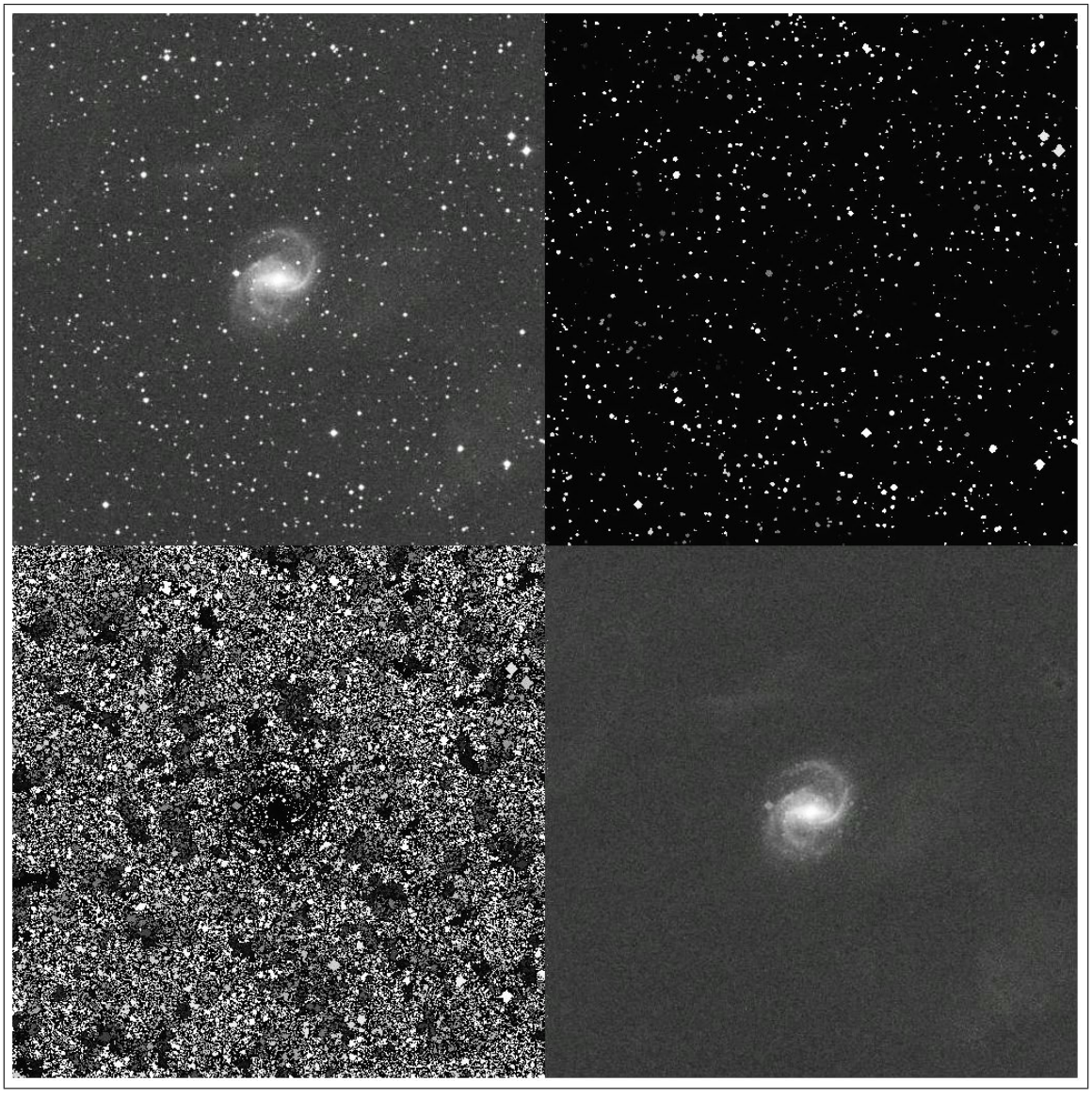
nfigi
*v*0.8.2

User's guide

A. BAILLARD
Institut d'Astrophysique de Paris

April 26, 2007



Contents

1 What is *nfigi*?

nfigi uses morphological operators to find and remove parasite stars from images of galaxies. There several constraints on the images:

- They must be in fits format.
- The galaxy must be centered in the image in order not to alter the nuclei of the galaxy.

This software is part of a bigger project called EFIGI[1] which goal is to compute morphological identification of galaxies. In order to achieve this goal, a cleaning procedure is needed, as explained in [1].

