A Multiple Algorithms Deconvolution Program

R. Ragazzoni[1,2], A. Baruffolo[1]
[1]Telescopio Nazionale GALILEO
[2]Osservatorio Astronomico di Padova

Abstract

In spite of the great number of writeable algorithms for image restoration, only very few deconvolution methods are popular among astronomers (*e.g.* Richardson–Lucy, MEM, maximum likelyhood, Clean, with slight modifications). An attempt to corrupt such situation is given in this paper where a program able to deal with at least 38 deconvolution algorithms is briefly described. The implemented algorithms follow the unification scheme given by Meinel [5] and include the above mentioned popular algorithms with some of their modifications (*e.g.* accelerated Richardson–Lucy). The program is designed to allow an easy implementation of further algorithms by the user.

1 Introduction

In his classical paper, Meinel [5] (M86 hereafter) pointed out that it is possible, at least in principle, to write a number of deconvolution algorithms, each one derived from different assumptions. For any algorithm a number of *free* parameters are foreseen, leading to an huge *degree of freedom* for the choice of a suitable deconvolution algorithm by the end user. Furthermore, it is to be pointed out that, even in the simpler case, a noticeable amount of code has to be written, checked and optimized, if one of the third–generation programming languages (like FORTRAN) is adopted. In a sense, this is one of the reasons of the great popularity within the astronomical community of packages like IDL [5], where one can deal with a number of pre–defined high–level functions and procedures. In this paper a further step is tentatively made in this direction, describing a DECONVolution compILer (namely `Deconvil`) able to translate a *metacode* description of the deconvolution algorithm into an IDL function. Anyone wanting to experiment with a specific method can give an high–level description of it in a concise manner into a description file (*e.g.* `Method.Def`), then compile it to obtain an executable function (*e.g.* `Method.Pro`) which can be applied to the data. This is accomplished in an automatic manner via the execution of the statement, following the example shown above, `Deconvil Method`.

2 Description of the package

The package consists of a *main* program, which purpose it is to translate from the metacode file into an IDL function, plus a number of auxiliary functions and procedures. The metacode file contains the description of the algorithm plus the default values to be assigned to the parameters eventually used in the algorithm itself. The produced function takes the raw image and the PSF as required inputs, while all other parameters are optional. A special case is represented by the number of iterations made in the deconvolution process. It is defaulted to 10 and can be overridden using the qualifier `NLoop`.

The metacode is structured as follows:

Proceedings of 5th ESO/ST-ECF Data Analysis Workshop, April 26-27, 1993

1. Each line beginning with a '%' character is treated as a comment;
2. Definition of parameters (choosen from the list given below and taking into account the case–sensitive feature of the package) in the form:

$$Param = Value \tag{1}$$

where $Value$ can be an expression containing other parameters from the above mentioned list;

3. The declaration of the algorithm as a single line statement in the form:

$$Ima = Expression \tag{2}$$

The allowed operators are the usual $(+, -, /, *, \hat{})$. Moreover a *convolution operator* is indicated using @, that is:

$$A@B \Rightarrow A \otimes B \tag{3}$$

In the produced code a FFT based version of the convolution algorithm will be used. It is possible to use all the functions provided by IDL both from its internal library and from any user defined one. Functions must be written in lower case (*i.e.* `sqrt` for the square root). Some functions commonly used by the deconvolution algorithms are provided as part of the package.

The allowed parameters are here listed together with their notation in M86.

