



A Full-Featured Open-Source Framework for Image Processing

## The Handbook

**Version 3.0.0**

© David Tschumperlé / GREYC / CNRS

2021/12/9

# Preamble

- This document is distributed under the **GNU Free Documentation License**, version 1.3.
- A **.pdf version** of this document is available.

# Version

**G'MIC:** GREYC's Magic for Image Computing

<https://gmic.eu>

Version **3.0.0**

Copyright © Since 2008, **David Tschumperlé / GREYC / CNRS**

<https://www.greyc.fr>

# Table of Contents

- **Usage**
- **Overall Context**
- **Image Definition and Terminology**
- **Items of a Processing Pipeline**
- **Input Data Items**
- **Command Items and Selections**
- **Input/Output Properties**
- **Substitution Rules**
- **Mathematical Expressions**
- **Image and Data Viewers**
- **Adding Custom Commands**
- **List of Commands**
- **Examples of Use**

# Usage

```
gmic [command1 [arg1_1,arg1_2,...]] ... [commandN [argN_1,argN_2,...]]
```

**gmic** is the open-source interpreter of the **G'MIC** language, a script-based programming language dedicated to the design of possibly complex image processing pipelines and operators. It can be used to convert, manipulate, filter and visualize image datasets made of one or several 1D/2D or 3D multi-spectral images.

This reference documentation describes all the technical aspects of the G'MIC framework, in its current version **3.0.0**.

As a starting point, you may want to visit our detailed tutorial pages, at: <https://gmic.eu/tutorial/>

# Overall Context

- At any time, **G'MIC** manages one list of numbered (and optionally named) pixel-based images, entirely stored in computer memory (uncompressed).
- The first image of the list has index **0** and is denoted by **[0]**. The second image of the list is denoted by **[1]**, the third by **[2]** and so on.
- Negative indices are treated in a periodic way: **[-1]** refers to the last image of the list, **[-2]** to the penultimate one, etc. Thus, if the list has 4 images, **[1]** and **[-3]** both designate the second image of the list.
- A named image may be also indicated by **[name]**, if **name** uses the character set **[a-zA-Z0-9\_]** and does not start with a number. Image names can be set or reassigned at any moment during the processing pipeline (see command **name** for this purpose).
- G'MIC defines a set of various commands and substitution mechanisms to allow the design of complex pipelines and operators managing this list of images, in a very flexible way: You can insert or remove images in the list, rearrange image order, process images (individually or grouped), merge image data together, display and output image files, etc.
- Such a pipeline can define a new custom G'MIC command (stored in a user command file), and re-used afterwards as a regular command, in a larger pipeline if necessary.

## Image Definition and Terminology

- In **G'MIC**, each image is modeled as a 1D, 2D, 3D or 4D array of scalar values, uniformly discretized on a rectangular/parallelepipedic domain.
- The four dimensions of this array are respectively denoted by:
  - **width**, the number of image columns (size along the **x-axis**).
  - **height**, the number of image rows (size along the **y-axis**).
  - **depth**, the number of image slices (size along the **z-axis**). The depth is equal to **1** for usual color or grayscale 2D images.
  - **spectrum**, the number of image channels (size along the **c-axis**). The spectrum is respectively equal to **3** and **4** for usual **RGB** and **RGBA** color images.
- There are no hard limitations on the size of the image along each dimension. For instance, the number of image slices or channels can be of arbitrary size within the limits of the available memory.
- The **width**, **height** and **depth** of an image are considered as spatial dimensions, while the **spectrum** has a multi-spectral meaning. Thus, a 4D image in G'MIC should be most often regarded as a 3D dataset of multi-spectral voxels. Most of the G'MIC commands will stick with this idea (e.g. command **blur** blurs images only along the spatial **xyz**-axes).
- G'MIC stores all the image data as buffers of **float** values (32 bits, value range **[-3.4E38, +3.4E38]**). It performs all its image processing operations with floating point numbers. Each image pixel takes then 32bits/channel (except if double-precision buffers have been enabled during the compilation of the software, in which case 64bits/channel can be the default).
- Considering **float**-valued pixels ensure to keep the numerical precision when executing image processing pipelines. For image input/output operations, you may want to prescribe the image

datatype to be different than `float` (like `bool`, `char`, `int`, etc.). This is possible by specifying it as a file option when using I/O commands. (see section [Output Properties](#) to learn more about file options).

## Items of a Processing Pipeline

- In **G'MIC**, an image processing pipeline is described as a sequence of items separated by the space character. Such items are interpreted and executed from the left to the right. For instance, the expression:

```
filename.jpg blur 3,0 sharpen 10 resize 200%,200% output file_out.jpg
```

defines a valid pipeline composed of nine G'MIC items.

- Each G'MIC item is a string that is either a **command**, a list of command **arguments**, a **filename** or a special **input string**.
- Escape characters " and double quotes " can be used to define items containing spaces or other special characters. For instance, the two strings `single\ item` and `"single item"` both define the same single item, with a space in it.

## Input Data Items

- If a specified **G'MIC** item appears to be an existing filename, the corresponding image data are loaded and inserted at the end of the image list (which is equivalent to the use of `input filename`).
- Special filenames `-` and `-.ext` stand for the standard input/output streams, optionally forced to be in a specific `ext` file format (e.g. `-.jpg` or `-.png`).
- The following special input strings may be used as G'MIC items to create and insert new images with prescribed values, at the end of the image list:
  - `[selection]` or `[selection]xN`: Insert 1 or N copies of already existing images. `selection` may represent one or several images (see section [Command Items and Selections](#) to learn more about selections).
  - `width[%],_height[%],_depth[%],_spectrum[%],_values[xN]`: Insert one or N images with specified size and values (adding % to a dimension means "**percentage of the size along the same axis**", taken from the last image `[-1]`). Any specified dimension can be also written as `[image]`, and is then set to the size (along the same axis) of the existing specified image `[image]`. `values` can be either a sequence of numbers separated by commas , or a mathematical expression, as e.g. in input item `256,256,1,3,[x,y,128]` which creates a `256x256` RGB color image with a spatial shading on the red and green channels. (see section [Mathematical Expressions](#) to learn more about mathematical expressions).
  - `(v1,v2,...)[xN]`: Insert one or `N` new images from specified prescribed values. Value separator inside parentheses can be , (column separator), ; (row separator), / (slice separator) or ^ (channel separator). For instance, expression `(1,2,3;4,5,6;7,8,9)` creates a 3x3 matrix (scalar image), with values running from 1 to 9.

- `('string'[:delimiter])[xN]` : Insert one or N new images from specified string, by filling the images with the character codes composing the string. When specified, `delimiter` tells about the main orientation of the image. Delimiter can be `x` (eq. to `,`, which is the default), `y` (eq. to `;`), `z` (eq. to `/`) or `c` (eq. to `^`). When specified delimiter is `,`, `;`, `/` or `^`, the expression is actually equivalent to `({'string'[:delimiter]})[xN]` (see section **Substitution Rules** for more information on the syntax).
  - `0[xN]` : Insert one or N new `empty` images, containing no pixel data. Empty images are used only in rare occasions.
- Input item `name=value` declares a new variable `name`, or assign a new string value to an existing variable. Variable names must use the character set `[a-zA-Z0-9_]` and cannot start with a number.
  - A variable definition is always local to the current command except when it starts by the underscore character `_`. In that case, it becomes also accessible by any command invoked outside the current command scope (global variable).
  - If a variable name starts with two underscores `__`, the global variable is also shared among different threads and can be read/set by commands running in parallel (see command `parallel` for this purpose). Otherwise, it remains local to the thread that defined it.
  - Numerical variables can be updated with the use of these special operators: `+=` (addition), `-=` (subtraction), `*=` (multiplication), `/=` (division), `%=` (modulo), `&=` (bitwise and), `|=` (bitwise or), `^=` (power), `<<=` and `>>` (bitwise left and right shifts). For instance, `foo=1 foo+=3`.
  - Input item `name.=string` appends specified `string` at the end of variable `name`.
  - Input item `name..=string` prepends specified `string` at the beginning of variable `name`.
  - Multiple variable assignments and updates are allowed, with expressions:  
`name1, name2, ..., nameN=value` or  
`name1, name2, ..., nameN=value1, value2, ..., valueN` where assignment operator `=` can be replaced by one of the allowed operators (e.g. `+=`).
  - Variables usually store numbers or strings. Use command `store` to assign variables from image data (and syntax `input $variable` to bring them back on the image list afterwards).

## Command Items and Selections

- A G'MIC item that is not a filename nor a special input string designates a `command` most of the time. Generally, commands perform image processing operations on one or several available images of the list.
- Recurrent commands have two equivalent names (`regular` and `short`). For instance, command names `resize` and `r` refer to the same image resizing action.
- A G'MIC command may have mandatory or optional `arguments`. Command arguments must be specified in the next item on the command line. Commas `,` are used to separate multiple arguments of a single command, when required.
- The execution of a G'MIC command may be restricted only to a `subset` of the image list, by appending `[selection]` to the command name. Examples of valid syntaxes for `selection` are:

- `command[-2]`: Apply command only on the penultimate image `[ -2 ]` of the list.
- `command[0,1,3]`: Apply command only on images `[ 0 ]`, `[ 1 ]` and `[ 3 ]`.
- `command[3-6]`: Apply command only on images `[ 3 ]` to `[ 6 ]` (i.e. `[ 3 ]`, `[ 4 ]`, `[ 5 ]` and `[ 6 ]`).
- `command[50%-100%]`: Apply command only on the second half of the image list.
- `command[0,-4--1]`: Apply command only on the first image and the last four images.
- `command[0-9:3]`: Apply command only on images `[ 0 ]` to `[ 9 ]`, with a step of 3 (i.e. on images `[ 0 ]`, `[ 3 ]`, `[ 6 ]` and `[ 9 ]`).
- `command[0-9:25%]`: Apply command only on images `[ 0 ]` to `[ 9 ]`, with a step of 25% (i.e. on images `[ 0 ]`, `[ 3 ]`, `[ 6 ]` and `[ 9 ]`).
- `command[0--1:2]`: Apply command only on images of the list with even indices.
- `command[0,2-4,50%--1]`: Apply command on images `[ 0 ]`, `[ 2 ]`, `[ 3 ]`, `[ 4 ]` and on the second half of the image list.
- `command[^0,1]`: Apply command on all images except the first two.
- `command[name1, name2]`: Apply command on named images `name1` and `name2`.

- Indices in selections are always sorted in increasing order, and duplicate indices are discarded. For instance, selections `[3-1,1-3]` and `[1,1,1,3,2]` are both equivalent to `[1-3]`. If you want to repeat a single command multiple times on an image, use a `repeat..done` loop instead. Inverting the order of images for a command is achieved by explicitly inverting the order of the images in the list, with command `reverse[selection]`.
- Command selections `[ -1 ]`, `[ -2 ]` and `[ -3 ]` are so often used they have their own shortcuts, respectively `.`, `..` and `...`. For instance, command `blur..` is equivalent to `blur[-2]`. These shortcuts work also when specifying command arguments.
- G'MIC commands invoked without `[selection]` are applied on all images of the list, i.e. the default selection is `[0--1]` (except for command `input` whose default selection is `[ -1 ]'`).
- Prepending a single hyphen `-` to a G'MIC command is allowed. This may be useful to recognize command items more easily in a one-liner pipeline (typically invoked from a shell).
- A G'MIC command prepended with a plus sign `+` does not act **in-place** but inserts its result as one or several new images at the end of the image list.
- There are two different types of commands that can be run by the G'MIC interpreter:
  - **Built-in commands** are the hard-coded functionalities in the interpreter core. They are thus compiled as binary code and run fast, most of the time. Omitting an argument when invoking a built-in command is not permitted, except if all following arguments are also omitted. For instance, invoking 'plasma 10,,5' is invalid but 'plasma 10' is correct.
  - **Custom commands**, are defined as G'MIC pipelines of built-in or other custom commands. They are parsed by the G'MIC interpreter, and thus run a bit slower than built-in commands. Omitting arguments when invoking a custom command is permitted. For instance, expressions `flower ,,,100,,2` or `flower ,` are correct.
- Most of the existing commands in G'MIC are actually defined as **custom commands**.
- A user can easily add its own custom commands to the G'MIC interpreter (see section [Adding Custom Commands](#) for more details). New built-in commands cannot be added (unless you modify the G'MIC interpreter source code and recompile it).

# Input/Output Properties

- G'MIC is able to read/write most of the classical image file formats, including:
  - 2D grayscale/color files: `.png`, `.jpeg`, `.gif`, `.pnm`, `.tif`, `.bmp`, ...
  - 3D volumetric files: `.dcm`, `.hdr`, `.ni`, `.cube`, `.pan`, `.inr`, `.pnk`, ...
  - Video files: `.mpeg`, `.avi`, `.mp4`, `.mov`, `.ogg`, `.flv`, ...
  - Generic text or binary data files: `.gmz`, `.cimg`, `.cimgz`, `flo`, `ggr`, `gpl`, `.dlm`, `.asc`, `.pfm`, `.raw`, `.txt`, `.h`.
  - 3D mesh files: `.off`, `.obj` (output only)
- When dealing with color images, G'MIC generally reads, writes and displays data using the usual sRGB color space.
- When loading a `.png` and `.tiff` file, the bit-depth of the input image(s) is returned to the status.
- G'MIC is able to manage **3D objects** that may be read from files or generated by G'MIC commands. A 3D object is stored as a one-column scalar image containing the object data, in the following order: `{ magic_number; sizes; vertices; primitives; colors; opacities }`. These 3D representations can be then processed as regular images (see command `split3d` for accessing each of these 3D object data separately).
- Be aware that usual file formats may be sometimes not adapted to store all the available image data, since G'MIC uses float-valued image buffers. For instance, saving an image that was initially loaded as a 16bits/channel image, as a `.jpg` file will result in a loss of information. Use the G'MIC-specific file extension `.gmz` to ensure that all data precision is preserved when saving images.
- Sometimes, file options may/must be set for file formats:
  - **Video files:** Only sub-frames of an image sequence may be loaded, using the input expression `filename.ext[,first_frame[,last_frame[,step]]]`. Set `last_frame== -1` to tell it must be the last frame of the video. Set `step` to `0` to force an opened video file to be opened/closed. Output framerate and codec can be also set by using the output expression `filename.avi,_fps,_codec,_keep_open` where `keep_open` can be `{ 0 | 1 }`. `codec` is a 4-char string (see <http://www.fourcc.org/codecs.php>) or `0` for the default codec. `keep_open` tells if the output video file must be kept open for appending new frames afterwards.
  - **.cimg[z] files:** Only crops and sub-images of .cimg files can be loaded, using the input expressions `filename.cimg,N0,N1`, `filename.cimg,N0,N1,x0,x1`, `filename.cimg,N0,N1,x0,y0,x1,y1`, `filename.cimg,N0,N1,x0,y0,z0,x1,y1,z1` or `filename.cimg,N0,N1,x0,y0,z0,c0,x1,y1,z1,c1`. Specifying `-1` for one coordinates stands for the maximum possible value. Output expression `filename.cimg[z][,datatype]` can be used to force the output pixel type. `datatype` can be `{ auto | bool | uchar | char | ushort | short | uint | int | uint64 | int64 | float | double }`.
  - **.raw binary files:** Image dimensions and input pixel type may be specified when loading `.raw` files with input expression `filename.raw[,datatype][,width][,height[,depth[,dim[,offset]]]]`. If no dimensions are specified, the resulting

image is a one-column vector with maximum possible height. Pixel type can also be specified with the output expression `filename.raw[,datatype]`. `datatype` can be the same as for `.cimg[z]` files.

- **.yuv files:** Image dimensions must be specified when loading, and only sub-frames of an image sequence may be loaded, using the input expression `filename.yuv,width,height[,chroma_subsampling[,first_frame[,last_frame[,step]]]]`. `chroma_subsampling` can be `{ 420 | 422 | 444 }`. When saving, chroma subsampling mode can be specified with output expression `filename.yuv[,chroma_subsampling]`.
  - **.tiff files:** Only sub-images of multi-pages tiff files can be loaded, using the input expression `filename.tif,_first_frame,_last_frame,_step`. Output expression `filename.tiff,_datatype,_compression,_force_multipage,_use_bigtiff` can be used to specify the output pixel type, as well as the compression method. `datatype` can be the same as for `.cimg[z]` files. `compression` can be `{ none (default) | lzw | jpeg }`. `force_multipage` can be `{ 0=no (default) | 1=yes }`. `use_bigtiff` can be `{ 0=no | 1=yes (default) }`.
  - **.pdf files:** When loading a file, the rendering resolution can be specified using the input expression `filename.pdf,resolution`, where `resolution` is an unsigned integer value.
  - **.gif files:** Animated gif files can be saved, using the input expression `filename.gif,fps>0,nb_loops`. Specify `nb_loops=0` to get an infinite number of animation loops (this is the default behavior).
  - **.jpeg files:** The output quality may be specified (in %), using the output expression `filename.jpg,30` (here, to get a 30% quality output). `100` is the default.
  - **.mnc files:** The output header can set from another file, using the output expression `filename.mnc,header_template.mnc`.
  - **.pan, .cpp, .hpp, .c and .h files:** The output datatype can be selected with output expression `filename[,datatype]`. `datatype` can be the same as for `.cimg[z]` files.
  - **.gmic files:** These filenames are assumed to be G'MIC custom commands files. Loading such a file will add the commands it defines to the interpreter. Debug information can be enabled/disabled by the input expression `filename.gmic[,add_debug_info]` where `debug_info` can be `{ 0=false | 1=true }`.
  - Inserting `ext:` on the beginning of a filename (e.g. `jpg:filename`) forces G'MIC to read/write the file as it would have been done if it had the specified extension `.ext`.
- Some input/output formats and options may not be supported, depending on the configuration flags that have been set during the build of the G'MIC software.

## Substitution Rules

- **G'MIC** items containing `$` or `{}` are substituted before being interpreted. Use these substituting expressions to access various data from the interpreter environment.
- `$name` and  `${name}` are both substituted by the value of the specified named variable (set previously by the item `name=value`). If this variable has not been already set, the expression is substituted by the highest positive index of the named image `[name]`. If no image has this name, the expression is substituted by the value of the OS environment variable with same name (it may be thus an empty string if it is not defined).

- The following reserved variables are predefined by the G'MIC interpreter:
  - `$!`: The current number of images in the list.
  - `$>` and `$<`: The increasing/decreasing index of the latest (currently running) `repeat...done` loop. `$>` goes from `0` (first loop iteration) to `nb_iterations - 1` (last iteration). `$<` does the opposite.
  - `$/`: The current call stack. Stack items are separated by slashes `/`.
  - `$|`: The current value (expressed in seconds) of a millisecond precision timer.
  - `$^`: The current verbosity level.
  - `$_cpus`: The number of computation cores available on your machine.
  - `$_flags`: The list of enabled flags when G'MIC interpreter has been compiled.
  - `$_host`: A string telling about the host running the G'MIC interpreter (e.g. `cli` or `gimp`).
  - `$_os`: A string describing the running operating system.
  - `$_path_rc`: The path to the G'MIC folder used to store configuration files (its value is OS-dependent).
  - `$_path_user`: The path to the G'MIC user file `.gmic` or `user.gmic` (its value is OS-dependent).
  - `$_path_commands`: A list of all imported command files (stored as a list-valued variable).
  - `$_pid`: The current process identifier, as an integer.
  - `$_pixeltype`: The type of image pixels (default: `float`).
  - `$_prerelease`: For pre-releases, the date of the pre-release as `yymmdd`. For stable releases, this variable is set to `0`.
  - `$_version`: A 3-digits number telling about the current version of the G'MIC interpreter (e.g. `300`).
  - `$_vt100`: Set to `1` if colored text output is allowed on the console. Otherwise, set to `0`.
- `$$name` and `$$ ${name}` are both substituted by the G'MIC script code of the specified named `custom command`, or by an empty string if no custom command with specified name exists.
- `${"-pipeline"}` is substituted by the **status value** after the execution of the specified G'MIC pipeline (see command `status`). Expression  `${}` thus stands for the current status value.
- `{` `string}` (starting with two backquotes) is substituted by a double-quoted version of the specified string.
- `/{string}` is substituted by the escaped version of the specified string.
- `{'string'[:delimiter]}` (between single quotes) is substituted by the sequence of character codes that composes the specified string, separated by specified delimiter. Possible delimiters are `,` (default), `;`, `/`, `^` or `'`. For instance, item `{'foo'}` is substituted by `102,111,111` and `{'foo';'}` by `102;111;111`.
- `{image,feature[:delimiter]}` is substituted by a specific feature of the image `[image]`. `image` can be either an image number or an image name. It can be also eluded, in which case, the last image `[-1]` of the list is considered for the requested feature. Specified `feature` can be one of:
  - `b`: The image basename (i.e. filename without the folder path nor extension).
  - `f`: The image folder name.

- `n`: The image name or filename (if the image has been read from a file).
  - `t`: The text string from the image values regarded as character codes.
  - `x`: The image extension (i.e the characters after the last `.` in the image name).
  - `^`: The sequence of all image values, separated by commas `,`.
  - `@subset`: The sequence of image values corresponding to the specified subset, and separated by commas `,`.
  - Any other `feature` is considered as a **mathematical expression** associated to the image `[image]` and is substituted by the result of its evaluation (float value). For instance, expression `{0,w+h}` is substituted by the sum of the width and height of the first image (see section **Mathematical Expressions** for more details). If a mathematical expression starts with an underscore `_`, the resulting value is truncated to a readable format. For instance, item `{_pi}` is substituted by `3.14159` (while `{pi}` is substituted by `3.141592653589793`).
  - A `feature` delimited by backquotes is replaced by a string whose character codes correspond to the list of values resulting from the evaluation of the specified mathematical expression. For instance, item `{`[102,111,111]`}` is substituted by `foo` and item `{`vector8(65)`}` by `AAAAAA`.
- `{*}` is substituted by the visibility state of the instant display window `#0` (can be `{ 0=closed | 1=visible }`).
- `{*[index], feature1, ..., featureN[:delimiter]}` is substituted by a specific set of features of the instant display window `#0` (or `#index`, if specified). Requested `features` can be:
- `u`: screen width (actually independent on the window size).
  - `v`: screen height (actually independent on the window size).
  - `uv`: screen width x screen height.
  - `d`: window width (i.e. width of the window widget).
  - `e`: window height (i.e. height of the window widget).
  - `de`: window width x window height.
  - `w`: display width (i.e. width of the display area managed by the window).
  - `h`: display height (i.e. height of the display area managed by the window).
  - `wh`: display width x display height.
  - `i`: X-coordinate of the display window.
  - `j`: Y-coordinate of the display window.
  - `n`: current normalization type of the instant display.
  - `t`: window title of the instant display.
  - `x`: X-coordinate of the mouse position (or -1, if outside the display area).
  - `y`: Y-coordinate of the mouse position (or -1, if outside the display area).
  - `b`: state of the mouse buttons `{ 1=left-but. | 2=right-but. | 4=middle-but. }`.
  - `o`: state of the mouse wheel.
  - `k`: decimal code of the pressed key if any, 0 otherwise.
  - `c`: boolean (0 or 1) telling if the instant display has been closed recently.

- `r`: boolean telling if the instant display has been resized recently.
- `m`: boolean telling if the instant display has been moved recently.
- Any other `feature` stands for a keycode name (in capital letters), and is substituted by a boolean describing the current key state `{ 0=pressed | 1=released }`.
- You can also prepend a hyphen `-` to a `feature` (that supports it) to flush the corresponding event immediately after reading its state (works for keys, mouse and window events).
- Item substitution is **never** performed in items between double quotes. One must break the quotes to enable substitution if needed, as in "`3+8 kg = "{3+8}" kg`". Using double quotes is then a convenient way to disable the substitutions mechanism in items, when necessary.
- One can also disable the substitution mechanism on items outside double quotes, by escaping the `{, }` or `$` characters, as in `\{3+4\}\ doesn't\ evaluate`.

## Mathematical Expressions

- **G'MIC** has an embedded **mathematical parser**, used to evaluate (possibly complex) math expressions specified inside braces `{}`, or formulas in commands that may take one as an argument (e.g. `fill` or `eval`).
- When the context allows it, a formula is evaluated **for each pixel** of the selected images (e.g. `fill` or `eval`).
- A math expression may return a **scalar** or a **vector-valued** result (with a fixed number of components).

The mathematical parser understands the following set of functions, operators and variables:

### Usual operators:

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

### Usual math functions:

```
abs(), acos(), acosh(), arg(), arg0(), argkth(), argmax(), argmaxabs(),
argmin(), argminabs(), asin(), asinh(), atan(), atan2(), atanh(), avg(),
bool(), cbrt(), ceil(), cos(), cosh(), cut(), deg2rad(), erf(), erfinv(),
exp(), fact(), fibo(), floor(), gauss(), gcd(), int(), isnan(), isnum(),
isinf(), isint(), isbool(), isexpr(), isfile(), isdir(), isin(), kth(), log(),
log2(), log10(), max(), maxabs(), med(), min(), minabs(), narg(), prod(),
rad2deg(), rol() (left bit rotation), ror() (right bit rotation), round(), sign(), sin(),
sinc(), sinh(), sqrt(), std(), srand(_seed), sum(), tan(), tanh(), var(),
xor() .
```

- `atan2(y,x)` is the version of `atan()` with two arguments `y` and `x` (as in C/C++).
- `permut(k,n,with_order)` computes the number of permutations of `k` objects from a set of `n` objects.

- `gauss(x,_sigma,_is_normalized)` returns  $\exp(-x^2/(2*s^2))/(is\_normalized? \sqrt{2*pi*sigma^2}:1)$ .
- `cut(value,min,max)` returns `value` if it is in range `[min,max]`, or `min` or `max` otherwise.
- `narg(a_1,...,a_N)` returns the number of specified arguments (here, `N`).
- `arg(i,a_1,...,a_N)` returns the `i`-th argument `a_i`.
- `isnum()`, `isnan()`, `isinf()`, `isint()`, `isbool()` test the type of the given number or expression, and return `0` (false) or `1` (true).
- `isfile('path')` (resp. `isdir('path')`) returns `0` (false) or `1` (true) whether its string argument is a path to an existing file (resp. to a directory) or not.
- `isvarname('str')` returns `0` (false) or `1` (true) whether its string argument would be a valid to name a variable or not.
- `isin(v,a_1,...,a_n)` returns `0` (false) or `1` (true) whether the first value `v` appears in the set of other values `a_i`.
- `inrange(value,m,M,include_m,include_M)` returns `0` (false) or `1` (true) whether the specified value lies in range `[m,M]` or not (`include_m` and `includeM` tells how boundaries `m` and `M` are considered).
- `argkth()`, `argmin()`, `argmax()`, `argminabs()`, `argmaxabs()`, `avg()`, `kth()`, `min()`, `max()`, `minabs()`, `maxabs()`, `med()`, `prod()`, `std()`, `sum()` and `var()` can be called with an arbitrary number of scalar/vector arguments.
- `vargkth()`, `vargmin()`, `vargmax()`, `vargminabs()`, `vargmaxabs()`, `vavg()`, `vkth()`, `vmin()`, `vmax()`, `vminabs()`, `vmaxabs()`, `vmed()`, `vprod()`, `vstd()`, `vsum()` and `vvar()` are the versions of the previous function with vector-valued arguments.
- `round(value,rounding_value,direction)` returns a rounded value. `direction` can be `{ -1=to-lowest | 0=to-nearest | 1=to-highest }`.
- `lerp(a,b,t)` returns `a*(1-t)+b*t`.
- `swap(a,b)` swaps the values of the given arguments.

## Variable names:

Variable names below are pre-defined. They can be overridden.

- `l`: length of the associated list of images.
- `k`: index of the associated image, in `[0,l-1]`.
- `w`: width of the associated image, if any (`0` otherwise).
- `h`: height of the associated image, if any (`0` otherwise).
- `d`: depth of the associated image, if any (`0` otherwise).
- `s`: spectrum of the associated image, if any (`0` otherwise).
- `r`: shared state of the associated image, if any (`0` otherwise).
- `wh`: shortcut for width x height.
- `whd`: shortcut for width x height x depth.
- `whds`: shortcut for width x height x depth x spectrum (i.e. number of image values).

- `iM`, `iM`, `ia`, `iv`, `is`, `ip`, `ic`, `in`: Respectively the minimum, maximum, average, variance, sum, product, median value and L2-norm of the associated image, if any (`0` otherwise).
- `xm`, `ym`, `zm`, `cm`: The pixel coordinates of the minimum value in the associated image, if any (`0` otherwise).
- `xM`, `yM`, `zM`, `cM`: The pixel coordinates of the maximum value in the associated image, if any (`0` otherwise).
- All these variables are considered as **constant values** by the math parser (for optimization purposes) which is indeed the case most of the time. Anyway, this might not be the case, if function `resize(#ind,...)` is used in the math expression. If so, it is safer to invoke functions `l()`, `w(_#ind)`, `h(_#ind)`, ... `s(_#ind)` and `in(_#ind)` instead of the corresponding named variables.
- `i`: current processed pixel value (i.e. value located at `(x,y,z,c)`) in the associated image, if any (`0` otherwise).
- `iN`: N-th channel value of current processed pixel (i.e. value located at `(x,y,z,N)` in the associated image, if any (`0` otherwise). `N` must be an integer in range `[0,9]`.
- `R`, `G`, `B` and `A` are equivalent to `i0`, `i1`, `i2` and `i3` respectively.
- `I`: current vector-valued processed pixel in the associated image, if any (`0` otherwise). The number of vector components is equal to the number of image channels (e.g. `I = [ R, G, B ]` for a `RGB` image).
- You may add `#ind` to any of the variable name above to retrieve the information for any numbered image `[ind]` of the list (when this makes sense). For instance `ia#0` denotes the average value of the first image of the list.
- `x`: current processed column of the associated image, if any (`0` otherwise).
- `y`: current processed row of the associated image, if any (`0` otherwise).
- `z`: current processed slice of the associated image, if any (`0` otherwise).
- `c`: current processed channel of the associated image, if any (`0` otherwise).
- `t`: thread id when an expression is evaluated with multiple threads (`0` means **master thread**).
- `n`: maximum number of threads when expression is evaluated in parallel (so that `t` goes from `0` to `n-1`).
- `e`: value of e, i.e. `2.71828...`.
- `pi`: value of pi, i.e. `3.1415926...`.
- `u`: a random value between `[0,1]`, following a uniform distribution.
- `g`: a random value, following a gaussian distribution of variance 1 (roughly in `[-6,6]`).
- `interpolation`: value of the default interpolation mode used when reading pixel values with the pixel access operators (i.e. when the interpolation argument is not explicitly specified, see below for more details on pixel access operators). Its initial default value is `0`.
- `boundary`: value of the default boundary conditions used when reading pixel values with the pixel access operators (i.e. when the boundary condition argument is not explicitly specified, see below for more details on pixel access operators). Its initial default value is `0`.

- The last image of the list is always associated to the evaluations of **expressions**, e.g. G'MIC sequence

```
256,128 fill {w}
```

will create a 256x128 image filled with value 256.

## Vector calculus:

Most operators are also able to work with vector-valued elements.

- [a<sub>0</sub>,a<sub>1</sub>,...,a<sub>N-1</sub>]** defines a **N**-dimensional vector with scalar coefficients **ak**.
- vectorN(a<sub>0</sub>,a<sub>1</sub>,...,a<sub>N-1</sub>)** does the same, with the **ak** being repeated periodically if only a few are specified.
- vector(#N,a<sub>0</sub>,a<sub>1</sub>,...,a<sub>N-1</sub>)** does the same, and can be used for any constant expression **N**.
- In previous expressions, the **ak** can be vectors themselves, to be concatenated into a single vector.
- The scalar element **ak** of a vector **X** is retrieved by **X[k]**.
- The sub-vector **[X[p],X[p+s]...X[p+s\*(q-1)]]** (of size **q**) of a vector **X** is retrieved by **X[p,q,s]**.
- expr(formula,\_w,\_h,\_d,\_s)** outputs a vector of size **w\*h\*d\*s** with values generated from the specified formula, as if one were filling an image with dimensions **(w,h,d,s)**.
- Equality/inequality comparisons between two vectors is done with operators **==** and **!=**.
- Some vector-specific functions can be used on vector values: **cross(X,Y)** (cross product), **dot(X,Y)** (dot product), **size(X)** (vector dimension), **sort(X,\_is\_increasing,\_nb\_elts,\_size\_elt)** (sorted values), **reverse(A)** (reverse order of components), **shift(A,\_length,\_boundary\_conditions)** and **same(A,B,\_nb\_vals,\_is\_case\_sensitive)** (vector equality test).
- Function **normP(u<sub>1</sub>,...,u<sub>n</sub>)** computes the LP-norm of the specified vector (**P** being an **unsigned integer** constant or **inf**). If **P** is omitted, the L2 norm is calculated.
- Function **resize(A,size,\_interpolation,\_boundary\_conditions)** returns a resized version of a vector **A** with specified interpolation mode. **interpolation** can be **{ -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }**, and **boundary\_conditions** can be **{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }**.
- Function **resize(A,ow,oh,od,os,nw,\_,nh,nd,ns,\_interpolation,\_boundary\_conditions,\_ax,** is an extended version of the previous function. It allows to resize the vector **A**, seen as an image of size 'ow x oh x od x os' as a new image of size 'nw x nh x nd x ns', with specified resizing options.
- Function **find(A,B,\_starting\_index,\_search\_step)** returns the index where sub-vector **B** appears in vector **A**, (or **-1** if **B** is not contained in **A**). Argument **A** can be also replaced by an image index **#ind**.
- A **2**-dimensional vector may be seen as a complex number and used in those particular functions/operators: **\*\*** (complex multiplication), **//** (complex division), **^^** (complex

exponentiation), `**=` (complex self-multiplication), `//=` (complex self-division), `^~=` (complex self-exponentiation), `cabs()` (complex modulus), `carg()` (complex argument), `cconj()` (complex conjugate), `cexp()` (complex exponential), `clog()` (complex logarithm), `ccos()` (complex cosine), `csin()` (complex sine), `ctan()` (complex tangent), `ccosh()` (complex hyperpolic cosine), `csinh()` (complex hyperbolic sine) and `ctanh()` (complex hyperbolic tangent).

- A  $MN$ -dimensional vector may be seen as a  $M \times N$  matrix and used in those particular functions/operators: `*` (matrix-vector multiplication), `det(A)` (determinant), `diag(V)` (diagonal matrix from a vector), `eig(A)` (eigenvalues/eigenvectors), `eye(n)` ( $n \times n$  identity matrix), `invert(A,_solver)` (matrix inverse), `mul(A,B,_nb_colsB)` (matrix-matrix multiplication), `pseudoinvert(A,_nb_colsA,_solver)`, `rot(u,v,w,angle)` (3D rotation matrix), `rot(angle)` (2D rotation matrix), `solve(A,B,_nb_colsB)` (solver of linear system  $A \cdot X = B$ ), `svd(A,_nb_colsA)` (singular value decomposition), `trace(A)` (matrix trace) and `transpose(A,nb_colsA)` (matrix transpose). Argument `nb_colsB` may be omitted if it is equal to `1`.
- `mproj(S,nb_colsS,D,nb_colsD,method,max_iter,max_residual)` projects a matrix `S` onto a dictionary (matrix) `D`. Equivalent to command `mproj` but inside the math evaluator.
- Specifying a vector-valued math expression as an argument of a command that operates on image values (e.g. `fill`) modifies the whole spectrum range of the processed image(s), for each spatial coordinates `(x,y,z)`. The command does not loop over the `c`-axis in this case.

## String manipulation:

Character strings are defined and managed as vectors objects. Dedicated functions and initializers to manage strings are:

- `['string']` and `'string'` define a vector whose values are the character codes of the specified `character string` (e.g. `'foo'` is equal to `[ 102, 111, 111 ]`).
- `_character` returns the (scalar) byte code of the specified character (e.g. `_['A']` is equal to `65`).
- A special case happens for `empty` strings: Values of both expressions `['']` and `''` are `0`.
- Functions `lowercase()` and `uppercase()` return string with all string characters lowercased or uppcased.
- Function `stov(str,_starting_index,_is_strict)` parses specified string `str` and returns the value contained in it.
- Function `vtos(expr,_nb_digits,_siz)` returns a vector of size `siz` which contains the character representation of values described by expression `expr`. `nb_digits` can be `{ -1=auto-reduced | 0=all | >0=max number of digits }`.
- Function `echo(str1,str2,...,strN)` prints the concatenation of given string arguments on the console.
- Function `string(_#siz,str1,str2,...,strN)` generates a vector corresponding to the concatenation of given string/number arguments.

## Dynamic arrays:

A dynamic array is defined as a one-column (or empty) image `[ind]` in the image list. It allows elements to be added or removed, each element having the same dimension (which is actually the number of channels of image `[ind]`). Dynamic arrays adapt their size to the number of elements

they contain.

A dynamic array can be manipulated in a math expression, with the following functions:

- `da_size(_#ind)`: Return the number of elements in dynamic array `[ind]`.
- `da_back(_#ind)`: Return the last element of the dynamic array `[ind]`.
- `da_insert(_#ind, pos, elt_1, elt_2, ..., elt_N)`: Insert `N` new elements `elt_k` starting from index `pos` in dynamic array `[ind]`.
- `da_push(_#ind, elt1, elt2, ..., eltN)`: Insert `N` new elements `elt_k` at the end of dynamic array `[ind]`.
- `da_pop(_#ind)`: Same as `da_back()` but also remove last element from the dynamic array `[ind]`.
- `da_remove(_#ind, _start, _end)`: Remove elements located between indices `start` and `end` (included) in dynamic array `[ind]`.
- The value of the  $k$ -th element of dynamic array `[ind]` is retrieved with `i[_#ind,k]` (if the element is a scalar value), or `I[_#ind,k]` (if the element is a vector).

In the functions above, argument `#ind` may be omitted in which case it is assumed to be `#-1`.

## Special operators:

- `;`: expression separator. The returned value is always the last encountered expression. For instance expression `1;2;pi` is evaluated as `pi`.
- `=`: variable assignment. Variables in mathematical parser can only refer to numerical values (vectors or scalars). Variable names are case-sensitive. Use this operator in conjunction with `;` to define more complex evaluable expressions, such as

```
t = cos(x); 3*t^2 + 2*t + 1
```

These variables remain **local** to the mathematical parser and cannot be accessed outside the evaluated expression.

- Variables defined in math parser may have a **constant** property, by specifying keyword `const` before the variable name (e.g. 'const foo = pi/4;'). The value set to such a variable must be indeed a **constant scalar**. Constant variables allows certain types of optimizations in the math JIT compiler.

## Specific functions:

- `addr(expr)`: return the pointer address to the specified expression `expr`.
- `fill(target,expr)` or `fill(target,index_name,expr)` fill the content of the specified target (often vector-valued) using a given expression, e.g. `V = vector16(); fill(V,k,k^2 + k + 1);`. For a vector-valued target, it is basically equivalent to:  
`for (index_name = 0, index_name<size(target), ++index_name, target[index_name]`
- `u(max)` or `u(min,max)`: return a random value between `[0,max]` or `[min,max]`, following a uniform distribution.
- `f2ui(value)` and `ui2f(value)`: Convert a large unsigned integer as a negative floating point value (and vice-versa), so that 32bits floats can be used to store large integers while keeping a

unitary precision.

- `i(_a,_b,_c,_d,_interpolation_type,_boundary_conditions)`: return the value of the pixel located at position `(a,b,c,d)` in the associated image, if any (`0` otherwise).  
`interpolation_type` can be `{ 0=nearest neighbor | 1=linear | 2=cubic }`.  
`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`. Omitted coordinates are replaced by their default values which are respectively `x, y, z, c`, `interpolation` and `boundary`. For instance command

```
fill 0.5*(i(x+1)-i(x-1))
```

will estimate the X-derivative of an image with a classical finite difference scheme.

- `j(_dx,_dy,_dz,_dc,_interpolation_type,_boundary_conditions)` does the same for the pixel located at position `(x+dx,y+dy,z+dz,c+dc)` (pixel access relative to the current coordinates).
- `i[offset,_boundary_conditions]` returns the value of the pixel located at specified `offset` in the associated image buffer (or `0` if offset is out-of-bounds).
- `j[offset,_boundary_conditions]` does the same for an offset relative to the current pixel coordinates `(x,y,z,c)`.
- `i(#ind,_x,_y,_z,_c,_interpolation,_boundary_conditions)`,  
`j(#ind,_dx,_dy,_dz,_dc,_interpolation,_boundary_conditions)`,  
`i[#ind,offset,_boundary_conditions]` and `i[offset,_boundary_conditions]` are similar expressions used to access pixel values for any numbered image `[ind]` of the list.
- `I/J[offset,_boundary_conditions]` and  
`I/J(#ind,_x,_y,_z,_interpolation,_boundary_conditions)` do the same as  
`i/j[offset,_boundary_conditions]` and  
`i/j(#ind,_x,_y,_z,_c,_interpolation,_boundary_conditions)` but return a vector instead of a scalar (e.g. a vector `[ R,G,B ]` for a pixel at `(a,b,c)` in a color image).
- `crop(_#ind,_x,_y,_z,_c,_dx,_dy,_dz,_dc,_boundary_conditions)` returns a vector whose values come from the cropped region of image `[ind]` (or from default image selected if `ind` is not specified). Cropped region starts from point `(x,y,z,c)` and has a size of `dx x dy x dz x dc`. Arguments for coordinates and sizes can be omitted if they are not ambiguous (e.g. `crop(#ind,x,y,dx,dy)` is a valid invocation of this function).
- `draw(_#ind,S,x,y,z,c,dx,_dy,_dz,_dc,_opacity,_M,_max_M)` draws a sprite `S` in image `[ind]` (or in default image selected if `ind` is not specified) at coordinates `(x,y,z,c)`. The size of the sprite `dx x dy x dz x dc` must be specified. You can also specify a corresponding opacity mask `M` if its size matches `S`.
- `polygon(_#ind,nb_vertices,coords,_opacity,_color)` draws a filled polygon in image `[ind]` (or in default image selected if `ind` is not specified) at specified coordinates. It draws a single line if `nb_vertices` is set to 2.
- `polygon(_#ind,-nb_vertices,coords,_opacity,_pattern,_color)` draws a outlined polygon in image `[ind]` (or in default image selected if `ind` is not specified) at specified coordinates and with specified line pattern. It draws a single line if `nb_vertices` is set to 2.
- `ellipse(_#ind,xc,yc,radius1,_radius2,_angle,_opacity,_color)` draws a filled ellipse in image `[ind]` (or in default image selected if `ind` is not specified) with specified coordinates.

- `ellipse(_#ind,xc,yc,-radius1,-radius2,_angle,_opacity,_pattern,_color)` draws an outlined ellipse in image `[ind]` (or in default image selected if `ind` is not specified).
- `resize(#ind,w,_h,_d,_s,_interp,_boundary_conditions,_cx,_cy,_cz,_cc)` resizes an image of the associated list with specified dimension and interpolation method. When using this function, you should consider retrieving the (non-constant) image dimensions using the dynamic functions `w(_#ind)`, `h(_#ind)`, `d(_#ind)`, `s(_#ind)`, `wh(_#ind)`, `whd(_#ind)` and `whds(_#ind)` instead of the corresponding constant variables.
- `if(condition,expr_then,expr_else)`: return value of `expr_then` or `expr_else`, depending on the value of `condition { 0=false | other=true }`. `expr_else` can be omitted in which case `0` is returned if the condition does not hold. Using the ternary operator `condition?expr_then[:expr_else]` gives an equivalent expression. For instance, G'MIC commands

```
fill if(x%10==0,255,i)
```

and

```
fill x%10?i:255
```

both draw blank vertical lines on every 10th column of an image.

- `do(expression,_condition)` repeats the evaluation of `expression` until `condition` vanishes (or until `expression` vanishes if no `condition` is specified). For instance, the expression:

```
if(N<2,N,n=N-1;F0=0;F1=1;do(F2=F0+F1;F0=F1;F1=F2,n=n-1))
```

returns the N-th value of the Fibonacci sequence, for `N>=0` (e.g., `46368` for `N=24`).

- `do(expression,condition)` always evaluates the specified expression at least once, then check for the loop condition. When done, it returns the last value of `expression`.

- `for(init,condition,_procedure,body)` first evaluates the expression `init`, then iteratively evaluates `body` (followed by `procedure` if specified) while `condition` holds (i.e. not zero). It may happen that no iterations are done, in which case the function returns `nan`. Otherwise, it returns the last value of `body`. For instance, the expression:

```
if(N<2,N,for(n=N;F0=0;F1=1,n=n-1,F2=F0+F1;F0=F1;F1=F2))
```

returns the `N`-th value of the Fibonacci sequence, for `N>=0` (e.g., `46368` for `N=24`).

- `while(condition,expression)` is exactly the same as `for(init,condition,expression)` without the specification of an initializing expression.
- `repeat(nb_iters,expr)` or `fill(nb_iters,iter_name,expr)` run `nb_iters` iterations of the specified expression `expr`, e.g.  
`V = vector16(); repeat(16,k,V[k] = k^2 + k + 1);`. It is basically equivalent to:  
`for (iter_name = 0, iter_name<nb_iters, ++iter_name, expr);`.
- `break()` and `continue()` respectively breaks and continues the current running bloc (loop, init or main environment).
- `fsize('filename')` returns the size of the specified `filename` (or `-1` if file does not exist).
- `date(attr,'path')` returns the date attribute for the given `path` (file or directory), with `attr` being `{ 0=year | 1=month | 2=day | 3=day of week | 4=hour | 5=minute | 6=second }`, or a vector of those values.

- `date(_attr)` returns the specified attribute for the current (locale) date (attributes being `t` *0...6=same meaning as above | 7=milliseconds* ).
- `print(expr1,expr2,...)` or `print(#ind)` prints the value of the specified expressions (or image information) on the console, and returns the value of the last expression (or `nan` in case of an image). Function `prints(expr)` also prints the string composed of the character codes defined by the vector-valued expression (e.g. `prints('Hello')`).
- `debug(expression)` prints detailed debug info about the sequence of operations done by the math parser to evaluate the expression (and returns its value).
- `display(_X,_w,_h,_d,_s)` or `display(#ind)` display the contents of the vector `X` (or specified image) and wait for user events. if no arguments are provided, a memory snapshot of the math parser environment is displayed instead.
- `begin(expression)` and `end(expression)` evaluates the specified expressions only once, respectively at the beginning and end of the evaluation procedure, and this, even when multiple evaluations are required (e.g. in 'fill ">begin(foo = 0); ++foo"'').
- `copy(dest,src,_nb_elts,_inc_d,_inc_s,_opacity)` copies an entire memory block of `nb_elts` elements starting from a source value `src` to a specified destination `dest`, with increments defined by `inc_d` and `inc_s` respectively for the destination and source pointers.
- `stats(#ind)` returns the statistics vector of the running image `[ind]`, i.e the vector `[im,iM,ia,iv,xm,ym,zm,cm,xM,yM,zM,cM,is,ip]` (14 values).
- `ref(expr,a)` references specified expression `expr` as variable name `a`.
- `unref(a,b,...)` destroys references to the named variable given as arguments.
- `breakpoint()` inserts a possible computation breakpoint (useless with the cli interface).
- '`(comment) expr`' just returns expression `expr` (useful for inserting inline comments in math expressions).
- `run('pipeline')` executes the specified G'MIC pipeline as if it was called outside the currently evaluated expression.
- `set(A,'variable_name')` set the G'MIC variable `$variable_name` with the value of expression `A`. If `A` is a vector-valued variable, it is assumed to encode a string.
- `store(A,'variable_name',_w,_h,_d,_s,_is_compressed)` transfers the data of vector `A` as a `w x h x d x s` image to the G'MIC variable `$variable_name`. Thus, the data becomes available outside the math expression (that is equivalent to using the regular command `store`, but directly in the math expression).
- `get('variable_name',_size,_return_as_string)` returns the value of the specified variable, as a vector of `size` values, or as a scalar (if `size` is zero or not specified).
- `name(_#ind,size)` returns a vector of size `size`, whose values are the characters codes of the name of image `[ind]` (or default image selected if `ind` is not specified).
- `correlate(I,wI,hI,dI,sI,K,wK,hK,dK,sK,_boundary_conditions,_is_normalized,_ch)` returns the correlation, unrolled as a vector, of the `wI x hI x dI x sI`-sized image `I` with the `wK x hK x dK x sK`-sized kernel `K` (the meaning of the other arguments are the same as in command `correlate`). Similar function `convolve(...)` is also defined for computing the convolution between `I` and `K`.

## User-defined macros:

- Custom macro functions can be defined in a math expression, using the assignment operator `=`, e.g.

```
foo(x,y) = cos(x + y); result = foo(1,2) + foo(2,3)
```

- Trying to override a built-in function (e.g. `abs()`) has no effect.
- Overloading macros with different number of arguments is possible. Re-defining a previously defined macro with the same number of arguments discards its previous definition.
- Macro functions are indeed processed as **macros** by the mathematical evaluator. You should avoid invoking them with arguments that are themselves results of assignments or self-operations. For instance,

```
foo(x) = x + x; z = 0; foo(++z)
```

returns `4` rather than expected value `2`.

