

# **CREYC's Magic for Image Computing** A Full-Featured Open-Source Framework for Image Processing

http://gmic.eu

Présentation Equipe IMAGE / GREYC, Avril 2017





- Research in the field of image processing at the GREYC lab of ENSICAEN / CNRS / University of Normandy (Caen).
- ⇒ Design of innovative algorithms to solve generic image processing problems (denoising, enhancement, segmentation, feature detection,...).





 Frequent collaborations with companies / laboratories having specific images to process.



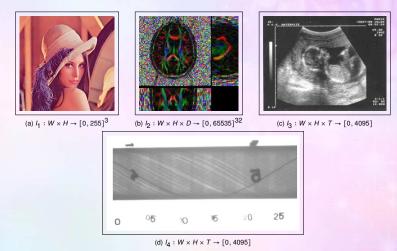
⇒ Various image data coming from very diverse sensors.

#### Context



(ロ) (目) (三) (三) (三) (日) (日)

- Image data are diverse: 2D, 2D+t, 3D, 3D+t, vector or matrix-valued pixels, float values, ...
- $\Rightarrow$  We stray far from usual 2D color pictures!





- Needs for tools to visualize / explore data, convert image formats, apply classical IP operators (filtering, geometric transformations, frequential analysis, ...) for very generic image data, sometimes on thousands of images at the same time.
- Typical "technical" question we can ask for:

"How to convolve 500 volumetric images having 32 channels each by 3d anisotropic gaussian kernels ?"



#### **Motivations**



- ⇒ Very few open-source tools exist for these tasks. They tend to be either:
  - Easy to use, but not generic enough for our image data (GIMP, ImageMagick, GraphicsMagick, ...).
  - Or very flexible, but reserved for experienced programmers (require the writing of code, using specialized external libraries).
- We did like others: The team has developed generic libraries for image processing: **CImg** and **Pandore** (in C++):



http://cimg.eu https://clouard.users.greyc.fr/Pandore/

#### **Motivations**



- In practice, the libraries are used only by a few hundred "experimented" programmers.
  - $\Rightarrow$  Cause: High diversity of people in the image processing field !



Mathematicians

Physicists



Programmers



Biologists ...

(ロ) (日) (日) (日) (日) (日) (日)

⇒ We clearly need more simpler interfaces (than C++ libraries) if we want to enlarge our audience.





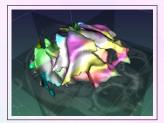
http://gmic.eu

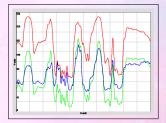
- For the users : Define different user interfaces to do image processing. (provided interfaces are more or less friendly and powerful depending on the user's skill.
- For the developers : Ease algorithm prototyping and maintenance.
- → Technical mean : Definition of a full-featured, concise script language for the processing of generic image data (G'MIC language). Interpreter used as a base layer for all user interfaces.



- Definition of a comprehensive and concise script language for the processing of generic image data (G'MIC language).
  - Full-featured: More than 950 commands available for the visualization, filtering, geometric and colorimetric transformations, feature extractions, 3d rendering, matrix calculation, primitive drawing, ...
  - $\rightarrow$  Documentation (.pdf) has more than 450 pages.
  - Reference documentation: http://gmic.eu/reference







< ロ > < 同 > < 三 > < 三 > < 三 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □



 Distribution of a open-source implementation of the G'MIC language interpreter (as a C++ library).

- Integration: Possible integration of G'MIC features in third-party software or plug-ins (photo retouching, digital painting, video editing software, ...).
- Free software license: Distributed under the CeCILL license (GPL-compatible).
- → Some existing integrations of libgmic to date:
  - ★ Krita, digital painting software.
  - \* Photoflow, non-destructive photo retouching software.
  - **±** *EKD*, video post-production software.



- Provide user interfaces for everyone, embedding the G'MIC language interpreter (multi-platform).
  - gmic: Command-line tool to manipulate generic images.
     Complementary to the CLI tools from ImageMagick / GraphicsMagick.

```
dtschump@ _____;~$ gmic ~/work/img/lena.bmp -blur 3 -mirror x
[omic]-0./ Start G'MIC parser.
[gmic]-0,/ Input custom commands file '/home/dtschump/work/src/resources.gmic' (added 14 commands, total 1121).
[gmic]-0./ Input custom commands file '/home/dtschump/work/src/gmic/src/gmic_def.gmic' (added 1107 commands, total 222
8).
[gmic]-0./ Set dynamic 3d rendering mode to flat-shaded.
[gmic]-0./ Input file '/home/dtschump/work/img/lena.bmp' at position [0] (1 image 512x512x1x3).
[gmic]-1./ Blur image [0], with standard deviation 3 and neumann boundary.
[gmic]-1./ Mirror image [0] along the 'x'-axis.
[qmic]-1./ Display image [0] = 'lena.bmp*'.
lena.bep* (512x512x1x3) : this = 0xbf9852e4, size = 1/16 [3072 Kb], data = (CImg<float>*)0xa027a44..0xa027a5b.
 [0]: this = 0xa027a44, size = (512,512,1,3) [3072 Kb], data = (float*)0xb73ba008..0xb76ba007 (non-shared) = [ 203.2
86 207.053 210.367 212.452 212.914 211.713 209 205.179 ... 65.5009 63.9167 62.7955 61.9959 61.279 60.5754 59.9074 59.3
564 ], min = 9.43401, max = 250.19, mean = 128.229, std = 55.711, coords_min = (511,440,0.1), coords_max = (68,57,0.0)
[gmic]−1./ End G'MIC parser.
dtschump@ :~$
```



\$ gmic lena.bmp -blur 3 -sharpen 1000 -noise 30 -+ "'cos(x/3)\*30'"

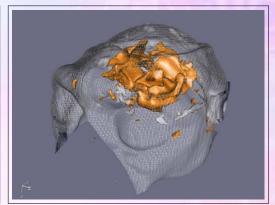


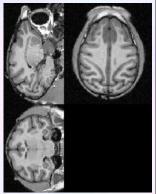




(ロ) (日) (日) (日) (日) (日) (日)

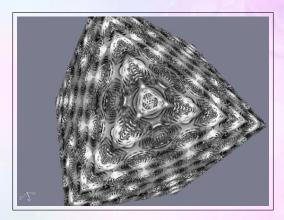
\$ gmic reference.inr -flood 23,53,30,50,1,1,1000 -flood[-2] 0,0,0,30,1,1,1000 -blur 1 -isosurface3d 900 -opacity3d[-2] 0.2 -color3d[-1] 255,128,0 -+3d







