GREYC's Magic Image Converter (G'MIC)

Cette page propose de présenter quelques fonctions imagées de G'MIC utilisé en ligne de commandes.
Toutes les commandes de ce document sont testées et le résultat affiché.
Les commandes sont identiques pour les différents systèmes d'exploitation.
Les filtres de G'MIC pour Gimp sont aussi accessibles en lignes de commandes. Ils sont présentés à la fin de cette page au chapitre : Test des "greffons de Gimp écrits en G'MIC" via l'Invite de commandes
Vous pouvez télécharger la version 53 de cette page au format PDF à partir de ce lien : http://www.aljacom.com/~gmic/commandes_gmic.pdf

Sommaire

polaroid	drop_shadow	tetris	mosaic	puzzle	sponge	hearts	color_ellipses	ellipsionism	whirls	cartoon	drawing
draw_whirl	stencil	stencilbw	pencilbw	sketchbw	ditheredbw	dotsbw	warhol	cubism	glow	old_photo	rodilius
texturize_paper	texturize_canvas	ripple	fire_edges	kuwahara							

Visualiser les effets des paramètres des effets artistiques en video
- Vidéo effet mosaic
- Vidéo effet puzzle
- Vidéo effet tetris

Déformation spatiale

euclidean2polar	polar2euclidean	warp_perspective	water	wave	twirl	map_sphere	flower	zoom	deform	fisheye	transform_polar
kaleidoscope	rotoidoscope										

Contours

gradient_norm	gradient_orientation	gradient2rgb	laplacian	divergence	Inn	Iee	curvature	edges	isophotes	topographic_map	segment_watershed

Manipulations géométriques

split_tiles	append_tiles	rr2d / resize_ratio2d	r2dx / resize2d x	r3dx / resize3d x	r2dy / resize2dy	r3dy / resize3d y	r3dz / resize3d z	upscale_smart	expand_x	expand_y	expand_z
expand_xy	expand_xyz	shrink_x	shrink_y	shrink_z	shrink_xy	elevate					

Entrées/Sorties

input	output	verbose	print	echo	warning	command	type	shell	shared	camera	display
display3d	plot	window	wait	select							
outputw	outputp	outputn	display0	display_fft	display_rgba	display_histogram	display_tensors	float2int8	int82float	float2fft8	fft82float
apply_camera	rainbow_lut	display_graph									

Mélanges d'images (fonctions "compose")

rgba	channels	average	multiply	screen	darken	lighten	difference	negation	exclusion	overlay	hardlight
softlight	dodge	colorburn	reflect	freeze	stamp	interpolation	xor	edges	fade	shapeaverage	compose_median
compose_divide											

Quelques fonctions 3D
- Rendre un objet 3D sur une image
- elevation3d
- extrude3d
  - Visualiser les effets 3D en video
    - Vidéo effet cube
- imagesphere3d
- spherical3d
- superformula3d
- pointcloud3d

Quelques fonctions importantes
- Récupérer les dimensions d'une image
- Ajouter un canal alpha
- Supprimer un canal alpha
- Créer du bruit

Présentation du programme G'MIC

G'MIC (http://gmic.sourceforge.net/) est un programme d'images fonctionnant sous plusieurs systèmes d'exploitation.
Il peut être utilisé en lignes de commandes ou au travers d'une interface utilisateur (GUI) via un greffon de Gimp.
Il dispose de très nombreux effets qui seront représentés plus loin dans cette page.
Les téléchargements sont disponibles aux pages http://sourceforge.net/projects/gmic/files/ (exécutables et sources) et http://gmic.sourceforge.net/gimp.shtml (greffon de Gimp).
Ce programme est construit autour de la bibliothèque CImg (http://cimg.sourceforge.net).
Le chef de projet est David Tschumperlé (Ronounours).
La licence d'utilisation est disponible à : en français http://www.cecill.info/licences/Licence_CeCILL_V2-fr.html en anglais http://www.cecill.info/licences/Licence_CeCILL_V2-en.html.
Un groupe de discussion, en anglais, est ouvert sur http://www.flickr.com/groups/gmic/.

Version 64 bits Windows

Depuis janvier 2012, nous proposons une version 64 bits pour Windows dans une archive zip qui contient le greffon pour Gimp ainsi que la version autonome de G'MIC (gmic.exe).
Cette version 64 bits pour Windows est beaucoup plus rapide que la version 32 bits.
Les liens pour la télécharger sont accessibles depuis notre blog http://samjcreations.blogspot.com à partir du libellé "G'MIC pour Gimp Windows" en haut à droite de la page.
Cette version est testée sérieusement, sur plusieurs versions de Gimp pour le greffon, avec un script de test pour G'MIC (gmic.exe).

Elle comporte aussi un démonstrateur qu'il faur appeler via un des fichiers `mega_demo.bat` ou `mega_demo - SILENT.bat` . Voici le contenu de `mega_demo.bat` :

```
start gmic -m mega_demo.txt -fonction_mega_demo
exit
```

Le fichier mega_demo.txt contient la fonction qui affichera les résultats de l'action des différents filtres.

Sous Linux il est possible d'utiliser ce démo, il suffit de télécharger
- http://www.aljacom.com/~gmic/mega_demo.txt
- http://www.aljacom.com/~gmic/geo.png
La ligne de commande pour appeler ce démo est : `gmic -m mega_demo.txt -fonction_mega_demo`
Ce démo fonctionne en 32 bits ou en 64 bits.

Quelques caractéristiques de cette version :
- Tout est placé dans un unique répertoire, est fonctionnel , est testé.
- Échanges via internet actif (test possible pour télécharger les mascottes).
- Fonctionnement plus rapide avec le mode silencieux (gmic-s).
- Quelques formats d'images testés

En sortie (output) :

```
JPEG : .jpg    (gmic geo.png -o test.jpg)
BMP  : .bmp    (gmic geo.png -o test.bmp)
PNG  : .png    (gmic geo.png -o test.png)
CIMGZ: .cimgz  (gmic geo.png -o test.cimgz)  Format CIMG compressé
CIMG : .cimg   (gmic geo.png -o test.cimg)   Format CIMG non compressé
~~TIFF~~ : G'MIC ne peut pas écrire des fichiers .tiff ou .tif d'une façon correcte (à vérifier).
```

En entrée (input) :

```
JPEG : .jpg    (gmic test.jpg)
BMP  : .bmp    (gmic test.bmp)
PNG  : .png    (gmic test.png)
CIMGZ: .cimgz  (gmic test.cimgz)  Format CIMG compressé
CIMG : .cimg   (gmic test.cimg)   Format CIMG non compressé
TIFF : .tif , .tiff (gmic mire.tiff) Avec libtiff, il lit les TIFF non compressés, compressés LZW , pack bits , déflation , CMJN (CMYK). Il ne peut pas lire les fax.
```

Accès à partir de Gimp

Sous Windows, après installation du paquet gmic_gimp_win32.zip, ce programme est disponible sur l'image par `Filtres > G'MIC` (copie d'écran).
Voici une structure possible après installation. Dans ce cas, le greffon sera disponible pour tous les utilisateurs :

```
Répertoire d'installation de Gimp
    └──lib
     └──gimp
      └──2.0
       └──plug-ins
        │ gmic_gimp.exe   (greffon de Gimp)
        │ libfftw3-3.dll   (bibliothèque transformations de Fourier)
        │ libpng3.dll         (bibliothèque images png)
        │ pthreadGC2.dll   (bibliothèque processus parallèles)
        │ libgcc_s_dw2-1.dll  (GCC runtime library)
        │
        │ (autres greffons de Gimp)
        │
        └──_gmic
          curl.exe    (transferts multi-protocoles)
```

Il existe aussi un installateur sous Windows : gmic_gimp_win32.exe (non testé, code source non disponible). Il semble qu'il installe le greffon dans le répertoire de l'utilisateur en cours, ce qui est un problème pour les ordinateurs multi-utilisateurs.
Pour les autres systèmes d'exploitation (Linux 32 et 64 bits, Snow Leopard), des paquets sont disponibles sur http://sourceforge.net/projects/gmic/files/.

David Tschumperle a créé un script-fu de démonstration "gmic_in_script.scm". Après installation il est accessible sur une image par : `Filtres > G'MIC Script test`.

Pour les programmeurs, voici les procédures pour appeler le greffon G'MIC de Gimp :

```
(register-procedure "plug-in-gmic"
  "G'MIC"
  "G'MIC"
  "David Tschumperlé"
  "David Tschumperlé"
  "2008"
  "GIMP Plug-In"
  (
    (
      "run-mode"
      "GIMP_PDB_INT32"
      "Interactive, non-interactive"
    )
    (
      "image"
      "GIMP_PDB_IMAGE"
      "Input image"
    )
    (
      "drawable"
      "GIMP_PDB_DRAWABLE"
      "Input drawable (unused)"
    )
    (
```

```
              "input"
              "GIMP_PDB_INT32"
              "Input layers mode, when non-interactive(0=none, 1=active, 2=all, 3=active & below, 4=active & above, 5=all visibles, 6=all invisibles, 7=all visibles (decr.), 8=all invisibles (decr.), 9=all
        (decr.))"
            )
            (
              "command"
              "GIMP_PDB_STRING"
              "G'MIC command string, when non-interactive"
            )
          )
          (
          )
        )
-gimp sponge 12,2
```

Test des "greffons de Gimp écrits en G'MIC" via l'Invite de commandes (ou une ligne de commande) sans lancer Gimp ⇨

Lien pour appeler une fonction "greffon de Gimp écrit en G'MIC" via le "shell" ⇨

Le greffon va créer différents fichiers (avec gmic dans leur nom) à ces emplacements : `C:\Users\[utilisateur]\AppData\Roaming` & `C:\Users\[utilisateur]\AppData` (sous W7).
Exemple des fichiers créés :

```
└──C:\Users\[utilisateur]\AppData\Roaming
          │
          │ gentlemanbeggar_gmic.gmic
          │ gmic_def.1509
          │ gmic_sources.cimgz
          │ iain_fergusson.gmic
          │ karos.gmic
          │ naggobot.gmic
          │ photocomix.gmic
          │ ronounours.gmic
          │ tomkeil.gmic
```

Liens vers deux pages de script-fu de démonstration utilisant G'MIC : script-fu-acid_patterns.html et Script-Fu_36_motifs_avec_GMIC.html

---

Utilisation de G'MIC en ligne de commandes

Les essais sont faits avec les versions 1.4.4.2 à 1.4.7.0 puis 1.5.0 9 64bits dans un environnement Windows 7 64bits.

Installation de la version 32 bits

Télécharger le paquet "gmic_[N° de version]_win32.zip" à partir de http://sourceforge.net/projects/gmic/files/ et décompresser cette archive.
Voici la structure :

```
└──gmic-[N° de version]_win32
    │ gmic.exe            (programme principal)
    │ gmic_gimp.exe       (greffon de Gimp)
    │ jpeg62.dll          (bibliothèque images jpeg)
    │ libfftw3-3.dll      (bibliothèque transformations de Fourier)
    │ libpng3.dll         (bibliothèque images png)
    │ pthreadGC2.dll      (bibliothèque processus parallèles)
    │ zlib1.dll           (bibliothèque compression de données)
    │         │ libgcc_s_dw2-1.dll  (GCC runtime library)
    │
    └──_gmic
       curl.exe      (transferts multi-protocoles)
```

Installation de la version 64 bits

Les liens pour la télécharger sont accessibles depuis notre blog http://samjcreations.blogspot.com à partir du libellé "G'MIC pour Gimp Windows" en haut à droite de la page.
Télécharger le paquet "gmic_[N° de version]_full_64bits_win.zip" puis décompresser cette archive.
Voici la structure principale (des fichiers de tests .bat peuvent être ajoutés dans le sous-répertoire gmic_standalone) :

```
+---gmic-[N° de version]_full_64bits_win
       ¦   lisez-moi.html
       ¦
       +---gmic_gimp_plugin
       ¦   ¦   gmic_gimp.exe                      (greffon de Gimp)
       ¦   ¦   gmic_in_script.scm                 (script-fu de Gimp)
       ¦   ¦   libfftw3-3.dll                     (bibliothèque transformations de Fourier)
       ¦   ¦   libgcc_s_sjlj-1.dll                (GCC runtime library)
       ¦   ¦   libpng15-15.dll                    (bibliothèque images png)
       ¦   ¦   libstdc++-6.dll                    (GNU Standard C++ Library)
       ¦   ¦   pthreadGC2.dll                     (bibliothèque processus parallèles)
       ¦   ¦
       ¦   +---_gmic
```

```
¦           curl.exe                          (transferts multi-protocoles)
¦
+---gmic_standalone
¦       gmic.exe                              (programme principal mode bavard)
¦       gmic-s.exe                            (programme principal mode silencieux)
¦       curl.exe                              (transferts multi-protocoles)
¦       mega_demo.bat                         (programme de démonstration)
¦       mega_demo.txt                         (fonctions de démonstration)
¦       libfftw3-3.dll                        (bibliothèque transformations de Fourier)
¦       libgcc_s_sjlj-1.dll                   (GCC runtime library)
¦       libjpeg-8.dll                         (bibliothèque images jpeg)
¦       libpng15-15.dll                       (bibliothèque images png)
¦       libtiff-3.dll                         (bibliothèque images tiff)
¦       libstdc++-6.dlll                      (GNU Standard C++ Library)
¦       zlib1.dll                             (bibliothèque compression de données)
¦       Start Terminal With Test G'MIC.bat    (Démarrage du terminal)
¦       mega_demo.txt                         (Fonctions du programme de démonstration)
¦       mega_demo.bat                         (Démarrage de la démonstration G'MIC en mode bavard)
¦       mega_demo - SILENT.bat                (Démarrage de la démonstration G'MIC en silencieux)
¦       geo.png                               (image de test 128*128 en couleurs)
¦       mire.tiff                             (image de test 1024*1024 en niveaux de gris)
¦       + différents autres fichiers de démonstration selon les versions de G'MIC
¦
+---sources
         gmic_[N° de version].tar.gz          (fichier ayant servi à la compilation)
```

Obtenir l'aide

Ouvrir "Invite de commandes", au choix, par :

- Menu Démarrer > Tous les programmes > Accessoires > Invite de commandes.
- Taper `cmd` dans la fenêtre de recherche du menu Démarrer.

Lorsque l'invite de commandes taper (architecture d'une version 64 bits) :

- `cd C:\[répertoire de décompression]\gmic-[N° de version]_full_64bits_win\gmic_standalone`
- Touche Entrée.
- `gmic --help > aide_gmic.txt`
- Touche Entrée.

Le fichier d'aide "aide_gmic.txt" est créé dans le répertoire de l'exécutable, un exemple est aussi consultable à partir de ce lien : aide_gmic.txt.

Remarques importantes

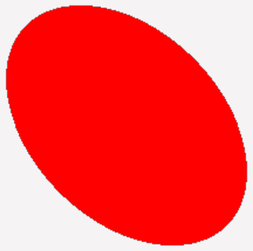
Avec l'Invite de commandes Windows les noms de répertoires ou les noms de fichiers placés dans la ligne de commande qui contiennent des espaces doivent obéir à cette règle :
Le caractère \ doit être placé avant l'espace. Exemple pour ouvrir le fichier 'ma belle image.png' avec G'MIC : `gmic ma\ belle\ image.png`
Les caractères accentués sont acceptés :
`gmic C:\été2010\3août2010_1.png`
Une ligne de commandes peut contenir plusieurs instructions qui seront exécutées successivement.
Des opérations, des tests, des boucles peuvent se faire à l'intérieur de la ligne de commande, par exemple : `-erode {2*3}` = `-erode 6`
Pour en savoir plus sur les opérations, les commandes, etc. il faut consulter l'aide : aide_gmic.txt.
Les images PNG et JPEG sont bien gérées. Par défaut la qualité JPEG est de 100%. Pour d'autres types d'images (BMP, etc), il est préférable d'utiliser un convertisseur comme Gimp, ImageMagick, FreeImage utilisé par Maringouin, etc.
Images 8 bits/canal hors normes (<0 et >255) :
La fonction d'interpolation "resize" est modifiée et il faut prendre certaines précautions avec l'interpolation "5=bicubic" , voir remarque.
Certaines fonctions renvoient des images hors normes qu'il faut traiter avec `-cut 0,255` (`-c 0,255`), voir le message Jeu Nov 04, 2010 7:04 pm.
Voici un exemple de commande pour tester les valeurs hors normes et obtenir un bon résultat sur n'importe quelle image avec la fonction "resize " :
`gmic [image(s) à traiter] -resize 180,180,1,3,5 "-if {im<0} -c 0,255 -elif {iM>255} -c 0,255 -endif" -o [image(s) traitée(s)]`
Remarque de David Tschumperlé : Certains résultats de filtre mériteraient d'être coupés entre [0,255] avant de sauver le résultat. C'est le cas par exemple pour '-richardson_lucy', '-oriented_richardson_lucy', '-unsharp', '-unsharp_octave', '-normalize_local', '-gradient_orientation', '-laplacian', '-divergence', '-Inn', 'Iee', '-curvature', '-display0' et la détection de bords '-deriche'.

Images 8 bits/canal à normaliser :
Remarque de David Tschumperlé : Certains résultats de filtre mériteraient d'être normalisés dans [0,255] avant de sauver le résultat. C'est le cas par exemple pour '-convolve', '-correlate', '-structuretensors', '-hessian','-haar','-dog','-convolve_fft', '-bandpass', '-split_frequency', '-plasma'.

Le programme est incapable de créer un répertoire, il faut créer le répertoire avant d'utiliser `gmic.exe`.
Exemple d'une image `ellipse.png` (256*256 32bits) affichée et créée dans `...\gmic-1.4.7.0_win32\essais` (à chaque fin de ligne, valider par la touche Entrée) :

- `cd C:\[répertoire de décompression]\gmic-[N° de version]_full_64bits_win\gmic_standalone`
- `md essais`
- `gmic 256,256,1,4 -ellipse 50%,50%,120,80,45,1,255,0,0 -display -output essais\ellipse.png`

Résultat :



Il est possible de créer ses propres fonctions en utilisant un langage de programmation décrit dans l'aide. Le(s) programme(s) créé(s) sera/seront sauvegardé(s) dans un fichier texte qui sera utilisé par l'exécutable gmic.
Un fichier de commandes par défaut est déjà fourni dans le package G'MIC. Il est situé à «http://gmic.sourceforge.net/gmic_def.xxxx», où «xxxx» doit être remplacé par les 4 chiffres de la version actuelle de G'MIC.
C'est un bon début pour apprendre à créer ses commandes personnalisées. Toutes les commandes contenues dans le fichier de commandes gmic_def.xxxx sont inclues par défaut dans la version xxxx de G'MIC.

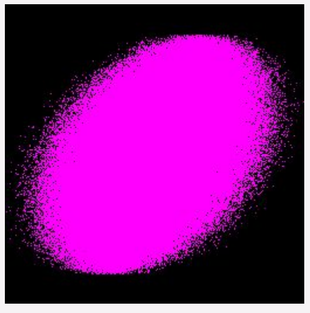
Exemple de contenu d'un programme sauvegardé dans le fichier `...gmic-[N° de version]_full_64bits_win\gmic_standalone\test1.txt` (lien téléchargement : test1.txt)

```
#@gmic fonction_test1 : : Tests programmation gmic
fonction_test1 :

# création image 256*256 pixels sans canal alpha
256,256,1,3

# création ellipse
-ellipse 50%,50%,120,80,-45,1,255,0,255

#ajout effet "spread"
-spread 10,0,0

#afficher le résultat
-display

#sauvegarder l'image en test1.jpg qualité 80% dans le répertoire de G'MIC
-output test1.jpg,80
```

La ligne de commande pour activer la fonction "fonction_test1" du programme "test1.txt" est :

```
gmic -m test1.txt -fonction_test1
```

Résultat :



G'MIC utilise FFMPEG pour obtenir des séquences vidéo. Le site http://ffmpeg.zeranoe.com/builds/ propose des versions compilées pour Windows.
Pour installer FFMPEG sous Windows, il suffet de télécharger l'archive 7Zip (64bits ou 32 bits / static ou shared), la décompresser et placer les fichiers `.exe` , `.dll` dans le répertoire de `gmic.exe`.
Suite aux essais avec la version 1.4.4.2 , nous n'avons pas su et pas pu obtenir automatiquement une vidéo sous Windows ou sous Ubuntu en utilisant une commande de ce type : `gmic geo.png -animate tetris,"10","20",11,0,animate.avi,40`, la vidéo est toujours découpée en petits morceaux.
Par contre, en utilisant la commande `gmic 1.png 2.png -morph 5,0.2,0.1 -o morph.mpeg` nous avons pu obtenir une vidéo mpeg lisible sur de nombreux lecteurs avec ces paramètres : `Output images [0,..,199] as file 'morph.avi', with 25 fps and bitrate 2048k`. Les autres formats (flv, ogg, mov) peuvent poser des problèmes de lecture, le format avi est lisible sur VLC.
Pour obtenir de plus nombreux paramètrages sur les vidéos il est préférable de créer des séquences d'images (exemple : `gmic geo.png -animate tetris,"10","20",11,0,animate.png,40`) et de les traiter dans FFMPEG, Virtualdub, Gimp avec GAP, Avidemux, etc. Pour convertir les vidéos au format Ogg Theora nous utilisons l'utilitaire ffmpeg2theora-0.28.exe.

Installation sous Linux

La remarque qui suit est ancienne, vérifier selon votre distribution ou télécharger les exécutables proposés sur http://sourceforge.net/projects/gmic/files/
Sous Ubuntu 10.10 32 bits : L'installation de la version 1.4.4.2 est compliquée, certaines bibliothèques ne sont pas dans les dépôts officiels.
Par Synaptic on peut installer gmic 1.3.5.7 puis vérifier à partir du terminal en visualisant une démo : `gmic -x_spline`

Effets des filtres de G'MIC via ligne de commande

Images utilisées pour les tests

L'image des deux perroquets réalisée par Kodak (kodim23.png) provient du site http://r0k.us/graphics/kodak/.

Cette image est réduite en dimensions avec G'MIC par cette ligne de commandes :

```
gmic kodim23.png -resize 30%,30% -output perroquets.png
```

Résultat :

C'est cette image réduite qui sera principalement utilisée lors des tests.

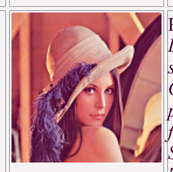
Pour les tests sur le bruit, l'image est une portion de l'image nikon-d3100-12800iso-nrstan-big.jpg de la page http://www.focus-numerique.com/test-1129/reflex-nikon-d3100-bruit-electronique-12.html
Cette portion d'image est obtenue avec cette ligne de commandes :

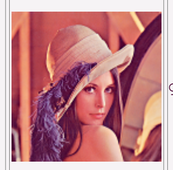
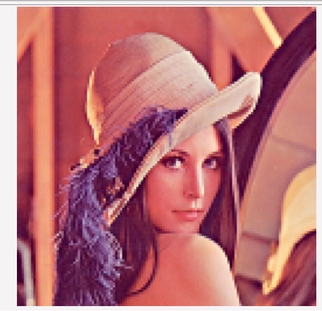
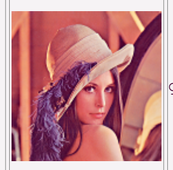
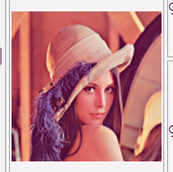
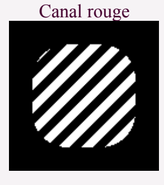
```
gmic nikon-d3100-12800iso-nrstan-big.jpg -crop 1056,816,1286,970 -o bruit.png
```

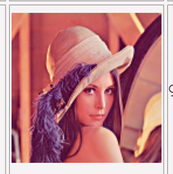
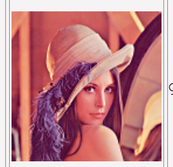
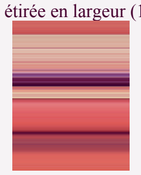
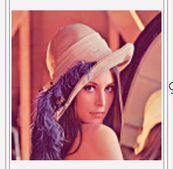
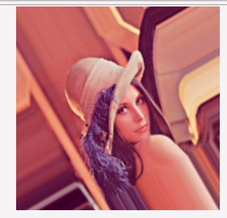
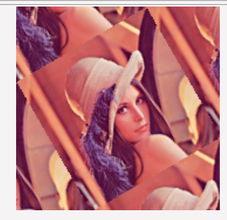
Résultat :

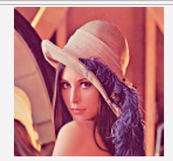
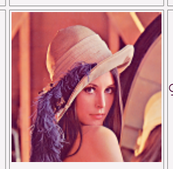
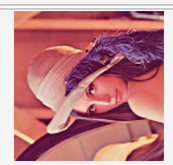
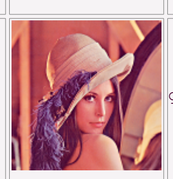
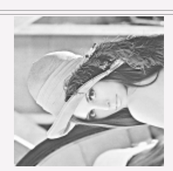
L'image réduite de Lena sera utilisée pour les corrections géométriques.

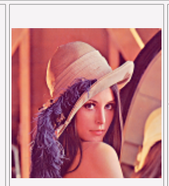
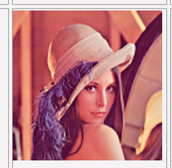
Corrections géométriques

Paramètres de la ligne de commande (aide de G'MIC)	Image(s) d'origine	Ligne de commande	Résultat
-resize [image], _interpolation, _borders, _cx, _cy, _cz, _cc | {[image_w] | width>0[%]}, _{[image_h] | height>0[%]}, _{[image_d] | depth>0[%]}, _{[image_s] | spectrum>0[%]}, _interpolation, _borders, _cx, _cy, _cz, _cc | (noargs) Resize selected images with specified geometry. (eq. to '-r'). 'interpolation' can be { -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }. 'borders' can be { -1=none | 0=dirichlet | 1=neumann | 2=cyclic }. 'cx,cy,cz,cc' set the centering mode when 'interpolation=0' (must be in [0,1]).		Remarque de David David Tschumperlé sur la fonction "resize" : *Lorsque l'on utilise la fonction d'interpolation bicubique (valeur de 5), les valeurs de l'image redimensionnée ne restent pas forcément dans l'intervalle d'origine (en général [0,255]), et donc lorsque l'on sauve un jpg, on obtient des points abbérants sur les pixels qui "débordent".* *Ce n'est pas un bug : par définition mathématique, l'interpolation bicubique ne préserve pas la plage des valeurs d'origine, contrairement par exemple à l'interpolation linéaire ou à l'interpolation au plus proche voisin. G'MIC considère que les pixels des images sont à valeurs flottantes et n'a donc aucune raison de 'couper' explicitement les valeurs de l'image redimensionnée entre [0,255] (c'est ce qui était fait avant, mais c'était une erreur !).* *Si on veut sauver une image flottante en s'assurant que les valeurs des pixels restent entre [0,255] (typiquement, si on veut sauver en JPG, PNG ou tout autre format 8bits / canal), il faut dire à G'MIC de 'couper' explicitement les valeurs entre [0,255], avec la commande '-cut' :* `gmic image.jpg -resize 30%,30%,1,3,5 -cut 0,255 -o resized.jpg` *Il n'y a pas de raison de privilégier un comportement de "coupure" par défaut lors d'une interpolation, car il y a d'autres situations où l'on ne souhaite pas ce comportement (typiquement quand on traite des images d'entrées à valeurs flottantes pas définies entre [0,255], ce qui arrive fréquement avec d'autres types de modalité que les photos couleurs classiques : imagerie médicale, etc..).* Voici un exemple de commande pour tester les valeurs hors normes et obtenit un bon résultat sur n'importe quelle image : `gmic [image(s) à traiter] -resize 180,180,1,3,5 "-if {im<0} -c 0,255 -elif {iM>255} -c 0,255 -endif" -o [image(s) traitée(s)]`	

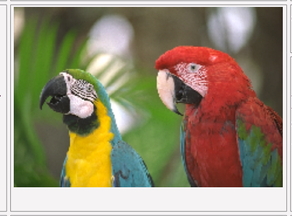
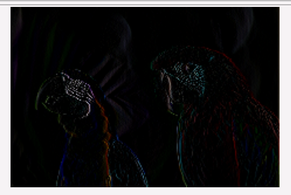
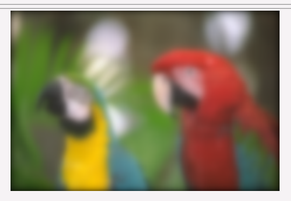
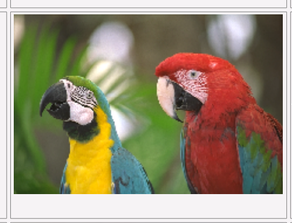
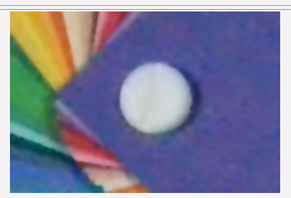
Their default values are '0'. (noargs) runs interactive mode (uses the instant window [0] if opened).		`gmic geo.png -resize 200%,50% -o resize.png`	
		`gmic geo.png -resize 180,180,1,3,5 -c 0,255 -o resize2.png`	
-resize2x Resize selected images using the Scale2x algorithm.		`gmic geo.png -resize2x -o resize2x.png`	
-resize3x Resize selected images using the Scale3x algorithm.		`gmic geo.png -resize3x -o resize3x.png`	Résultat identique à -resize2x ! 
-crop x0[%],x1[%], _borders | x0[%],y0[%],x1[%],y1[%], _borders | x0[%],y0[%],z0[%],x1[%],y1[%],z1[%], _borders | x0[%],y0[%],z0[%],c0[%],x1[%],y1[%],z1[%],c1[%], _borders | (noargs) Crop selected images with specified region coordinates. 'borders' can be { 0=dirichlet | 1=neumann }. (noargs) runs interactive mode (uses the instant window [0] if opened).		`gmic geo.png -crop 15%,5%,70%,40% -o crop.png`	
		`gmic geo.png -crop 20%,20%,80%,80% -o crop2.png`	
-autocrop value1,value2,.. Autocrop selected images by specified vector-valued intensity.		`gmic geo2.png -autocrop 0,255,255 -o autocrop.png` (Pour un contour transparent utiliser : `-autocrop 0`)	
-channels { [image0] | c0[%] }, _{ [image1] | c1[%] } Select specified channels of selected images.		`gmic geo2.png -channels 0 -o channels_R.png`	Canal rouge 

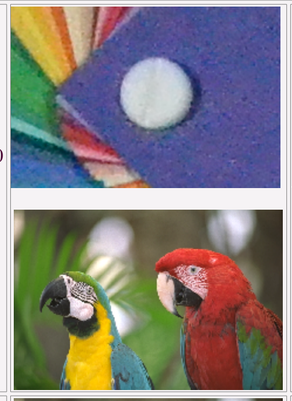
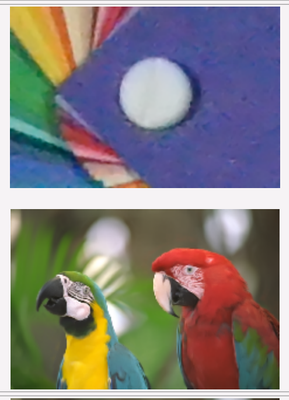
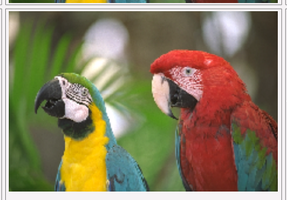
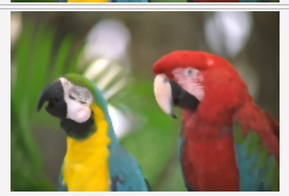
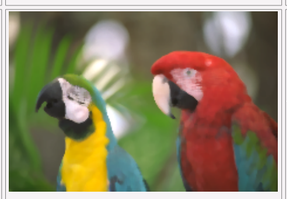
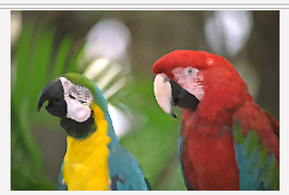
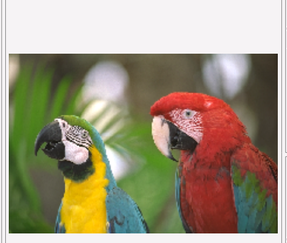
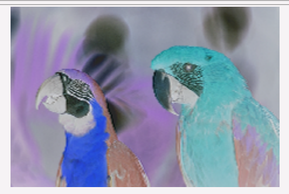
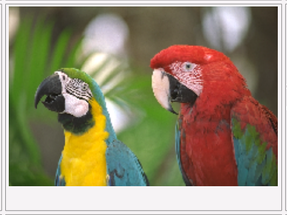
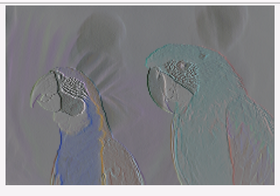
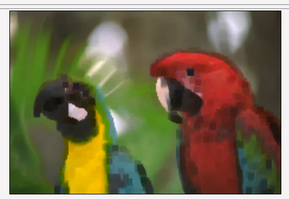
		`gmic geo2.png -channels 1 -o channels_V.png`	Canal vert 
		`gmic geo2.png -channels 2 -o channels_B.png`	Canal bleu 
-slices { [image0] | z0[%] }, _{ [image1] | z1[%] } Select specified slices of selected images.	objet 3D	`gmic -sphere3d 200,1 -slices[-1] 10 -o slices.png` (à vérifier)	
-lines { [image0] | y0[%] }, _{ [image1] | y1[%] } Select specified lines of selected images.		`gmic geo.png -lines 80 -o lines.png`	Image origine  Image étirée en hauteur (100px)
-columns { [image0] | x0[%] }, _{ [image1] | x1[%] } Select specified columns of selected images.		`gmic geo.png -columns 60 -o columns.png`	Image origine  Image étirée en largeur (100px) 
-rotate angle, _borders, _interpolation, _cx[%], _cy[%], _zoom Rotate selected images with specified angle (in deg.). 'borders' can be { 0=dirichlet | 1=neumann | 2=cyclic }. 'interpolation' can be { 0=none | 1=linear | 2=bicubic }. When rotation center ('cx','cy') is specified, the size of the image is preserved.		`gmic geo.png -rotate -30,0,1 -o rotate.png`	
		`gmic geo.png -rotate -30,1,1 -o rotate2.png`	
		`gmic geo.png -rotate -30,2,1 -o rotate3.png`	

