Tutorial: Image Restoration: deconvolution

This tutorial aims to study and test different image restoration methods for blurred and noisy images. Altough the different software tools contain functions for each filter the objective is to exhaustively code these methods to fully understand their principles. The reader can refer to [1] for more informations on the algorithms.
Some of the images are based on observations made with the NASA/ESA Hubble Space Telescope, and obtained from the Hubble Legacy Archive, which is a collaboration between the Space Telescope Science Institute (STScI/NASA), the Space Telescope European Coordinating Facility (ST-ECF/ESA) and the Canadian Astronomy Data Centre (CADC/NRC/CSA). You may access to these resources at https://hla.stsci.edu/

The different processes will be applied on the following images.

1 Damage modeling

A damage process can be modeled by both a damage function D and an additive noise n, acting on an input image f for producing the damaged image g.

$$g = D(f) + n \tag{1}$$

Knowing g, the objective of restoration is to get an estimate $\hat{f}$ (the restored image) of the original image f. If D is a linear process, spatially invariant, then the equation can be simplified as:

$$g = h * f + n \tag{2}$$

where h is the spatial representation of the damage function (called the Point Spread Function - PSF), and $*$ denotes the convolution operation. In general, the more knowledge you have about the function H and the noise n, the closer $\hat{f}$ is to f.

This equation can be written in the frequency domain:

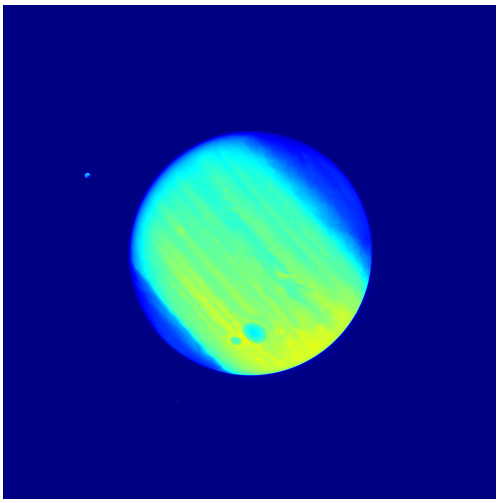
$$G = H.F + N \tag{3}$$

where the letters in uppercase are the Fourier transforms of the corresponding terms in the eq. 2.

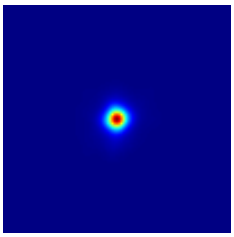
- Generate the 'chessboard' image.
- Generate a PSF corresponding to a blur (motion or isotropic).

Figure 1: Images that can be used for testing the different restoration algorithms. Intensity levels are represented in false colors.

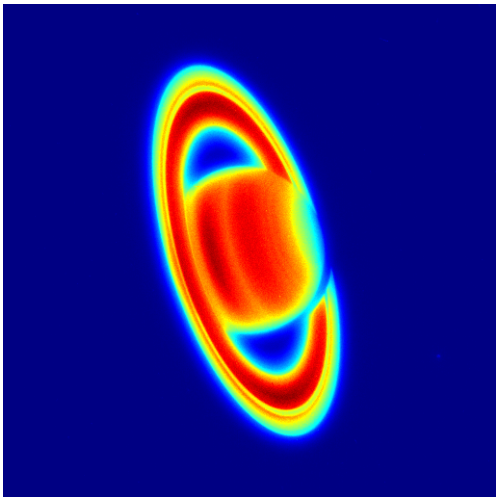
(a) Jupiter, from Hubble Space Telescope, ads/Sa.HST#ic3g01qlq.



(b) PSF for Jupiter image



(c) Saturn, HST.



(d) PSF for Saturn image

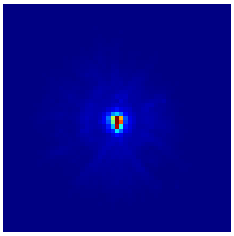
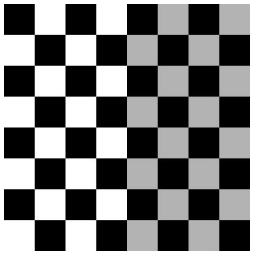


Figure 2: Chessboard image.



- Create a damaged image (a circular option is used, because the FFT implies that functions are periodical) and add a Gaussian noise.
- Visualize the different images.

MATLAB® has a checkerboard function for the 'chessboard' image. The fspecial function generates a convolution matrix.

The following function generates in python a checkerboard.

```
def checkerboard2(s=8):
    return np.kron([[1, 0] * 4, [0, 1] * 4] * 4, np.ones
        ↪ ((s, s)))
```

2 Simple case: no noise

The objective is now to restore the damaged image.

For the first steps, no noise will be added. The degradation functions is $g = h * f$. Knowing H (Fourier Transform of the psf h) and ignoring the noise in the damage model, a simple estimate of the initial image can be done by the inverse filtering:

$$f = FT^{-1}\left(\frac{G}{H}\right) \tag{4}$$

where FT^{-1} denotes the inverse Fourier transform.