- When substituted, macro arguments are placed inside parentheses, except if a number sign `#` is located just before or after the argument name. For instance, expression

```
foo(x,y) = x*y; foo(1+2,3)
```

returns `9` (being substituted as `(1+2)*(3)`), while expression

```
foo(x,y) = x#*y#; foo(1+2,3)
```

returns `7` (being substituted as `1+2*3`).

- Number signs appearing between macro arguments function actually count for **empty** separators. They may be used to force the substitution of macro arguments in unusual places, e.g. as in

```
str(N) = ['I like N#'];
```

## Multi-threaded and in-place evaluation:

- If your image data are large enough and you have several CPUs available, it is likely that the math expression passed to a `fill`, `eval` or `input` commands is evaluated in parallel, using multiple computation threads.
- Starting an expression with `:` or `*` forces the evaluations required for an image to be run in parallel, even if the amount of data to process is small (beware, it may be slower to evaluate in this case!). Specify `:` (rather than `*`) to avoid possible image copy done before evaluating the expression (this saves memory, but do this only if you are sure this step is not required!).
- If the specified expression starts with `>` or `<`, the pixel access operators `i()`, `i[]`, `j()` and `j[]` return values of the image being currently modified, in forward (`>`) or backward (`<`) order. The multi-threading evaluation of the expression is disabled in this case.
- Function `critical(expr)` forces the execution of the given expression in a single thread at a time.
- `begin_t(expr)` and `end_t(expr)` evaluates the specified expression once for each running thread (so possibly several times) at the beginning and the end of the evaluation procedure.
- `merge(variable,operator)` tells to merge the local variable value computed by threads, with the specified operator, when all threads have finished computing.

- Expressions `i(_#ind,x,_y,_z,_c)=value`, `j(_#ind,x,_y,_z,_c)=value`, `i[_#ind,offset]=value` and `j[_#ind,offset]=value` set a pixel value at a different location than the running one in the image `[ind]` (or in the associated image if argument `#ind` is omitted), either with global coordinates/offsets (with `i(...)` and `i[...]`), or relatively to the current position `(x,y,z,c)` (with `j(...)` and `j[...]`). These expressions always return `value`.

## Image and Data Viewers

- **G'MIC** has some very handy embedded **visualization modules**, for 1D signals (command `plot`), 1D/2D/3D images (command `display`) and 3D vector objects (command `display3d`). It manages interactive views of the selected image data.
- The following actions are available in the interactive viewers:
  - `(mousewheel)`: Zoom in/out.
  - `ESC`: Close window.
  - `CTRL+D`: Increase window size.
  - `CTRL+C`: Decrease window size.
  - `CTRL+R`: Reset window size.
  - `CTRL+F`: Toggle fullscreen mode.
  - `CTRL+S`: Save current view as a numbered file `gmic_xxxx.ext`.
  - `CTRL+0`: Save copy of the viewed data, as a numbered file `gmic_xxxx.ext`.
- Actions specific to the **1D/2D image viewer** (command `display`) are:
  - `Left mouse button`: Create an image selection and zoom into it.
  - `Middle mouse button`, or `CTRL+left mouse button`: Move image.
  - `Mouse wheel` or `PADD+/-`: Zoom in/out.
  - `Arrow keys`: Move image left/right/up/down.
  - `CTRL+A`: Enable/disable transparency (show alpha channel).
  - `CTRL+N`: Change normalization mode (can be `{ none | normal | channel-by-channel }`).
  - `CTRL+SPACE`: Reset view.
  - `CTRL+X`: Show/hide axes.
  - `CTRL+Z`: Hold/release aspect ratio.
- Actions specific to the **3D volumetric image viewer** (command `display`) are:
  - `CTRL+P`: Play z-stack of frames as a movie.
  - `CTRL+V`: Show/hide 3D view on bottom right zone.
  - `CTRL+X`: Show/hide axes.
  - `CTRL+(mousewheel)`: Go up/down.
  - `SHIFT+(mousewheel)`: Go left/right.

- **Numeric PAD**: Zoom in/out (+/-) and move through zoomed image (digits).
- **BACKSPACE**: Reset zoom scale.

- Actions specific to the **3D object viewer** (command `display3d`) are:

- **(mouse)+(left mouse button)**: Rotate 3D object.
- **(mouse)+(right mouse button)**: Zoom 3D object.
- **(mouse)+(middle mouse button)**: Shift 3D object.
- **F1 ... F6**: Toggle between different 3D rendering modes.
- **F7/F8**: Decrease/increase focale.
- **F9**: Select animation mode.
- **F10**: Select animation speed.
- **SPACE**: Start/stop animation.
- **CTRL+A**: Show/hide 3D axes.
- **CTRL+B**: Switch between available background.
- **CTRL+G**: Save 3D object, as numbered file `gmic_xxxx.obj`.
- **CTRL+L**: Show/hide outline.
- **CTRL+P**: Print current 3D pose on stderr.
- **CTRL+T**: Switch between single/double-sided 3D modes.
- **CTRL+V**: Start animation with video output.
- **CTRL+X**: Show/hide 3D bounding box.
- **CTRL+Z**: Enable/disable z-buffered rendering.

## Adding Custom Commands

- New custom commands can be added by the user, through the use of **G'MIC custom commands files**.
- A command file is a simple text file, where each line starts either by

```
command_name: command_definition
```

or

```
command_definition (continuation)
```

- At startup, G'MIC automatically includes user's command file `$HOME/.gmic` (on **Unix**) or `%APPDATA%/user.gmic` (on **Windows**). The CLI tool `gmic` automatically runs the command `cli_start` if defined.
- Custom command names must use character set `[a-zA-Z0-9_]` and cannot start with a number.
- Any `# comment` expression found in a custom commands file is discarded by the G'MIC parser, wherever it is located in a line.

- In a custom command, the following **\$-expressions** are recognized and substituted:
  - `$*` is substituted by a copy of the specified string of arguments.
  - `$*"**` is substituted by a copy of the specified string of arguments, each being double-quoted.
  - `$#` is substituted by the maximum index of known arguments (either specified by the user or set to a default value in the custom command).
  - `$[]` is substituted by the list of selected image indices that have been specified in the command invocation.
  - `$?` is substituted by a printable version of `$[]` to be used in command descriptions.
  - `$i` and  `${i}` are both substituted by the `i`-th specified argument. Negative indices such as  `${-j}` are allowed and refer to the `j`-th latest argument. `$0` is substituted by the custom command name.
  - `${i=default}` is substituted by the value of `$i` (if defined) or by its new value set to `default` otherwise (`default` may be a **\$-expression** as well).
  - `${subset}` is substituted by the argument values (separated by commas `,`) of a specified argument subset. For instance expression  `${2--2}` is substituted by all specified command arguments except the first and the last one. Expression  `${^0}` is then substituted by all arguments of the invoked command (eq. to `$*` if all arguments have been indeed specified).
  - `$=var` is substituted by the set of instructions that will assign each argument `$i` to the named variable `var$i` (for `i` in `[0...$#]`). This is particularly useful when a custom command want to manage variable numbers of arguments. Variables names must use character set `[a-zA-Z0-9_]` and cannot start with a number.
- These particular **\$-expressions** for custom commands are **always substituted**, even in double-quoted items or when the dollar sign `$` is escaped with a backslash `\$`. To avoid substitution, place an empty double quoted string just after the `$` (as in  `$"1`).
- Specifying arguments may be skipped when invoking a custom command, by replacing them by commas `,` as in expression
 

```
flower , ,3
```

 Omitted arguments are set to their default values, which must be thus explicitly defined in the code of the corresponding custom command (using default argument expressions as  `${1=default}`).
- If one numbered argument required by a custom command misses a value, an error is thrown by the G'MIC interpreter.
- It is possible to specialize the invocation of a `+command` by defining it as
 

```
+command_name: command_definition
```

 • A +-specialization takes priority over the regular command definition when the command is invoked with a prepended `+`.
- When only a +-specialization of a command is defined, invoking `command` is actually equivalent to `+command`.

## List of Commands

All available **G'MIC** commands are listed below, by categories. An argument specified between **[ ]** or starting by **\_** is optional except when standing for an existing image **[image]**, where **image** can be either an index number or an image name. In this case, the **[ ]** characters are mandatory when writing the item. Note that all images that serve as illustrations in this reference documentation are normalized in range **[0, 255]** before being displayed. You may need to do this explicitly (command **normalize 0, 255**) if you want to save and view images with the same aspect than those illustrated in the example codes.

## Categories:

- **Global Options**
- **Input / Output**
- **List Manipulation**
- **Mathematical Operators**
- **Values Manipulation**
- **Colors**
- **Geometry Manipulation**
- **Filtering**
- **Features Extraction**
- **Image Drawing**
- **Matrix Computation**
- **3D Meshes**
- **Control Flow**
- **Neural Networks**
- **Arrays, Tiles and Frames**
- **Artistic**
- **Warpings**
- **Degradations**
- **Blending and Fading**
- **Image Sequences and Videos**
- **Convenience Functions**
- **Other Interactive Commands**
- **Command Shortcuts**

## Global Options:

<b>debug</b>	<b>help</b>	<b>version</b>
--------------	-------------	----------------

## Input / Output:

<b>camera</b>	<b>clut</b>	<b>command</b>	<b>cursor</b>	<b>delete</b>
<b>display</b>	<b>display0</b>	<b>display2d</b>	<b>display3d</b>	<b>display_array</b>
<b>display_camera</b>	<b>display_fft</b>	<b>display_graph</b>	<b>display_histogram</b>	<b>display_parametric</b>
<b>display_parallel</b>	<b>display_parallel0</b>	<b>display_polar</b>	<b>display_quiver</b>	<b>display_rgba</b>
<b>display_tensors</b>	<b>display_warp</b>	<b>echo</b>	<b>echo_file</b>	<b>function1d</b>

input	input_565	input_csv	input_cube	input_flo
input_glob	input_gpl	input_cached	input_text	lorem
network	output	output_565	output_cube	output_flo
output_ggr	output_gmz	output_obj	output_text	outputn
outputp	outputw	outputx	parse_cli	parse_gmd
gmd2html	gmd2ascii	parse_gui	pass	plot
print	random_pattern	screen	select	serialize
shape_circle	shape_cupid	shape_diamond	shape.dragon	shape_fern
shape_gear	shape_heart	shape_polygon	shape_snowflake	shape_star
shared	sample	srand	store	testimage2d
uncommand	uniform_distribution	unserialize	update	verbose
wait	warn	window		

## List Manipulation:

keep	move	name	remove	remove_duplicates
remove_empty	remove_name_d	reverse	sort_list	

## Mathematical Operators:

abs	acos	acosh	add	and
argmax	argmaxabs	argmin	argminabs	asin
asinh	atan	atan2	atanh	bsl
bsr	cos	cosh	deg2rad	div
div_complex	eq	erf	exp	ge
gt	le	lt	log	log10
log2	max	maxabs	mdiv	med
min	minabs	mod	mmul	mul
mul_channels	mul_complex	neq	or	pow
rad2deg	rol	ror	sign	sin
sinc	sinh	sqr	sqrt	sub
tan	tanh	xor		

## Values Manipulation:

apply_curve	apply_gamma	balance_gamma	cast	complex2polar
compress_clut	compress_rle	cumulate	cut	decompress_clut
decompress_c_lut_rbf	decompress_c_lut_pde	decompress_rle	discard	eigen2tensor
endian	equalize	fill	index	inrange
map	mix_channels	negate	noise	noise_perlin
noise_poisson_disk	normp	norm	normalize	normalize_l2
normalize_sum	not	orientation	oneminus	otsu
polar2complex	quantize	quantize_area	rand	replace
replace_inf	replace_nan	replace_naninf	replace_seq	replace_str
round	roundify	set	threshold	vector2tensor

## Colors:

adjust_colors	apply_channels	autoindex	bayer2rgb	deltaE
cmy2rgb	cmyk2rgb	colorblind	colormap	compose_channels
direction2rgb	ditheredb	fill_color	gradient2rgb	hcy2rgb
hsi2rgb	hsi82rgb	hsl2rgb	hsl82rgb	hsv2rgb
hsv2rgb	int2rgb	jzazbz2rgb	jzazbz2xyz	lab2lch
lab2rgb	lab2srgb	lab82srgb	lab2xyz	lab82rgb
lch2lab	lch2rgb	lch82rgb	luminance	lightness
lut_contrast	map_clut	mix_rgb	oklab2rgb	palette
pseudogray	replace_color	retinex	rgb2bayer	rgb2cmy
rgb2cmyk	rgb2hcy	rgb2hs	rgb2hs8	rgb2hsl
rgb2hsl8	rgb2hsv	rgb2hsv8	rgb2int	rgb2jzazbz
rgb2lab	rgb2lab8	rgb2lch	rgb2lch8	rgb2luv
rgb2oklab	rgb2ryb	rgb2srgb	rgb2xyz	rgb2xyz8
rgb2yiq	rgb2yiq8	rgb2ycbcr	rgb2yuv	rgb2yuv8
remove_opacity	ryb2rgb	select_color	sepia	solarize
split_colors	split_opacity	srgb2lab	srgb2lab8	srgb2rgb
to_a	to_color	to_colormode	to_gray	to_graya
to_pseudogray	to_rgb	to_rgba	transfer_histogram	transfer_pca
transfer_rgb	xyz2jzazbz	xyz2lab	xyz2rgb	xyz82rgb
ycbcr2rgb	yiq2rgb	yiq82rgb	yuv2rgb	yuv82rgb

# Geometry Manipulation:

<code>append</code>	<code>append_tiles</code>	<code>apply_scales</code>	<code>autocrop</code>	<code>autocrop_components</code>
<code>autocrop_seq</code>	<code>channels</code>	<code>columns</code>	<code>crop</code>	<code>diagonal</code>
<code>elevate</code>	<code>expand_x</code>	<code>expand_xy</code>	<code>expand_xyz</code>	<code>expand_y</code>
<code>expand_z</code>	<code>extract</code>	<code>extract_region</code>	<code>montage</code>	<code>mirror</code>
<code>permute</code>	<code>resize</code>	<code>resize_as_image</code>	<code>resize_mn</code>	<code>resize_pow2</code>
<code>resize_ratio2d</code>	<code>resize2din</code>	<code>resize3din</code>	<code>resize2dout</code>	<code>resize3dout</code>
<code>resize2dx</code>	<code>resize2dy</code>	<code>resize3dx</code>	<code>resize3dy</code>	<code>resize3dz</code>
<code>rotate</code>	<code>rotate_tileable</code>	<code>rows</code>	<code>scale2x</code>	<code>scale3x</code>
<code>scale_dcci2x</code>	<code>seamcarve</code>	<code>shift</code>	<code>shrink_x</code>	<code>shrink_xy</code>
<code>shrink_xyz</code>	<code>shrink_y</code>	<code>shrink_z</code>	<code>slices</code>	<code>sort</code>
<code>split</code>	<code>split_tiles</code>	<code>undistort</code>	<code>unroll</code>	<code>upscale_smart</code>
<code>warp</code>	<code>warp_patch</code>	<code>warp_rbf</code>		

# Filtering:

<code>bandpass</code>	<code>bilateral</code>	<code>blur</code>	<code>blur_angular</code>	<code>blur_bloom</code>
<code>blur_linear</code>	<code>blur_radial</code>	<code>blur_selective</code>	<code>blur_x</code>	<code>blur_xy</code>
<code>blur_xyz</code>	<code>blur_y</code>	<code>blur_z</code>	<code>boxfilter</code>	<code>bump2normal</code>
<code>compose_freq</code>	<code>convolve</code>	<code>convolve_fft</code>	<code>correlate</code>	<code>cross_correlation</code>
<code>curvature</code>	<code>dct</code>	<code>deblur</code>	<code>deblur_goldmean</code>	<code>deblur_richardsonlucy</code>
<code>deconvolve_fft</code>	<code>deinterlace</code>	<code>denoise</code>	<code>denoise_haar</code>	<code>denoise_cnn</code>
<code>denoise_patch_pca</code>	<code>deriche</code>	<code>dilate</code>	<code>dilate_circ</code>	<code>dilate_oct</code>
<code>dilate_threshold</code>	<code>divergence</code>	<code>dog</code>	<code>diffusion_tensors</code>	<code>edges</code>
<code>erode</code>	<code>erode_circ</code>	<code>erode_oct</code>	<code>erode_threshold</code>	<code>fft</code>
<code>gradient</code>	<code>gradient_norm</code>	<code>gradient_orientation</code>	<code>guided</code>	<code>haar</code>
<code>heat_flow</code>	<code>hessian</code>	<code>idct</code>	<code>iee</code>	<code>ifft</code>
<code>ihaar</code>	<code>ilaplacian</code>	<code>inn</code>	<code>inpaint</code>	<code>inpaint_pde</code>
<code>inpaint_flow</code>	<code>inpaint_holes</code>	<code>inpaint_morpho</code>	<code>inpaint_matchpatch</code>	<code>kuwahara</code>
<code>laplacian</code>	<code>lic</code>	<code>map_tones</code>	<code>map_tones_fast</code>	<code>meancurvature_flow</code>
<code>median</code>	<code>nlmeans</code>	<code>nlmeans_core</code>	<code>normalize_local</code>	<code>normalized_cross_correlation</code>

percentile	peronamalik_flow	phase_correlation	pde_flow	periodize_poisson
rbf	red_eye	remove_hotpixels	remove_pixels	rolling_guidance
sharpen	smooth	split_freq	solve_poisson	split_details
structuretensors	solidify	syntexturize	syntexturize_matchpatch	tv_flow
unsharp	unsharp_octave	vanvliet	voronoi	watermark_fo urier
watershed				

## Features Extraction:

area	area_fg	at_line	at_quadrangle	barycenter
delaunay	detect_skin	displacement	distance	fftpolar
histogram	histogram_nd	histogram_cumul	histogram_pointwise	hough
fftpolar	isophotes	label	label_fg	laar
max_patch	min_patch	minimal_path	mse	patches
matchpatch	plot2value	pointcloud	psnr	segment_watershed
shape2bump	skeleton	slic	ssd_patch	thinning
tones	topographic_map	tsp	variance_patch	

## Image Drawing:

arrow	axes	ball	chessboard	cie1931
circle	close_binary	ellipse	flood	gaussian
graph	grid	image	line	linethick
mandelbrot	marble	maze	maze_mask	newton_fractal
object3d	pack_sprites	piechart	plasma	point
polka_dots	polygon	quiver	rectangle	rorschach
sierpinski	spiralbw	spline	tetraedron_shade	text
text_outline	triangle_shade	truchet	turbulence	yinyang

## Matrix Computation:

dijkstra	eigen	invert	orthogonalize	meigen

mproj	solve	svd	transpose	trisolve
-------	-------	-----	-----------	----------

## 3D Meshes:

add3d	animate3d	apply_camera3d	apply_matrix3d	array3d
arrow3d	axes3d	boundingbox3d	box3d	center3d
circle3d	circles3d	color3d	colorcube3d	cone3d
cubes3d	cup3d	cylinder3d	delaunay3d	distribution3d
div3d	double3d	elevation3d	empty3d	extrude3d
focale3d	gaussians3d	gmic3d	gyroid3d	histogram3d
image6cube3d	imageblocks3d	imagecube3d	imageplane3d	imagepyramid3d
imagerubik3d	imagesphere3d	isoline3d	isosurface3d	label3d
label_points3d	lathe3d	light3d	line3d	lissajous3d
mode3d	moded3d	mul3d	normalize3d	opacity3d
parametric3d	pca_patch3d	plane3d	point3d	pointcloud3d
pose3d	primitives3d	projections3d	pyramid3d	quadrangle3d
random3d	reverse3d	rotate3d	rotation3d	sierpinski3d
size3d	skeleton3d	snapshot3d	spec13d	specs3d
sphere3d	spherical3d	spline3d	split3d	sprite3d
sprites3d	star3d	streamline3d	sub3d	superformula3d
tensors3d	text_pointcloud3d	text3d	texturize3d	torus3d
triangle3d	volume3d	weird3d		

## Control Flow:

apply_parallel	apply_parallel_channels	apply_parallel_overlap	apply_tiles	apply_timeout
check	check3d	continue	break	do
done	elif	else	fi	endlocal
error	eval	exec	exec_out	for
if	local	mutex	noarg	onfail
parallel	progress	quit	repeat	return
rprogress	run	skip	status	while

## Neural Networks:

nn_lib	nn_init	nn_check_layer	nn_layer_input	nn_layer_add
nn_layer_append	nn_layer_avgpool2d	nn_layer_batcnorm	nn_layer_clone	nn_layer_conv2d
nn_layer_conv2dbnnl	nn_layer_conv2dnl	nn_layer_crop	nn_layer_fc	nn_layer_fcbn_nl
nn_layer_fcnl	nn_layer_maxpool2d	nn_layer_nl	nn_layer_rename	nn_layer_reshape
nn_layer_resize	nn_layer_run	nn_layer_split	nn_loss_mse	nn_trainer
nn_load	nn_save			

## Arrays, Tiles and Frames:

array	array_fade	array_mirror	array_random	frame_blur
frame_cube	frame_fuzzy	frame_painting	frame_pattern	frame_round
frame_seamless	frame_x	frame_xy	frame_xyz	frame_y
img2ascii	imagegrid	imagegrid_hexagonal	imagegrid_triangular	linearize_tiles
map_sprites	pack	puzzle	quadratize_tiles	rotate_tiles
shift_tiles	taquin	tunnel		

## Artistic:

boxfitting	brushify	cartoon	color_ellipses	cubism
draw_whirl	drawing	drop_shadow	ellipsionism	fire_edges
fractalize	glow	halftone	hardsketchbw	hearts
oughsketchbw	lightrays	light_relief	linify	mosaic
old_photo	pencilbw	pixelsort	polaroid	Polygonize
poster_edges	poster_hope	rodilius	sketchbw	sponge
stained_glass	stars	stencil	stencilbw	stylize
tetris	warhol	weave	whirls	

## Warpings:

deform	euclidean2polar	equirectangular2nadirzenith	fisheye	flower
kaleidoscope	map_sphere	nadirzenith2e	polar2euclide	raindrops

		quirectangular	an	
ripple	rotoidoscope	spherize	symmetrize	transform_polar
twirl	warp_perspective	water	wave	wind
zoom				

## Degradations:

cracks	light_patch	noise_hurl	pixelize	scanlines
shade_stripes	shadow_patch	spread	stripes_y	texturize_canv as
texturize_paper	vignette	watermark_vis ible		

## Blending and Fading:

blend	blend_edges	blend_fade	blend_median	blend_seamles s
fade_diamond	fade_linear	fade_radial	fade_x	fade_y
fade_z	sub_alpha			

## Image Sequences and Videos:

animate	apply_camera	apply_files	apply_video	average_files
average_video	fade_files	fade_video	files2video	median_files
median_video	morph	morph_files	morph_rbf	morph_video
register_nonrigid	register_rigid	transition	transition3d	video2files

## Convenience Functions:

alert	arg	arg0	arg2img	arg2var
autocrop_coordinates	average_colors	base642img	base642uchar	basename
bin	bin2dec	covariance_colors	dec	dec2str
dec2bin	dec2hex	dec2oct	fact	fibonacci
file_mv	file_rand	filename	files	files2img
fitratio_wh	fitscreen	fontchart	fps	gcd

hex	hex2dec	hex2img	hex2str	img2base64
img2hex	img2str	img2text	img82hex	hex2img8
is_3d	is_change	is_half	is_ext	is_image_arg
is_pattern	is_percent	is_videofilena me	is_macos	is_windows
math_lib	mad	max_w	max_h	max_d
max_s	max_wh	max_whd	max_whds	median_color
min_w	min_h	min_d	min_s	min_wh
min_whd	min_whds	named	normalize_file name	oct
oct2dec	padint	path_cache	path_current	path_gimp
path_tmp	remove_copy mark	reset	rgb	rgba
shell_cols	size_value	std_noise	str	str2hex
strcapitalize	strcontains	strlen	strreplace	strlowercase
strupercase	strvar	strcasevar	strver	tic
toc	to_clutname	uchar2base64		

## Other Interactive Commands:

demos	tixy	x_2048	x_blobs	x_bouncing
x_color_curves	x_colorize	x_connect4	x_crop	x_cut
x_fire	x_fireworks	x_fisheye	x_fourier	x_grab_color
x_hanoi	x_histogram	x_hough	x_jawbreaker	x_landscape
x_life	x_light	x_mandelbrot	x_mask_color	x_metaballs3d
x_minesweeper	x_minimal_path	x_morph	x_pacman	x_paint
x_plasma	x_quantize_rg b	x_reflection3d	x_rubber3d	x_segment
x_select_color	x_select_functi on1d	x_select_palett e	x_shadebobs	x_spline
x_starfield3d	x_tetris	x_threshold	x_tictactoe	x_warp
x_waves	x_whirl			

## Command Shortcuts:

Shortcut name	Equivalent command name
<b>h</b>	help
<b>m</b>	command
<b>d</b>	display
<b>d0</b>	display0
<b>d2d</b>	display2d
<b>d3d</b>	display3d

<b>da</b>	display_array
<b>dc</b>	display_camera
<b>dfft</b>	display_fft
<b>dg</b>	display_graph
<b>dh</b>	display_histogram
<b>dp</b>	display_parallel
<b>dp0</b>	display_parallel0
<b>dq</b>	display_quiver
<b>drgba</b>	display_rgba
<b>dt</b>	display_tensors
<b>dw</b>	display_warp
<b>e</b>	echo
<b>i</b>	input
<b>ig</b>	input_glob
<b>it</b>	input_text
<b>o</b>	output
<b>ot</b>	output_text
<b>on</b>	outputn
<b>op</b>	outputp
<b>ow</b>	outputw
<b>ox</b>	outputx
<b>p</b>	print
<b>sh</b>	shared
<b>sp</b>	sample
<b>um</b>	uncommand
<b>up</b>	update
<b>v</b>	verbose
<b>w</b>	window
<b>k</b>	keep
<b>mv</b>	move
<b>nm</b>	name
<b>rm</b>	remove
<b>rmn</b>	remove_named
<b>rv</b>	reverse
<b>+</b>	add
<b>&amp;</b>	and
<b>&lt;&lt;</b>	bsl
<b>&gt;&gt;</b>	bsr
<b>/</b>	div
<b>==</b>	eq
<b>&gt;=</b>	ge
<b>&gt;</b>	gt
<b>&lt;=</b>	le

<	lt
m/	mdiv
%	mod
m*	mmul
*	mul
!=	neq
	or
^	pow
-	sub
c	cut
f	fill
ir	inrange
n	normalize
=	set
ac	apply_channels
fc	fill_color
a	append
z	crop
r	resize
ri	resize_as_image
rr2d	resize_ratio2d
r2din	resize2din
r3din	resize3din
r2dout	resize2dout
r3dout	resize3dout
r2dx	resize2dx
r2dy	resize2dy
r3dx	resize3dx
r3dy	resize3dy
r3dz	resize3dz
s	split
y	unroll
b	blur
g	gradient
j	image
j3d	object3d
t	text
to	text_outline
+3d	add3d
c3d	center3d
col3d	color3d
/3d	div3d

db3d	double3d
f3d	focale3d
l3d	light3d
m3d	mode3d
md3d	moded3d
*3d	mul3d
n3d	normalize3d
o3d	opacity3d
p3d	primitives3d
rv3d	reverse3d
r3d	rotate3d
sl3d	spec13d
ss3d	specs3d
s3d	split3d
-3d	sub3d
t3d	texturize3d
ap	apply_parallel
apc	apply_parallel_channels
apo	apply_parallel_overlap
at	apply_tiles
endl	endlocal
x	exec
xo	exec_out
l	local
q	quit
u	status
frame	frame_xy
nmd	named
xz	x_crop

## Examples of Use

`gmic` is a generic image processing tool which can be used in a wide variety of situations. The few examples below illustrate possible uses of this tool:

### View a list of images:

```
$ gmic file1.bmp file2.jpeg
```

### Convert an image file:

```
$ gmic input.bmp output output.jpg
```

Create a volumetric image from a movie sequence:

```
$ gmic input.mpg append z output output.hdr
```

Compute image gradient norm:

```
$ gmic input.bmp gradient_norm
```

Denoise a color image:

```
$ gmic image.jpg denoise 30,10 output denoised.jpg
```

Compose two images using overlay layer blending:

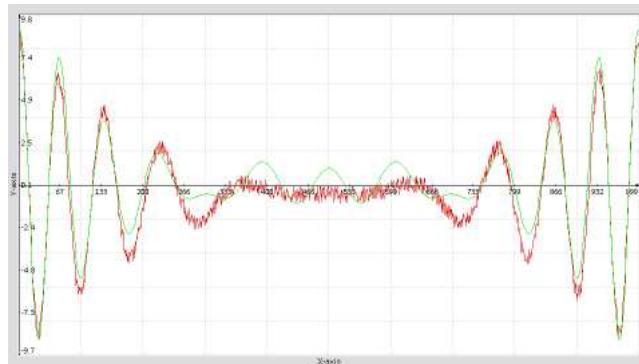
```
$ gmic image1.jpg image2.jpg blend overlay output blended.jpg
```

Evaluate a mathematical expression:

```
$ gmic echo "cos(pi/4)^2+sin(pi/4)^2={cos(pi/4)^2+sin(pi/4)^2}"
```

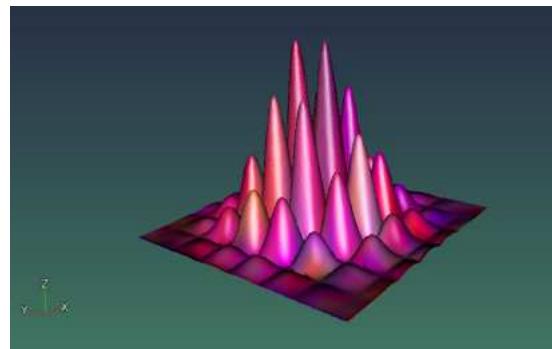
Plot a 2D function:

```
$ gmic 1000,1,1,2 fill "X=3*(x-500)/500;X^2*sin(3*X^2)+if(c==0,u(0,-1),cos(X*10))" plot
```



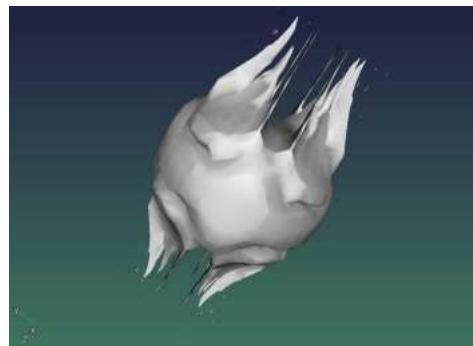
Plot a 3D elevated function in random colors:

```
$ gmic 128,128,1,3,"u(0,255)" plasma 10,3 blur 4 sharpen 10000 n 0,255 elevate "(x-64)/6;Y=(y-64)/6;100*exp(-(X^2+Y^2)/30)*abs(cos(X)*sin(Y))'"
```



Plot the isosurface of a 3D volume:

```
$ gmic mode3d 5 moded3d 5 double3d 0 isosurface3d "'x^2+y^2+abs(z)^abs(4*cos(pi*x))'
```



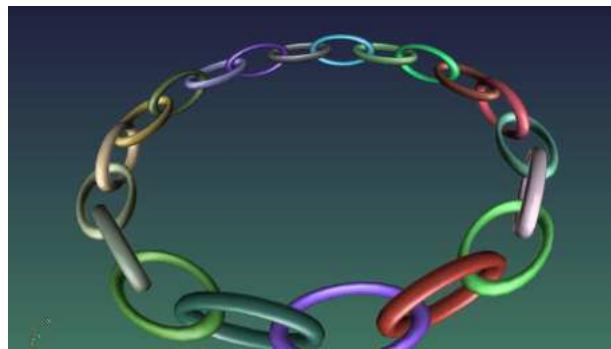
Render a G'MIC 3D logo:

```
$ gmic 0 text G\'MIC,0,0,53,1,1,1,1 expand_xy 10,0 blur 1 normalize 0,100 +r
```



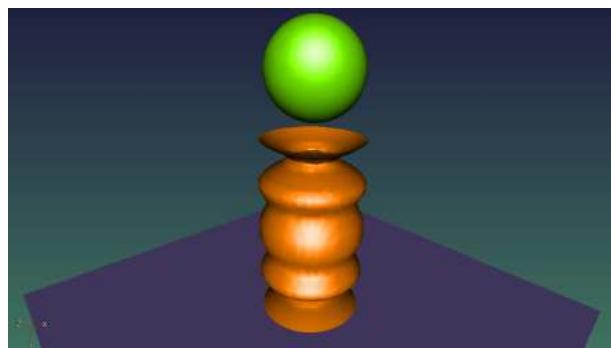
Generate a 3D ring of torii:

```
$ gmic repeat 20 torus3d 15,2 color3d[-1] "{u(60,255)},{u(60,255)}, {u(60,255)}" *3d[-1] 0.5,1 if " ${$>%2}" rotate3d[-1] 0,1,0,90 fi add3d[-1] 70 add3d rotate3d 0,0,1,18 done r
```



Create a vase from a 3D isosurface:

```
$ gmic moded3d 4 isosurface3d "'x^2+2*abs(y/2)*sin(2*y)^2+z^2-3',0" sphere3d 1.5 sub3d[-1] 0,5 plane3d 15,15 rotate3d[-1] 1,0,0,90 center
```



Launch a set of interactive demos:

```
$ gmic demos
```

---

## abs

Built-in command

No arguments

**Description:**

Compute the pointwise absolute values of selected images.

**Examples of use:**

- **Example #1**

```
$ gmic image.jpg +sub {ia} abs[-1]
```



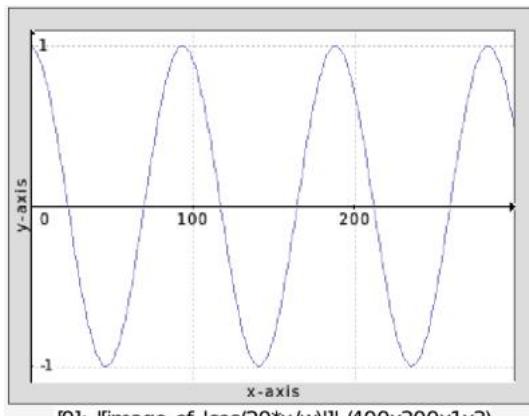
[0]: 'image.jpg' (640x427x1x3)



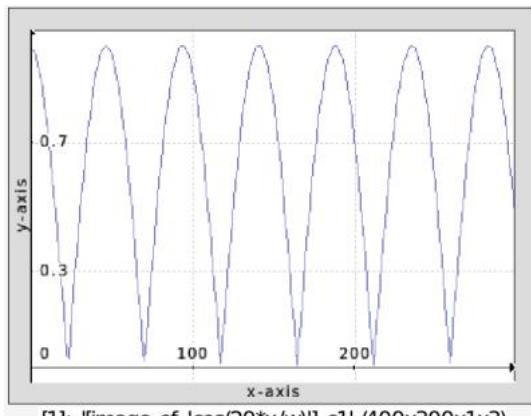
[1]: 'image\_c1.jpg' (640x427x1x3)

- **Example #2**

```
$ gmic 300,1,1,1,'cos(20*x/w)' +abs display_graph 400,300
```



[0]: '[image of "cos(20\*x/w)"]' (400x300x1x3)



[1]: '[image of "cos(20\*x/w)"]\_c1' (400x300x1x3)

---

## ACOS

Built-in command

### No arguments

### Description:

Compute the pointwise arccosine of selected images.

This command has a [tutorial page](#).

### Examples of use:

- **Example #1**

```
$ gmic image.jpg +normalize -1,1 acos[-1]
```



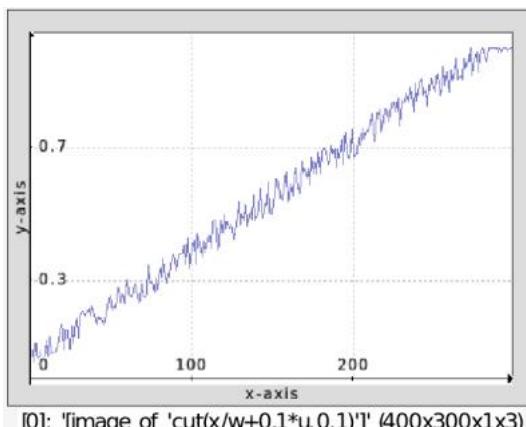
[0]: 'image.jpg' (640x427x1x3)



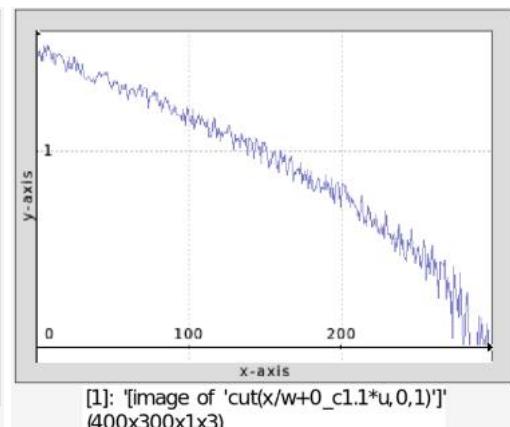
[1]: 'image\_c1.jpg' (640x427x1x3)

## • Example #2

```
$ gmic 300,1,1,1,'cut(x/w+0.1*u,0,1)' +acos display_graph 400,300
```



[0]: '[image of 'cut(x/w+0.1\*u,0,1)']' (400x300x1x3)



[1]: '[image of 'cut(x/w+0\_c1.1\*u,0,1)']' (400x300x1x3)

---

## acosh

Built-in command

### No arguments

### Description:

Compute the pointwise hyperbolic arccosine of selected images.

---

## add

Built-in command

### Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

### Description:

Add specified value, image or mathematical expression to selected images, or compute the pointwise sum of selected images.

(equivalent to shortcut command `[+]`).

## Examples of use:

- **Example #1**

```
$ gmic image.jpg +add 30% cut 0,255
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

- **Example #2**

```
$ gmic image.jpg +blur 5 normalize 0,255 add[1] [0]
```



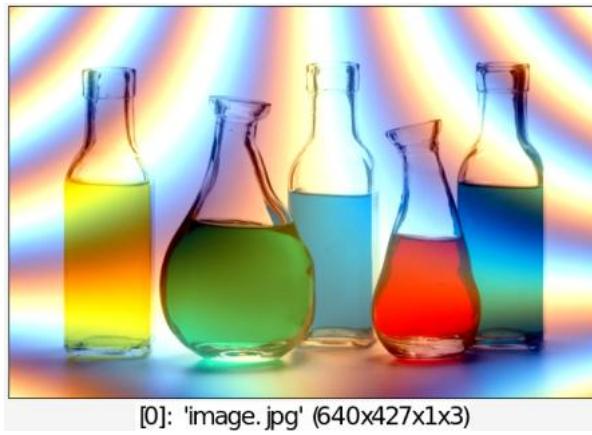
[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

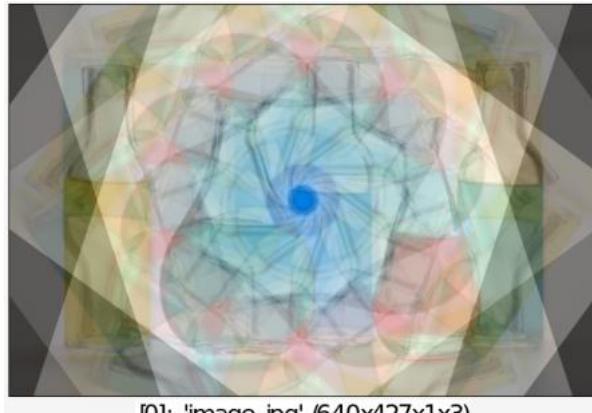
- **Example #3**

```
$ gmic image.jpg add '80*cos(80*(x/w-0.5)*(y/w-0.5)+c)' cut 0,255
```



- **Example #4**

```
$ gmic image.jpg repeat 9 +rotate[0] {$>*36},1,0,50%,50% done add div  
10
```



---

## add3d

Built-in command

### Arguments:

- `tx,_ty,_tz` or
- `[object3d]` or
- `(no arg)`

### Description:

Shift selected 3D objects with specified displacement vector, or merge them with specified 3D object, or merge all selected 3D objects together.

(equivalent to shortcut command `+3d`).

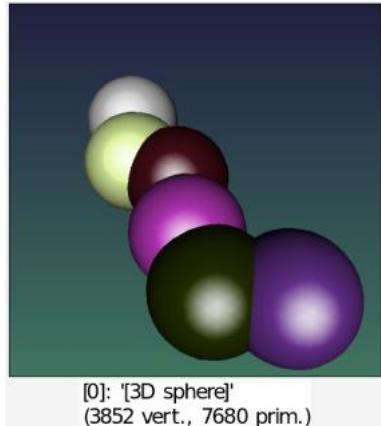
### Default values:

`ty=tz=0`.

### Examples of use:

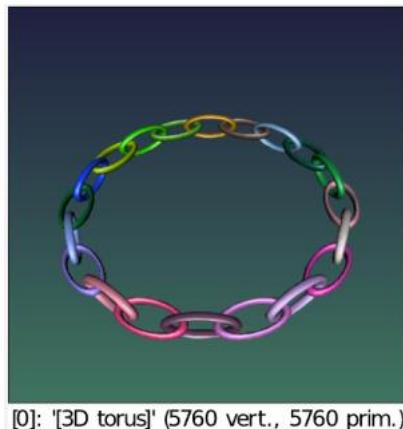
- **Example #1**

```
$ gmic sphere3d 10 repeat 5 +add3d[-1] 10,{u(-10,10)},0 color3d[-1]
 ${-rgb} done add3d
```



- **Example #2**

```
$ gmic repeat 20 torus3d 15,2 color3d[-1] ${-rgb} mul3d[-1] 0.5,1 if
 $>%2 rotate3d[-1] 0,1,0,90 fi add3d[-1] 70 add3d rotate3d[-1]
 0,0,1,18 done double3d 0
```



---

## adjust\_colors

### Arguments:

- `-100<=_brightness<=100, -100<=_contrast<=100, -100<=_gamma<=100, -100<=_hue_shi`

### Description:

Perform a global adjustment of colors on selected images.

Range of correct image values are considered to be in [value\_min,value\_max] (e.g. [0,255]).

If `value_min==value_max==0`, value range is estimated from min/max values of selected images.  
Processed images have pixel values constrained in [value\_min,value\_max].

## Default values:

```
brightness=0, contrast=0, gamma=0, hue_shift=0, saturation=0,  
value_min=value_max=0.
```

## Example of use:

```
$ gmic image.jpg +adjust_colors 0,30,0,0,30
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## alert

### Arguments:

- `_title,_message,_label_button1,_label_button2,...`

### Description:

Display an alert box and wait for user's choice.

If a single image is in the selection, it is used as an icon for the alert box.

## Default values:

'title=[G'MIC Alert]' and 'message=This is an alert box.'

---

## and

Built-in command

### Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

## Description:

Compute the bitwise AND of selected images with specified value, image or mathematical expression, or compute the pointwise sequential bitwise AND of selected images.

(*equivalent to shortcut command* `&`).

## Examples of use:

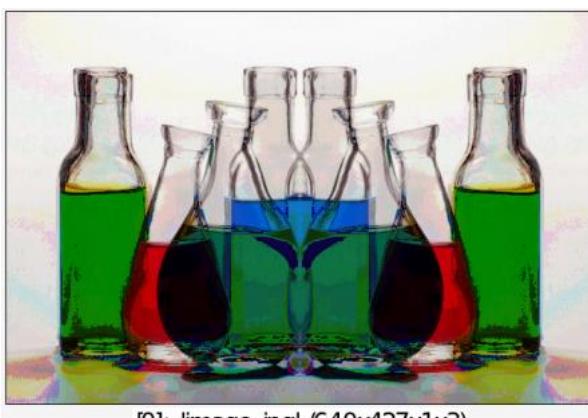
- Example #1

```
$ gmic image.jpg and {128+64}
```



- Example #2

```
$ gmic image.jpg +mirror x and
```



---

## animate

### Arguments:

- `filter_name,"param1_start,...,paramN_start","param1_end,...,paramN_end",nb_f`  
`{ 0 | 1 },_output_filename` or
- `delay>0,_back and forth={ 0 | 1 }`

## Description:

Animate filter from starting parameters to ending parameters or animate selected images in a display window.

## Default values:

`delay=30`.

## Example of use:

```
$ gmic image.jpg animate flower,"0,3","20,8",9
```



[0]: 'image\_c1.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)



[3]: 'image\_c1.jpg' (640x427x1x3)



[4]: 'image\_c1.jpg' (640x427x1x3)



[5]: 'image\_c1.jpg' (640x427x1x3)



[6]: 'image\_c1.jpg' (640x427x1x3)

[7]: 'image\_c1.jpg' (640x427x1x3)



[8]: 'image\_c1.jpg' (640x427x1x3)

---

## animate3d

### Arguments:

- `nb_frames>0, _step_angle_x, _step_angle_y, _step_angle_z, _zoom_factor, 0<=_fake_`

### Description:

Generate 3D animation frames of rotating 3D objects.

Frames are stacked along the z-axis (volumetric image).

Frame size is the same as the size of the `[background]` image (or 800x800 if no background specified).

### Default values:

`filename=(undefined)`.

---

## append

Built-in command

### Arguments:

- `[image], axis, _centering` or
- `axis, _centering`

## Description:

Append specified image to selected images, or all selected images together, along specified axis.

(equivalent to shortcut command `a`).

`axis` can be `{ x | y | z | c }`.

Usual `centering` values are `{ 0=left-justified | 0.5=centered | 1=right-justified }`.

## Default values:

`centering=0`.

## Examples of use:

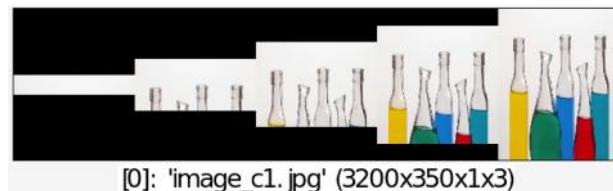
- Example #1

```
$ gmic image.jpg split y,10 reverse append y
```



- Example #2

```
$ gmic image.jpg repeat 5 +rows[0] 0,{10+18*$>}% done remove[0] append x,0.5
```



- Example #3

```
$ gmic image.jpg append[0] [0],y
```



---

## append\_tiles

### Arguments:

- `_M>=0, _N>=0, 0<=_centering_x<=1, 0<=_centering_y<=1`

### Description:

Append MxN selected tiles as new images.

If `N` is set to 0, number of rows is estimated automatically.

If `M` is set to 0, number of columns is estimated automatically.

If `M` and `N` are both set to `0`, auto-mode is used.

If `M` or `N` is set to 0, only a single image is produced.

`centering_x` and `centering_y` tells about the centering of tiles when they have different sizes.

### Default values:

`M=0, N=0, centering_x=centering_y=0.5`.

### Example of use:

```
$ gmic image.jpg split xy,4 append_tiles ,
```



# apply\_camera

## Arguments:

- `_command", _camera_index>=0, _skip_frames>=0, _output_filename`

## Description:

Apply specified command on live camera stream, and display it on display window [0].

This command requires features from the OpenCV library (not enabled in G'MIC by default).

## Default values:

`command=""`, `camera_index=0` (default camera), `skip_frames=0` and  
`output_filename=""`.

---

# apply\_camera3d

## Arguments:

- `pos_x, pos_y, pos_z, target_x, target_y, target_z, up_x, up_y, up_z`

## Description:

Apply 3D camera matrix to selected 3D objects.

## Default values:

`target_x=0`, `target_y=0`, `target_z=0`, `up_x=0`, `up_y=-1` and `up_z=0`.

---

# apply\_channels

## Arguments:

- `"command", color_channels, _value_action={ 0=none | 1=cut | 2=normalize }`

## Description:

Apply specified command on the chosen color channel(s) of each selected images.

(*equivalent to shortcut command `ac`*).

Argument `color_channels` refers to a colorspace, and can be basically one of  
`{ all | rgba | [s]rgb | ryb | lrgb | ycbcr | lab | lch | hsv | hsi | hsl | cmy | cmyk | yiq }`.

You can also make the processing focus on a few particular channels of this colorspace, by setting `color_channels` as `colorspace_channel` (e.g. `hsv_h` for the hue). All channel values are considered to be provided in the [0,255] range.

## Default values:

`value_action=0`.

## Example of use:

```
$ gmic image.jpg +apply_channels "equalize blur 2",ycbcr_cbc
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## apply\_curve

### Arguments:

- `0<=smoothness<=1,x0,y0,x1,y1,x2,y2,...,xN,yN`

### Description:

Apply curve transformation to image values.

## Default values:

`smoothness=1, x0=0, y0=100`.

## Example of use:

```
$ gmic image.jpg +apply_curve 1,0,0,128,255,255,0
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## apply\_files

### Arguments:

- `"filename_pattern", "command", _first_frame>=0, _last_frame={ >=0 | -1=last }, _frame_step>=1, _output_filename`

### Description:

Apply a G'MIC command on specified input image files, in a streamed way.

