annealing, Science, Vol 220, Number 4598, pages 671-680, 1983.