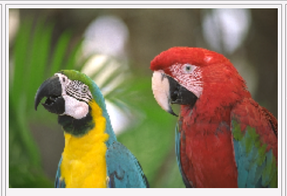
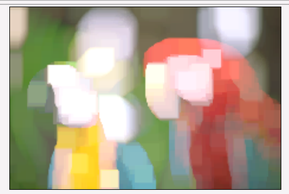
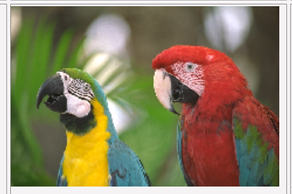
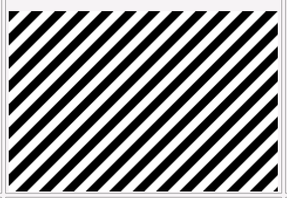
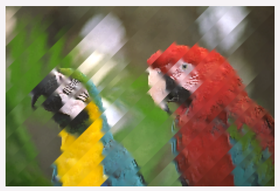
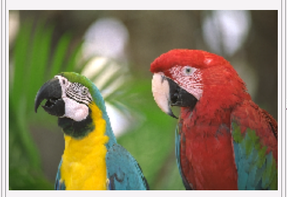
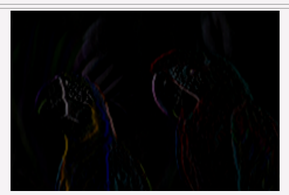
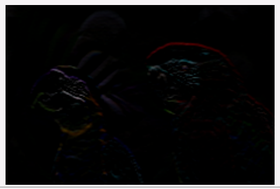
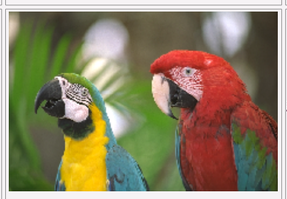
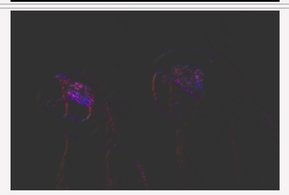
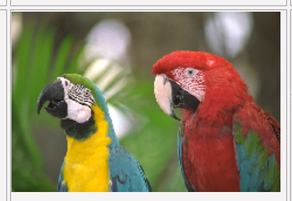
		`gmic geo.png -rotate -30,0,1,30,60 -o rotate4.png`	
-mirror axis={ x | y | z | c } Mirror selected images along specified axis.		`gmic geo.png -mirror x -o mirror.png`	
-shift vx[%], _vy[%], _vz[%], _vc[%], _borders Shift selected images by specified displacement vector. 'borders' can be { 0=dirichlet | 1=neumann | 2=cyclic }.		`gmic geo2.png -shift 20 -o shift.png`	
-transpose Transpose selected images.		`gmic geo.png -transpose -o transpose.png`	
-invert Compute the inverse of the selected matrices.	.	`gmic (0,-1;-1,0) -invert -o invert_m2.png` `Visualisation matrice origine :` `gmic (200,30,50;44,221,66;255,0,127) -n 0,255 -o o_invert.png` `gmic (200,30,50;44,221,66;255,0,127) -invert -n 0,255 -o invert.png`	zoom image *10  zoom image * 10  zoom image * 10 
-solve [image] Solve linear system AX = B for selected B-vectors and specified A-matrix.	.	Voir la fonction `_function1d` du fichier gmic_def.1442.	.
-trisolve [image] Solve tridiagonal system AX = B for selected B-vectors and specified tridiagonal A-matrix. Tridiagonal matrix must be stored as a 3 column vector, where 2nd column contains the diagonal coefficients, while 1st and 3rd columns contain the left and right coefficients.	.	Non testé	.
-eigen Compute the eigenvalues and eigenvectors of specified symmetric matrices.	.	Non testé ([matrice]) -eigen	.
-dijkstra starting_node>=0,ending_node>=0 Compute minimal distances and pathes from specified adjacency matrices by the Dijkstra algorithm.	.	Non testé	.
-permute permutation Permute selected image axes by specified permutation. 'permutation' is a combination of the character set {x|y|z|c}, e.g. 'xycz', 'cxyz', ..		`gmic geo.png -permute "yxcz" -o permute.png`	
-unroll axis={ x | y | z | c } Unroll selected images along specified axis.	.	Non testé (Exemples dans le fichier gmic_def.1442).	.

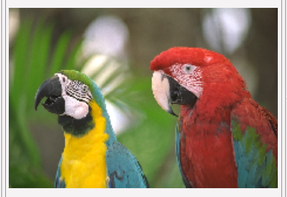
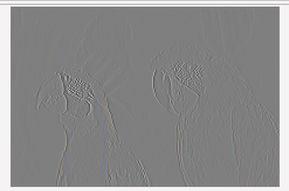
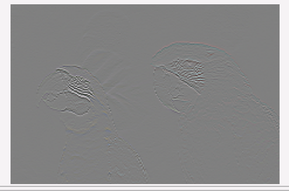
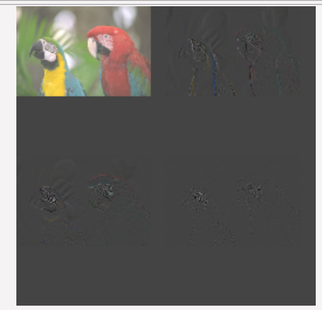
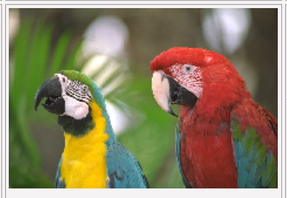
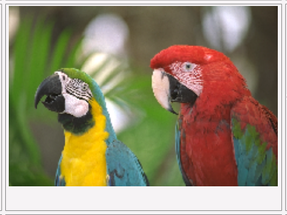
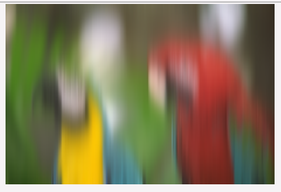
-split axis={ x | y | z | c }, _nb_parts | patch_x>0, _patch_y>0, _patch_z>0, _patch_v>0,borders | value, _keep_splitting_values={ + | - } Split selected images along specified axis, patch or scalar value. (eq. to '-s'). 'nb_parts' can be { 0=maximum split | >0=split in N parts | <0=split in parts of size -N }. 'borders' can be { 0=dirichlet | 1=neumann }.		`gmic geo.png -split y -o split.png`	128 images de 128px*1px de split_000000.png à split_000127.png
-append axis={ x | y | z | c }, _alignment Append selected images along specified axis. (eq. to '-a'). 'alignment' can be { p=left | c=center | n=right }.	.	`geo.png -append x`	.
-warp [image], _is_relative={ 0 | 1 }, _interpolation={ 0 | 1 }, _borders, _nb_frames>0 Warp selected image with specified displacement field. 'borders' can be { 0=dirichlet | 1=neumann | 2=cyclic }.	 	`gmic geo.png geo2.png -warp[-2] [-1],1,1,0 -o warp.png`	

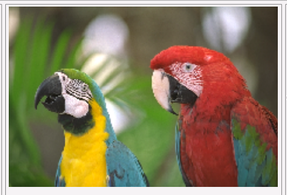
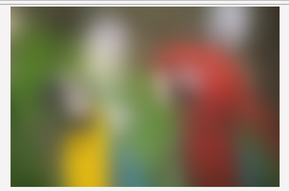
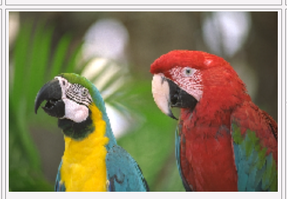
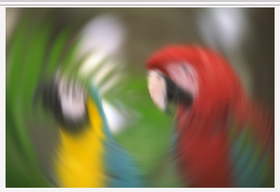
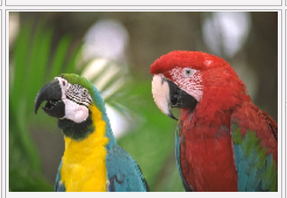
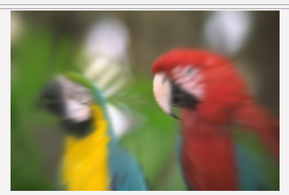
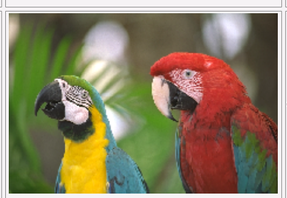
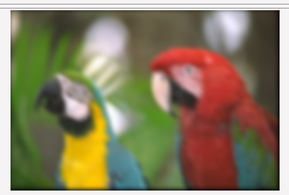
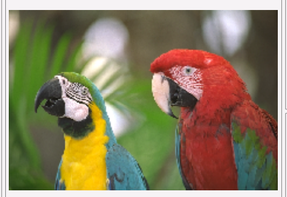
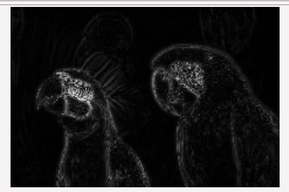
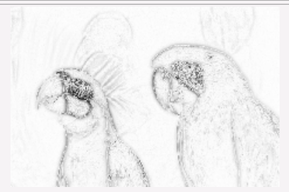
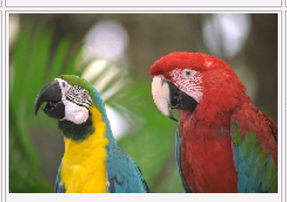
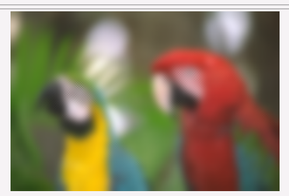
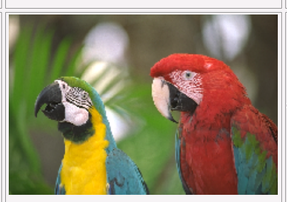
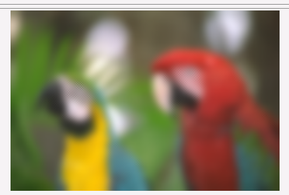
Les tests sur les filtres

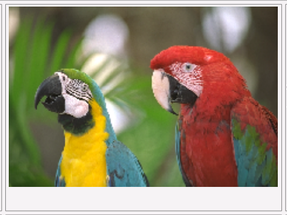
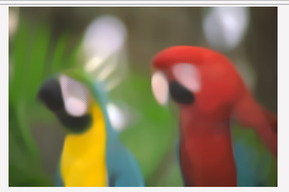
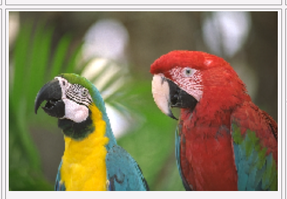
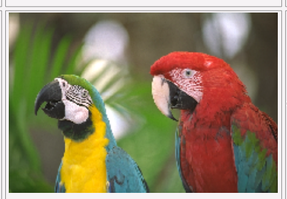
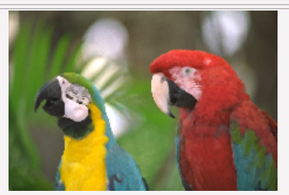
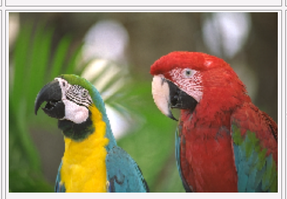
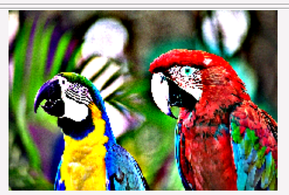
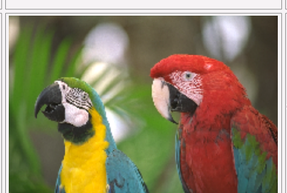
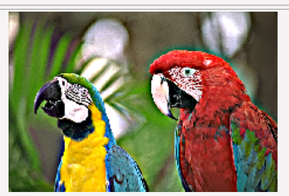
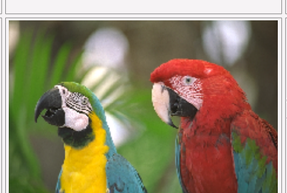
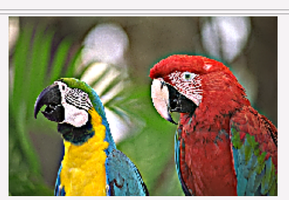
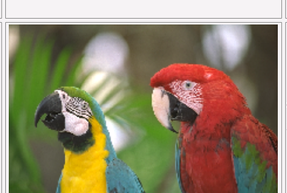
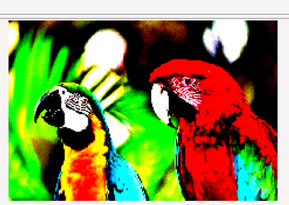
Paramètres de la ligne de commande (aide de G'MIC)	Image(s) d'origine	Ligne de commande (`gmic image -[filtre] -o resultat.png`)	Résultat
-deriche std_variation>=0[%],order={ 0 | 1 | 2 },axis={ x | y | z | c }, _borders Apply Deriche recursive filter with specified standard deviation, order, axis and border conditions on selected images. 'borders' can be { 0=dirichlet | 1=neumann }.		`gmic perroquets.png -deriche 0.5,1,x,0 -c 0,255 -o deriche.png` Exemple de commande qui teste les valeurs hors normes : `gmic geo.png -deriche 0.5,1,x,0 "-if {im<0} -c 0,255 -elif {iM>255} -c 0,255 -endif" -o test_validite_deriche.png`	
-blur std_variation>=0[%], _borders Blur selected images by quasi-gaussian recursive filter. 'borders' can be { 0=dirichlet | 1=neumann }.		`-blur 3,0`	
-bilateral std_variation_s>0[%],std_variation_r>0 Blur selected images by anisotropic bilateral filtering. 'borders' can be { 0=dirichlet | 1=neumann }.		`-bilateral 25,20`	
-denoise std_variation_s>=0, _std_variation_p>=0, _patch_size>0, _lookup_size>0, _smoothness, _fast_approx={ 0 | 1 } Denoise selected images by non-local patch averaging.		`-denoise 10,30,3,5,0,1`	

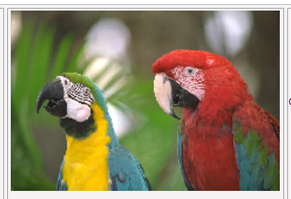
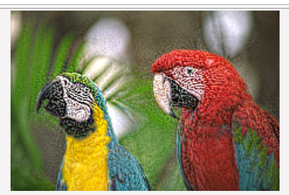
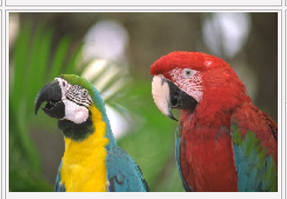
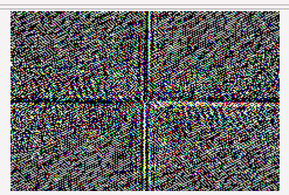
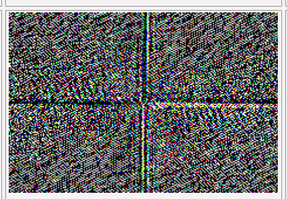
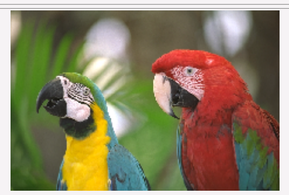
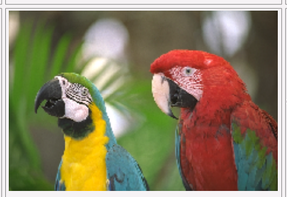
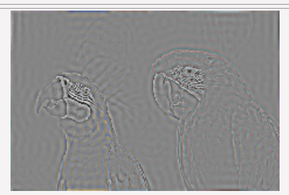
-smooth amplitude>=0, _sharpness>=0, _anisotropy, _alpha, _sigma, _dl>0, _da>0, _precision>0, interpolation, _fast_approx={ 0 | 1 } | nb_iterations>=0, _sharpness>=0, _anisotropy, _alpha, _sigma, _dt>0,0 | [image], _amplitude>=0, _dl>0, _da>0, _precision>0, _interpolation, _fast_approx={ 0 | 1 } | [image], _nb_iters>=0, _dt>0,0 Smooth selected images anisotropically using diffusion PDE's, with specified field of diffusion tensors. 'anisotropy' must be in [0,1]. 'interpolation' can be { 0=nearest | 1=linear | 2=runge-kutta }.		`-smooth 70`	
-median radius>=0 Apply median filter of specified radius on selected images.		`-median 4`	
-sharpen amplitude>=0 | amplitude>=0,1, _edge>=0, _alpha, _sigma Sharpen selected images by inverse diffusion or shock filters methods.		`-sharpen 200`	
-convolve [image], _borders, _is_normalized={ 0 | 1 } Convolve selected images by specified mask. 'borders' can be { 0=dirichlet | 1=neumann }.		`gmic perroquets.png (-1,-0.5;-1,-1.5;-1,1) -convolve[0] [1] -n 0,255 -o[0] convolve.png`	
		`gmic perroquets.png (-1,-1,0.5) -convolve[0] [1] -n 0,255 -o[0] convolve1.png`	
-correlate [image], _borders, _is_normalized={ 0 | 1 } Correlate selected images by specified mask. 'borders' can be { 0=dirichlet | 1=neumann }.		`gmic perroquets.png (1,-1.2) -correlate [1] -n 0,255 -o[0] correlate.png`	
-erode size>=0' | size_x>=0,size_y>=0, _size_z>=0 | [image], _borders, _borders, _is_normalized={ 0 | 1 } Erode selected images by a rectangular or the specified structuring element. 'borders' can be { 0=dirichlet | 1=neumann }.		`-erode 5`	

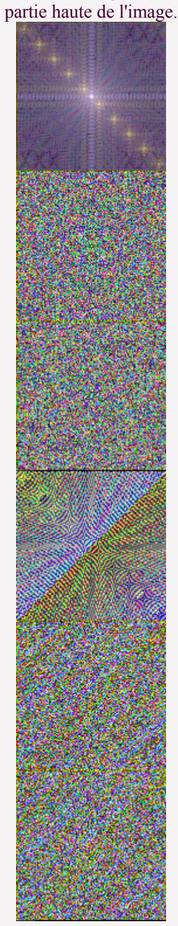
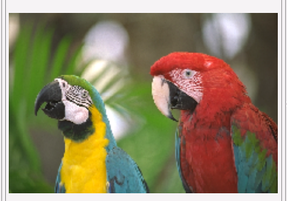
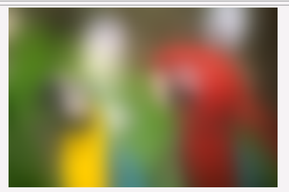
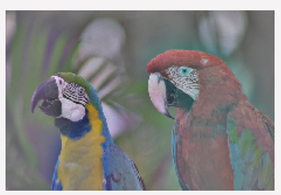
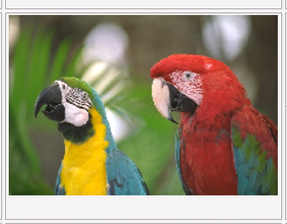
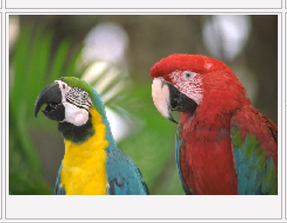
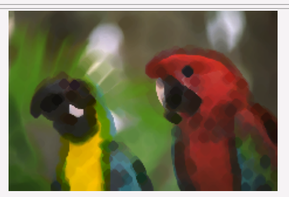
-dilate size>=0 | size_x>=0,size_y>=0,size_z>=0 | [image], _borders, _borders, _is_normalized={ 0 | 1 } Dilate selected images by a rectangular or the specified structuring element. 'borders' can be { 0=dirichlet | 1=neumann }.		-dilate 15	
-inpaint [image] Inpaint selected images by specified mask.	 	gmic perroquets.png fond_1.png -inpaint [1] [0] -o[0] inpaint.png	
-gradient { x | y | z }..{ x | y | z }, _scheme | (no args) Compute the gradient components (first derivatives) of selected images. 'scheme' can be { -1=backward | 0=centered | 1=forward | 2=sobel | 3=rotation-invariant (default) | 4=recursive }. (no args) compute all significant 2d/3d components.		gmic perroquets.png -gradient -c 0,255 -o gradient.png	 
-structuretensors _scheme Compute the structure tensor field of selected images. 'scheme' can be { 0=centered | 1=forward-backward1 | 2=forward-backward2 }.		gmic perroquets.png -structuretensors -n 0,255 -o structuretensors.png	
-edgetensors sharpness>=0, _anisotropy, _alpha, _sigma,is_sqrt={ 0 | 1 } Compute the diffusion tensors of selected images for edge-preserving smoothing algorithms. 'anisotropy' must be in [0,1].		gmic perroquets.png -edgetensors 0.1,0.1,1,1 -window -wait 2000 -n 0,255 -o edgetensors.png	

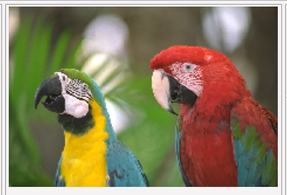
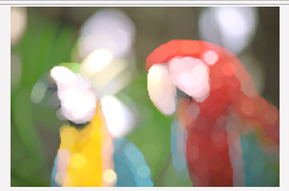
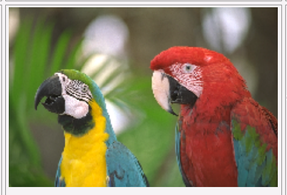
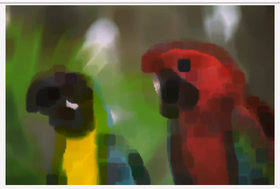
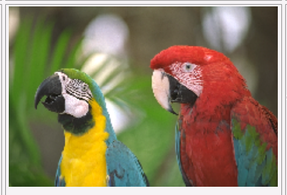
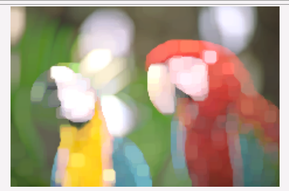
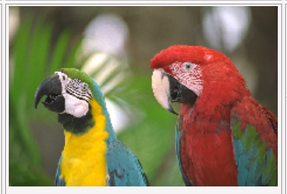
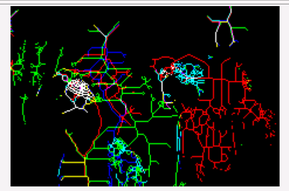
-hessian { xx | xy | xz | yy | yz | zz }..{ xx | xy | xz | yy | yz | zz } | (no args) Compute the hessian components (second derivatives) of selected images. (no args) compute all significant components.		`gmic perroquets.png -hessian -n 0,255 -o hessian.png`	  
-haar scale>0 Compute the direct haar multiscale wavelet transform of selected images.		`gmic perroquets.png -crop 0,0,255,255 -haar 0.5 -n 0,255 -o haar.png` (`-crop 0,0,255,255` pour obtenir une image de 256*256)	
-ihaar scale>0 Compute the inverse haar multiscale wavelet transform of selected images.	512_512.png (512pixels*512pixels) perroquets.png	`gmic 512_512.png -haar 2 -ihaar 2 -o resultat.png` `gmic perroquets.png -crop 0,0,255,255 -haar 2 -ihaar 2 -o resultat2.png`	resultat.png=512_512.png resultat2.png=perroquets.png(256*256)
-fft Compute the direct fourier transform of selected images.	image origine	`gmic [image origine] -fft [opérations] -ifft -o [résultat]` Lien sur les transformées de Fourrier avec G'MIC Pour faire une conversion : - fftpolar et l'inverse ifftpolar - gmic geo.png -float2fft8 -o geo2fft.png et l'inverse gmic geo2fft.png -fft82float -c 0,255 -o fft2geo.png	.
-ifft Compute the inverse fourier transform of selected images.	image origine		
-blur_x amplitude[%]>=0, _borders={ 0 | 1 } Blur selected images along the X-axis.		`gmic perroquets.png -blur_x 10 -o blur_x.png`	
-blur_y amplitude[%]>=0, _borders={ 0 | 1 } Blur selected images along the Y-axis.		`gmic perroquets.png -blur_y 10 -o blur_y.png`	
-blur_z amplitude[%]>=0, _borders={ 0 | 1 } Blur selected images along the Z-axis.	3D	.	.

-blur_xy amplitude_x[%],amplitude_y[%], _borders={ 0 | 1 } Blur selected images along the X and Y axes.		gmic perroquets.png -blur_xy 10,10 -o blur_xy.png	
-blur_xyz amplitude_x[%],amplitude_y[%],amplitude_z, _borders={ 0 | 1 } Blur selected images along the X, Y and Z axes.	3D	.	.
-blur_angular _amplitude[%], _cx, _cy Apply angular blur on selected images.		gmic perroquets.png -blur_angular 2,0.5,0.5 -o blur_angular.png	
-blur_radial _amplitude[%], _cx, _cy Apply radial blur on selected images.		gmic perroquets.png -blur_radial 2,0.5,0.5 -o blur_radial.png	
-blur_linear _amplitude1[%], _amplitude2[%], _angle=0, _borders={ 0=dirichlet | 1=neumann } Apply linear blur on selected images, with specified angle and amplitudes.		gmic perroquets.png -blur_linear 2,2,30,0 -o blur_linear.png	
-dog _sigma1>=0[%], _sigma2>=0[%] Compute difference of gaussian on selected images.		gmic perroquets.png -dog 1.5,0 -n 0,255 -o dog.png	
		gmic perroquets.png -dog 1.5,0 -n 0,255 -negative -o negative_dog.png	
-pde_flow _nb_iter>=0, _dt, _velocity_command, _sequence_flag={ 0 | 1 } Apply iterations of a generic PDE flow on selected images. paramètre 3 : | laplacian | Iee | curvature |		gmic perroquets.png -pde_flow 20,40,laplacian,0 -o pde_flow.png	
-heat_flow _nb_iter>=0, _dt, _sequence_flag={ 0 | 1 } Apply iterations of the heat flow on selected images.		gmic perroquets.png -heat_flow 20,40,0 -o heat_flow.png	

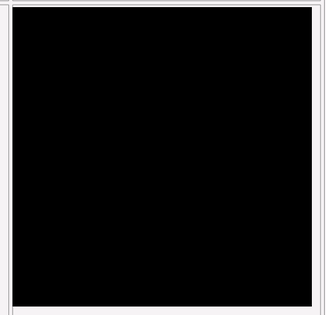
-meancurvature_flow _nb_iter>=0, _dt, _sequence_flag={ 0 | 1 } Apply iterations of the mean curvature flow on selected images.		`gmic perroquets.png -meancurvature_flow 20,40,0 -o meancurvature_flow.png`	
-tv_flow _nb_iter>=0, _dt, _sequence_flag={ 0 | 1 } Apply iterations of the total variation flow on selected images.		`gmic perroquets.png -tv_flow 5,10,0 -o tv_flow.png`	Très peu de différence par rapport à l'image d'origine.
-inpaint_flow _nb_iter1>=0, _nb_iter2>=0, _dt>=0, _alpha, _sigma Apply iteration of the inpainting flow on selected images.	.	?	?
-remove_hotpixels _mask_size>0, _threshold[%]>0 Remove hot pixels in selected images.		`gmic perroquets.png -remove_hotpixels 3,10 -o remove_hotpixels.png`	
-richardson_lucy amplitude[%]>=0, _nb_iter>=0, _dt>=0, _regul>=0, _regul_type={ 0=Tikhonov | 1=meancurv. | 2=TV } Deconvolve image with the iterative Richardson-Lucy algorithm.		`gmic perroquets.png -richardson_lucy 10,20,40,0,0 -c 0,255 -o richardson_lucy.png`	
-oriented_richardson_lucy amplitude1[%]>=0, _amplitude2[%]>=0, _angle, _nb_iter>=0, _dt>=0, _regul>=0, _regul_type={ 0=Tikhonov | 1=meancurv. | 2=TV } Deconvolve image with the iterative Richardson-Lucy algorithm for oriented kernels.		`gmic perroquets.png -oriented_richardson_lucy 10,10,30,20,40,0,0 -c 0,255 -o oriented_richardson_lucy.png`	
-unsharp _radius[%]>=0, _amount>=0, _threshold[%]>=0 Apply unsharp mask on selected images.		`gmic perroquets.png -unsharp 3,2,5 -c 0,255 -o unsharp.png`	
-unsharp_octave _nb_scales>0, _radius[%]>=0, _amount>=0,threshold[%]>=0 Apply octave sharpening on selected images.		`gmic perroquets.png -unsharp_octave 2,3,2,5 -c 0,255 -o unsharp_octave.png`	
-normalize_local _amplitude>=0, _radius>0, _n_smooth>=0[%], _a_smooth>=0[%], _is_cut={ 0 | 1 }, _min=0, _max=255 Normalize selected images locally.		`gmic perroquets.png -normalize_local 10,3,10,10,0,0,255 -c 0,255 -o normalize_local.png`	

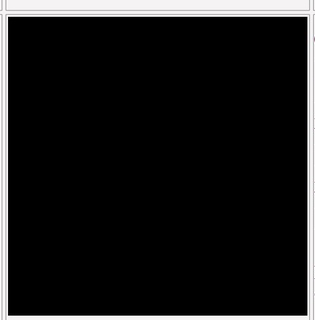
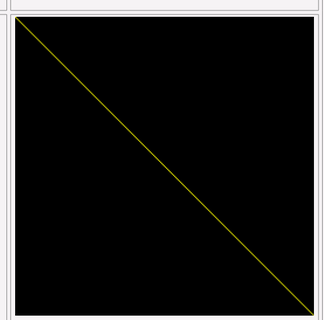
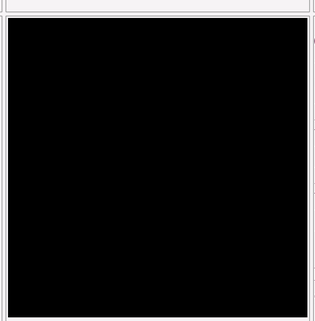
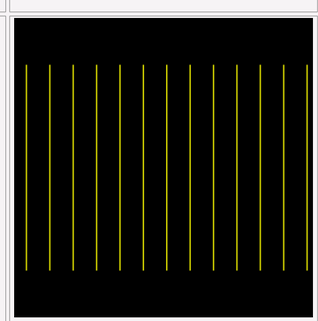
-map_tones _threshold>=0, _gamma>=0, _smoothness>=0,iter>=0 Apply tone mapping operator based on Poisson equation.		`gmic perroquets.png -map_tones 5,1,10,2 -o map_tones.png`	
-fftpolar Compute fourier transform of selected images, as centered magnitude/phase images.		`gmic perroquets.png -fftpolar -o[-1] fftpolar.png`	
-ifftpolar Compute inverse fourier transform of selected images, from centered magnitude/phase images.		`gmic perroquets.png -ifftpolar -o ifftpolar.png`	
-convolve_fft Convolve selected images two-by-two through fourier transforms.	 	gmic geo.png geo2.png -convolve_fft[-2,-1] -n 0,255 -o convolve_fft.png	? 
-deconvolve_fft Deconvolve selected images two-by-two through fourier transforms.	.	?	.
-bandpass _min_freq[%], _max_freq[%] Apply bandpass filter to selected images.		`gmic perroquets.png -bandpass 5%,95% -n 0,255 -o bandpass.png`	
-watermark_fourier _text, _size>0 Add an textual watermark in the frequency domain of selected images.	geo2.png (3couleurs) 	`gmic geo2.png -watermark_fourier XXX,24 -c 0,255 -o watermark_fourier.png`	watermark_fourier.png (l'image semble identique mais le codage donne 1021 couleurs) 