If a display window is opened, rendered frames are displayed in it during processing.  
The output filename may have extension `.avi` or `.mp4` (saved as a video), or any other usual image file extension (saved as a sequence of images).

### Default values:

`command=(undefined), first_frame=0, last_frame=-1, frame_step=1` and  
`output_filename=(undefined)`.

---

## apply\_gamma

### Arguments:

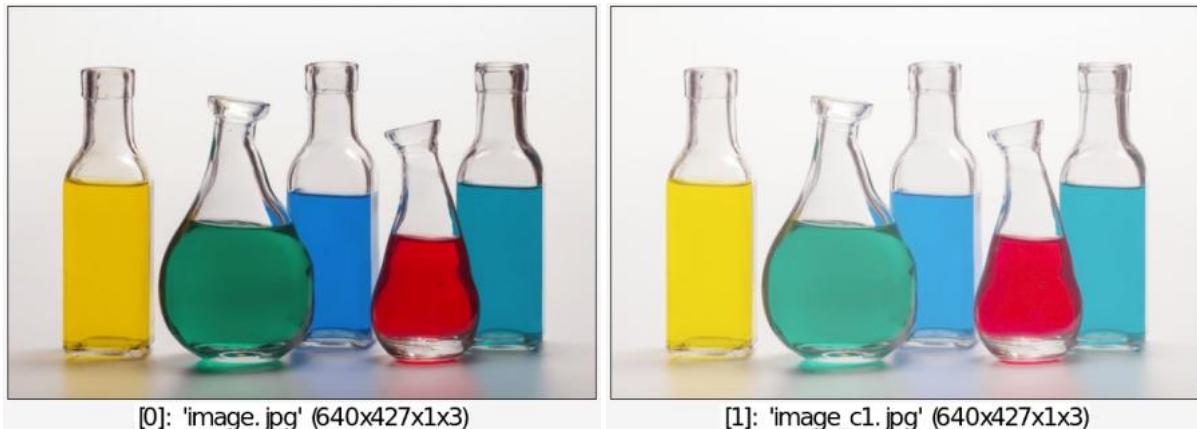
- `gamma>=0`

### Description:

Apply gamma correction to selected images.

### Example of use:

```
$ gmic image.jpg +apply_gamma 2
```



---

## apply\_matrix3d

### Arguments:

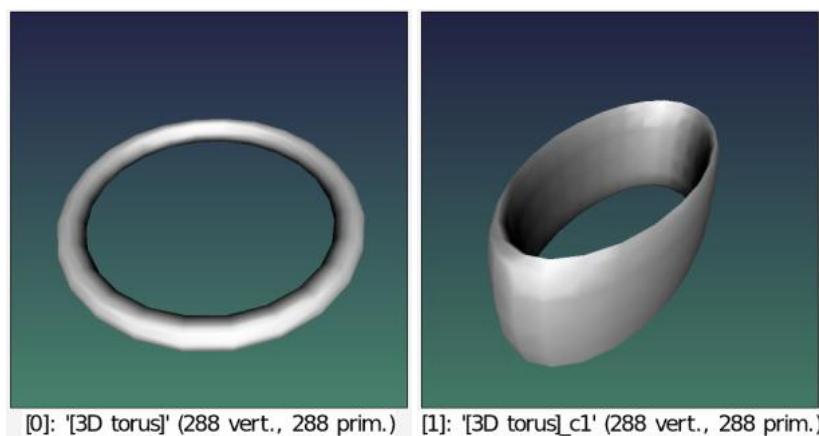
- `a11,a12,a13,...,a31,a32,a33`

### Description:

Apply specified 3D rotation matrix to selected 3D objects.

### Example of use:

```
$ gmic torus3d 10,1 +apply_matrix3d {mul(rot(1,0,1,-15°),  
[1,0,0,0,2,0,0,0,8],3)} double3d 0
```



---

## apply\_parallel

### Arguments:

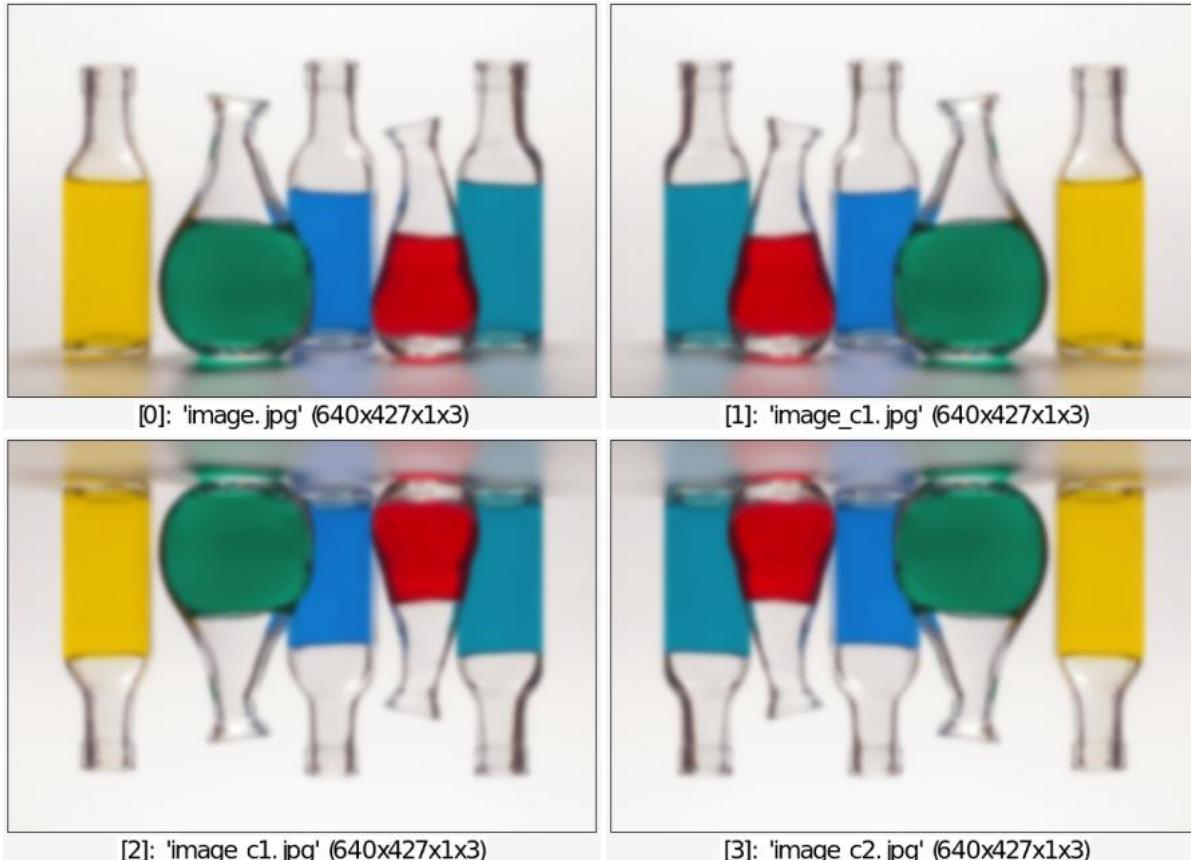
- `"command"`

### Description:

Apply specified command on each of the selected images, by parallelizing it for all image of the list.  
(equivalent to shortcut command `ap`).

## Example of use:

```
$ gmic image.jpg +mirror x +mirror y apply_parallel "blur 3"
```



---

## apply\_parallel\_channels

### Arguments:

- "command"

### Description:

Apply specified command on each of the selected images, by parallelizing it for all channel of the images independently.  
(equivalent to shortcut command `apc`).

## Example of use:

```
$ gmic image.jpg apply_parallel_channels "blur 3"
```



---

## apply\_parallel\_overlap

### Arguments:

- "command", overlap[%], nb\_threads={ 0=auto | 1 | 2 | 4 | 8 | 16 }

### Description:

Apply specified command on each of the selected images, by parallelizing it on **nb\_threads** overlapped sub-images.  
(equivalent to shortcut command [apo](#)).

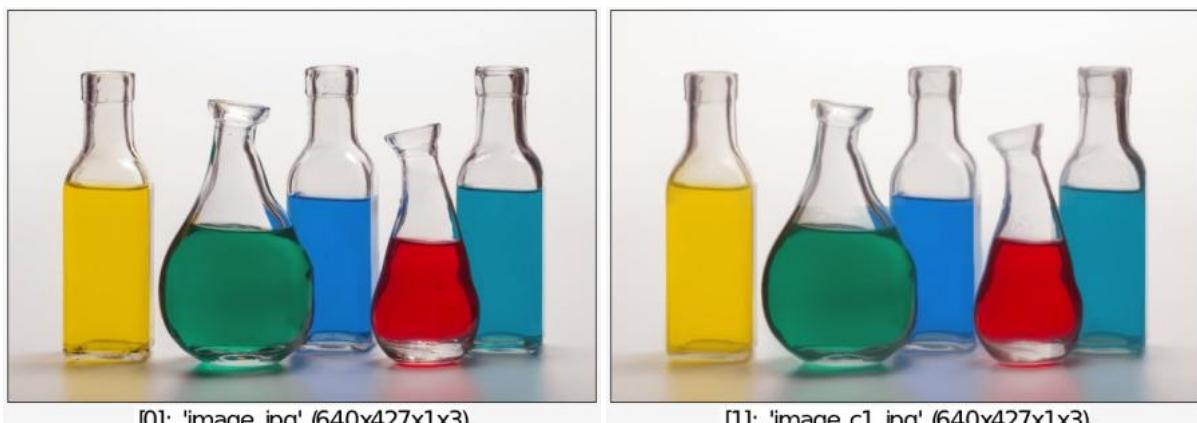
**nb\_threads** must be a power of 2.

### Default values:

**overlap=0**, **nb\_threads=0**.

### Example of use:

```
$ gmic image.jpg +apply_parallel_overlap "smooth 500,0,1",1
```



# apply\_scales

## Arguments:

- "command", `number_of_scales>0`, `_min_scale[%]>=0`, `_max_scale[%]>=0`, `_scale_gamma>0`

## Description:

Apply specified command on different scales of selected images.

`interpolation` can be { `0=none` | `1=nearest` | `2=average` | `3=linear` | `4=grid` | `5=bicubic` | `6=lanczos` }.

## Default values:

`min_scale=25%`, `max_scale=100%` and `interpolation=3`.

## Example of use:

```
$ gmic image.jpg apply_scales "blur 5 sharpen 1000",4
```



[0]: 'image.jpg' (160x107x1x3)



[1]: 'image.jpg' (320x214x1x3)



[2]: 'image.jpg' (480x320x1x3)



[3]: 'image.jpg' (640x427x1x3)

# apply\_tiles

## Arguments:

- `"command",_tile_width[%]>0,_tile_height[%]>0,_tile_depth[%]>0,_overlap_width { 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

## Description:

Apply specified command on each tile (neighborhood) of the selected images, eventually with overlapping tiles.

(equivalent to shortcut command `at`).

## Default values:

`tile_width=tile_height=tile_depth=10%,overlap_width=overlap_height=overlap_depth=0` and `boundary_conditions=1`.

## Example of use:

```
$ gmic image.jpg +equalize[0] 256 +apply_tiles[0] "equalize 256",16,16,1,50%,50%
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)



[2]: '[unnamed]\_c1' (640x427x1x3)

## apply\_timeout

### Arguments:

- `"command",_timeout={ 0=no timeout | >0=with specified timeout (in seconds) }`

## Description:

Apply a command with a timeout.

Set variable `$is_timeout` to `1` if timeout occurred, `0` otherwise.

## Default values:

`timeout=20`.

---

# apply\_video

## Arguments:

- `video_filename, "command", _first_frame>=0, _last_frame={ >=0 | -1=last }`, `_frame_step>=1, _output_filename`

## Description:

Apply a G'MIC command on all frames of the specified input video file, in a streamed way.

If a display window is opened, rendered frames are displayed in it during processing.  
The output filename may have extension `.avi` or `.mp4` (saved as a video), or any other usual image file extension (saved as a sequence of images).  
This command requires features from the OpenCV library (not enabled in G'MIC by default).

## Default values:

`first_frame=0`, `last_frame=-1`, `frame_step=1` and `output_filename=(undefined)`.

---

# area

## Arguments:

- `tolerance>=0, is_high_connectivity={ 0 | 1 }`

## Description:

Compute area of connected components in selected images.

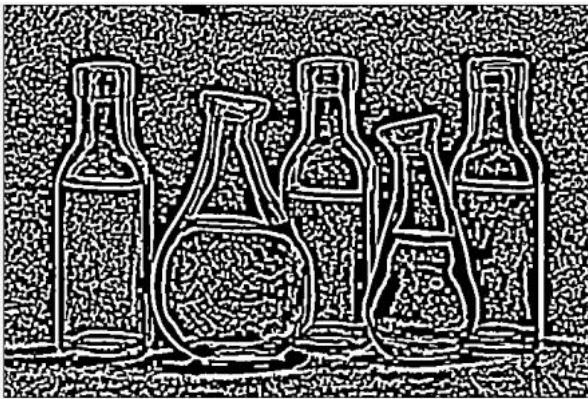
## Default values:

`is_high_connectivity=0`.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg luminance stencil[-1] 1 +area 0
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)

## area\_fg

### Arguments:

- `tolerance>=0, is_high_connectivity={ 0 | 1 }`

### Description:

Compute area of connected components for non-zero values in selected images.

Similar to `area` except that 0-valued pixels are not considered.

### Default values:

`is_high_connectivity=0`.

## Example of use:

```
$ gmic image.jpg luminance stencil[-1] 1 +area_fg 0
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)

# arg

## Arguments:

- `n>=1,_arg1,...,_argN`

## Description:

Return the n-th argument of the specified argument list.

---

# arg0

## Arguments:

- `n>=0,_arg0,...,_argN`

## Description:

Return the n-th argument of the specified argument list (where `n` starts from `0`).

---

# arg2img

## Arguments:

- `argument_1,...,argument_N`

## Description:

Split specified list of arguments and return each as a new image (as a null-terminated string).

---

# arg2var

## Arguments:

- `variable_name,argument_1,...,argument_N`

## Description:

For each i in [1...N], set `variable_name$i=argument_i`.

The variable name should be global to make this command useful (i.e. starts by an underscore).

---

# argmax

## No arguments

### Description:

Compute the argmax of selected images. Returns a single image

with each pixel value being the index of the input image with maximal value.

### Example of use:

```
$ gmic image.jpg sample lena,lion,square +argmax
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'lena' (512x512x1x3)



[2]: 'lion' (640x600x1x3)



[3]: 'square' (750x500x1x3)



[4]: '[argmax]\_c1' (750x600x1x3)

## argmaxabs

### No arguments

## Description:

Compute the argmaxabs of selected images. Returns a single image

with each pixel value being the index of the input image with maxabs value.

---

# argmin

## No arguments

## Description:

Compute the argmin of selected images. Returns a single image

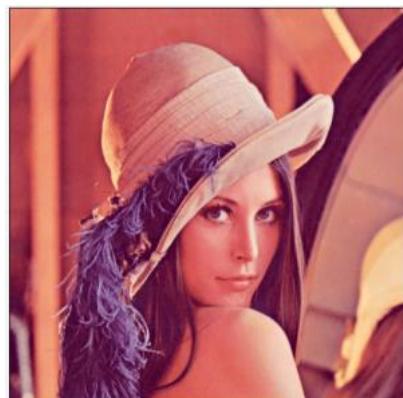
with each pixel value being the index of the input image with minimal value.

## Example of use:

```
$ gmic image.jpg sample lena,lion,square +argmin
```



[0]: 'image.jpg' (640x427x1x3)



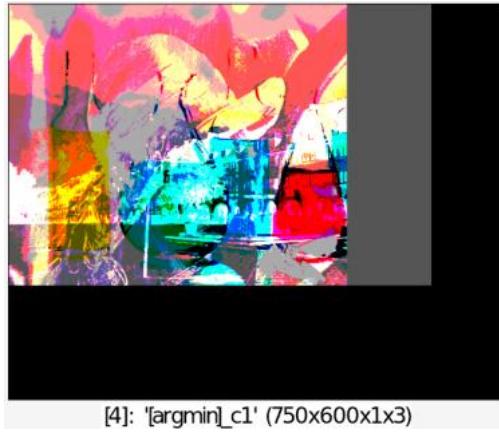
[1]: 'lena' (512x512x1x3)



[2]: 'lion' (640x600x1x3)



[3]: 'square' (750x500x1x3)



---

## argminabs

**No arguments**

**Description:**

Compute the argminabs of selected images. Returns a single image

with each pixel value being the index of the input image with minabs value.

---

## array

**Arguments:**

- `M>0, N>0, _expand_type={ 0=min | 1=max | 2=all }`

**Description:**

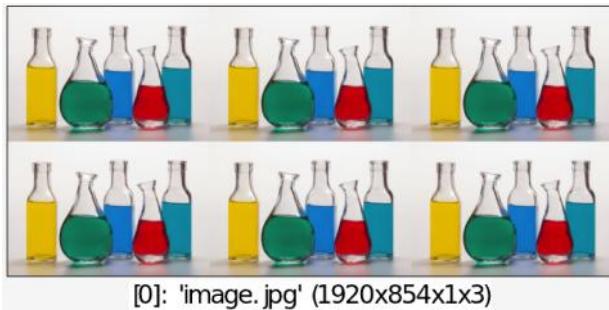
Create MxN array from selected images.

**Default values:**

`N=M` and `expand_type=0`.

**Example of use:**

```
$ gmic image.jpg array 3,2,2
```



---

## array3d

### Arguments:

- `_size_x>=1,_size_y>=1,_size_z>=1,_offset_x[%],_offset_y[%],_offset_z[%]`

### Description:

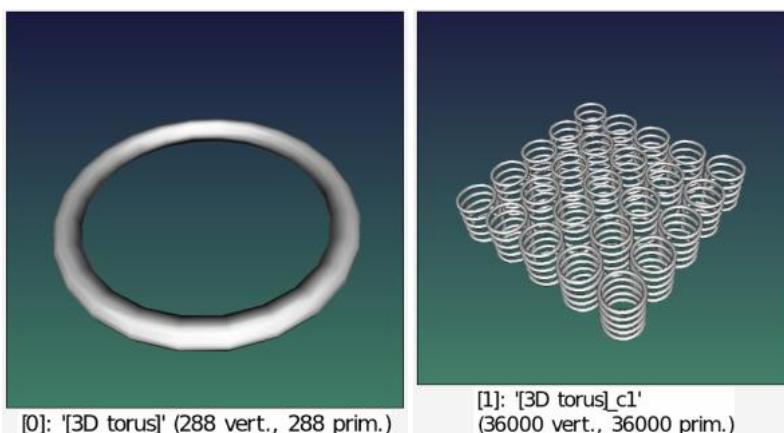
Duplicate a 3D object along the X,Y and Z axes.

### Default values:

`size_y=1`, `size_z=1` and `offset_x=offset_y=offset_z=100%`.

### Example of use:

```
$ gmic torus3d 10,1 +array3d 5,5,5,110%,110%,300%
```



---

## array\_fade

### Arguments:

- `M>0,_N>0,0<=_fade_start<=100,0<=_fade_end<=100,_expand_type={0=min | 1=max | 2=all}`

### Description:

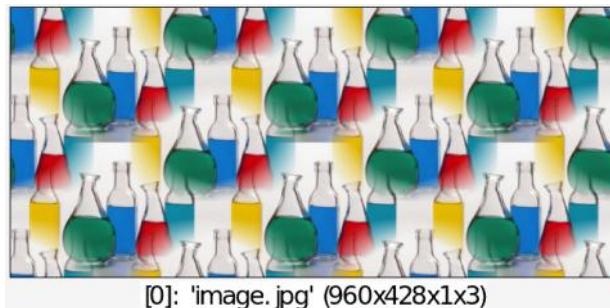
Create MxN array from selected images.

## Default values:

`N=M`, `fade_start=60`, `fade_end=90` and `expand_type=1`.

## Example of use:

```
$ gmic image.jpg array_fade 3,2
```



---

## array\_mirror

### Arguments:

- `N>=0`, `_dir={ 0=x | 1=y | 2=xy | 3=tri-xy }`, `_expand_type={ 0 | 1 }`

## Description:

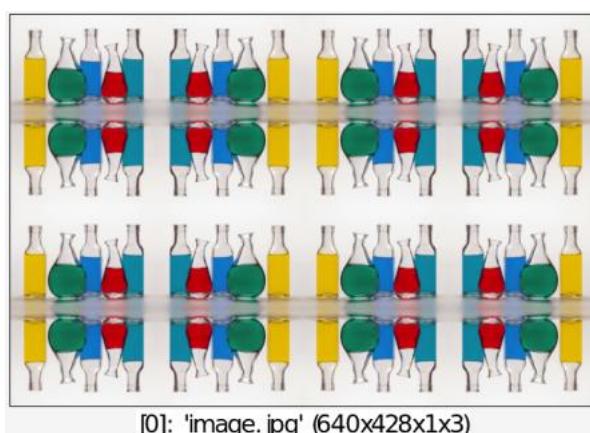
Create  $2^N \times 2^N$  array from selected images.

## Default values:

`dir=2` and `expand_type=0`.

## Example of use:

```
$ gmic image.jpg array_mirror 2
```



---

# array\_random

## Arguments:

- `Ms>0, _Ns>0, _Md>0, _Nd>0`

## Description:

Create MdxNd array of tiles from selected MsxNs source arrays.

## Default values:

`Ns=Ms`, `Md=Mn` and `Nd=Ns`.

## Example of use:

```
$ gmic image.jpg +array_random 8,8,15,10
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (1200x540x1x3)

---

# arrow

## Arguments:

- `x0[%],y0[%],x1[%],y1[%],_thickness[%]>=0,_head_length[%]>=0,_head_thickness[%]`

## Description:

Draw specified arrow on selected images.

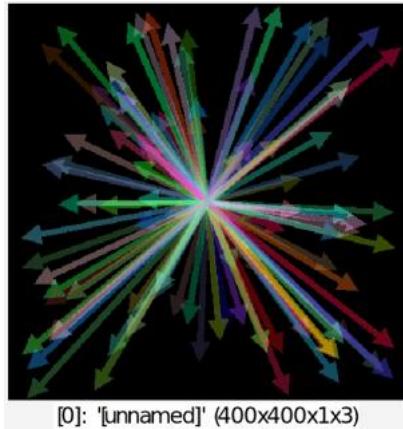
`pattern` is an hexadecimal number starting with `0x` which can be omitted even if a color is specified. If a pattern is specified, the arrow is drawn outlined instead of filled.

## Default values:

`thickness=1%`, `head_length=10%`, `head_thickness=3%`, `opacity=1`, `pattern=(undefined)` and `color1=0`.

## Example of use:

```
$ gmic 400,400,1,3 repeat 100 arrow 50%,50%,{u(100)}%,
{u(100)}%,3,20,10,0.3,${-rgb} done
```



---

## arrow3d

### Arguments:

- `x0,y0,z0,x1,y1,z1,_radius[%]>=0,_head_length[%]>=0,_head_radius[%]>=0`

### Description:

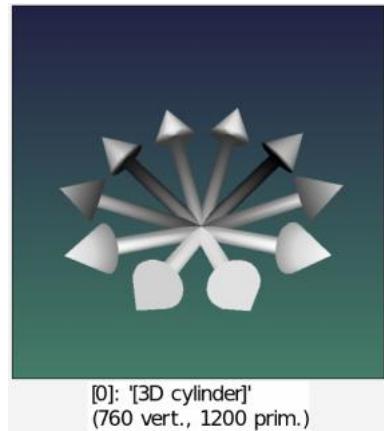
Input 3D arrow with specified starting and ending 3D points.

### Default values:

`radius=5%`, `head_length=25%` and `head_radius=15%`.

## Example of use:

```
$ gmic repeat 10 a={$>*2*pi/10} arrow3d 0,0,0,{cos($a)},
{sin($a)},-0.5 done +3d
```



# asin

Built-in command

No arguments

## Description:

Compute the pointwise arcsine of selected images.

This command has a [tutorial page](#).

## Examples of use:

- **Example #1**

```
$ gmic image.jpg +normalize -1,1 asin[-1]
```



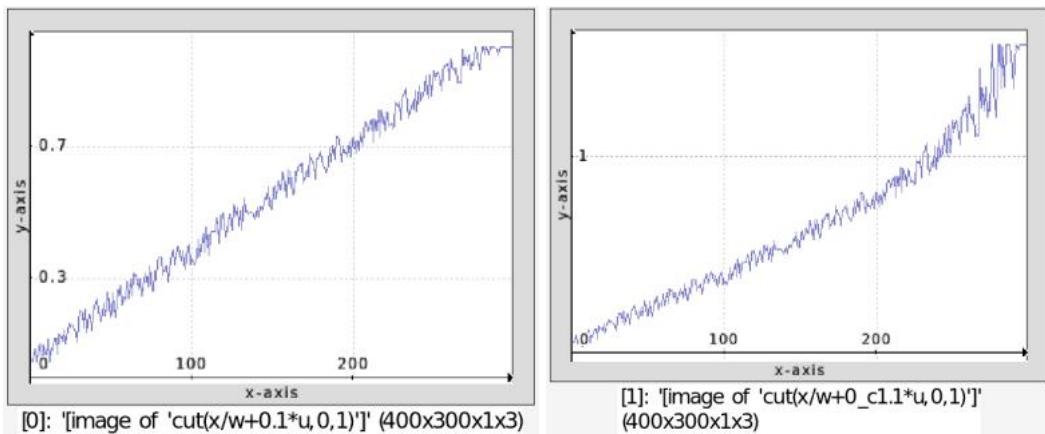
[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

- **Example #2**

```
$ gmic 300,1,1,1,'cut(x/w+0.1*u,0,1)' +asin display_graph 400,300
```



## asinh

**Built-in command**

**No arguments**

**Description:**

Compute the pointwise hyperbolic arcsine of selected images.

## at\_line

**Arguments:**

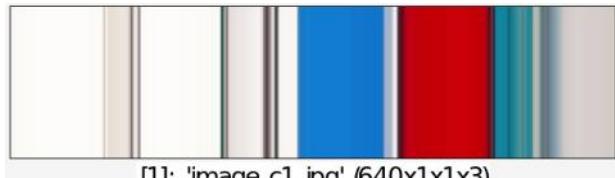
- `x0[%],y0[%],z0[%],x1[%],y1[%],z1[%]`

**Description:**

Retrieve pixels of the selected images belonging to the specified line  $(x_0,y_0,z_0)-(x_1,y_1,z_1)$ .

**Example of use:**

```
$ gmic image.jpg +at_line 0,0,0,100%,100%,0 line[0]
0,0,100%,100%,1,0xFF00FF00,255,0,0
```



---

# at\_quadrangle

## Arguments:

- `x0[%],y0[%],x1[%],y1[%],x2[%],y2[%],x3[%],y3[%],_interpolation,_boundary_conditions`  
or
- `x0[%],y0[%],z0[%],x1[%],y1[%],z1[%],x2[%],y2[%],z2[%],x3[%],y3[%],z3[%],_interpolation,_boundary_conditions`

## Description:

Retrieve pixels of the selected images belonging to the specified 2D or 3D quadrangle.

`interpolation` can be `{ 0=nearest-neighbor | 1=linear | 2=cubic }`.  
`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

## Example of use:

```
$ gmic image.jpg params=5%,5%,95%,5%,60%,95%,40%,95% +at_quadrangle  
$params polygon.. 4,$params,0.5,255
```



---

# atan

Built-in command

## No arguments

## Description:

Compute the pointwise arctangent of selected images.

This command has a [tutorial page](#).

## Examples of use:

- Example #1

```
$ gmic image.jpg +normalize 0,8 atan[-1]
```



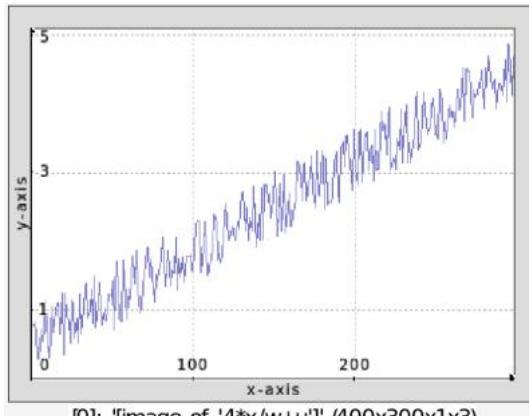
[0]: 'image.jpg' (640x427x1x3)



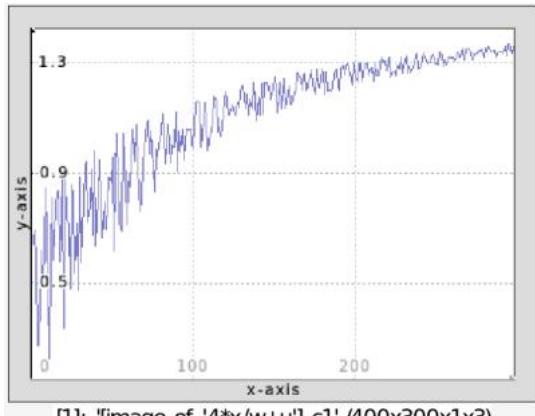
[1]: 'image\_c1.jpg' (640x427x1x3)

## • Example #2

```
$ gmic 300,1,1,1,'4*x/w+u' +atan display_graph 400,300
```



[0]: '[image of "4\*x/w+u"]' (400x300x1x3)



[1]: '[image of "4\*x/w+u"]\_c1' (400x300x1x3)

# atan2

Built-in command

## Arguments:

- `[x_argument]`

## Description:

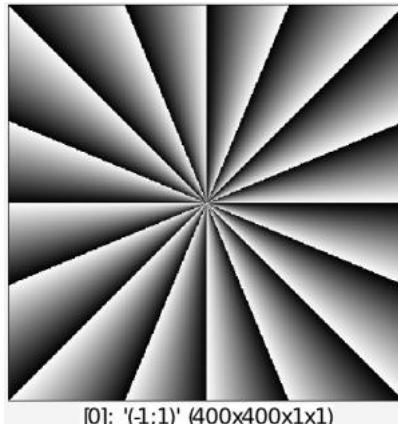
Compute the pointwise oriented arctangent of selected images.

Each selected image is regarded as the y-argument of the arctangent function, while the specified image gives the corresponding x-argument.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic (-1,1) (-1;1) resize 400,400,1,1,3 atan2[1] [0] keep[1] mod {pi/8}
```



---

## atanh

Built-in command

**No arguments**

**Description:**

Compute the pointwise hyperbolic arctangent of selected images.

---

## autocrop

Built-in command

**Arguments:**

- `value1,value2,...` or
- `(no arg)`

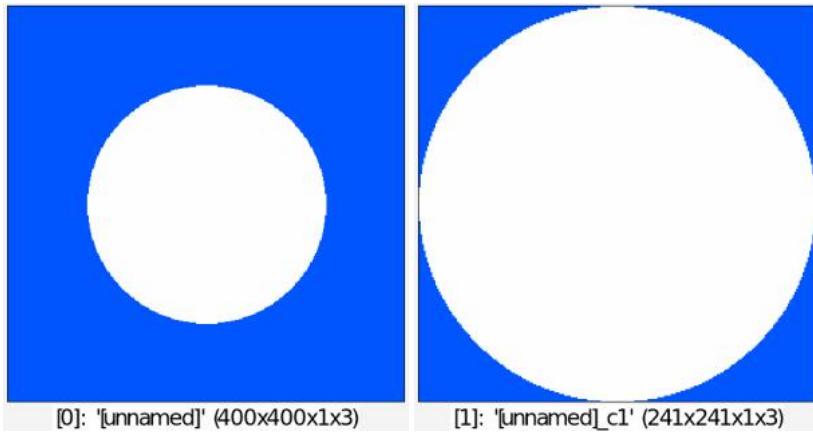
**Description:**

Autocrop selected images by specified vector-valued intensity.

If no arguments are provided, cropping value is guessed.

**Example of use:**

```
$ gmic 400,400,1,3 fill_color 64,128,255 ellipse  
50%,50%,120,120,0,1,255 +autocrop
```



---

## autocrop\_components

### Arguments:

- `_threshold[%], _min_area[%]>=0, _is_high_connectivity={ 0 | 1 }, _output_type={ 0=crop | 1=segmentation | 2=coordinates }`

### Description:

Autocrop and extract connected components in selected images, according to a mask given as the last channel of

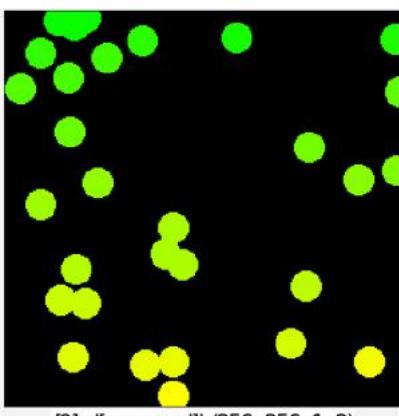
each of the selected image (e.g. alpha-channel).

### Default values:

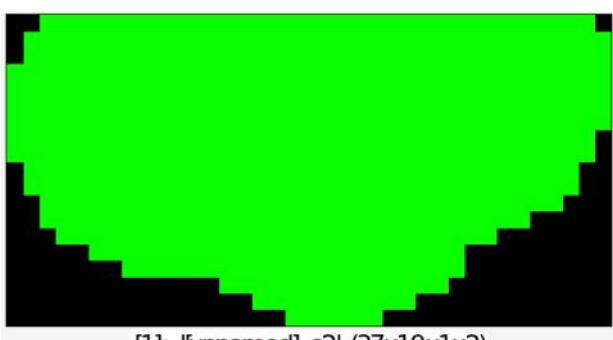
`threshold=0%`, `min_area=0.1%`, `is_high_connectivity=0` and `output_type=1`.

### Example of use:

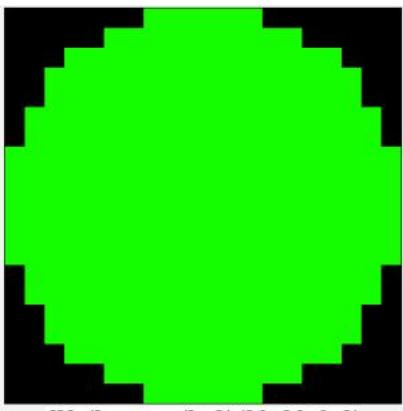
```
$ gmic 256,256 noise 0.1,2 eq 1 dilate_circ 20 label_fg 0,1 normalize  
0,255 +neq 0 *[-1] 255 append c +autocrop_components ,
```



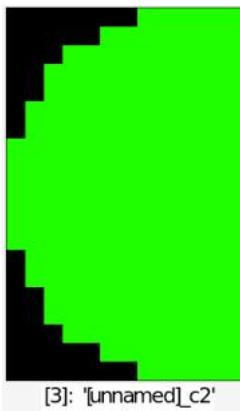
[0]: '[unnamed]' (256x256x1x2)



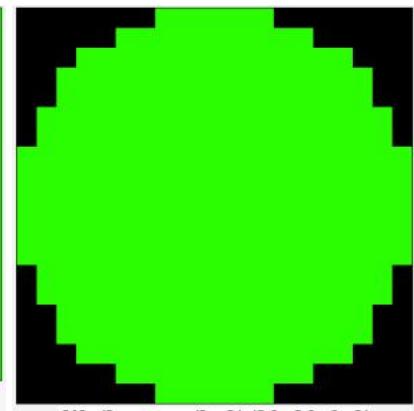
[1]: '[unnamed]\_c2' (37x19x1x2)



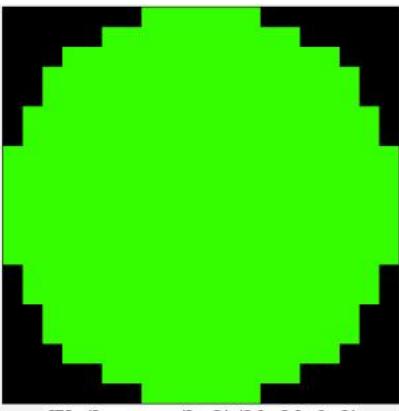
[2]: '[unnamed]\_c2' (20x20x1x2)



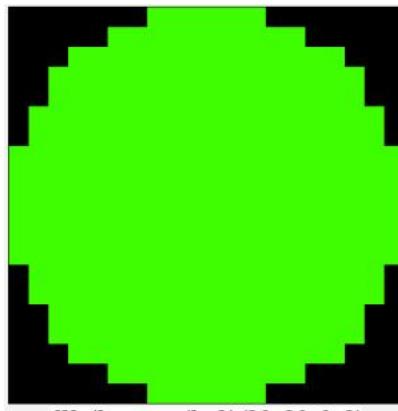
[3]: '[unnamed]\_c2' (13x20x1x2)



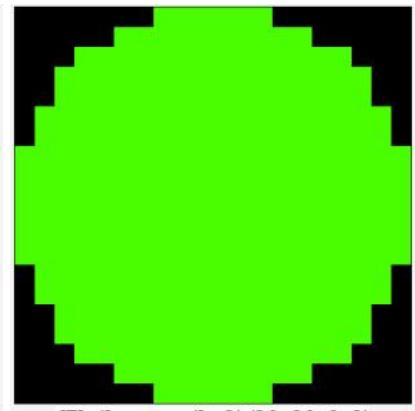
[4]: '[unnamed]\_c2' (20x20x1x2)



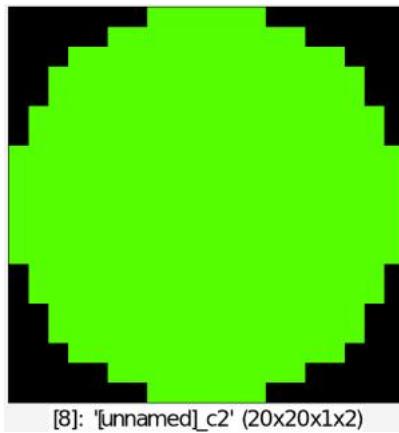
[5]: '[unnamed]\_c2' (20x20x1x2)



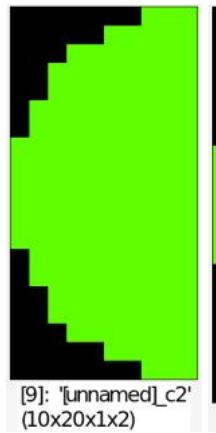
[6]: '[unnamed]\_c2' (20x20x1x2)



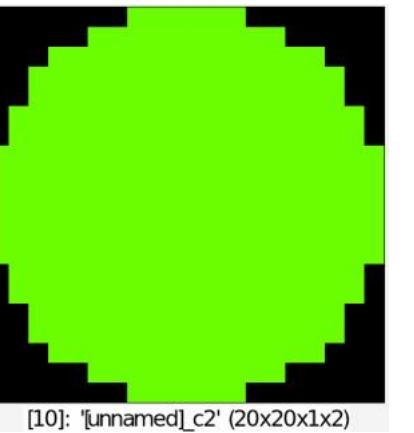
[7]: '[unnamed]\_c2' (20x20x1x2)



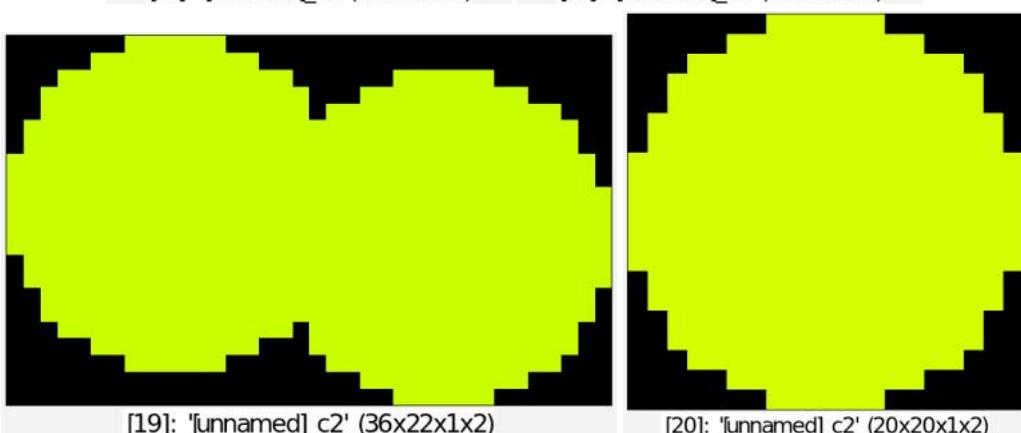
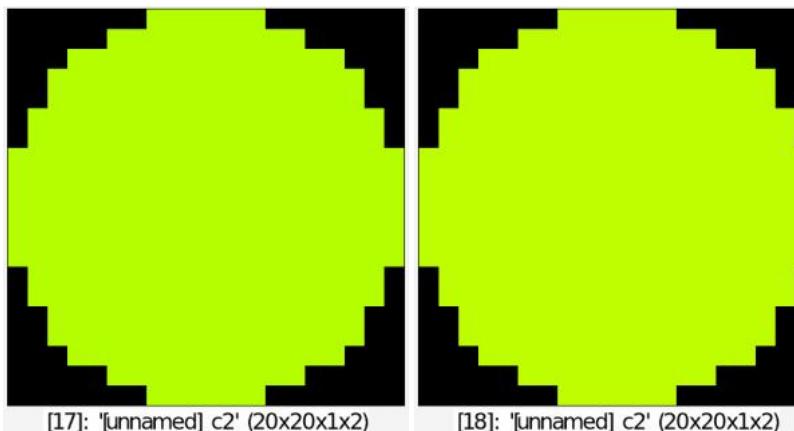
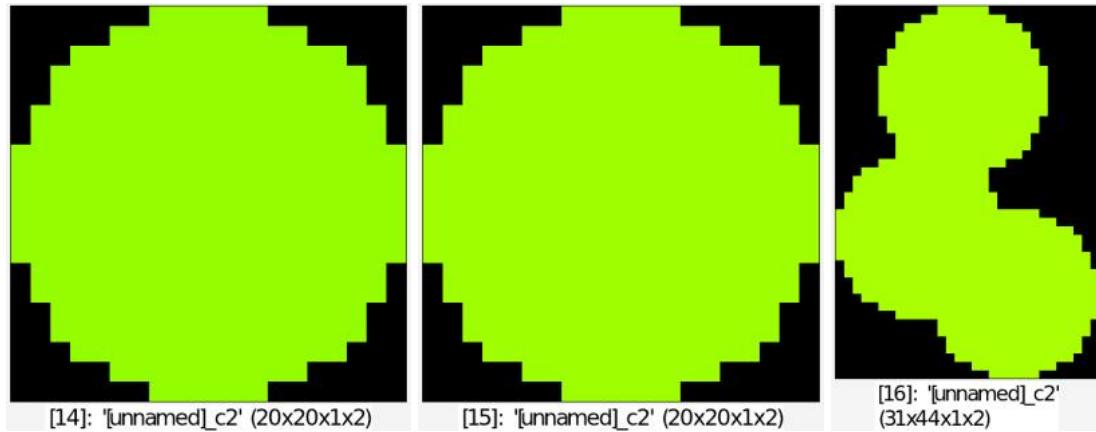
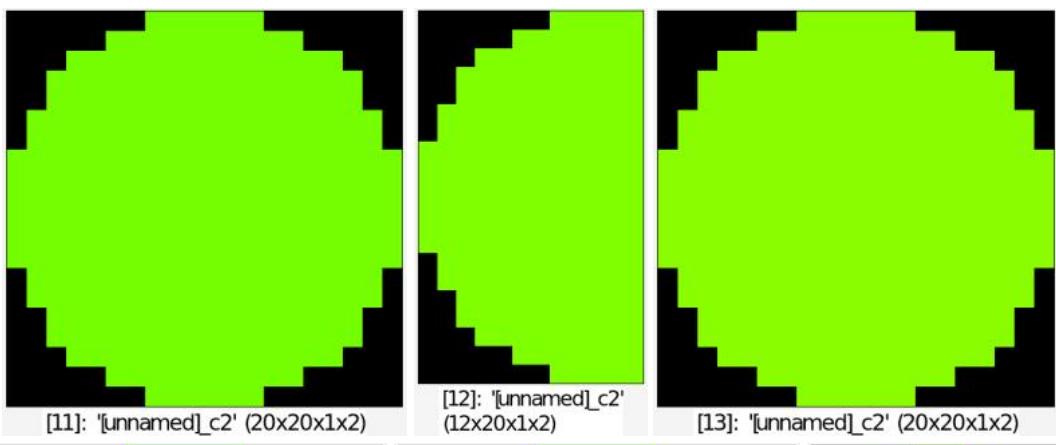
[8]: '[unnamed]\_c2' (20x20x1x2)

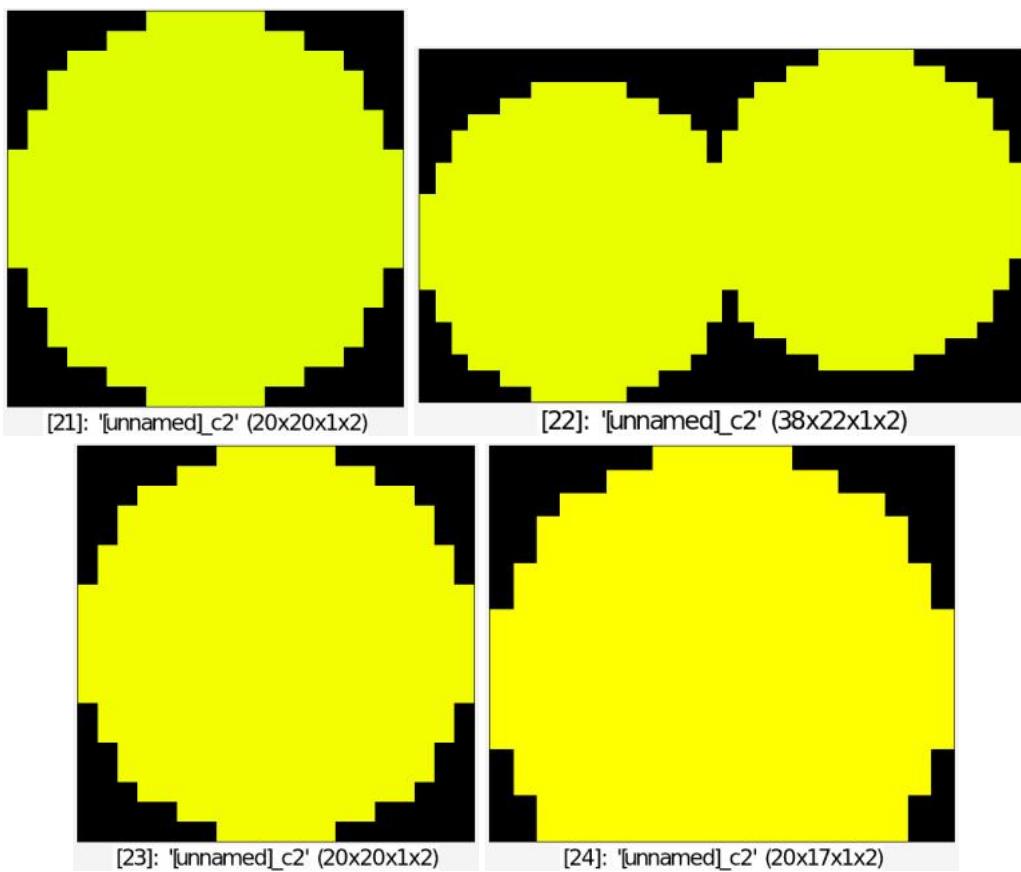


[9]: '[unnamed]\_c2' (10x20x1x2)



[10]: '[unnamed]\_c2' (20x20x1x2)





---

## autocrop\_coords

### Arguments:

- `value1,value2,... | auto`

### Description:

Return coordinates ( $x_0, y_0, z_0, x_1, y_1, z_1$ ) of the autocrop that could be performed on the latest of the selected images.

### Default values:

`auto`

---

## autocrop\_seq

### Arguments:

- `value1,value2,... | auto`

### Description:

Autocrop selected images using the crop geometry of the last one by specified vector-valued intensity,

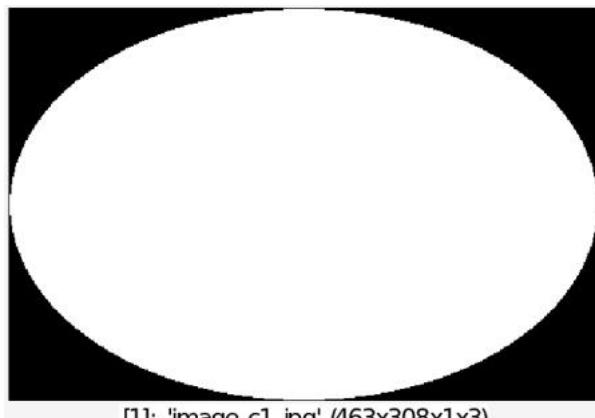
or by automatic guessing the cropping value.

## Default values:

auto mode.

## Example of use:

```
$ gmic image.jpg +fill[-1] 0 ellipse[-1] 50%,50%,30%,20%,0,1,1  
autocrop_seq 0
```



---

## autoindex

### Arguments:

- `nb_colors>0, 0<=_dithering<=1, _method={ 0=median-cut | 1=k-means }`

### Description:

Index selected vector-valued images by adapted colormaps.

## Default values:

`dithering=0` and `method=1`.

## Example of use:

```
$ gmic image.jpg +autoindex[0] 4 +autoindex[0] 8 +autoindex[0] 16
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)



[3]: 'image\_c1.jpg' (640x427x1x3)

---

## average\_colors

**No arguments**

**Description:**

Return the average vector-value of the latest of the selected images.

