

User Guide for OPTMIsME

OPTIMIsationStochastique Imaging MultispectralE , Image-J plugin

Dominique Benielli

Labex Archimède

9 mai 2017



Table des matières

1	Introduction	2
1.1	Historical	2
1.2	abstract	2
1.3	Scientifics Issues	2
1.4	Optimisme-ImageJ	2
1.4.1	ImageJ	2
2	Plugins OPTIMISME	3
2.1	Private Distribution, .java	3
2.2	Private Distribution, .class	3
2.3	Private Distribution, .jar	3
2.4	Public Distribution	3
2.5	Libraries	3
2.6	Installation	4
2.7	Launch	4
2.7.1	Input parameters	4
2.8	Input Files	6
2.9	Calculus	6
2.10	Results	6

3	package OPTIMISME	6
3.1	The java class	6
3.2	Test Coverage	7
3.3	Tests	8

1 Introduction

1.1 Historical

Optimism is a french project for young researchers financed by the Gdr ISIS in 2013. After it has been continued and extant with the financial support of CNRS within the project Imag'In in 2015-2016.

This project is based on the restauration of data in bi-photon microscopy. It composed in two part a theoritical approach and application.

The theorical part linkes Majorization-Minimization approach with proximals ways. This both approaches will be introduced in the applicativ part of the image restauration from bi-photon photometer. Furthermore this approach included a analyse of noise and the non-stationarity of the PSF.

1.2 abstract

Modern approachs of invers problems resolution in the field of multi or hyperspectral imaging are based on variational formulations The goal of this study is to provide a generalization of parallel methods used recent advances of stochastic optimization, for the treatment of massive big data. Provided algorithms resolve multispectral deconvolution problem in biphoton microscopy.

1.3 Scientifics Issues

The in vivo observation of a mouse brain by a photonic microscop leads to direct images at cellular scales , this images need to be reconstructed. This problem resolution is equivalent to solve a inverse problem, that is the object of optimisme algoritnms. The knowledge issue of the PSF is solved by observation of fluorescent micro-drop. PSF are obtained by the observation of (0.5 μ m) micro-drop for differents depth. then the PSF varies with the depth (z coordinate).

1.4 Optimisme-ImageJ

1.4.1 ImageJ

ImageI is a software of images processing dedicated to multidimentional scientifics images. ImageJ can be easily extended adding pluggins and scripts. imageJ web site :

- imageJ homepage

2 Plugins OPTIMISME

ImageJ gives the possibility to user to develop and install our own pluggin

- imageJ Plugin Install

2.1 Private Distribution, .java

- Add the directory OPTIMISM in the plugin directory of ImageJ. Then start ImageJ, and
- Select on Plugins > Compile and Run.

2.2 Private Distribution, .class

A other way is to use the .class and only

- Copy the *.class file to the ImageJ/Plugins folder or subfolder.
- Go to Help>Refresh Menus or (better) restart ImageJ.

2.3 Private Distribution, .jar

Copy the OPTIMISME.jar file to the ImageJ/Plugins folder plugin/jars Go to Help>Refresh Menus or (better) restart ImageJ.

2.4 Public Distribution

For version 1.0 a recording of the pluggin by the meccanism of ImageJ pluggin registration is wishing. We recommended a private site repository lighther than the imageJ git repository.

- update_imageJ

2.5 Libraries

Libraries used for the Optimims algorithm are :

- *edu.emory.mathcs.jtransforms* For 3D et 2D FFT and IFFT. on licence BSD clause 2. Non contaminant licence.
- *org.ejml* a part of ejml library for matrix calculus and pseudoInvers function. on Apache licence Non contaminant licence.
- *org.apache.commons.math3* a part of apache common library for tricubic intepolation. on Apache licence Non contaminant licence.

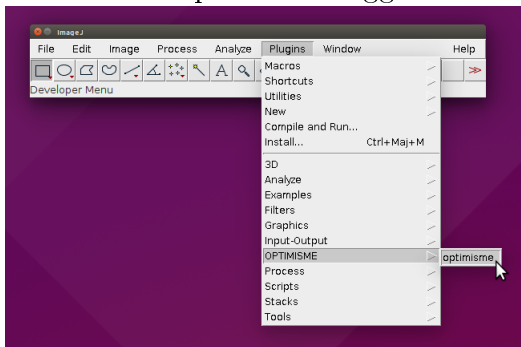
2.6 Installation

In the case of the java installation Copy the OPTIMISME directorie in the pluggins dirrectory of ImageJ and In the menu, select *Plugins/Compile and Run*. The file browser open itself and you can select the file `_optimisme.java`

2.7 Launch

The Pluggin is accessible through the pluggin menu of ImageJ. The Optimimisme pluggin assume at least two open images -the image to decovolved and the PSF. Note : that the present version is established for the mode with one PSF.