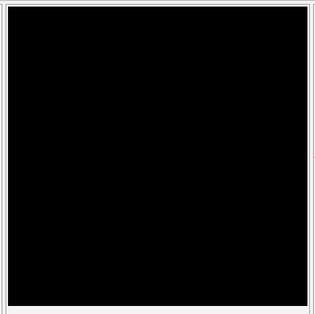
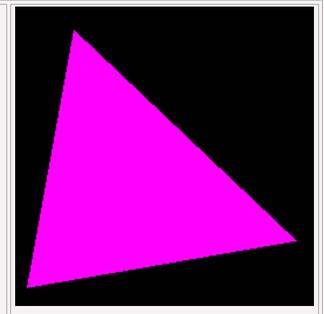
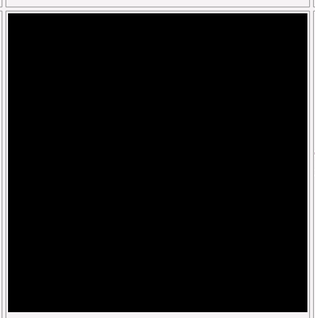
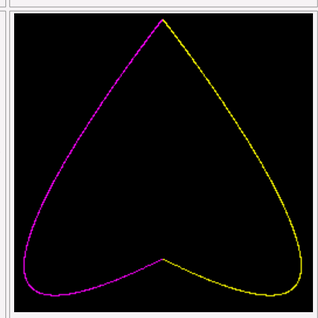
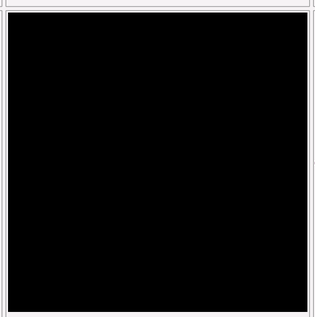
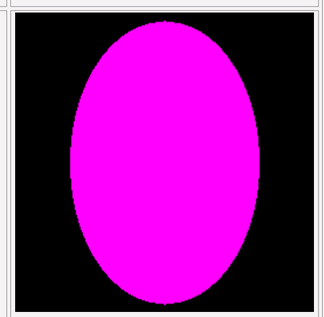
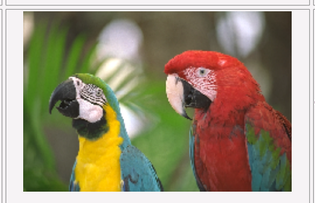
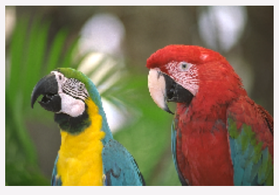
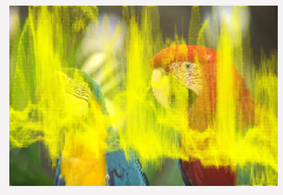
	watermark_fourier.png 	Visualiser le message inscrit par la fonction -watermark_fourier : `gmic watermark_fourier.png -float2fft8 -o visuwatermark_fourier.png`	Il est possible de voir le message XXX aux 4 coins de la partie haute de l'image. 
-split_freq smoothness>0[%] Split selected images into low and high frequency parts.		`gmic perroquets.png -split_freq 10 -n 0,255 -o split_freq.png`	 
-compose_freq Compose selected low and high frequency parts into new images.		`gmic perroquets.png -compose_freq -o compose_freq.png`	? Résultat identique à l'image d'origine.
-erode_oct _size>=0 Apply octagonal erosion of selected images by specified size.		`gmic perroquets.png -erode_oct 10 -o erode_oct.png`	

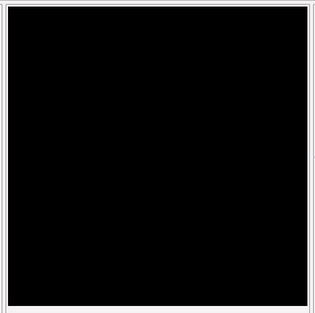
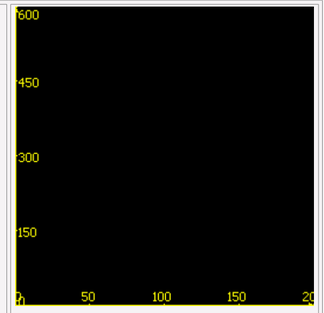
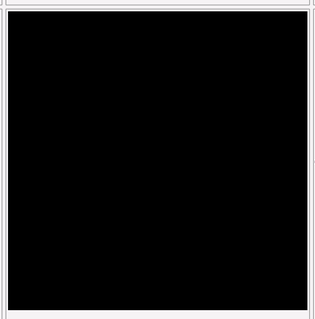
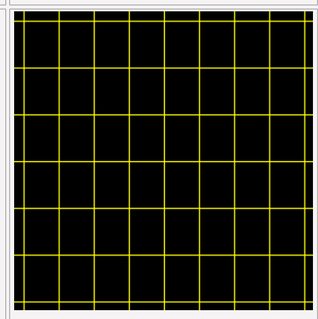
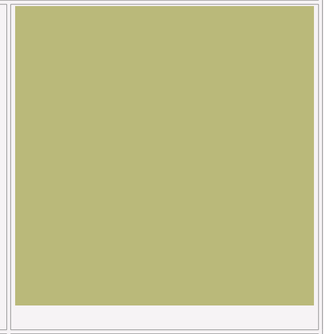
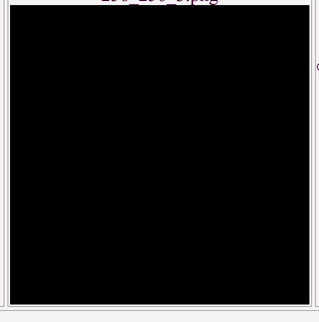
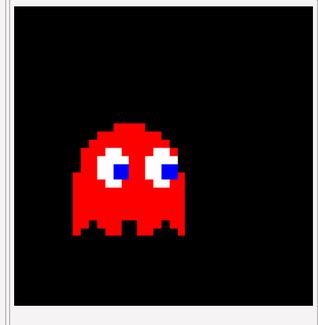
-dilate_oct _size>=0 Apply octagonal dilation of selected image by specified size.		`gmic perroquets.png -dilate_oct 10 -o dilate_oct.png`	
-erode_circ _size>=0 Apply circular erosion of selected images by specified size.		`gmic perroquets.png -erode_circ 10 -o erode_circ.png`	
-dilate_circ _size>=0 Apply circular dilation of selected image by specified size.		`gmic perroquets.png -dilate_circ 10 -o dilate_circ.png`	
-skeleton Compute skeleton of binary shapes using morphological thinning.		`gmic perroquets.png -skeleton -n 0,255 -o skeleton.png`	

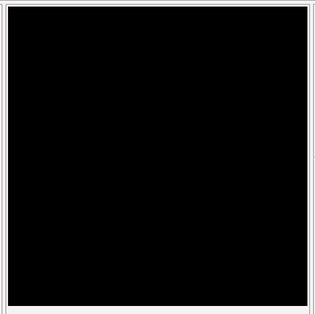
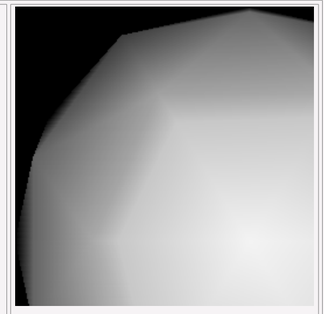
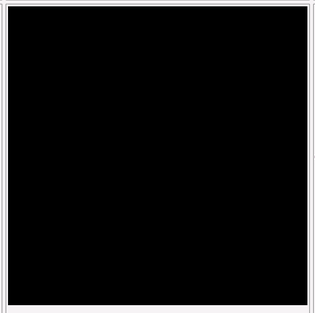
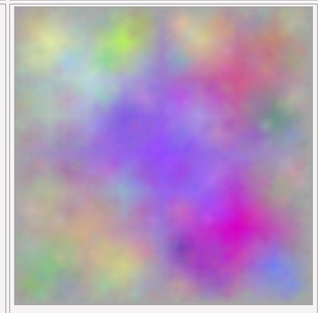
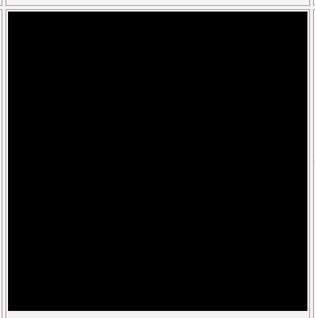
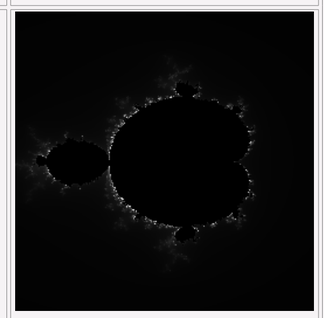
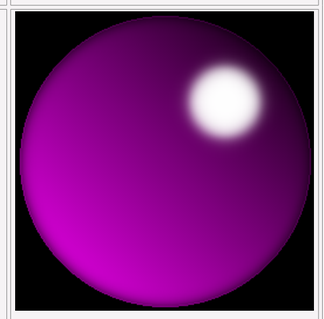
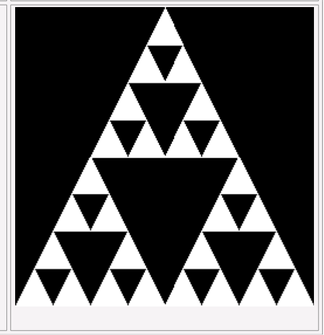
Création d'images et dessin

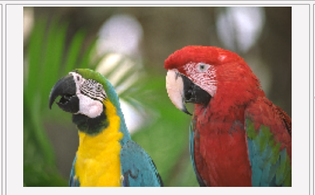
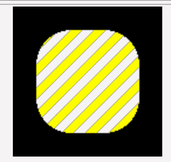
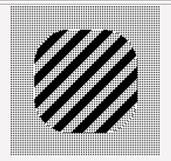
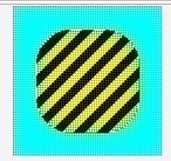
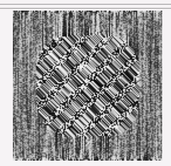
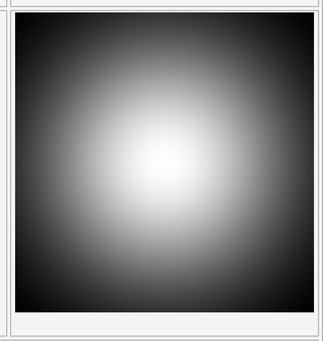
Paramètres de la ligne de commande (aide de G'MIC)	Image(s) d'origine	Ligne de commande	Résultat
Créer image noire de 1*1px (1 canal niveaux de gris)	aucune	`gmic 1 -o 1_1.png`	.
Créer image noire de 256*256px (3 canaux RVB)	aucune	`gmic 256,256,1,3 -o 256_256_3.png`	
Créer image transparente de 256*256px (4 canaux RVB)	aucune	`gmic 256,256,1,4 -o 256_256_4.png`	

-histogram nb_levels>0[%], _val0[%], _val1[%] Compute the histogram of selected images. If value range is specified, the histogram is estimated only for pixels in the specified value range.	.	Voir les fonctions `histogram_cumul`, `plot2value`, etc. du fichier gmic_def.1442.	.
-distance isovalue Compute the unsigned distance function to specified isovalue.	.	Voir les fonctions `edges`, `sponge`, etc. du fichier gmic_def.1442.	.
-eikonal nb_iterations>=0, _band_size>=0 Compute iterations of the eikonal equation (signed distance function) on selected images.	.	?	.
-watershed [priority_image], _fill_lines={ 0 | 1 } Compute the watershed transform of selected images.	.	Voir la fonction `segment_watershed` du fichier gmic_def.1442.	.
-label Label connected components in selected images.	.	?	.
-displacement [source_image], _smoothness>=0, _precision>0, _nb_scales>=0,iteration_max>=0, is_backward={ 0 | 1 } Estimate displacement field between selected images and specified source. If 'nbscales'=0, the number of needed scales is estimated from the image size.	.	Voir les fonctions `morph`, `deinterlace`, etc. du fichier gmic_def.1442.	.
-sort _ordering={ + | - }, _axis={ x | y | z | c } Sort pixel values of selected images.	.	`-sort[-1] +,y`	.
-mse Compute MSE (Mean-Squared Error) matrix between selected images.	.	?	.
-psnr _max_value Compute PSNR (Peak Signal-to-Noise Ratio) matrix between selected images.	.	?	.
-point x[%],y[%], _z[%], _opacity, _color1,.. Set specified colored pixel on selected images. Default color value is '0'.		Contenu du fichier de commandes point.txt `#@gmic test_point : : Tester la fonction point` `test_point :` `-input 256_256_3.png` `X=0` `-do` `-point $X,$X,0,1,255,255,0` `X={$X+1}` `-while {$X<256}` `-o point.png` Ligne de commandes : `gmic -m point.txt -test_point`	
-line x0[%],y0[%],x1[%],y1[%], _opacity, _pattern, _color1,..' Draw specified colored line on selected images. 'pattern' is an hexadecimal number starting with '0x' which can be omitted even if a color is specified. Default color value is '0'.		Contenu du fichier de commandes line.txt `#@gmic test_line : : Tester la fonction line` `test_line :` `-input 256_256_3.png` `X=10` `-do` `-line $X,40,$X,215,1,255,255,0` `X={$X+20}` `-while {$X<256}` `-o line.png` Ligne de commandes : `gmic -m line.txt -test_line`	

-polygon N,x1[%],y1[%],...,xN[%],yN[%], _opacity, _pattern, _color1,.. Draw specified colored N-vertices polygon on selected images. 'pattern' is an hexadecimal number starting with '0x' which can be omitted even if a color is specified. If a pattern is specified, the polygon is drawn outlined instead of filled. Default color value is '0'.		`gmic 256_256_3.png -polygon 3,50,20,240,200,10,240,1,255,0,255 -o polygon.png`	
-spline x0[%],y0[%],u0[%],v0[%],x1[%],y1[%],u1[%],v1[%], _opacity, _pattern, _color1,.. Draw specified colored spline curve on selected images. 'pattern' is an hexadecimal number starting with '0x' which can be omitted even if a color is specified. Default color value is '0'.		`gmic 256_256_3.png -spline 127,5,0,0,127,210,800,-400,1,255,0,255 -spline 127,5,0,0,127,210,-800,-400,1,255,255,0 -o spline.png`	
-ellipse x[%],y[%],R[%],r[%], _angle, _opacity, _color1,.. Draw specified colored ellipse on selected images. 'pattern' is an hexadecimal number starting with '0x' which can be omitted even if a color is specified. If a pattern is specified, the ellipse is drawn outlined instead of filled. Default color value is '0'.		`gmic 256_256_3.png -ellipse 50%,50%,120,80,90,1,255,0,255 -o ellipse2.png`	
-text text, _x[%], _y[%], _font_height>=0, _opacity, _color1,.. Draw specified colored text string on selected images. Exact pre-defined sizes are '13','24','32' and '57'. Default color value is '0'. Specifying a target image with a size of 1x1x1x1 resizes it to new dimensions such that the image contains the entire text string.		`gmic perroquets.png -text Exemple\ G\'MIC,20,20,32,1,1,255,255,255 -o text.png`	
-graph [function_image], _plot_type, _vertex_type, _ymin, _ymax, _opacity, _pattern, _color1,.. | 'formula', _resolution>=0, _plot_type, _vertex_type, _xmin,xmax, _ymin, _ymax, _opacity, _pattern, _color1,.. Draw specified function graph on selected images. 'plot_type' can be { 0=none | 1=lines | 2=splines | 3=bar }. 'vertex_type' can be { 0=none | 1=points | 2,3=crosses | 4,5=circles | 6,7=squares }. 'pattern' is an hexadecimal number starting with '0x' which can be omitted even if a color is specified. Default color value is '0'.		`gmic perroquets.png -graph [-1],1,0,255,0,0.05,255,255,0 -o graph.png`	

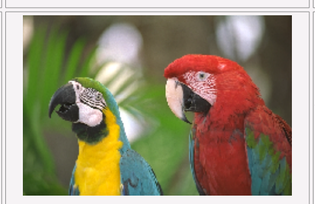
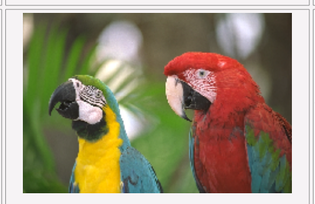
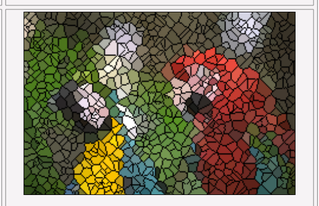
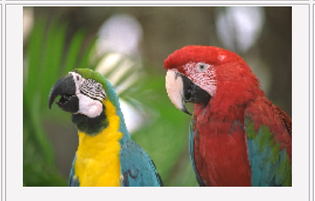
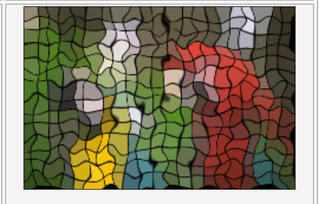
-axes x0,x1,y0,y1, _opacity, _pattern, _color1,.. Draw xy-axes on selected images. 'pattern' is an hexadecimal number starting with '0x' which can be omitted even if a color is specified. Default color value is '0'.		gmic 256_256_3.png -axes 0,200,600,0,1,255,255,0 -o axes.png	
-grid sizex[%]>=0,sizey[%]>=0, _offsetx[%], _offsety[%], _opacity, _pattern, _color1,.. Draw xy-grid on selected images. 'pattern' is an hexadecimal number starting with '0x' which can be omitted even if a color is specified. Default color value is '0'.		gmic 256_256_3.png -grid 30,40,8,8,1,255,255,0 -o grid.png	
-quiver [function_image], _sampling>0, _factor, _is_arrow={ 0 | 1 }, _opacity, _pattern, _color1,.. Draw specified 2d vector/orientation field on selected images. 'pattern' is an hexadecimal number starting with '0x' which can be omitted even if a color is specified. Default color value is '0'.	La commande '-quiver' permet d'afficher un champ de vecteur sur une image, et peut s'utiliser par exemple comme ceci : gmic 256,256,1,2,"cos(c*x/10)*sin(y/20)" --r 300%,300% -n[-1] 0,255 -quiver[-1] [0],20,18 -rm[0]		
-flood x[%], _y[%], _z[%], _tolerance>=0, _opacity, _color1,.. Flood-fill selected images using specified value and tolerance. Default color value is '0'.	Image transparente 256_256_4.png	Remplissage de jaune, opacité=0.5 gmic 256_256_4.png -flood 100,200,0,1,0.5,255,255,0 -o flood.png	
-image [sprite], _x[%], _y[%], _z[%], _c[%], _opacity, _[sprite_mask] Draw specified sprite image on selected images.	 sprite.png 256_256_3.png 	gmic 256_256_3.png sprite.png -image[-2] [-1],50,100,0,0,1 -o[0] image.png	

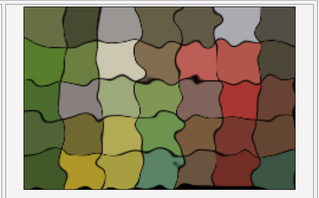
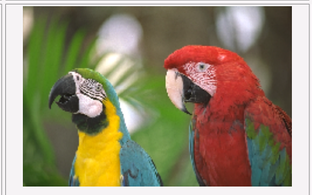
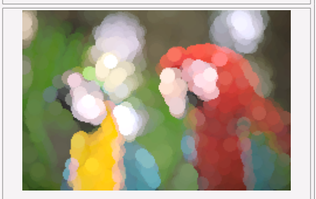
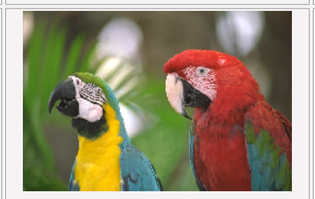
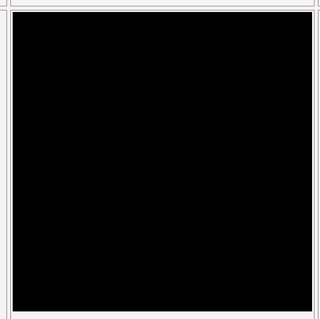
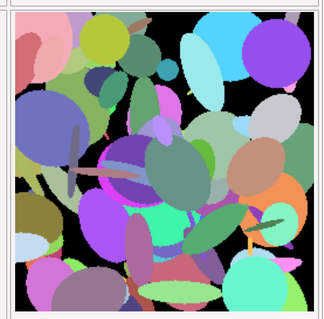
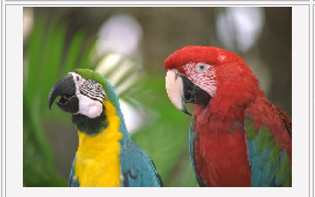
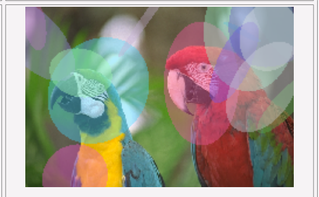
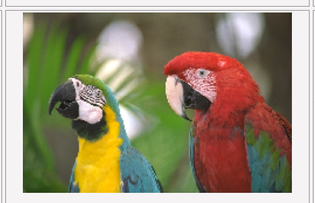
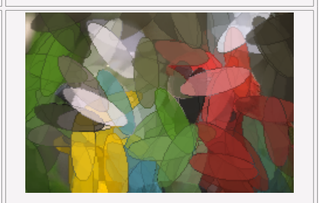
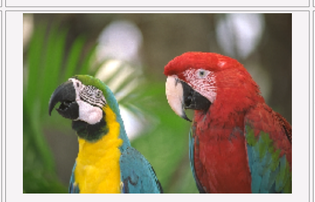
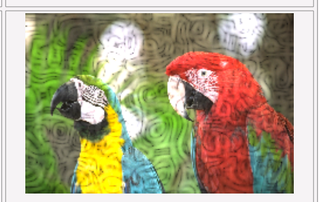
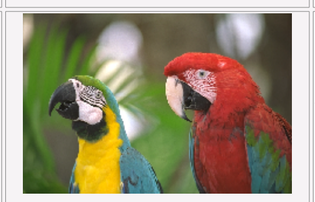
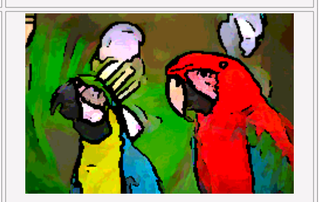
-object3d [object3d], _x[%], _y[%], _z, _opacity, _is_zbuffer={ 0 | 1 } Draw specified 3d object on selected images.		`gmic 256_256_3.png -sphere3d 200,1 -object3d [-1],200,200,0,1,0 -o[0] object3d.png`	
-plasma alpha, _beta, _opacity Draw a random colored plasma on selected images.		`gmic 256_256_3.png -plasma 4,0.1 -n 0,255 -o plasma.png`	
-mandelbrot z0r,z0i,z1r,z1i, _iteration_max>=0, _is_julia={ 0 | 1 }, _c0r, _c0i, _opacity Draw mandelbrot/julia fractal on selected images.		`gmic 256_256_3.png -mandelbrot[-1] -1.5,1.5,0.9,-1.5,250,0,0.3,0.03,1 -o mandelbrot.png`	
-ball _R, _G, _B Draw a colored RGBA ball sprite on selected images.	sans	`gmic 256,256,1,3 -ball 255,0,255 -to_colormode 3 -o ball.png`	
-sierpinski recursion_level>=0 Draw Sierpinski triangle on selected images.	sans	`gmic 256,256,1,3 -sierpinski 3 -o sierpinski.png`	

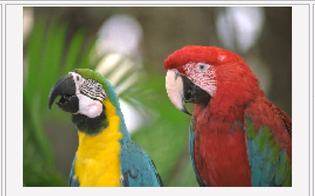
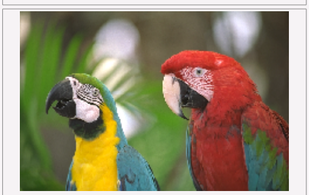
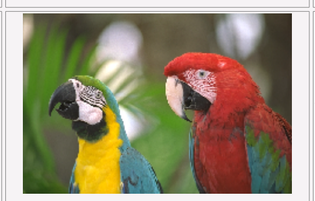
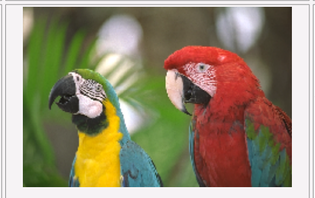
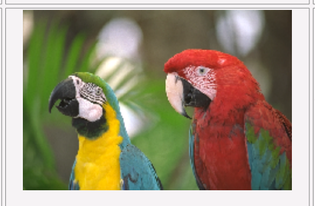
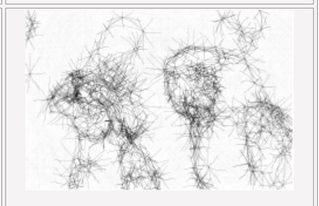
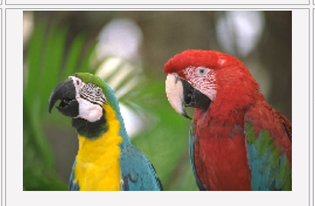
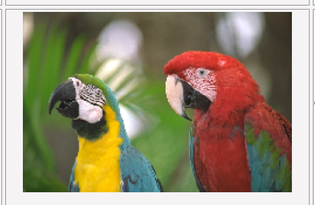
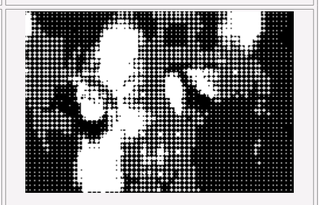
-text_outline text, _x[%], _y[%], _font_height>0, _outline>=0, _opacity, _color1,.. Draw specified colored and outlined text string on selected images.		`gmic perroquets.png -text_outline Exemple\ G\'MIC,0,0,32,1,1,255,255,255 -o text_outline.png`	
-histogram_cumul _nb_levels>0, _is_normalized={ 0 | 1 } Compute cumulative histogram of selected images.	.	Voir la fonction `transfer_histogram` du fichier gmic_def.1442	.
-direction2rgb Compute RGB representation of selected 2d direction fields.	.	Voir la fonction `gradient2rgb` du fichier gmic_def.1442	.
-vector2tensor Convert selected vector fields to corresponding diffusion tensor fields.		`gmic geo2.png -vector2tensor -o vector2tensor.png`	
-rgb2bayer _start_pattern=0, _color=0 Transform selected color images to RGB-Bayer sampled images.		`gmic geo2.png -rgb2bayer 0,0 -o rgb2bayer.png`	image en niveaux de gris, 2 couleurs 
-bayer2rgb _GM_smoothness, _RB_smoothness1, _RB_smoothness2 Transform selected RGB-Bayer sampled images to color images.		`gmic rgb2bayer.png -bayer2rgb 5,5,5 -o bayer2rgb.png`	
-lic _amplitude>0, _channels>0 Generate LIC representation of vector field.		`gmic geo2.png -lic 100,1 -o lic.png`	
-gaussian _sigma1[%], _sigma2[%], _angle Draw a centered gaussian on selected images, with specified standard deviations and orientation.	sans	`gmic 256,256,1,1 -gaussian 30%,30%,0 -n 0,255 -o gaussian.png`	
-function1d 0<=smoothness<=1,x0,y0,x1,y1,...,xn,yn Generate continuous 1d function from specified list of keypoints (xk,yk) in range [0,max(xk)] (xk are integers).	`La commande '-function1d' permet de générer une fonction lisse 1d à partir de points clés. Par exemple :` `gmic -function1d 1,0,10,30,50,70,-20,100,20 -plot`		
-pointcloud Convert a Nx1, Nx2 or Nx3 image as a point cloud in a 1d/2d or 3d binary image.	.	Voir la fonction `sierpinski3d` du fichier gmic_def.1442	.