---

## average\_files

**Arguments:**

- `"filename_pattern", _first_frame>=0, _last_frame={ >=0 | -1=last }, _frame_step>=1, _output_filename`

**Description:**

Average specified input image files, in a streamed way.

If a display window is opened, rendered frames are displayed in it during processing.  
The output filename may have extension `.avi` or `.mp4` (saved as a video), or any other usual image file extension (saved as a sequence of images).

**Default values:**

```
first_frame=0, last_frame=-1, frame_step=1 and output_filename=(undefined).
```

---

## average\_video

### Arguments:

- `video_filename, _first_frame>=0, _last_frame={ >=0 | -1=last }, _frame_step>=1, _output_filename`

### Description:

Average frames of specified input video file, in a streamed way.

If a display window is opened, rendered frames are displayed in it during processing.  
The output filename may have extension `.avi` or `.mp4` (saved as a video), or any other usual image file extension (saved as a sequence of images).  
This command requires features from the OpenCV library (not enabled in G'MIC by default).

### Default values:

```
first_frame=0, last_frame=-1, frame_step=1 and output_filename=(undefined).
```

---

## axes

### Arguments:

- `x0,x1,y0,y1,_font_height>=0,_opacity,_pattern,_color1,...`

### Description:

Draw xy-axes on selected images.

`pattern` is an hexadecimal number starting with `0x` which can be omitted even if a color is specified.

To draw only one x-axis at row Y, set both `y0` and `y1` to Y.

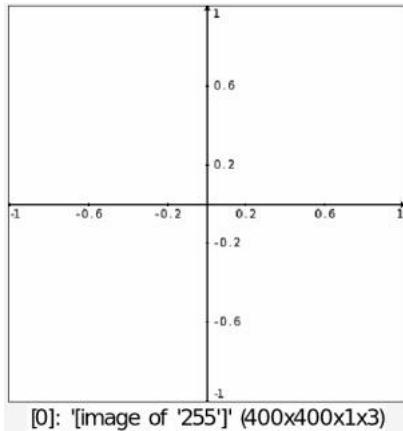
To draw only one y-axis at column X, set both `x0` and `x1` to X.

### Default values:

```
font_height=14, opacity=1, pattern=(undefined) and color1=0.
```

### Example of use:

```
$ gmic 400,400,1,3,255 axes -1,1,1,-1
```



## axes3d

### Arguments:

- `_size_x,_size_y,_size_z,_font_size>0,_label_x,_label_y,_label_z,_is_origin={ 0=no | 1=yes }`

### Description:

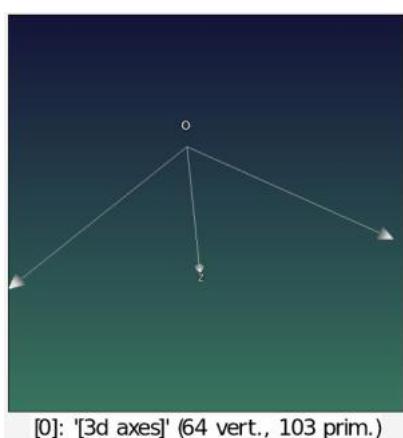
Input 3D axes with specified sizes along the x,y and z orientations.

### Default values:

`size_x=size_y=size_z=1`, `font_size=23`, `label_x=X`, `label_y=Y`, `label_z=Z` and `is_origin=1`

### Example of use:

```
$ gmic axes3d ,
```



## balance\_gamma

### Arguments:

- [\\_ref\\_color1](#),...

## Description:

Compute gamma-corrected color balance of selected image, with respect to specified reference color.

## Default values:

[ref\\_color1=128](#).

## Example of use:

```
$ gmic image.jpg +balance_gamma 128,64,64
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

## ball

### Arguments:

- [\\_size>0](#), [\\_R](#), [\\_G](#), [\\_B](#), [0<=\\_specular\\_light<=8](#), [0<=\\_specular\\_size<=8](#), [\\_shadow>=0](#)

## Description:

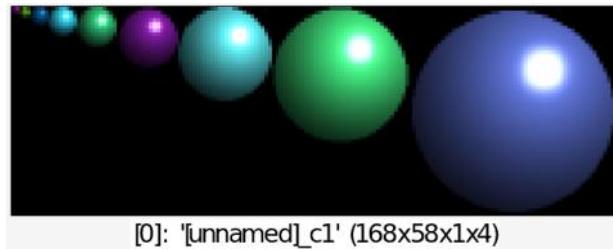
Input a 2D RGBA colored ball sprite.

## Default values:

[size=64](#), [R=255](#), [G=R](#), [B=R](#), [specular\\_light=0.8](#), [specular\\_size=1](#) and [shading=1.5](#).

## Example of use:

```
$ gmic repeat 9 ball {1.5^{($>+2)}}, ${-rgb} done append x
```



---

## bandpass

### Arguments:

- `_min_freq[%]`, `_max_freq[%]`

### Description:

Apply bandpass filter to selected images.

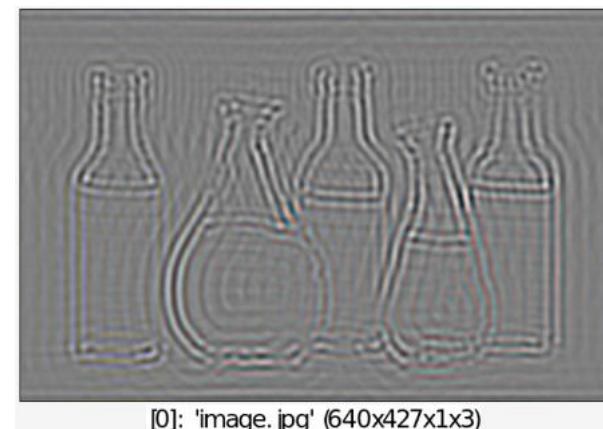
### Default values:

`min_freq=0` and `max_freq=20%`.

This command has a [tutorial page](#).

### Example of use:

```
$ gmic image.jpg bandpass 1%,3%
```



---

## barycenter

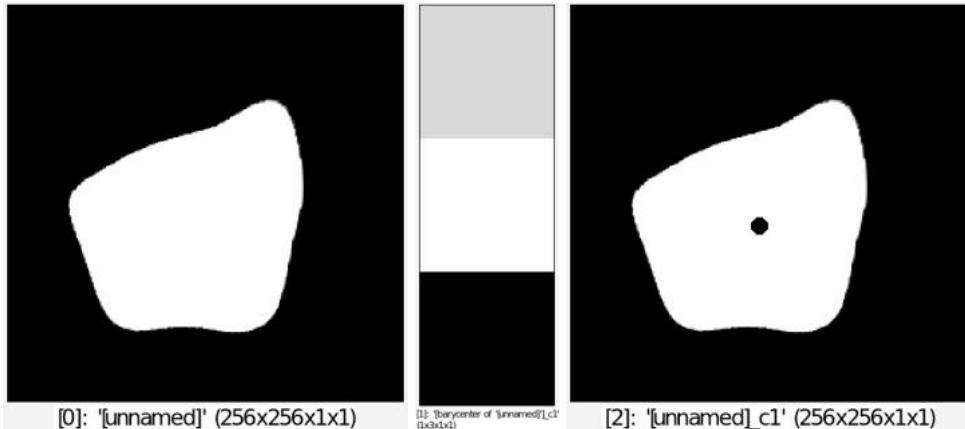
### No arguments

### Description:

Compute the barycenter vector of pixel values.

## Example of use:

```
$ gmic 256,256 ellipse 50%,50%,20%,20%,0,1,1 deform 20 +barycenter +ellipse[-2] {@0,1},5,5,0,10
```



---

## base642img

### Arguments:

- "base64\_string"

### Description:

Decode given base64-encoded string as a newly inserted image at the end of the list.

The argument string must have been generated using command [img2base64](#).

---

## base642uchar

### Arguments:

- "base64\_string"

### Description:

Decode given base64-encoded string as a newly inserted 1-column image at the end of the list.

The argument string must have been generated using command [uchar2base64](#).

---

## basename

### Arguments:

- `file_path,_variable_name_for_folder`

## Description:

Return the basename of a file path, and opt. its folder location.

When specified `variable_name_for_folder` must starts by an underscore (global variable accessible from calling function).

---

# bayer2rgb

## Arguments:

- `_GM_smoothness,_RB_smoothness1,_RB_smoothness2`

## Description:

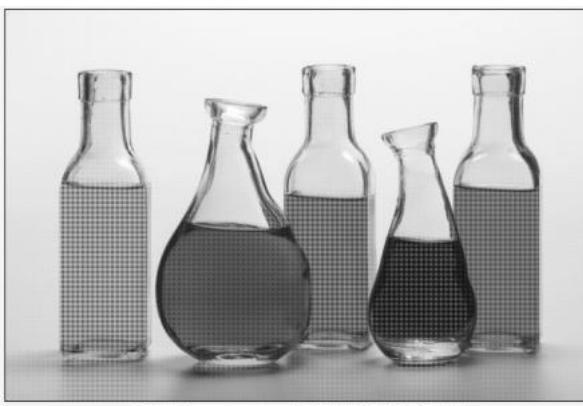
Transform selected RGB-Bayer sampled images to color images.

## Default values:

`GM_smoothness=RB_smoothness=1` and `RB_smoothness2=0.5`.

## Example of use:

```
$ gmic image.jpg rgb2bayer 0 +bayer2rgb 1,1,0.5
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x3)

# bilateral

Built-in command

## Arguments:

- `[guide],std_deviation_s[%]>=0,std_deviation_r[%]>=0,_sampling_s>=0,_sampling_r>=0`  
or
- `std_deviation_s[%]>=0,std_deviation_r[%]>=0,_sampling_s>=0,_sampling_r>=0`

## Description:

Blur selected images by anisotropic (eventually joint/cross) bilateral filtering.

If a guide image is provided, it is used for drive the smoothing filter.  
A guide image must be of the same xyz-size as the selected images.  
Set **sampling** arguments to **0** for automatic adjustment.

## Example of use:

```
$ gmic image.jpg repeat 5 bilateral 10,10 done
```



---

## bin

### Arguments:

- **binary\_int1,...**

## Description:

Print specified binary integers into their octal, decimal, hexadecimal and string representations.

---

## bin2dec

### Arguments:

- **binary\_int1,...**

## Description:

Convert specified binary integers into their decimal representations.

---

## blend

## Arguments:

- `[layer],blending_mode,_opacity[%],_selection_is={ 0=base-layers | 1=top-layers }` or
- `blending_mode,_opacity[%]`

## Description:

Blend selected G,GA,RGB or RGBA images by specified layer or blend all selected images together, using specified blending mode.

`blending_mode` can be { add | alpha | and | average | blue | burn | darken | difference | divide | dodge | edges | exclusion | freeze | grainextract | grainmerge | green | hardlight | hardmix | hue | interpolation | lchlightness | lighten | lightness | linearburn | linearlight | luminance | multiply | negation | or | overlay | pinlight | red | reflect | saturation | seamless | seamless\_mixed | screen | shapeareamax | shapeareamax0 | shapeareamin | shapeareamin0 | shapeaverage | shapeaverage0 | shapemedian | shapemedian0 | shapemin | shapemin0 | shapemax | shapemax0 | softburn | softdodge | softlight | stamp | subtract | value | vividlight | xor }.

`opacity` should be in `[0,1]`, or `[0,100]` if expressed with a `%`.

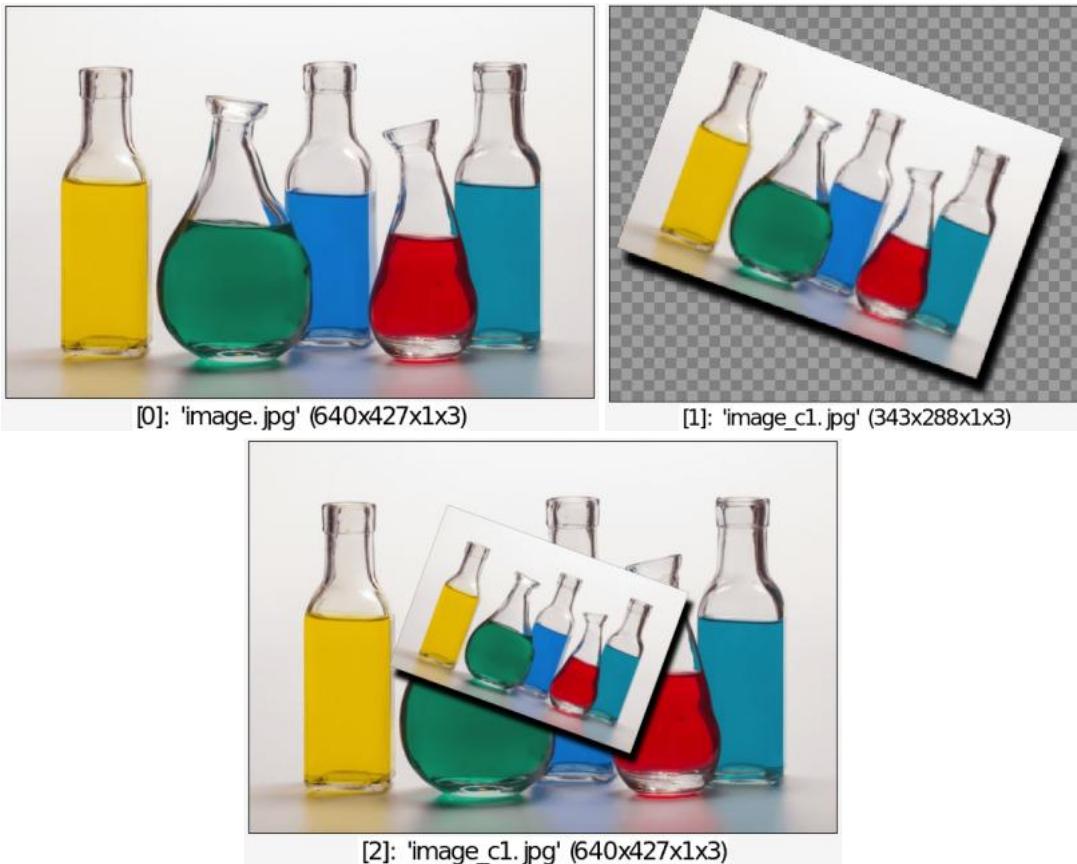
## Default values:

`blending_mode=alpha`, `opacity=1` and `selection_is=0`.

## Examples of use:

- **Example #1**

```
$ gmic image.jpg +drop_shadow , resize2dy[-1] 200 rotate[-1] 20  
+blend alpha display_rgba[-2]
```



- **Example #2**

```
$ gmic image.jpg testimage2d {w},{h} blend overlay
```

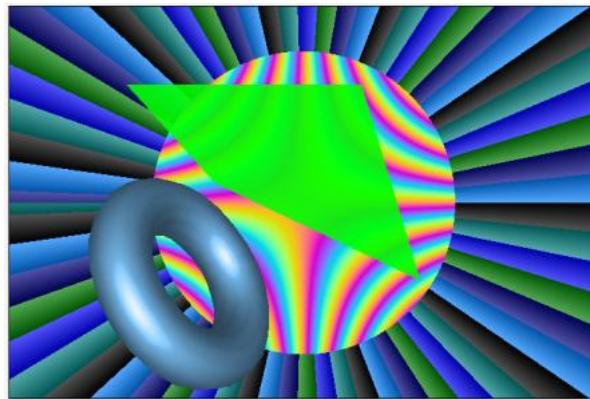


- **Example #3**

```
$ gmic command "ex : $""=arg repeat $""# +blend[0,1] ${arg{$>+1}}
text_outline[-1] Mode:\\" \${arg{$>+1}},2,2,23,2,1,255 done"
image.jpg testimage2d {w},{h} ex
add,alpha,blend,average,blue,burn,darken
```



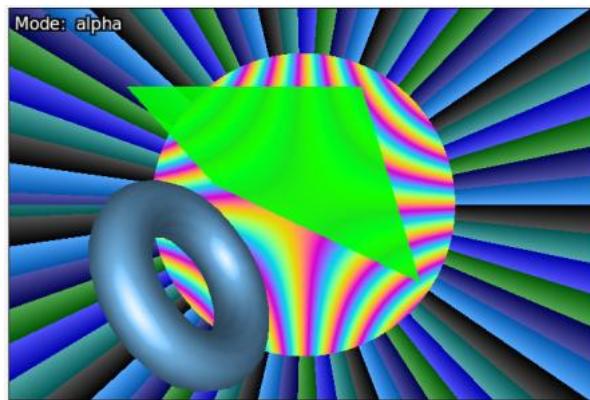
[0]: 'image.jpg' (640x427x1x3)



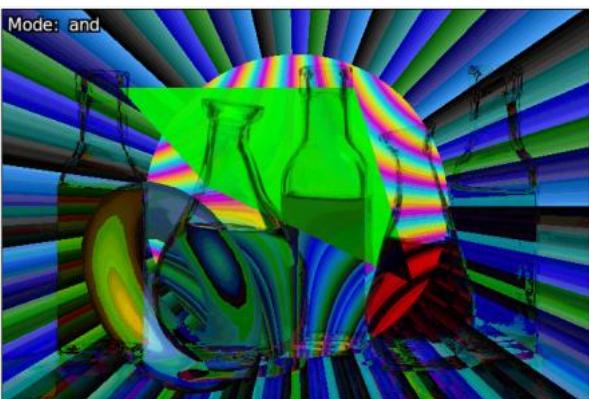
[1]: '[2D test image]' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)



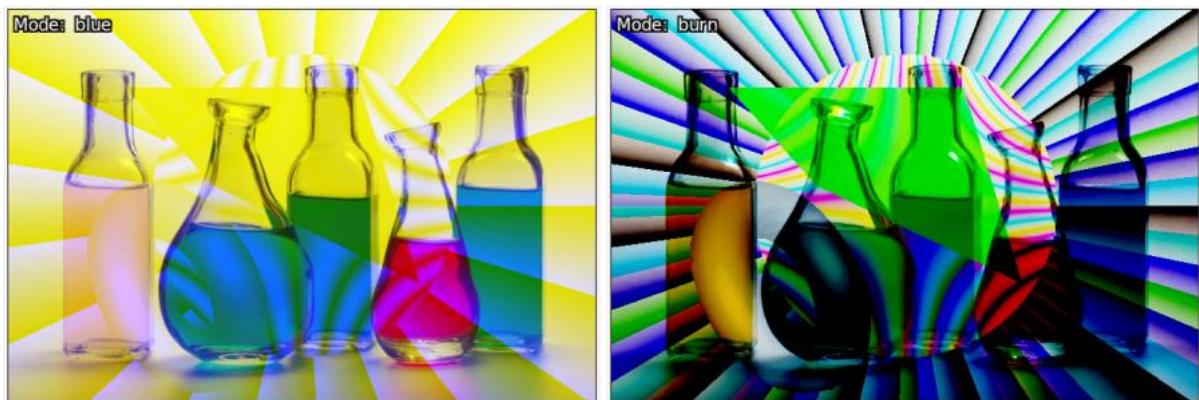
[3]: 'image\_c1.jpg' (640x427x1x3)



[4]: 'image\_c1.jpg' (640x427x1x3)



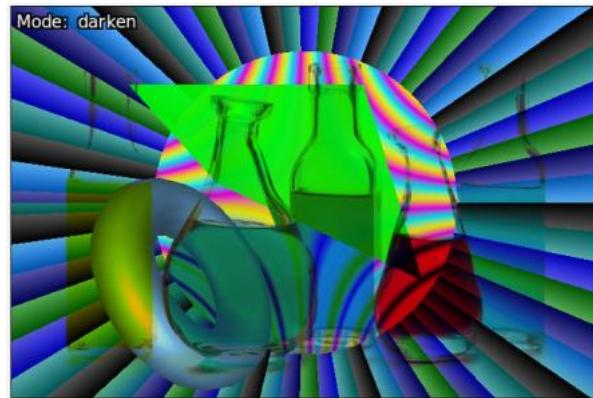
[5]: 'image\_c1.jpg' (640x427x1x3)



[6]: 'image\_c1.jpg' (640x427x1x3)



[7]: 'image\_c1.jpg' (640x427x1x3)



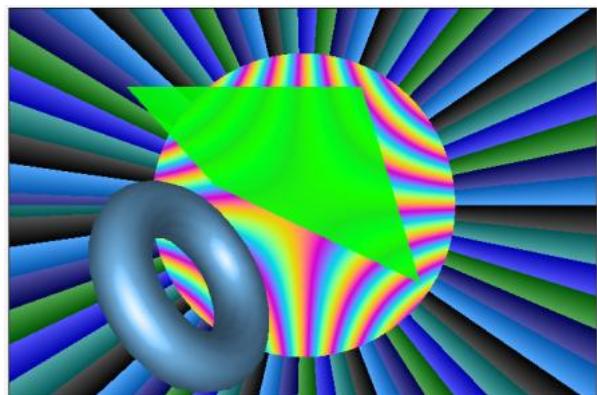
[8]: 'image\_c1.jpg' (640x427x1x3)

#### • Example #4

```
$ gmic command "ex : $""=arg repeat $""# +blend[0,1] ${arg{$>+1}}
text_outline[-1] Mode:\\" \"$${arg{$>+1}},2,2,23,2,1,255 done"
image.jpg testimage2d {w},{h} ex
difference,divide,dodge,exclusion,freeze,grainextract,grainmerge
```



[0]: 'image.jpg' (640x427x1x3)



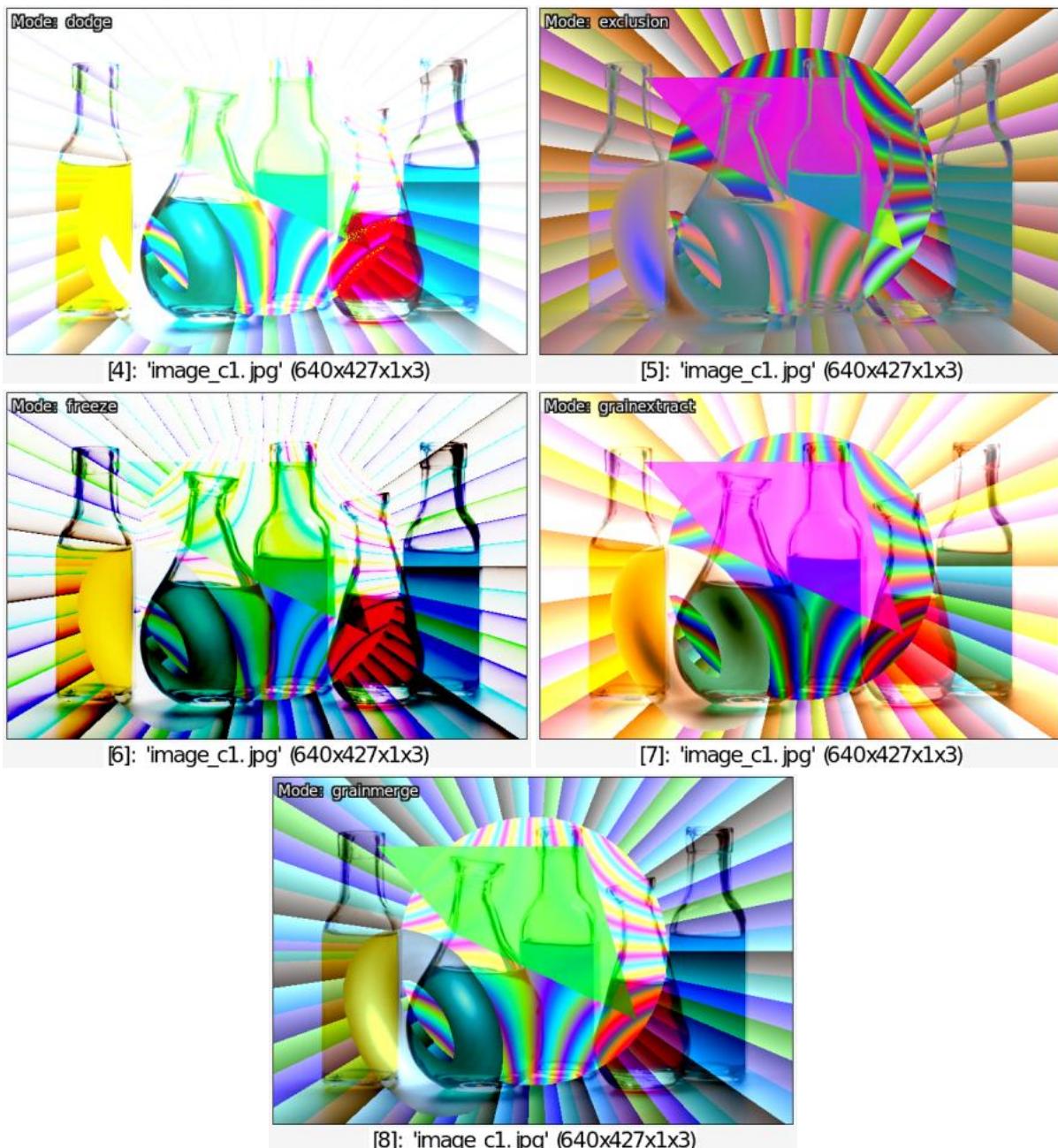
[1]: '[2D test image]' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)



[3]: 'image\_c1.jpg' (640x427x1x3)

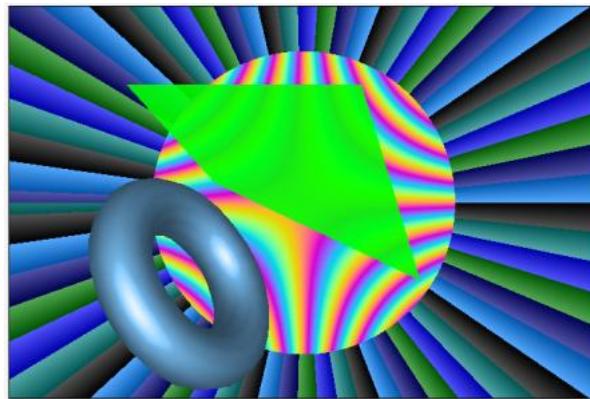


## • Example #5

```
$ gmic command "ex : $""=arg repeat $""# +blend[0,1] ${arg{$>+1}}
text_outline[-1] Mode:\\" \"${arg{$>+1}},2,2,23,2,1,255 done"
image.jpg testimage2d {w},{h} ex
green,hardlight,hardmix,hue,interpolation,lighten,lightness
```



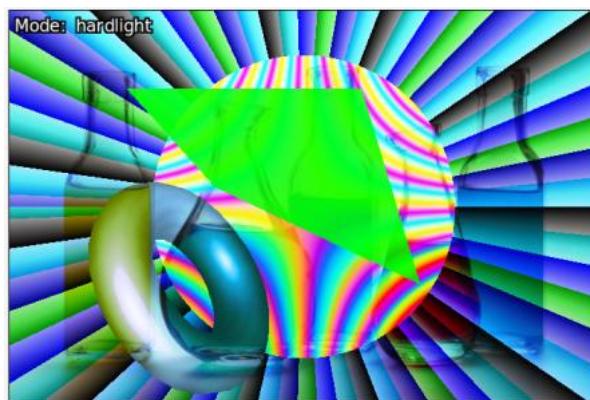
[0]: 'image.jpg' (640x427x1x3)



[1]: '[2D test image]' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)



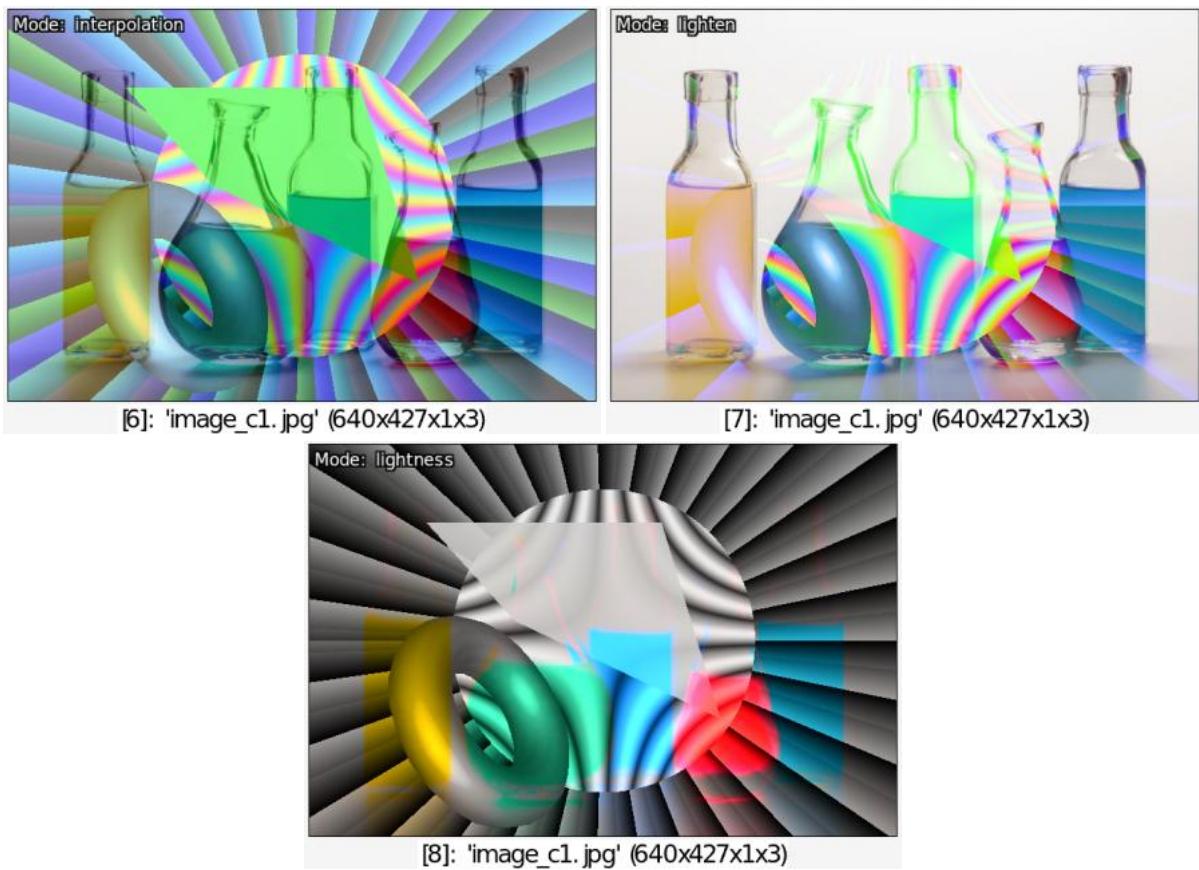
[3]: 'image\_c1.jpg' (640x427x1x3)



[4]: 'image\_c1.jpg' (640x427x1x3)

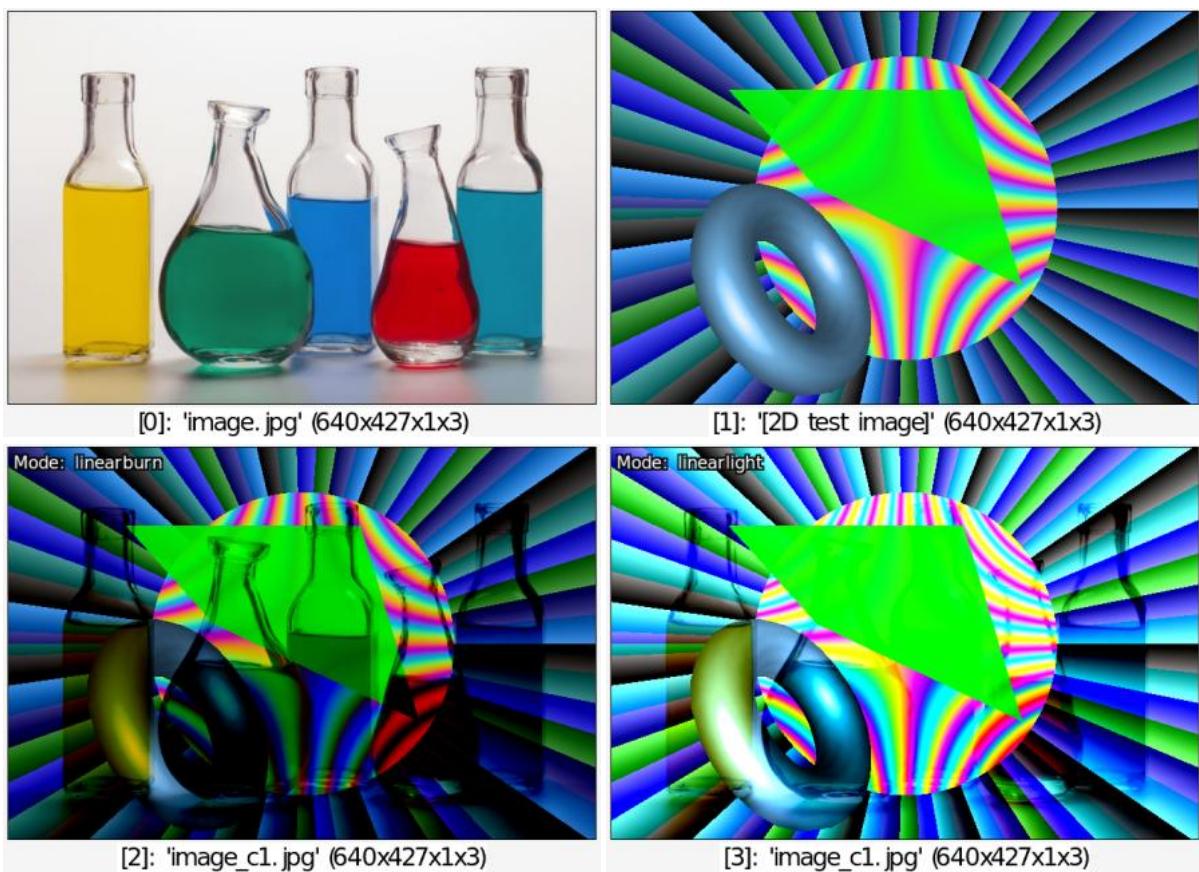


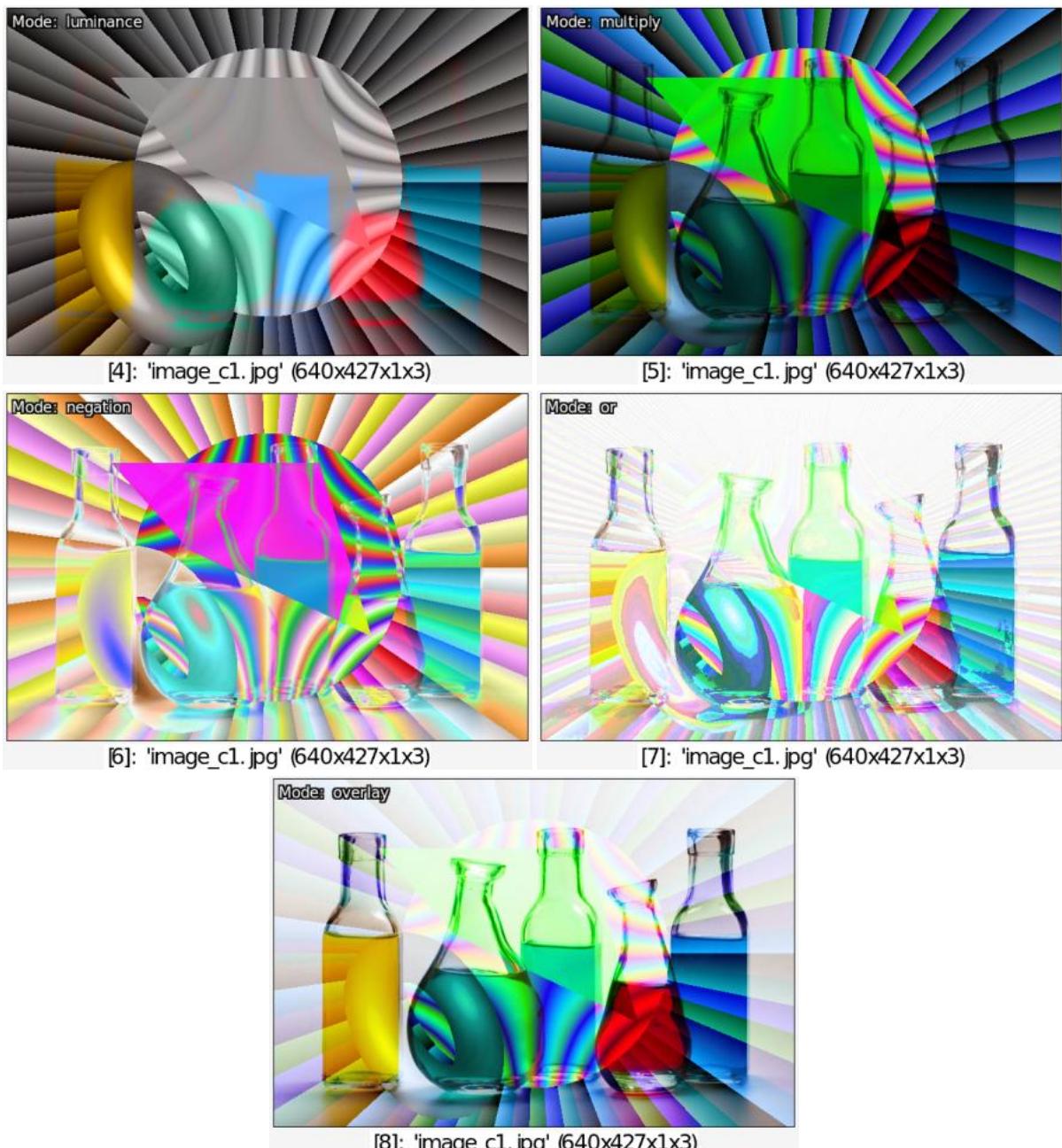
[5]: 'image\_c1.jpg' (640x427x1x3)



## • Example #6

```
$ gmic command "ex : $""=arg repeat $""# +blend[0,1] ${arg{$>+1}}
text_outline[-1] Mode:\\" \${arg{$>+1}},2,2,23,2,1,255 done"
image.jpg testimage2d {w},{h} ex
linearburn,linearlight,luminance,multiply,negation,or,overlay
```



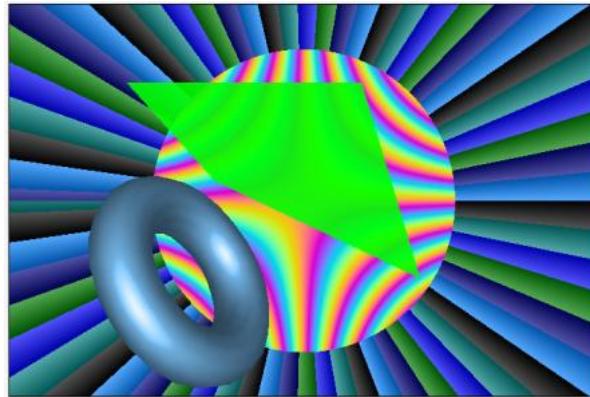


## • Example #7

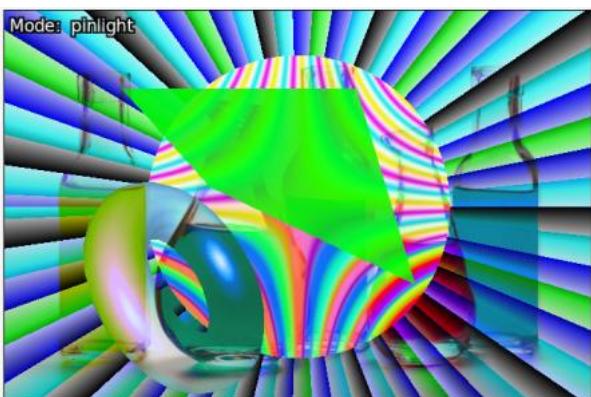
```
$ gmic command "ex : $""=arg repeat $""# +blend[0,1] ${arg{$>+1}}
text_outline[-1] Mode:\\" \"$${arg{$>+1}},2,2,23,2,1,255 done"
image.jpg testimage2d {w},{h} ex
pinlight,red,reflect,saturation,screen,shapeaverage,softburn
```



[0]: 'image.jpg' (640x427x1x3)



[1]: '[2D test image]' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)



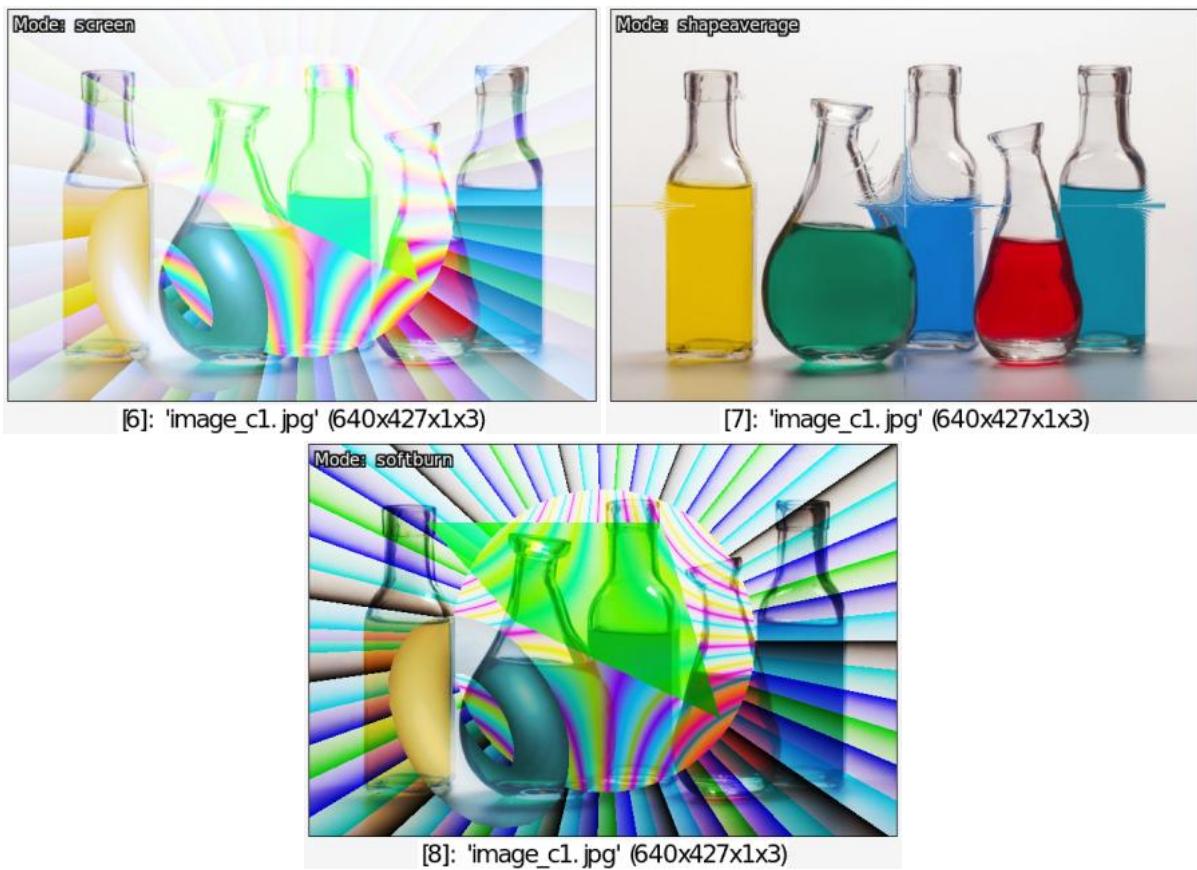
[3]: 'image\_c1.jpg' (640x427x1x3)



[4]: 'image\_c1.jpg' (640x427x1x3)

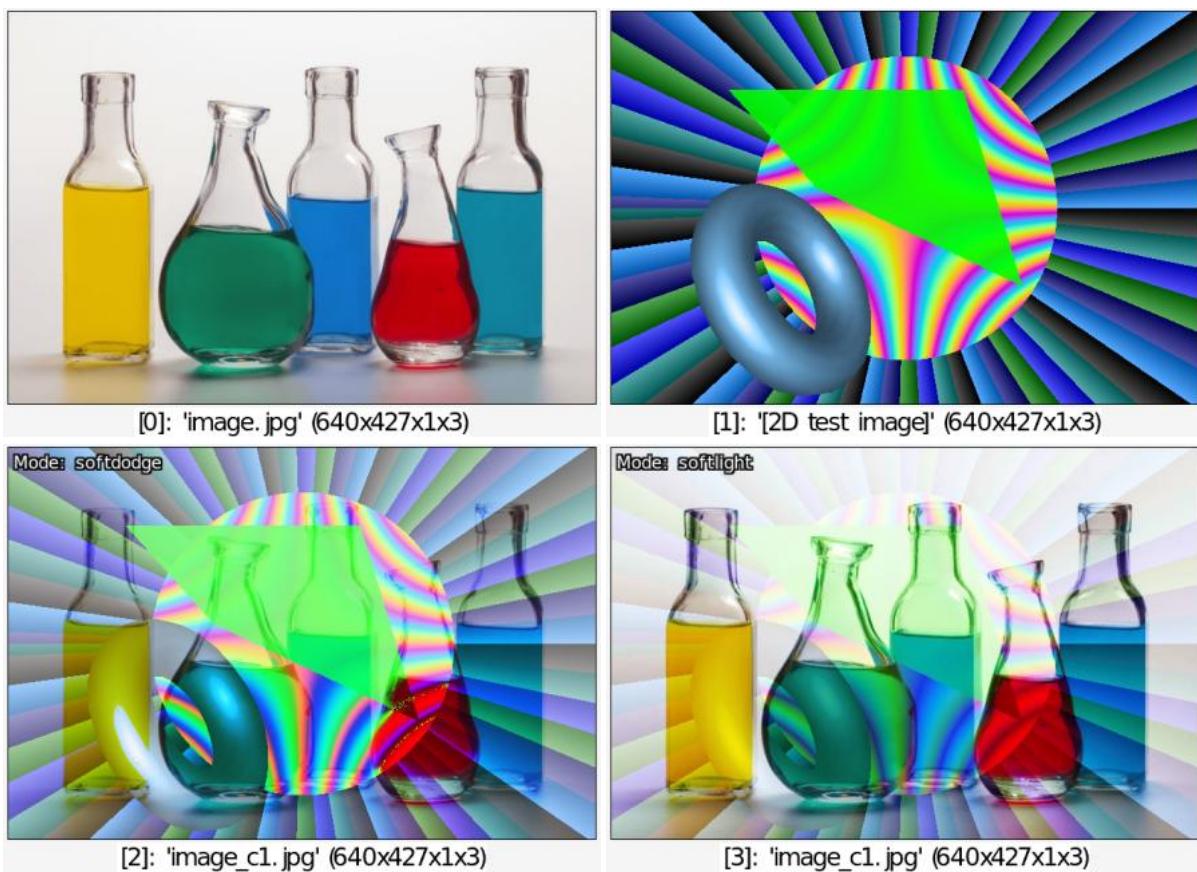


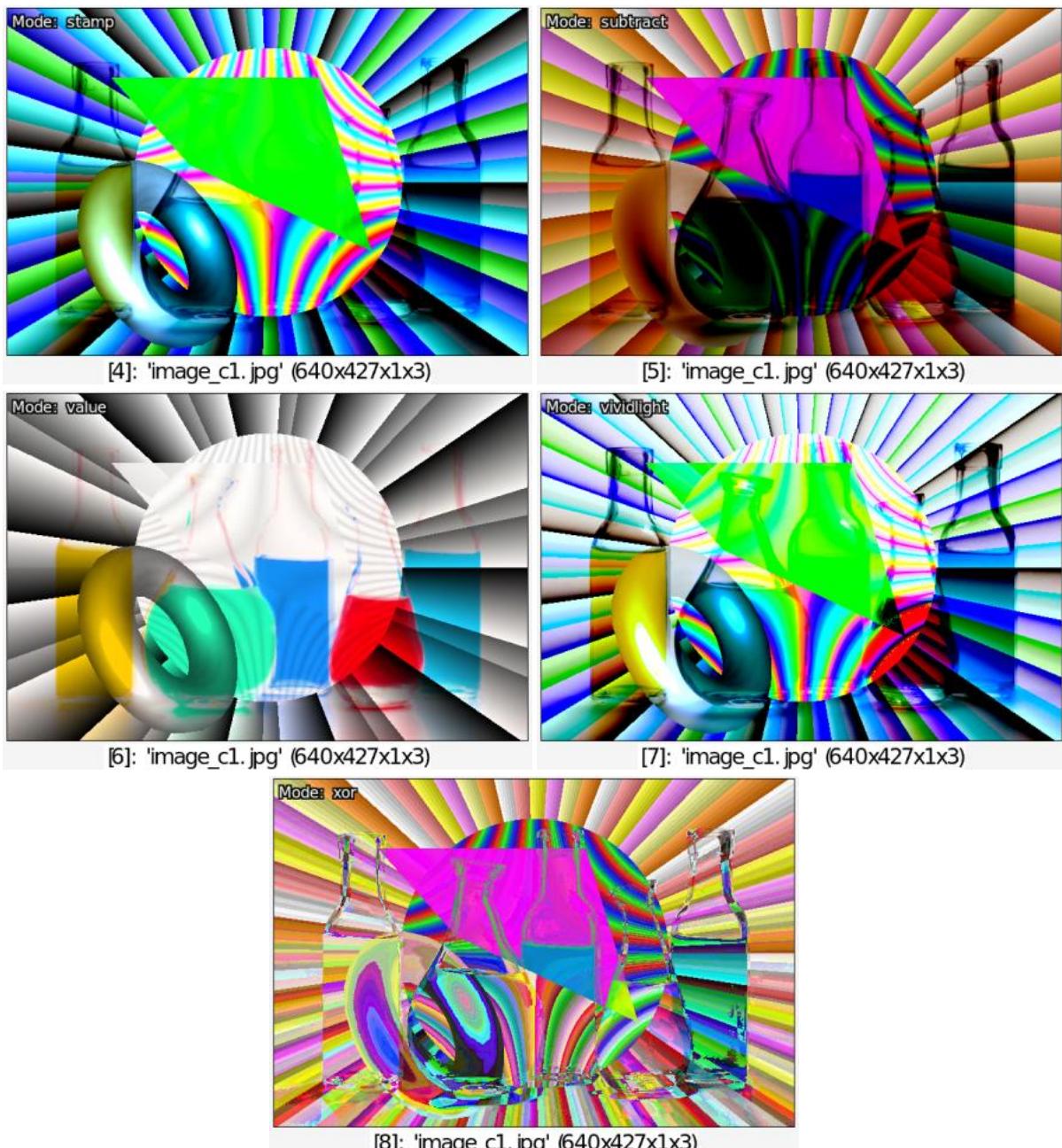
[5]: 'image\_c1.jpg' (640x427x1x3)



## • Example #8

```
$ gmic command "ex : $""=arg repeat $""# +blend[0,1] ${arg{$>+1}}
text_outline[-1] Mode:\\" \"${arg{$>+1}},2,2,23,2,1,255 done"
image.jpg testimage2d {w},{h} ex
softdodge,softlight,stamp,subtract,value,vividlight,xor
```





## blend\_edges

### Arguments:

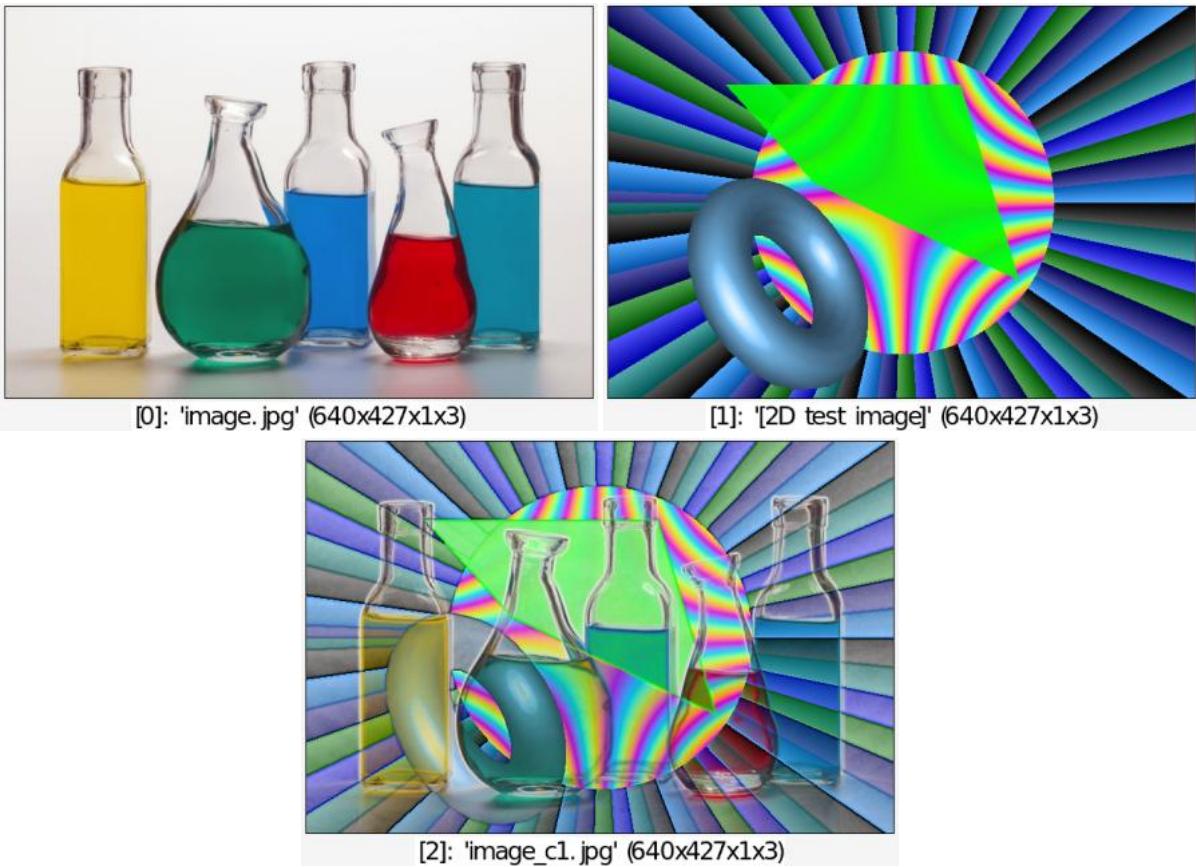
- smoothness [%]  $\geq 0$

### Description:

Blend selected images together using **edges** mode.