#### \$ gmic -isosurface3d "'sin(x\*y\*z)'",0,-10,-10,-10,10,10,10,128,128,64



### Example: using the CLI tool "gmic"

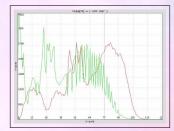


\$ gmic milla.bmp -f '255\*(i/255)^1.7' -histogram 128,0,255 -a c -plot

#### is the G'MIC equivalent to this C++ code (using CImg):

```
#include "CImg.h"
using namespace cimg_library;
int main(int argc,char **argv) {
    const CImg<>
    ing ("milla.hmp"),
    hist = img.get_histogram(128,0,255),
    img2 = img.get_histogram(128,0,255);
    (hist,hist2).get_append('c').display_graph("Histograms");
    return 0;
```





◆□ ▶ ◆□ ▶ ◆□ ▶ ◆□ ▶ ◆□ ● ● ● ●



< ロ > < 同 > < 三 > < 三 > < 三 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

- First release done in August 2008  $\implies$  very few downloads (approx. 300/month).
- But... Writing image processing pipelines in G'MIC also allows to design and develop artistic filters and effects easily...

```
$ gmic lena.jpg -pencilbw 0.3 -o gmic_lena1.jpg
$ gmic lena.jpg -flower 10 -o gmic_lena4.jpg
```



→ Why not writing a G'MIC plug-in for GIMP ?



- Provide user interfaces for everyone, embedding the G'MIC language interpreter (multi-platform).
  - gmic\_gimp: Plug-in for GIMP that provides hundred of G'MIC -based image filters for 2D RGB or RGBA images.

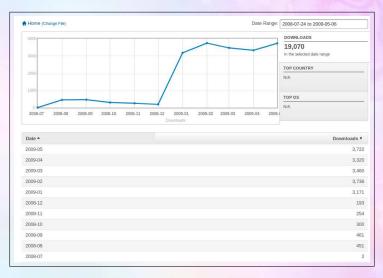


#### Before / after the GIMP plug-in



< ロ > < 同 > < 三 > < 三 > < 三 > の Q ()

First release of the G'MIC plug-in GIMP in January 2009.
 → Significant increase of the downloads and page views.





(ロ) (目) (三) (三) (三) (日) (日)

- Provide user interfaces for everyone, embedding the G'MIC language interpreter (multi-platform).
  - G'MIC Online: Web service for applying image filters and effects online (requires only a web browser).

https://gmicol.greyc.fr

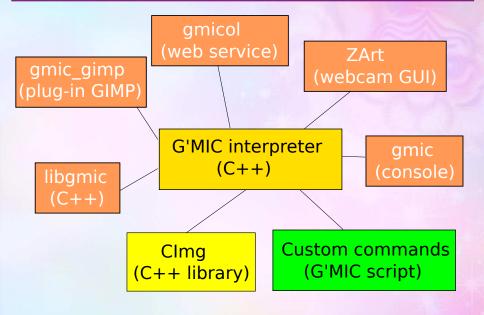
Prof	204 Bites Frame Exclance Statutes Statutes Statutes Statutes Statutes Statutes Statutes Statutes Statutes Statutes Statutes Statutes Constat C	Nilocs       Readular     0       Biocolones     0       Seconstraine     0       Seconstraines     0 <tr< th=""></tr<>
		Reset Resder full image Cancel



- Provide user interfaces for everyone, embedding the G'MIC language interpreter (multi-platform).
  - ZArt: Qt-based interface for the manipulation of video sequences (webcam or video file). Used as a demonstration platform.







<ロ> < 母> < 国> < 国> < 国> < 国> < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 >



(ロ) (日) (日) (日) (日) (日) (日)

- Script language (interpreted).
- Define a set of native "low-level" commands :
   e.g, -convolve, -display, -if, -then, -else, ...
   → C++, compiled, often multi-threaded (OpenMP).
- Define a set of custom commands :
   e.g, -polygonize, -apply\_gamma, -x\_pacman, ...
   → Grouped in the G'MIC standard library.
- Users can defined their own custom libraries of commands :
   e.g, \$ gmic user.gmic -my\_command ...
   → Versatile and evolutive framework (standard library updatable from network).
- Most G'MIC commands are actually custom commands. 200 native, +750 custom.
- Commands to manage also display windows and user events.

## The G'MIC programming language



(ロ) (目) (三) (三) (三) (日) (日)

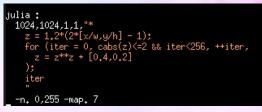
• Portion of the -x\_pacman command:

```
score0="10" score1="100" score2="1000" score3="5000" score4="Argh!"
-repeat 5
 0 -t. ${score$>},0,0,13,1,255,255,255 -autocrop. 0 -expand_xy. 1,0 --dilate. 3
 -nm. scorem$> -nm.. score$>
-done
time4=255,255,255 time3=255,255,32 time2=255,128,32 time1=255,32,32
-repeat 11 0 -t. $<" s".0.0.23.1.${time{min(4.round(($<+1)/2))}} -nm, time$< -done
0 -t. "Get Ready!",0.0.32.1.255 -autocrop. 0 -expand xy. 4.0 --dilate. 8 -r., 100%,100%.1.3
-nm., get_ready -nm, get_readym
0 -t. "Game\n0ver!",0,0,53,1,255 -autocrop. 0 -expand_xy. 4.0 --dilate. 8 -r., 100%,100%,1,3
-nm., game_over -nm, game_overm
# Start game.
score=0 level=-1 lives=3 is_quit=0
-do
  # Build new level if necessary.
  -if {$level<0}
    _rlevel=33 _glevel=33 _blevel=255
    -_pacman_map_level{((-$level-1)%6)+1} mu={w} mh={h} mu2={int(w/2)} mh2={int(h/2)}
    -if {$level<-6} -replace, 3.2 -endif
    -nm, map0 -i[map] .
    # Precompute valid directions on each map point, and shortest path to the ghost's home.
--shift[map] -1,0 --shift[map] 0,-1 --shift[map] 1,0 --shift[map] 0,1 -a[-4--1] z -!=, 1 -r
    --==[map] 1 100%,100% -=, 1,$mw2,$mh2 -distance, 1,...,3 -rm...
    -f, 'if(i==2.0.if(i==8.1.if(i==1.2.if(i==4.3.i))))' -nm, path
    --==[map] 2 pacdots={is} -rm.
    level={-$level}
  -endif
```

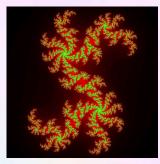


イロト ( 同 ) ( 三 ) ( 三 ) ( 三 ) の( ()

Embedded math expression evaluator:



• \$ gmic user.gmic -julia (0.631 seconds to run)





(ロ) (目) (三) (三) (三) (日) (日)

• Embedded math expression evaluator:

```
julia :
1024,1024,1,1,"*
z = 1,2*(2*[x/w,y/h] - 1);
for (iter = 0, cabs(z)<=2 && iter<256, ++iter,
z = z**z + [0,4,0,2]
);
iter
"
_ -n. 0,255 -map. 7
```

- → JIT compiler : expression is compiled by G'MIC into specific bytecode for faster evaluation.
- → OpenMP : expression is evaluated with multiple threads (when possible).
- → Manage usual calculations with scalars, complexes, matrices (SVD, solve, ...).
- → Expression may contain variables, loops, conditions, etc... (looks like C code).