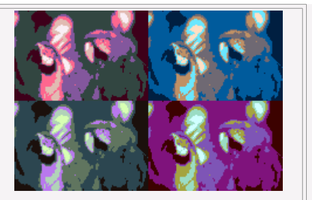
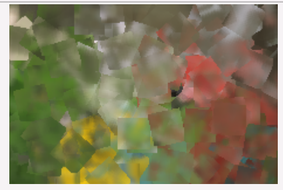
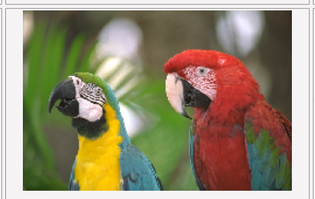
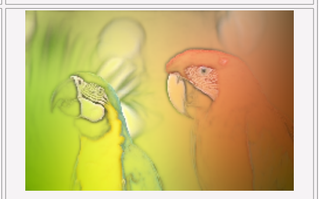
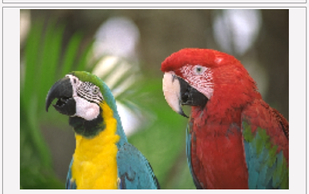
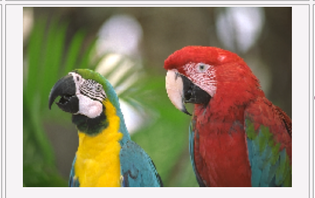
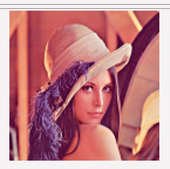
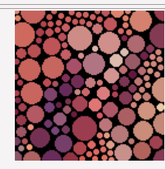
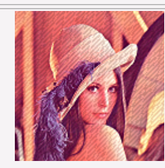
-snowflake _recursion>=0, _x0, _y0, _x1, _y1, _x2, _y2, _opacity, _col1,..._colN Draw a Koch snowflake on selected images.		`gmic geo2.png -snowflake 5,20,20,64,64,107,107,1 -o[-1] snowflake.png`	
- maze _width>0,_height>0,_cell_size>0 Generate maze with specified size.	sans	`gmic -maze 16,16,8 -n 0,255 -o maze.png`	

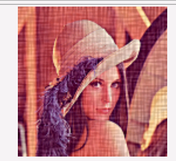
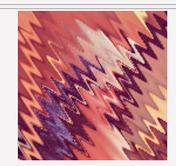
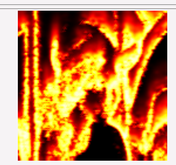
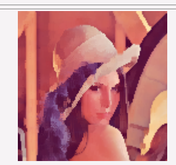
Effets artistiques

Paramètres de la ligne de commande (aide de G'MIC)	Image(s) d'origine	Ligne de commande (`gmic image -[filtre] -o resultat.png`)	Résultat
-polaroid _size1>=0, _size2>=0 Create polaroid effect in selected images.		`-polaroid 10,20`	
-drop_shadow _offset_x[%], _offset_y[%], _smoothness[%] Drop shadow behind selected images.		`-drop_shadow 6,6,3`	
-tetris _scale>0 Apply tetris effect on selected images.		`-tetris 9`	
-mosaic _density>=0, _edges={ 0 | 1 } Create random mosaic from selected images.		`-mosaic 5,1`	
-puzzle _scale>=0 Apply puzzle effect on selected images.		`-puzzle 9`	

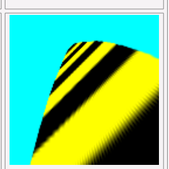
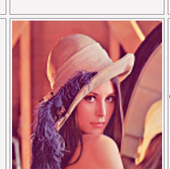
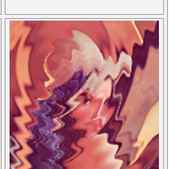
		`-puzzle 3`	
-sponge _size>0 Apply sponge effect on selected images.		`-sponge 12`	
-hearts _density>=0 Apply heart effect on selected images.		`-hearts 10`	
-color_ellipses _count>0, _radius>=0, _opacity>=0 Add random color ellipses to selected images.		`-color_ellipses 100,10,1`	
		`-color_ellipses 15,20,0.4`	
-ellipsionism _R>0[%], _r>0[%], _smoothness>=0[%], _opacity, _outline>0, _density>0 Apply ellipsionism filter to selected images.		`-ellipsionism 30,10,0.5,0.7,3,0.1`	
-whirls _texture>=0, _smoothness>=0, _darkness>=0, _lightness>=0 Add random whirl texture to selected images.		`-whirls 7,2,0.16,1.8`	
-cartoon _smoothness, _sharpening, _threshold>=0, _thickness>=0, _color>=0,quantization>0 Apply cartoon effect on selected images.		`-cartoon 1.3,140,30,0.15,2.6,10`	

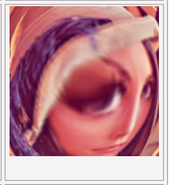
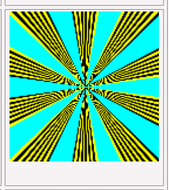
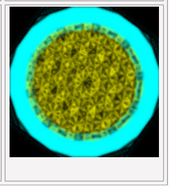
-drawing _amplitude>=0 Apply drawing effect on selected images.		`-drawing 80`	
-draw_whirl _amplitude>=0 Apply whirl drawing effect on selected images.		`-draw_whirl 80`	
-stencil _radius[%]>=0, _smoothness>=0, _iterations>=0 Apply stencil filter on selected images.		`-stencil 3,6,66`	
-stencilbw _edges>=0, _smoothness>=0 Apply B&W stencil effect on selected images.		`-stencilbw 22.7,11.5`	
-pencilbw _size>=0, _amplitude>=0 Apply B&W pencil effect on selected images.		`-pencilbw 3.1,10`	
-sketchbw _nb_orients>0, _start_angle, _angle_range>=0, _length>=0, _threshold>=0, _opacity, _bgfactor>=0, _density>0, _sharpness>=0, _anisotropy>=0, _smoothness>=0, _coherence>=0, _is_boost={ 0 | 1 }, _is_curved={ 0 | 1 } Apply sketch effect to selected images.		`-sketchbw 4,45,180,30,0.86,0.03,0.05,0.6,0.1,0.6,0,0.73`	
-ditheredbw Create dithered B&W version of selected images.		`-ditheredbw`	
-dotsbw _nb_scales>=0, _resolution>0, _radius>=0 Apply B&W dots effect on selected images.		`gmic perroquets.png -luminance -dotsbw 10,26.5,2.25 -* 255 -o[-1] dotsbw.png`	

-warhol _M>0, _N>0, _smoothness>=0, _color>=0 Create MxN Andy Warhol-like artwork from selected images.		`-warhol 2,2,1,22`	
-cubism _nb_iter>=0, _bloc_size>0, _max_angle, _opacity, _smoothness>=0 Apply cubism effect on selected images.		`-cubism 207,11.61,199.74,0.8,1.55`	
-glow _amplitude>=0 Add soft glow on selected images.		`-glow 50`	
-old_photo Apply old photo effect on selected images.		`-old_photo`	
-rodilius 0<=_amplitude<=100,_0<=thickness<=100, _sharpness>=0,_nb_orientations>0, _offset,_color_mode={ 0=darker | 1=brighter} Apply rodilius (fractalius-like) filter on selected images.		`gmic perroquets.png -gimp_rodilius 14,10,300,5,30,1,0 -o rodilius.png`	
-truchet : _scale>0,_radius>=0,_pattern_type={ 0=straight | 1=curved } Fill selected images with random truchet patterns.	sans	`gmic 128,128,1,3 -truchet 16,3,1 -n 0,225 -o truchet.png`	
		`gmic 128,128,1,3 -truchet 16,3,0 -n 0,255 -o truchet2.png`	
-circlism _radius_min>0,_radius_max>0,_smoothness[%]>=0,_radius_linearity>=0,_location_linearity>=0 Apply circlism effect on selected images (effect inspired by Ben Heine).		`gmic geo.png -circlism 2,10,8,4,4 -o circlism.png`	
-texturize_paper Add paper texture to selected images.		`gmic geo.png -texturize_paper -o texturize_paper.png`	

-texturize_canvas _amplitude>=0,_fibrousness>=0,_emboss_level>=0 Add paint canvas texture to selected images.		`gmic geo.png -texturize_canvas 30,4,0.5 -o texturize_canvas.png`	
-ripple _amplitude,_frequency,_shape={ 0=bloc | 1=triangle | 2=sine | 3=sine+ | 4=random }, _angle,_offset Apply ripple deformation on selected images.		`gmic geo.png -ripple 15,10,2,45,0 -o ripple.png`	
-fire_edges _edges>=0,0<=_attenuation<=1,_smoothness>=0,_threshold>=0 ,_nb_frames>0,_starting_frame>=0,frame_skip>=0 Generate fire effect from edges of selected images.		`gmic geo.png -fire_edges 0.5,0.1,0.6 -o fire_edges.png`	
-kuwahara size>0 Apply Kuwahara filter of specified size on selected images.		`gmic geo.png -kuwahara 3 -o kuwahara.png`	

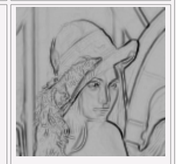
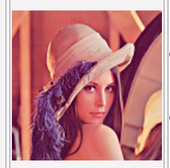
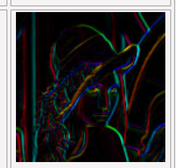
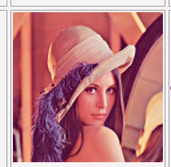
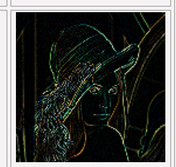
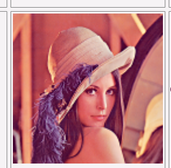
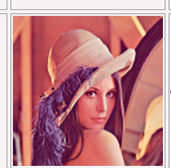
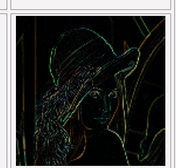
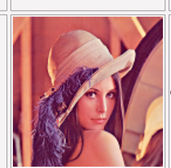
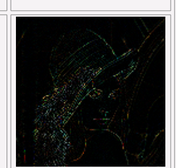
Déformation spatiale

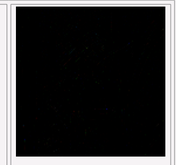
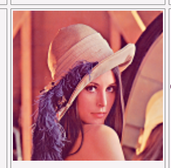
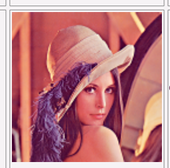
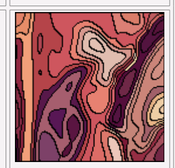
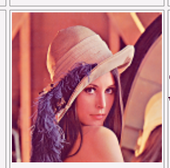
Paramètres de la ligne de commande (aide de G'MIC)	Image(s) d'origine	Ligne de commande	Résultat
-euclidean2polar _cx, _cx, _n>0, _borders={ 0=dirichlet | 1=neumann | 2=cyclic } Apply euclidean to polar transform on selected images.		`gmic geo2.png -euclidean2polar 0.5,0.5,1,1 -o euclidean2polar.png`	
-polar2euclidean _cx, _cy, _n>0, _borders={ 0=dirichlet | 1=neumann | 2=cyclic } Apply polar to euclidean transform on selected images.		`gmic geo2.png -polar2euclidean 0.5,0.5,1,1 -o polar2euclidean.png`	
-warp_perspective _x-angle, _y-angle, _zoom>0, _x-center, _y-center, _borders={ 0=dirichlet | 1=neumann | 2=cyclic } Warp selected images with perspective deformation.		`gmic geo2.png -warp_perspective 2,2,0.5,50,20,1 -o warp_perspective.png`	
-water _amplitude>=0, _smoothness>=0 Apply water deformation on selected images.		`gmic geo2.png -water 25,1.2 -o water.png`	
-wave _amplitude>=0, _frequency>=0, _center_x, _center_y Apply wave deformation on selected images.		`gmic geo.png -wave 3,0.5,30,30 -o wave.png`	

-twirl _amplitude, _cx, _cy, _borders={ 0=dirichlet | 1=neumann | 2=cyclic } Apply twirl deformation on selected images.		`gmic geo2.png -twirl 1.5,0.5,0.5,1 -o twirl.png`	
-map_sphere _width>0, _height>0, _radius, _dilation>0 Map selected images on a sphere.		`gmic geo.png -map_sphere 128,128,100,0.7 -o map_sphere.png`	
-flower _amplitude, _frequency, _offset_r[%], _angle, _cx, _cy, _borders={ 0=dirichlet | 1=neumann | 2=cyclic } Apply flower deformation on selected images.		`gmic geo2.png -flower 30,6,0,0,0.5,0.5,1 -o flower.png`	
-zoom _factor, _cx, _cy, _cz, _borders={ 0=dirichlet | 1=neumann | 2=cyclic } Apply zoom factor to selected images.		`gmic geo2.png -zoom 1.3,0.5,0.5,0,0 -o zoom.png`	
-deform _amplitude>=0 Apply random smooth deformation on selected images.		`gmic geo2.png -deform 15 -o deform.png`	
-fisheye _x, _y,0<=_radius<=100, _amplitude>=0 Apply fish-eye deformation on selected images.		`gmic geo.png -fisheye 50,50,100,2 -o fish_eye.png`	
-transform_polar "expr_radius", _"expr_angle", _x_center, _y_center, _borders={ 0 | 1 } Apply user-defined transform on polar representation of selected images.		`gmic geo2.png c=50 a=$c+10 b=60*cos(a*5) -transform_polar $b,$a,$c,$c -o transform_polar.png`	
-kaleidoscope _cx, _cy, _radius, _angle, _borders={ 0=dirichlet | 1=neumann | 2=cyclic } Create kaleidoscope effect from selected images.		`gmic geo2.png -kaleidoscope 0.5,0.5,30,8,1 -o kaleidoscope.png`	
-rotoidoscope _cx, _cy, _tiles>0, _smoothness[%]>=0, _borders={ 0=dirichlet | 1=neumann | 2=cyclic} Create rotational kaleidoscope effect from selected images.		`gmic geo2.png -rotoidoscope 50%,50%,10,0.1,0 -o rotoidoscope.png`	

Contours

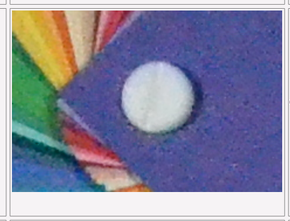
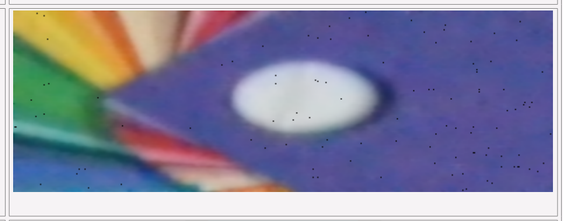
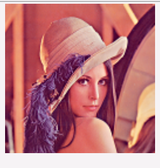
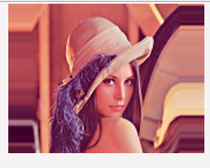
Paramètres de la ligne de commande (aide de G'MIC)	Image(s) d'origine	Ligne de commande	Résultat

-gradient_norm Compute gradient norm of selected images.		`gmic geo.png -gradient_norm -c 0,255 -o gradient_norm.png`	
		`gmic geo.png -gradient_norm -c 0,255 -negative -o gradient_norm_negative.png`	
-gradient_orientation _dimension={1,2,3} Compute N-D gradient orientation of selected images.		`gmic geo.png -gradient_orientation 1 -c 0,255 -o gradient_orientation.png` `gmic geo.png -gradient_orientation 1 -c 0,255 -n 0,255 -o gradient_orientation_n.png`	 Version n 0,255 
-gradient2rgb _orientation={ 0 | 1 } Compute RGB representation of 2d gradient of selected images.		`gmic geo.png -gradient2rgb 0 -o gradient2rgb.png`	
		`gmic geo.png -gradient2rgb 0 -negative -o gradient2rgb_negative.png`	
-laplacian Compute Laplacian of selected images.		`gmic geo.png -laplacian -c 0,255 -o laplacian.png`	
-divergence Compute divergence of selected vector fields.		`gmic geo.png -divergence -c 0,255 -o divergence.png`	
-Inn Compute gradient-directed 2nd derivative of image(s).		`gmic geo.png -Inn -c 0,255 -o inn.png`	
-Iee Compute gradient-orthogonal-directed 2nd derivative of image(s).		`gmic geo.png -Iee -c 0,255 -o iee.png`	

-curvature Compute isophote curvatures on selected images.		gmic geo.png -curvature -c 0,255 -o curvature.png	
-edges _threshold[%]>=0 Estimate contours of selected images.		gmic geo.png -edges 19% -n 0,255 -o edges.png	
-isophotes _nb_levels>0 Render isophotes of selected images on a transparent background.		gmic geo.png -isophotes 5 -o isophotes.png	
-topographic_map _nb_levels>0, _smoothness Render selected images as topographic maps.		gmic geo.png -topographic_map 20,5 -o topographic_map.png	
-segment_watershed _threshold>=0,_keep_watershed={ 0 | 1 } ~~_threshold>=0, _edge_threshold>0, _keep_watershed={ 0 | 1 }~~ Apply watershed segmentation on selected images.		gmic geo.png -segment_watershed 30,1 -n 0,255 -o segment_watershed.png ~~gmic geo.png -segment_watershed 30,10,1 -n 0,255 -o segment_watershed.png~~	

Manipulations géométriques

Paramètres de la ligne de commande (aide de G'MIC)	Image(s) d'origine	Ligne de commande	Résultat
-split_tiles M!=0, _N!=0, _is_homogeneous={ 0 | 1 } Split selected images as a MxN array of tiles.		Découpage de l'image en 4 parties (2*2) gmic geo2.png -split_tiles 2,2 -o split_tiles.png	image 1  image 2  image 3  image 4 
-append_tiles M>0, _N>0 Append MxN selected tiles as a new image.	image 1  image 2 	Assemblage de 2 images gmic split_tiles_000000.png split_tiles_000001.png -append_tiles 2,1 -o append_tiles.png	

-rr2d eq. to '-resize_ratio2d'. -resize_ratio2d width>0,height>0, _mode={ 0=inside | 1=outside | 2=padded },0=<_interpolation_type<=6 Resize selected images while preserving their aspect ratio.		`gmic geo2.png -resize_ratio2d 80,80,1,2 -o resize_ratio2d.png`	
-r2dx eq. to '-resize2dx'. -resize2dx width>0, _interpolation_type={0,1,2,3,4,5} Resize selected images along the X-axis, preserving 2d ratio. (eq. to '-r2dx').		`gmic geo2.png --resize2dx 60,1 -o[-1] resize2dx.png`	
-r3dx eq. to '-resize3dx'. -resize3dx width>0, _interpolation_type={0,1,2,3,4,5} Resize selected images along the X-axis, preserving 3d ratio. (eq. to '-r3dx').		`gmic geo2.png -resize3dx 60,1 -o[-1] resize3dx.png`	
-r2dy eq. to '-resize2dy'. -resize2dy height>0, _interpolation_type={0,1,2,3,4,5} Resize selected images along the Y-axis, preserving 2d ratio. (eq. to '-r2dy').		`gmic geo2.png -resize2dy 40,3 -o[-1] resize2dy.png`	
-r3dy eq. to '-resize3dy'. -resize3dy height>0, _interpolation_type={0,1,2,3,4,5} Resize selected images along the Y-axis, preserving 3d ratio. (eq. to '-r3dy').		`gmic geo2.png -resize3dy 40,3 -o[-1] resize3dy.png`	
-r3dz eq. to '-resize3dz'. -resize3dz depth>0, _interpolation_type={0,1,2,3,4,5} Resize selected images along the Z-axis, preserving 3d ratio. (eq. to '-r3dz').	objet 3d	.	.
-upscale_smart width,height, _depth,smoothness>=0, _anisotropy=[0,1],sharpening>=0 Upscale selected images with an edge-preserving algorithm.		`gmic bruit.png -upscale_smart 200%,100%,4,8,0.1 -o upscale_smart.png`	
-expand_x size_x>=0, _borders={ 0=dirichlet | 1=neumann | 2=cyclic } Expand selected images along the X-axis.		`gmic geo2.png -expand_x 20,0 -o expand_x.png`	
		`gmic geo.png -expand_x 20,1 -o expand_x2.png`	

-expand_y size_y>=0, _borders={ 0=dirichlet | 1=neumann | 2=cyclic } Expand selected images along the Y-axis.		`gmic geo2.png -expand_y 20,0 -o expand_y.png`	
-expand_z size_z>=0, _borders={ 0=dirichlet | 1=neumann | 2=cyclic } Expand selected images along the Z-axis.	objet 3d	.	.
-expand_xy size>=0, _borders={ 0=dirichlet | 1=neumann | 2=cyclic } Expand selected images along the XY-axes.		`gmic geo2.png -expand_xy 20,0 -o expand_xy.png`	
-expand_xyz size>=0, _borders={ 0=dirichlet | 1=neumann | 2=cyclic } Expand selected images along the XYZ-axes.	objet 3d	.	.
-shrink_x size_x>=0 Shrink selected images along the X-axis.		`gmic geo2.png -shrink_x 40 -o shrink_x.png`	
-shrink_y size_y>=0 Shrink selected images along the Y-axis.		`gmic geo2.png -shrink_y 40 -o shrink_y.png`	
-shrink_z size_z>=0 Shrink selected images along the Z-axis.	objet 3d	.	.
-shrink_xy size>=0 Shrink selected images along the XY-axes.		`gmic geo2.png -shrink_xy 40 -o shrink_xy.png`	
-elevate _depth, _is_plain, _is_colored Elevate selected 2d images into 3d volumes.	.	`gmic geo.png -elevate 20,1,1`	?

Entrées/Sorties

Paramètres de la ligne de commande (aide de G'MIC)	Image(s) d'origine	Ligne de commande	Résultat

-input filename | [image]x_nb_copies>0 | { width>0[%] | [image_w] },{ _height>0[%] | [image_h] },{ _depth>0[%] | [image_d] }, { _spectrum>0[%] | [image_s] }, _value1, _value2,.. | (value1{,|;|/|^}value2{,|;|/|^}..) Insert a new image taken from a filename or from a copy of an existing image ['indice'], or insert new image with specified dimensions and values. (eq. to '-i' | (no args)).		gmic geo2.png -i 50%,50% -fill_color[-1] 127 -o insert.png	
-output filename, _format_options Output selected images as one or several numbered file(s). (eq. to '-o').	.	Les exemples utilisent cette fonction pour obtenir le résultat.	.
-verbose level | { + | - } Set or increment/decrement the verbosity level. (eq. to '-v'). When 'level'>=0, G'MIC log messages are displayed on the standard output. Default value for the verbosity level is 0.	.	Rendre moins bavard : gmic -v - geo2.png	C:\gmic_1.4.5.0_win32>gmic -v - geo2.png
-print Output informations on selected images, on the standard output.	.	Affichage des caractéristiques de l'image sans afficher l'image : gmic geo2.png -print	.
-echo message Output specified message, on the standard output. (eq. to '-e'). Command subset (if any) stands for displayed scope indices instead of image indices.	.	Envoyer un message à la console : gmic toto=Bonjour -e $toto	C:\gmic-1.4.4.2_win32>gmic toto=Bonjour -e $toto [gmic]-0./ Start G'MIC instance. [gmic]-0./ Push labelled item toto='Bonjour' on the local stack, at position 0. [gmic]-0./ Bonjour [gmic]-0./ End G'MIC instance.
-warning message Print specified warning message, on the standard output. Command subset (if any) stands for displayed scope indices instead of image indices.	.	Envoyer un message à la console : gmic toto=Attention\ aux\ orignaux -warning $toto	C:\gmic-1.4.4.2_win32>gmic toto=Attention\ aux\ orignaux -warning $toto [gmic]-0./ Start G'MIC instance. [gmic]-0./ Push labelled item toto='Attention aux orignaux' on the local stack, at position 0. [gmic]-0./ *** Warning in ./ *** Attention aux orignaux [gmic]-0./ End G'MIC instance.
-command filename | "string" Import G'MIC custom command(s) from specified file or string. (eq. to '-m'). Imported commands are available directly after the '-command' invokation.	.	gmic -m test1.txt -fonction_test1	.
-type datatype Set pixel datatype for all images of the list. 'datatype' can be { bool | uchar | char | ushort | short | uint | int | float | double }.	.	2 exemples pour convertir des images en 8 bits ou 16 bits (source http://gmic.sourceforge.net/tutorial.shtml) : #@gmic write16 : '"outfile"' : Write 16 bit image to file. write16 : -e "Writing image$?." -v - -repeat @# -c 0,65536 -type ushort -o $1 -mv[-1] 0 -done -v + #@gmic write8 : '"outfile"' : Write 8 bit image to file. write8 : -e "Writing image$?." -v - -repeat @# -c 0,255 -type uchar -o $1 -mv[-1] 0 -done -v +	.

<table>
<tr>
<td>-shell

Start interactive shell environment, with selected images.</td>
<td>  </td>
<td>Démarre le mode interactif de la console.

Pour en sortir :

-q
-quit

Appeler une fonction "greffon de Gimp" via le "shell" (effet sponge) :

C:\gmic-1.4.4.2_win32>gmic geo.png -shell
[gmic]-0./ Start G'MIC instance.
[gmic]-0./ Input file 'geo.png' at position [0] (1 image 128x128x1x3).
[gmic]-1./ Start interactive shell, with image [0].
[gmic]-1./*>> -gimp_sponge 12,2
[gmic]-1./*>/gimp_sponge/apply_channels/ Apply command '-sponge 12' on RGB channels of image [0].
[gmic]-1./*>> -o shell_sponge.png
[gmic]-1./*>/ Output image [0] as file 'shell_sponge.png' (1 image 128x128x1x3).
[gmic]-1./*>> -q
[gmic]-1./*>/ Quit G'MIC instance.</td>
<td>C:\gmic-1.4.4.2_win32>gmic geo.png -shell
[gmic]-0./ Start G'MIC instance.
[gmic]-0./ Input file 'geo.png' at position [0] (1 image 128x128x1x3).
[gmic]-1./ Start interactive shell, with image [0].
[gmic]-1./*>> -noise 100,0
[gmic]-1./*>/ Add gaussian noise to image [0], with standard deviation 100.
[gmic]-1./*>> -q
[gmic]-1./*>/ Quit G'MIC instance.

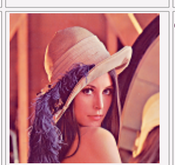
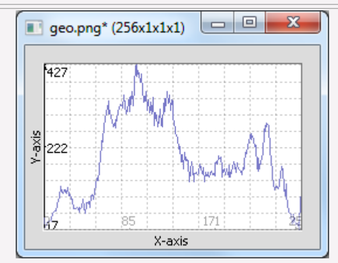
  </td>
</tr>
<tr>
<td>-shared x0[%],x1[%],y[%],z[%],v[%] |
y0[%],y1[%],z[%],v[%] |
z0[%],z1[%],v[%] |
v0[%],v1[%] |
(no args)

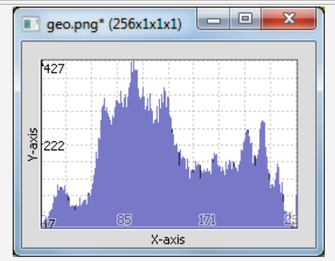
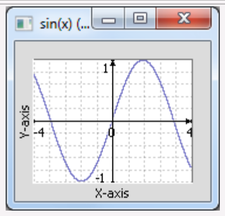
Insert shared buffers from (opt. points/lines/planes/channels of) selected images.
(eq. to '-sh').</td>
<td>.</td>
<td>gmic geo.png -shared 1,1</td>
<td>.</td>
</tr>
<tr>
<td>-camera _camera_index>=-1, _nb_frames>0, _skip_frames>=0,release_camera={ 0 | 1 }

Insert one or several frames from specified camera, with custom delay between frames (in ms).
Set 'camera_index' to -1 to use the default camera device.
When 'release_camera' is set to 1, the camera stream is released and no images are inserted.</td>
<td>.</td>
<td>non testé</td>
<td>.</td>
</tr>
<tr>
<td>-display

Display selected images in an interactive viewer (use the instant window [0] if opened).
(eq. to '-d').</td>
<td>.</td>
<td>gmic 128,128,1,3 -fill_color 255,255,0 -display -noise 100,1 -display</td>
<td>.</td>
</tr>
<tr>
<td>-display3d

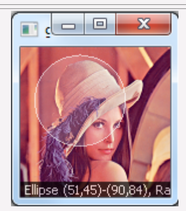
Display selected 3d objects in an interactive viewer (use the instant window [0] if opened).
(eq. to '-d3d').</td>
<td>.</td>
<td>gmic -sphere3d 200,1 -display3d -box3d 200,200,200 -display3d[-1]</td>
<td>.</td>
</tr>
<tr>
<td>-plot _plot_type, _vertex_type, _xmin, _xmax, _ymin, _ymax |
'formula', _resolution>=0, _plot_type, _vertex_type, _xmin,xmax, _ymin, _ymax

Display selected image or formula in an interactive viewer (use the instant window [0] if
opened).
'plot_type' can be { 0=none | 1=lines | 2=splines | 3=bar }.
'vertex_type' can be { 0=none | 1=points | 2,3=crosses | 4,5=circles | 6,7=squares }.
'xmin','xmax','ymin','ymax' set the coordinates of the</td>
<td>  </td>
<td>gmic geo.png -histogram 256 -plot</td>
<td> 
</td>
</tr>
</table>

<table>
<tr><td rowspan="2">displayed xy-axes.</td><td></td><td>gmic geo.png -histogram 256 -plot 3</td><td> 
</td></tr>
<tr><td>sans</td><td>gmic -plot 'sin(x)'</td><td> 
</td></tr>
<tr><td>-window _width[%]>=-1, _height[%]>=-1, _normalization, _fullscreen, _title

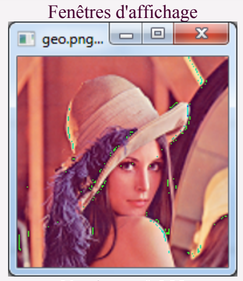
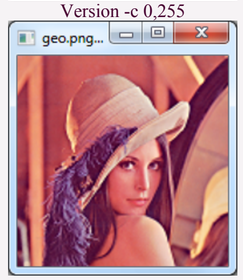
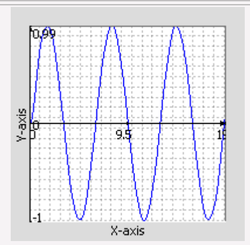
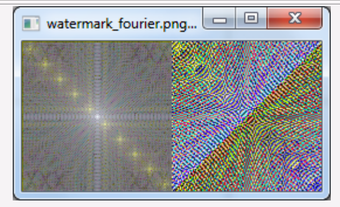
Display selected images into an instant window with specified size, normalization type,
fullscreen mode and title.
(eq. to '-w').
If 'width' or 'height' is set to -1, the corresponding dimension is adjusted to the window
or image size.
'width'=0 or 'height'=0 closes the instant window.
'normalization' can be { -1=keep same | 0=none | 1=always | 2=1st-time | 3=auto }.
'fullscreen' can be { -1=keep same | 0=no | 1=yes }.
You can manage up to 10 different instant windows by using the numbered variants
'-w0' (default, eq. to '-w'),'-w1',..,'-w9' of the command '-w'.</td><td>.</td><td>gmic geo.png -w 800,800 -wait 2000</td><td>.</td></tr>
<tr><td>-wait delay |
(no args)

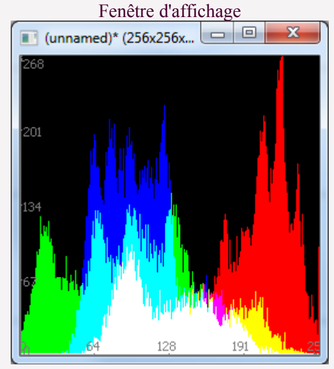
Wait for a given delay (in ms) or for an user event occuring on the selected instant window.
'delay' can be { <0=delay+flush | 0=event | >0=delay }.
Command subset (if any) stands for instant window indices instead of image indices.</td><td>.</td><td>gmic geo.png -w 800,800 -wait 2000</td><td>.</td></tr>
<tr><td>-select feature

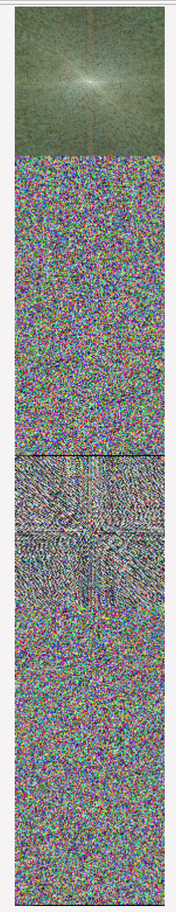
Interactively select a feature from selected images (use the instant window [0] if opened).
'feature' can be { 0=point | 1=segment | 2=rectangle | 3=ellipse }.
The retrieved feature is returned as a vector containing the feature coordinates.</td><td>.</td><td>gmic geo.png -select 3</td><td> 
</td></tr>
<tr><td>-ow eq. to '-outputw'.
-outputw

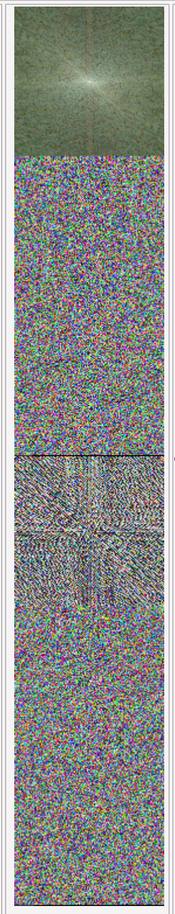
Output selected images by overwritting their original location.
(eq. to '-ow').</td><td>image outputw.png
identique à geo2.png</td><td>Écrase le contenu de l'image d'origine par les effets appliqués :
gmic outputw.png -ellipse 50%,50%,30,40,45,1,255,0,0 -outputw</td><td>  </td></tr>
<tr><td>-op eq. to '-outputp'.
-outputp prefix

Output selected images as prefixed versions of their original filenames.
(eq. to '-op').</td><td>geo2.png
  </td><td>Ajoute un préfixe au nom de l'image :
gmic geo2.png -ellipse 50%,50%,30,40,90,1,255,0,255 -outputp op_</td><td>op_geo2.png
  </td></tr>
</table>

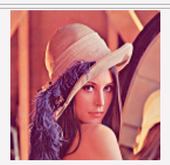
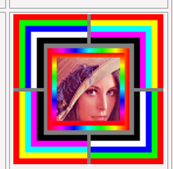
-on eq. to '-outputn'. -outputn filename Output selected images as automatically numbered filenames in repeat..done loops. (eq. to '-on').	.	Indexe automatiquement les noms d'images dans des boucles.	.
-d0 eq. to '-display0'. -display0 Display selected images without value normalization. (eq. to '-d0').		gmic geo.png -resize 180,180,1,3,5 -display0 version normalisée avec -c 0,255 gmic geo.png -resize 180,180,1,3,5 -c 0,255 -display0	Fenêtres d'affichage  Version -c 0,255 
-dg eq. to '-display_graph'. -display_graph _width>32, _height>32, _plot_type, _vertex_type, _xmin, _xmax, _ymin, _ymax Plot type = "None","Lines","Splines","Bars" Vertex type = "None","Points","Crosses 1","Crosses 2","Circles 1","Circles 2","Square 1","Square 2"	sans	gmic 20,1,1,3,"X=sin(x)" -dg 200,200,2,0 -o display_graph.png	
-dfft eq. to '-display_fft'. -display_fft Display fourier transform of selected images, with centered log-module and argument. (eq. to '-dfft').		gmic geo.png -display_fft	Fenêtres d'affichage 
	watermark_fourier.png 	Affichage du XXX appliqué par le filtre "watermark_fourier" gmic watermark_fourier.png -display_fft	
-drgba eq. to '-display_rgba'. -display_rgba Render selected RGBA images over a checkerboard background. (eq. to '-drgba').		gmic sprite.png -display_rgba	Fenêtre d'affichage 

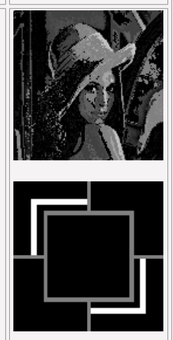
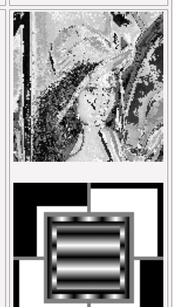
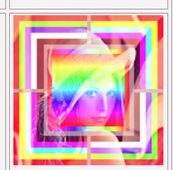
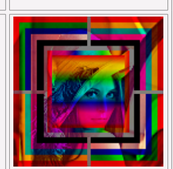
-dh eq. to '-display_histogram'. -display_histogram _width>0, _height>0, _clusters>0, _max_value>0, _show_axes={ 0 | 1 } Render a channel-by-channel histogram. (eq. to '-dh').		gmic geo.png -display_histogram 256,256	Fenêtre d'affichage 
-dt eq. to '-display_tensors'. -display_tensors _size_factor>0, _ellipse_factor>=0, _opacity, _pattern, _color1,.. Render field of 2x2 tensors with ellipses. (eq. to '-dt').		gmic geo.png -display_tensors	Affichage réduit 
-float2int8 Convert selected float-valued images to 8bits integer representations.	.	Conversion non testée gmic geo.png -resize 180,180,1,3,5 -float2int8 -display0	.
-int82float Convert selected 8bits integer representations to float-valued images.	.	Conversion non testée	.