### Example of use:

```
$ gmic image.jpg testimage2d {w},{h} +blend_edges 0.8
```



---

## blend\_fade

### Arguments:

- `[fading_shape]`

### Description:

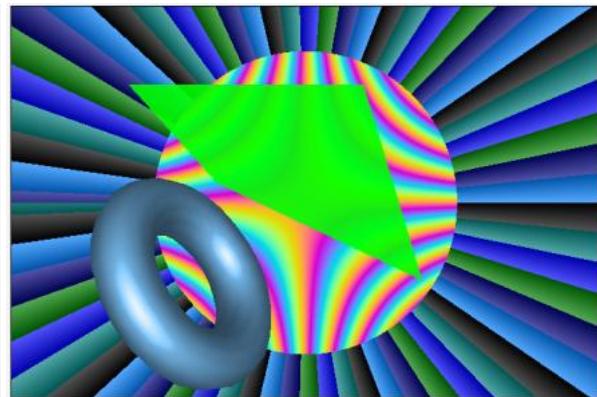
Blend selected images together using specified fading shape.

### Example of use:

```
$ gmic image.jpg testimage2d {w},{h} 100%,100%,1,1,'cos(y/10)'  
normalize[-1] 0,1 +blend_fade[0,1] [2]
```



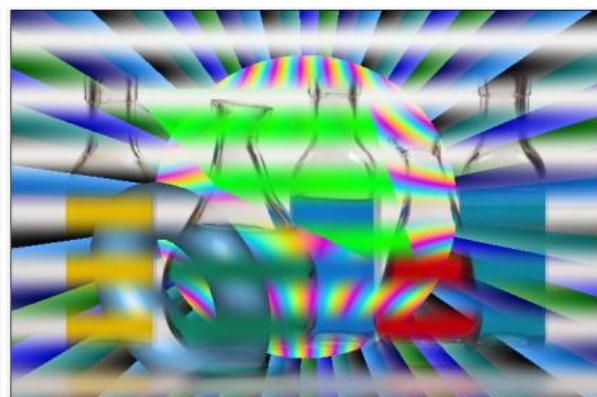
[0]: 'image.jpg' (640x427x1x3)



[1]: '[2D test image]' (640x427x1x3)



[2]: '[image of 'cos(y/10)']' (640x427x1x1)



[3]: 'image\_c1.jpg' (640x427x1x3)

## blend\_median

**No arguments**

**Description:**

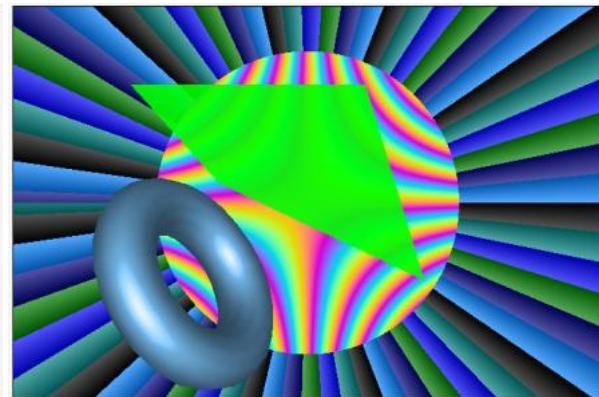
Blend selected images together using **median** mode.

**Example of use:**

```
$ gmic image.jpg testimage2d {w},{h} +mirror[0] y +blend_median
```



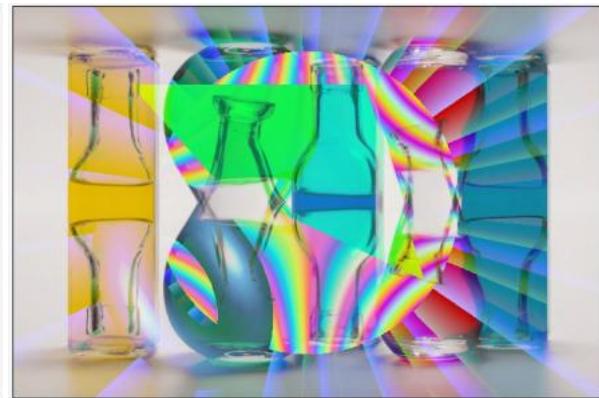
[0]: 'image.jpg' (640x427x1x3)



[1]: '[2D test image]' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)



[3]: '[image of 'med(#[#0])\']\_c1' (640x427x1x3)

## blend\_seamless

### Arguments:

- `_is_mixed_mode={ 0 | 1 },_inner_fading[%]>=0,_outer_fading[%]>=0`

### Description:

Blend selected images using a seamless blending mode (Poisson-based).

### Default values:

`is_mixed=0`, `inner_fading=0` and `outer_fading=100%`.

## blur

Built-in command

### Arguments:

- `std_deviation>=0[%],_boundary_conditions,_kernel` or
- `axes,std_deviation>=0[%],_boundary_conditions,_kernel`

### Description:

Blur selected images by a deriche or gaussian filter (recursive implementation).

(equivalent to shortcut command `b`).

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

`kernel` can be `{ 0=deriche | 1=gaussian }`.

When specified, argument `axes` is a sequence of `{ x | y | z | c }`.

Specifying one axis multiple times apply also the blur multiple times.

## Default values:

`boundary_conditions=1` and `kernel=1`.

This command has a [tutorial page](#).

## Examples of use:

- Example #1

```
$ gmic image.jpg +blur 5,0 +blur[0] 5,1
```



[0]: 'image.jpg' (640x427x1x3)



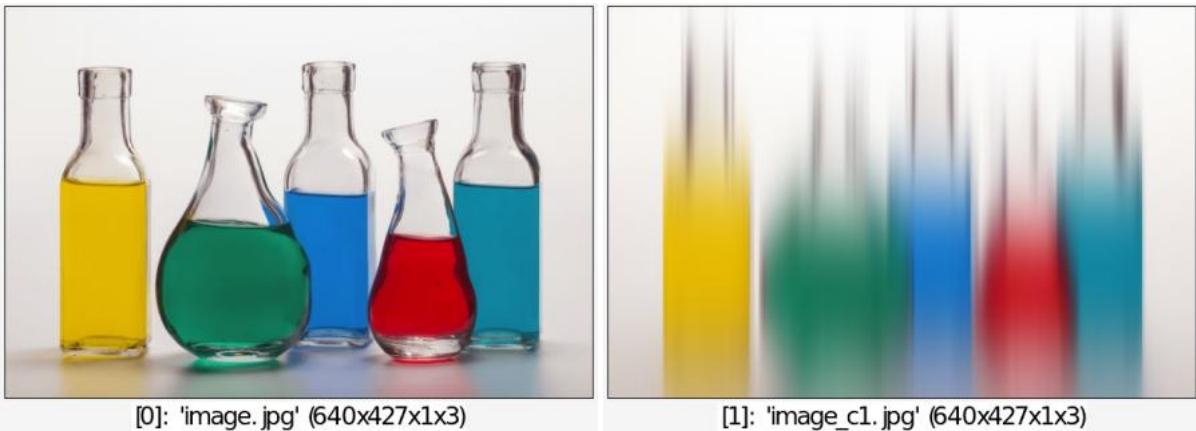
[1]: 'image\_c1.jpg' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)

- Example #2

```
$ gmic image.jpg +blur y,10%
```



---

## blur\_angular

### Arguments:

- `amplitude[%], _center_x[%], _center_y[%]`

### Description:

Apply angular blur on selected images.

### Default values:

`center_x=center_y=50%`.

This command has a [tutorial page](#).

### Example of use:

```
$ gmic image.jpg blur-angular 2%
```



---

## blur\_bloom

### Arguments:

- `_amplitude>=0, _ratio>=0, _nb_iter>=0, _blend_operator={ + | max | min }`, `_kernel={ 0=deriche | 1=gaussian | 2=box | 3=triangle | 4=quadratic }`, `_normalize_scales={ 0 | 1 } , _axes`

## Description:

Apply a bloom filter that blend multiple blur filters of different radii,

resulting in a larger but sharper glare than a simple blur.

When specified, argument `axes` is a sequence of `{ x | y | z | c }`.

Specifying one axis multiple times apply also the blur multiple times.

Reference: Masaki Kawase, "Practical Implementation of High Dynamic Range Rendering", GDC 2004.

## Default values:

```
amplitude=1, ratio=2, nb_iter=5, blend_operator=+, kernel=1,
normalize_scales=0 and axes=(all)
```

## Example of use:

```
$ gmic image.jpg blur_bloom ,
```



## blur\_linear

### Arguments:

- `amplitude1[%],_amplitude2[%],_angle,_boundary_conditions={ 0=dirichlet | 1=neumann }`

## Description:

Apply linear blur on selected images, with specified angle and amplitudes.

## Default values:

```
amplitude2=0, angle=0 and boundary_conditions=1.
```

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg blur_linear 10,0,45
```



---

## blur\_radial

### Arguments:

- `amplitude[%],_center_x[%],_center_y[%]`

### Description:

Apply radial blur on selected images.

### Default values:

`center_x=center_y=50%`.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg blur_radial 2%
```



---

# blur\_selective

## Arguments:

- `sigma>=0, _edges>0, _nb_scales>0`

## Description:

Blur selected images using selective gaussian scales.

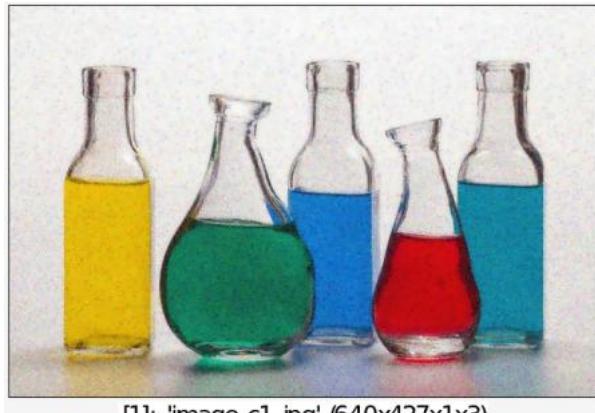
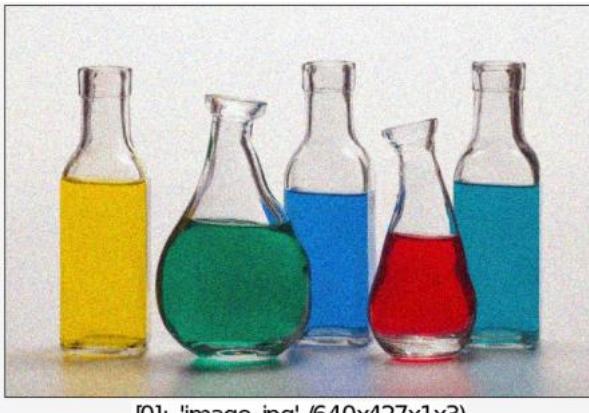
## Default values:

`sigma=5, edges=0.5` and `nb_scales=5`.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg noise 20 cut 0,255 +local[-1] repeat 4  
blur_selective , done endlocal
```



---

# blur\_x

## Arguments:

- `amplitude[%]>=0, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

## Description:

Blur selected images along the x-axis.

## Default values:

`boundary_conditions=1`.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg +blur_x 6
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## blur\_xy

### Arguments:

- `amplitude_x[%], amplitude_y[%], _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

### Description:

Blur selected images along the X and Y axes.

### Default values:

`boundary_conditions=1`.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg +blur_xy 6
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## blur\_xyz

### Arguments:

- `amplitude_x[%], amplitude_y[%], amplitude_z, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

### Description:

Blur selected images along the X, Y and Z axes.

### Default values:

`boundary_conditions=1`.

This command has a [tutorial page](#).

---

## blur\_y

### Arguments:

- `amplitude[%]>=0, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

### Description:

Blur selected images along the y-axis.

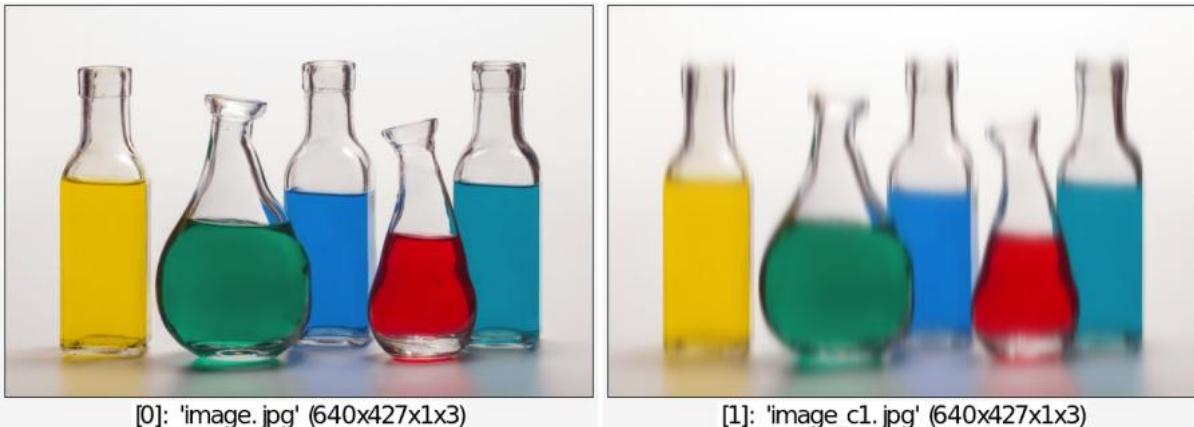
### Default values:

`boundary_conditions=1`.

This command has a [tutorial page](#).

### Example of use:

```
$ gmic image.jpg +blur_y 6
```



---

## blur\_z

### Arguments:

- `amplitude[%]>=0, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

### Description:

Blur selected images along the z-axis.

### Default values:

`boundary_conditions=1`.

This command has a [tutorial page](#).

---

## boundingbox3d

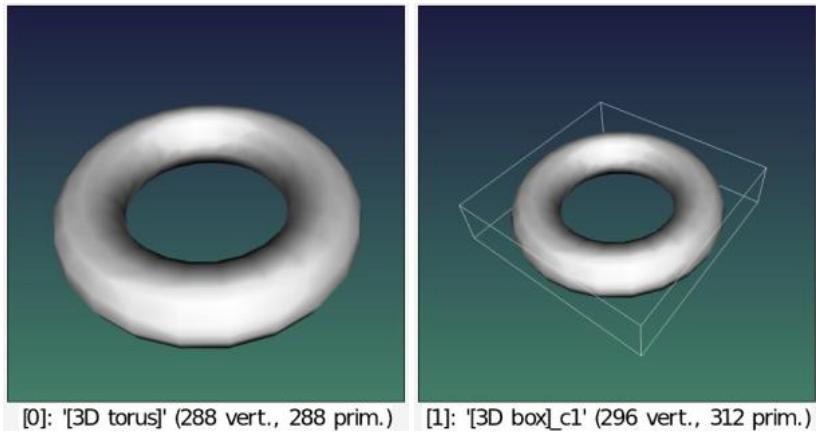
### No arguments

### Description:

Replace selected 3D objects by their 3D bounding boxes.

### Example of use:

```
$ gmic torus3d 100,30 +boundingbox3d +3d[-1] [-2]
```



## box3d

### Arguments:

- `_size_x, _size_y, _size_z`

### Description:

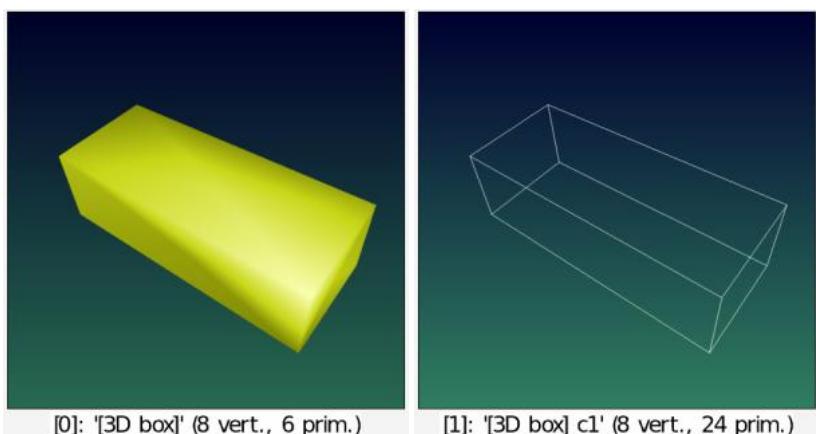
Input 3D box at (0,0,0), with specified geometry.

### Default values:

`size_x=1` and `size_z=size_y=size_x`.

### Example of use:

```
$ gmic box3d 100,40,30 +primitives3d 1 color3d[-2] ${-rgb}
```



## boxfilter

Built-in command

### Arguments:

- `size>=0[%], _order, _boundary_conditions, _nb_iter>=0` or

- `axes`, `size>=0[%]`, `_order`, `_boundary_conditions`, `_nb_iter>=0`

## Description:

Blur selected images by a box filter of specified size (fast recursive implementation).

`order` can be `{ 0=smooth | 1=1st-derivative | 2=2nd-derivative }`.

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

When specified, argument `axes` is a sequence of `{ x | y | z | c }`.

Specifying one axis multiple times apply also the blur multiple times.

## Default values:

`order=0`, `boundary_conditions=1` and `nb_iter=1`.

## Examples of use:

- Example #1

```
$ gmic image.jpg +boxfilter 5%
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

- Example #2

```
$ gmic image.jpg +boxfilter y,3,1
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

# boxfitting

## Arguments:

- `_min_box_size>=1, _max_box_size>=0, _initial_density>=0, _nb_attempts>=1`

## Description:

Apply box fitting effect on selected images, as displayed the web page:

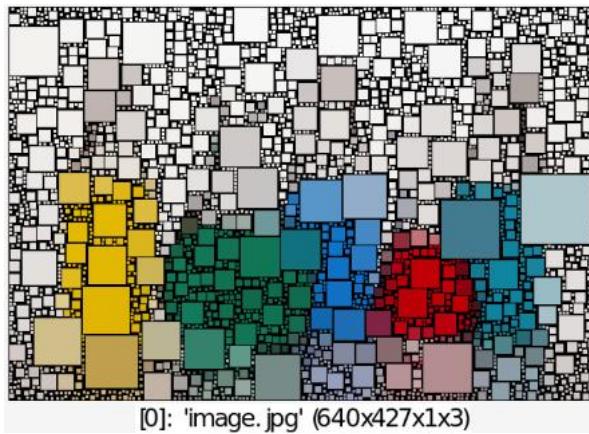
<http://www.complexification.net/gallery/machines/boxFittingImg/>.

## Default values:

`min_box_size=1`, `max_box_size=0`, `initial_density=0.1` and `nb_attempts=3`.

## Example of use:

```
$ gmic image.jpg boxfitting ,
```



---

# break

Built-in command

No arguments

## Description:

Break current `repeat...done`, `do...while` or `local...endlocal` block.

## Example of use:

```
$ gmic image.jpg repeat 10 blur 1 if l==1 break fi deform 10 done
```



---

## brushify

### Arguments:

- [brush], \_brush\_nb\_sizes>=1, 0<=\_brush\_min\_size\_factor<=1, \_brush\_nb\_orientatio

### Description:

Apply specified brush to create painterly versions of specified images.

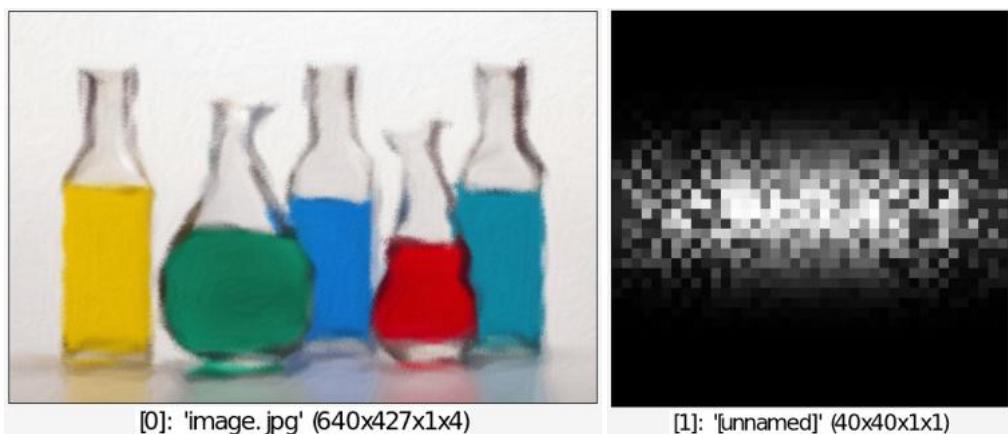
`brush_light_type` can be { `0=none` | `1=flat` | `2=darken` | `3=lighten` | `4=full` }.

### Default values:

```
brush_nb_sizes=3, brush_min_size_factor=0.66, brush_nb_orientations=12,  
brush_light_type=0, brush_light_strength=0.25, brush_opacity=0.8,  
painting_density=20%, painting_contours_coherence=0.9,  
painting_orientation_coherence=0.9, painting_coherence_alpha=1,  
painting_coherence_sigma=1, painting_primary_angle=0,  
painting_angle_dispersion=0.2
```

### Example of use:

```
$ gmic image.jpg 40,40 gaussian[-1] 10,4 spread[-1] 10,0 brushify[0]  
[1],1
```



# bsl

Built-in command

## Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

## Description:

Compute the bitwise left shift of selected images with specified value, image or mathematical expression, or compute the pointwise sequential bitwise left shift of selected images.

(equivalent to shortcut command `<<`).

## Example of use:

```
$ gmic image.jpg bsl 'round(3*x/w,0)' cut 0,255
```



# bsr

Built-in command

## Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

## Description:

Compute the bitwise right shift of selected images with specified value, image or mathematical expression, or compute the pointwise sequential bitwise right shift of selected images.

(equivalent to shortcut command `>>`).

## Example of use:

```
$ gmic image.jpg bsr 'round(3*x/w,0)' cut 0,255
```



---

## bump2normal

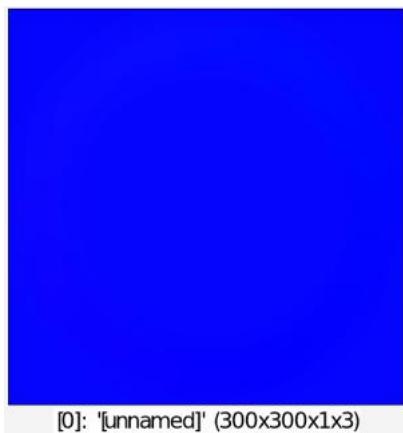
No arguments

### Description:

Convert selected bumpmaps to normalmaps.

## Example of use:

```
$ gmic 300,300 circle 50%,50%,128,1,1 blur 5% bump2normal
```



---

## camera

Built-in command

### Arguments:

- `_camera_index>=0, _nb_frames>0, _skip_frames>=0, _capture_width>=0, _capture_hei`

## Description:

Insert one or several frames from specified camera.

When `nb_frames==0`, the camera stream is released instead of capturing new images.  
This command requires features from the OpenCV library (not enabled in G'MIC by default).

## Default values:

`camera_index=0` (default camera), `nb_frames=1`, `skip_frames=0` and  
`capture_width=capture_height=0` (default size).

---

# cartoon

## Arguments:

- `_smoothness, _sharpening, _threshold>=0, _thickness>=0, _color>=0, quantization>0`

## Description:

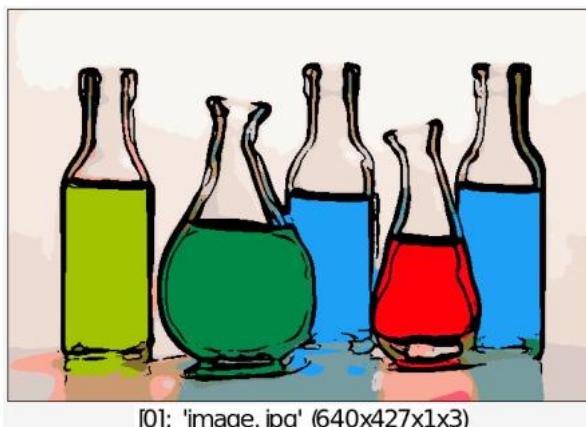
Apply cartoon effect on selected images.

## Default values:

`smoothness=3`, `sharpening=150`, `threshold=20`, `thickness=0.25`, `color=1.5` and  
`quantization=8`.

## Example of use:

```
$ gmic image.jpg cartoon 3,50,10,0.25,3,16
```



---

# cast

## Arguments:

- `datatype_source,datatype_target`

## Description:

Cast datatype of image buffer from specified source type to specified target type.

`datatype_source` and `datatype_target` can be `{ uchar | char | ushort | short | uint | int | uint64 | int64 | float | double }`.

---

## center3d

### No arguments

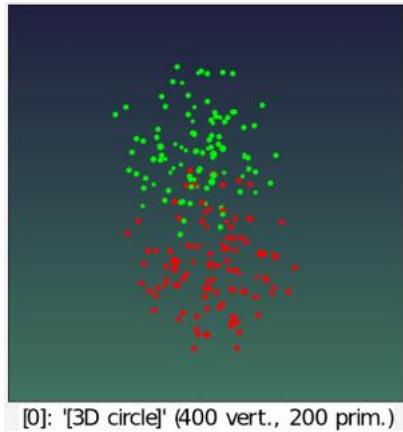
## Description:

Center selected 3D objects at (0,0,0).

(equivalent to shortcut command `c3d`).

## Example of use:

```
$ gmic repeat 100 circle3d {u(100)},{u(100)},{u(100)},2 done add3d  
color3d[-1] 255,0,0 +center3d color3d[-1] 0,255,0 add3d
```



## channels

## Arguments:

- `c0[%],_c1[%]`

## Description:

Keep only specified channels of selected images.

Dirichlet boundary is used when specified channels are out of range.

## Default values:

`c1=c0`.

## Examples of use:

- Example #1

```
$ gmic image.jpg channels 0,1
```



[0]: 'image.jpg' (640x427x1x2)

- Example #2

```
$ gmic image.jpg luminance channels 0,2
```



[0]: 'image.jpg' (640x427x1x3)

---

## check

Built-in command

### Arguments:

- `condition`

### Description:

Evaluate specified condition and display an error message if evaluated to false.

---

## check3d

Built-in command

### Arguments:

- `_is_full_check={ 0 | 1 }`

### Description:

Check validity of selected 3D vector objects, and display an error message

if one of the selected images is not a valid 3D vector object.

Full 3D object check is slower but more precise.

### Default values:

`is_full_check=1`.

---

## chessboard

### Arguments:

- `_size1>0,_size2>0,_offset1,_offset2,_angle,_opacity,_color1,...,_color2,...`

### Description:

Draw chessboard on selected images.

### Default values:

`size2=size1`, `offset1=offset2=0`, `angle=0`, `opacity=1`, `color1=0` and `color2=255`.

### Example of use:

```
$ gmic image.jpg chessboard 32,32,0,0,25,0.3,255,128,0,0,128,255
```



---

## cie1931

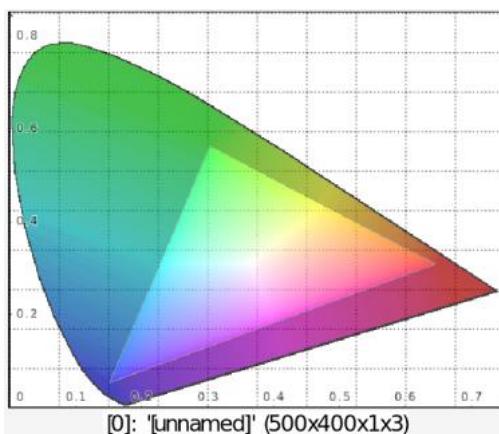
**No arguments**

**Description:**

Draw CIE-1931 chromaticity diagram on selected images.

**Example of use:**

```
$ gmic 500,400,1,3 cie1931
```



---

## circle

**Arguments:**

- `x[%],y[%],R[%],_opacity,_pattern,_color1,...`

**Description:**

Draw specified colored circle on selected images.

A radius of `100%` stands for `sqrt(width^2+height^2)`.

**pattern** is an hexadecimal number starting with **0x** which can be omitted even if a color is specified. If a pattern is specified, the circle is drawn outlined instead of filled.

## Default values:

**opacity=1**, **pattern=(undefined)** and **color1=0**.

## Example of use:

```
$ gmic image.jpg repeat 300 circle {u(100)}%,{u(100)}%,
{u(30)},0.3,${-rgb} done circle 50%,50%,100,0.7,255
```



---

## circle3d

### Arguments:

- **\_x0,\_y0,\_z0,\_radius>=0**

### Description:

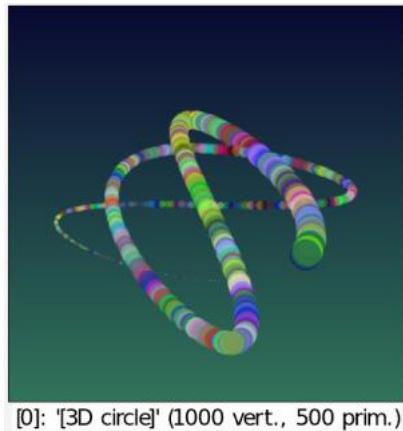
Input 3D circle at specified coordinates.

## Default values:

**x0=y0=z0=0** and **radius=1**.

## Example of use:

```
$ gmic repeat 500 a={$>*pi/250} circle3d {\cos(3*$a)},{\sin(2*$a)},0,
{$a/50} color3d[-1] ${-rgb},0.4 done add3d
```



---

## circles3d

### Arguments:

- `_radius>=0, _is_wireframe={ 0 | 1 }`

### Description:

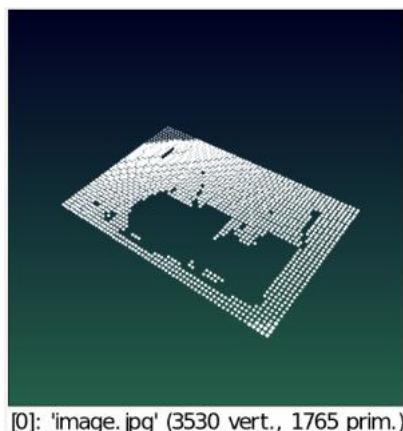
Convert specified 3D objects to sets of 3D circles with specified radius.

### Default values:

`radius=1` and `is_wireframe=1`.

### Example of use:

```
$ gmic image.jpg luminance resize2dy 40 threshold 50% * 255  
pointcloud3d color3d[-1] 255,255,255 circles3d 0.7
```



---

## close\_binary

### Arguments:

- `0<=_endpoint_rate<=100, _endpoint_connectivity>=0, _spline_distmax>=0, _segment { 0 | 1 }`

## Description:

Automatically close open shapes in binary images (defining white strokes on black background).

## Default values:

```
endpoint_rate=75, endpoint_connectivity=2, spline_distmax=80,  
segment_distmax=20, spline_anglemax=90, spline_roundness=1, area_min=100,  
allow_self_intersection=1.
```

---

# clut

## Arguments:

- `"clut_name", _resolution>0, _cut_and_round={ 0=no | 1=yes }`

## Description:

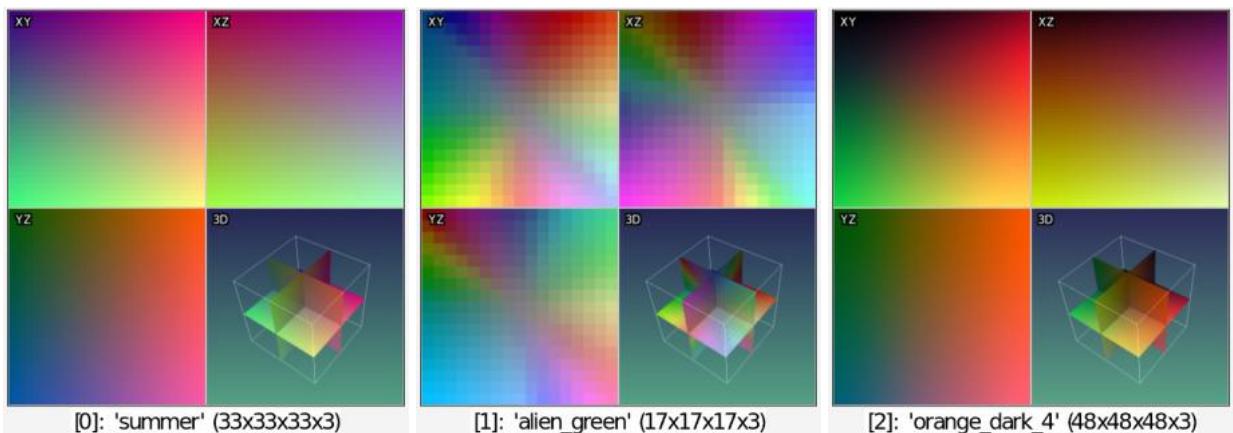
Insert one of the 958 pre-defined CLUTs at the end of the image list.

## Default values:

Default values: `resolution=33` and `cut_and_round=1`.

## Example of use:

```
$ gmic clut summer clut alien_green,17 clut orange_dark4,48
```



---

# cmy2rgb

## No arguments

### Description:

Convert color representation of selected images from CMY to RGB.

---

## cmyk2rgb

## No arguments

### Description:

Convert color representation of selected images from CMYK to RGB.

---

## color3d

Built-in command

### Arguments:

- `R, _G, _B, _opacity` or
- `(no arg)`

### Description:

Set color and opacity of selected 3D objects.

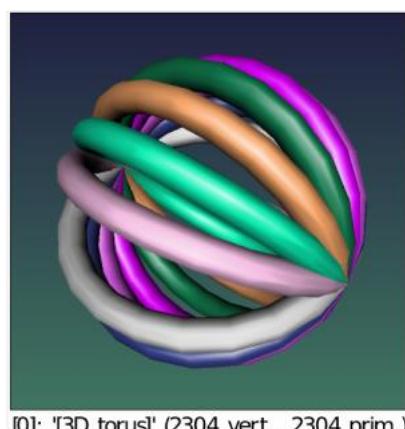
(equivalent to shortcut command `col3d`).

### Default values:

`B=G=R` and `opacity=(undefined)`.

### Example of use:

```
$ gmic torus3d 100,10 double3d 0 repeat 7 +rotate3d[-1] 1,0,0,20  
color3d[-1] ${-rgb} done add3d
```



---

# color\_ellipses

## Arguments:

- `_count>0, _radius>=0, _opacity>=0`

## Description:

Add random color ellipses to selected images.

## Default values:

`count=400, radius=5` and `opacity=0.1`.

## Example of use:

```
$ gmic image.jpg +color_ellipses , ,0.15
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

# colorblind

## Arguments:

- `type={ 0=protanopia | 1=protanomaly | 2=deutanopia | 3=deuteranomaly | 4=tritanopia | 5=tritanomaly | 6=achromatopsia | 7=achromatomaly }`

## Description:

Simulate color blindness vision.

Simulation method of Vienot, Brettel & Mollon 1999, "Digital video colourmaps for checking the legibility of displays by dichromats".

The dichromacy matrices of the paper were adapted to sRGB (RGB->XYZ).

Anomalous trichromacy simulated via linear interpolation with the identity and a factor of 0.6.

## Example of use:

```
$ gmic image.jpg +colorblind 0
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## colorcube3d

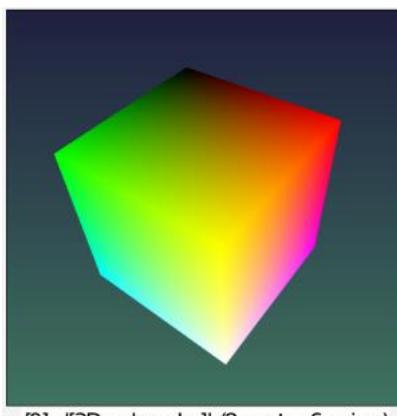
### No arguments

### Description:

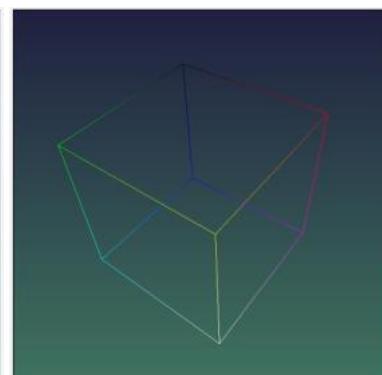
Input 3D color cube.

### Example of use:

```
$ gmic colorcube3d mode3d 2 +primitives3d 1
```



[0]: '[3D colorcube]' (8 vert., 6 prim.)



[1]: '[3D colorcube]\_c1'  
(8 vert., 24 prim.)

---

## colormap

### Arguments:

- `nb_levels>=0, _method={ 0=median-cut | 1=k-means }, _sort_vectors`

## Description:

Estimate best-fitting colormap with `nb_colors` entries, to index selected images.

Set `nb_levels==0` to extract all existing colors of an image.

`sort_vectors` can be `{ 0=unsorted | 1=by increasing norm | 2=by decreasing occurrence }`.

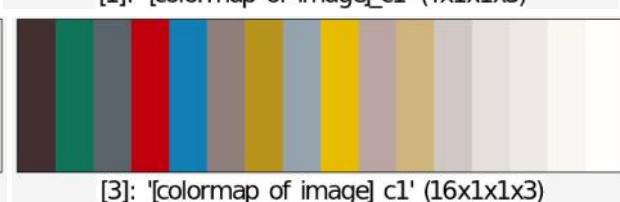
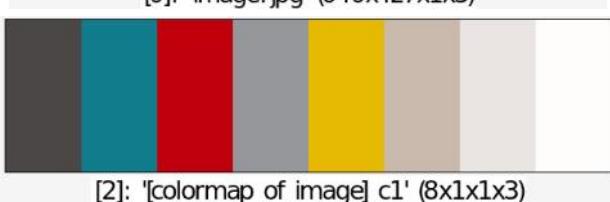
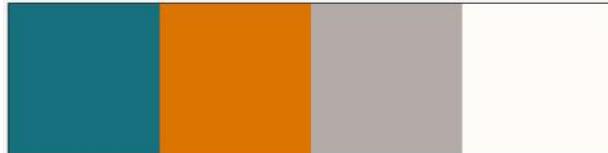
## Default values:

`method=1` and `sort_vectors=1`.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg +colormap[0] 4 +colormap[0] 8 +colormap[0] 16
```



## columns

### Arguments:

- `x0[%],_x1[%]`

## Description:

Keep only specified columns of selected images.

Dirichlet boundary is used when specified columns are out of range.

## Default values:

`x1=x0`.

## Example of use:

```
$ gmic image.jpg columns -25%,50%
```



## command

Built-in command

### Arguments:

- `_add_debug_info={ 0 | 1 },{ filename | http[s]://URL | "string" }`

### Description:

Import G'MIC custom commands from specified file, URL or string.

(equivalent to shortcut command [m](#)).

Imported commands are available directly after the `command` invocation.

### Default values:

`add_debug_info=1`.

## Example of use:

```
$ gmic image.jpg command "foo : mirror y deform $""1" +foo[0] 5  
+foo[0] 15
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)

---

## complex2polar

**No arguments**

**Description:**

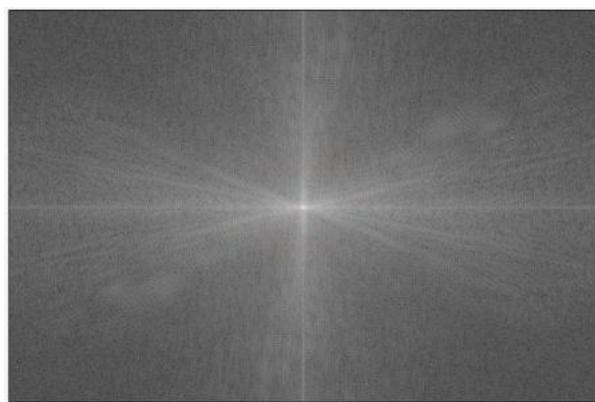
Compute complex to polar transforms of selected images.

**Example of use:**

```
$ gmic image.jpg +fft complex2polar[-2,-1] log[-2] shift[-2]  
50%,50%,0,0,2 remove[-1]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

# compose\_channels

No arguments

## Description:

Compose all channels of each selected image, using specified arithmetic operator (+,-,or,min,...).

## Default values:

**1=+**.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg +compose_channels and
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x1)

# compose\_freq

No arguments

## Description:

Compose selected low and high frequency parts into new images.

## Example of use:

```
$ gmic image.jpg split_freq 2% mirror[-1] x compose_freq
```



## compress\_clut

### Arguments:

- `_max_error>0, _avg_error>0, _max_nbpoints>=8 | 0 (unlimited), _error_metric={ 0=L2-norm | 1=deltaE_1976 | 2=deltaE_2000 }, _reconstruction_colorspace={ 0=srgb | 1=rgb | 2=lab }, _try_rbf_first={ 0 | 1 }`

### Description:

Compress selected color LUTs as sequences of colored keypoints.

### Default values:

`max_error=1.5, avg_error=0.75, max_nb_points=2048, error_metric=2, reconstruction_colorspace=0` and `try_rbf_first=1`.

---

## compress\_rle

### Arguments:

- `_is_binary_data={ 0 | 1 }, _maximum_sequence_length>=0`

### Description:

Compress selected images as 2xN data matrices, using RLE algorithm.

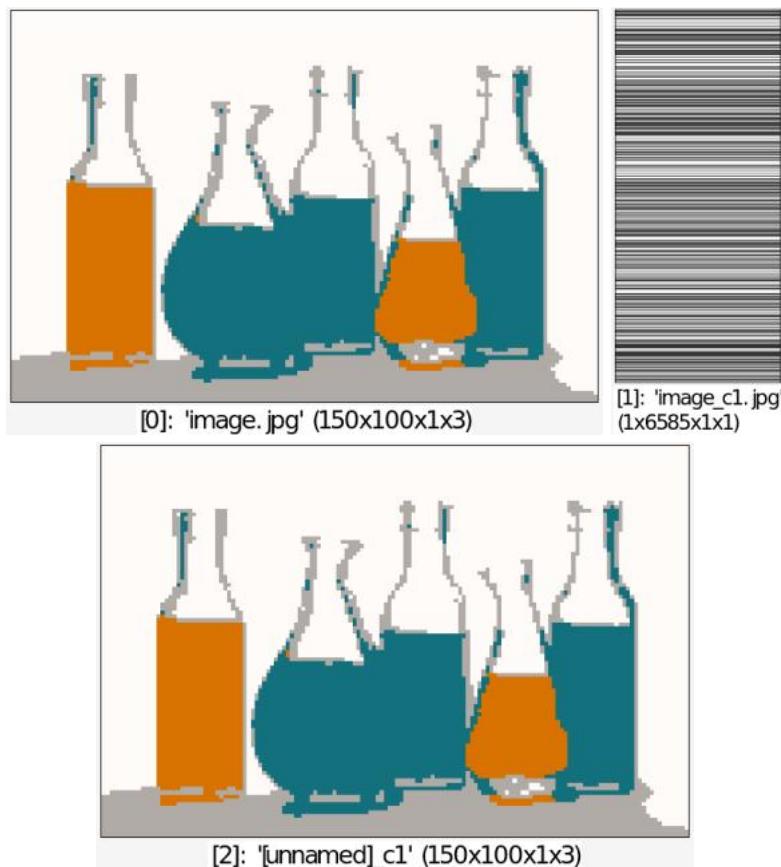
Set `maximum_sequence_length=0` to disable maximum length constraint.

### Default values:

`is_binary_data=0` and `maximum_sequence_length=0`.