#### • NL-means code :

```
nlmeans_expr : -check "${1=10}>0 && isint(${2=3}) && $2>0 && isint(${3=1}) && $3>0"
       -fill "
         const sigma = $1: # Denoising strength.
         const hl = $2: # Lookup half-size.
4.
         const hp = $3; # Patch half-size.
         value = 0;
6.
         sum weights = 0;
         for (a = -hl, a<=hl, ++a.</pre>
8.
           for (p = -hl, p<=hl, ++p,</pre>
9.
             diff = 0:
             for (s = -hp, s <= hp, ++s,
              for (r = -hp, r <= hp, ++r)
                 diff += (i(x+p+r,y+q+s) - i(x+r,y+s))^2
             weight = exp(-diff/(2*sigma)^2);
             value += weight*i(x+p,y+q);
             sum_weights += weight
         value/(1e-5 + sum weights)
```

• Run with \$ gmic user.gmic leno.png -nlmeans\_expr 35,3,1

# The G'MIC programming language



#### • \$ gmic user.gmic leno.png -nlmeans\_expr 35,3,1



Fig.5.1. Crop of a noisy version of the Leno image, degraded with gaussian noise, std=20.

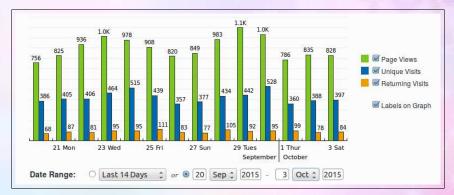
Fig.5.2. Denoised version using the NL-means algorithm (custom command -nlmeans\_expr).

- Takes 3.156 seconds for a 512x512 RGB image, with 24 cores used.
- → Very convenient for quick algorithm prototyping.



< ロ > < 同 > < 三 > < 三 > < 三 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

- Today, G'MIC is a project with:
  - Approx. 150.000 lines of code (without the code of interfaces) (in C++ and G'MIC language mainly).
  - 700+ downloads / day (more than. 3.000.000 since August 2008).
  - 400+ unique visitors / day on the project web pages.
  - ⇒ Unexpected results considering the first targeted audience (researchers in image processing!).





- More users: G'MIC becomes referenced on forums, blogs, news articles (about computer graphics or free software): framasoft, linuxfr, webupd8, libregraphicsworld, pcastuces, gimpfr, linuxgraphics, gimpusers, ...).
- More contributors: Help from beta-testers, packagers (Debian, Ubuntu, Arch, Mageia, Gentoo, Windows, MacOSX,...), bug reports, language translations, design of mascots, new filters, ...



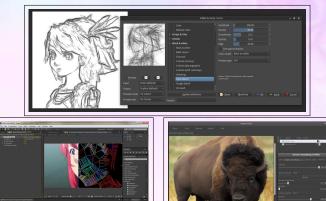


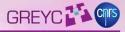
• G'MIC has a great community of users: Flickr (+800 followers), Pixls.us, GimpChat, Twitter, Google+ (+2200 followers), ...





• Recent development of some G'MIC -based interfaces by external developers: Plug-ins for Krita, After Effects, Natron, PhotoFlow.





Why G'MIC does raise interest ?

#### Demonstration of some G'MIC features

(for artistic purposes)



# Filter Showcase:

Rodilius



- Goal: Exaggerate the structure and length of the image contours to make them more visible.
- **Principle:** Several image convolution with oriented gaussian kernels are computed along different orientations of the plane. The resulting images are simply combined with layer blending modes Lighten only or Darken only. Finally, we smooth the blended image anisotropically, then sharpen its contours.
- Similar to: Filter banks for the geometric analysis of images (contourlets).

#### Artistic: Rodilius



◆□ ▶ ◆□ ▶ ◆三 ▶ ◆三 ▶ ● ● のへで



Open input image.

#### Artistic: Rodilius



-	G'MIC for GIMP 1.5.8.5 (beta) -	- Linux 64 bits	- = ×
Preview       •         Input / Output :       •         Input / Output :       •         Output mode       •         Output preview       •         Output preview       •         Output preview       •	Available filters (646) : Cubism Dream smoothing Ellipsionism Felt pen Graphic boost 2 Graphic novel Hope poster Kuwahara Lylejk's painting Make squiggly Morphology painting Pastell art Pen drawing Phoenix steam-pencil Photoillustration Polygonize Poster edges Psychedelic glasswork tik Rodillus Shapeism + © Internet ©	Channel(s) All Preview type Full Authors: Rod/ClimpChat and David Tschumperlé. Latest update: 2013/03/05.	7.30 : 19.03 : 491.15 : 16 : 30.00 : 2 : : :
		X Cancel	ply ✓ ОК

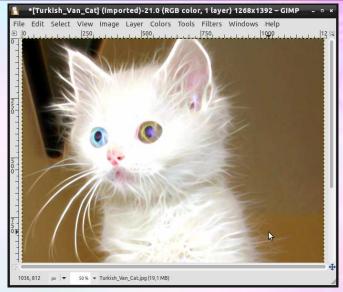
Invoke G'MIC plug-in and select Artistic / Rodilius.

◆□ ▶ ◆□ ▶ ◆三 ▶ ◆三 ▶ ● ● のへで

## Artistic: Rodilius



(ロ) (日) (日) (日) (日) (日) (日)



Wait a little bit, then enjoy ! (recently parallelized for speeding up FFTs).



(ロ) (日) (日) (日) (日) (日) (日)



Two other examples, works quite well on fur.

## Artistic: Rodilius





Another example: with Darken only blending mode used.