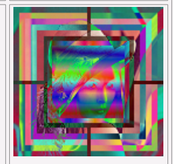
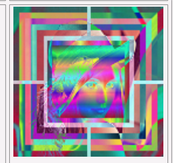
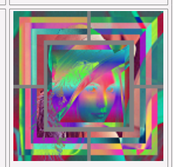
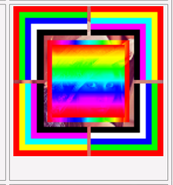
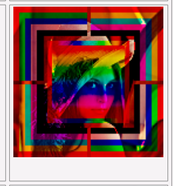
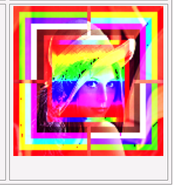
-float2fft8 Convert selected float-valued images to 8bits fourier representations.		gmic geo.png -float2fft8 -o geo2fft.png	

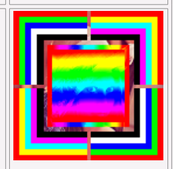
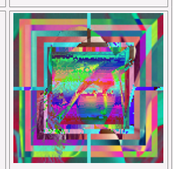
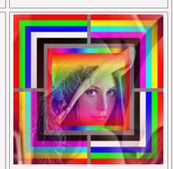
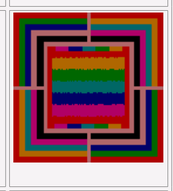
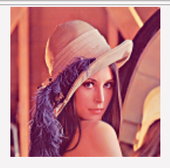
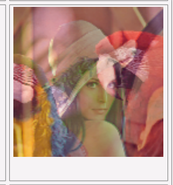
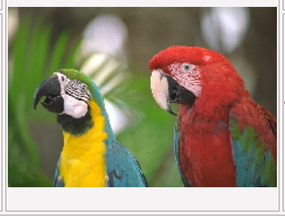
-fft82float Convert selected 8bits fourier representations to float-valued images.		gmic geo2fft.png -fft82float -c 0,255 -o fft2geo.png	
-apply_camera _command, _camera_index>=-1, _skip_frames>=0 Apply specified command on live camera stream, and display it on display window [0].	.	non testé	.
-rainbow_lut Generate a 256-entries RGB colormap of rainbow colors.	sans	gmic -rainbow_lut -o rainbow_lut.png	Image étirée en hauteur (100px) 

Mélanges d'images

Paramètres de la ligne de commande (aide de G'MIC)	Image(s) d'origine	Ligne de commande	Résultat
-compose_rgba Compose selected RGBA images two-by-two, over RGB background.		gmic geo.png mire.png -compose_rgba -o compose_rgba.png	

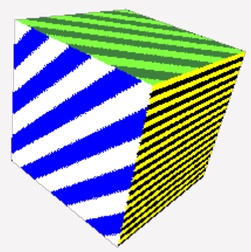
-compose_channels Compose all channels of each selected image, using specified arithmetic operator (+,-,or,min,...).	mire avec la partie centrale transparente 	gmic geo.png mire.png -compose_channels - -o compose_channels_moins.png	
		gmic geo.png mire.png -compose_channels and -o compose_channels_and.png	
		gmic geo.png mire.png -compose_channels xor -o compose_channels_xor.png	
-compose_average Compose selected images two-by-two, using average mode.		gmic geo.png mire.png -compose_average -o compose_average.png	
-compose_multiply Compose selected images two-by-two, using multiply mode.		gmic geo.png mire.png -compose_multiply -o compose_multiply.png	
-compose_screen Compose selected images two-by-two, using screen mode.		gmic geo.png mire.png -compose_screen -o compose_screen.png	
-compose_darken Compose selected images two-by-two, using darken mode.		gmic geo.png mire.png -compose_darken -o compose_darken.png	

-compose_lighten Compose selected images two-by-two, using lighten mode.		`gmic geo.png mire.png -compose_lighten -o compose_lighten.png`	
-compose_difference Compose selected images two-by-two, using difference mode.		`gmic geo.png mire.png -compose_difference -o compose_difference.png`	
-compose_negation Compose selected images two-by-two, using negation mode.		`gmic geo.png mire.png -compose_negation -o compose_negation.png`	
-compose_exclusion Compose selected images two-by-two, using exclusion mode.		`gmic geo.png mire.png -compose_exclusion -o compose_exclusion.png`	
-compose_overlay Compose selected images two-by-two, using overlay mode.		`gmic geo.png mire.png -compose_overlay -o compose_overlay.png`	
-compose_hardlight Compose selected images two-by-two, using hard light mode		`gmic geo.png mire.png -compose_hardlight -o compose_hardlight.png`	
-compose_softlight Compose selected images two-by-two, using soft light mode.		`gmic geo.png mire.png -compose_softlight -o compose_softlight.png`	
-compose_dodge Compose selected images two-by-two, using dodge mode.		`gmic geo.png mire.png -compose_dodge -o compose_dodge.png`	
-compose_colorburn Compose selected images two-by-two, using color burn mode.		`gmic geo.png mire.png -compose_colorburn -o compose_colorburn.png`	
-compose_reflect Compose selected images two-by-two, using reflect mode.		`gmic geo.png mire.png -compose_reflect -o compose_reflect.png`	

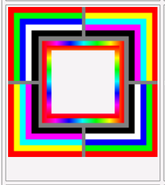
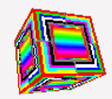
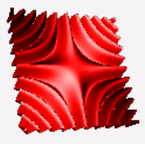
-compose_freeze Compose selected images two-by-two, using freeze mode.		`gmic geo.png mire.png -compose_freeze -o compose_freeze.png`	
-compose_stamp Compose selected images two-by-two, using stamp mode.		`gmic geo.png mire.png -compose_stamp -o compose_stamp.png`	
-compose_interpolation Compose selected images two-by-two, using interpolation mode.		`gmic geo.png mire.png -compose_interpolation -o compose_interpolation.png`	
-compose_xor Compose selected images two-by-two, using xor mode.		`gmic geo.png mire.png -compose_xor -o compose_xor.png`	
-compose_edges _smoothness[%]>=0 Compose selected images togethers using edge composition.		`gmic geo.png mire.png -compose_edges 10 -o compose_edges.png`	
-compose_fade Compose selected images togethers using a given fading (defined as the latest image).		`gmic geo.png mire.png -compose_fade -o compose_fade.png`	
-compose_shapeaverage Compose selected images two-by-two, using shape average mode.		`gmic geo.png mire.png -compose_shapeaverage -o compose_shapeaverage.png`	
-compose_median Compose selected images together using median mode.		`gmic geo.png perroquets.png -compose_median -o compose_median.png`	
-compose_divide Compose selected images two-by-two, using divide mode.		`gmic geo.png perroquets.png -compose_divide -o compose_divide.png`	

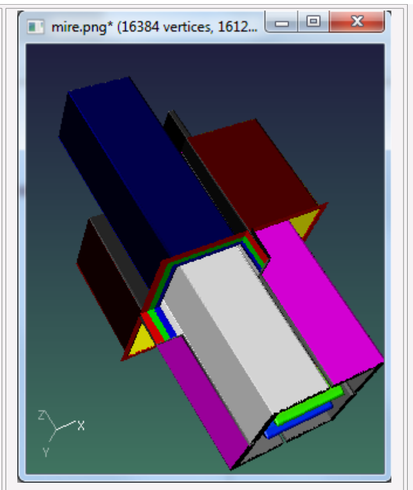
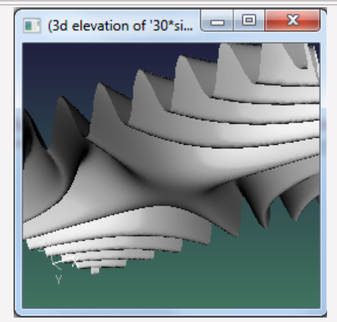
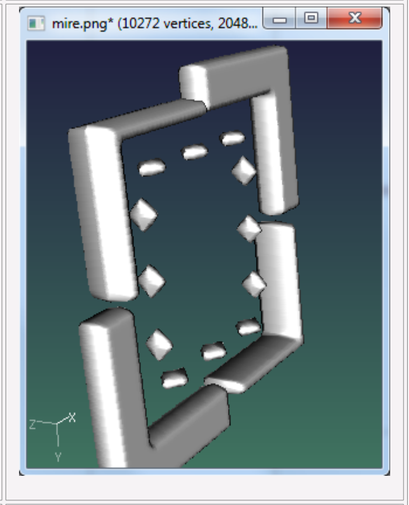
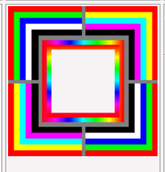
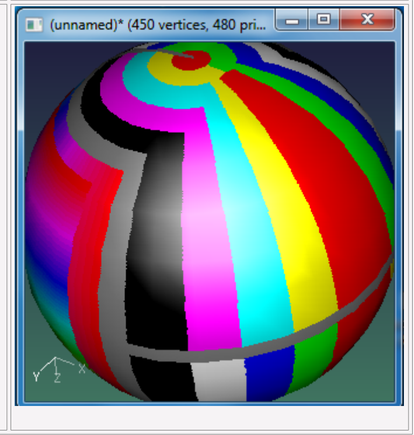
Quelques fonctions 3D

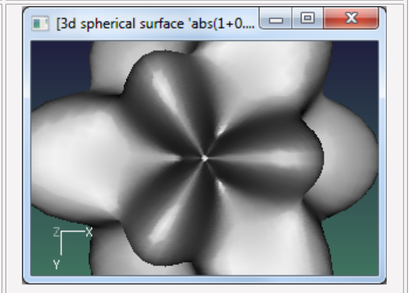
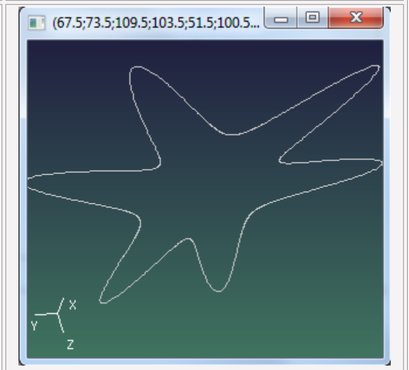
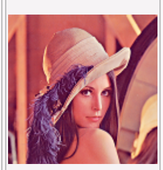
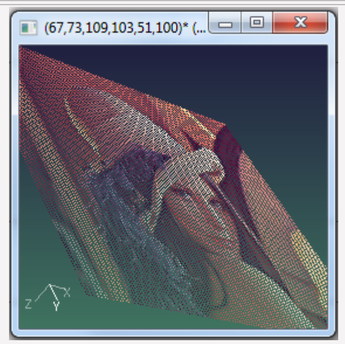
<table>
<tr><th>Paramètres de la ligne de commande (aide de G'MIC)</th><th>Image(s) d'origine</th><th>Ligne de commande</th><th>Résultat</th></tr>
<tr><td rowspan="3"></td><td>  </td><td>Voici une macro pour plaquer 6 images sur un cube :
cube6images.gmic

<pre>gmic -m cube6images.gmic 1.png 2.png 3.png 4.png 5.png 6.png -cube6images 256,256,0.5,160,40,12 -o cube6images.png</pre></td><td>  </td></tr>
<tr><td>Rendre un objet 3D sur une image

La fonction "gimp_render3d" permet de plaquer un objet 3D sur une image 2D.

Les paramètres de cette fonction -gimp_render3d sont dans l'ordre (les valeurs peuvent être modifiées) :
Width = _int(1024,8,4096)
Height = _int(1024,8,4096)
Object size = float(0.8,0,3)
X-angle = float(25,0,360)
Y-angle = float(0,0,360)
Z-angle = float(21,0,360)
FOV = float(45,1,90)
X-light = float(0,-100,100)
Y-light = float(0,-100,100)
Z-light = float(-100,-100,0)
Specularity = float(0.5,0,1)
Shininess = float(0.7,0,3)
Rendering mode = choice(2,"Dots","Wireframe","Flat","Flat shaded","Gouraud","Phong")</td><td>  </td><td><pre>gmic mire.png -imagecube3d -gimp_render3d 128,128,0.4,25,25,10,45,0,0,-100,0.5,0.7,2 -o rendu1.png</pre></td><td>  </td></tr>
<tr><td>sans</td><td><pre>gmic -elevation3d "'30*sin(x*y)^2'",-4,-4,4,4 -color3d 255,0,0,1 -gimp_render3d 256,256,0.4,25,25,10,45,0,0,-100,0.5,0.7,3 -o rendu2.png</pre></td><td>  </td></tr>
<tr><td>  </td><td><pre>gmic 1.png 2.png 3.png 4.png 5.png 6.png -image6cube3d -gimp_render3d 256,256,0.4,30,30,15,45,0,0,-100,0.5,0.7,3 -autocrop 0 -o rendu3.png</pre></td><td>  </td></tr>
</table>

-elevation3d z-factor | [image] | 'formula',_x0,_y0,_x1,y1,_dx[%],_dy[%] | (no args) Create 3d elevation of selected images or specified formula, with specified elevation map. If a z-factor is specified, each elevation map is computed as the pointwise L2 norm of the selected images. Else, elevation values are taken from the specified image or formula.		`gmic mire.png -elevation3d` Pour le rendu, voir "Rendre un objet 3D sur une image" ou gimp_elevation3d	
sans		Exemple donné par David Tschumperlé : `gmic -elevation3d "'30*sin(x*y)^2'",-4,-4,4,4` Pour le rendu, voir "Rendre un objet 3D sur une image"	
-extrude3d _depth>0,_resolution>0,_smoothness[%]>=0 Generated extruded 3d object from selected binary profiles.		`gmic mire.png -extrude3d` Pour le rendu, voir "Rendre un objet 3D sur une image" ou gimp_extrude3d	
-imagesphere3d -imagesphere3d _resolution1>=3,_resolution2>=3 Generate 3d mapped sphere from selected images.		`gmic mire.png -imagesphere3d` Voir sujet : http://www.flickr.com/groups/gmic/discuss/72157625597354886/	

		Exemple de rendu : `gmic mire.png -imagesphere3d 100,100 -gimp_render3d 300,300,0.4,25,25,10,45,0,0,-100,0.5,0.7,2 -autocrop 0 -o rendu_sphere3d.png`	
-spherical3d _nb_azimuth>=3,_nb_zenith>=3, _radius_function(phi,theta)	rien	`gmic -spherical3d 64,64,"abs(1+0.5*cos(3*phi)*sin(4*theta))"`	
		`gmic -spherical3d 64,64,"abs(1+0.5*cos(3*phi)*sin(4*theta))" -color3d 63,127,255,1 -gimp_render3d 256,256,0.4,25,25,10,45,0,0,-100,0.5,0.7,3 -o spherical3d.png`	
-superformula3d : resolution>1,m>=1,n1,n2,n3 Generate 2d superformula curve as a 3d object.	rien	`gmic -superformula3d 1024,6,2,4,16`	
-pointcloud3d Generate 3d point cloud from selected planar or volumetric images.		`gmic geo.png -n 0,255 -pointcloud3d`	

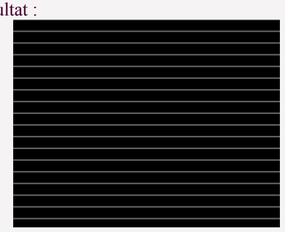
Quelques fonctions importantes

Récupérer les dimensions d'une image

Exemple : Récupérer les dimensions d'une image, puis créer une nouvelle image avec une largeur augmentée de 100 pixels (variable largeur) et une hauteur augmentée de 50pixels (variable hauteur) qui sera remplie de lignes horizontales grises tous les 10 pixels :

```
gmic geo2.png largeur={w+100} hauteur={h+50} $largeur,$hauteur,1,3 -fill[-1] "if(y%10==0,127,i)" -o[-1] lplus100_hplus50.png
```

Résultat :



Remarques au sujet des opérateurs et des variables w, h, y, i, etc. (aide de G'MIC) :

Usual operators :

- || (logical or)
- && (logical and)
- | (bitwise or)
- & (bitwise and)
- !=
- ==
- <=
- >=
- <
- >
- << (left bitwise shift)
- >> (right bitwise shift)
- -
- +
- *
- /
- % (modulo)
- ^ (power)
- ! (logical not)
- ~ (bitwise not)

These special variable names are pre-defined. They cannot be overloaded :

- 'w' : width of the associated image, if any (else 0).
- 'h' : height of the associated image, if any (else 0).
- 'd' : depth of the associated image, if any (else 0).
- 's' : spectrum of the associated image, if any (else 0).
- 'x' : current processed column of the associated image, if any (else 0).
- 'y' : current processed line of the associated image, if any (else 0).
- 'z' : current processed slice of the associated image, if any (else 0).
- 'c' : current processed channel of the associated image, if any (else 0).
- 'i' : current processed pixel value (i.e. value located at (x,y,z,c)) of the associated image, if any (else 0).
- 'im','iM','ia','iv' : Respectively the minimum, maximum, average values and variance of the associated image, if any (else 0).
- 'xm','ym','zm','cm' : The pixel coordinates of the minimum value in the associated image, if any (else 0).
- 'xM','yM','zM','cM' : The pixel coordinates of the maximum value in the associated, image, if any (else 0).
- 'pi' : value of pi, i.e. 3.1415926...
- 'e' : value of e, i.e. 2.71828...
- '?' or 'u' : a random value between [0,1], following an uniform distribution.
- 'g' : a random value, following a gaussian distribution of variance 1 (roughly in [-5,5]).

Ajouter un canal alpha

2 exemples : Ajouter un canal alpha à une image à 3 canaux :

```
gmic geo2.png -to_colormode 4 -o geo2_alpha.png (-to_colormode  mode={ 0=adaptive | 1=G | 2=GA | 3=RGB | 4=RGBA })
gmic geo2.png -to_rgba -o geo2_alpha2.png
```

Supprimer un canal alpha

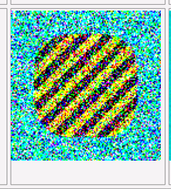
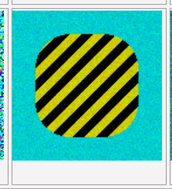
3 exemples : Supprimer le canal alpha d'une image à 4 canaux

```
gmic geo2_alpha.png -remove_opacity -o geo2_sans_alpha.png
gmic geo2_alpha.png -to_colormode 3 -o geo2_sans_alpha2.png
gmic geo2_alpha.png -to_rgb -o geo2_sans_alpha3.png
```

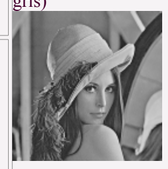
Créer du bruit

Les 5 types de bruits disponibles via la commande -noise std_variation>=0[%], _noise_type ('noise_type' can be { 0=gaussian | 1=uniform | 2=salt&pepper | 3=poisson | 4=rice }) :

```
gmic 64,64,1,3 -noise 100,0 -o bruit_gaussian.png
gmic 64,64,1,3 -noise 100,1 -o bruit_uniform.png
gmic geo2.png -noise 50,2 -o geo_bruit_2.png
gmic geo2.png -noise 50,3 -n 0,255 -o geo_bruit_3.png
gmic geo2.png -noise 50,4 -n 0,255 -o geo_bruit_4.png
```

gaussian (0)	uniform (1)	salt&pepper (2)	poisson (3)	rice (4)
				

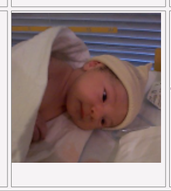
Convertir une image en niveaux de gris

Paramètres de la ligne de commande (aide de G'MIC)	Image d'origine	Ligne de commande	Résultat
-to_gray Force selected images to be in GRAY mode.		`gmic geo.png -to_gray -o to_gray.png`	1 canal (198 niveaux de gris) 
-luminance Compute luminance of selected images.		`gmic geo.png -luminance -o to_gray.png`	
-normalize -n { value0[%] | [image0] },{ value1[%] | [image1] } Linearly normalize values of selected images in specified range. (eq. to '-n').		`gmic geo.png -to_gray -n 0,255 -o to_gray2.png`	1 canal (247 niveaux de gris) 

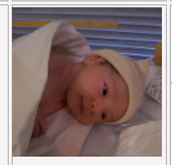
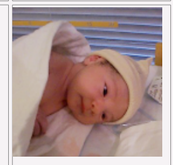
Inverser les couleurs

Paramètres de la ligne de commande (aide de G'MIC)	Image d'origine	Ligne de commande	Résultat
-negative Compute negative of selected images.		`gmic geo2.png -negative -o negative.png`	

Correction du gamma

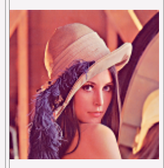
Paramètres de la ligne de commande (aide de G'MIC)	Image d'origine	Ligne de commande	Résultat
-apply_gamma gamma Apply gamma correction to selected images.		`gmic bebe.png -apply_gamma 1.8 -o gamma.png`	

Appliquer une courbe de couleurs

Paramètres de la ligne de commande (aide de G'MIC)	Image d'origine	Ligne de commande	Résultat
-apply_curve 0<=smoothness<=1,x0,y0,x1,y1,x2,y2,...,xN,yN Apply curve transformation to image values.		`gmic bebe.png -apply_curve 1,0,0,128,175,255,255 -o[-1] apply_curve.png`	

Seuils

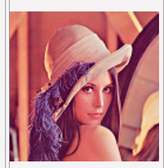
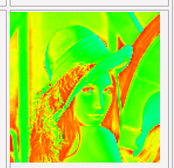
Les résultats contiennent les couleurs suivantes : Blanc (255,255,255), noir (0,0,0), rouge (255,0,0), cyan (0,255,255), vert (0,255,0), magenta (255,0,255), bleu (0,0,255), jaune (255,255,0).

Paramètres de la ligne de commande (aide de G'MIC)	Image d'origine	Ligne de commande	Résultat
-t2 eq. to '-threshold2'. -threshold2 _min[%], _max[%]		`gmic geo.png -threshold2 25%,75% -n 0,255 -o threshold2.png`	
		`gmic geo.png -threshold2 0,50% -n 0,255 -o threshold2_2.png`	

Remplir d'une couleur

Paramètres de la ligne de commande (aide de G'MIC)	Image d'origine	Ligne de commande	Résultat
-fc eq. to '-fill_color'. -fill_color col1,...,colN Fill selected images with specified color. (eq. to '-fc').	sans	`gmic 64,64,1,3 -fill_color 255,255,0 -o fill_color.png`	

Solarisation

Paramètres de la ligne de commande (aide de G'MIC)	Image d'origine	Ligne de commande	Résultat
-solarize Solarize selected images.		`gmic geo.png -solarize -o solarizec.png`	
		`gmic geo.png -solarize -to_gray -o solarize.png`	

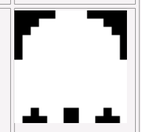
		`gmic geo.png -solarize -to_gray -n 0,255 -o solarize2.png`	

Sepia

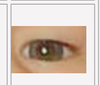
Paramètres de la ligne de commande (aide de G'MIC)	Image d'origine	Ligne de commande	Résultat
-sepia Apply sepia tones effect on selected images.		`gmic geo.png -sepia -o sepia.png`	

Enlever couleurs et opacité

L'opacité devient noire, le reste blanc.

Paramètres de la ligne de commande (aide de G'MIC)	Image d'origine	Ligne de commande	Résultat
-split_opacity Split color and opacity parts of selected images.		`gmic sprite.png -split_opacity -o[-1] split_opacity.png`	

Correction des yeux rouges

Paramètres de la ligne de commande (aide de G'MIC)	Image d'origine	Ligne de commande	Résultat
-red_eye 0<=_threshold<=100, _smoothness>=0,0<=attenuation<=1 Attenuate red-eye effect in selected images.		`gmic yr.png -red_eye 75,3.5,0.1 -o red_eye.png`	

Sélectionner une couleur

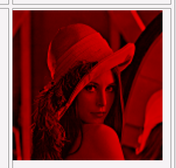
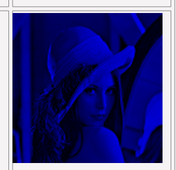
La couleur sélectionnée est blanche, le reste est noir.

Paramètres de la ligne de commande (aide de G'MIC)	Image d'origine	Ligne de commande	Résultat
-select_color _tolerance[%]>=0,col1,..,colN Select pixels with specified color in selected images.		`gmic sprite.png -select_color 1,255,255,255 -n 0,255 -o select_color.png`	
		`gmic geo2.png -select_color 1,0,0,0 -n 0,255 -o select_color2.png`	

Remplacer une couleur

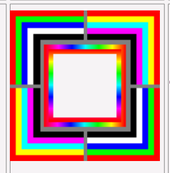
Paramètres de la ligne de commande (aide de G'MIC)	Image d'origine	Ligne de commande	Résultat
-replace_color _tolerance[%]>=0, _smoothness[%]>=0,src1,...,srcN,dest1,...,destN Replace pixels from/to specified colors in selected images.		`gmic geo2.png -replace_color 1,1,0,0,0,127,127,127 -n 0,255 -o replace_color.png`	

Changer de couleurs via une matrice 3*3

Paramètres de la ligne de commande (aide de G'MIC)	Image d'origine	Ligne de commande	Résultat
-mix_rgb a11,a12,a13,a21,a22,a23,a31,a32,a33 Apply 3x3 specified matrix to RGB colors of selected images.		`gmic geo.png -mix_rgb 0,1,0,0,0,0,0,0,0 -o mix_rgb.png`	
		`gmic geo.png -mix_rgb 0,0,0,0,1,0,0,0,0 -o mix_rgb2.png`	
		`gmic geo.png -mix_rgb 0,0,0,0,0,0,0,1,0 -o mix_rgb3.png`	

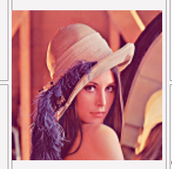
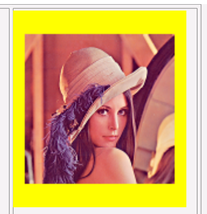
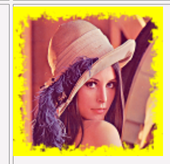
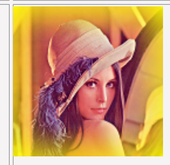
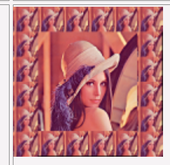
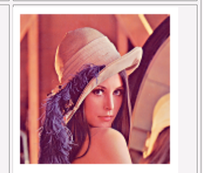
Remplacer les couleurs d'une image par celles d'une autre image

Cette fonction disponible pour les versions > 1.4.5.2 permet de modifier les couleurs de la deuxième image par celles de la première.
Les tests sont faits avec la version 1.4.5.2 sous Windows et il a fallu importer cette fonction dans un fichier transfer_colors.txt

Paramètres de la ligne de commande (aide de G'MIC)	Image d'origine	Ligne de commande	Résultat
transfer_colors : _transfer_brightness={ 0 | 1 } Transfer colors of the first selected image to the other ones.		`gmic -m transfer_colors.txt mire.png geo.png -transfer_colors 0 -o transfer_colors0.png`	
		`gmic -m transfer_colors.txt mire.png geo.png -transfer_colors 1 -o transfer_colors1.png`	

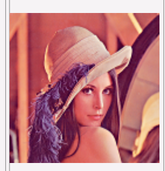
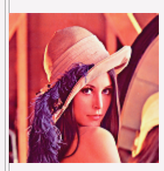
Cadres

Paramètres de la ligne de commande (aide de G'MIC)	Image d'origine	Ligne de commande	Résultat

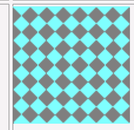
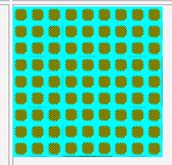
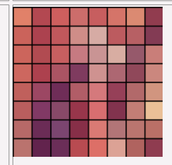
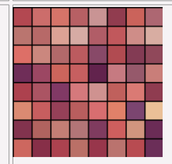
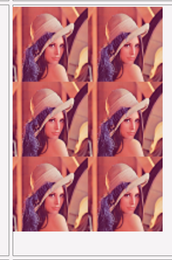
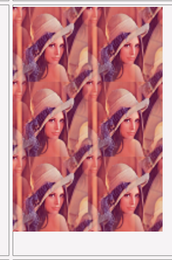
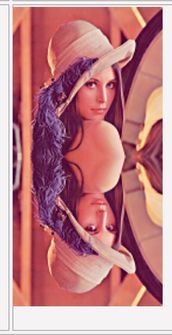
-frame _size_x[%]>=0, _size_y[%]>=0, _col1,..., _colN Insert RGBA-colored frame in selected images.		gmic geo.png -frame 10,20,255,255,0 -o frame.png	
-frame_fuzzy _size_x>=0, _size_y>=0, _fuzzyness>=0, _smoothness>=0, _R, _G, _B, _A Draw RGBA-colored fuzzy frame in selected images.		gmic geo.png -frame_fuzzy 10,10,5,1,255,255,0,255 -o frame_fuzzy.png	
-frame_round _sharpness>0, _size>=0, _smoothness, _shade, _R, _G, _B, _A Draw RGBA-colored round frame in selected images.		gmic geo.png -frame_round 40,40,5,1,255,255,0,255 -o frame_round.png	
-frame_pattern _M>=3, _pattern = { 0=first image | 1=self }, _constrain_size = { 0 | 1 } Insert selected pattern frame in selected images.		gmic geo.png -frame_pattern 6,1,1 -o frame_pattern.png	
-polaroid _size1>=0, _size2>=0 Create polaroid effect in selected images.		gmic geo.png -polaroid 10,20 -o[-1] polaroid2.png	
-drop_shadow _offset_x[%], _offset_y[%], _smoothness[%] Drop shadow behind selected images.		gmic geo.png -drop_shadow 6,6,3 -o drop_shadow2.png	
-frame_blur _sharpness>0, _size>=0, _smoothness, _shade, _blur Draw RGBA-colored round frame in selected images.		gmic geo2.png -frame_blur 1,100,0,1,100% -o frame_blur.png	

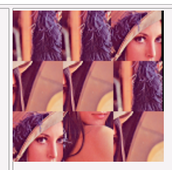
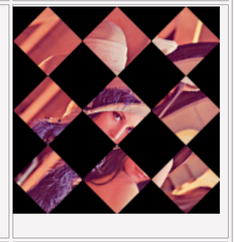
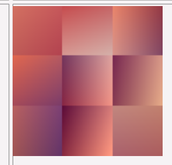
Contrastes

Voici l'effet d'un petit programme de la page http://gmic.sourceforge.net/tutorial.shtml pour jouer sur les contrastes :

Programme	Image d'origine	Ligne de commande	Résultat
Contenu du programme : `#@gmic contrast_stretch : cut_low[%], cut_high[%], normalize_low[%], normalize_high[%] :` `Stretch contrast of image.` `contrast_stretch :` `-e "Stretching contrast of image$?."` `-v - -repeat @#` `-c $1,$2 -n $3,$4` `-mv[-1] 0 -done -v +` Lien direct de téléchargement : contrast_stretch.txt		`gmic -m contrast_stretch.txt geo.png -contrast_stretch 15%,85%,-15%,115% -o contrast_stretch.png`	

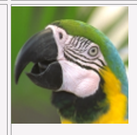
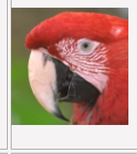
Créer des damiers, motifs ajustables

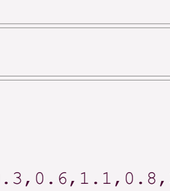
Paramètres de la ligne de commande (aide de G'MIC)	Image d'origine	Ligne de commande	Résultat
-chessboard size1>0, _size2>0, _offset1, _offset2, _angle, _opacity, _color1,.., _color2,.. Draw opaque chessboard on selected images.	sans	Créer un damier avec des cases orientées à 45° gmic 100,100,1,3 -chessboard 10,10,0,0,45,1,127,255,255,127,127,127 -o chessboard.png	
-frame_pattern _M>=3, _pattern = { 0=first image | 1=self }, _constrain_size = { 0 | 1 } Insert selected pattern frame in selected images		Créer un ensemble de 9*9 images gmic geo2.png -frame_pattern 3,1,1 -frame_pattern 3,1,1 -o frame_pattern2.png	
-imagegrid _M>0, _N>0 Create MxN image grid from selected images.		Créer une grille de 8*8 gmic geo.png -imagegrid {w/8},{h/8} -o imagegrid.png	
-taquin _M>0, _N>0 Create MxN taquin puzzle from selected images.		Créer un damier de 5*5 gmic geo.png -taquin 5,5 -o taquin.png	
//		Créer une grille de 8*8 gmic geo.png -taquin 8,8 -imagegrid {w/8},{h/8} -o taquin2.png	
-array _M>0, _N>0, _expand_type={0,1,2} Create MxN array from selected images.		Créer un damier de 2*3 gmic geo.png -array 2,3,1 -o array.png	
-array_fade _M>0, _N>0,0<=_fade_start<=100,0<=_fade_end<=100, _expand_type={0,1,2} Create MxN array from selected images.		Motif raccordable gmic geo.png -array_fade 2,3,0,100,1 -o array_fade.png	
-array_mirror _N>=0, _dir={0,1,2}, _expand_type={ 0 | 1 } Create 2^Nx2^N array from selected images.		gmic geo.png -array_mirror 1,1,1 -o array_mirror.png	

-array_random _Ms>0, _Ns>0, _Md>0, _Nd>0 Create MdxNd array of tiles from selected MsxNs source arrays.		Résultat aléatoire `gmic geo.png -array_random 3,3,3,3`	
-rotate_tiles _angle, _M>0,N>0 Apply MxN tiled-rotation effect on selected images.		`gmic geo.png -rotate_tiles 45,3,3 -o rotate_tiles.png`	
-linearize_tiles _M>0, _N>0 Linearize MxN tiles on selected images.		`gmic geo.png -linearize_tiles 3,3 -c 0,255 -o linearize_tiles.png`	
-quadratize_tiles _M>0, _N>0 Quadratize MxN tiles on selected images.		`gmic geo.png -quadratize_tiles 3,3 -c 0,255 -o quadratize_tiles.png`	

Créer des vidéos

Remarques au sujet des vidéos ⇨
Pour obtenir un fichier d'aide sur FFMPEG utiliser la commande suivante : `ffmpeg -h >aide_ffmpeg.txt`

Paramètres de la ligne de commande (aide de G'MIC)	Image(s) d'origine	Ligne de commande	Résultat
-morph nb_frames>0,_smoothness>=0,_precision>0 Create morphing sequence between selected images.	 	Important : Installer FFMPEG avant d'utiliser la ligne de commande. `gmic m1.png m2.png -morph 50,0.2,0.1 -o morph.mpeg`	Lien téléchargement : morph.mpeg
-animate filter_name,"param1_start,..,paramN_start","param1_end,..,paramN_end",nb_frames>=0,_output_frames={ 0 | 1 },_filename | delay>0 Animate filter from starting parameters to ending parameters.		Phase 1 : Créer une séquence de 50 images PNG de `ani_000000_000000.png` à `ani_000000_000049.png` `gmic 320_240.png -animate tetris,"1","50",50,1,ani.png` Phase 2 : Convertir cette séquence d'images en vidéo avec FFMPEG. `ffmpeg -f image2 -i ani_000000_%6d.png video.avi`	Lien téléchargement : video.avi

		Phase 1 : Créer une séquence de 50 images JPEG de `ani_000000_000000.jpg` à `ani_000000_000049.jpg` `gmic 320_240.png -animate blur_x,"1","50",50,1,ani.jpg` Phase 2 : Convertir cette séquence d'images en vidéo avec OGG Theora. `ffmpeg2theora-0.27.exe ani_000000_%6d.jpg`	Lien téléchargement : ani_000000_%6d.ogv
D'autres vidéos (séquences d'images) sont disponibles sur ce site : • Vidéo effet cube • Vidéo filtre blur_x • Vidéo filtre dilate • Vidéo filtre erode • Vidéo effet mosaic • Vidéo effet puzzle • Vidéo effet tetris			
gimp_fire_edges Edges = float(0.7,0,3) Attenuation = float(0.25,0,1) Smoothness = float(0.5,0,5) Threshold = float(25,0,100) Number of frames = _int(20,1,999) Starting frame = int(20,0,199) Frame skip = _int(0,0,20)		`gmic geo.png -gimp_fire_edges 0.5,0.1,0.6,20,80,1,0 -n 0,255 -o gimp_fire_edges.png`	80 images dont les noms sont indexés pour créer une vidéo d'une image en feu. (volume du fichier trop important)

Nombres aléatoires

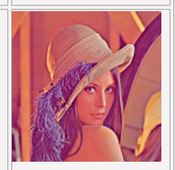
Les nombres aléatoires sont générés à partir de variables pré-définies :

- '?' or 'u' : a random value between [0,1], following an uniform distribution.
- 'g' : a random value, following a gaussian distribution of variance 1 (roughly in [-5,5]).