### Example of use:

```
$ gmic image.jpg resize2dy 100 quantize 4 round +compress_rle ,  
+decompress_rle[-1]
```



---

## cone3d

### Arguments:

- `_radius, _height, _nb_subdivisions>0`

### Description:

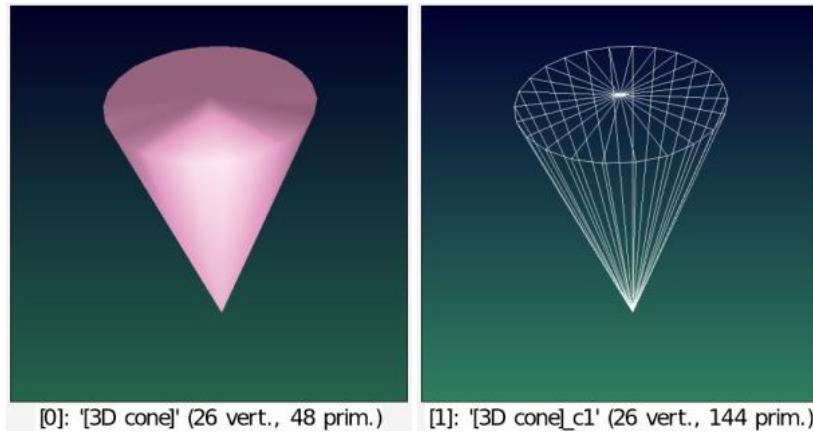
Input 3D cone at (0,0,0), with specified geometry.

### Default values:

`radius=1, height=1` and `nb_subdivisions=24`.

### Example of use:

```
$ gmic cone3d 10,40 +primitives3d 1 color3d[-2] ${-rgb}
```



---

## continue

Built-in command

No arguments

**Description:**

Go to end of current `repeat...done`, `do...while` or `local...endlocal` block.

**Example of use:**

```
$ gmic image.jpg repeat 10 blur 1 if 1==1 continue fi deform 10 done
```



[0]: 'image.jpg' (640x427x1x3)

---

## convolve

Built-in command

**Arguments:**

- `[mask],_boundary_conditions,_is_normalized={ 0 | 1 }`  
`},_channel_mode,_xcenter,_ycenter,_zcenter,_xstart,_ystart,_zstart,_xend,_ye`

**Description:**

Convolve selected images by specified mask.

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

`channel_mode` can be `{ 0=sum input channels | 1=one-for-one | 2=expand }`.

`interpolation_type` can be `{ 0=nearest-neighbor | 1=linear }`.

## Default values:

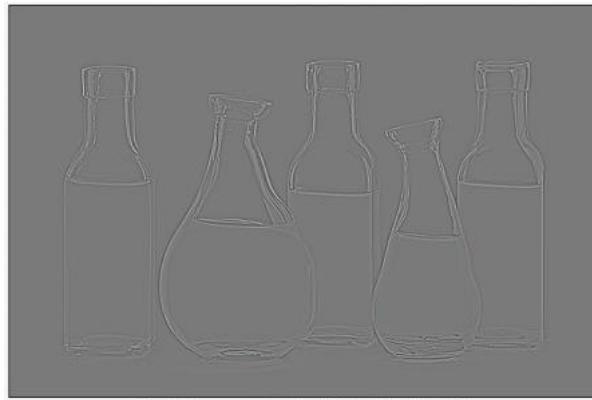
`boundary_conditions=1`, `is_normalized=0`, `channel_mode=1`,  
`xcenter=ycenter=zcenter=-1` (-1=centered), `xstart=ystart=zstart=0`,  
`xend=yend=zend=(max-coordinates)`, `xstride=ystride=zstride=1`,  
`xdilation=ydilation=zdilation=1` and `interpolation_type=0`.

This command has a [tutorial page](#).

## Examples of use:

- Example #1

```
$ gmic image.jpg (0,1,0;1,-4,1;0,1,0) convolve[-2] [-1] keep[-2]
```



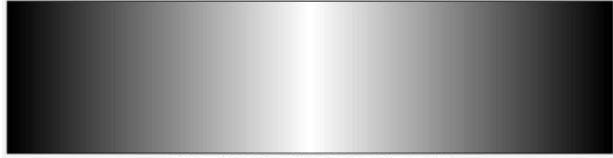
[0]: 'image.jpg' (640x427x1x3)

- Example #2

```
$ gmic image.jpg (0,1,0) resize[-1] 130,1,1,1,3 +convolve[0] [1]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: '(0,1,0)' (130x1x1x1)



[2]: 'image\_c1.jpg' (640x427x1x3)

---

## convolve\_fft

### Arguments:

- `[mask], _boundary_conditions`

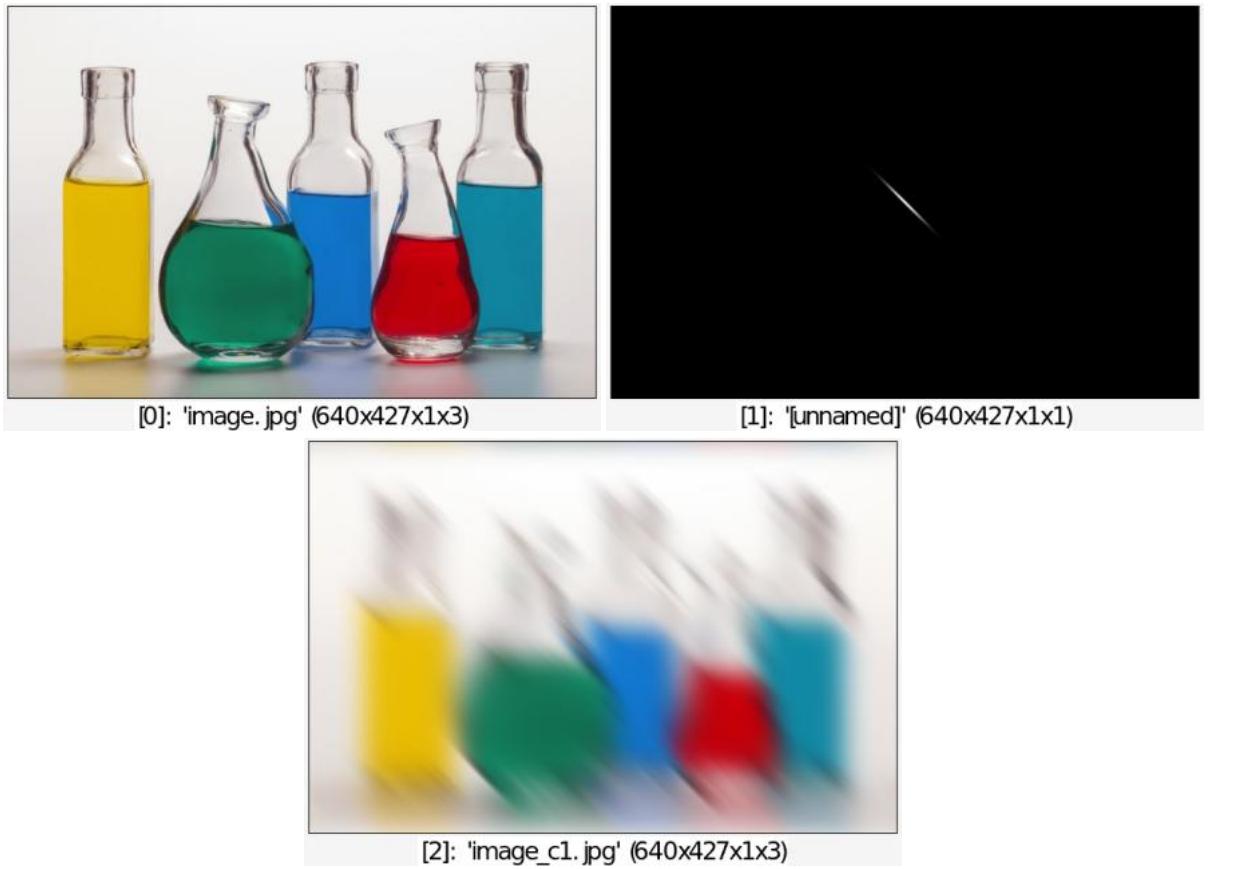
### Description:

Convolve selected images with specified mask, in the fourier domain.

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

### Example of use:

```
$ gmic image.jpg 100%,100% gaussian[-1] 20,1,45 +convolve_fft[0] [1]
```



## correlate

Built-in command

### Arguments:

- `[mask], _boundary_conditions, _is_normalized={ 0 | 1 }, _channel_mode, _xcenter, _ycenter, _zcenter, _xstart, _ystart, _zstart, _xend, _yend, _zend, _xstride, _ystride, _zstride, _xdilation, _ydilation, _zdilation`

### Description:

Correlate selected images by specified mask.

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

`channel_mode` can be `{ 0=sum input channels | 1=one-for-one | 2=expand }`.

`interpolation_type` can be `{ 0=nearest-neighbor | 1=linear }`.

### Default values:

`boundary_conditions=1, is_normalized=0, channel_mode=1, xcenter=ycenter=zcenter=-1` (-1=centered), `xstart=ystart=zstart=0, xend=yend=zend=(max-coordinates), xstride=ystride=zstride=1, xdilation=ydilation=zdilation=1` and `interpolation_type=0`.

### Examples of use:

- Example #1

```
$ gmic image.jpg (0,1,0;1,-4,1;0,1,0) correlate[-2] [-1] keep[-2]
```



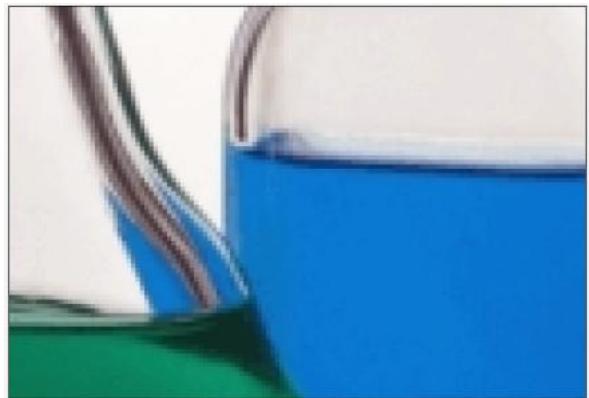
[0]: 'image.jpg' (640x427x1x3)

- **Example #2**

```
$ gmic image.jpg +crop 40%,40%,60%,60% +correlate[0] [-1],0,1
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (128x87x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)

---

## COS

Built-in command

### No arguments

### Description:

Compute the pointwise cosine of selected images.

This command has a [tutorial page](#).

## Examples of use:

- Example #1

```
$ gmic image.jpg +normalize 0,{2*pi} cos[-1]
```



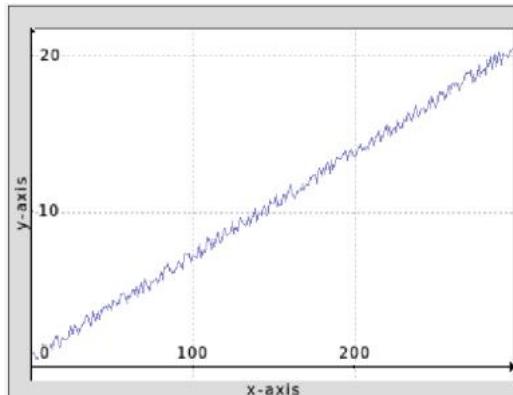
[0]: 'image.jpg' (640x427x1x3)



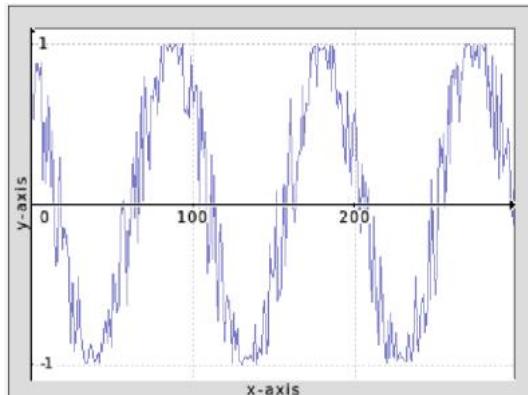
[1]: 'image\_c1.jpg' (640x427x1x3)

- Example #2

```
$ gmic 300,1,1,1,'20*x/w+u' +cos display_graph 400,300
```



[0]: '[image of "20\*x/w+u"]' (400x300x1x3)



[1]: '[image of "20\*x/w+u"]\_c1' (400x300x1x3)

## cosh

Built-in command

No arguments

Description:

Compute the pointwise hyperbolic cosine of selected images.

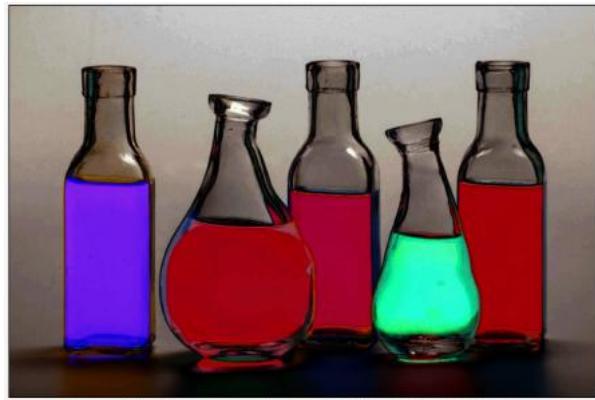
## Examples of use:

- Example #1

```
$ gmic image.jpg +normalize -3,3 cosh[-1]
```



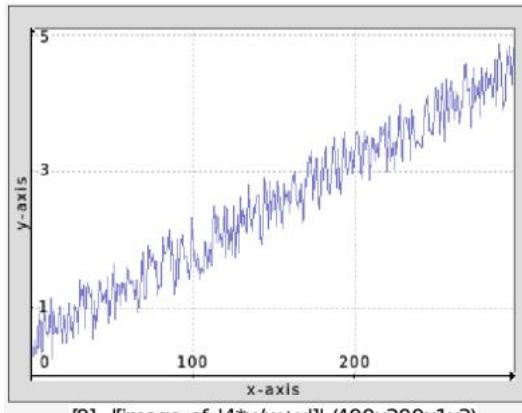
[0]: 'image.jpg' (640x427x1x3)



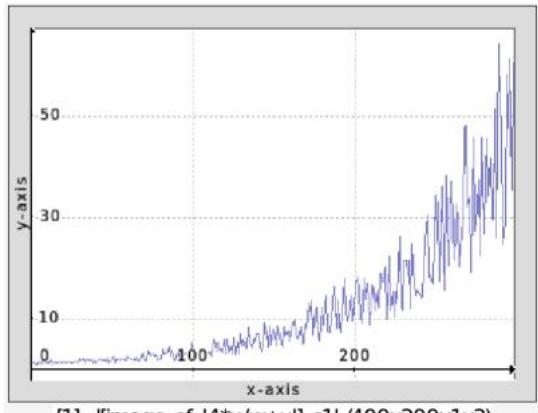
[1]: 'image\_c1.jpg' (640x427x1x3)

## • Example #2

```
$ gmic 300,1,1,1,'4*x/w+u' +cosh display_graph 400,300
```



[0]: '[image of "4\*x/w+u"]' (400x300x1x3)



[1]: '[image of "4\*x/w+u"]\_c1' (400x300x1x3)

---

## covariance\_colors

### Arguments:

- avg\_outvarname

### Description:

Return the covariance matrix of the vector-valued colors in the latest of the selected images

(for arbitrary number of channels).

Parameter **avg\_outvarname** is used as a variable name that takes the value of the average vector-value.

---

## cracks

## Arguments:

- `0<=_density<=100,_is_relief={ 0 | 1 },_opacity,_color1,...`

## Description:

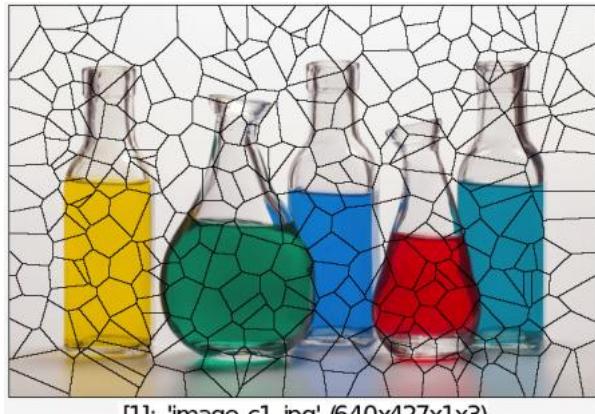
Draw random cracks on selected images with specified color.

## Default values:

`density=25`, `is_relief=0`, `opacity=1` and `color1=0`.

## Example of use:

```
$ gmic image.jpg +cracks ,
```



## crop

Built-in command

## Arguments:

- `x0[%],x1[%],_boundary_conditions` or
- `x0[%],y0[%],x1[%],y1[%],_boundary_conditions` or
- `x0[%],y0[%],z0[%],x1[%],y1[%],z1[%],_boundary_conditions` or
- `x0[%],y0[%],z0[%],c0[%],x1[%],y1[%],z1[%],c1[%],_boundary_conditions`

## Description:

Crop selected images with specified region coordinates.

(equivalent to shortcut command `z`).

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

## Default values:

`boundary_conditions=0`.

## Examples of use:

- **Example #1**

```
$ gmic image.jpg +crop -230,-230,280,280,1 crop[0]  
-230,-230,280,280,0
```



- **Example #2**

```
$ gmic image.jpg crop 25%,25%,75%,75%
```



---

## cross\_correlation

### Arguments:

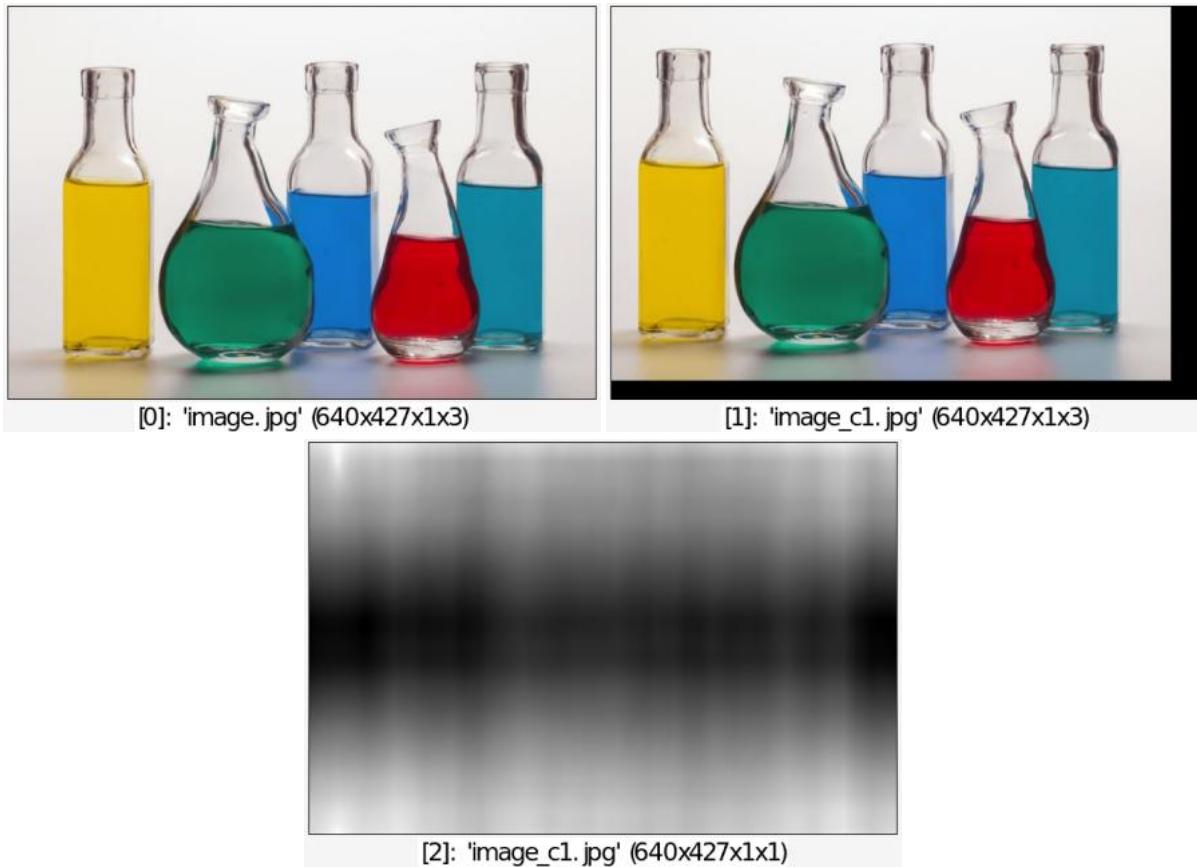
- `[mask]`

### Description:

Compute cross-correlation of selected images with specified mask.

### Example of use:

```
$ gmic image.jpg +shift -30,-20 +cross_correlation[0] [1]
```



## cubes3d

### Arguments:

- `_size>=0`

### Description:

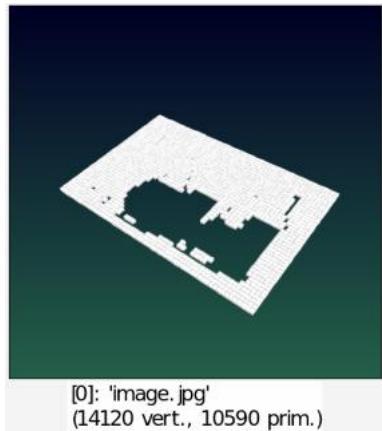
Convert specified 3D objects to sets of 3D cubes with specified size.

### Default values:

`size=1`.

### Example of use:

```
$ gmic image.jpg luminance resize2dy 40 threshold 50% * 255
pointcloud3d color3d[-1] 255,255,255 cubes3d 1
```



---

## cubism

### Arguments:

- `_density>=0, 0<=_thickness<=50, _max_angle, _opacity, _smoothness>=0`

### Description:

Apply cubism effect on selected images.

### Default values:

`density=50, thickness=10, max_angle=75, opacity=0.7` and `smoothness=0`.

### Example of use:

```
$ gmic image.jpg cubism ,
```



---

## cumulate

Built-in command

### Arguments:

- `{ x | y | z | c }...{ x | y | z | c }` or

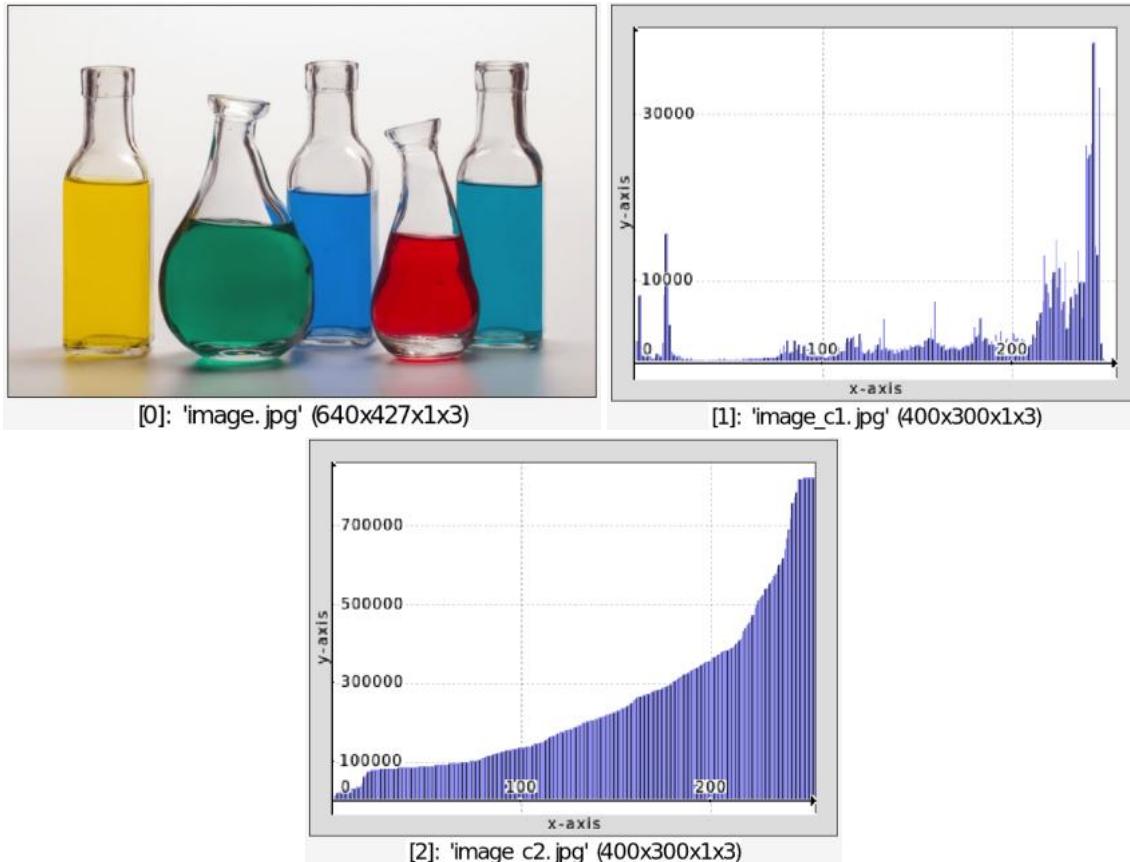
- (no arg)

## Description:

Compute the cumulative function of specified image data, optionally along the specified axes.

## Example of use:

```
$ gmic image.jpg +histogram 256 +cumulate[-1] display_graph[-2,-1]
400,300,3
```



## cup3d

### Arguments:

- `_resolution>0`

## Description:

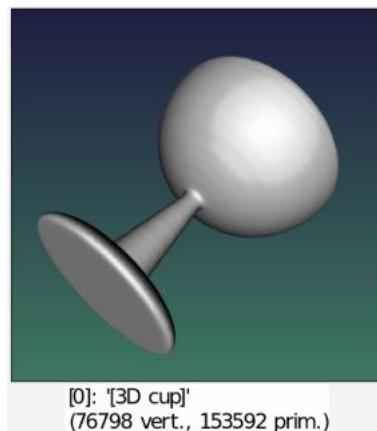
Input 3D cup object.

## Default values:

`resolution=128`.

## Example of use:

```
$ gmic cup3d ,
```



---

## cursor

Built-in command

### Arguments:

- `_mode = { 0=hide | 1=show }`

### Description:

Show or hide mouse cursor for selected instant display windows.

Command selection (if any) stands for instant display window indices instead of image indices.

### Default values:

`mode=1`.

---

## curvature

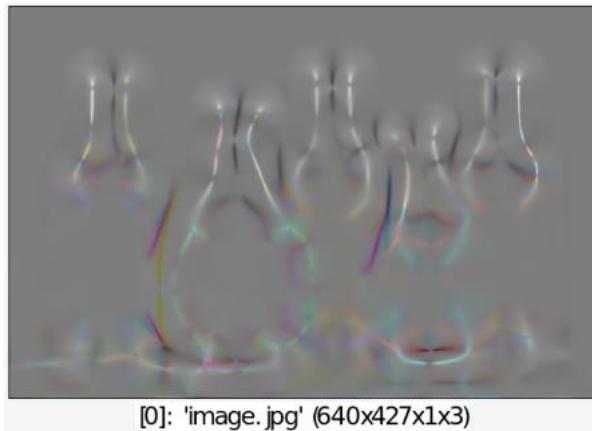
### No arguments

### Description:

Compute isophote curvatures on selected images.

### Example of use:

```
$ gmic image.jpg blur 10 curvature
```



# cut

Built-in command

## Arguments:

- { value0[%] | [image0] }, { value1[%] | [image1] } or
- [image]

## Description:

Cut values of selected images in specified range.

(equivalent to shortcut command [c](#)).

## Examples of use:

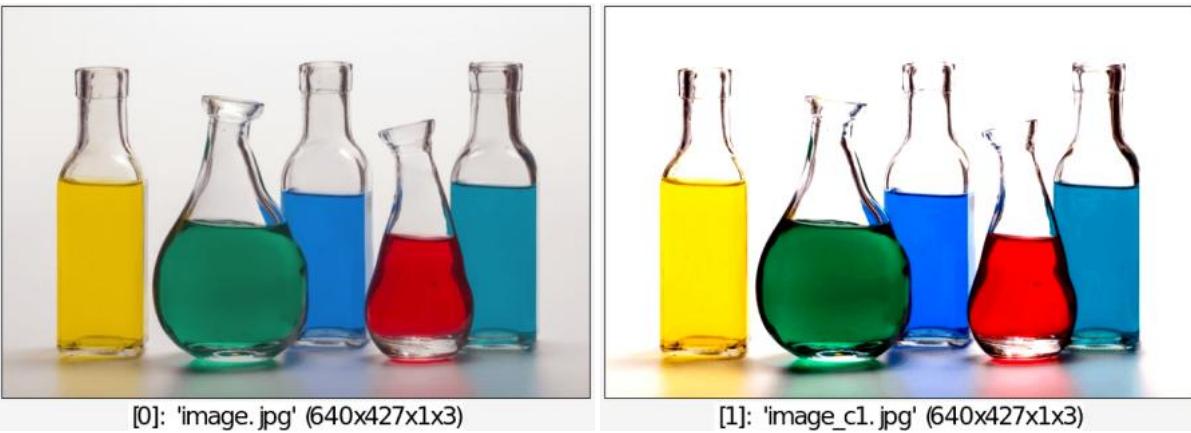
- Example #1

```
$ gmic image.jpg +add 30% cut[-1] 0,255
```



- Example #2

```
$ gmic image.jpg +cut 25%,75%
```



## cylinder3d

### Arguments:

- `_radius, _height, _nb_subdivisions>0`

### Description:

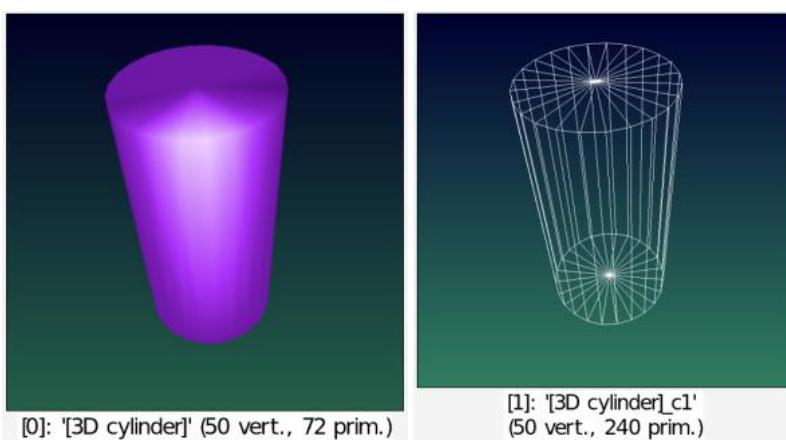
Input 3D cylinder at (0,0,0), with specified geometry.

### Default values:

`radius=1, height=1` and `nb_subdivisions=24`.

### Example of use:

```
$ gmic cylinder3d 10,40 +primitives3d 1 color3d[-2] ${-rgb}
```



## dct

### Arguments:

- `_{{ x | y | z }}...{{ x | y | z }}` or

- (no arg)

## Description:

Compute the discrete cosine transform of selected images, optionally along the specified axes only.

Output images are always evenly sized, so this command may change the size of the selected images.

## Default values:

(no arg)

## See also:

**idct**.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg +dct +idct[-1] abs[-2] +[-2] 1 log[-2]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x428x1x3)



[2]: 'image\_c2.jpg' (640x428x1x3)

---

## deblur

## Arguments:

- `amplitude[%]>=0, _nb_iter>=0, _dt>=0, _regul>=0, _regul_type={ 0=Tikhonov | 1=meancurv. | 2=TV }`

## Description:

Deblur image using a regularized Jansson-Van Cittert algorithm.

## Default values:

`nb_iter=10`, `dt=20`, `regul=0.7` and `regul_type=1`.

## Example of use:

```
$ gmic image.jpg blur 3 +deblur 3,40,20,0.01
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

## deblur\_goldmeinel

### Arguments:

- `sigma>=0, _nb_iter>=0, _acceleration>=0, _kernel_type={ 0=deriche | 1=gaussian }.`

## Description:

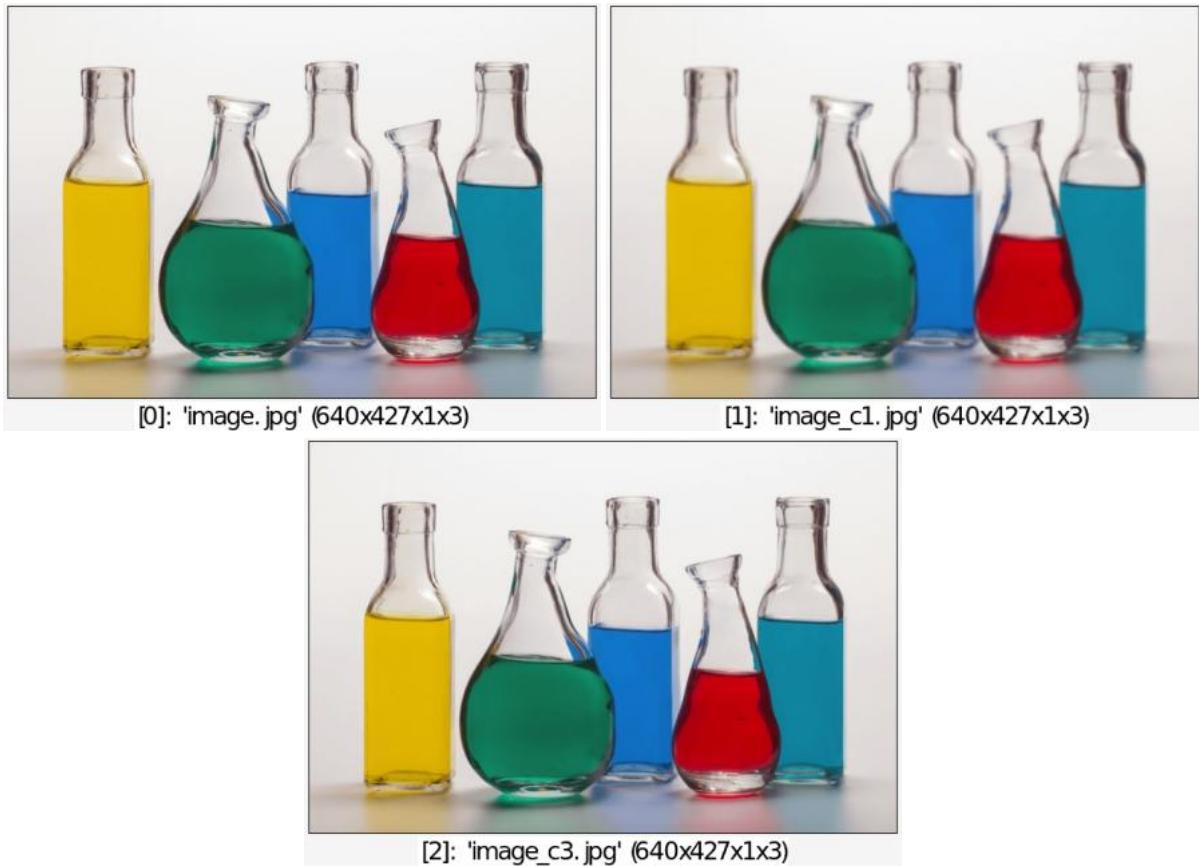
Deblur selected images using Gold-Meinel algorithm

## Default values:

`nb_iter=8`, `acceleration=1` and `kernel_type=1`.

## Example of use:

```
$ gmic image.jpg +blur 1 +deblur_goldmeinel[-1] 1
```



---

## deblur\_richardsonlucy

### Arguments:

- `sigma>=0, nb_iter>=0, _kernel_type={ 0=deriche | 1=gaussian }.`

### Description:

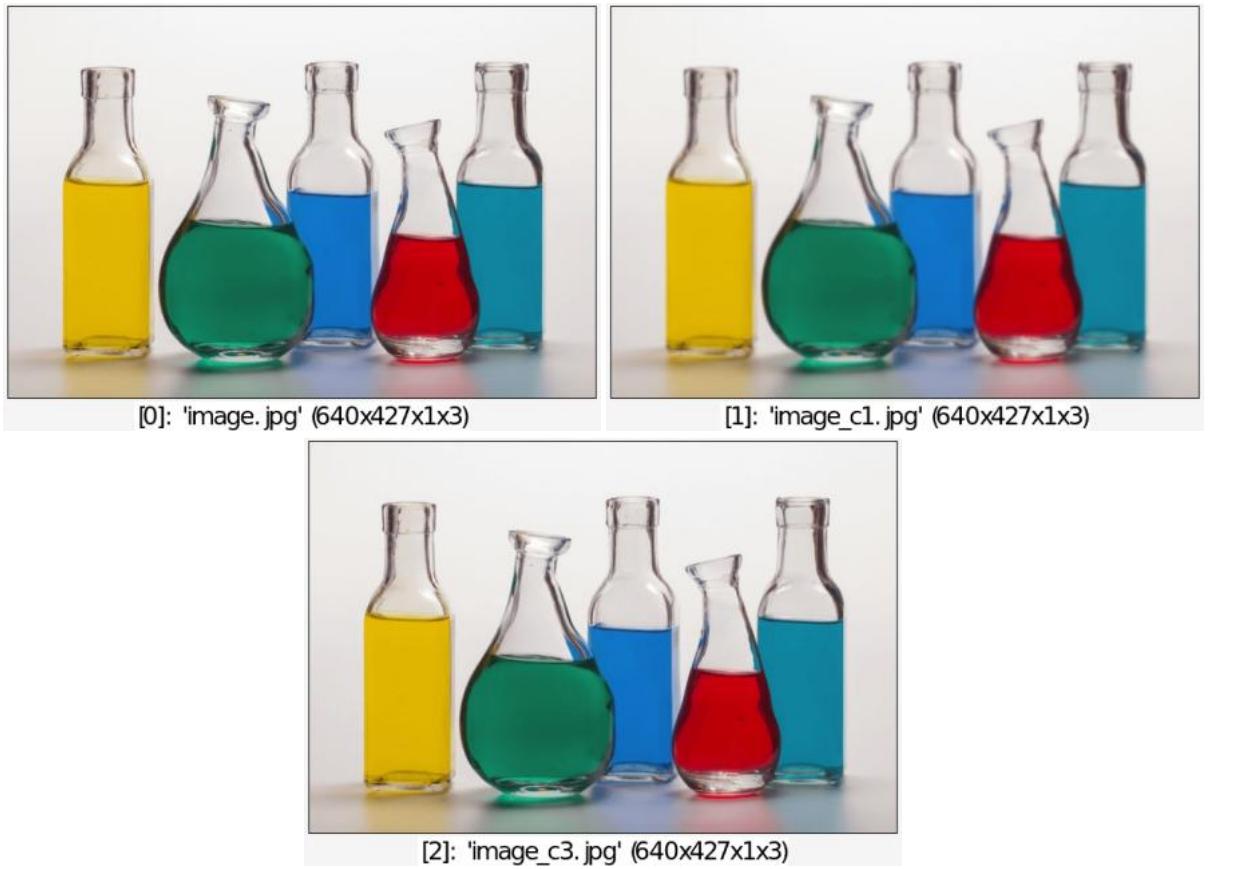
Deblur selected images using Richardson-Lucy algorithm.

### Default values:

`nb_iter=50` and `kernel_type=1`.

### Example of use:

```
$ gmic image.jpg +blur 1 +deblur_richardsonlucy[-1] 1
```



## debug

Built-in command

### No arguments

### Description:

Activate debug mode.

When activated, the G'MIC interpreter becomes very verbose and outputs additional log messages about its internal state on the standard output (stdout).

This option is useful for developers or to report possible bugs of the interpreter.

## dec

### Arguments:

- `decimal_int1,...`

### Description:

Print specified decimal integers into their binary, octal, hexadecimal and string representations.

## dec2bin

## **Arguments:**

- `decimal_int1,...`

## **Description:**

Convert specified decimal integers into their binary representations.

---

## **dec2hex**

### **Arguments:**

- `decimal_int1,...`

### **Description:**

Convert specified decimal integers into their hexadecimal representations.

---

## **dec2oct**

### **Arguments:**

- `decimal_int1,...`

### **Description:**

Convert specified decimal integers into their octal representations.

---

## **dec2str**

### **Arguments:**

- `decimal_int1,...`

### **Description:**

Convert specifial decimal integers into its string representation.

---

## **decompress\_clut**

### **Arguments:**

- `_width>0, _height>0, _depth>0, _reconstruction_colorspace={ 0=srgb | 1=rgb | 2=lab }`

## Description:

Decompress selected colored keypoints into 3D CLUTs, using a mixed RBF/PDE approach.

## Default values:

`width=height=depth=33` and `reconstruction_colorspace=0`.

---

# decompress\_clut\_pde

## Arguments:

- `_width>0, _height>0, _depth>0, _reconstruction_colorspace={ 0=srgb | 1=rgb | 2=lab }`

## Description:

Decompress selected colored keypoints into 3D CLUTs, using multiscale diffusion PDE's.

## Default values:

`width=height=depth=33` and `reconstruction_colorspace=0`.

---

# decompress\_clut\_rbf

## Arguments:

- `_width>0, _height>0, _depth>0, _reconstruction_colorspace={ 0=srgb | 1=rgb | 2=lab }`

## Description:

Decompress selected colored keypoints into 3D CLUTs, using RBF thin plate spline interpolation.

## Default values:

`width=height=depth=33` and `reconstruction_colorspace=0`.

---

# decompress\_rle

## No arguments

## Description:

Decompress selected data vectors, using RLE algorithm.

---

## deconvolve\_fft

### Arguments:

- [kernel], \_regularization $\geq 0$

### Description:

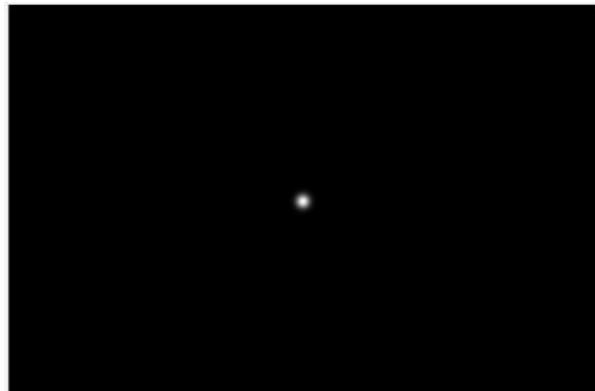
Deconvolve selected images by specified mask in the fourier space.

### Default values:

`regularization $\geq 0$` .

### Example of use:

```
$ gmic image.jpg +gaussian 5 +convolve_fft[0] [1] +deconvolve_fft[-1]  
[1]
```



---

## deform

## Arguments:

- `_amplitude>=0, _interpolation`

## Description:

Apply random smooth deformation on selected images.

`interpolation` can be `{ 0=none | 1=linear | 2=bicubic }`.

## Default values:

`amplitude=10`.

## Example of use:

```
$ gmic image.jpg +deform[0] 10 +deform[0] 20
```



---

## deg2rad

### No arguments

## Description:

Convert pointwise angle values of selected images, from degrees to radians (apply `i*pi/180`).

# deinterlace

## Arguments:

- `_method={ 0 | 1 }`

## Description:

Deinterlace selected images (`method` can be `{ 0=standard or 1=motion-compensated }`).

## Default values:

`method=0`.

## Example of use:

```
$ gmic image.jpg +rotate 3,1,1,50%,50% resize 100%,50% resize  
100%,200%,1,3,4 shift[-1] 0,1 add +deinterlace 1
```



[0]: 'image.jpg' (640x428x1x3)



[1]: 'image\_c1.jpg' (640x428x1x3)

---

# delaunay

## Arguments:

- `_output_type={ 0=image | 1=coordinates/triangles }`

## Description:

Generate discrete 2D Delaunay triangulation of non-zero pixels in selected images.

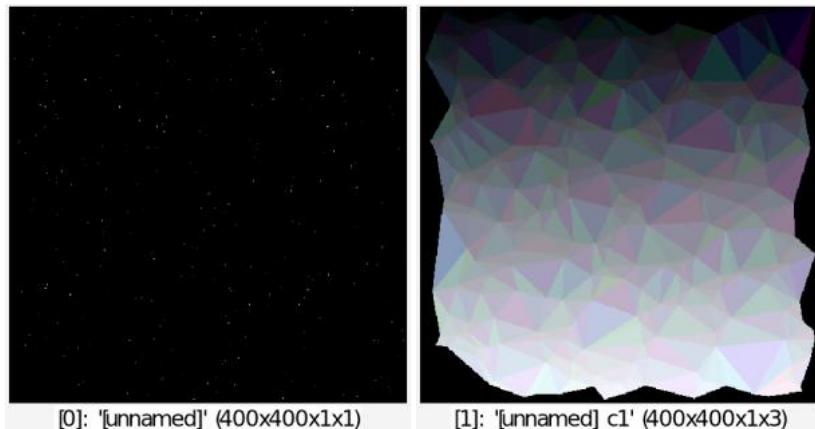
Input images must be scalar.

Each pixel of the output image is a triplet (a,b,c) meaning the pixel belongs to the Delaunay triangle `ABC` where `a, b, c` are the labels of the pixels `A, B, C`.

## Examples of use:

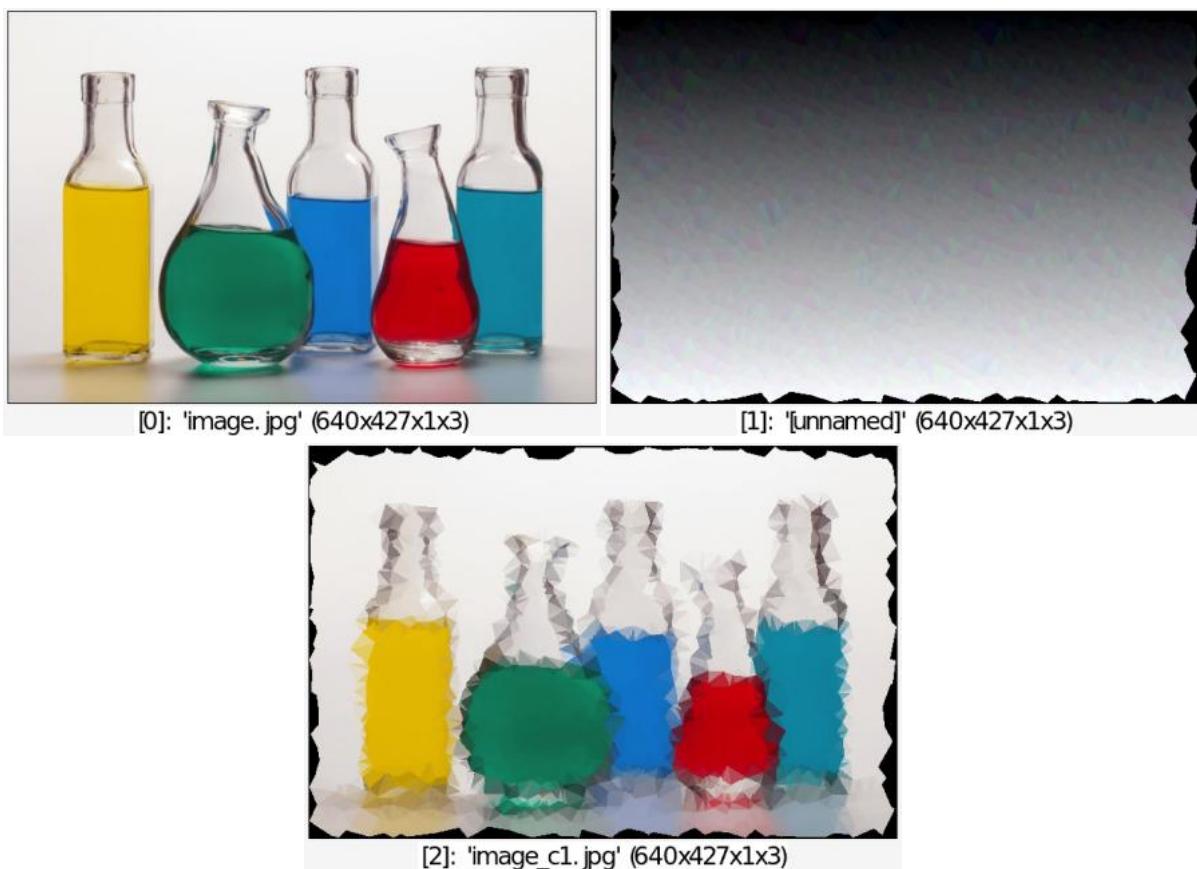
- Example #1

```
$ gmic 400,400 rand 32,255 100%,100% noise. 0.4,2 eq. 1 mul +delaunay
```



- **Example #2**

```
$ gmic image.jpg 100%,100% noise. 2,2 eq. 1 delaunay. +blend  
shapeaverage0
```



---

## delaunay3d

No arguments

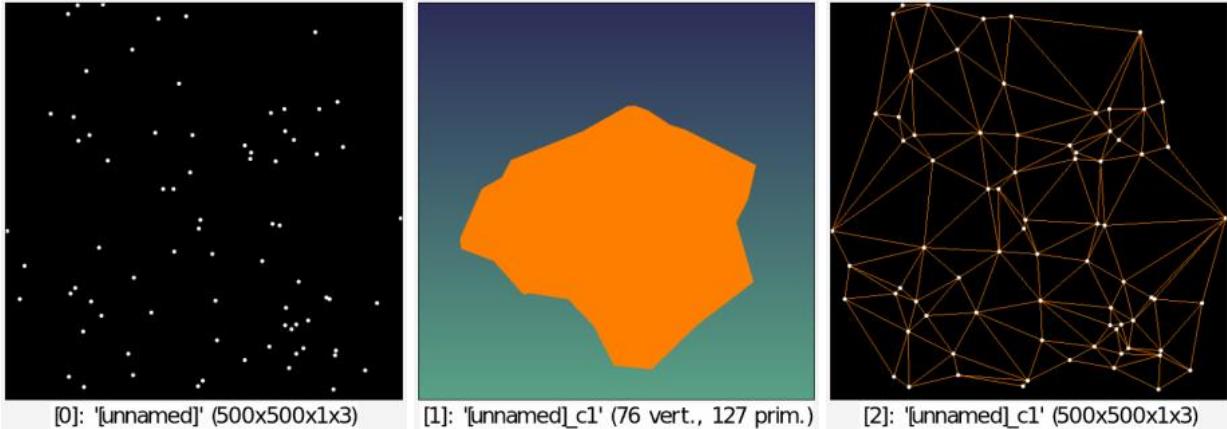
**Description:**

Generate 3D Delaunay triangulations from selected images.

One assumes that the selected input images are binary images containing the set of points to mesh. The output 3D object is a mesh composed of non-oriented triangles.

## Example of use:

```
$ gmic 500,500 noise 0.05,2 eq 1 * 255 +delaunay3d color3d[1]
255,128,0 dilate_circ[0] 5 to_rgb[0] +object3d[0] [1],0,0,0,1,1
max[-1] [0]
```



## delete

Built-in command

### Arguments:

- `[filename1[,filename2,...]]`

### Description:

Delete specified filenames on disk. Multiple filenames must be separated by commas.

## deltaE

### Arguments:

- `[image], _metric={ 0=deltaE_1976 | 1=deltaE_2000 },"_to_Lab_command"`

### Description:

Compute the CIE DeltaE color difference between selected images and specified [image].

Argument `to_Lab_command` is a command able to convert colors of [image] into a Lab representation.

### Default values:

`metric=1` and `to_Lab_command="srgb2lab"`.

## Example of use:

```
$ gmic image.jpg +blur 2 +deltaE[0] [1],1,srgb2lab
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x1)

## demos

### Arguments:

- `_run_in_parallel={ 0=no | 1=yes | 2=auto }`

### Description:

Show a menu to select and view all G'MIC interactive demos.

