White Paper: Open Source Deconvolution Components implemented using ImageJ2/Imglib2 framework

Brian Northan*,
* True North Intelligent Algorithms, Guilderland, New York

***Abstract*—There is a need for improved deconvolution algorithms for ImageJ. Current solutions all have drawbacks. DeconvolutionLab implements a variety of algorithms, however it is closed source which makes it difficult to extend. Other solutions, including Iterative Deconvolution (Dougherty) and Parallel Iterative Deconvolution (Wendykier) are based on the ImageJ1 framework. This White Paper describes a strategy to implement deconvolution into the next generation ImageJ2 platform. Preliminary results are presented. The submitted results were generated using the Richardson-Lucy Algorithm with total variation regularization, non-circulant edge handling, and vector extrapolation acceleration. Non-overlapping convolution and correlation operators were ported to Java based on a preexisting Matlab implementation. The Total Variation algorithm was ported to Java based on a preexisting open source Python implementation. All code is open source and has been placed in publically available git repositories.**

I. Introduction and Notations

Deconvolution is a complicated task that requires many steps beyond just algorithm selection. These include:

1) Boundary conditions: In Convolution and Deconvolution calculations are performed on a neighbourhood of pixels. A strategy must be devised to perform calculations for pixels near the boundaries.
2) Regularization
3) Acceleration
4) Stopping Criteria
5) PSF

Images may need to be extended to deal with boundary conditions. The PSF must be generated (or preprocessed if a measured PSF is used). Regularization parameters and stopping criteria need to be selected. One possible approach to the problem is to combine components from different open source code bases. Multiple code bases allow more flexibility. For example, one code base may have a deconvolution algorithm that is very robust to noise, while another code base may have efficient Fast Fourier Transform (FFT). For the deconvolution challenge, the Imglib2 code base [1] was used for FFTs and extension of images. The related Imagej2 code base [2] was used for file IO, scripting, and visualization of results. The IOCBio [4] code base was used as a template for the total variation regularization deconvolution algorithm. And finally Matlab scripts provided by the course organizers [6] were used as a template for non-overlapping forward and adjunct operators.

In this document, the following matrix/vector notations are used:

- $\mathbf{i}$ is the observed image;
- $\mathbf{o}$ is the true representation of the object;
- $\mathbf{h}$ is the point spread function (PSF);
- $\mathbf{o}_k$ is an estimate of $\mathbf{o}$ obtained after k iterations of the algorithm.

The source code used to generate these results can be found on Github at https://github.com/bnorthan/.

II. Algorithm

The algorithm that was used was Richardson Lucy Total Variation originally presented by Dey [3]. An open source version of the algorithm was implemented by Laasmaa [4] in the Python language. This implementation was ported to Java so that it could be integrated into the ImageJ2 framework. Richardson Lucy Total Variation is an iterative procedure that consists of the repeated application of the following update rule:

$$\mathbf{o}_{k+1}(s) = \left\{ \left[\frac{i(s)}{(o_k * h)(s)} \right] * h(-s) \right\} \times \frac{o_k(s)}{1 - \lambda_{TV} div \left(\frac{\nabla o_k(s)}{|\nabla o_k(s)|} \right)} \quad (1)$$

III. Cost Function

The Richardson-Lucy algorithm is a maximum-likelihood algorithm that is based on a Poisson noise model. The Richardson-Lucy with Total Variation extends the Richardson-Lucy algorithm and minimizes a functional composed of an energy term $\mathbf{J_1}$ and a regularization term $\mathbf{J_{reg}}$. The regularization term is based on Total Variation as in [5].

$$J_{reg}(o) = \lambda_{TV} \sum_s |\nabla o(s)| \quad (2)$$

The total functional to be minimized is:

$$J_1(o) + J_{reg}(o) = \sum_s (-i(s) \log[(o * h)(s)] + (o * h)(s)) + \lambda_{TV} \sum_s |\nabla o(s)| \quad (3)$$

The update rule in equation 1 is derived by minimizing equation 3 using the EM procedure.

IV. Choice of the parameters

For Richardson-Lucy with Total Variation, two parameters need to be chosen: the regularization parameter λ and the number of iterations k. To deal with boundary conditions, the image can be extended. The size of the boundaries in each dimension (b_x, b_y, and b_z) and a strategy to assign values to boundary voxels need to be chosen. The boundary conditions were assigned based on the forward model used to generate the test images [6]. For a measured image $\mathbf{i}$ of size $[K_x, K_y, K_z]$ and PSF $\mathbf{h}$ of size $[L_x, L_y, L_z]$ the object space $\mathbf{i}$ is potentially of size $[N_x, N_y, N_z] = [K_x+L_x-1, K_y+L_y-1, K_z+L_z-1]$. The object space is larger then the image space which allows the possibility of reconstruction outside the observation window. In this work the object first guess was a constant value. At each iteration the estimate was normalized by the matrix γ following the procedure in the matlab script $RLdeblur.m$ that was provided with [6].

One strategy to choose the parameters is to run the algorithm using various settings and then choose the result that is most pleasing visually. The weakness in this strategy is that it becomes impractical to test all permutations of multiple parameters and that human perception may not correlate with quantitative criteria. The following parameters were chosen:

$k = 1000, \lambda = 0.0005, b_x = 64, b_y = 64, b_z = 64$

The boundary values of the image and PSF were assigned using a constant value of 0. The size of the boundary was assigned to avoid overlap in the convolution and adjunct calculation [6]. Note that the regularization parameter λ is smaller than that used in [3] ($\lambda = 0.002$). When testing with the value $\lambda = 0.002$, the result appeared artificial (over smoothed). Since the images were generated by simulation the optimal result may be perceived as artificial by a human.

Future work should be done to compare the λ chosen by human judgment vs. the optimal λ determined by quantitative criteria. This could be done when the ground truth images become available.

References

[1] T. Pietzsch, S. Preibisch, P. Tomancak and S. Saalfeld, "ImgLib2 - Generic image processing in Java", Bioinformatics, Vol. 28(22), pp. 3009-3011 (2012).

[2] C.T. Rueden, J. Schindelin, B.E. Dezonia, A.R. Grislis, M.C. Hiner and K.W. Eliceiri (2013). "Open Source BioImage Informatics: Tools for Interoperability. Microscopy and Microanalysis", Vol. 19 (Suppl. 2) , pp 754-755 (2013).

[3] N. Dey, L. Blank-Feraud, C. Zimmer, P. Roux, Z. Kam, J-C, Olivo-Marin, J. Zerubia, "Richardson-Lucy Algorithm with Total Varation Regularization for 3D Confocal Microscope Deconvolution", Microscopy Research and Technique, Vol. 69(1), pp. 260-266 (2006).

[4] M. Laasmaa, M. Vendelin, P. Peterson, "Application of regularized Richardson-Lucy algorithm for deconvolution of confocal microscopy images", Journal of Microscopy, Vol. 243(2), pp. 124-140 (2011).

[5] L. Rudin, S. Osher, E. Fatemi, "Nonlinear total variation noise removal algorithm", Physica D., Vol. 60(2), pp. 259-266 (1992).

[6] C. Vonesch (2013), "Second International Challenge on 3D Deconvolution Microscopy", http://bigwww.epfl.ch/deconvolution/challenge.

[7] D. Biggs, M. Andrews, "Acceleration of iterative image restoration algorithms", Applied Optics, Vol. 36(8), pp. 1766-75 (1997).

[8] S. Remmele, J. Hesser, "Vector Extrapolation-Based Acceleration of Regularized Richardson Lucy Image Deblurring", Informatik Actuell, pp. 400-404 (2009).