As an efficient cleaning technique could be useful for other aims, *nfigi* (*n* coming for *nettoyage*) is released as a single package.

2 Installing the software

2.1 Obtaining *nfigi*

The easiest way to obtain *nfigi* is to download it from the official website[2], or alternatively from the current official public subversion repository[3]. At this address the latest versions of the program are available as standard `.tar.gz` Unix source archives as well as documentation.

2.2 Software and hardware requirements

nfigi has been developed on a GNU/Linux system and should compile on any POSIX-compliant system.

The software is run in (ANSI) text-mode from a shell. A window system is therefore unnecessary with present versions.

Memory requirements depend on the size of the input image. The algorithm requires 10 times the memory size of the image for temporary structures that are used during the building of the tree. This is the main weakness of a tree construction and using swap space could severely lower the performance.

2.3 Installation

To install, you must first uncompress and unarchive the archive:

```
tar xvzf nfigi-x.x.x.tar.gz
```

A new directory called `nfigi-x.x.x` should now appear at the current position on your disk. You should then just enter the directory and follow the instructions in the file called "`INSTALL`".

3 Technical overview

3.1 Quantization

To improve the speed of the cleaning process, it is adviced to convert to the image to a grey level one if necessary. The quantization formulae is the following:

[1] `http://www.efigi.org`
[2] `http://terapix.iap.fr/soft/nfigi`
[3] `http://terapix.iap.fr/wsvn/index/public/software/nfigi/`

$$Image_{int}[i] = round(\frac{Image_{float}[i] - min(Image_{float})}{q})$$

where $Image_{int}[i]$ is the i^th pixel of the resulting integer image, $Image_{float}[i]$ the i^th pixel of the source floating-point image $min(Image_{float})$ the minimal grey level within the floating-point image and q the quantization step. Function $round$ rounds to the nearest integer.

This quantization technique that has several advantages :

- the image is positive (min substraction)
- the dynamic range of the image depends of the defined step and does not depend of the maximum value of the floating-point image. This is very convenient to set a treshold for the batch mode (see next section).

Basically, we want to process some image segmentation to remove some components.

3.2 Tree representation

As discussed in [5, 3], an efficient technique for image processing is to represent an image as a tree. Salembier also explains how to use morphological filters on such a tree. Mathematical morphology is a tool for extracting image components that are useful for representation and description, it was originally developed by [6] but area operators were specified by [8] and granulometry was introduced by [2]. Morphology can provide boundaries of objects, their skeletons, etc. It is also useful for many pre- and post-processing techniques, and that is what we are interested in.

Generally speaking most morphological operators are based on simple expanding and shrinking operations. These operators are particularly interesting because they are easy to understand and fast to compute on binary or grey level images. We use attribute operators that are shape preserving in order not to alterate galaxies.

Our max tree computation is based on the method proposed by [4]. It is a quasi-linear algorithm based on Tarjan's union-find principle. Building the component tree is easily understandable when using a topographical analogy. We can see the image as a relief with the grey-level of a point corresponding to its altitude. The surface is completely covered by water, then the level of water decreases. Islands (maxima) appear. These islands form the leafs of the tree. As the level of water decreases, islands grow, building the branches of the tree. When several islands merge, they create the forks of the tree. The root is the lower grey-level isophote (the background) and represents the whole image.

By building the tree, we can add some usefull information to be used later. Currently, our nodes include the lower grey-value and the global area of the corresponding component. One can see how easy it is, for example, to apply an area opening on such a tree as we just need to cut branches which root nodes have an area lower than the given parameter.

The main difference with Najman's structure is that we do not use lists of sons but a left-son right-brother architecture. As our images are grey level floating-point images, we have to quantify them by building the tree because computing morphological operators on integer values is much faster.