## denoise

Built-in command

### Arguments:

- `[guide],_std_deviations[%]>=0,_std_deviations_r[%]>=0,_patch_size>0,_lookup_s { 0 | 1 } or`
- `std_deviations[%]>=0,_std_deviations_r[%]>=0,_patch_size>0,_lookup_size>0,_s { 0 | 1 }`

## Description:

Denoise selected images by non-local patch averaging.

## Default values:

`std_deviation_p=10`, `patch_size=5`, `lookup_size=6` and `smoothness=1`.

## Example of use:

```
$ gmic image.jpg +denoise 5,5,8
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## denoise\_cnn

### Arguments:

- `_noise_type={ 0=soft | 1=heavy | 2=heavy (faster) | 3=poisson+gaussian }`, `_patch_size>0`

## Description:

Denoise selected images using a convolutional neural network (CNN).

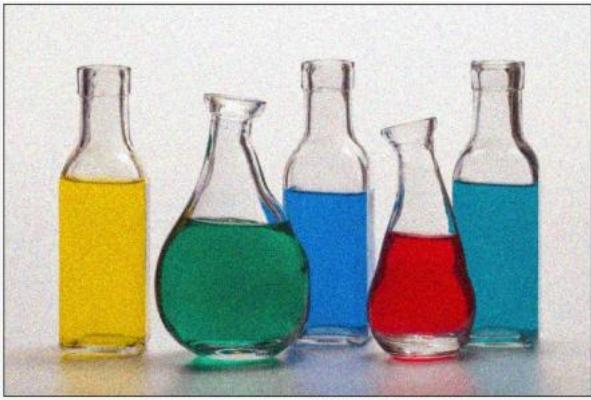
Input value range should be [0,255]. Output value range is [0,255].

## Default values:

`patch_size=64`.

## Example of use:

```
$ gmic image.jpg noise 20 cut 0,255 +denoise_cnn
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## denoise\_haar

### Arguments:

- `_threshold>=0, _nb_scales>=0, _cycle_spinning>0`

### Description:

Denoise selected images using haar-wavelet thresholding with cycle spinning.

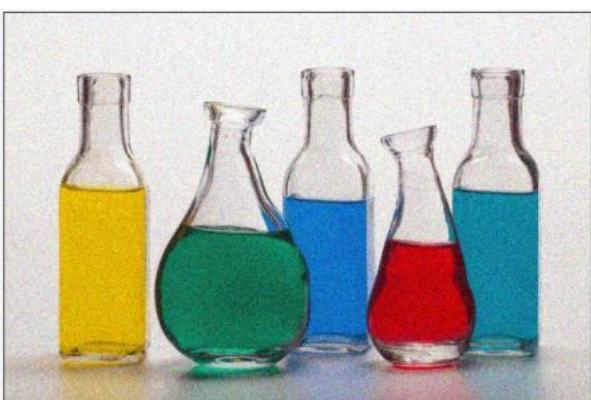
Set `nb_scales==0` to automatically determine the optimal number of scales.

### Default values:

`threshold=1.4`, `nb_scale=0` and `cycle_spinning=10`.

### Example of use:

```
$ gmic image.jpg noise 20 cut 0,255 +denoise_haar[-1] 0.8
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## denoise\_patchpca

### Arguments:

- `_strength>=0, _patch_size>0, _lookup_size>0, _spatial_sampling>0`

## Description:

Denoise selected images using the patch-pca algorithm.

## Default values:

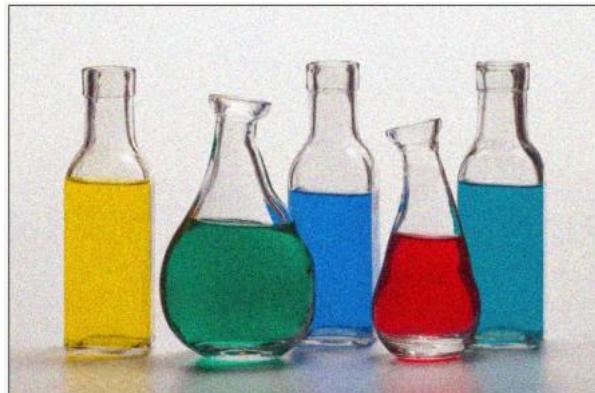
`patch_size=7, lookup_size=11, details=1.8` and `spatial_sampling=5`.

## Example of use:

```
$ gmic image.jpg +noise 20 cut[-1] 0,255 +denoise_patchpca[-1] ,
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)



[2]: 'image\_c2.jpg' (640x427x1x3)

## deriche

Built-in command

## Arguments:

- `std_deviation>=0[%],order={ 0 | 1 | 2 },axis={ x | y | z | c },_boundary_conditions`

## Description:

Apply Deriche recursive filter on selected images, along specified axis and with specified standard deviation, order and boundary conditions.

`boundary_conditions` can be { `0=dirichlet` | `1=neumann` | `2=periodic` | `3=mirror` }.

## Default values:

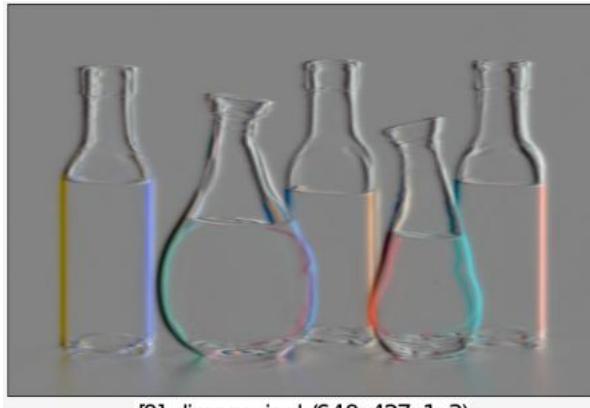
`boundary_conditions=1`.

This command has a [tutorial page](#).

## Examples of use:

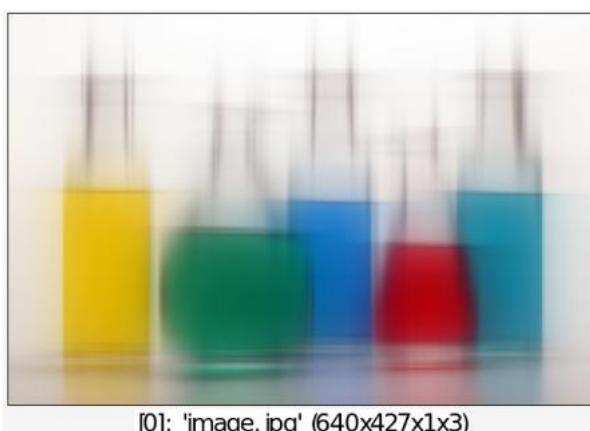
- Example #1

```
$ gmic image.jpg deriche 3,1,x
```



- Example #2

```
$ gmic image.jpg +deriche 30,0,x deriche[-2] 30,0,y add
```



---

## detect\_skin

### Arguments:

- `0<=tolerance<=1,_skin_x,_skin_y,_skin_radius>=0`

## Description:

Detect skin in selected color images and output an appertenance probability map.

Detection is performed using CbCr chromaticity data of skin pixels.

If arguments `skin_x`, `skin_y` and `skin_radius` are provided, skin pixels are learnt from the sample pixels inside the circle located at (`skin_x`, `skin_y`) with radius `skin_radius`.

## Default values:

`tolerance=0.5` and `skin_x=skin_y=radius=-1`.

---

# diagonal

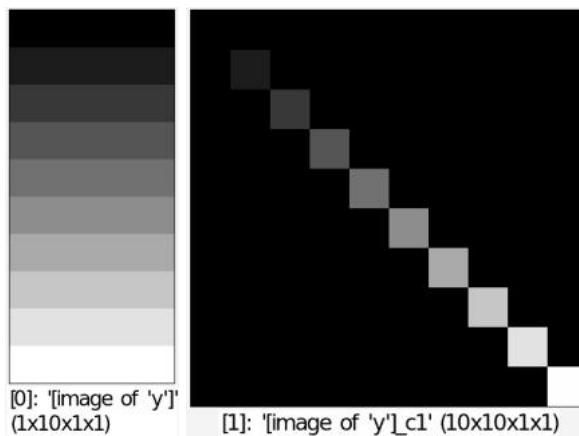
## No arguments

## Description:

Transform selected vectors as diagonal matrices.

## Example of use:

```
$ gmic 1,10,1,1,'y' +diagonal
```



# diffusiontensors

## Arguments:

- `_sharpness>=0,0<=_anisotropy<=1,_alpha[%],_sigma[%],is_sqrt={ 0 | 1 }`

## Description:

Compute the diffusion tensors of selected images for edge-preserving smoothing algorithms.

## Default values:

`sharpness=0.7`, `anisotropy=0.3`, `alpha=0.6`, `sigma=1.1` and `is_sqrt=0`.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg diffusiontensors 0.8 abs pow 0.2
```



---

## dijkstra

Built-in command

### Arguments:

- `starting_node>=0,ending_node>=0`

### Description:

Compute minimal distances and paths from specified adjacency matrices by the Dijkstra algorithm.

---

## dilate

Built-in command

### Arguments:

- `size>=0` or
- `size_x>=0,size_y>=0,size_z>=0` or
- `[kernel],_boundary_conditions,_is_real={ 0=binary-mode | 1=real-mode }`

### Description:

Dilate selected images by a rectangular or the specified structuring element.

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

### Default values:

`size_z=1`, `boundary_conditions=1` and `is_real=0`.

## Example of use:

```
$ gmic image.jpg +dilate 10
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## dilate\_circ

### Arguments:

- `_size>=0`, `_boundary_conditions`, `_is_normalized={ 0 | 1 }`

### Description:

Apply circular dilation of selected images by specified size.

### Default values:

`boundary_conditions=1` and `is_normalized=0`.

## Example of use:

```
$ gmic image.jpg +dilate_circ 7
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

# dilate\_oct

## Arguments:

- `_size>=0, _boundary_conditions, _is_normalized={ 0 | 1 }`

## Description:

Apply octagonal dilation of selected images by specified size.

## Default values:

`boundary_conditions=1` and `is_normalized=0`.

## Example of use:

```
$ gmic image.jpg +dilate_oct 7
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

# dilate\_threshold

## Arguments:

- `size_x>=1, size_y>=1, size_z>=1, _threshold>=0, _boundary_conditions`

## Description:

Dilate selected images in the (X,Y,Z,I) space.

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

## Default values:

`size_y=size_x`, `size_z=1`, `threshold=255` and `boundary_conditions=1`.

---

# direction2rgb

No arguments

## Description:

Compute RGB representation of selected 2D direction fields.

## Example of use:

```
$ gmic image.jpg luminance gradient append c blur 2 orientation  
+direction2rgb
```



[0]: 'image.jpg' (640x427x1x2)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

# discard

Built-in command

## Arguments:

- `_value1,_value2,...` or
- `{ x | y | z | c}...{ x | y | z | c},_value1,_value2,...` or
- `(no arg)`

## Description:

Discard specified values in selected images or discard neighboring duplicate values,

optionally only for the values along the first of a specified axis.

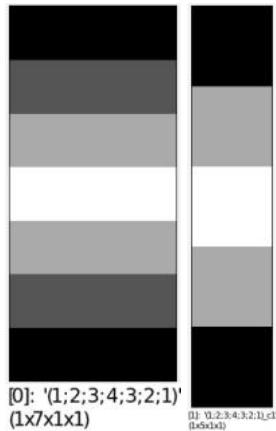
If no arguments are specified, neighboring duplicate values are discarded.

If all pixels of a selected image are discarded, an empty image is returned.

## Examples of use:

### • Example #1

```
$ gmic (1;2;3;4;3;2;1) +discard 2
```



## • Example #2

```
$ gmic (1,2,2,3,3,3,4,4,4,4) +discard x
```



# displacement

Built-in command

## Arguments:

- `[source_image],_smoothness,_precision>=0,_nb_scales>=0,_iteration_max>=0,is_{ 0 | 1 },_guide]`

## Description:

Estimate displacement field between specified source and selected target images.

If `smoothness>=0`, regularization type is set to isotropic, else to anisotropic.

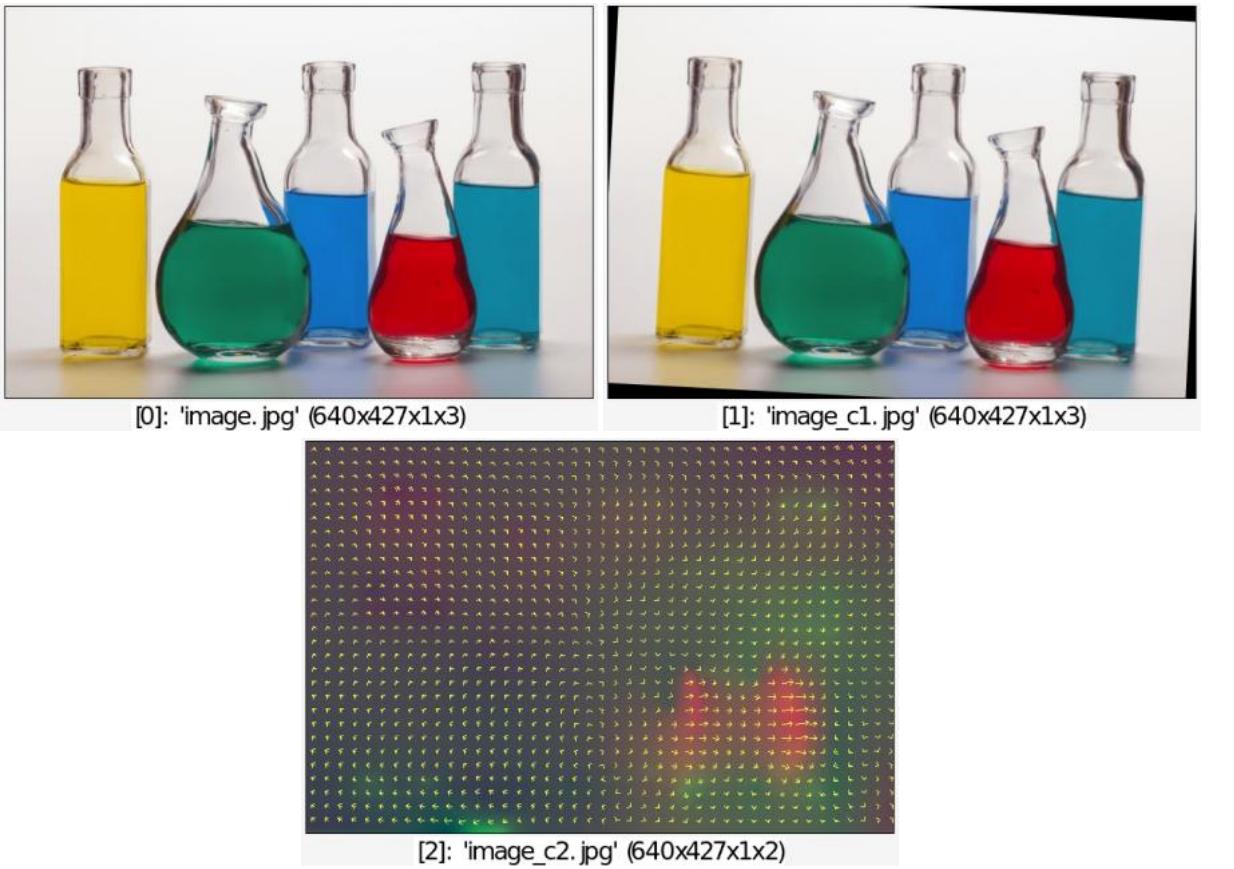
If `nbscales==0`, the number of scales used is estimated from the image size.

## Default values:

`smoothness=0.1, precision=5, nb_scales=0, iteration_max=10000, is_backward=1`  
and `[guide]=(unused)`.

## Example of use:

```
$ gmic image.jpg +rotate 3,1,0,50%,50% +displacement[-1] [-2]
quiver[-1] [-1],15,1,1,1,{1.5*iM}
```



## display

Built-in command

### Arguments:

- `_X[%]>=0,_Y[%]>=0,_Z[%]>=0,_exit_on_anykey={ 0 | 1 }`

### Description:

Display selected images in an interactive viewer (use the instant display window [0] if opened).

(equivalent to shortcut command `d`).

Arguments `X`, `Y`, `Z` determine the initial selection view, for 3D volumetric images.

### Default values:

`X=Y=Z=0` and `exit_on_anykey=0`.

This command has a [tutorial page](#).

## display0

### No arguments

## Description:

Display selected images without value normalization.

(equivalent to shortcut command `d0`).

---

# display2d

## No arguments

## Description:

Display selected 2d images in an interactive window.

(equivalent to shortcut command `d2d`).

This command is used by default by command `display` when displaying 2d images.

If selected image is a volumetric image, each slice is displayed on a separate display window (up to 10 images can be displayed simultaneously this way), with synchronized moves.

When interactive window is opened, the following actions are possible:

- Left mouse button: Create an image selection and zoom into it.
  - Middle mouse button, or CTRL+left mouse button: Move image.
  - Mouse wheel or PADD+/-: Zoom in/out.
  - Arrow keys: Move image left/right/up/down.
  - `CTRL + A`: Enable/disable transparency (show/hide alpha channel).
  - `CTRL + C`: Decrease window size.
  - `CTRL + D`: Increase window size.
  - `CTRL + F`: Toggle fullscreen mode.
  - `CTRL + N`: Change normalization mode (can be `{ none | normal | channel-by-channel }`).
  - `CTRL + O`: Save a copy of the input image, as a numbered file `gmic_xxxxxxx.gmz`.
  - `CTRL + R`: Reset both window size and view.
  - `CTRL + S`: Save a screenshot of the current view, as a numbered file `gmic_xxxxxxx.png`.
  - `CTRL + SPACE`: Reset view.
  - `CTRL + X`: Show/hide axes.
  - `CTRL + Z`: Hold/release aspect ratio.
- 

# display3d

## Arguments:

- `_background_image`, `_exit_on_anykey={ 0 | 1 }` or
- `_exit_on_anykey={ 0 | 1 }`

## Description:

Display selected 3D objects in an interactive viewer (use the instant display window [0] if opened).

(*equivalent to shortcut command* `d3d`).

## Default values:

`[background_image]=(default)` and `exit_on_anykey=0`.

---

# display\_array

## Arguments:

- `_width>0`, `_height>0`

## Description:

Display images in interactive windows where pixel neighborhoods can be explored.

## Default values:

`width=13` and `height=width`.

---

# display\_camera

## No arguments

## Description:

Open camera viewer.

This command requires features from the OpenCV library (not enabled in G'MIC by default).

---

# display\_fft

## No arguments

## Description:

Display fourier transform of selected images, with centered log-module and argument.

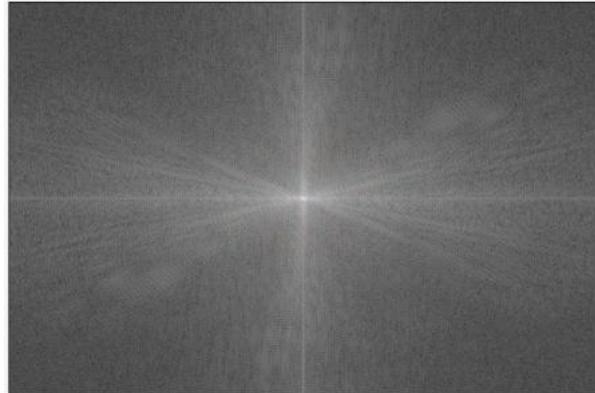
(equivalent to shortcut command `dfft`).

## Example of use:

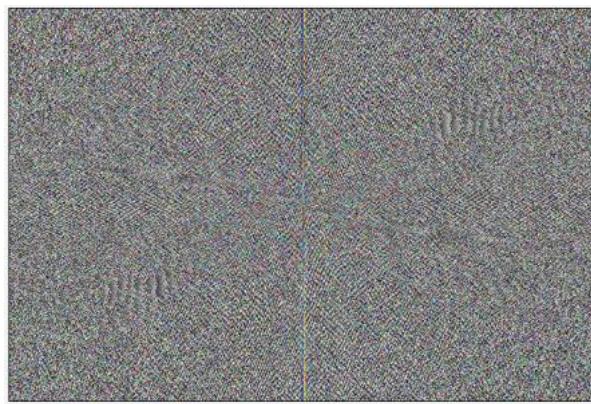
```
$ gmic image.jpg +display_fft
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)



[2]: 'image\_c2.jpg' (640x427x1x3)

## display\_graph

### Arguments:

- `_width>=0,_height>=0,_plot_type,_vertex_type,_xmin,_xmax,_ymin,_ymax,_xlabel`

### Description:

Render graph plot from selected image data.

`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 displayed xy-axes.

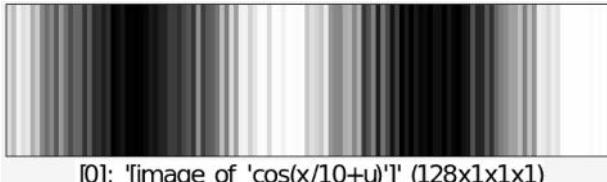
if specified `width` or `height` is `0`, then image size is set to half the screen size.

### Default values:

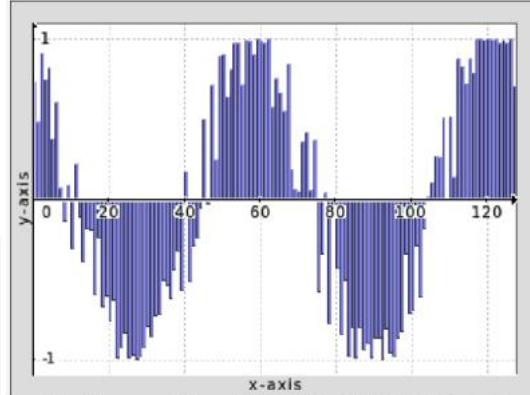
```
width=0, height=0, plot_type=1, vertex_type=1, 'xmin=xmax=ymin=ymax=0 (auto)',  
xlabel="x-axis" and ylabel="y-axis".
```

## Example of use:

```
$ gmic 128,1,1,1,'cos(x/10+u)' +display_graph 400,300,3
```



[0]: '[image of 'cos(x/10+u)']' (128x1x1x1)



# display\_histogram

## Arguments:

- `_width>=0,_height>=0,_clusters>0,_min_value[%],_max_value[%],_show_axes={0 | 1 },_expression.`

## Description:

Render a channel-by-channel histogram.

If selected images have several slices, the rendering is performed for all input slices.

`expression` is a mathematical expression used to transform the histogram data for visualization purpose.

(equivalent to shortcut command `dh`).

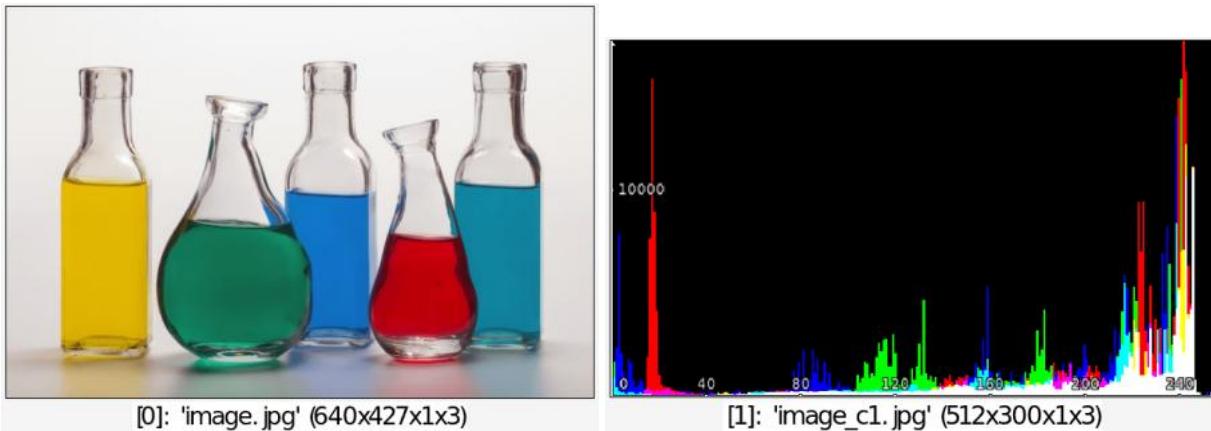
if specified `width` or `height` is `0`, then image size is set to half the screen size.

## Default values:

```
width=0, height=0, clusters=256, min_value=0%, max_value=100%, show_axes=1  
and expression=i.
```

## Example of use:

```
$ gmic image.jpg +display_histogram 512,300
```



## display\_parallel

**No arguments**

**Description:**

Display each selected image in a separate interactive display window.

(equivalent to shortcut command `dp`).

## display\_parallel0

**No arguments**

**Description:**

Display each selected image in a separate interactive display window, without value normalization.

(equivalent to shortcut command `dp0`).

## display\_parametric

**Arguments:**

- `_width>0,_height>0,_outline_opacity,_vertex_radius>=0,_is_antialiased={ 0 | 1 },_is_decorated={ 0 | 1 },_ xlabel,_ylabel`

**Description:**

Render 2D or 3D parametric curve or point clouds from selected image data.

Curve points are defined as pixels of a 2 or 3-channel image.

If the point image contains more than 3 channels, additional channels define the (R,G,B) color for

each vertex.

If `outline_opacity>1`, the outline is colored according to the specified vertex colors and `outline_opacity-1` is used as the actual drawing opacity.

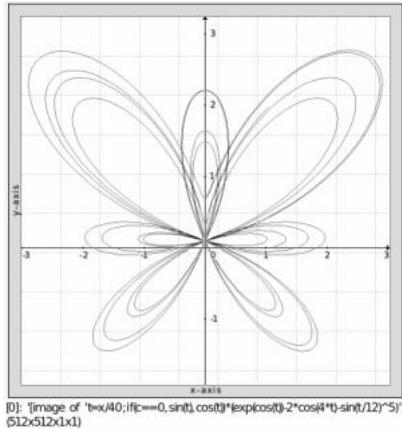
## Default values:

```
width=512, height=width, outline_opacity=3, vertex_radius=0,  
is_antialiased=1, is_decorated=1, xlabel="x-axis" and ylabel="y-axis".
```

## Examples of use:

- Example #1

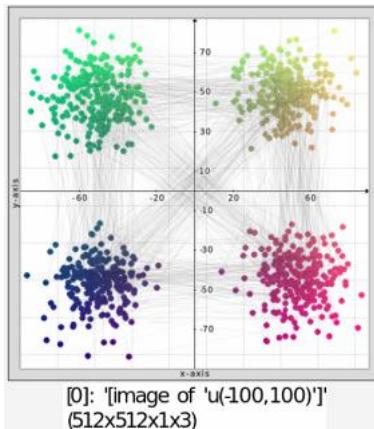
```
$ gmic 1024,1,1,2,'t=x/40;if(c==0,sin(t),cos(t))*  
(exp(cos(t))-2*cos(4*t)-sin(t/12)^5)' display_parametric 512,512
```



[0]: '[image of "t=x/40;if(c==0,sin(t),cos(t))\*  
(exp(cos(t))-2\*cos(4\*t)-sin(t/12)^5)"]'  
(512x512x1x1)

- Example #2

```
$ gmic 1000,1,1,2,u(-100,100) quantize 4,1 noise 12 channels 0,2  
+normalize 0,255 append c display_parametric 512,512,0.1,8
```



[0]: '[image of "u(-100,100)"]'  
(512x512x1x3)

---

## display\_polar

### Arguments:

- `_width>32,_height>32,_outline_type,_fill_R,_fill_G,_fill_B,_theta_start,_the`

## Description:

Render polar curve from selected image data.

`outline_type` can be `{ r<0=dots with radius -r | 0=no outline | r>0=lines+dots with radius r }`.

`fill_color` can be `{ -1=no fill | R,G,B=fill with specified color }`.

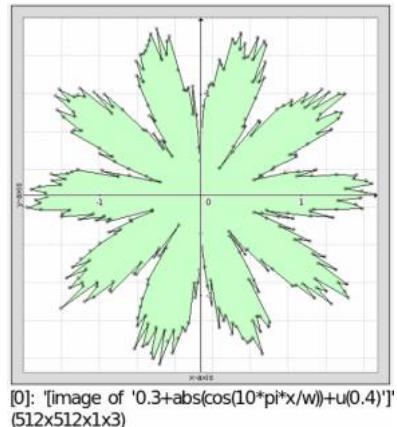
## Default values:

`width=500, height=width, outline_type=1, fill_R=fill_G=fill_B=200, theta_start=0, theta_end=360, xlabel="x-axis" and ylabel="y-axis"`.

## Examples of use:

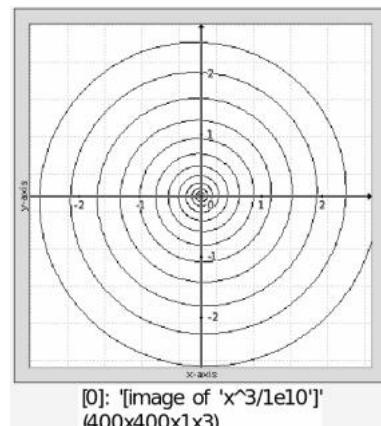
- Example #1

```
$ gmic 300,1,1,1,'0.3+abs(cos(10*pi*x/w))+u(0.4)' display_polar  
512,512,4,200,255,200
```



- Example #2

```
$ gmic 3000,1,1,1,'x^3/1e10' display_polar 400,400,1,-1,,,0,{15*360}
```



# display\_quiver

## Arguments:

- `_size_factor>0, _arrow_size>=0, _color_mode={ 0=monochrome | 1=grayscale | 2=color }`

## Description:

Render selected images of 2D vectors as a field of 2D arrows.

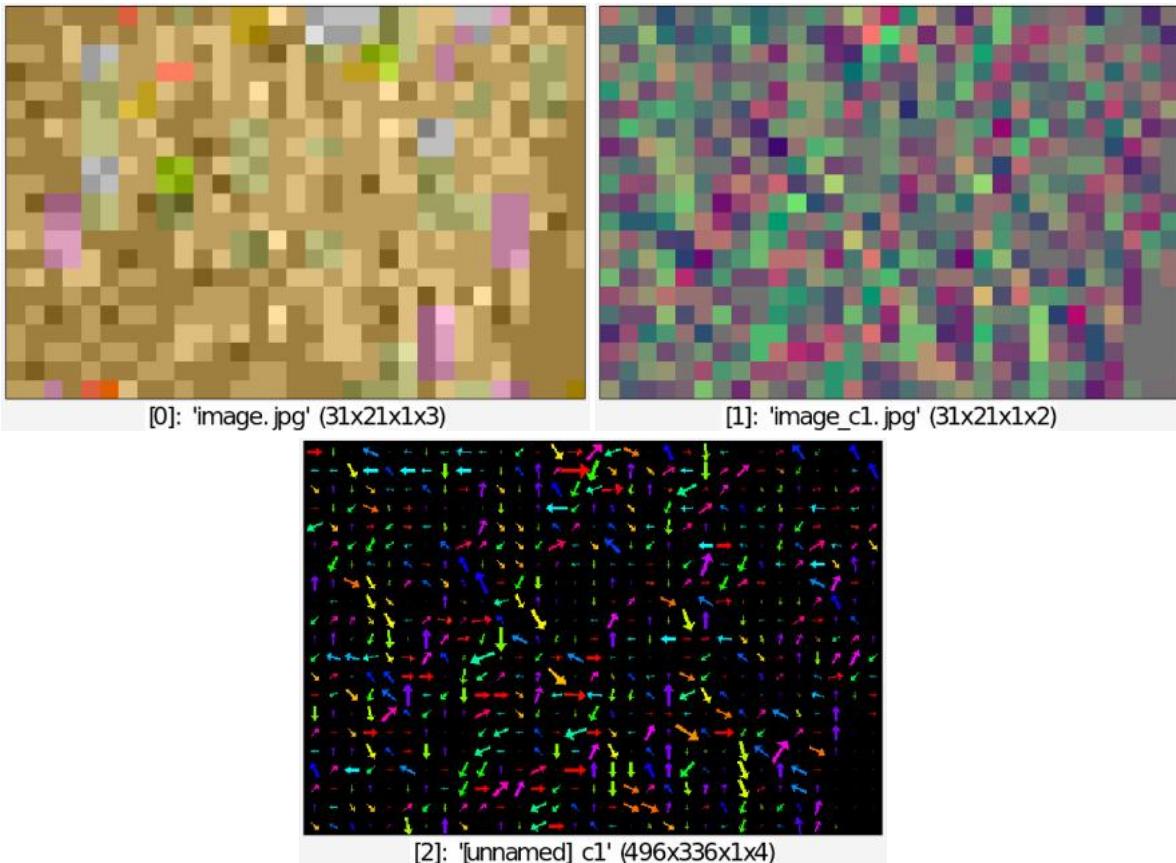
(equivalent to shortcut command `dq`).

## Default values:

`size_factor=16, arrow_size=1.5` and `color_mode=1`.

## Example of use:

```
$ gmic image.jpg +luminance gradient[-1] xy rv[-2,-1] *[-2] -1  
a[-2,-1] c crop 60,10,90,30 +display_quiver[1] ,
```



---

# display\_rgba

## Arguments:

- `_background_RGB_color`

## Description:

Render selected RGBA images over a checkerboard or colored background.

(*equivalent to shortcut command* `drgba`).

## Default values:

`background_RGB_color=undefined` (checkerboard).

## Example of use:

```
$ gmic image.jpg +norm threshold[-1] 40% blur[-1] 3 normalize[-1]
0,255 append c display_rgba
```



# display\_tensors

## Arguments:

- `_size_factor>0, _ellipse_size>=0, _color_mode={ 0=monochrome | 1=grayscale | 2=color }, _outline>=0`

## Description:

Render selected images of tensors as a field of 2D ellipses.

(*equivalent to shortcut command* `dt`).

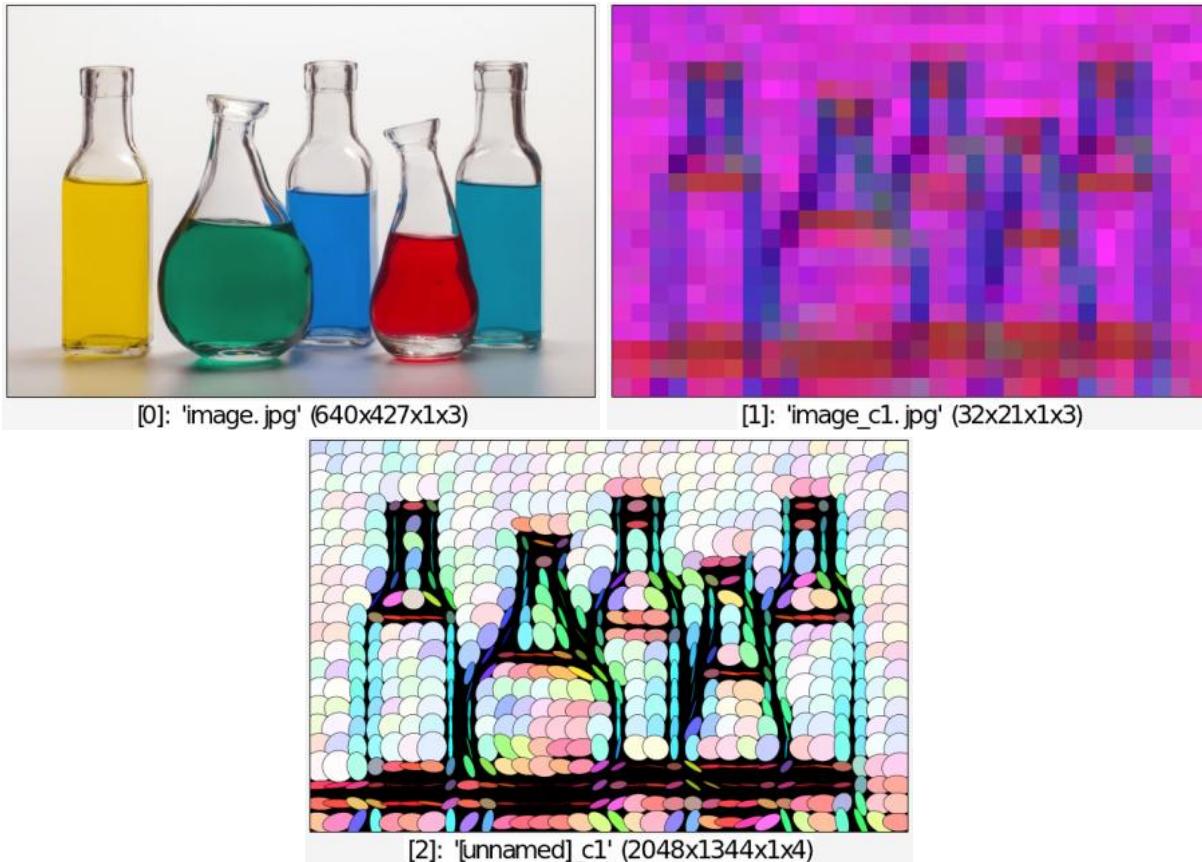
## Default values:

`size_factor=16, ellipse_size=1.5, color_mode=2` and `outline=2`.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg +diffusiontensors 0.1,0.9 resize2dx. 32  
+display_tensors. 64,2
```



---

## display\_warp

### Arguments:

- `_cell_size>0`

### Description:

Render selected 2D warping fields.

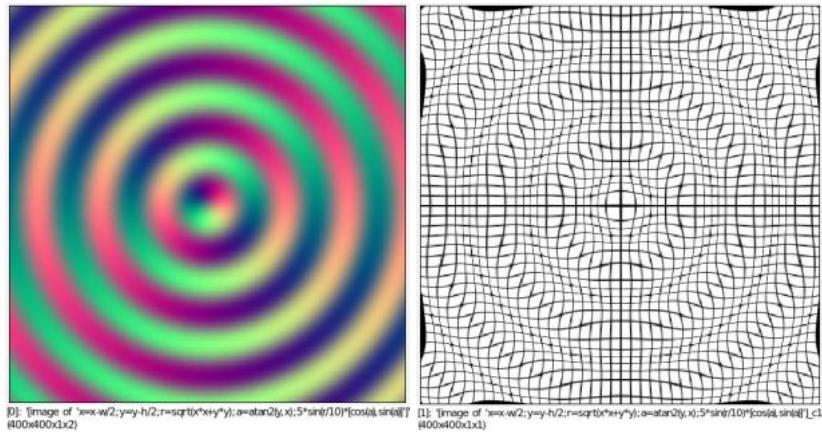
(equivalent to shortcut command `dw`).

### Default values:

`cell_size=15`.

### Example of use:

```
$ gmic 400,400,1,2,'x=x-w/2;y=y-  
h/2;r=sqrt(x*x+y*y);a=atan2(y,x);5*sin(r/10)*[cos(a),sin(a)]'  
+display_warp 10
```



# distance

Built-in command

## Arguments:

- `isovalue[%],_metric` or
- `isovalue[%],[metric],_method`

## Description:

Compute the unsigned distance function to specified isovalue, opt. according to a custom metric.

`metric` can be { `0=chebyshev` | `1=manhattan` | `2=euclidean` | `3=squared-euclidean` }.

`method` can be { `0=fast-marching` | `1=low-connectivity dijkstra` | `2=high-connectivity dijkstra` | `3=1+return path` | `4=2+return path` }.

## Default values:

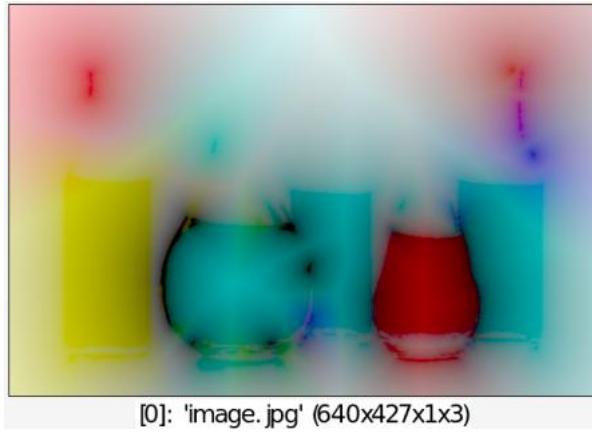
`metric=2` and `method=0`.

This command has a [tutorial page](#).

## Examples of use:

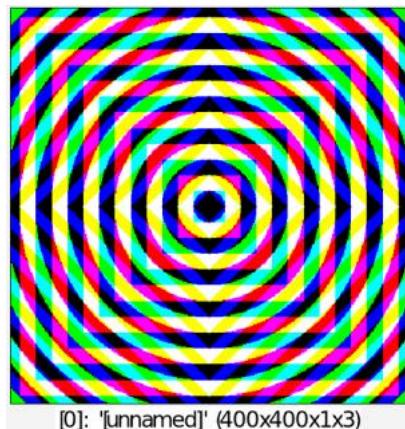
- **Example #1**

```
$ gmic image.jpg threshold 20% distance 0 pow 0.3
```



- **Example #2**

```
$ gmic 400,400 set 1,50%,50% +distance[0] 1,2 +distance[0] 1,1  
distance[0] 1,0 mod 32 threshold 16 append c
```



---

## distribution3d

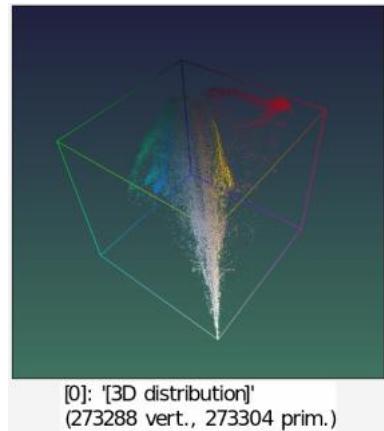
**No arguments**

**Description:**

Get 3D color distribution of selected images.

**Example of use:**

```
$ gmic image.jpg distribution3d colorcube3d primitives3d[-1] 1 add3d
```



[0]: '[3D distribution]'  
(273288 vert., 273304 prim.)

---

## ditheredbw

**No arguments**

**Description:**

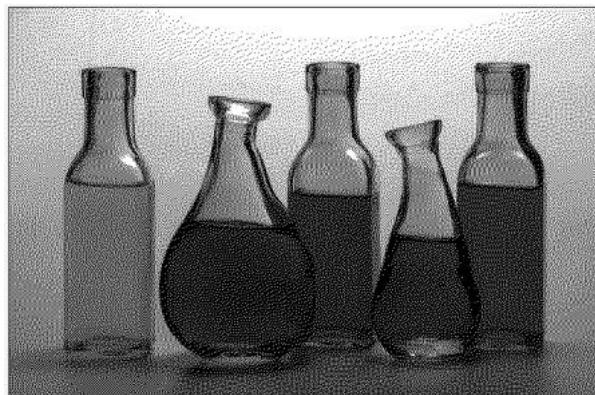
Create dithered B&W version of selected images.

**Example of use:**

```
$ gmic image.jpg +equalize ditheredbw[-1]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x1)

---

## div

Built-in command

**Arguments:**

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

**Description:**

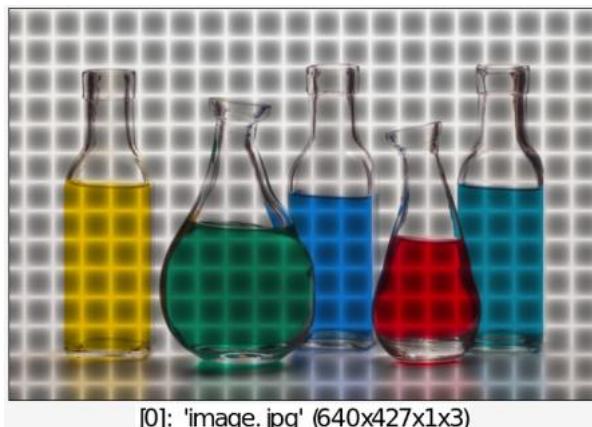
Divide selected images by specified value, image or mathematical expression, or compute the pointwise quotient of selected images.

(equivalent to shortcut command `/`).

## Examples of use:

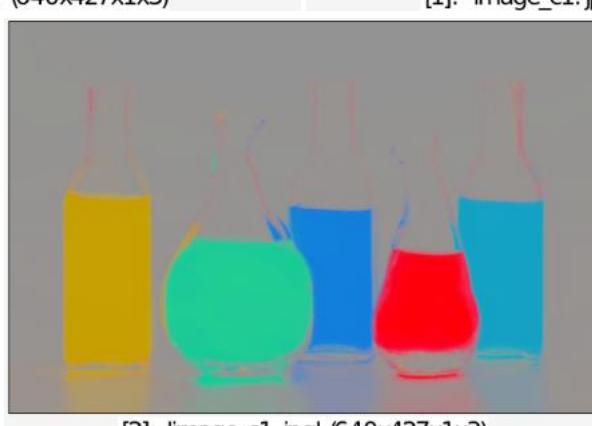
- Example #1

```
$ gmic image.jpg div '1+abs(cos(x/10)*sin(y/10))'
```



- Example #2

```
$ gmic image.jpg +norm add[-1] 1 +div
```



# div3d

Built-in command

## Arguments:

- `factor` or
- `factor_x,factor_y,_factor_z`

## Description:

Scale selected 3D objects isotropically or anisotropically, with the inverse of specified factors.

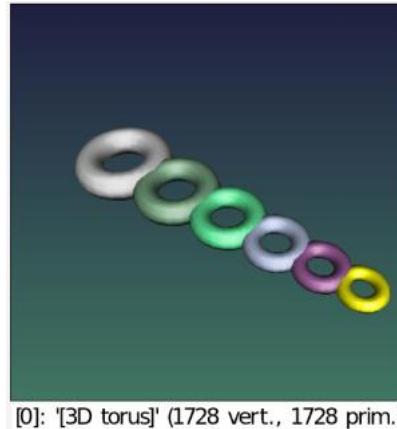
(equivalent to shortcut command `/3d`).

## Default values:

`factor_z=0`.

## Example of use:

```
$ gmic torus3d 5,2 repeat 5 +add3d[-1] 12,0,0 div3d[-1] 1.2  
color3d[-1] ${-rgb} done add3d
```



# div\_complex

## Arguments:

- `[divider_real,divider_imag],_epsilon>=0`

## Description:

Perform division of the selected complex pairs (real1,imag1,...,realN,imagN) of images by specified complex pair of images (divider\_real,divider\_imag).  
In complex pairs, the real image must be always located before the imaginary image in the image list.

## Default values:

`epsilon=1e-8`.

---

# divergence

No arguments

## Description:

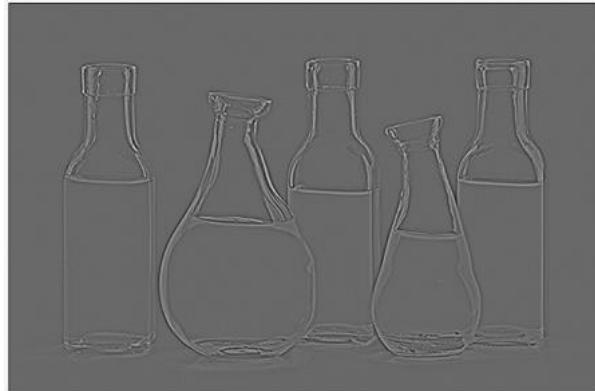
Compute divergence of selected vector fields.

## Example of use:

```
$ gmic image.jpg luminance +gradient append[-2,-1] c divergence[-1]
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)

# do

Built-in command

---

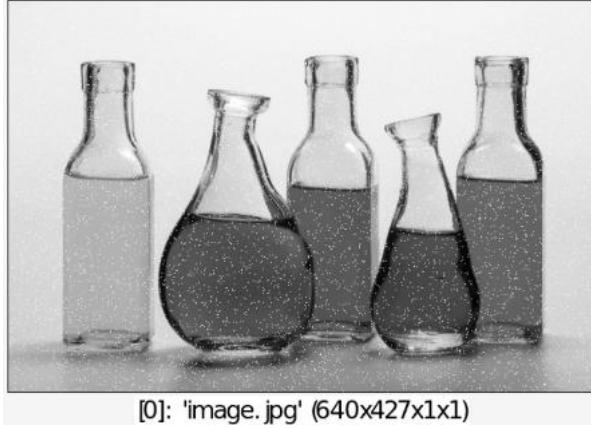
No arguments

## Description:

Start a `do...while` block.