- **`Raw`**
  Is the raw, original image to be deconvolved. In M86 it is quoted as i;
- **`PSF`**
  Is the assumed Point Spread Function. In order to work properly the peak of this bidimensional function has to be placed at $(0, 0)$. This can be accomplished via `Shift_OO` procedure provided with the package. In M86 it is indicated as s;
- **`Ima`**
  Is the deconvolved image, denoted by o in M86;
- **`First_Guess`**
  Is the first value used for `Ima` or o;
- **`PSF_Target`**
  Is the desired PSF of the deconvolved image. Being o_{true} the true object image, the deconvolution will offer $o = o_{true} \otimes$ `PSF_Target`;
- **`P`**
  Acceleration parameter used in many algorithms (p in M86);
- **`W`**
  The bidimensional weighting function (w in M86);
- **`Gamma`**
  Degree of smoothness (γ in M86);
- **`Q`**
  A priori knowledge of the image, if any, denoted by Q in M86;

- `Model`
  A model for the expected image, quoted as M in M86. In a sense it is a sort of *a priori* knowledge of the image itself. The difference in respect to `Q` being the fact that the latter is weighted by the parameter `G`;

- `G`
  Is the complement to one of f, the confidence in the former parameter **`Model`**. Denoted as g in M86, and $g = 1 - f$ holds;

- `Mask`
  Spatially averaging function, t in M86;

- `Omega`
  Is the estimated standard deviation of **`Ima`**, ω in M86;

- `Sigma`
  Is the estimated variance, σ in M86, of the expected noise on **`Ima`** in the case of Gaussian distribution.

3 Examples

In the following section a few examples of how the package works are shown. For some deconvolution methods we give the relative metacode description together with the proper reference to M86, for the last one the produced IDL function is also attached. It is noticeable that the resulting function is fully editable. As an example it is easy to introduce any check of the convergence of the deconvolution process.

3.1 Tarasko–Richardson–Lucy

The *popular* deconvolution algorithm was developed in 1969 by Tarasko [7], and independently derived by Richardson [6] in 1972, and Lucy [4] in 1974. In M86 its accelerated version is given in Equation (31).

```
%
% Lucy.Def
% ------------------------------------------------
% The 'accelerated' version of the Richardson-Lucy
% deconvolution algorithm. P is the acceleration
% parameter, defaulted to no-acceleration = 1.0
%
P=1.0
Ima=Ima*((Raw/(PSF@Ima))@PSF)^P
%
```

3.2 Burke

A maximum likelyhood deconvolution method, derived by Burke [1] in 1974, from the assumption of Poissonian noise. When photon counting devices are used, or when read–out noise for CCDs is kept very low, this is the most proper description of the noise. Described in M86 by Equation (32).

```
%
% Burke.Def
% ----------------------------------------
% Burke deconvolution algorithm.
%
W=Raw
P=1.0
Ima=Ima*exp(W*(PSF@(Raw/(PSF@Ima))-1.0)^P)
```

3.3 Hunt modified

This is a deconvolution algorithm developed by Hunt [2] in a bayesian framework and successively modified by Meinel in M86, Equation (75).

```
%
% Hunt.Def
% ----------------------------------------
% Modified versione of the Hunt algorithm.
%
Omega = 1.0
Sigma = 1.0
Mask  = Raw
Ima   = (avg(Ima)+(Omega/Sigma)^2.0*PSF@(Raw-PSF@Ima)+Mask@(Ima-avg(Ima)))
```

In this last example we hereafter show the IDL code generated by the package.

```
Function Hunt,Raw,PSF,NLoop=NLoop,W=W,First_Guess=First_Guess, $_
              Gamma=Gamma,PSF_Target=PSF_Target,Q=Q,Mask=Mask, $_
              Model=Model,G=G,Omega=Omega,Sigma=Sigma
If Not(KeyWord_Set(NLoop))       Then NLoop=10
If Not(KeyWord_Set(Omega))       Then Omega=1.0
If Not(KeyWord_Set(Sigma))       Then Sigma=1.0
If Not(KeyWord_Set(Mask))        Then Mask=Raw
If Not(KeyWord_Set(First_Guess)) Then First_Guess=Raw
Ima=First_Guess
For I=0,NLoop Do Begin
  Ima=(avg(Ima)+(Omega/Sigma)^2.0*FT_Convol(PSF,(Raw-FT_Convol(PSF,Ima)))+ $_
      FT_Convol(Mask,(Ima-avg(Ima))))
EndFor
Return,Ima
End
```

4 Conclusions

The `Deconvil` package has been briefly described together with some significant examples. Indications for further modification of the program were also discussed. In Figure 1 some examples obtained using this software are briefly shown.