2 exemples pour créer une image 64*64 remplie d'une couleur aléatoire :

```
gmic 64,64,1,3 rouge={round(u*255)} vert={round(u*255)} bleu={round(u*255)} -fill_color $rouge,$vert,$bleu -o alea_fill_color.png

gmic 64,64,1,3 rouge={round((g+5)*25.5)} vert={round((g+5)*25.5)} bleu={round((g+5)*25.5)} -fill_color $rouge,$vert,$bleu -o alea_fill_color.png
```

Lumière douce

Image d'origine	Ligne de commande	Résultat
	`gmic geo.png --luminance -negative[-1] -blur[-1] 1 -n[-1] 0,255 -compose_softlight 1 -o lnbncs.png`	

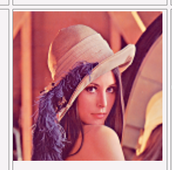
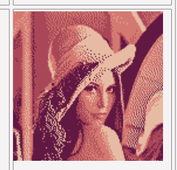
Dessin, peinture

Image d'origine	Ligne de commande	Résultat
	Voir page test_dessin_peinture_gmic.html pour plus d'exemples. `gmic geo.png -gimp_anisotropic_smoothing 80,1,0.3,0.6,1.1,0.8,30,2,0,1,5,1,1 -n 127,255 -sharpen 100 -cartoon 2,60,30,0.18,0.75,256 -sharpen 200 -o geo_dp.png`	

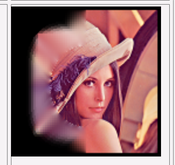
Récupération des couleurs dominantes d'une image (colormap)

Paramètres de la ligne de commande	Image d'origine	Ligne de commande	Résultat
-colormap nb_colors>0,_method={ 0=median-cut | 1=k-means } Estimate best-fitting colormap with 'nb_colors' entries, to index selected images.		`gmic geo.png -colormap 20,0 -o colormap.png`	(image agrandie) 

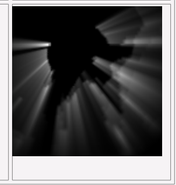
Indexation de l'image avec la meilleure palette des couleurs (autoindex)

Paramètres de la ligne de commande	Image d'origine	Ligne de commande	Résultat
-autoindex nb_colors>0,_dithering>=0,_method={ 0=median-cut | 1=k-means } Command '-autoindex' which indexes image values with the best possible colormap.		`gmic geo.png -autoindex 4,1,1 -o autoindex.png`	

Remplacement des zones transparentes via une extension des couleurs adjacentes par interpolation (solidify)

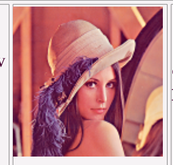
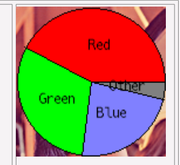
Paramètres de la ligne de commande	Image d'origine	Ligne de commande	Résultat
-solidify Replace transparent regions of a RGBA image by morphologically interpolated color.		`gmic geo_trans.png -solidify -o solidify.png`	

Créer des rayons lumineux (lightrays)

Paramètres de la ligne de commande	Image d'origine	Ligne de commande	Résultat
-lightrays 100<=_density<=0 , _cx , _cy , _ray_length>=0 , _ray_attenuation>=0 Generate ray lights from the edges of selected images.		`gmic geo.png -lightrays 53.68,0.35,0.25,0.02,0.11 -o lightrays.png`	

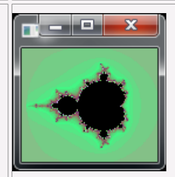
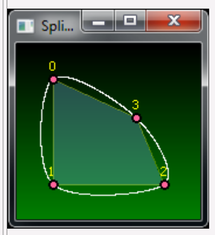
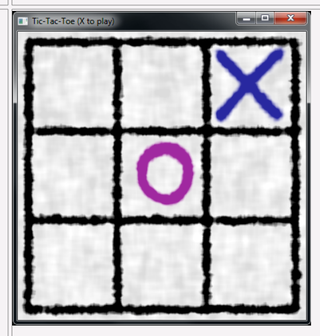
Dessiner des camemberts pour statistiques (lightrays)

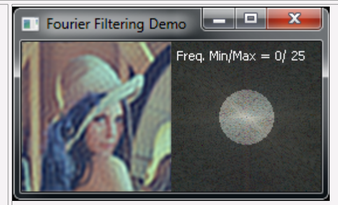
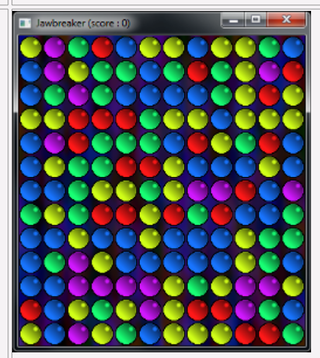
Paramètres de la ligne de commande	Image d'origine	Ligne de commande	Résultat

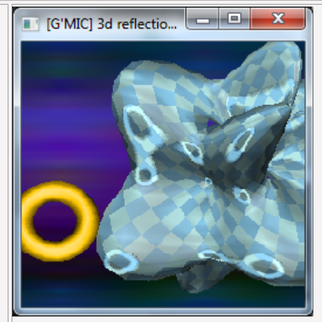
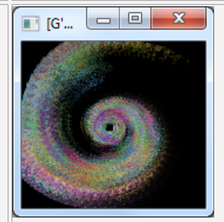
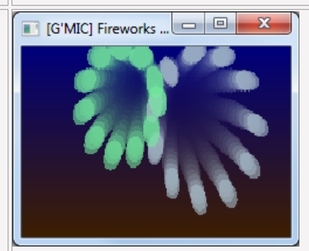
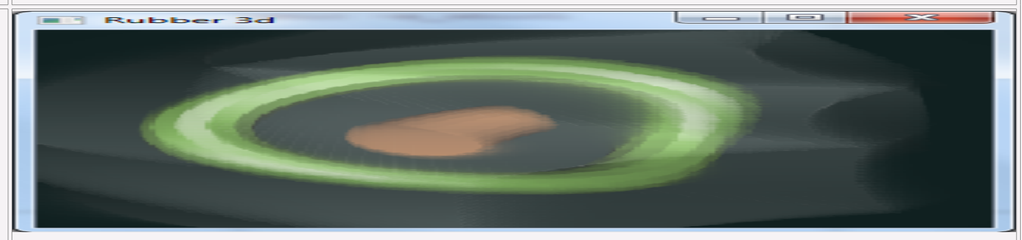
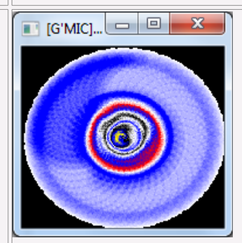
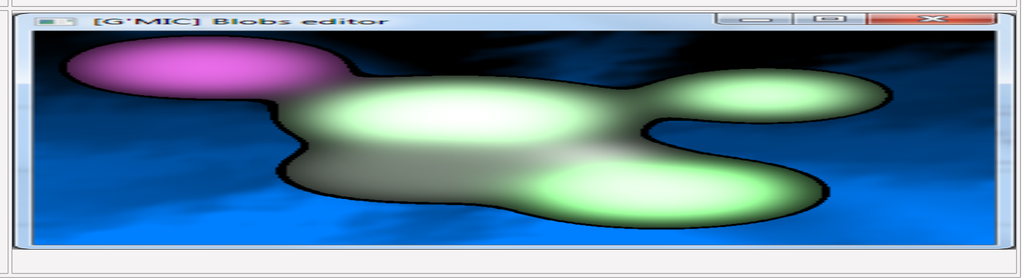
-piechart label_height>=0,"label_color","label1",value1,"color1"...,"labelN",valueN,"colorN" Draw pie chart on selected images..		`gmic 256,256,1,3 -piechart 30,0,0,0,"Red",55,255,0,0,"Green",40,0,255,0,"Blue",30,128,128,255,"Other",5,128,128,128 -o piechart.png`	

Démos

G'MIC contient des programmes de démonstration accessibles via une ligne de commande.

Paramètres de la ligne de commande (aide de G'MIC)	Ligne de commande	Options	Copies d'écran
-x_mandelbrot _julia={ 0 | 1 }, _c0r, _c0i Launch Mandelbrot/Julia explorer.	`gmic -x_mandelbrot`	`------ Mandelbrot/Julia explorer -----------------` `----` `---- Select zooming region with mouse.` `---- Click once to reset zoom factor.` `---- Keys 'ESC' or 'Q' to exit.` `---- Key 'C' to print current fractal coordinates.` `----` `-------------------------------------------------`	
-x_fish_eye Launch fish-eye demo.	`gmic -x_fish_eye`	`------ Fish-eye demo --------------------` `----` `---- Mouse pointer moves fish-eye center.` `---- Mouse buttons set fish-eye size.` `---- Keys 'ESC' or 'Q' to exit.` `----` `----------------------------------------`	
-x_spline Launch spline curve editor.	`gmic -x_spline`	`------ Spline curve editor --------------------------` `----` `---- Mouse to insert/move/delete points.` `---- Key 'R' to reset the curve.` `---- Key 'SPACE' to shows/hide spline curve.` `---- Key 'P' to shows/hide control points.` `---- Key 'ENTER' to shows/hide control polygon.` `---- Key 'T' to shows/hide point tangents.` `---- Key 'I' to shows/hide point indices.` `---- Key 'C' to shows/hide point coordinates.` `---- Keys '+' and '-' to increase/decrease roundness.` `---- Keys 'ESC' or 'Q' to exit.` `----` `-------------------------------------------------------`	
-x_tictactoe Launch tic-tac-toe game.	`gmic -x_tictactoe`	`------ Tic-Tac-Toe game -----------------` `----` `---- Use mouse to select positions of the` `---- symbols. Close window to exit game.` `----` `----------------------------------------`	

-x_fourier Launch fourier filtering demo.	gmic geo.png -x_fourier	------ Fourier-filtering demo ---------- ---- ---- Mouse buttons on the right image to set min/max frequencies. ---- Keys 'ESC' or 'Q' to exit. ---- ----------	
-x_life Launch the game of life.	gmic -x_life	------ The game of life ---------- ---- ---- The goal is to create the biggest possible biological ---- system. You start with a stock of cells which you can ---- spread over the board. For each new cells created ---- simultaneously and spontaneously by your system, you ---- gain more new cells to scatter. ---- ---- Left mouse button to scatter cells in stock. ---- Right mouse button to reset game. ---- Key 'S' to save snapshot of the current view. ---- Keys 'ESC' or 'Q' to exit. ---- ----------	
-x_fire Launch the fire demo.	gmic -x_fire	------ Fire demo ---------- ---- ---- Keys 'ESC' or 'Q' to exit. ---- ----------	
-x_light Launch the light demo.	gmic -x_light	------ Light demo ---------- ---- ---- Move light position with mouse. ---- Mouse buttons fade light in/out. ---- Keys 'ESC' or 'Q' to exit. ---- ----------	
-x_jawbreaker 0<_width<20,0<_height<20,0<_balls<=8 Launch the Jawbreaker game.	gmic -x_jawbreaker	------ Jawbreaker ---------- ---- ---- The goal of the game is to remove the maximum number of ---- balls on the board, simply by clicking on them. But a ---- colored ball can disappear only if it is grouped with at ---- least one ball of the same color. The score is higher if ---- you destroy larger sets of connected colored balls. ---- ---- Left mouse button to select/destroy balls on board. ---- Key 'BACKSPACE' or 'SPACE' to undo the last move. ---- Key 'S' to save snapshot of the current view. ---- Keys 'ESC' or 'Q' to exit. ---- ----------	
-x_paint Launch the interactive painter.	gmic -x_paint	------ Interactive painter ---------- ---- ---- Use mouse to select color and brush. ---- Left button draws a colored stroke. ---- Right button fills a colored region. ---- Arrow keys or SPACE and BACKSPACE to swap ---- between available images. ---- Key 'S' to save snapshot of the current view. ---- Keys 'ESC' or 'Q' to exit. ---- ----------	

-x_reflection3d Launches the 3d reflection demo.	`gmic -x_reflection3d`	aucune	
x_rubber3d -x_whirl Launches a kind of Fractal whirl animated demo.	`gmic -x_whirl`	aucune	
-x_fireworks Launches a simple fireworks animated demo.	`gmic -x_fireworks`	aucune	
-x_rubber3d Launches a 3d rubber object demo.	`gmic -x_rubber3d`	aucune	
-x_shadebobs Launches a classical shade bobs animation.	`gmic -x_shadebobs`	aucune	
-x_blobs Launches a small and interactive blobs editor.	`gmic -x_blobs`	`----------  Blobs editor  ----------` `----` `----  Mouse to insert/move/delete blobs.` `----  Keys 'ESC' or 'Q' to exit.` `----` `-------------------------------------------------`	

-x_minimal_path Command which runs a minimal path computation demo, for segmenting images.	`gmic -x_minimal_path`	aucune	
-x_hough Launches an interactive demo that illustrates the use of the hough-tranform to detect lines in images.	`gmic -x_hough`	`------ Hough-transform demo ----------------------------------` `----` `---- Mouse buttons on the vote image to draw corresponding line.` `---- Mouse buttons on the image to vote for all lines crossing.` `---- the clicked point.` `---- Key 'SPACE' to reset the hough window.` `---- Keys 'ESC' or 'Q' to exit.` `----` `----------------------------------------------------------------`	Voir image en dessous
-houghsketchbw _density>=0,_radius>0,0<=_threshold<=100,0<=_opacity<=1,_votesize[%]>0	`gmic geo.png --houghsketchbw`		

Utilisation des raccourcis pour les commandes

Certaines commandes de G'MIC ont deux orthographes, voici une table des correspondances :

-add	-+
-add3d	-+3d
-append	-a -x_paint Launch the interactive painter.
-background3d	-b3d
-blur	-b
~~-break~~	
-bsl	-<<
-bsr	->>
-center3d	-c3d
-color3d	-col3d
-command	-m
-crop	-z
-cut	-c
-display	-d
-display3d	-d3d
-display_graph	-dg

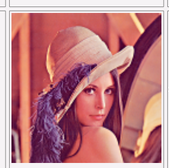
-display_warp	-dw
-div	-/
-div3d	-/3d
-double3d	-db3d
-echo	-e
-endlocal	-endl
-eq	-==
-exec	-x
-fill	-f
-gradient'	-g
-focale3d	-f3d
-ge	->=
-gt	->
-help	-h
-image	-j
-input	-i
-keep	-k
-le	-<=
-light3d	-l3d
-local	-l
-lt	-<
-mdiv	-// (équivalent à)
-mmul	-**
-mode3d	-m3d
-moded3d	-md3d
-move	-mv
-mul	-*
-mul3d	-*3d
-name	-nm
-neq	-!=
-normalize	-n
-normalize3d	-n3d
-opacity3d	-o3d
-output	-o
-pop	-pp
-pow	-^
-primitives3d	-p3d
-push	-p

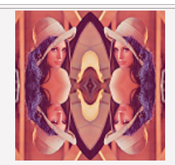
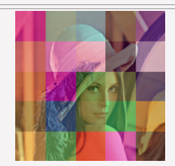
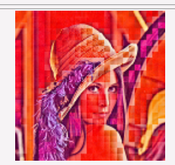
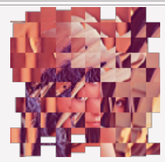
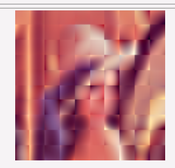
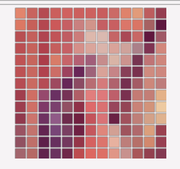
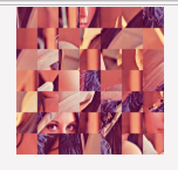
-push=	-p=
-quit	-q
-remove	-rm
-resize	-r
-reverse	-rv
-reverse3d	-rv3d
-rotate3d	-rot3d
-set	-=
-shared	-sh
-specl3d	-sl3d
-specs3d	-ss3d
-split	-s
-split3d	-s3d
-status	-u
-sub	--
-sub3d	--3d
-texturize3d	-t3d
-threshold	-t
-unroll	-y
-update	-up
-verbose	-v
-window	-w

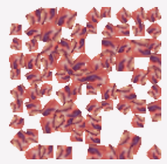
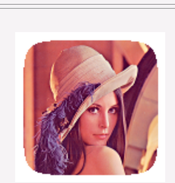
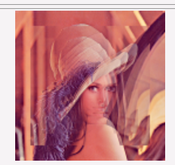
Test des "greffons de Gimp écrits en G'MIC" via l'Invite de commandes (Résultats obtenus sans démarrer Gimp)

G'MIC, comme programme autonome, permet d'obtenir les effets du GUI de Gimp.
Ici le classement des filtres est semblable à celui du GUI de Gimp.
Certaines fonctions sont identiques à celles présentées au début de cette page.

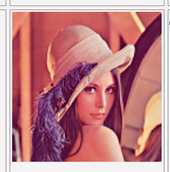
Arrays & Frames

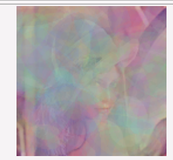
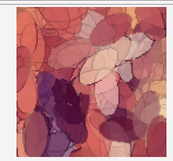
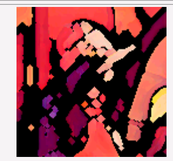
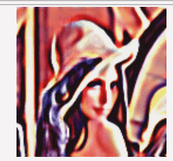
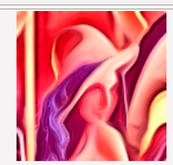
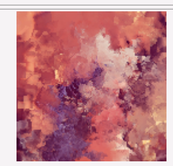
Commande et paramètres de la ligne de commande.	Image d'origine	Ligne de commande	Résultat
-gimp_array X-tiles = int(2,1,10) Y-tiles = int(2,1,10) X-offset = float(0,0,100) Y-offset = float(0,0,100) Mirror = choice("None","X-axis","Y-axis","XY-axes") Size = _choice("Shrink", "Expand", "Repeat [Memory consuming !]")		`gmic geo.png -gimp_array 2,2,0,0,1,0 -o gimp_array.png`	
-gimp_array_fade X-tiles = int(2,1,10) Y-tiles = int(2,1,10) X-offset = float(0,0,100) Y-offset = float(0,0,100) Fade start = float(80,1,100) Fade end = float(90,1,100) Mirror = choice("None","X-axis","Y-axis","XY-axes") Size = _choice("Shrink", "Expand", "Repeat [Memory consuming !]")		`gmic geo.png -gimp_array_fade 2,2,0,0,80,90,1,0 -o gimp_array_fade.png`	

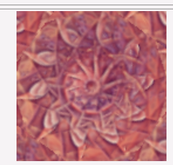
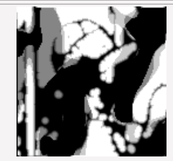
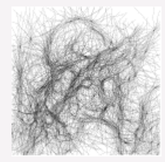
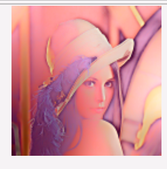
-gimp_array_mirror Iterations = int(1,1,10) X-offset = float(0,0,100) Y-offset = float(0,0,100) Array mode = choice(2,"X-axis","Y-axis","XY-axes") Mirror = choice("None","X-axis","Y-axis","XY-axes") Expand size = _bool(false)	`gmic geo.png -gimp_array_mirror 1,0,0,2,0,0 -o gimp_array_mirror.png`	
-array_random Source X-tiles = int(5,1,20) Source Y-tiles = int(5,1,20) Destination X-tiles = int(7,1,20) Destination Y-tiles = int(7,1,20)	`gmic geo.png -array_random 5,5,7,7 -o array_random2.png`	
-gimp_array_color X-tiles = int(5,1,20) Y-tiles = int(5,1,20) Opacity = float(0.5,0,1)	`gmic geo.png -gimp_array_color 5,5,0.5 -o gimp_array_color.png`	
-gimp_rotate_tiles X-tiles = int(5,1,80) Y-tiles = int(5,1,80) Angle = float(15,0,360) Opacity = float(1,0,1)	`gmic geo.png -gimp_rotate_tiles 5,5,15,1 -o gimp_rotate_tiles.png`	
-gimp_normalize_tiles X-tiles = int(25,1,80) Y-tiles = int(25,1,80) Minimal value = float(0,0,255) Maximal value = float(255,0,255)	`gmic geo.png -gimp_normalize_tiles 25,25,0,255 -o gimp_normalize_tiles.png`	
-gimp_shift_tiles X-tiles = int(10,1,30) Y-tiles = int(10,1,30) Amplitude = float(10,0,100) Opacity = float(1,0,1)	`gmic geo.png -gimp_shift_tiles 10,10,10,1 -o gimp_shift_tiles.png`	
-gimp_parameterize_tiles X-tiles = int(10,1,30) Y-tiles = int(10,1,30) Fitting function = choice("Linear","Quadratic")	`gmic geo.png -gimp_parameterize_tiles 10,10,1 -o gimp_parameterize_tiles.png`	
-gimp_imagegrid X-size = int(10,2,100) Y-size = int(10,2,100)	`gmic geo.png -gimp_imagegrid 10,10 -o gimp_imagegrid.png`	
-taquin X-tiles = int(7,1,20) Y-tiles = int(7,1,20)	`gmic geo.png -taquin 7,7 -o taquin3.png`	

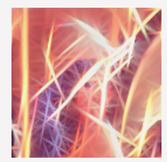
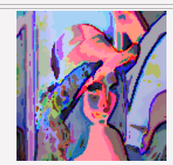
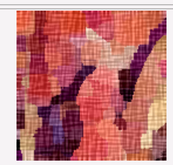
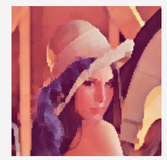
-gimp_array_pattern X-tiles = int(10,1,30) Y-tiles = int(10,1,30) Density = float(80,0,100) Angle = float(180,0,180) Zoom = float(30,0,100) Opacity = float(1,0,1) Image size = _choice("Shrink", "Expand", "Repeat [Memory consuming !]")		gmic geo.png -gimp_array_pattern 10,10,80,180,30,1,1 -c 0,255 -o gimp_array_pattern.png	
-gimp_frame X-start = int(0,0,100) X-end = int(100,0,100) Y-start = int(0,0,100) Y-end = int(100,0,100) Width = int(10,0,100) Height = int(10,0,100) Color = color(0,0,0,255) Outline size = int(1,0,100) Outline color = color(255,255,255,255)		gmic geo.png -gimp_frame 0,100,0,100,10,10,0,0,0,255,1,255,255,255,255 -o gimp_frame.png	
-gimp_frame_pattern Tiles = int(10,3,30) Pattern = choice(1,"Top layer","Self image") Iterations = int(1,1,10) Constrain image size = _bool(1)		gmic geo.png -gimp_frame_pattern 10,1,1,1 -o gimp_frame_pattern.png	
-gimp_frame_fuzzy Width = int(10,0,99) Height = int(10,0,99) Fuzzyness = float(10,0,40) Smoothness = float(1,0,5) Color = color(255,255,255,255)		gmic geo.png -gimp_frame_fuzzy 10,10,10,1,255,255,255,255 -o gimp_frame_fuzzy.png	
-gimp_frame_round Sharpness = float(6,0.1,40) Size = float(20,0,100) Smoothness = float(0.1,0,15) Shade = float(0,0,1) Color = color(255,255,255,255) Blur frame = float(0,0,100) Blur shade = float(0.1,0,1) Blur amplitude = float(3,0,10)		gmic geo.png -gimp_frame_round 6,20,0.1,0,255,255,255,255,0,0.1,3 -o gimp_frame_round.png	
-gimp_tunnel Depth = int(4,1,100) Factor = float(80,1,99) X-center = float(0.5,0,1) Y-center = float(0.5,0,1) Opacity = float(0.2,0,1)		gmic geo.png -gimp_tunnel 4,80,0.5,0.5,0.2 -o gimp_tunnel.png	

Artistic

Commande et paramètres de la ligne de commande.	Image d'origine	Ligne de commande	Résultat
-gimp_polaroidgimp_warhol.png Frame size = int(10,1,400) Bottom size = int(20,1,400) X-shadow = float(0,-20,20) Y-shadow = float(0,-20,20) Smoothness = float(3,0,5) Angle = float(20,0,360)		gmic geo.png -gimp_polaroid 10,20,0,0,3,30 -o gimp_polaroid.png	

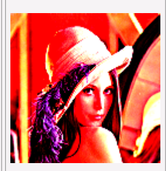
- old_photo		`gmic geo.png -old_photo -o gimp_old_photo.png`	
-gimp_reflect Height = float(50,0,100) Attenuation = float(1,0.1,4) Color = color(110,160,190,64) Waves amplitude = float(0,0,100) Waves smoothness = float(1.5,0,4) X-angle = float(0,-10,10) Y-angle = float(-3.30,-10,10) Focale = float(7,0,10) Zoom = float(1.5,1,5)		`gmic geo.png -gimp_reflect 50,1,110,160,190,64,0,1.5,0,-3,7,1.5 -o gimp_reflect.png`	
-gimp_color_ellipses Density = int(400,0,3000) Radius = float(8,0,30) Opacity = float(0.1,0.01,0.5)		`gmic geo.png -gimp_color_ellipses 400,20,0.1 -o gimp_color_ellipses.png`	
-gimp_ellipsionism Primary radius = float(20,1,100) Secondary radius = float(10,1,100) Smoothness = float(0.5,0,10) Opacity = float(0.7,0,1) Outline = float(8,1,3) Density = float(0.5,0.1,2)		`gmic geo.png -gimp_ellipsionism 20,10,0.5,0.7,8,0.5 -o gimp_ellipsionism.png`	
-cartoon Smoothness = float(2,0,10) Sharpening = float(200,0,400) Edge threshold = float(10,1,30) Edge thickness = float(0.25,0,1) Color strength = float(1.5,0,3) Color quantization = int(32,2,256)		`gmic geo.png -cartoon 2,200,10,0.25,1.5,32 -o gimp_cartoon.png`	
-gimp_pen_drawing Amplitude = float(10,0,30)		`gmic geo.png -gimp_pen_drawing 10 -o gimp_pen_drawing.png`	
-draw_whirl Amplitude = float(20,0,100)		`gmic geo.png -draw_whirl 10 -o gimp_draw_whirl.png`	
-gimp_painting Abstraction = int(1,1,10) Smoothness = float(1.5,0,5) Color = float(2,0,4)		`gmic geo.png -gimp_painting 1,1.5,2 -o gimp_painting.png`	
-cubism Iterations = int(300,1,2000) Bloc size = float(10,0,40) Angle = float(90,0,360) Opacity = float(0.7,0.01,1) Smoothness = float(0,0,5)		`gmic geo.png -cubism 300,10,90,0.7,0 -o gimp_cubism.png`	

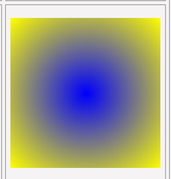
-gimp_kaleidoscope X-center = float(0.5,0,1) Y-center = float(0.5,0,1) X-offset = float(0,0,100) Y-offset = float(0,0,100) Radius cut = float(100,0,100) Angle cut = float(10,0,100) Borders = choice(2,"Black","Nearest","Repeat")		`gmic geo.png -gimp_kaleidoscope 0.5,0.5,0,0,100,10,1 -o gimp_kaleidoscope.png`	
-gimp_rotoidoscope X-center = float(0.5,0,1) Y-center = float(0.5,0,1) Angular tiles = int(10,1,72) Smoothness = float(0.5,0,5) Borders = choice(2,"Black","Nearest","Repeat")		`gmic geo.png -gimp_rotoidoscope 0.5,0.5,10,0.5,2 -o gimp_rotoidoscope.png`	
-gimp_stencilbw Threshold = float(10,0,30) Smoothness = float(10,0,30) Hue = float(0,0,360) Saturation = float(0,0,1)		`gmic geo.png -gimp_stencilbw 8,23,0,0 -o gimp_stencilbw.png`	
-gimp_pencilbw Size = float(0.3,0,5) Amplitude = float(60,0,200) Hue = float(0,0,360) Saturation = float(0,0,1)		`gmic geo.png -gimp_pencilbw 0.3,60,0,0 -o gimp_pencilbw.png`	
-gimp_hardsketchbw Amplitude = float(1000,0,4000) Sampling = float(3,1,100) Smoothness = float(1,0,10) Opacity = float(0.1,0,1) Edge = float(20,0,100) Negative = bool(0)		Ajout depuis la version 1.4.5.0 Utilisation sous Gimp : http://www.flickr.com/groups/gmic/discuss/72157625338708940/ `gmic geo.png -gimp_hardsketchbw 64.94,1,0.42,0.07,21.1,0 -o gimp_hardsketchbw.png`	
-gimp_sketchbw Number of orientations = int(2,1,16) Starting angle = float(45,0,180) Angle range = float(180,0,180) Stroke length = float(30,0,1000) Contour threshold = float(1,0,3) Opacity = float(0.03,0,0.3) Background intensity = float(0,0,2) Density = float(0.6,0,5) Sharpness = float(0.1,0,1.5) Anisotropy = float(0.6,0,1) Smoothness = float(0.25,0,10) Coherence = float(1,0,10) Boost stroke = bool(0) Curved stroke = bool(1) Color model = choice("Black on white","White on black","Black on transparent white","White on transparent black")		`gmic geo.png -gimp_sketchbw 9,45,180,30,1,0.03,0,0.6,0.1,0.6,0.25,1,1,1,0 -o gimp_sketchbw.png`	
-warhol X-tiles = int(3,1,10) Y-tiles = int(3,1,10) Smoothness = float(2,0,10) Color = float(40,0,60)		`gmic geo.png -warhol 3,3,2,40 -o gimp_warhol.png`	
-gimp_glow Amplitude = float(1,0,20) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue","Saturation", "Value","Key","Alpha","ch-components","c-component","h-component")		`gmic geo.png -gimp_glow 25,3 -o gimp_glow.png`	