 Reproduces the 'Fractalius' effect (49\$ plug-in for Photoshop) but for 0\$ and 10 lines of G'MIC code !):



Redfield Fractalius



**G'MIC** Rodilius

(ロ) (日) (日) (日) (日) (日) (日)



```
1. rodilius : -check "${1=10}>=0 && $1<=200 && ${2=10}>=0 && $2<=100 && ${3=400}>=0 && ${4=7}>0" -skip ${5=0}, ${6=1}

2. -v - repeat $! -1[$>] -split_opacity -rv

3. -if {!$6} -negative. -endif

4. --f. 0 -nm. {-2,n}

5. -repeat [round($4)]

6. angle={$5+$>*180/round($4)}

7. --blur_linear.. $1%, {$1*$2/100}%, $angle,1 -b. 0.7 -sharpen. $3 -max[-2,-1]

8. -done -rm..

9. -if {!$6} -negative. -endif

10. -rv ac - endl -done -v +
```







Filter Showcase:

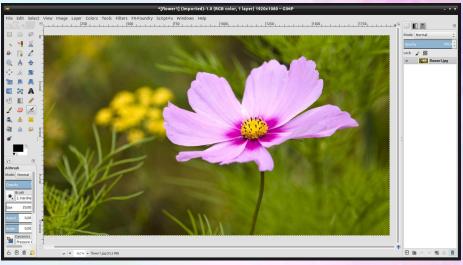
Color transfer



- Goal: Give a color ambiance to an image, from a reference image.
- **Principle:** We register two colorimetric functions in the RGB cube to determine a color correspondence map to apply to the input image.
- Similar to: Optical flow, image registration.



◆□▶ ◆□▶ ◆三▶ ◆三▶ 三 りへで



Open input image.

## Colors: Color transfer



<ロ> <=> <=> <=> <=> <=> <=> のへで



Open reference image (as a new layer).

#### Colors: Color transfer

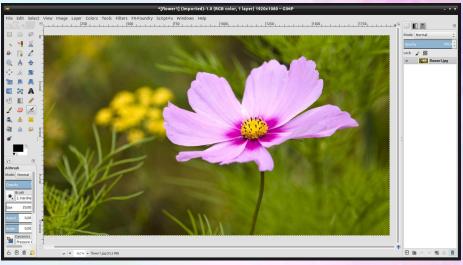


 G'MIC for GIMP 1.6.8 [pre-release #110215] - L	inux 64 bits	- • ×
C'Mic for GIMP 1.6.8 [pre-release #110215] - L Available filters (41) Curves [KBD] Curves [KBD] Curves [KBD] Curves [KBD] Curves [KBD] Curves [KBD] Curves [KBD] Curves [KBD] Hue lighten-darken Metallic look Mixer [CMYK] Mixer [Lab] Mixer [KBD] Mixer [KBD] Mixer [KBD] Mixer [CMYK] Mixer [CMS] Mixer [C	Transfer colors [advanced] Regularization	20 C
× Cancel	CFullscreen CReset Apply	ок

Invoke G'MIC plug-in and select Colors / Transfer color [advanced].



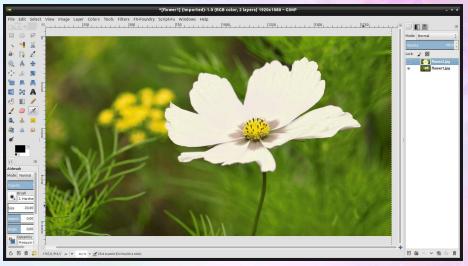
◆□▶ ◆□▶ ◆三▶ ◆三▶ 三 りへで



Original image.



◆□▶ ◆□▶ ◆三▶ ◆三▶ 三 りへで



Color-transferred result.

## Colors: Color transfer



<ロ> <=> <=> <=> <=> <=> <=> のへで

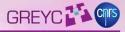


Reference image (reminder).





Other examples.



#### Filter Showcase:

#### Extract foreground [interactive]



< ロ > < 同 > < 三 > < 三 > < 三 > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > < ○ > <

- Goal: Extract foreground objects from background in an image, and get the result as two distinct (complementary) layers.
- **Principle:** Same as before, but done only with key points having labels "foreground" or "background" instead of colors.





Open input image (single-layer color photograph).



	Available filters (441)	Extract foreground [Interactive]						
	About	Feathering	0	0.00	-			
	<ul> <li>Arrays &amp; tiles</li> <li>Artistic</li> </ul>	Dilation	0	0	\$			
No preview	Black & white	Output mode	RGBA foreground + backgroun	nd (2 layers)	\$			
available	Colors	View resolution	Medium					
	* Contours	viewresotderon	medium		•			
	Convolve	Description:						
States and the second second	Curvature	This filter allows to quickly extract foreground objects from background in opaque RGB images. Click on the Apply or OK buttons below to open						
	Difference of gaussians	the interactive wind	ow and start adding foreground and t	background				
4	Distance cransform		you're done, exit the interactive wir d will be transfered back to GIMP.	ndow: your				
🗹 Preview 📃 🛨 C								
	Edges offsets		If you are not satisfied with the result, click on Apply once again to modify your control points defined previously. To remove all control points, click					
	Extract foreground [interactive]	on the Reset button	below.					
	Sradient norm	1						
	Gradient RGB	Interactions:	tions in the interactive window to bu	iild your				
	Isophotes	extraction mask :						
	Laplacian	- Left mouse butto	or key F create a new foreground co	ontrol point (or				
Input / Output	Local orientation	move an existing on	e).					
and the second sec	Morphological filter	<ul> <li>Right mouse butto (or move an existing</li> </ul>	on or key B create a new background one).	control point				
Input layers ‡	Segmentation		eys CTRL+arrows UP/DOWN zoom v s the extraction mask.	/iew in/out.				
Output mode ‡	Skeleton	- Key TAB toggles b	ackground view modes.					
	Thin edges	- Key M toggles mai	ker view modes. leletes the last control point added.					
Output messages ‡	<ul> <li>Deformations</li> </ul>	- Key PAGE UP incr	eases background opacity.					
Preview mode ‡	Degradations	- Key PAGE DOWN - Keys CTRL+D incr	decreases background opacity.					
Trenew modelin +	Details	- Keys CTRL+C decr	ease window size.					
Large ‡	<ul> <li>Film emulation</li> </ul>	- Keys CTRL+R rese - Keys FSC O or FN	t window size. TER exit the interactive window.					
	+ C 🗹 Internet 🕑	Author: David Tschu						

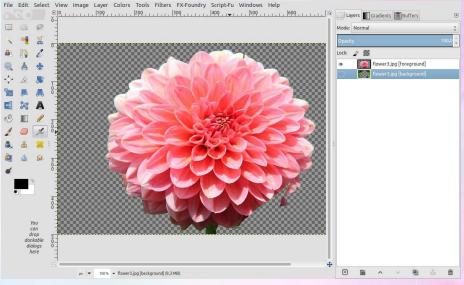
Invoke G'MIC plug-in and select Contours / Extract foreground [interactive].





