Least-cost Deconvolution

Warren D. Smith, July 2025 [`warren.wds@gmail.com`].

Abstract. I formulate the "least cost deconvolution" problem for images, and within my formulation show that with any particular cost-function choices, it (1) has a *unique* solution, and (2) that solution may be found (for N-pixel images, each pixel value in $[0,2^V)$, to D-decimal accuracy) by a *polynomial(N,V,D)-time algorithm*. "Deconvolution" algorithms have been around since the work of W.H.Richardson, L.B.Lucy, and Jan Högbom in the early 1970s, but it annoyed me that uniqueness and computational-efficiency theorems had not been stated in that literature (at least that I noticed); and indeed some of the prior algorithms had indications to the contrary, e.g. Liu et al 2025.

1. The problem and the cost function

The input is an "image" – a finite array of pixel values, each a nonnegative integer. (Any finite dimensionality for the array.) The output is another such image with nonnegative real pixel values. The task is to find the output image with minimum "cost." The cost is the sum of two kinds of subcosts: "matching" costs and "a priori pixel costs":

$$\text{cost} = \sum_j(\text{matching cost}_j) + \sum_j(\text{pixel cost}_j).$$

Each pixel cost_j is a known concave-∪ function $\Psi(q)$ of the j^{th} output pixel value q, which is infinite when either $q<0$ or $q\to+\infty$, and finite for at least some (a priori known) value of q. For example this

$$\Psi(q) = \max\{ [\ln(q/Q)^2+1]^{1/2}, A+Bq/Q \} - 1 \quad \text{if } q/Q>0, \text{ otherwise } \Psi(q) = +\infty$$

(where Q-1 is the corresponding *input* pixel data value, so $Q\ge1$, and $A\approx0.64698363225$ and $B\approx0.28487766611$) has the following properties:

1. It is a continuous concave-∪ function of q for all q with $q/Q>0$.
2. It is infinite when $q/Q\to0+$.
3. It has continuous first and second derivatives for all q with $q/Q>0$.
4. It is positive if $q\neq Q$, but zero if $q=Q$.
5. Its third derivative is discontinuous at $q/Q\approx1.97847788442$.

There also is a known nontrivial "point spread function" PSF_{jk} where j and k are integer pixel-indices.

$$\text{Matching cost}_j \equiv \Phi_j(Q_j - \sum_k \text{PSF}_{jk}\, q_k)$$

where the $\Phi_j(x)$ are known *strictly*-concave-∪ functions of real x with $\Phi(0)=0$ and $\Phi(x)>0$ if $x\neq0$ and $\Phi(x)\to\infty$ when $|x|\to\infty$. For example, $\Phi_j(x)=x^2/Q_j$ would meet those criteria.

2. Example applications.

1. View the sky through a telescope. Because of unhappy effects such as diffraction and atmospheric turbulence, point stars will appear not to be points, but rather extended objects, in the resulting image. The PSFs can be worked out by observing a collection of standard stars at all orientations of your telescope. (These PSFs actually will depend not only on image-pixel locations, but also light color and polarization.) Using deconvolution one can attempt to remove those bad effects. This should be most effective with radio and infra-red (as opposed to optical and UV) telescopes because with longer wavelengths there is more diffraction, and also it is easier to make your image-sensor's pixel size be well below diffraction lengths ("oversampling").

2. Using gamma ray detectors we obtain an image of the gamma-ray sky. But due to scattering, the claimed direction angles of those photons unfortunately will get "blurred." The PSFs can be worked out by experiments with a point gamma-ray standard source. Using deconvolution one can attempt to remove those blurring effects.

3. If PSF_{jk} is a function solely of the geometric *distance* between pixels j and k, then the $\sum_k PSF_{jk} q_k$ is a "convolution" and all N values of these sums can be computed in O(NlogN) operations using FFT-based "fast convolution" algorithms, as opposed to the naive operation count N^2 for a matrix-vector multiplication. Speedups also are possible if the matrix happens to be "Toeplitz" or "banded."

4. One way to find "error bars" for your output image: artificially pollute your input image with realistic amounts of random noise, compute the resulting output image, and repeat all that 18 times, thus obtaining a set of 19 output images. Then any particular given output pixel will lie within the interval determined by the min and max among its 19 values, with confidence 90%.

Warning: For problems arising from the real world in which there really is a "true" image that your output is trying to approximate, it is rather important that you design your Ψ and Φ cost-functions so that the cost-minimization has some worthwhile relationship to the real world, if you want to hope those "confidences" pertain to the true image rather than just to the algorithm's output image.

3. Facts about concave-∪ functions with range $\mathbb{R} \cup \infty$ and domain $\mathbb{R}^N$.

1. The sum of concave-∪ functions is concave-∪.
2. For any constant A>0, the product of A times a concave-∪ function is concave-∪.
3. The max of concave-∪ functions is concave-∪.
4. A strictly concave-∪ function has at most one local minimum (which therefore must be the global min).
5. A strictly concave-∪ function which is bounded below by some constant, and which grows unboundedly if you walk far enough in either direction along any line, necessarily has a minimum.
6. Also (to extend 5), within any convex compact subdomain of $\mathbb{R}^N$ containing at least one point at which the function is finite-valued, the function necessarily has a unique minimum.
7. If the function in (6) is differentiable and there is a polynomial-time algorithm to evaluate it and its derivative at any given input point to D-decimal accuracy, and a point is known at which

the function is finite, and a ball or hypercube is known (containing that point), then: there is a polynomial(D,N)-time algorithm to evaluate the location of, and function value at, the min within that ball or hypercube. (E.g: "Khachiyan's ellipsoid algorithm" see Grötschel et al 1993.)

Remark: if the function has continuous second derivative and the min is located interior to the ball or hypercube, then "conjugate gradient" minimization algorithms involving "line searches" tend to work better than the ellipsoid algorithm for most purposes.

Conclusion.

Theorem. If the N+N cost-defining functions $\Psi_j(x)$ and $\Phi_j(x)$ are differentiable and polynomial-time evaluable, and the N^2 values of PSF_{jk} are polynomial-time evaluable from the integers j,k, then the least-cost output image with, e.g. each pixel value restricted to the interval $[0, 2^V-1]$ for some specified $V \geq 0$, is polynomial(N,V,D)-time computable.

Proof. Use the ellipsoid algorithm with initial guess being the input image. **Q.E.D.**

References

Martin Grötschel, Laszlo Lovasz, Alexander Schrijver: Geometric Algorithms and Combinatorial Optimization, Springer 1993 (2nd ed). QA167.G76.

Yiming Liu, Spozmai Panezai, Yutong Wang, Sjoerd Stallinga: Noise amplification and ill-convergence of Richardson-Lucy deconvolution, Nature Communications 16,1 (January 2025) #911 open access.

Leon B. Lucy: An iterative technique for the rectification of observed distributions, Astronomical Journal 79,6 (1974) 745-754.

William H. Richardson: Bayesian-Based Iterative Method of Image Restoration, Journal of the Optical Society of America 62,1 (1972) 55-59.

Wikipedia: CLEAN (algorithm).