-gimp_tetris Scale = int(10,1,20)		`gmic geo.png -gimp_tetris 10 -o gimp_tetris.png`	
-gimp_rodilius Amplitude = float(10,0,30) Thickness = float(10,0,100) Sharpness = float(300,0,1000) Orientations = int(5,2,36) Offset = float(30,0,180) Color mode = choice(1,"Darker","Lighter") Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue","Saturation", "Value","Key","Alpha","ch-components","c-component","h-component","Red","Green","Blue","Alpha")		mic geo.png -gimp_rodilius 18,10,300,5,30,1,0 -o gimp_rodilius.png	
-gimp_color_abstraction Smoothness = float(1,0,10) Levels = int(10,2,255) Contrast = float(0.2,0.01,1)		`gmic geo.png -gimp_color_abstraction 1,10,0.2 -n 0,255 -o gimp_color_abstraction.png`	
-gimp_lylejk_painting Iterations = int(2,1,20) Radius = int(4,1,30) Canvas = float(10,0,100)		`gmic geo.png -gimp_lylejk_painting 5,4,50 -o gimp_lylejk_painting.png`	
-gimp_kuwahara Iterations = int(2,1,20) Radius = int(5,1,30) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances", "Blue chrominance","Red chrominance","Lightness","ab-components", "a-component","b-component","Hue","Saturation","Value", "Key","Alpha","ch-components","c-component","h-component", "Red","Green","Blue","Alpha")		`gmic geo.png -gimp_kuwahara 1,3,0 -o gimp_kuwahara.png`	

Colors

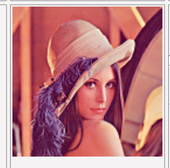
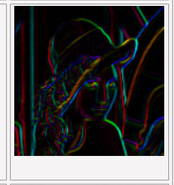
Les filtres ne sont pas tous illustrés car ils ont souvent des fonctions similaires.
Lors des tests, "gimp_blackandwhite" causait des problèmes sur les versions 1.4.4.2 (gmic.exe) & 1.4.5.0 (greffon de Gimp).
Ligne de commande : `gmic geo.png -gimp_blackandwhite 0.299,0,0.587,0,0.114,0,1,1,0,0,0,0,0,0,2,0,2,0,16,0 -o gimp_blackandwhite.png`

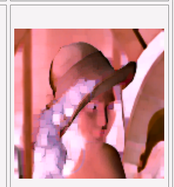
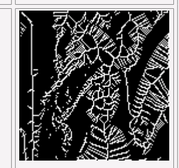
Commande et paramètres de la ligne de commande.	Image(s) d'origine	Ligne de commande	Résultat
-gimp_mix_rgb Red contrast = float(1,0,4) Red brightness = float(0,-255,255) Red smoothness = float(0,0,10) Green contrast = float(1,0,4) Green brightness = float(0,-255,255) Green smoothness = float(0,0,10) Blue contrast = float(1,0,4) Blue brightness = float(0,-255,255) Blue smoothness = float(0,0,10) Tones range = choice("All tones","Shadows","Mid-tones","Highlights") Tones smoothness = float(2,0,10)		`gmic geo.png -gimp_mix_rgb 4,0,0,4,0,0,4,0,0,0,3 -o gimp_mix_rgb.png`	
-gimp_sepia Gamma = float(1,0.01,5) Contrast = float(1,0,4) Brightness = float(0,-255,255)		`gmic geo.png -gimp_sepia 1,1.5,0 -o gimp_sepia.png`	

-gimp_bwrecolorize Gamma = float(1,0.01,5) Contrast = float(1,0,4) Brightness = float(0,-255,255) Normalize = bool(0) Gradient preset = choice("User-defined","Black to white","White to black","Sepia","Solarize") Interpolation type = choice(1,"Nearest","Linear","Cubic","Lanczos") Preserve initial brightness = bool(0) Number of tones = int(5,2,8) 1st tone = color(0,0,0,255) 2nd tone = color(43,25,55,255) 3rd tone = color(158,137,189,255) 4th tone = color(224,191,228,255) 5th tone = color(255,255,255,255) 6th tone = color(255,255,255,255) 7th tone = color(255,255,255,255) 8th tone = color(255,255,255,255)		Number of tones = 2 `gmic geo.png -gimp_bwrecolorize 1,1,1,0,0,1,1,2,255,0,0,255,127,127,127,255 -o gimp_bwrecolorize.png`	
-gimp_map_tones Threshold = float(0.5,0,1) Gamma = float(0.7,0,1) Smoothness = float(0.1,0,10) Iterations = int(30,0,500) Channel(s) = choice(3,"All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic geo.png -gimp_map_tones 0.5,0.5,0.1,30,3 -o gimp_map_tones.png`	
-gimp_normalize_local Amplitude = float(2,0,60) Radius = int(6,1,64) Neighborhood smoothness = float(5,0,40) Average smoothness = float(20,0,40) Constrain values = bool(1) Channel(s) = choice(3,"All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic geo.png -gimp_normalize_local 2,6,5,20,1,3 -o gimp_normalize_local.png`	
-gimp_select_color Similarity space = choice(0,"RGB[A]","RGB","YCbCr","Red","Green","Blue","Opacity", "Luminance","Blue & Red chrominances","Hue","Saturation") Tolerance = float(20,0,100) Smoothness = float(0,0,10) Selected color = color(255,255,255,255) Output as = choice(0,"Selected colors","Selected mask") Invert selection = bool(0)		Invert selection =0 `gmic geo.png -gimp_select_color 0,20,0,226,116,115,255,0,0 -o gimp_select_color.png`	
		Invert selection =1 `gmic geo.png -gimp_select_color 0,20,0,226,116,115,255,0,1 -o gimp_select_color_i.png`	
-gimp_replace_color Tolerance = float(100,1,450) Smoothness = float(0,0,10) Selected color = color(255,255,255,255) Replaced color = color(0,0,0,0)		`gmic sprite.png -gimp_replace_color 100,0,255,0,0,255,127,255,255,255 -o gimp_replace_color.png`	
-gimp_linear_gradient Starting color = color(0,0,0,255) Ending color = color(255,255,255,255) Swap colors = bool(0) Angle = float(45,0,360) Fade start = float(0,0,100) Fade end = float(100,0,100)	sans	`gmic 128,128,1,4 -gimp_linear_gradient 0,0,255,255,255,255,0,255,0,45,0,100 -o gimp_linear_gradient.png`	
-gimp_radial_gradient Starting color = color(0,0,0,255) Ending color = color(255,255,255,255) Swap colors = bool(0) Fade start = float(0,0,100) Fade end = float(100,0,100) X-center = float(50,0,100) Y-center = float(50,0,100)	sans	`gmic 128,128,1,4 -gimp_radial_gradient 0,0,255,255,255,255,0,255,0,0,100,50,50 -o gimp_radial_gradient.png`	

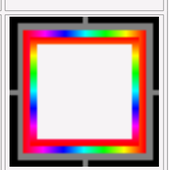
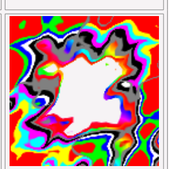
-gimp_corner_gradient Color 1 (up/left corner) = color(255,255,255,128) Color 2 (up/right corner) = color(255,0,0,255) Color 3 (bottom/left corner) = color(0,255,0,255) Color 4 (bottom/right corner) = color(0,0,255,255)	sans	`gmic 128,128,1,4 -gimp_corner_gradient 0,0,255,255,255,255,0,255,0,255,0,255,255,0,255,255 -o gimp_corner_gradient.png`	
-gimp_colormap choice[1,"Adaptive","Custom","Standard (256)","HSV (256)","Lines (256)","Hot (256)", "Cool (256)","Jet (256)","Flag (256)","Cube (256)"] Dithering = float(1,0,1) Number of tones = int(32,2,256) Number of colors = int(8,2,8) 1st color = color(0,0,0) 2nd color = color(255,255,255) 3rd color = color(255,0,0) 4th color = color(0,255,0) 5th color = color(0,0,255) 6th color = color(255,255,0) 7th color = color(255,0,255) 8th color = color(0,255,255)		`gmic geo.png -gimp_colormap 0,0,4,6 -o gimp_colormap.png`	Résultat image à 4 couleurs 
-gimp_metallic Strength = float(1,0,1) Smoothness = float(0,0,20) Metal = choice("silver","gold","copper","bronze","blue steel")		`gmic geo.png -gimp_metallic 1,0,1 -o gimp_metallic.png`	

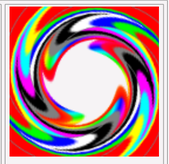
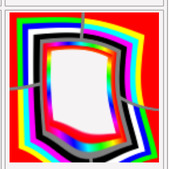
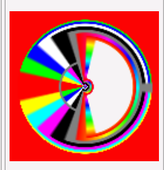
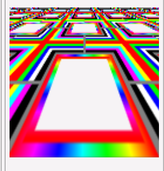
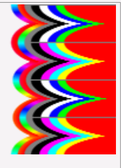
Contours

Commande et paramètres de la ligne de commande.	Image(s) d'origine	Ligne de commande	Résultat
-gimp_gradient_norm Smoothness = float(0,0,10) Linearity = float(0.5,0,1.5) Min threshold = float(0,0,100) Max threshold = float(100,0,100) Negative colors = bool(0)		Negative colors = 1 `gmic geo.png -gimp_gradient_norm 0.9,0.5,21,80,1 -o gimp_gradient_norm.png`	
-gimp_gradient2rgb Smoothness = float(0,0,10) Min threshold = float(0,0,100) Max threshold = float(100,0,100) Orientation only = bool(0) Negative colors = bool(0)		`gmic geo.png -gimp_gradient2rgb 0,0,100,0,0 -o gimp_gradient2rgb.png`	
-gimp_local_orientation Smoothness = float(0,0,5) Min threshold = float(0,0,100) Max threshold = float(100,0,100) Negative colors = bool(0) Channel(s) = choice(3,"All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic geo.png -gimp_local_orientation 0,0,100,0,3 -o gimp_local_orientation.png`	
-gimp_curvature Smoothness = float(2,0,10) Min threshold = float(0,0,100) Max threshold = float(100,0,100) Absolute value = bool(0) Negative colors = bool(0)		`gmic geo.png -gimp_curvature 2,0,100,0,0 -o gimp_curvature.png`	
-gimp_edges Smoothness = float(0,0,10) Threshold = float(15,0,50) Negative colors = bool(0)		`gmic geo.png -gimp_edges 0,15,0 -o gimp_edges.png`	

-gimp_thin_edges Smoothness = float(0,0,10) Threshold = float(15,0,50) Negative colors = bool(0)		`gmic geo.png -gimp_thin_edges 0,15,0 -o gimp_thin_edges.png`	
-gimp_edge_offsets Smoothness = float(0,0,10) Threshold = float(15,0,50) Scale = int(4,0,32) Thickness = int(1,0,16) Negative colors = bool(0)		`gmic geo.png -gimp_edge_offsets 0,15,4,1,0 -o gimp_edge_offsets.png`	
-gimp_segment_watershed Edge threshold = float(2,0,5) Smoothness = float(1,0,5) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic geo.png -gimp_segment_watershed 2,1,3 -o gimp_segment_watershed.png`	
-gimp_morpho "Original - Erosion","Dilation - Original","Original - Opening","Closing - Original") Size = int(5,2,60) Invert colors = bool(false) Shape = choice(0,"Square","Octagonal","Circular") Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance","Red chrominance", "Lightness","ab-components","a-component","b-component","Hue","Saturation", "Value","Key","Alpha","ch-components","c-component","h-component") #@gimp : Scale = bool(true)		`gmic geo.png -gimp_morpho 3,5,1,0,3,1 -o gimp_morpho.png`	
-gimp_skeleton Method = choice("Distance","Thinning") Smoothness = float(0,0,10) Curviness = float(0,0,10) Multiple channels = bool(1)		`gmic geo.png -gimp_skeleton 0,0,0,0 -o gimp_skeleton.png`	

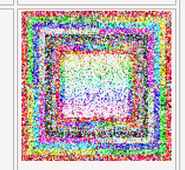
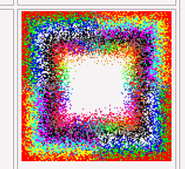
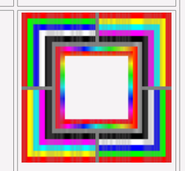
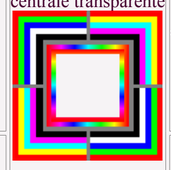
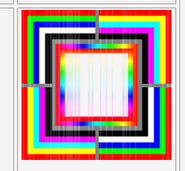
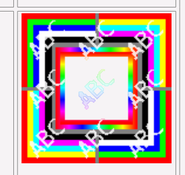
Deformations

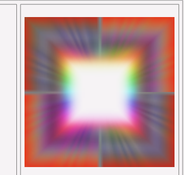
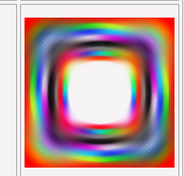
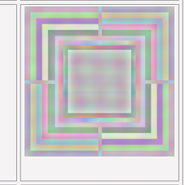
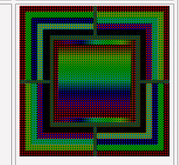
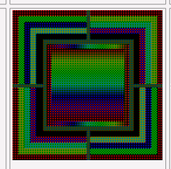
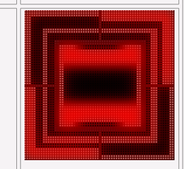
Commande et paramètres de la ligne de commande.	Image(s) d'origine	Ligne de commande	Résultat
-gimp_zoom Factor = float(2,0.01,10) X-center = float(0.5,0,1) Y-center = float(0.5,0,1) Borders = choice(0,"Black","Nearest","Repeat")	mire avec la partie centrale transparente 	`gmic mire.png -gimp_zoom 1.5,0.5,0.5,0 -o gimp_zoom.png`	
-water Amplitude = float(30,0,300) Smoothness = float(1.5,0,4)		`gmic mire.png -water 30,1.5 -o gimp_water.png`	
-wave Amplitude = float(10,0,30) Frequency = float(0.4,0,2) X-center = float(50,0,100) Y-center = float(50,0,100)		`gmic mire.png -wave 10,0.4,50,50 -o gimp_wave.png`	

-twirl Amplitude = float(1,-5,5) X-center = float(0.5,0,1) Y-center = float(0.5,0,1) Borders = choice(1,"Black","Nearest","Repeat")		`gmic mire.png -twirl 1,0.5,0.5,1 -o gimp_twirl.png`	
-gimp_flower Amplitude = float(30,-100,100) Petals = int(6,0,20) Offset = float(0,0,100) Angle = float(0,0,360) X-center = float(0.5,0,1) Y-center = float(0.5,0,1) Borders = choice(1,"Black","Nearest","Repeat")		`gmic mire.png -gimp_flower 30,6,0,0,0.5,0.5,1 -o gimp_flower.png`	
-deform Amplitude = float(10,0,100)		`gmic mire.png -deform 10 -o gimp_deform.png`	
-fisheye X-center = float(50,0,100) Y-center = float(50,0,100) Radius = float(70,0,100) Amplitude = float(1,0,2)		`gmic mire.png -fisheye 50,50,80,1 -o gimp_fish_eye.png`	
~~-gimp_map_sphere~~ ~~Width = _int(512,1,4096)~~ ~~Height = _int(512,1,4096)~~ ~~Radius = float(90,0,400)~~ ~~Dilation = float(0.5,0,1)~~ ~~Angle = float(0,-50,50)~~		~~`gmic mire.png -gimp_map_sphere 128,128,90,0.5,0 -o gimp_map_sphere.png`~~	
-gimp_map_sphere Width = _int(512,1,4096) Height = _int(512,1,4096) Radius = float(90,0,400) Dilation = float(0.5,0,1) Angle = float(0,-50,50) Border smoothness = float(0,0,200) Border width = float(20,0,100) Orientation = choice("0 deg.","90 deg.","180 deg.","270 deg.") Background = choice("Transparent","Mean color") Fading = float(0,0,100) Fading shape = float(0.5,0,3)		`gmic mire.png -gimp_map_sphere 128,128,90,0.5,0,0,20,0,1,0,0 -o gimp_map_sphere2.png`	
-gimp_warp_perspective X-angle = float(1.73,-4,4) Y-angle = float(0,-4,4) Zoom = float(1,0.1,4) X-center = float(50,0,100) Y-center = float(50,0,100) X-offset = float(0,0,100) Y-offset = float(0,0,100) Borders = choice(2,"Black","Nearest","Repeat")		`gmic mire.png -gimp_warp_perspective 1.73,0,1,50,50,0,0,2 -o gimp_warp_perspective.png`	
-gimp_euclidean2polar X-center = float(0.5,0,1) Y-center = float(0.5,0,1) Dilation = float(1,0.1,10) Borders = choice(1,"Black","Nearest","Repeat") Inverse transform = bool(0)		`gmic mire.png -gimp_euclidean2polar 0.5,0.5,1,1,0 -o gimp_euclidean2polar.png`	
-gimp_custom_deformation X-warping = text{"(w+h)/20 * cos(y*20/h)"} Y-warping = text{"(w+h)/20 * sin(x*20/w)"} Relative warping = bool(1) Interpolation = choice(1,"Nearest neighbor","Linear") Borders = choice(1,"Black","Nearest","Repeat")		`gmic mire.png -gimp_custom_deformation (w+h)/20*cos(y*20/h),(w+h)/20*sin(x*20/w),1,1,1 -o gimp_custom_deformation.png`	

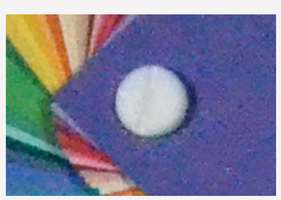
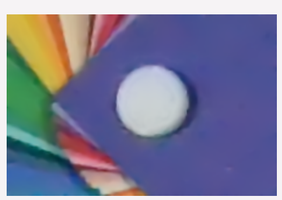
-gimp_transform_polar Preset = choice("Custom transform","Inverse radius","Swap radius/angle") X-center = float(50,0,100) Y-center = float(50,0,100) Radius = text{"r + R/10*cos(a*5)"} Angle = text{"a"} Borders = choice(1,"Black","Nearest","Repeat")		`Preset = Inverse radius` `gmic mire.png -gimp_transform_polar 1,50,50,r+R/10*cos(a*5),a,1 -o gimp_transform_polar.png`	

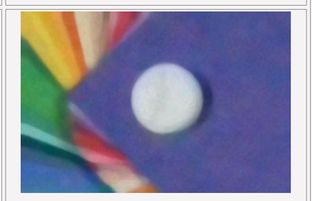
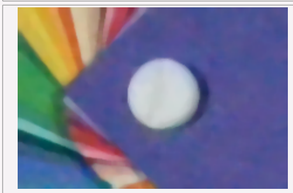
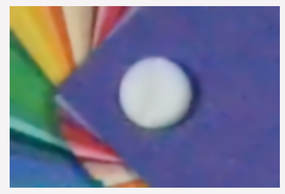
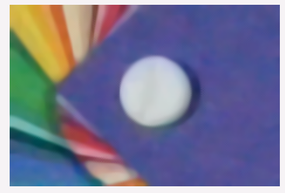
Degradations

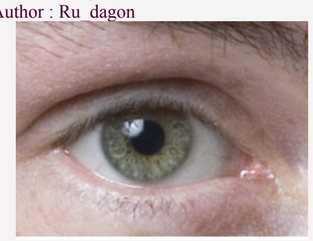
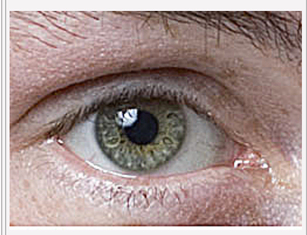
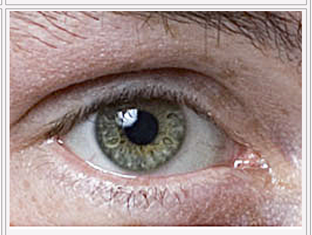
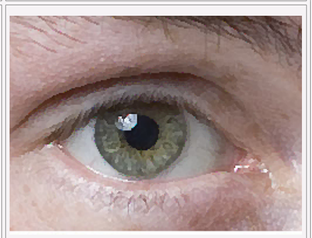
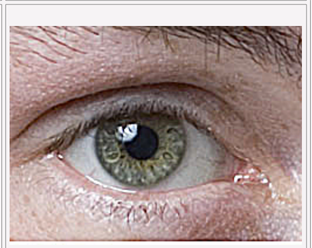
Commande et paramètres de la ligne de commande.	Image(s) d'origine	Ligne de commande	Résultat
-gimp_noise Amplitude = float(10,0,200) Noise type = choice("Gaussian","Uniform","Salt and pepper","Poisson") Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component") Value range = choice("Cut","Normalize")		`gmic mire.png -gimp_noise 180,0,0,0 -o gimp_noise.png`	
-gimp_spread X-variations = float(4,0,20) Y-variations = float(4,0,20) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic mire.png -gimp_spread 4,4,0 -o gimp_spread.png`	
-gimp_shade_stripes Frequency = float(30,1,100) Orientation = choice(1,"Horizontal","Vertical") Darkness = float(0.8,0,3) Lightness = float(1.3,0,3) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic mire.png -gimp_shade_stripes 70,1,0.8,1.3,0 -o gimp_shade_stripes.png`	
-gimp_stripes_y Frequency = float(10,0,100) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")	mire avec la partie centrale transparente 	`gmic mire.png -gimp_stripes_y 80,0 -o gimp_stripes_y.png`	
-gimp_watermark_visible Text = text("\251 G'MIC") Opacity = float(0.4,0.1,0.9) Size = int(57,13,128) Angle = float(25,0,360) Lightness = choice(1,"Darker","Brighter")		`gmic mire.png -gimp_watermark_visible "ABC",0.85,24,315,1 -o gimp_watermark_visible.png`	
-gimp_gaussian_blur XY-amplitude = float(3,0,20) X-amplitude = float(0,0,20) Y-amplitude = float(0,0,20) Border conditions = choice(1,"Black","Nearest") Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component") Value range = choice("Cut","Normalize")		`gmic mire.png -gimp_gaussian_blur 3,0,0,1,0,1 -o gimp_gaussian_blur.png`	
-gimp_blur_linear Tangent radius = float(10,0,100) Orthogonal radius = float(0.5,0,100) Angle = float(0,0,180) Border conditions = choice(1,"Black","Nearest") Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances", "Blue chrominance","Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component") Value range = choice("Cut","Normalize")		`gmic mire.png -gimp_blur_linear 10,0.5,0,1,0,1 -o gimp_blur_linear.png`	

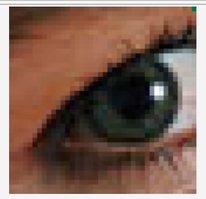
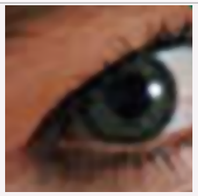
-gimp_blur_radial Amplitude = float(3,0,20) X-center = float(0.5,0,1) Y-center = float(0.5,0,1) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances", "Blue chrominance","Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component") Value range = choice("Cut","Normalize")		gmic mire.png -gimp_blur_radial 3,0.5,0.5,0,1 -o gimp_blur_radial.png	
-gimp_blur_angular Amplitude = float(2,0,10) X-center = float(0.5,0,1) Y-center = float(0.5,0,1) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances", "Blue chrominance","Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component") Value range = choice("Cut","Normalize")		gmic mire.png -gimp_blur_angular 2,0.5,0.5,0,1 -o gimp_blur_angular.png	
-gimp_bandpass Low frequency = float(0,0,100) High frequency = float(100,0,100) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances", "Blue chrominance","Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component") Value range = choice(1,"Cut","Normalize")		gmic mire.png -gimp_bandpass 1,100,0,1 -o gimp_bandpass.png	
-rgb2bayer Starting pattern = choice(0,"Red-Green","Blue-Green","Green-Red","Green-Blue") Keep colors = bool(1)		gmic mire.png -rgb2bayer 0,1 -o gimp_rgb2bayer.png	
-bayer2rgb G/M smoothness = _float(6,0,20) R/B smoothness (principal) = _float(6,0,20) R/B smoothness (secondary) = _float(4,0,20)		gmic gimp_rgb2bayer.png -bayer2rgb 6,6,4 -o gimp_bayer2rgb.png	
-gimp_8bits Scale = float(25,1,100) Dithering = float(800,0,10000) Levels = int(16,2,256) Preview type = choice("Full","Forward horizontal","Forward vertical","Backward horizontal","Backward vertical")		gmic geo.png -gimp_8bits 25,800,16,1 -o gimp_8bits.png	

Enhancement

Commande et paramètres de la ligne de commande.	Image(s) d'origine	Ligne de commande	Résultat
-gimp_anisotropic_smoothing Amplitude = float(60,0,1000) Sharpness = float(0.7,0,2) Anisotropy = float(0.3,0,1) Gradient smoothness = float(0.6,0,10) Tensor smoothness = float(1.1,0,10) Spatial precision = float(0.8,0.1,2) Angular precision = float(30,1,180) Value precision = float(2,0.1,5) Interpolation = choice(0,"Nearest neighbor","Linear","Runge-Kutta") Fast approximation = bool(1) Iterations = int(1,1,10) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component") Tiles = int(1,1,10)		Remarques : - Ce filtre remplace l'ancien GREYCstoration dans Gimp, il permet de diminuer le bruit et de créer des effets à base de flou. - Un moyen simple pour diminuer le bruit est d'augmenter le nombe "Iterations", ici = 3 gmic bruit.png -gimp_anisotropic_smoothing 60,0.7,0.3,0.6,1.1,0.8,30,2,0,1,3,2,1 -o gimp_anisotropic_smoothing.png	

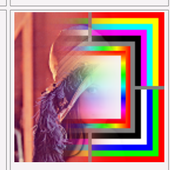
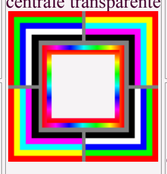
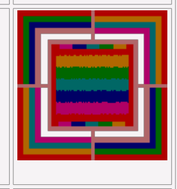
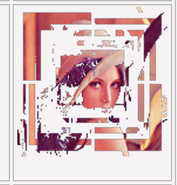
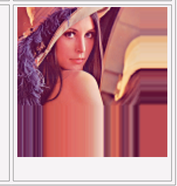
-gimp_patch_smoothing Spatial variance = float(10,0.1,200) Patch variance = float(10,0.1,200) Patch size = int(3,2,21) Lookup size = int(5,2,21) Patch smoothness = float(0,0,4) Fast approximation = bool(1) Iterations = int(1,1,10) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic bruit.png -gimp_patch_smoothing 10,10,3,5,0,1,4,2 -o gimp_patch_smoothing.png`	
-gimp_bilateral Spatial variance = float(10,0,100) Value variance = float(7,0,100) Iterations = int(2,1,10) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic bruit.png -gimp_bilateral 10,15,2,2 -o gimp_bilateral.png`	
-gimp_remove_hotpixels Mask size = int(3,3,20) Threshold = float(10,0,200)		`gmic bruit.png -gimp_remove_hotpixels 10,7 -o gimp_remove_hotpixels.png`	
-gimp_median Radius = int(3,1,20) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic bruit.png -gimp_median 7,2 -o gimp_median.png`	
-gimp_edgepreserving_smoothing Sharpness = float(0.7,0,2) Anisotropy = float(0.3,0,1) Gradient smoothness = float(0.6,0,10) Tensor smoothness = float(1.1,0,10) Time step = float(15,5,50) Iterations = int(8,1,100) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic bruit.png -gimp_edgepreserving_smoothing 0.7,0.3,0.6,1.1,15,30,2 -o gimp_edgepreserving_smoothing.png`	
-gimp_meancurvature_smoothing Time step = float(30,5,50) Iterations = int(4,1,10) Keep iterations as different layers = bool(false) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic bruit.png -gimp_meancurvature_smoothing 50,9,0,2 -o gimp_meancurvature_smoothing.png`	
-gimp_tv_smoothing Time step = float(30,5,100) Iterations = int(10,1,40) Keep iterations as different layers = bool(false) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic bruit.png -gimp_tv_smoothing 30,25,0,2 -o gimp_tv_smoothing.png`	

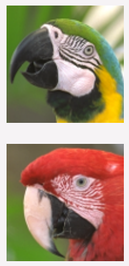
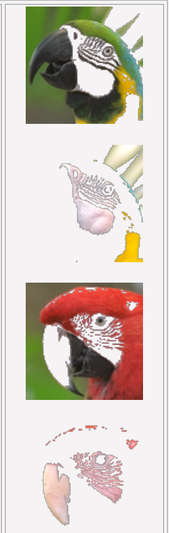
-gimp_unsharp Sharpening type = choice(0,"Gaussian","Bilateral") Spatial radius = float(1.25,0,20) Bilateral radius = float(30,0,60) Amount = float(3,0,10) Threshold = float(0,0,20) Darkness level = float(1,0,4) Lightness level = float(1,0,4) Iterations = int(1,1,10) Negative effect = bool(0) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")	Source image : http://en.wikipedia.org/wiki/File:Unsharped_eye.jpg Date : 17 May 2007 Author : Ru_dagon 	`gmic Ru_dagon.png -gimp_unsharp 0,1.25,30,3,0,1,1,1,0,2 -o gimp_unsharp.png`	
-gimp_unsharp_octave Scales = int(4,1,10) Maximal radius = float(5,0,20) Amount = float(3,0,10) Threshold = float(0,0,255) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic Ru_dagon.png -gimp_unsharp_octave 4,5,3,0,2 -o gimp_unsharp_octave.png`	
-gimp_sharpen_inversediff Amplitude = float(50,1,300) Iterations = int(2,1,10) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic Ru_dagon.png -gimp_sharpen_inversediff 50,2,2 -o gimp_sharpen_inversediff.png`	
-gimp_sharpen_shock Amplitude = float(150,1,400) Edge threshold = float(0.1,0,0.7) Gradient smoothness = float(0.8,0,10) Tensor smoothness = float(1.1,0,10) Iterations = int(1,1,10) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic Ru_dagon.png -gimp_sharpen_shock 150,0.1,0.8,1.1,1,2 -o gimp_sharpen_shock.png`	
-gimp_richardson_lucy Radius = float(2,0,20) Iterations = int(10,0,100) Time step = float(20,0,50) Smoothness = float(0.1,0,10) Regularization = choice(1,"Tikhonov","Mean curvature","Total variation") Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic Ru_dagon.png -gimp_richardson_lucy 2,10,20,0.1,1,2 -o gimp_richardson_lucy.png`	
-deinterlace Algorithm = choice("Standard","Motion-compensated")		Filtre pour images vidéo entrelacées. `gmic v.png -deinterlace 1 -o deinterlace.png`	
-red_eye Threshold = float(75,0,100) Smoothness = float(3.5,0,20) Factor = float(0.1,0,1)		`gmic yr.png -red_eye 75,3.5,0.1 -o red_eye_2.png`	

-gimp_scalenx Scaling factor = choice("x 2","x 3","x 4","x 6","x 8","x 9","x 12","x 16","x 18","x 27")		`gmic 40_40.png -gimp_scalenx 3 -o gimp_scalenx.png`	
-gimp_upscale_smart Width = text("200%") Height = text("200%") Smoothness = float(2,0,20) Anisotropy = float(0.4,0,1) Sharpness = float(10,0,100)		`gmic 40_40.png -gimp_upscale_smart 400%,400%,2,0.4,10 -o gimp_upscale_smart.png`	
-gimp_solidify This filter replaces transparent regions by morphologically interpolated colors. It may take long to render !		`gmic geo_trans.png -gimp_solidify`	Rendu identique à la commande -solidify Cliquer ici ⇨

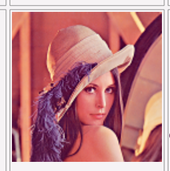
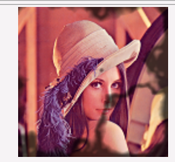
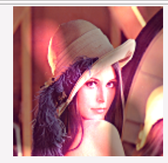
Layers

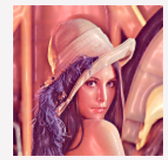
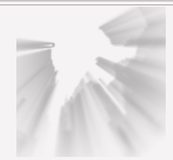
Pour plus de descriptions de filtres voir : Mélanges d'images (fonctions "compose") ⇨