Place some "foreground" and "background" key points.





Result of the filter: 2 layers (foreground shown here).

◆□▶ ◆□▶ ◆三 > ◆三 > ● ● ●





Result of the filter: Image after modification of the color hue on foreground layer only.





Another example of result, processing background and foreground independently.



Filter Showcase:

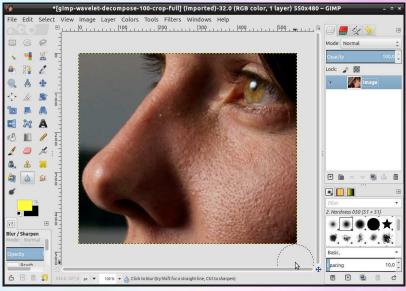
Split details



- Goal: Decompose an image into several levels of details, in order to work separately on the different image scales before recomposing the image.
- **Principle:** The image is decomposed/recomposed using a pyramidal representation obtained by the iterative convolution by gaussians kernels + residues.
- Similar to: Scale space analysis.



<ロ> < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ >



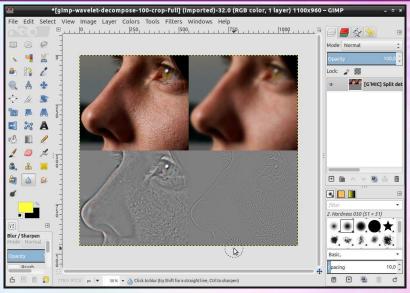
Open input image.



	G'MIC for GIMP 1.5.8.5 (beta) - Li	nux 64 bits		- • ×
1 2/4	Available filters (646) : Sharpen [richardson-lucy] Sharpen [shock filters]	Split details : Action	Decompose/update	:
	Sharpen [unsharp mask] Sharpen [unsharp mask] Sharpen [whiten] Simple local contrast	Decompose options: Number of scales	-0-3	•
1 AL	Split details Spotify	Base scale	5.00	÷
	Texture enhance p Tone mapping	Output mode	Tiling	\$
	Tone mapping [fast] : Yag effect	Preview type	All scales	:
Preview Input / Output : All Output mode Output messages Output preview Output preview	<ul> <li>Film emulation</li> <li>Frames</li> <li>Layers</li> <li>Lights &amp; shadows</li> <li>Patterns</li> <li>Rendering</li> <li>Repair</li> <li>Sequences</li> <li>Spectral filters</li> <li>+ C ginternet</li> </ul>	parameters. Then clic 2. Process your image scales using the neutral 3. Use CTR.4-Fo app resulting recomposed canvas. 4. Repeat steps 2. an 5. When you are happ again, set parameter 6. Cet your final mod Author: Jérome Bould Latest update: 2013/	e. For instance, remove details in the appropriat rat color (728.78,728). If the filter again (update mode). The preview of dimage updated and shown at the top left of yo d 3. sam any times as you want. py with the result, open the C/MIC plug-in once Action to Recompose, and click OK. Iffied image alone. anger and David Tschumperlé.	e f the ur
		× Cancel	set Maximize Apply	ок

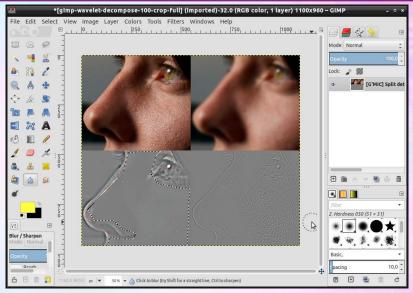
Invoke G'MIC plug-in and select Details / Split details.





You get your input (top-left) + the decomposition into scales (here 3 scales).





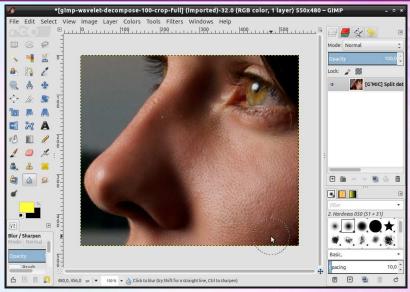
Do what you want on the scales (here, we simply erase the skin defects on the middle scale).



	MIC for GIMP 1.5.8.5 (beta) - Li	nux 64 bits			• ×
	Available filters (646) :	Split details :			
	Local normalization	Action	Recompose		\$
	Local variance normalizatic Make up	Decompose options:	₩		
ALC: NOT THE REAL PROPERTY OF	Mask creator	Number of scales	-0	3	÷
No preview available	Portrait retouching Sharpen [deblur]	Base scale		5.00	+
	Sharpen [gold-meinel]	Details scale	-0	1.74	\$
0	Sharpen [inverse diffusion] Sharpen [octave sharpening	Output mode	Tiling		\$
	Sharpen [richardson-lucy] Sharpen [shock filters]	Preview type	All scales		\$
Preview  Preview  All  Output rode  Output messages  Output preview  Cutput preview	Sharpen [unsharp mask] Sharpen [whiten] Simple local contrast Split details Spotify Texture enhance p Tone mapping Tone mapping [fast] Yageffect + C V Internet	parameters. Then clic 2. Process your image scales using the neutr 3. Use <i>CTRL+F</i> to appl resulting recomposed canvas. 4. Repeat steps 2. an 5. When you are happ again, set parameter 6. Get your final mod	a. For instance, remove details in the app al color (728.78,78). by the filter again (update mode). The pre- limage updated and shown at the top lef d 3. as many times as you want. by with the result, open the G'MIC plug- in Action to Recompose, and Click OK. (field image alone. Ingger and David Tschumperié. 12/14.	ropriate eview of th t of your	

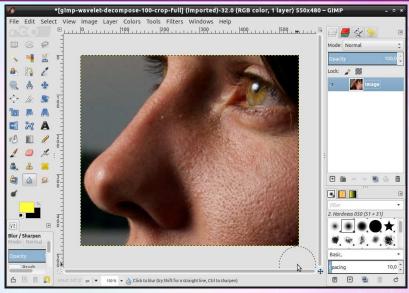
Invoke G'MIC plug-in again, to recompose the final image.





Result of the recomposition, with cleaner skin (5mn work !).





Comparison with initial image.



Filter Showcase:

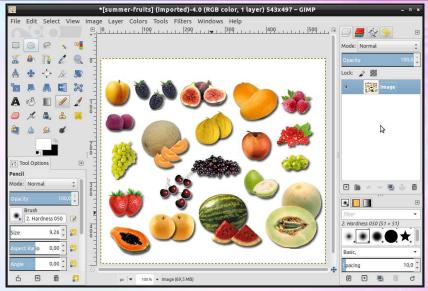
Extract objects



- Goal: Extract independent objects localized on a single image.
- **Principle:** Background detection is performed, then residual pixels are grouped as several connexe regions.
- Similar to: Object detection and segmentation on static background.