- Try the inverse filter in the Fourier space. Why is this filter not working?
- Try to prevent division by 0, with $f = FT^{-1}\left(\frac{G}{H+\alpha}\right)$, α being a small constant value (e.g. $\alpha = 0.1$).

3 Wiener filter

The Wiener filter is an optimal filter that minimizes the following error:

$$e^2 = E\{(f - \hat{f})^2\}$$

where E denotes the expected (mean) value, f is the original image and $\hat{f}$ is the restored image.

The solution to this expression, within the frequency domain, is given by:

$$\hat{F} = \underbrace{\left(\frac{1}{H}\frac{|H|^2}{|H|^2 + R}\right)}_{H_w} G$$

The value of R can be arbitrarily fixed. Another way is to define $R = S_n/S_f$, where S_n and S_f denote the power spectrum of the noise n and the original image f, i.e. $|N|^2$ and $|F|^2$ (matrices), respectively. This ratio can be defined globally (as the constant) or locally (i.e. it is computed for each pixel). If the ratio S_n/S_f (function) is non null, it can be passed to the Wiener filtering process.

3.1 Simple case: no noise

In the noiseless case, the Wiener filter reduces to the inverse filter:

$$H_w = \begin{cases} \dfrac{1}{H(u,v)} & \text{if } |H(u,v)| \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

Try a Wiener filter for the noiseless case.

3.2 Noisy images

Generally, the ratio S_n/S_f used in the Wiener filter is replaced by a constant value equal to the ratio of the mean power spectrum:

$$R = \frac{\frac{1}{PQ} \cdot \sum_u \sum_v S_n(u,v)}{\frac{1}{PQ} \cdot \sum_u \sum_v S_f(u,v)} \tag{5}$$

where PQ denotes the matrix size (the number of elements).

see numel function.

1. Calculate the constant ratio R and code the Wiener filter .
2. Test the different algorithms on the image 'brain' damaged by a blur and an additive noise.

4 Iterative filters for noisy images

4.1 Van Cittert iterative filter

The VanCittert iterative filter is based on the following iterative relation :

$$f_{k+1} = f_k + \beta(g - h * f_k)$$

with $f_0 = g$, the damaged image.

Code and test on different cases a function that performs the VanCittert restoration filter.

It should have the following prototype:

```
function [ fr ] = vca( g, psf, n_iter)
% Van Cittert iteration algorithm for deconvolution
%
% g: degraded image
% psf: point spread function
% n_iter: number of iterations
```

```
def vca(g, psf, n_iter, beta=.01):
    """
    Van Cittert iterative filter
    g: noisy image
    psf: point spread function
    n_iter: number of iterations
    beta: Jansson parameter
    """
```

4.2 Lucy-Richardson filtering

We are now going to use the nonlinear restoration filtering of Lucy-Richardson that is based on a maximum likelihood formulation in which the image is modeled by Poisson statistics. This optimization leads to the solution if the following iterative equation converges:

$$f_{k+1}(x,y) = f_k(x,y) \cdot \left(h(-x,-y) * \frac{g(x,y)}{h(x,y) * f_k(x,y)} \right) \tag{6}$$

Code and test your own function for the Lucy-Richardson filtering process. It should have the same prototype as the VanCittert function. Remember that $\cdot$ is the classical multiplication (point to point) and that $*$ is the convolution operation. You then have the choice to make a direct call to the convolution function, or to perform this operation in the Fourier space.

5 Blind deconvolution: restoration with no a priori

If the PSF is unknown, one of the main difficulties in image restoration is to get an accurate estimate of this PSF in order to use it in the deconvolution algorithms. These methods are called blind deconvolution methods. Generally, the PSF is estimated in an iterative way from an initial PSF.

1. Generate a damaged image of 'chessboard' with a Gaussian blur and an additive Gaussian noise.
2. Restore from a blind deconvolution the damaged image and compare the results with the previous ones.

Use the function deconvblind.

Informations

Use the function skimage.restoration.unsupervised_wiener for example.

6 Astronomy images

A classical file format used in astronomy is the FITS format.

The **FITS** files can be read using the function fitsread. For example, the following commands will read an image and its PSF.

```
% 1st example
saturn = fitsread('saturn.fits');
psf = fitsread('saturn_psf.fits');

% 2nd example
jupiter = fitsread('ic3g01qlq_flt.fits', 'image', 1);
psf = fitsread('PSFSTD_WFC3UV_F275W.fits');
jupiter_psf = p(:,:,1); % select 1st psf only
```

With python, fits images can be manipulated with the astropy module. You can use the following code to display an image.

```
hdu_list = fits.open('ic3g01qlq_drz.fits');
image = hdu_list[1].data;
image = np.nan_to_num(image);
plt.imshow(image, cmap='gray')
plt.show()
```

References

[1] Daniel Sage, Lauréne Donati, Ferréol Soulez, Denis Fortun, Guillaume Schmit, Arne Seitz, Romain Guiet, Cédric Vonesch, and Michael Unser. Deconvolutionlab2: An open-source software for deconvolution microscopy. *Methods*, 115:28 – 41, 2017. Image Processing for Biologists. 1