Commande et paramètres de la ligne de commande.	Image(s) d'origine	Ligne de commande	Résultat
-gimp_compose_fade Preset = choice{1,"Custom","Linear","Circular","Wave","Keftales"} Offset = float(0,-1,1) Thinness = float(0,0,10) Sharpness = float(5,1,20) Sharpest = bool(0) Revert layers = bool(0) 1st parameter = float(0,-1,1) 2nd parameter = float(0,-1,1) 3rd parameter = float(0,-1,1) Formula = text{"cos(4*pi*x/w) * sin(4*pi*y/h)"}		`gmic geo.png mire.png -gimp_compose_fade 1,0,0,5,0,0,0,0,0,cos(4*pi*x/w)*sin(4*pi*y/h) -o gimp_compose_fade.png`	
		`gmic geo.png mire.png -gimp_compose_fade 1,0,0,5,0,0,0.5,0.5,0,0 -o gimp_compose_fade2.png`	
-gimp_compose_shapeaverage Preserve shading = bool(1) Transparency = bool(0)	mire avec la partie centrale transparente 	`gmic mire.png geo.png -gimp_compose_shapeaverage 0,1 -o gimp_compose_shapeaverage.png`	
-gimp_transparent_diff Threshold = float(1,0,100) Smoothness = float(0,0,10) Opaque pixels = choice(0,"From 1st layer","From 2nd layer") Transparent pixels = choice(1,"From same values","From different values")		`gmic geo.png mire.png -gimp_transparent_diff 50,0,0,1 -o gimp_transparent_diff.png`	
-gimp_align_layers Alignment type = choice(0,"Rigid","Non-rigid") Smoothness = float(0.7,0,1) Scales = choice(0,"Auto","1","2","3","4","5","6","7","8")		`gmic mire.png geo.png -gimp_align_layers 0,0.1,0 -o[-1] gimp_align_layers.png`	

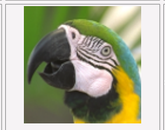
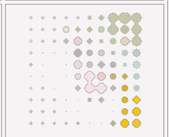
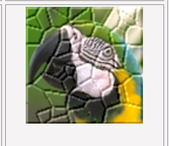
-gimp_split_tones Number of tones = int(3,2,10)		`gmic m1.png m2.png -gimp_split_tones 2 -o gimp_split_tones.png`	
-gimp_morph Frames = _int(10,2,100) Smoothness = _float(0.2,0,2) Precision = _float(0.1,0,2)		`gmic m1.png m2.png -gimp_morph 30,0.2,0.1 -o gimp_morph.png`	Résultat des 30 images assemblées en gif animé

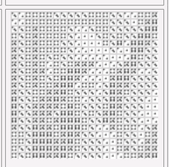
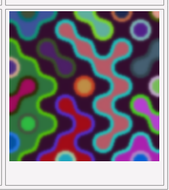
Lights & Shadows

Commande et paramètres de la ligne de commande.	Image d'origine	Ligne de commande	Résultat
-gimp_drop_shadow X-shadow = float(3,-20,20) Y-shadow = float(3,-20,20) Smoothness = float(1.8,0,5) Angle = float(0,0,360)		`gmic geo.png -gimp_drop_shadow 3,3,1.8,30 -o gimp_drop_shadow.png`	
-gimp_shadow_patch Opacity = float(0.7,0,1) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic geo.png -gimp_shadow_patch 0.3,2 -o gimp_shadow_patch.png`	
-gimp_light_patch ensity = int(5,2,30) Darkness = float(0.7,0,1) Lightness = float(2.5,1,4) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic geo.png -gimp_light_patch 5,0.7,2.5,2 -o gimp_light_patch.png`	

-gimp_light_relief Ambient lightness = float(0.3,0,5) Specular lightness = float(0.2,0,2) Specular size = float(0.2,0,1) Darkness = float(0,0,1) Light smoothness = float(0.5,0,5) X-light = float(0.5,0,1) Y-light = float(0.5,0,1) Z-light = float(5,0,20) Z-scale = float(0.5,0,3) Opacity as bumpmap = bool(0) Image smoothness = float(0,0,10)		`gmic geo.png -gimp_light_relief 0.3,0.2,0.2,0,2,0.5,0.5,5,0.5,0,0 -o gimp_light_relief.png`	
-gimp_lightrays Density = float(80,0,100) X-center = float(0.5,0,1) Y-center = float(0.5,0,1) Length = float(1,0,1) Attenuation = float(0.5,0,1) Transparency = bool(0)		`gmic geo.png -gimp_lightrays 53.68,0.35,0.25,0.02,0.11,1 -o gimp_lightrays.png`	

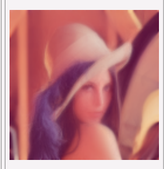
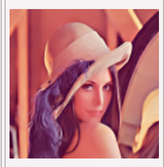
Patterns

Commande et paramètres de la ligne de commande.	Image d'origine	Ligne de commande	Résultat
-gimp_stencil Radius = float(3,0,10) Smoothness = float(0,0,30) Iterations = int(8,1,100) Aliasing = float(0,0,5) Stencil type = choice(0,"Black & White","Shaded","Color") Transparency = bool(0)		`gmic m1.png -gimp_stencil 3,0,8,0,2,0 -o gimp_stencil.png`	
-gimp_dots Number of scales = int(10,1,20) Resolution = float(10,1,100) Radius = float(3,0.1,10) Stencil type = choice(0,"Black & White","Shaded","Color") Transparency = bool(0)		`gmic m1.png -gimp_dots 30,10,3,3,1 -o gimp_dots.png`	
-gimp_puzzle Scale = float(6,1,20) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic m1.png -gimp_puzzle 6,2 -o gimp_puzzle.png`	
- Density = float(1,0.1,10) Keep edges = bool(true) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic m1.png -gimp_mosaic 1,1,2 -o gimp_mosaic.png`	
-gimp_cracks Density = float(1,0.1,10) Amplitude = float(-80,-255,255) Relief = bool(true) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		`gmic m1.png -gimp_cracks 1,-80,1,2 -o gimp_cracks.png`	

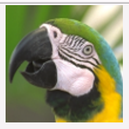
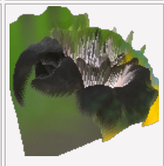
-gimp_whirls Density = int(7,3,20) Smoothness = float(2,0,10) Darkness = float(0.2,0,1) Lightness = float(1.8,1,3) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		gmic m1.png -gimp_whirls 7,2,0.2,1.8,2 -o gimp_whirls.png	
-gimp_paper Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		gmic m1.png -gimp_paper 2 -o gimp_paper.png	
-gimp_hearts Density = float(10,0,100) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		gmic m1.png -gimp_hearts 10,2 -o gimp_hearts.png	
-gimp_sponge Size = int(13,3,21) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		gmic m1.png -gimp_sponge 13,2 -o gimp_sponge.png	
-gimp_canvas Amplitude = float(70,0,300) Angle = float(45,0,180) Sharpness = float(400,0,2000) Activate second direction = bool(true) Amplitude = float(70,0,300) Angle = float(135,0,180) Sharpness = float(400,0,2000)		gmic m1.png -gimp_canvas 70,45,400,1,70,135,400 -o gimp_canvas.png	
-gimp_dices Resolution = float(2,1,10) Size = int(24,8,64) Color model = choice("Black dices","White dices")		gmic geo.png -gimp_dices 1,6,1 -o gimp_dices.png	
-gimp_plaid_texture Line = float(50,0,100) Number of angles = int(2,1,8) Starting angle = float(0,0,360) Angle range = float(90,0,360) Smoothness = float(1,0,5) Sharpen = float(300,0,1000)		gmic geo.png -gimp_plaid_texture 30,2,30,90,3,400 -o gimp_plaid_texture.png	
-gimp_truchet Scale = int(32,1,256) Radius = int(5,1,64) Smoothness = float(1,0,10) Type = choice(1,"Straight","Curved") Colorize randomly = bool(0)	sans	gmic 128,128,1,4 -gimp_truchet 16,3,1,1,1 -n 0,225 -o gimp_truchet.png	

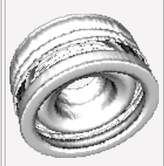
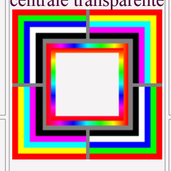
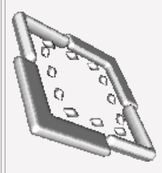
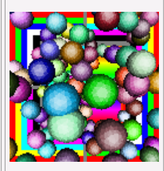
Presets

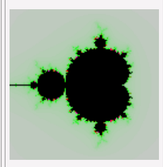
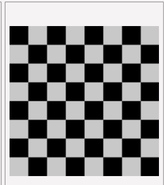
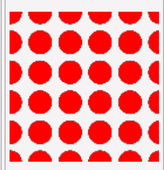
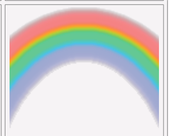
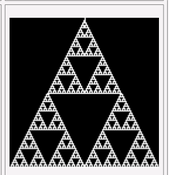
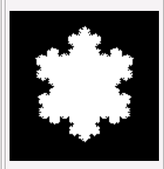
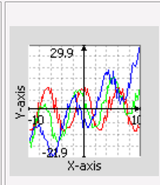
Commande et paramètres de la ligne de commande.	Image d'origine	Ligne de commande	Résultat

-gimp_anisotropic_smoothing Amplitude = float(60,0,1000) Sharpness = float(0.16,0,2) Anisotropy = float(0.63,0,1) Gradient smoothness = float(0.6,0,10) Tensor smoothness = float(2.35,0,10) Spatial precision = float(0.8,0.1,2) Angular precision = float(30,1,180) Value precision = float(2,0.1,5) Interpolation = choice(0,"Nearest neighbor","Linear","Runge-Kutta") Fast approximation = bool(1) Iterations = int(1,1,10) Channel(s) = choice("RGB","Luminance","Blue & Red chrominances","Blue chrominance","Red chrominance") Tiles = int(1,1,10)		PhotoComiX smoothing gmic geo.png -gimp_anisotropic_smoothing 60,0.16,0.63,0.6,2.35,0.8,30,2,0,1,1,1,1 -o PhotoComiX_smoothing.png	
-gimp_anisotropic_smoothing Amplitude = float(60,0,1000) Sharpness = float(0.9,0,2) Anisotropy = float(0.64,0,1) Gradient smoothness = float(3.1,0,10) Tensor smoothness = float(1.10,0,10) Spatial precision = float(0.8,0.1,2) Angular precision = float(30,1,180) Value precision = float(2,0.1,5) Interpolation = choice(0,"Nearest neighbor","Linear","Runge-Kutta") Fast approximation = bool(1) Iterations = int(1,1,10) Channel(s) = choice("RGB","Luminance","Blue & Red chrominances","Blue chrominance","Red chrominance") Tiles = int(1,1,10)		Thin brush smoothing gmic geo.png -gimp_anisotropic_smoothing 60,0.9,0.64,3.1,1.1,0.8,30,2,0,1,1,1,1 -o Thin_brush_smoothing.png	
-gimp_lylejk_stencil Amplitude = int(5,1,10) Sharpness = float(10,0,100) Radius = float(3,0,10) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances","Blue chrominance", "Red chrominance","Lightness","ab-components","a-component","b-component","Hue", "Saturation","Value","Key","Alpha","ch-components","c-component","h-component")		gmic geo.png -gimp_lylejk_stencil 5,10,3,2 -o gimp_lylejk_stencil.png	

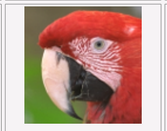
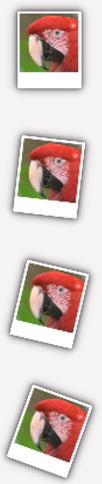
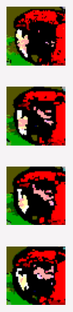
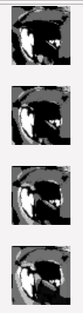
Rendering

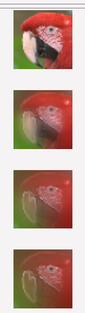
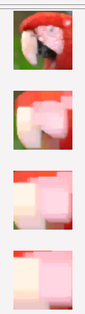
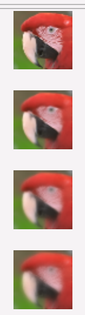
Commande et paramètres de la ligne de commande.	Image(s) d'origine	Ligne de commande	Résultat
-gimp_elevation3d Factor = float(100,-1000,1000) Smoothness = float(1,0,10) Width = _int(1024,8,4096) Height = _int(1024,8,4096) Size = float(0.8,0,3) X-angle = float(25,0,360) Y-angle = float(0,0,360) Z-angle = float(21,0,360) FOV = float(45,1,90) X-light = float(0,-100,100) Y-light = float(0,-100,100) Z-light = float(-100,-100,0) Specularity = float(0.5,0,1) Shininess = float(0.7,0,3) Rendering = choice(2,"Dots","Wireframe","Flat","Flat shaded","Gouraud","Phong")		gmic m1.png -gimp_elevation3d 100,1,1024,1024,0.8,25,0,21,45,0,0,-100,0.5,0.7,2 -resize 128,128 -o gimp_elevation3d.png	

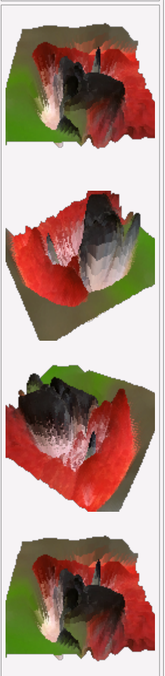
-gimp_imageobject3d Type = choice{1,"Plane","Cube","Pyramid","Sphere","Torus","Gyroid","Weird","Cup"} Width = _int(1024,1,4096) Height = _int(1024,1,4096) Size = float(0.5,0,3) X-angle = float(57,0,360) Y-angle = float(41,0,360) Z-angle = float(21,0,360) FOV = float(45,1,90) X-light = float(0,-100,100) Y-light = float(0,-100,100) Z-light = float(-100,-100,0) Specularity = float(0.5,0,1) Shininess = float(0.7,0,3) Rendering = choice(4,"Dots","Wireframe","Flat","Flat shaded","Gouraud","Phong")		`gmic m1.png -gimp_imageobject3d 1,152,152,0.5,57,41,21,45,0,0,-100,0.5,0.7,4 -autocrop 0 -o gimp_imageobject3d.png`	
-gimp_lathing3d Resolution = int(76,1,1024) Smoothness = float(0.6,0,3) Max angle = float(361,0,361) Width = _int(1024,1,4096) Height = _int(1024,1,4096) Size = float(0.5,0,3) X-angle = float(57,0,360) Y-angle = float(41,0,360) Z-angle = float(21,0,360) FOV = float(45,1,90) X-light = float(0,-100,100) Y-light = float(0,-100,100) Z-light = float(-100,-100,0) Specularity = float(0.5,0,1) Shininess = float(0.7,0,3) Rendering = choice(4,"Dots","Wireframe","Flat","Flat shaded","Gouraud","Phong")		`gmic m1.png -gimp_lathing3d 76,0.6,361,232,232,0.5,57,41,21,45,0,0,-100,0.5,0.7,4 -autocrop 0 -o gimp_lathing3d.png`	
-gimp_extrude3d Depth = float(10,1,256) Resolution = int(512,1,1024) Smoothness = float(0.6,0,3) Width = _int(1024,1,4096) Height = _int(1024,1,4096) Size = float(0.5,0,3) X-angle = float(57,0,360) Y-angle = float(41,0,360) Z-angle = float(21,0,360) FOV = float(45,1,90) X-light = float(0,-100,100) Y-light = float(0,-100,100) Z-light = float(-100,-100,0) Specularity = float(0.5,0,1) Shininess = float(0.7,0,3) Rendering = choice(4,"Dots","Wireframe","Flat","Flat shaded","Gouraud","Phong")	mire avec la partie centrale transparente 	`gmic mire.png -gimp_extrude3d 10,512,0.6,291,291,0.5,57,41,21,45,0,0,-100,0.5,0.7,4 -autocrop 0 -o gimp_extrude3d.png`	
-gimp_random3d Type = choice("Cube","Cone","Cylinder","Sphere","Torus") Density = int(50,1,300) Size = float(3,1,20) Z-range = float(100,0,300) FOV = float(45,1,90) X-light = float(0,-100,100) Y-light = float(0,-100,100) Z-light = float(-100,-100,0) Specularity = float(0.5,0,1) Shininess = float(0.7,0,3) Rendering = choice(3,"Dots","Wireframe","Flat","Flat shaded","Gouraud","Phong") Opacity = float(1,0,1)		`gmic mire.png -gimp_random3d 3,140,6,100,45,0,0,-100,0.5,0.7,3,1 -o gimp_random3d.png`	

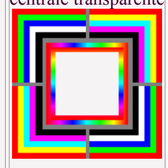
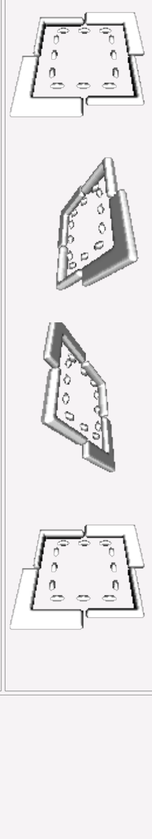
-gimp_mandelbrot X-center = float(0,-2,2) Y-center = float(0,-2,2) Zoom = float(0,0,100) Iterations = int(128,1,512) Fractal set = choice(Mandelbrot, Julia) X-seed (Julia) = text("0.317") Y-seed (Julia) = text("0.03") Color 1 = color(0,0,255) Color 2 = color(0,255,0) Color 3 = color(255,0,0) Color 4 = color(255,0,255)	sans	gmic 128,128,1,3 -gimp_mandelbrot -0.4,0,0.5,128,0,0.317,0.03,200,200,200,0,255,0,255,0,0,255,0,255 -o gimp_mandelbrot.png	
-gimp_chessboard First size = int(64,1,512) Second size = int(64,1,512) First offset = int(0,0,512) Second offset = int(0,0,512) Angle = float(0,0,180) Opacity = float(0.5,0,1) First color = color(0,0,0,255) Second color = color(255,255,255,255)	sans	gmic 128,128,1,4 -gimp_chessboard 16,16,0,0,0,1,0,0,0,255,200,200,200,255 -o gimp_chessboard.png	
-gimp_polka_dots Size = float(80,0,100) Density = float(20,0.1,100) First offset = float(50,0,100) Second offset = float(50,0,100) Angle = float(0,0,180) Aliasing = float(0.5,0.1,1) Shading = float(0.1,0.1,1) Opacity = float(1,0,1) Color = color(255,0,0,255)	sans	gmic 128,128,1,4 -gimp_polka_dots 80,20,50,50,0,0.5,0.1,1,255,0,0,255 -o gimp_polka_dots.png	
-gimp_rainbow Left position = float(80,0,100) Right position = float(80,0,100) Left slope = float(175,0,400) Right slope = float(175,0,400) Thinness = float(3,0.1,8) Opacity = float(80,0,199)	sans	gmic 128,128,1,4 -gimp_rainbow 80,80,175,175,1.4,100 -o gimp_rainbow.png	
-sierpinski Recursions = int(6,0,10) 1st X-coord = float(50,0,100) 1st Y-coord = float(0,0,100) 2nd X-coord = float(0,0,100) 2nd Y-coord = float(100,0,100) 3rd X-coord = float(100,0,100) 3rd Y-coord = float(100,0,100)	sans	gmic 128,128,1,3 -sierpinski 6,50,0,0,100,100,100 -o gimp_sierpinski.png	
-snowflake Recursions = int(3,0,5) 1st X-coord = float(20,0,100) 1st Y-coord = float(70,0,100) 2nd X-coord = float(80,0,100) 2nd Y-coord = float(70,0,100) 3rd X-coord = float(50,0,100) 3rd Y-coord = float(10,0,100) Opacity = float(1,0,1) Color = color(255,255,255)	sans	gmic 128,128,1,3 -snowflake 3,20,70,80,70,50,10,1,255,255,255 -o gimp_snowflake.png	
-gimp_equation_plot Equation = text{"X*c+10*cos(X+c+?)"} X-min = float(-10,-100,100) X-max = float(10,-100,100) Resolution = int(100,2,1024) Channels = int(3,1,32) Plot type = choice(2,"None","Lines","Splines","Bars") Vertex type = choice(0,"None","Points","Crosses 1","Crosses 2","Circles 1","Circles 2","Square 1","Square 2")	sans	gmic 128,128,1,3 -gimp_equation_plot X*c+10*cos(X+c+?),-10,10,100,3,2,0 -o gimp_equation_plot.png	
-gimp_maze Cell size = int(24,1,256) Thickness = int(1,1,10) Masking = choice("None","Render on dark areas","Render on white areas") Preserve image dimension = bool(1) Maze type = choice("Dark walls","White walls")	sans	gmic 128,128,1,3 -gimp_maze 9,2,0,1,1 -o gimp_maze.png	

Sequences

Commande et paramètres de la ligne de commande.	Image(s) d'origine	Ligne de commande	Résultat
-gimp_animate_polaroid Frames = _int(10,2,100) Output frames = _bool(1) Output files = _bool(0) Output folder = _folder() Frame size = int(10,1,400) Bottom size = int(20,1,400) X-shadow = float(0,-20,20) Y-shadow = float(0,-20,20) Smoothness = float(3,0,5) Angle = float(0,0,360) Zoom = float(1,0.01,1) Frame size = int(10,1,400) Bottom size = int(20,1,400) X-shadow = float(0,-20,20) Y-shadow = float(0,-20,20) Smoothness = float(3,0,5) Angle = float(20,0,360) Zoom = float(1,0.01,1)		`gmic m2.png -gimp_animate_polaroid 4,1,0,toto,10,20,0,0,3,0,1,10,20,0,0,3,20,1 -resize 50%,50% -o gimp_animate_polaroid.png`	
-gimp_animate_edges Frames = _int(10,2,100) Output frames = _bool(1) Output files = _bool(0) Output folder = _folder() Negative colors = bool(0) Smoothness = float(0,0,10) Edge threshold = float(10,0,30) Smoothness = float(0,0,10) Edge threshold = float(30,0,30)		`gmic m2.png -gimp_animate_edges 4,1,0,toto,0,0,10,0,30 -resize 50%,50% -o gimp_animate_edges.png`	
-gimp_animate_cartoon Frames = _int(10,2,100) Output frames = _bool(1) Output files = _bool(0) Output folder = _folder() Color quantization = int(4,2,256) Smoothness = float(0.5,0,2) Sharpening = float(200,0,400) Edge threshold = float(10,1,30) Edge thickness = float(0.1,0,1) Color strength = float(1.5,0,3) Smoothness = float(3,0,2) Sharpening = float(200,0,400) Edge threshold = float(10,1,30) Edge thickness = float(0.1,0,1) Color strength = float(1.5,0,3)		`gmic m2.png -gimp_animate_cartoon 4,1,0,toto,4,0.5,200,10,0,1.5,3,200,10,0.1,1.5 -resize 50%,50% -o gimp_animate_cartoon.png`	
-gimp_animate_stencilbw Frames = _int(10,2,100) Output frames = _bool(1) Output files = _bool(0) Output folder = _folder() Edge threshold = float(10,0,30) Smoothness = float(10,0,30) Edge threshold = float(10,0,30) Smoothness = float(20,0,30)		`gmic m2.png -gimp_animate_stencilbw 4,1,0,toto,10,10,10,20 -resize 50%,50% -o gimp_animate_stencilbw.png`	

-gimp_animate_pencilbw Frames = _int(10,2,100) Output frames = _bool(1) Output files = _bool(0) Output folder = _folder() Pencil type = float(2.3,0,5) Amplitude = float(100,0,200) Pencil type = float(0.3,0,5) Amplitude = float(60,0,200)		`gmic m2.png -gimp_animate_pencilbw 4,1,0,toto,2.3,100,0.3,60 -resize 50%,50% -o gimp_animate_pencilbw.png`	
-gimp_animate_glow Frames = _int(10,2,100) Output as frames = _bool(1) Output as files = _bool(0) Output folder = _folder() Amplitude = float(0,0,8) Amplitude = float(3,0,8)		`gmic m2.png -gimp_animate_glow 4,1,0,toto,0,30 -resize 50%,50% -o gimp_animate_glow.png`	
-gimp_animate_morpho Frames = _int(10,2,100) Output as frames = _bool(1) Output as files = _bool(0) Output folder = _folder() Action = choice("Erosion","Dilation","Opening","Closing","Original - Erosion", "Dilation - Original","Original - Opening","Closing - Original") Invert colors = bool(false) Shape = choice(0,"Square","Octagonal","Circular") Size = int(5,1,100) Size = int(50,2,100)		`gmic m2.png -gimp_animate_morpho 4,1,0,toto,1,0,0,5,50 -resize 50%,50% -o gimp_animate_morpho.png`	
-gimp_animate_anisotropic_smoothing Frames = _int(10,2,100) Output as frames = _bool(1) Output as files = _bool(0) Output folder = _folder() Spatial precision = float(0.8,0.1,2) Angular precision = float(30,1,180) Value precision = float(2,0.1,5) Interpolation type = choice(0,"Nearest neighbor","Linear","Runge-Kutta") Fast approximation = bool(1) Iterations = int(1,1,10) Channel(s) = choice("All","RGBA","RGB","Luminance","Blue/red chrominances", "Blue chrominance","Red chrominance","Alpha") Tile subdivisions = int(1,1,10) Amplitude = float(60,0,1000) Sharpness = float(0.7,0,2) Anisotropy = float(0.3,0,1) Gradient smoothness = float(0.6,0,10) Tensor smoothness = float(1.1,0,10) Amplitude = float(60,0,1000) Sharpness = float(0.7,0,2) Anisotropy = float(0.3,0,1) Gradient smoothness = float(0.6,0,10) Tensor smoothness = float(1.1,0,10)		`gmic m2.png -gimp_animate_anisotropic_smoothing 4,1,0,toto,0.8,30,2,0,1,1,2,1,60,0.7,0.3,0.6,1.1,60,0.7,0.3,20,1.1 -resize 50%,50% -o gimp_animate_anisotropic_smoothing.png`	

-gimp_animate_imageobject3d Frames = _int(10,2,100) Output as frames = _bool(1) Output as files = _bool(0) Output folder = _folder() Type = choice{1,"Plane","Cube","Pyramid","Sphere", "Torus","Gyroid","Weird","Cup"} Width = _int(1024,1,4096) Height = _int(1024,1,4096) Rendering = choice(4,"Dots","Wireframe","Flat","Flat shaded","Gouraud","Phong") Size = float(0.5,0,3) X-angle = float(57,0,360) Y-angle = float(41,0,360) Z-angle = float(21,0,360) FOV = float(45,1,90) X-light = float(0,-100,100) Y-light = float(0,-100,100) Z-light = float(-100,-100,0) Specularity = float(0.5,0,1) Shininess = float(0.7,0,3) Size = float(0.5,0,3) X-angle = float(57,0,1440) Y-angle = float(401,0,1440) Z-angle = float(21,0,1440) FOV = float(45,1,90) X-light = float(0,-100,100) Y-light = float(0,-100,100) Z-light = float(-100,-100,0) Specularity = float(0.5,0,1) Shininess = float(0.7,0,3)		`gmic m2.png -gimp_animate_imageobject3d 4,1,0,toto,1,128,128,4,0.5,57,41,21,45,0,0,-100,0.5,0.7,0.5,57,401,21,45,0,0,-100,0.5,0.7 -autocrop 0 -o gimp_animate_imageobject3d.png`	
-gimp_animate_elevation3d Frames = _int(10,2,100) Output as frames = _bool(1) Output as files = _bool(0) Output folder = _folder() Factor = float(100,-1000,1000) Smoothness = float(1,0,10) Width = _int(1024,8,4096) Height = _int(1024,8,4096) Rendering = choice(2,"Dots","Wireframe","Flat","Flat shaded","Gouraud","Phong") Size = float(0.8,0,3) X-angle = float(35,0,360) Y-angle = float(0,0,360) Z-angle = float(0,0,360) FOV = float(45,1,90) X-light = float(0,-100,100) Y-light = float(0,-100,100) Z-light = float(-100,-100,0) Specularity = float(0.5,0,1) Shininess = float(0.7,0,3) Size = float(0.8,0,3) X-angle = float(35,0,1440) Y-angle = float(0,0,1440) Z-angle = float(360,0,1440) FOV = float(45,1,90) X-light = float(0,-100,100) Y-light = float(0,-100,100) Z-light = float(-100,-100,0) Specularity = float(0.5,0,1) Shininess = float(0.7,0,3)		`gmic m2.png -gimp_animate_elevation3d 4,1,0,toto,100,1,128,128,2,0.8,35,0,0,45,0,0,-100,0.5,0.7,0.8,35,0,360,45,0,0,-100,0.5,0.7 -o gimp_animate_elevation3d.png`	

-gimp_animate_extrude3d Frames = _int(10,2,100) Output as frames = _bool(1) Output as files = _bool(0) Output folder = _folder() Depth = float(10,1,256) Resolution = int(512,1,1024) Smoothness = float(0.6,0,3) Width = _int(1024,8,4096) Height = _int(1024,8,4096) Rendering = choice(4,"Dots","Wireframe","Flat","Flat shaded","Gouraud","Phong") Size = float(0.8,0,3) X-angle = float(35,0,360) Y-angle = float(0,0,360) Z-angle = float(0,0,360) FOV = float(45,1,90) X-light = float(0,-100,100) Y-light = float(0,-100,100) Z-light = float(-100,-100,0) Specularity = float(0.5,0,1) Shininess = float(0.7,0,3) Size = float(0.8,0,3) X-angle = float(35,0,1440) Y-angle = float(360,0,1440) Z-angle = float(0,0,1440) FOV = float(45,1,90) X-light = float(0,-100,100) Y-light = float(0,-100,100) Z-light = float(-100,-100,0) Specularity = float(0.5,0,1) Shininess = float(0.7,0,3)	mire avec la partie centrale transparente 	gmic mire.png -gimp_animate_extrude3d 4,1,0,toto,10,512,0.6,128,128,4,0.8,35,0,0,45,0,0,-100,0.5,0.7,0.8,35,360,0,45,0,0,-100,0.5,0.7 -o gimp_animate_extrude3d.png	

Divers

Pour améliorer cette page sur G'MIC vous pouvez intervenir sur http://www.gimp-attitude.org/forum2/viewtopic.php?f=53&t=7024 ou nous contacter sur notre blog : http://samjcreations.blogspot.com à partir du libellé "G'MIC pour Gimp Windows" en haut à droite de la page.
Auteur : samj
Corrections et suggestions : zigomar, dtschump.
Versions :

Version 53	Modification gimp_map_sphere. Ajout de piechart.
Version 52	Corrections.
Version 51	Corrections 64bits . Ajout pointcloud3d .
Version 50	Corrections fisheye , version 64bits , segment_watershed , gimp_skeleton
Version 49	Notification version 64 bits Windows. Mise à jour des raccourcis. Mise à jour des commandes G'.MIC jusqu'à la version 1.5.0.9 (4 janvier 2012) faite avec version 64bits : rodilius , spherical3d , x_reflection3d , colormap , gimp_colormap , gimp_rodilius , gimp_dices , autoindex , solidify , gimp_solidify , x_fireworks , x_whirl , lightrays , gimp_ lightrays , gimp_8bits , superformula3d , truchet , gimp_truchet , compose_median , compose_divide circlism , gimp_color_abstraction , texturize_paper , x_rubber3d , gimp_lylejk_painting , texturize_canvas , gimp_metallic , maze , gimp_maze , ripple , x_shadebobs , fire_edges , gimp_fire_edges , x_blobs , x_minimal_path , kuwahara , gimp_kuwahara , gimp_plaid_texture , x_hough , -houghsketchbw .
Version 48	Ajout Dessin, peinture
Version 47	version 1.4.7.0 Ajouts : Lumière douce, imagesphere3d
Version 46	Ajout transfer_colors
Version 45	Ajout nombres aléatoires

Version 44	version 1.4.5.2
Version 43	Ajout fichier log & version 1.4.5.1.
Version 41	Ajout de quelques fonctions 3D et vidéo
Version 37	Version complétée du 10 novembre 2010
Version 1	Version d'origine du 28 octobre 2010 .

À faire :
-index , **-apply_pose3d** , [[gmic geo.png --histogram 256 --**cumul**[-1] -display_graph[-2,-1] 400,300,3]] ,
-apply_camera3d , **-gimp_superformula** , [**-tic** & **-toc**] , [**-area** & **-area_fg**] , **-output_pink3d** , **-replace_nan** ,
[**-min_patch** & **-max_patch**] , **-compose_alpha** , [**-tensor2eigen** & **-eigen2tensor**] , **-uncase** ,
-minimal_path , [**-rgb2srgb** & **-srgb2rgb**] , **-discard** , [**-otsu** http://en.wikipedia.org/wiki/Otsu's_method] ,
[**-hough** http://en.wikipedia.org/wiki/Hough_transform] ,

Les filtres avec trop de paramètres :
-gimp_novelfx , gimp_graphic_boost , gimp_vintage , gimp_ink_wash