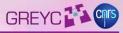
# Arrays & Tiles: Extract objects





Open input image (single-layer).

## Arrays & Tiles: Extract objects



	AIC for GIMP 1.5.8.5 (beta)	- Linux 64 bits – 🕫	×
Preview  Input Joutput :  Input layers  Output mode  Output messages  Output preview	Available filters (646) : * About * Arrays & tiles Array [faded] Array [random] Array [rendom colors] Array [regular] Chessboard Dices Extract objects Grid Puzzle Taquin Tileable rotation Tiled isolation Tiled normalization Tiled normalization Tiled random shifts Tiled rotation * Artistic * Black & white * C (mathematical contents)	Extract objects :       0.00         X-background       0.00         Color tolerance       20         Opacity threshold       50         Minimal area       Crop         Connectivity       Located crop         Output type       Located segmentation         Author: David Tschumperlé       Latest update: 2014/02/14.         Filter explained here	
		X Cancel Reset Maximize Apply OK	

Invoke G'MIC plug-in and select Arrays & tiles / Extract object.

# Arrays & Tiles: Extract objects

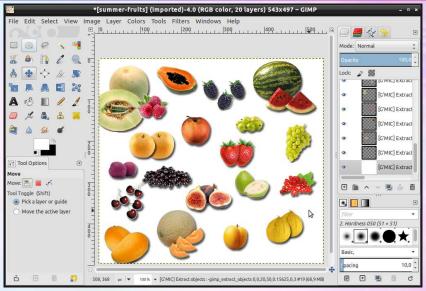




Output looks similar as input, but is divided into several layers.

## Arrays & Tiles: Extract objects





Managing each object independently is now possible (here, position change).

◆□ ▶ ◆□ ▶ ◆目 ▶ ◆目 ● ● ●



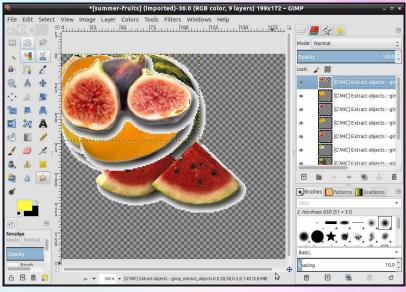
**Filter Showcase:** 

Pack sprites



- **Goal:** Generate a synthetic image where multiple image thumbnails have been packed together without overlapping (resized and rotated).
- **Principle:** Valid pseudo-random positions are iteratively checked for the insertion of new objects, with decreasing dimensions.
- Similar to: Bin-packing problem (NP-hard).





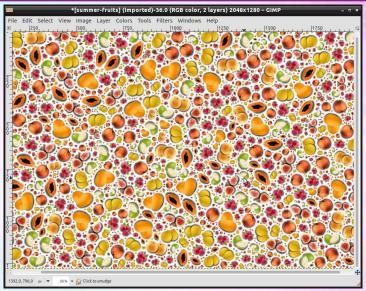
Select your objects to pack (multi-layer image).



-@	G'MIC for GIMP 1	1.5.8.5 (beta) - Linux	64 bits	- • ×		
ee No preview available ♥ Preview ♥ Preview ■ e 1 1 1 1 1 1 1 1 1 1 1 1 1	Available filters (646) : Lignes & snaoows Patterns Bayer filter Box fitting Canvas Canvas texture Clouds Cracks Halftone Halftone Halftone Halftone Halftone Halftone Marble Maze Mineral mosaic Mosaic Op art Pack sprites	Pack sprites : Number of scales Minimal scale [%] Allow angle Spacing Precision Masking Width Height Notes: - Parameters Width and is isselected.	Pack sprites :         Number of scales         Minimal scale [%]         Allow angle         Any         Spacing         Precision         Masking         No masking         Width         Height         • Farameters Width and Height are considered only when No masking         • Set different sprites on different layers to pack multiple sprites at the same time.			
Output messages : Output preview :	Pack sprites Paper texture Periodic dots Plaid					
Considered Strategy	+ C Internet •	×	Cancel Cancel Maximize Ap	ply У ОК		

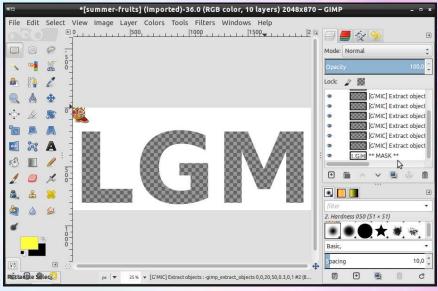
Invoke G'MIC plug-in and select Patterns / Pack sprites.





Get your image with randomly packed sprites (after a while).





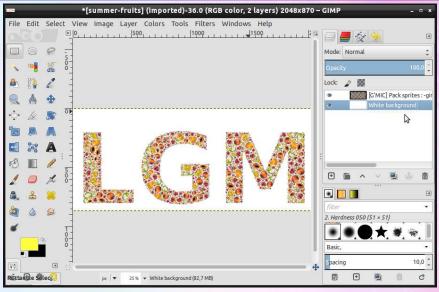
Now, you can add a bottom layer to restrict packing on transparent regions.



Invoke G'MIC again, and select Mask: Mask as bottom layer.

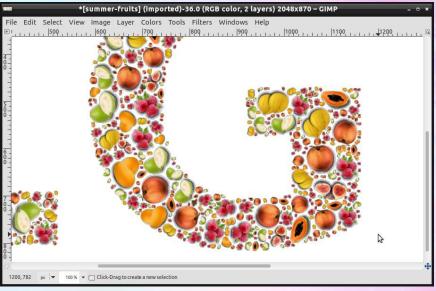


< ロ > < 同 > < 三 > < 三 > < 三 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □



Go for a coffee, and you get this.





Detail of the result.





Example of rendering, by Chris Fiedler, on GimpChat.



# Filter Showcase:

Shapeism



イロト ( 同 ) ( 三 ) ( 三 ) ( 三 ) の( ()

- Goal: Use the previous Sprite packing filter to create image abstraction (such as the Circlism from artist Ben Heine).
- **Principle:** Monochrome shapes are packed together at different scales, with constraints to put only small shapes on image contours. Shape colorization is performed afterwards by averaging the color pixels covered by each shape.





Open input image.