3.3 Morphological operator sequence

We build the component tree of the image and then apply the following basic operators:

(a) Area opening with a given size to locate bright areas and white top-hat to keep only pixels from the area opening. In our algorithm, only the white top-hat is computed (as it is just the substraction of the area opening from the source image) ;

(b) Tresholding to create a mask that contains only the brightest spots, areas that are much brighter than their neighbourhood ;

(c) Labeling using a 4-connexity to obtain distinct components ;

(d) Central component removal corresponding to the nuclei of the galaxy ;

(e) Dilatation(s) using a 8-connexity to retrieve a replacement value for each labeled component ;

(f) Cleaning according to the position of the component:

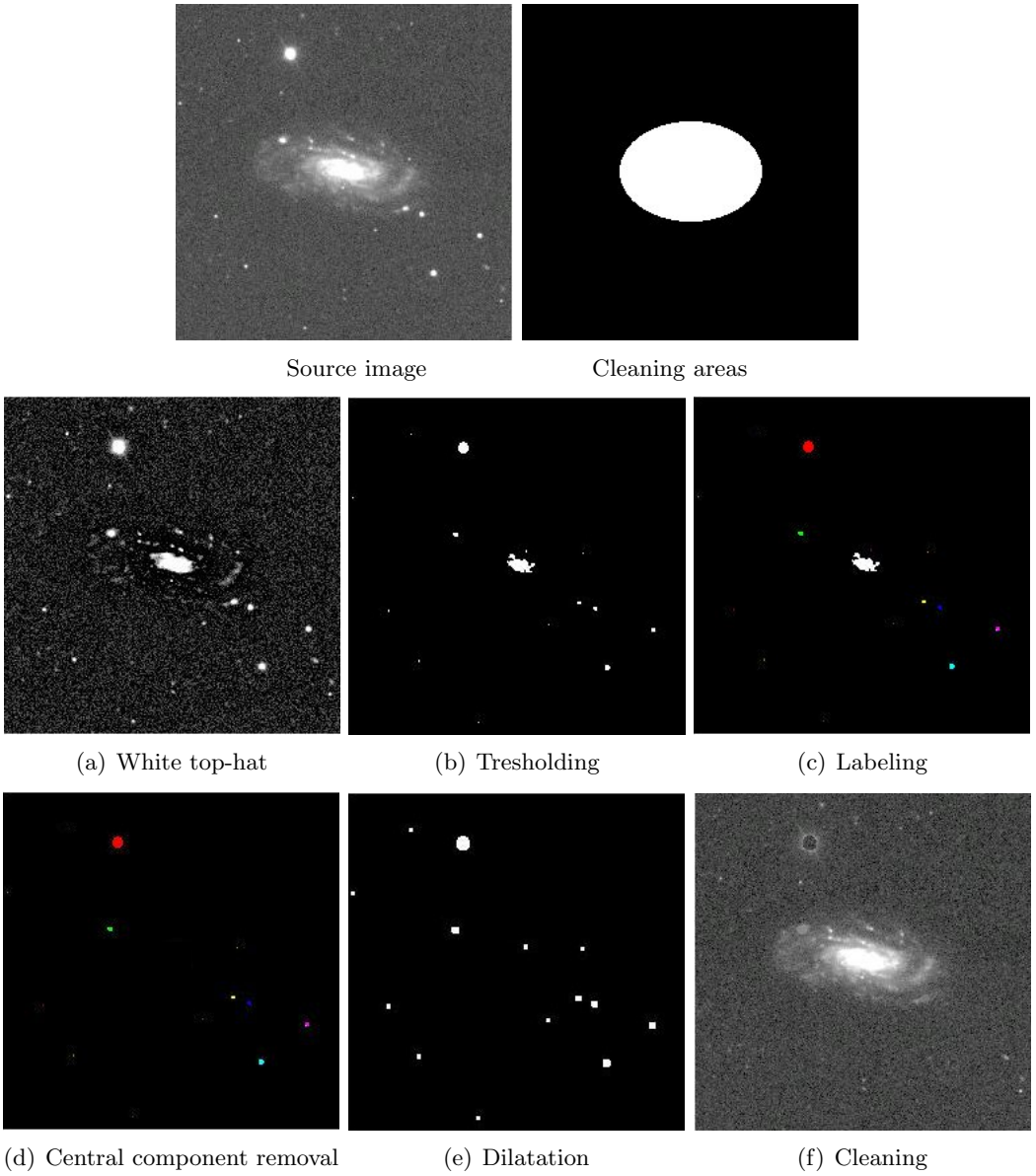
- Inside the galaxy: local minima on the original image to level stars according to the dilated components ;
- Outside the galaxy: Inpainting with the symmetric part of the image.

Step 5 is really specific to galaxies. It implies that the galaxy is centered on the image. As we do not want to modify the center of the galaxy, we remove its corresponding component from the tree. Figure 1 illustrates the whole process on a very common image.