The software described here obviously does not represent an ultimate tool for deconvolution. Nevertheless, we think that the guidelines sketched here are to be taken at least as a reminder if one wants to bridge the gap between the mathematically developed algorithms and those available in computer usable form.

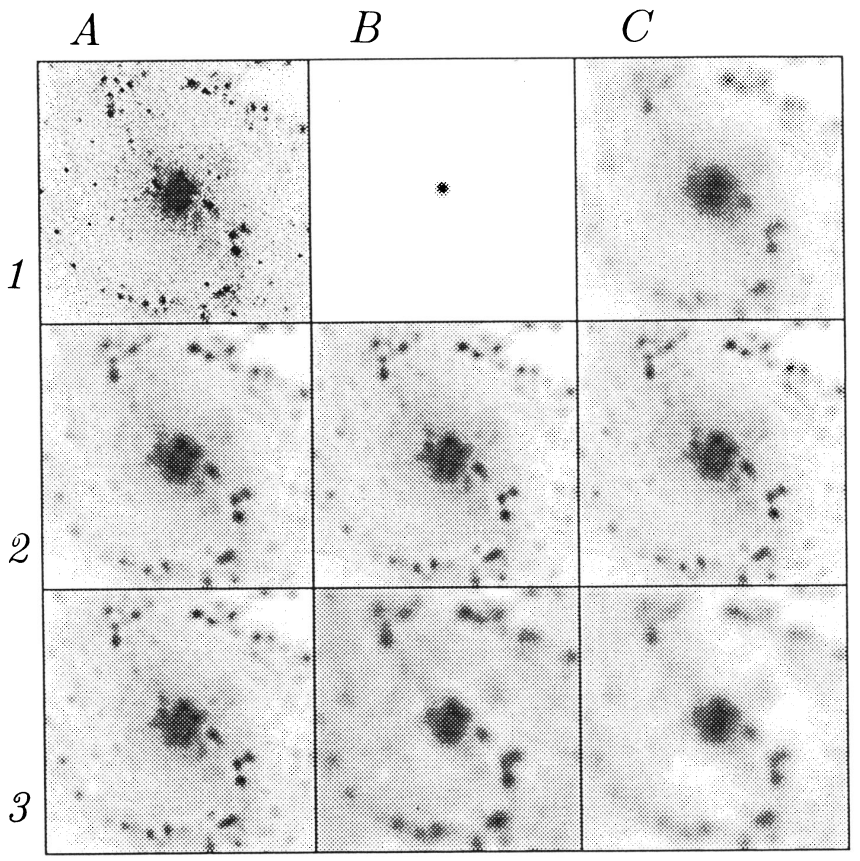

Figure 1: **A1**: The original test–image; **B1**: The adopted PSF (gaussian shape with $\sigma = 2pix$); **C1**: The raw image obtained convolving the original one with the adopted PSF; **A2**: Lucy 10 iterations; **B2**: Lucy 30 iterations; **C2**: accelerated Lucy 10 iterations with $p = 2$; **A3**: Burke 30 iterations ($w = 1$); **B3**: Hunt 10 iterations; **C3**: Hunt 30 iterations.

References

[1] Burke, J.J.: 1974, "Estimating objects from their blurred and grainy images", in *Electro–Optical Systems Design Conf.*, San Francisco, CA

[2] Hunt, B.R.: 1977, *IEEE Trans. Comput.*, **C–26**, 219

[3] RSI, *IDL User's Guide*, Research System Inc., Boulder, CO

[4] Lucy, L.B.: 1974, *Astron. J.*, **79**, 745

[5] Meinel, E.S.: 1986, *J. Opt. Soc. Am. A*, **3**, 787

[6] Richardson, W.H.: 1972, *J. Opt. Soc. Am.*, **62**, 55

[7] Tarasko, M.Z.: 1969, preprint, *FEI–156*, Obninsk (in Russian)