< ロ > < 母 > < 目 > < 目 > < 目 > < 目 > の へ の

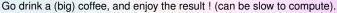
-	G'MIC for GIMP 1.5.8.5 (beta) - Li	nux 64 bits			- • ×
	Available filters (646) : Make squiggly Morphology painting	Shapelsm : Shape Circles			•
	Painting	Branches	-0	3	÷
	Pastell art Pen drawing	Thickness	0	1.00	2
	Phoenix steam-pencil	Angle	-0	29.14	\$
	Photoillustration Polygonize Posterized dithering		rs Branches, Thickness and Angle r Custom shapes.		
C. C		🗹 Antialiasi	ng		
	Poster edges     Psychedelic glasswork tile	Scales		8	
Preview		Maximal size		64	*
Input / Output :	Shapeism	Minimal size	0	6	\$
Input layers ‡	Simple noise canvas Warhol	Allow angle	0 deg.		\$
Output mode ‡	Watercolor Whirl drawing	Spacing		-1	-
Output messages 🗘	Wild cartoonizer	Precision		9	*
Output preview	Black & white	Edges		0.50	•
	+ C 🗹 Internet Đ	Smoothness	-0	1.00	\$
		× Cancel	Reset Maximize Apply		ок

Invoke G'MIC plug-in and select Artistic / Shapeism.



<ロ> < 同 > < 同 > < 三 > < 三 > < 三 > の Q ()

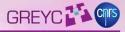








Result with another shape selected (a star).



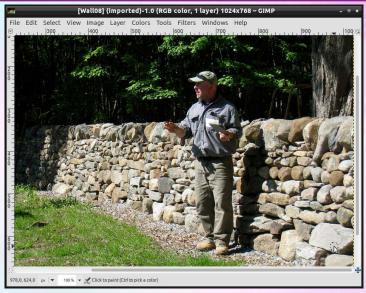
## Filter Showcase:

### Inpainting [patch-based]



- Goal: Allow the reconstruction of "holes" in images (groups of pixels considered as missing or invalid).
- **Principle:** Implementation of an extension to the inpairing algorithm of Criminisi-Perez-etal + patch blending technique.
- Similar to: Inpainting, "classical" (and hard-to-solve!) reconstruction problem in image processing.





Open input image.



イロト ( 同 ) ( 三 ) ( 三 ) ( 三 ) の( ()



Draw an inpainting mask directly on it (with a constant known color).

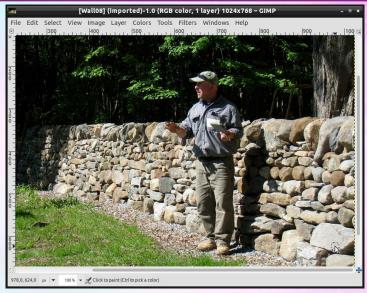


	MIC for GIMP 1.5.8.5 (beta) - L	inux 64 bits			- • ×	
	Available filters (646) :	Inpaint [patch-	-based] :			
No preview available	Banding denoise Bayer reconstruction Deinterlace Despeckle Iain's fast denoise p Iain's noise reduction Inpaint [diffusion] Inpaint [patch-based]	Patch size	0	17	÷	
				16.00	\$	
		Lookup factor	-0	0.10	:	
		Blend size		1.20	*	
		Blend threshold	0	0.00	2	
		Blend decay	-0	0.05	2	
Preview	Inpaint [solidify]	Blend scales		10	:	
Input / Output :	Local similarity mask Pixel denoise	Allow outer				
Input layers ‡	Recursive median Red-eve attenuation	Mask type	Mask by color		:	
Output mode ‡	Remove hot pixels	If Mask by color ha	s been selected :			
Output messages ‡	Repair scanned documen Smooth [anisotropic]	Mask color				
Output preview ‡	Cmaath [hilataral]	A quick tutorial on how to use this filter can be found here:				
	+ C 🗹 Internet 🕑	<u>G'MIC Inp</u>	ainting tutorial on Patrick David's	<u>blog.</u>		
		× Cancel	Reset Maximize Apply		ок	

Invoke G'MIC plug-in and select Repair / Inpaint [patch-based].



< ロ > < 母 > < 目 > < 目 > 目 ) 回 の へ で



Input image.





If you choose carefully the parameters, this is what you get.



◆□ ▶ ◆□ ▶ ◆三 ▶ ◆三 ▶ ● ● ● ●

















◆□ ▶ ◆□ ▶ ◆□ ▶ ◆□ ▶ ● ● ● ●



Example from Patrick David: Input image.



(□) <</li>
 (□) <</li>
 (□) <</li>
 (□) <</li>



Example from Patrick David: Inpainted image.



・ロシ・日本・モン・モン 田 うらの



Example from Patrick David: Input image.



・ロシ・日本・モン・モン 田 うらの



Example from Patrick David: Inpainted image.



・ロシ・日本・モン・モン 田 うへで



(Extreme case!): Input image (boat to be removed).





Result by the G'MIC inpainting algorithm.



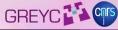
Filter Showcase:

Denoising filters

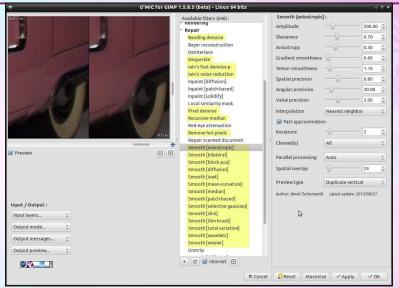


イロト ( 同 ) ( 三 ) ( 三 ) ( 三 ) の( ()

- Goal: Algorithms to smooth an image while preserving the image details and textures.
- **Principle:** Reccuring issue in image processing, with a lot of algorithms existing (PDE's, Wavelets, Patch-based smoothing, etc...).



(ロ) (目) (ヨ) (ヨ) (ヨ) (ロ)



Invoke G'MIC plug-in, and select one of the denoising filters

(more than 20 methods available).





Comparison between original / denoised image (equalized images for clarity).



• G'MIC is one of the few software to offer efficient image denoising algorithms:

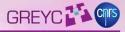




◆□ ▶ ◆□ ▶ ◆三 ▶ ◆三 ▶ ● ● ● ●

• G'MIC is one of the few software to offer efficient image denoising algorithms:





Filter Showcase:

Dream smoothing



イロト ( 同 ) ( 三 ) ( 三 ) ( 三 ) の( ()

- Goal: Apply one of the previous image smoothing technique, deliberatly exaggerated and make the colors more contrasted to create a painting effect.
- **Principle:** We apply multiple iterations of anisotropic smoothing with an "aggressive" color mix in the Lab color space.