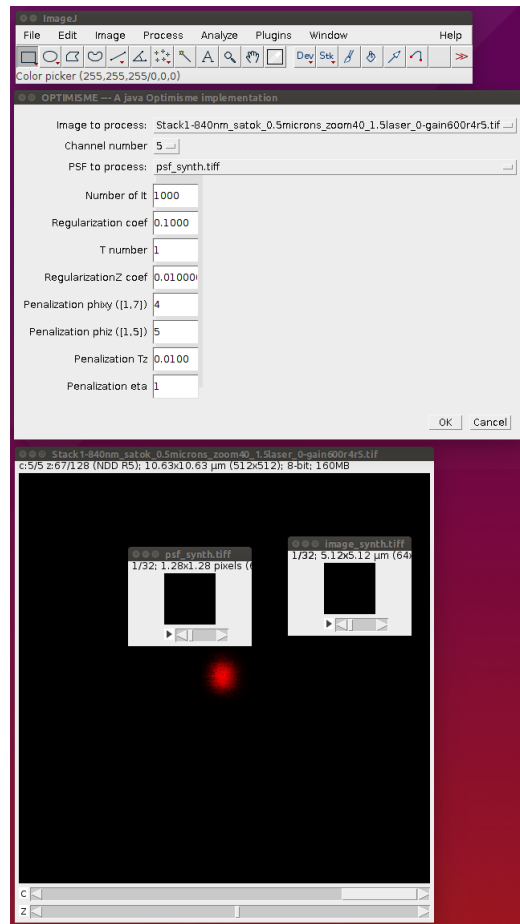
Launch the Optimism Pluggin as follow :



The initial window of the pluggin Optimisme, the user specifies input image, and its the channel number, psf image and others parameters of the algorithm.

2.7.1 Input parameters

IN the window of Optimism Pluggin, fields of Input parameters are filled whitth the default values, they can be modified by the user.



- NbIt : iteration number
- regul : regularization coefficient
- T :
- regulZ : regularizationcoefficient for Z coordinate (depth)
- phixy :
- phiz :
- TZ :
- eta :
- Channel number : In the case of the input image contains multi chanel the user can select the the channel image used for convolution

2.8 Input Files

Before your launch Optimisme Optimisme needs as input file two images (image and psf). ImageJ accept different format (tiff format,lsm directly from microscope) only

- File>Open open at least two files one for Image and one for the PSF Indeed, Optimisme assumes 3D files, images with a depth dimension.

2.9 Calculus

The algorithm first compares the resolution image and PSF, If both resolutions are different the re-sampling of PSF is process a, at the end of this first stage the image PSF resampled is plotted. The algorithm makes after the process itself, and finish by the plot of the deconvolved image.

2.10 Results

The optimisme pluggins draws at least one image

- at the end of the calculation a image is open it is the decoconvolved image, the user can save it with the use of menu File>Save as
- In the case of the both input image (image and psf) do not have the same resolution the algorithm first regrids the psf, at the end of the regridding the programm displays the regradind psf with the same resolution on input image, The user can save it for futher run of the program and use it for all image correspondind to the same resolution by File>Save as menu.

3 package OPTIMISME

3.1 The java class

The java class of algorithm Majorisation/Minimisation in ImageJ pluggin, and other dependencies

```
/ImageJ/pluggins/OPTIMISME/  
├─ org/  
│   └─ ejlm/  
│       ├── alg/  
│       ├── data/  
│       ├── factory/  
│       ├── interfaces/  
│       ├── ops/  
│       ├── simple/  
│       └─ UtilEjml.class  
└─ ...
```




```

fftj.jar
doc/
|_ index.html
|_ ...
optimisme_.java
data
|_ image_synth.tiff
|_ psf_synth.tiff
|_ psf_synth_subsampled.tiff
optimisme
|_ MM.java
|_ PSFPreparator.java
|_ MMCal.java
|_ test
|_ AllTests.java
|_ MMCalTest.java
|_ MMTTest.m
|_ PSFPreparatorTest.m










```





3.2 Test Coverage

Tests files are in test folder the coverage by Emma leads to :

 optimisme (3 avr. 2017 18:01:58) >  OPTIM >  OPTIM



















OPTIM

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 optimisme		92 %		89 %	93	549	120	1 317	20	133	1	6
 default		0 %		0 %	30	30	128	128	4	4	1	1
 optimisme.test		99 %		90 %	17	237	53	1 816	6	132	1	4
Total	1 568 of 56 366	97 %	162 of 1 066	85 %	140	816	301	3 261	30	269	3	11

 optimisme (3 avr. 2017 18:01:58) >  OPTIM >  OPTIM >  optimisme

 [Source Files](#)  [Sessions](#)

optimisme

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 MM		82 %		60 %	52	104	89	463	10	30	0	1
 MMCal		98 %		95 %	27	364	14	643	2	84	0	1
 MM.MMPParameter		28 %		n/a	1	2	11	24	1	2	0	1
 TriCubicInterpolator		0 %		n/a	7	7	6	6	7	7	1	1
 MMCal.Squeeze		100 %		95 %	6	67	0	124	0	8	0	1
 PSFPreparator		100 %		100 %	0	5	0	57	0	2	0	1
Total	719 of 9 528	92 %	89 of 804	89 %	93	549	120	1 317	20	133	1	6

3.3 Tests

source files	file size	%CPU	%MEM	%TIME
stark1_840 *2	342 (512*512*261)	100	61	763 -stop
stark1_450 *2	225 (512*512*172)	100	61	763 -stop
stark1_840 * psf_synth	167 (512*512*128)	100	7.4	22 :20 ?
stark1_840 * psf_synth	167 (512*512*128)	324	81	3 :00