The limit between the two cleaning regions (inside and outside the galaxy) is computed according to the statistics of the images. Indeed, this limit is an ellipse for which:

- the center of the ellipse is the center of the image.
- the axes are horizontal and vertical.
- each axe containing 3/4 of the flux along this axis.

This technique was previously used in [7].



Source image Cleaning areas

(a) White top-hat (b) Tresholding (c) Labeling

(d) Central component removal (e) Dilatation (f) Cleaning

Figure 1: Example of a cleaning process.

The area opening and the tresholding need a parameter to be set. Obviously, the area opening needs an area size (700 for images of 256x256 pixels as on figure 1). If the size is too small, big stars will not be totally selected, only there heart will be leveled. If the area is too large, different components could be joined. The treshold must also be set to compute a significant mask. It is sensitive to set a good treshold value in order not to alterate the whole image by treating local maxima that are part of a peculiar galaxy or just background or noise.

3.4 Future improvment

There are two main features that should be achieved in a future version of *nfigi*:

- The limit between the two treatment areas (local minimum and inpainting) should be defined bu some other statistics, for example according to the centered second-order moments or to the histogram of the image. An other possibility is to use morphological data present in the FITS header.

- The local minimum within the galaxy should be replaced by a sort of local inpainting that interpolates the edges detected around the component to remove.

Some other minor features could be added as, for instance, connexities for building and dilatations. The minimum value used for the quantization could be replaced by a value of the background level.

4 Using *nfigi*

nfigi is run from the shell with one of the following syntaxes:

```
nfigi -i input-file -o output-file
        [-a size] [-t threshold] [-q step] [-l num] [-n]

nfigi -d directory
        [-a size] [-t threshold] [-q step] [-l num] [-n]

nfigi -h | --help
```

The parts enclosed within brackets are optional.

4.1 Detailed description of the configuration parameters

4.1.1 File management

nfigi can handle FITS file. Use option -i to clean a single image. You can also use option -o to choose the name of the output file. If no output file is given, *nfigi* automatically creates an image called source.*rs*.fits if the input file was source.fits.

nfigi can also be used in batch mode. In that case, option -d gets a directory, read every fits file in it and clean them. Output images are automatically named sourceN.*rs*.fits. Options -i and -o are ignored when -d is invoked.

4.1.2 Area opening size

The area opening parameter is a strictly positive integer. Giving an invalid value reset to the default value *40*.

4.1.3 Treshold value

The treshold value is a positive or null integer value. Giving an invalid value reset to the default value *100*.

4.1.4 Quantization step

The quantization step is a positive floating-point value. Giving an invalid value reset to the default value *0.1*.

4.1.5 Number of dilatations

The number of dilatations to process is a positive or null integer value. Giving an invalid value reset to the default value *1*.

4.1.6 Intermediary images

nfigi can output images for each step of the cleaning process. These images are the following:

- source.*rs1wth*.fits is the result of the white-top hat and the first step.
- source.*rs2tre*.fits is the result of the treshold and the second step.
- source.*rs3com*.fits is the result of the labeling and the third step. Each component has an unique label (from 1 to the number of components).
- source.*rs4dil*.fits is the result of the dilatation(s) and the fourth step.
- source.*rslim*.fits is the representation of the two cleaning regions used for the last step.

References

[1] A. Baillard and al. Project efigi: Automatic classification of galaxies. In *ADASS XV*, volume 351, pages 236–239, 2005.

[2] E.J. Breen and R. Jones. Attribute openings, thinnings and granulometries. *Computer Vision and Image Understanding*, 64:377–389, 1996.

[3] Henk J. A. M. Heijmans. Connected morphological operators for binary images. *Computer Vision and Image Understanding: CVIU*, 73(1):99–120, 1999.

[4] L. Najman and M. Couprie. Quasilinear algorithm for the component tree. In *Vision Geometry XII. Edited by Latecki, Longin Jan; Mount, David M.; Wu, Angela Y. Proceedings of the SPIE, Volume 5300, pp. 98-107 (2004).*, pages 98–107, April 2004.

[5] P. Salembier and L. Garrido. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. In *IEEE Transactions on Image Processing*, 2000.

[6] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982.

[7] M. Thonnat. Automatic morphological description of galaxies and classification by an expert system. *Rapports de recherche INRIA*, 387, March 1985.

[8] L. Vincent. Morphological area openings and closings for greyscale images. In *Proc. Shape in Picture '92, NATO Workshop*, Driebergen, The Netherlands, September 1992. Springer-Verlag.