イロト イロト イヨト イヨト ヨ のへで



#### Open input image.



◆□ ▶ ◆□ ▶ ◆□ ▶ ◆□ ▶ ● ● ● ●

	G'MIC for G	MP 1.5.8.5 (beta) - Linux 64 b	its		>
		Available filters (646) :	softer image as does lower opacity values. Practical modes for merging are Alpha and Average. Note that results are resolution dependent.		
			General settings Iterations Sequalize at each		3 *
			Merging of iteration Merging option Opacity Reverse order Settings for layer m Smoothness	alpha	0.80 ÷
Preview Input / Output : Input layers Output mode Output messages Output preview Output preview		Dream smoothing Ellipsionism Felt pen Graphic boost 2 Graphic novel Hope poster Kuwahara Lylejk's painting	Parallel processing Spatial overlap Preview type Author: Arto Huotari La	rtical bands appear.	24 ÷
		× Cancel	Reset	imize 🗸 Apply	✓ ОК

Invoke G'MIC plug-in and select Artistic / Dream Smoothing.



イロト ( 同 ) ( 三 ) ( 三 ) ( 三 ) の( ()



Enjoy your result ! (takes some time to render, recently parallelized).



(ロ) (日) (日) (日) (日) (日) (日)



How artists use it for real: Processing done by Zarir Madon.



◆□ ▶ ◆□ ▶ ◆□ ▶ ◆□ ▶ ● ● ● ●



How artists use it for real: Processing done by Arto Huotari.



**Filter Showcase:** 

Poisson editing



イロト ( 同 ) ( 三 ) ( 三 ) ( 三 ) の( ()

- Goal: Copy/paste a piece of image into another one, without visible seams in the result.
- **Principle:** Solving the Poisson equation to reconstruct the final image from the gradient map where the paste has been done.

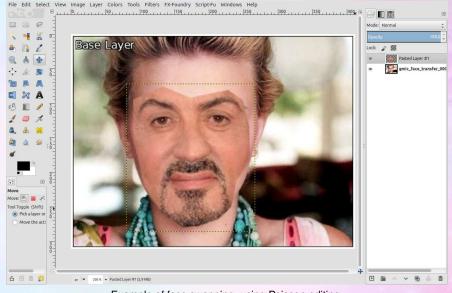


















< ロ > < 母 > < 目 > < 目 > 目 ) 回 の へ で



Example of object insertion, using Poisson editing.



< ロ > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 > < 日 = < 日 = < 日 = < 1 = < 1 = < 1 = < 1 = < 1 = < 1 = < 1 = < 1





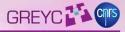


Example of face swapping, using Poisson editing (on the same input picture).





Example of face swapping, using Poisson editing (on the same input picture).



# Filter Showcase:

#### Sketch



- Goal: Algorithms to transform a picture into a sketch.
- **Principle:** Pencil strokes are iteratively simulated on a white canvas, by analyzing the contour geometry of the original picture.
- Similar to: Contour detection and extraction, texture analysis.





(Courtesy of Tom Keil)



(ロ) (日) (日) (日) (日) (日) (日)



(Courtesy of Tom Keil)



<ロ> < 母> < 国> < 国> < 国> < 国> < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 > < 国 >



(Courtesy of Tom Keil)





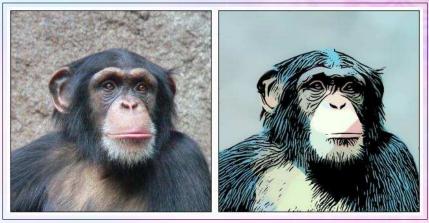




(Courtesy of Tom Keil)



◆□▶ ◆□▶ ◆三▶ ◆三▶ 三 りへで



(Example of another similar filter in G'MIC : Engrave)

 $\Rightarrow$  +440 filters like this available in the G'MIC plug-in for GIMP!



# **Filter Showcase:**

Film emulation



- Goal: Provide free film emulation filters, similar to what proprietary DXO FilmPack proposes.
- Made by: Patrick requested David to make his color profiles easily available for everyone.
- How is this done? Color transformations are encoded as RGB CLUT files, stored on the G'MIC server. Each color profile is downloaded on demand.
- 476 lines of G'MIC code (mostly for GUI). (all included: GUI description + algorithm).





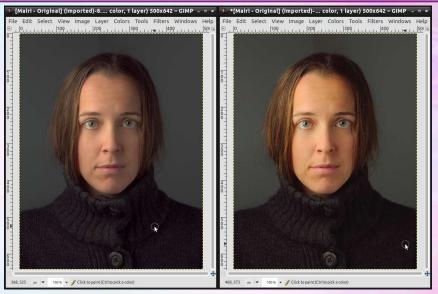
#### Open input image.



Invoke G'MIC plug-in, and choose one filter in folder Film emulation/.

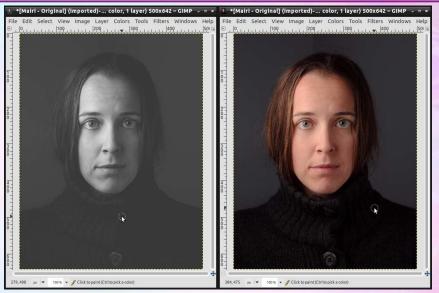


・ロト・日・・ヨ・・ヨ・ ヨ のへで



Comparison: Before (left) / After (right).





Two other examples: TMAX-3200 (left) and Kodak Kodachrome 64 (right).

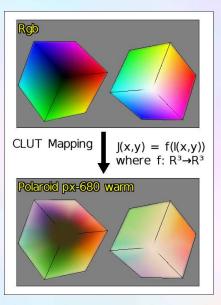


(ロ) (日) (日) (日) (日) (日) (日)



Patrick David has indeed done a lot of presets (here, a sample of them).





# Technically speaking:

- Each preset defines a mapping function from RGB to RGB (CLUT).
- The values of these functions are explicitly stored for all RGB colors.
- To avoid huge datasets, we consider 64x64x64 downsampled versions of the CLUTs and interpolate intermediate colors.
- → 77Mb of data for 271 film emulation presets.
- As the original color mappings are smooth functions, interpolation has almost no incidence on the quality.



イロト ( 同 ) ( 三 ) ( 三 ) ( 三 ) の( ()

# G'MIC [http://gmic.eu]

- A full-featured open-source framework for image processing: Several user interfaces available, more to come.
- Positive for the GREYC: Evolving software.
   "Image Processing" showcase for the general public.
- Positive for the IMAGE team: Useful software on a daily basis, for the analysis and exploration of image data, and the fast prototyping of new algorithms + derived publications.
- C Time consuming: developing/maintenance, community animation, web pages, answering questions... (approx. 10-15h of work / week).