## Example of use:

```
$ gmic image.jpg luminance i={ia+2} do set 255,{u(100)}%,{u(100)}%
while ia<$i
```



---

## dog

### Arguments:

- `_sigma1>=0[%]`, `_sigma2>=0[%]`

### Description:

Compute difference of gaussian on selected images.

### Default values:

`sigma1=2%` and `sigma2=3%`.

### Example of use:

```
$ gmic image.jpg dog 2,3
```



---

## done

Built-in command

No arguments

### Description:

End a `repeat/for...done` block, and go to associated `repeat/for` position, if iterations remain.

---

## double3d

Built-in command

### Arguments:

- `_is_double_sided={ 0 | 1 }`

### Description:

Enable/disable double-sided mode for 3D rendering.

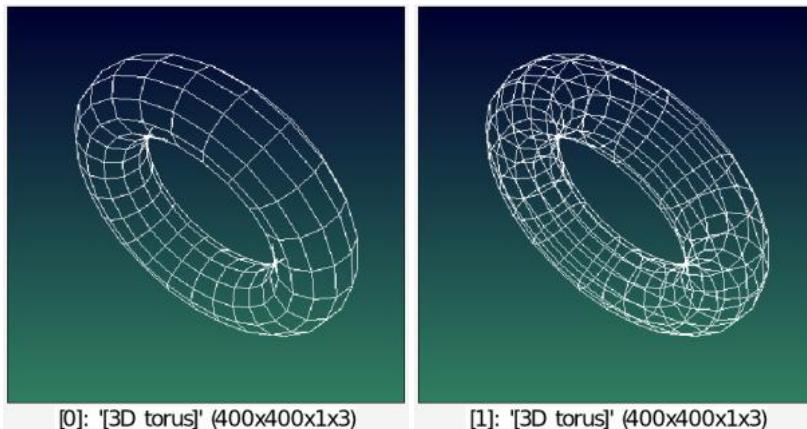
(equivalent to shortcut command `db3d`).

### Default values:

`is_double_sided=1`.

### Example of use:

```
$ gmic mode3d 1 repeat 2 torus3d 100,30 rotate3d[-1] 1,1,0,60  
double3d $> snapshot3d[-1] 400 done
```



## draw\_whirl

### Arguments:

- `_amplitude>=0`

### Description:

Apply whirl drawing effect on selected images.

### Default values:

amplitude=100 .

## Example of use:

```
$ gmic image.jpg draw_whirl ,
```



# drawing

## Arguments:

- `_amplitude>=0`

## Description:

Apply drawing effect on selected images.

## **Default values:**

amplitude=200 .

## Example of use:

```
$ gmic image.jpg +drawing ,
```



[0]: 'image.jpg' (640x427x1x3)



```
[1]: 'image c1.jpg,1' (640x427x1x3)
```

# drop\_shadow

## Arguments:

- `_offset_x[%],_offset_y[%],_smoothness[%]>=0,0<=_curvature<=1,_expand_size={ 0 | 1 }`

## Description:

Drop shadow behind selected images.

## Default values:

`offset_x=20`, `offset_y=offset_x`, `smoothness=5`, `curvature=0` and `expand_size=1`.

## Example of use:

```
$ gmic image.jpg drop_shadow 10,20,5,0.5 expand_xy 20,0 display_rgba
```



---

# echo

Built-in command

## Arguments:

- `message`

## Description:

Output specified message on the error output.

(equivalent to shortcut command `e`).

Command selection (if any) stands for displayed call stack subset instead of image indices. When invoked with a `+` prefix (i.e. `+echo`), the command output its message on stdout rather than stderr.

---

# echo\_file

## Arguments:

- `filename, message`

## Description:

Output specified message, appending it to specified output file.

(similar to `echo` for specified output file stream).

---

# edges

## Arguments:

- `_threshold[%]>=0`

## Description:

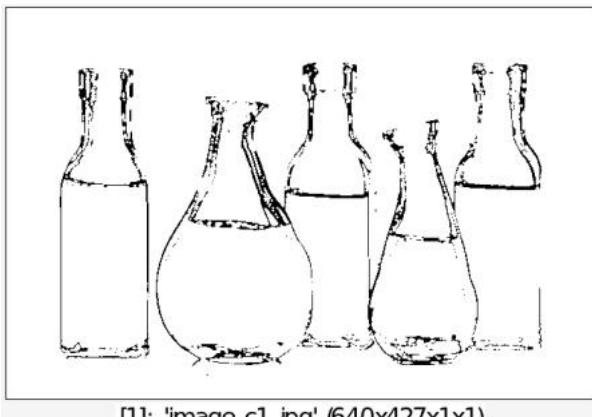
Estimate contours of selected images.

## Default values:

`edges=15%`

## Example of use:

```
$ gmic image.jpg +edges 15%
```



# eigen

Built-in command

No arguments

## Description:

Compute the eigenvalues and eigenvectors of selected symmetric matrices or matrix fields.

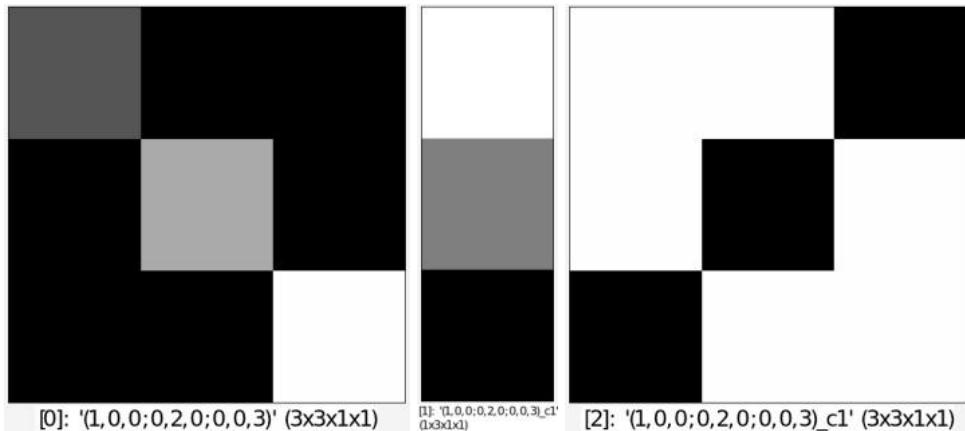
If one selected image has 3 or 6 channels, it is regarded as a field of 2x2 or 3x3 symmetric matrices, whose eigen elements are computed at each point of the field.

This command has a [tutorial page](#).

## Examples of use:

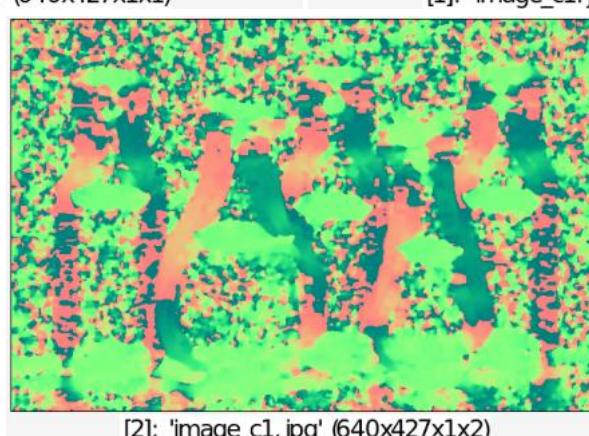
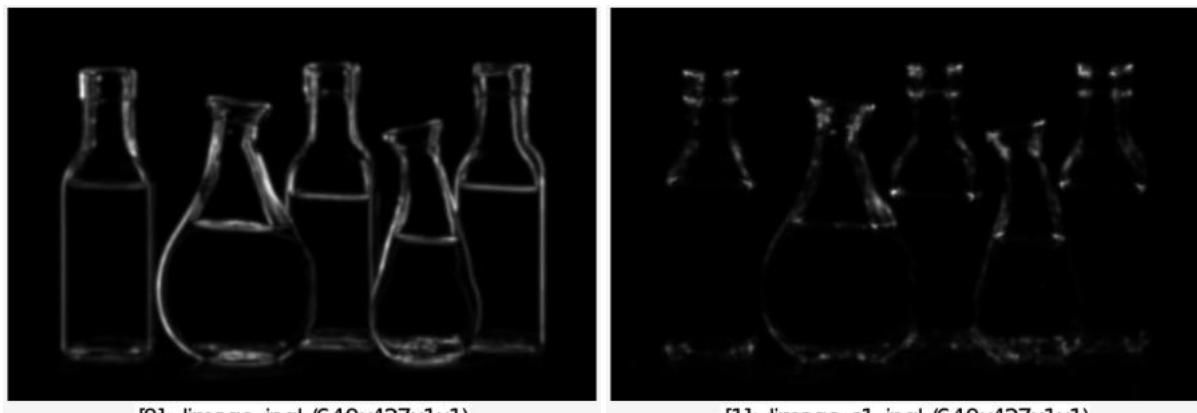
- **Example #1**

```
$ gmic (1,0,0;0,2,0;0,0,3) +eigen
```



- **Example #2**

```
$ gmic image.jpg structuretensors blur 2 eigen split[0] c
```



---

# eigen2tensor

## No arguments

### Description:

Recompose selected pairs of eigenvalues/eigenvectors as 2x2 or 3x3 tensor fields.

This command has a [tutorial page](#).

---

# elevate

## Arguments:

- `_depth,_is_plain={ 0 | 1 },_is_colored={ 0 | 1 }`

### Description:

Elevate selected 2D images into 3D volumes.

## Default values:

`depth=64`, `is_plain=1` and `is_colored=1`.

---

# elevation3d

## Arguments:

- `{ z-factor | [elevation_map] | 'formula' },base_height={ -1 | >=0 }` or
- `(no arg)`

### Description:

Generate 3D elevation of selected images, opt. with a specified elevation map.

When invoked with (no arg) or `z-factor`, the elevation map is computed as the pointwise L2 norm of the pixel values. Otherwise, the elevation map is taken from the specified image or formula.

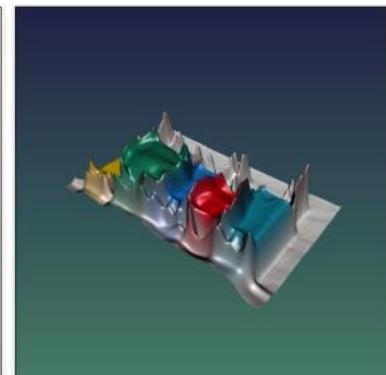
## Examples of use:

- **Example #1**

```
$ gmic image.jpg +blur 5 elevation3d. 0.75
```



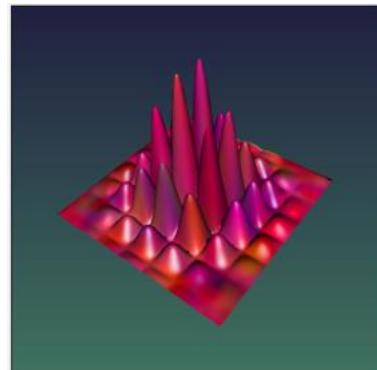
[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg'  
(273280 vert., 272214 prim.)

## • Example #2

```
$ gmic 128,128,1,3,u(255) plasma 10,3 blur 4 sharpen 10000 n 0,255  
elevation3d[-1] 'X=(x-64)/6;Y=(y-64)/6;-100*exp(-  
(X^2+Y^2)/30)*abs(cos(X)*sin(Y))'
```



[0]: '[image of 'u(255)']'  
(16384 vert., 16129 prim.)

# elif

Built-in command

## Arguments:

- `condition`

## Description:

Start a `elif...[else]...fi` block if previous `if` was not verified

and test if specified condition holds

`condition` is a mathematical expression, whose evaluation is interpreted as `{ 0=false | other=true }`.

This command has a [tutorial page](#).

# ellipse

Built-in command

## Arguments:

- `x[%],y[%],R[%],r[%],_angle,_opacity,_pattern,_color1,...`

## Description:

Draw specified colored ellipse on selected images.

A radius of `100%` stands for `sqrt(width^2+height^2)`.

`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 values:

`opacity=1`, `pattern=(undefined)` and `color1=0`.

## Example of use:

```
$ gmic image.jpg repeat 300 ellipse {u(100)}%,{u(100)}%,{u(30)},  
{u(30)},{u(180)},0.3,${-rgb} done ellipse 50%,50%,100,100,0,0.7,255
```



[0]: 'image.jpg' (640x427x1x3)

---

# ellipsonism

## Arguments:

- `_R>0[%],_r>0[%],_smoothness>=0[%],_opacity,_outline>0,_density>0`

## Description:

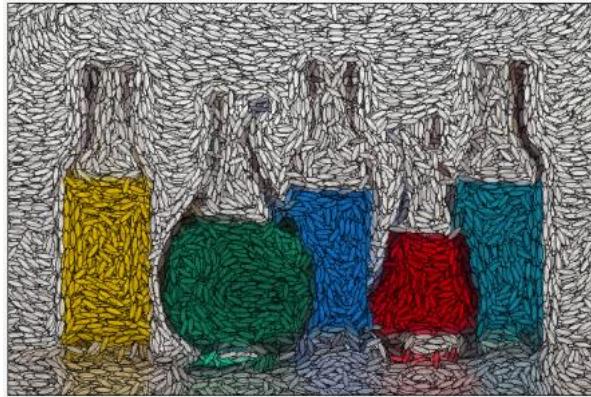
Apply ellipsonism filter to selected images.

## Default values:

`R=10`, `r=3`, `smoothness=1%`, `opacity=0.7`, `outline=8` and `density=0.6`.

## Example of use:

```
$ gmic image.jpg ellipsoidism ,
```



[0]: 'image.jpg' (640x427x1x3)

---

## else

Built-in command

No arguments

### Description:

Execute following commands if previous `if` or `elif` conditions failed.

This command has a [tutorial page](#).

---

## empty3d

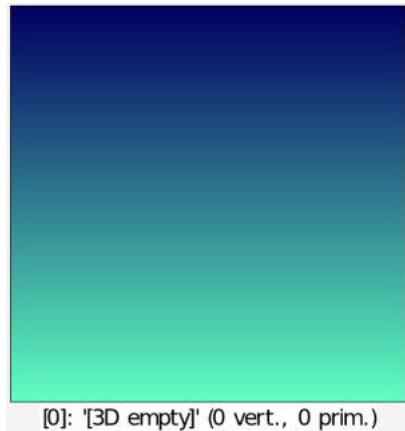
No arguments

### Description:

Input empty 3D object.

## Example of use:

```
$ gmic empty3d
```



---

## endian

Built-in command

### Arguments:

- `_datatype`

### Description:

Reverse data endianness of selected images, eventually considering the pixel being of the specified datatype.

`datatype` can be `{ bool | uchar | char | ushort | short | uint | int | uint64 | int64 | float | double }`.

This command does nothing for `bool`, `uchar` and `char` datatypes.

---

## endlocal

Built-in command

### No arguments

### Description:

End a `local...endlocal` block.

(equivalent to shortcut command `endl`).

---

## eq

Built-in command

### Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

## Description:

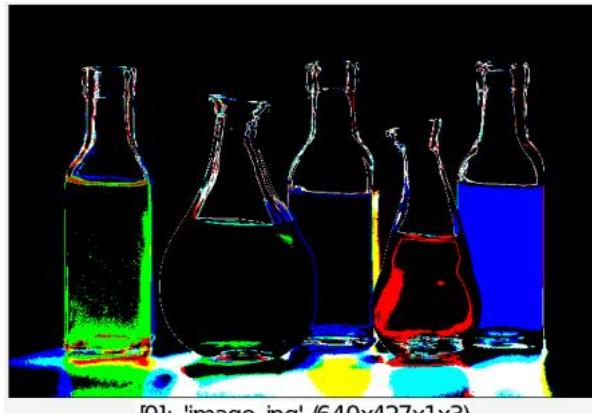
Compute the boolean equality of selected images with specified value, image or mathematical expression, or compute the boolean equality of selected images.

(*equivalent to shortcut command* `==`).

## Examples of use:

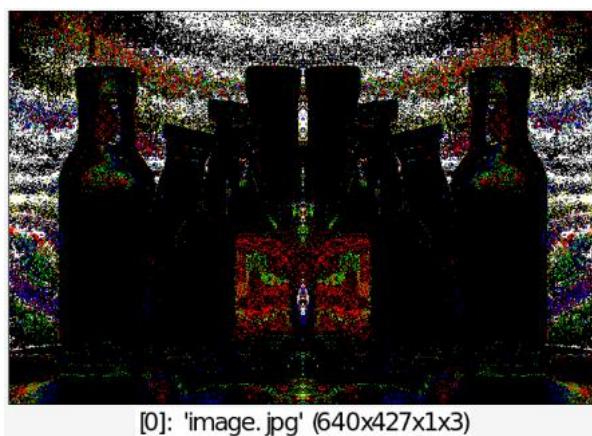
- Example #1

```
$ gmic image.jpg round 40 eq {round(ia,40)}
```



- Example #2

```
$ gmic image.jpg +mirror x eq
```



---

## equalize

Built-in command

## Arguments:

- `_nb_levels>0[%]`, `_value_min[%]`, `_value_max[%]`

## Description:

Equalize histograms of selected images.

If value range is specified, the equalization is done only for pixels in the specified value range.

## Default values:

`nb_levels=256`, `value_min=0%` and `value_max=100%`.

## Examples of use:

- Example #1

```
$ gmic image.jpg +equalize
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

- Example #2

```
$ gmic image.jpg +equalize 4,0,128
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

## equirectangular2nadirzenith

No arguments

## Description:

Transform selected equirectangular images to nadir/zenith rectilinear projections.

# erf

Built-in command

No arguments

## Description:

Compute the pointwise error function of selected images.

## Examples of use:

- Example #1

```
$ gmic image.jpg +normalize 0,2 erf[-1]
```



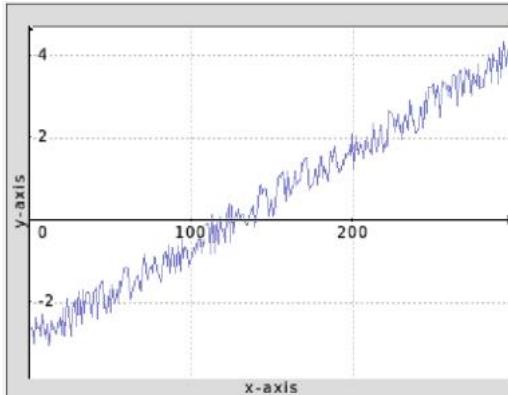
[0]: 'image.jpg' (640x427x1x3)



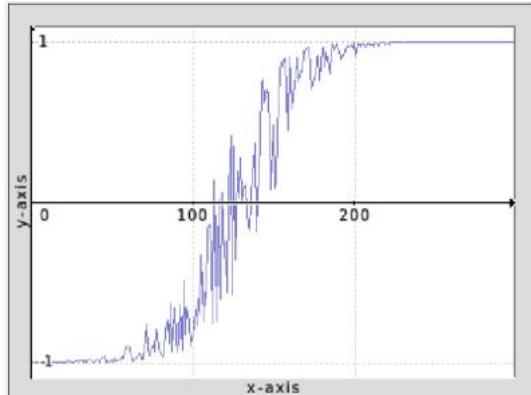
[1]: 'image\_c1.jpg' (640x427x1x3)

- Example #2

```
$ gmic 300,1,1,1,'7*x/w-3.5+u' +erf display_graph 400,300
```



[0]: '[image of "7\*x/w-3.5+u"]' (400x300x1x3)



[1]: '[image of "7\*x/w-3\_c1.5+u"]' (400x300x1x3)

# erode

Built-in command

## Arguments:

- `size>=0` or
- `size_x>=0, size_y>=0, size_z>=0` or
- `[kernel], _boundary_conditions, _is_real={ 0=binary-mode | 1=real-mode }`

## Description:

Erode selected images by a rectangular or the specified structuring element.

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

## Default values:

`size_z=1`, `boundary_conditions=1` and `is_real=0`.

## Example of use:

```
$ gmic image.jpg +erode 10
```



---

## erode\_circ

## Arguments:

- `_size>=0, _boundary_conditions, _is_normalized={ 0 | 1 }`

## Description:

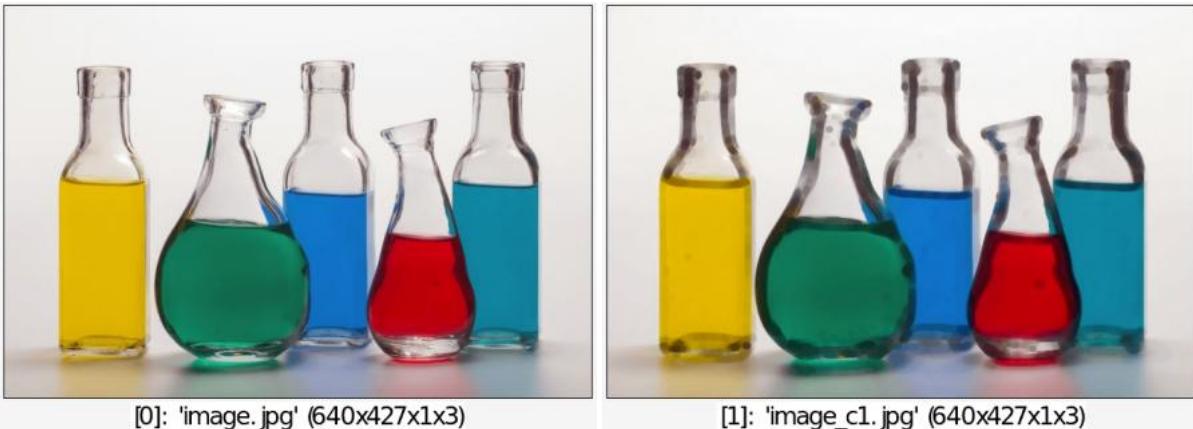
Apply circular erosion of selected images by specified size.

## Default values:

`boundary_conditions=1` and `is_normalized=0`.

## Example of use:

```
$ gmic image.jpg +erode_circ 7
```



## erode\_oct

### Arguments:

- `_size>=0, _boundary_conditions, _is_normalized={ 0 | 1 }`

### Description:

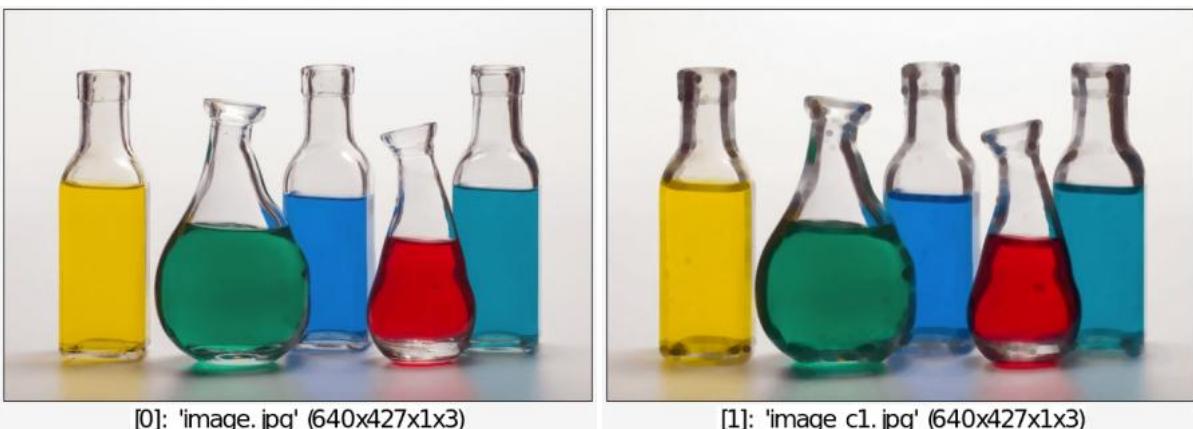
Apply octagonal erosion of selected images by specified size.

### Default values:

`boundary_conditions=1` and `is_normalized=0`.

### Example of use:

```
$ gmic image.jpg +erode_oct 7
```



## erode\_threshold

### Arguments:

- `size_x>=1,size_y>=1,size_z>=1,_threshold>=0,_boundary_conditions`

## Description:

Erode selected images in the (X,Y,Z,I) space.

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

## Default values:

`size_y=size_x`, `size_z=1`, `threshold=255` and `boundary_conditions=1`.

---

# error

Built-in command

## Arguments:

- `message`

## Description:

Print specified error message on the standard error (stderr) and exit interpreter, except

if error is caught by a `onfail` command.

Command selection (if any) stands for displayed call stack subset instead of image indices.

---

# euclidean2polar

## Arguments:

- `_center_x[%],_center_y[%],_stretch_factor>0,_boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

## Description:

Apply euclidean to polar transform on selected images.

## Default values:

`center_x=center_y=50%`, `stretch_factor=1` and `boundary_conditions=1`.

## Example of use:

```
$ gmic image.jpg +euclidean2polar ,
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

## eval

Built-in command

### Arguments:

- `expression`

### Description:

Evaluate specified math expression.

- If no command selection is specified, the expression is evaluated once and its result is set to status.
- If command selection is specified, the evaluation is looped over selected images. Status is not modified.

(in this latter case, `eval` is similar to `fill` without assigning the image values).

## exec

Built-in command

### Arguments:

- `_is_verbose={ 0 | 1 }, "command"`

### Description:

Execute external command using a system call.

The status value is then set to the error code returned by the system call.

If `is_verbose=1`, the executed command is allowed to output on stdout/stderr.

(equivalent to shortcut command `x`).

### Default values:

`is_verbose=1`.

# exec\_out

## Arguments:

- `_mode, "command"`

## Description:

Execute external command using a system call, and return resulting `stdout` and/or `stderr`.

`mode` can be `{ 0=stdout | 1=stderr | 2=stdout+stderr }`.

---

# exp

Built-in command

## No arguments

## Description:

Compute the pointwise exponential of selected images.

## Examples of use:

- Example #1

```
$ gmic image.jpg +normalize 0,2 exp[-1]
```



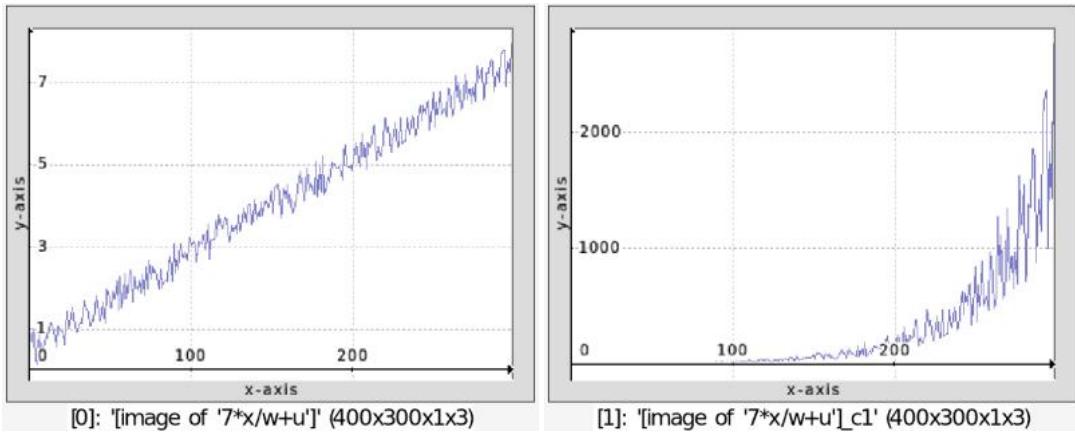
[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

- Example #2

```
$ gmic 300,1,1,1,'7*x/w+u' +exp display_graph 400,300
```



## expand\_x

### Arguments:

- `size_x>=0, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

### Description:

Expand selected images along the x-axis.

### Default values:

`boundary_conditions=1`.

### Example of use:

```
$ gmic image.jpg expand_x 30,0
```



## expand\_xy

### Arguments:

- `size>=0, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

## Description:

Expand selected images along the xy-axes.

## Default values:

`boundary_conditions=1`.

## Example of use:

```
$ gmic image.jpg expand_xy 30,0
```



[0]: 'image.jpg' (700x487x1x3)

---

## expand\_xyz

### Arguments:

- `size>=0, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

## Description:

Expand selected images along the xyz-axes.

## Default values:

`boundary_conditions=1`.

---

## expand\_y

### Arguments:

- `size_y>=0, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

## Description:

Expand selected images along the y-axis.

## Default values:

`boundary_conditions=1`.

## Example of use:

```
$ gmic image.jpg expand_y 30,0
```



[0]: 'image.jpg' (640x487x1x3)

---

## expand\_z

### Arguments:

- `size_z>=0, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

## Description:

Expand selected images along the z-axis.

## Default values:

`boundary_conditions=1`.

---

## extract

### Arguments:

- "condition", \_output\_type={ 0=xyzc-coordinates | 1=xyz-coordinates | 2=scalar-values | 3=vector-values }

## Description:

Extract a list of coordinates or values from selected image, where

specified mathematical condition holds.

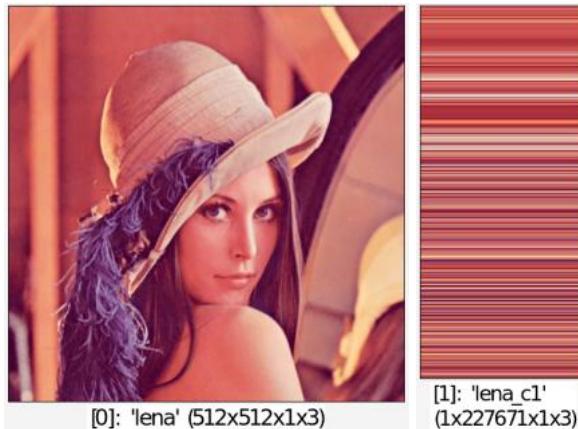
For N coordinates matching, result is a 1xNx1x4 image.

## Default values:

`output_type=0`.

## Example of use:

```
$ gmic sp lena +extract "norm(I)>128",3
```



## extract\_region

### Arguments:

- `[label_image], _extract_xyz_coordinates={ 0 | 1 }, _label_1,...,_label_M`

## Description:

Extract all pixels of selected images whose corresponding label in `[label_image]` is equal to `label_m`,

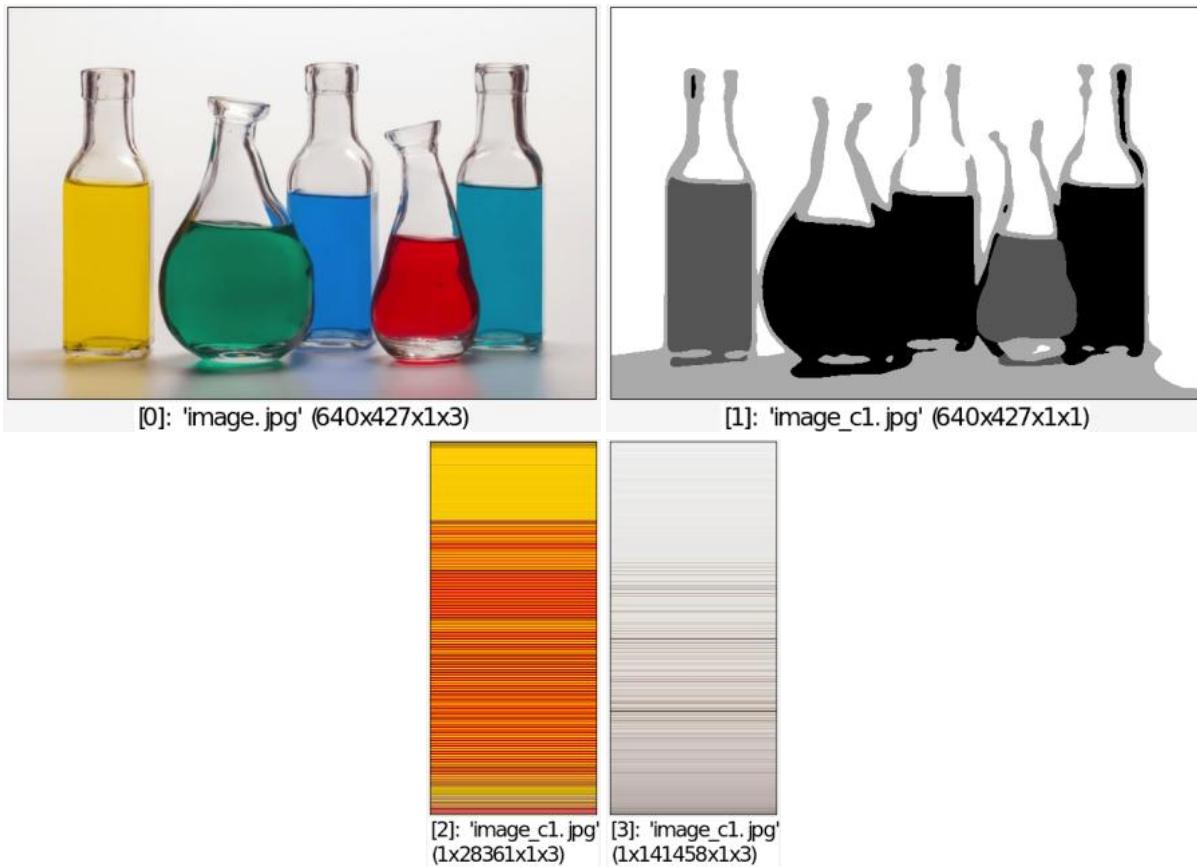
and output them as M column images.

## Default values:

`extract_xyz_coordinates=0`.

## Example of use:

```
$ gmic image.jpg +blur 3 quantize. 4,0 +extract_region[0] [1],0,1,3
```



## extrude3d

### Arguments:

- `_depth>0, _resolution>0, _smoothness[%]>=0`

### Description:

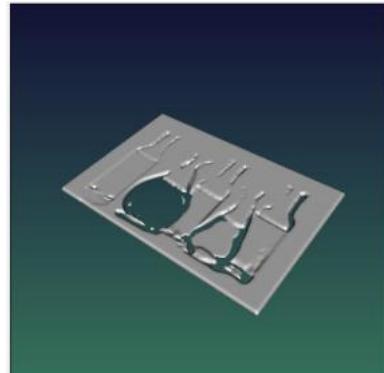
Generate extruded 3D object from selected binary XY-profiles.

### Default values:

`depth=16`, `resolution=1024` and `smoothness=0.5%`.

### Example of use:

```
$ gmic image.jpg threshold 50% extrude3d 16
```



[0]: 'image.jpg'  
(486554 vert., -2.2320184707641602 prim.)

---

## fact

### Arguments:

- `value`

### Description:

Return the factorial of the specified value.

---

## fade\_diamond

### Arguments:

- `0<=_start<=100,0<=_end<=100`

### Description:

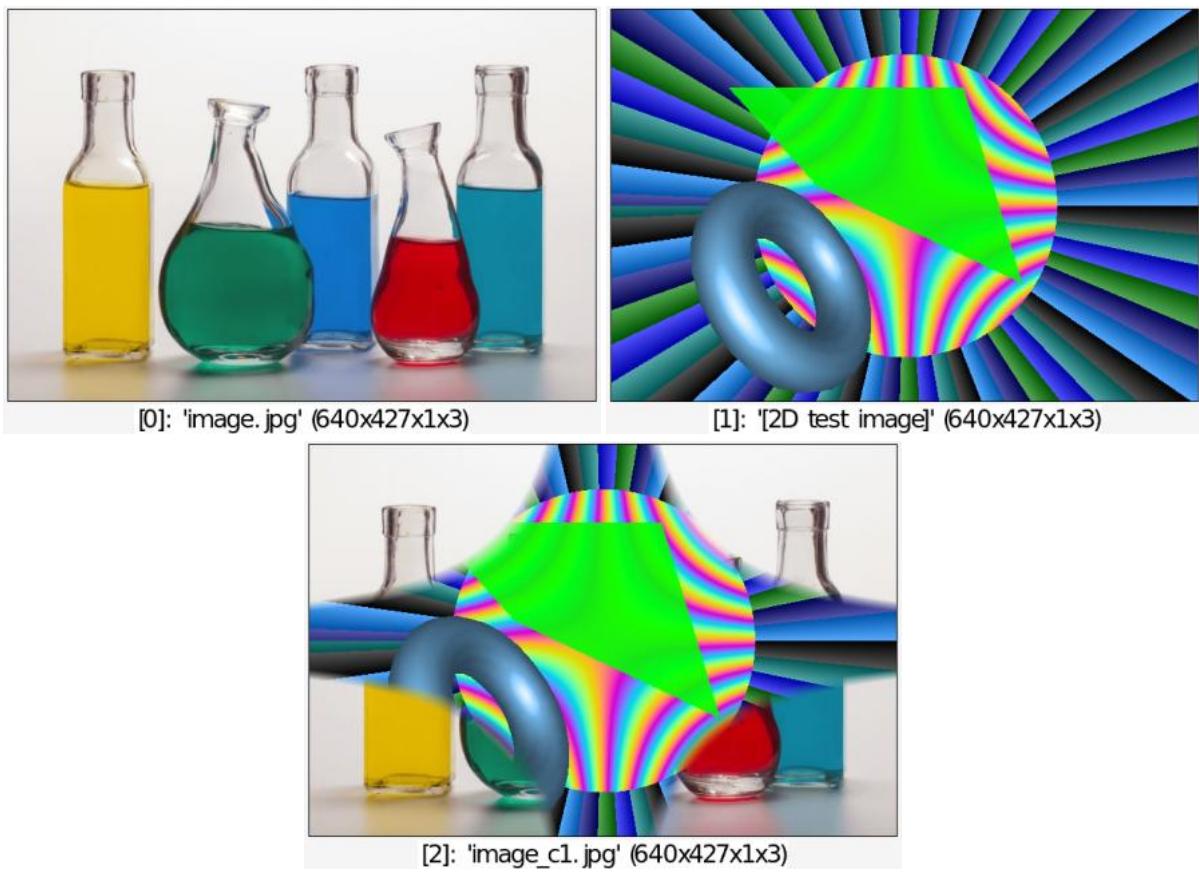
Create diamond fading from selected images.

### Default values:

`start=80` and `end=90`.

### Example of use:

```
$ gmic image.jpg testimage2d {w},{h} +fade_diamond 80,85
```



## fade\_files

### Arguments:

- `"filename_pattern", _nb_inner_frames>0, _first_frame>=0, _last_frame={ >=0 | -1=last }, _frame_step>=1, _output_filename`

### Description:

Generate a temporal fading from specified input image files, in a streamed way.

If a display window is opened, rendered frames are displayed in it during processing.  
The output filename may have extension `avi` or `mp4` (saved as a video), or any other usual image file extension (saved as a sequence of images).

### Default values:

`nb_inner_frames=10, first_frame=0, last_frame=-1, frame_step=1` and `output_filename=(undefined)`.

## fade\_linear

### Arguments:

- `_angle, 0<=_start<=100, 0<=_end<=100`

## Description:

Create linear fading from selected images.

## Default values:

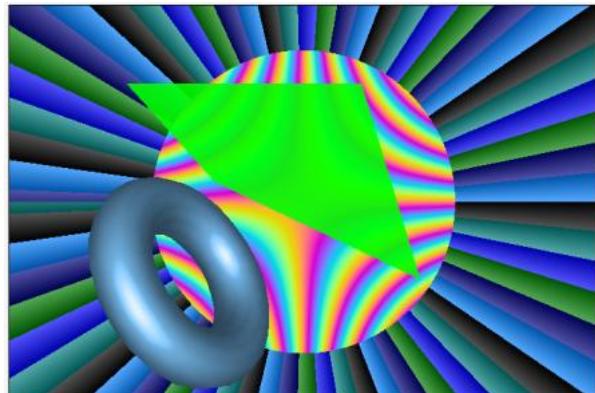
`angle=45`, `start=30` and `end=70`.

## Example of use:

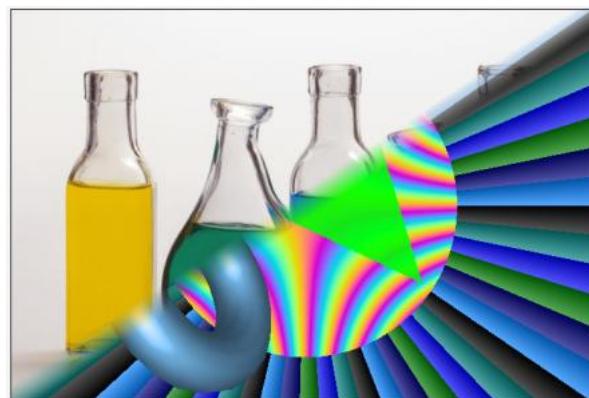
```
$ gmic image.jpg testimage2d {w},{h} +fade_linear 45,48,52
```



[0]: 'image.jpg' (640x427x1x3)



[1]: '[2D test image]' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)

## fade\_radial

### Arguments:

- `0<=_start<=100, 0<=_end<=100`

## Description:

Create radial fading from selected images.

## Default values:

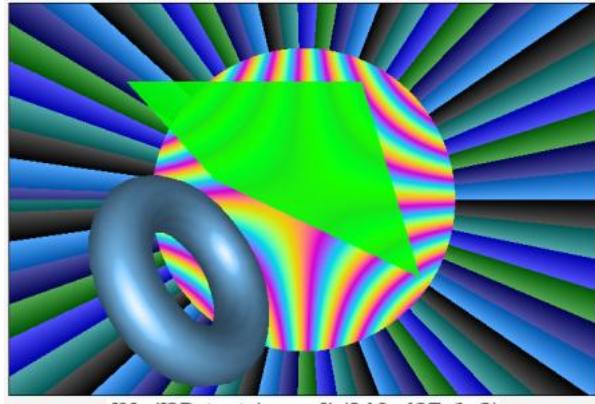
`start=30` and `end=70`.

## Example of use:

```
$ gmic image.jpg testimage2d {w},{h} +fade_radial 30,70
```



[0]: 'image.jpg' (640x427x1x3)



[1]: '[2D test image]' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)

---

## fade\_video

### Arguments:

- `video_filename,_nb_inner_frames>0,_first_frame>=0,_last_frame={ >=0 | -1=last },_frame_step>=1,_output_filename`

### Description:

Create a temporal fading sequence from specified input video file, in a streamed way.

If a display window is opened, rendered frames are displayed in it during processing.  
This command requires features from the OpenCV library (not enabled in G'MIC by default).

### Default values:

`nb_inner_frames=10, first_frame=0, last_frame=-1, frame_step=1` and  
`output_filename=(undefined)`.

---

## fade\_x

## Arguments:

- `0<=_start<=100,0<=_end<=100`

## Description:

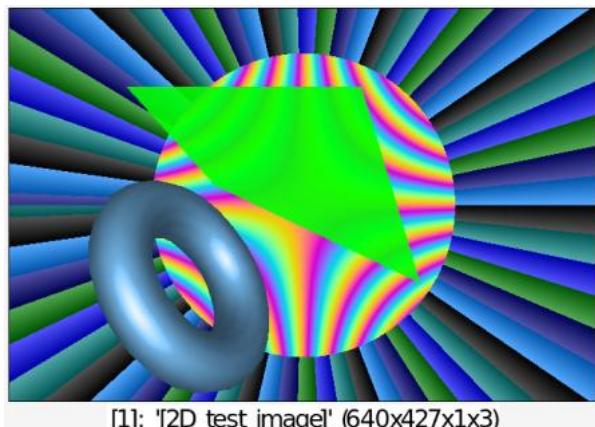
Create horizontal fading from selected images.

## Default values:

`start=30` and `end=70`.

## Example of use:

```
$ gmic image.jpg testimage2d {w},{h} +fade_x 30,70
```



---

## fade\_y

## Arguments:

- `0<=_start<=100,0<=_end<=100`

## Description:

Create vertical fading from selected images.

## Default values:

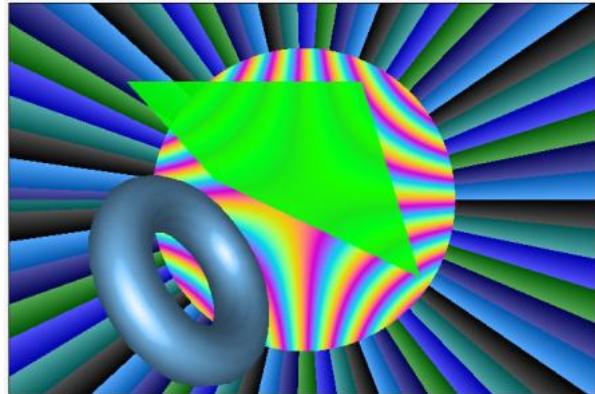
`start=30` and `end=70`.

## Example of use:

```
$ gmic image.jpg testimage2d {w},{h} +fade_y 30,70
```



[0]: 'image.jpg' (640x427x1x3)



[1]: '[2D test image]' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)

## fade\_z

### Arguments:

- `0<=_start<=100, 0<=_end<=100`

### Description:

Create transversal fading from selected images.

## Default values:

`start=30` and `end=70`.

## fft

Built-in command

## Arguments:

- `_ { x | y | z } ... { x | y | z }`

## Description:

Compute the direct fourier transform (real and imaginary parts) of selected images, optionally along the specified axes only.

## See also:

[ifft](#).

This command has a [tutorial page](#).

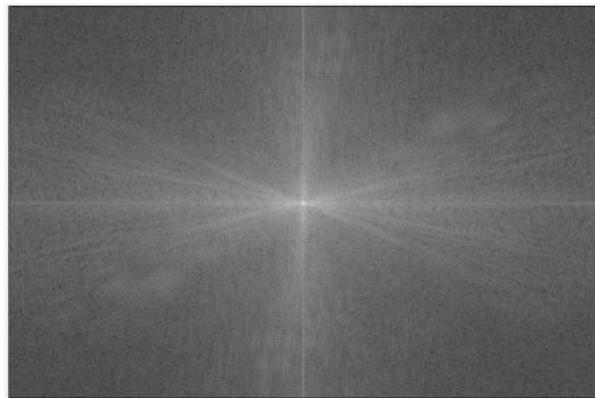
## Examples of use:

- **Example #1**

```
$ gmic image.jpg luminance +fft append[-2,-1] c norm[-1] log[-1]
shift[-1] 50%,50%,0,0,2
```



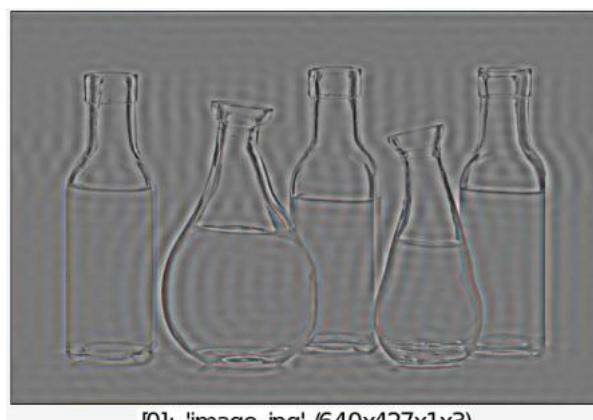
[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)

- **Example #2**

```
$ gmic image.jpg w2={int(w/2)} h2={int(h/2)} fft shift $w2,$h2,0,0,2
ellipse $w2,$h2,30,30,0,1,0 shift -$w2,-$h2,0,0,2 ifft remove[-1]
```



[0]: 'image.jpg' (640x427x1x3)

---

# fftpolar

No arguments

## Description:

Compute fourier transform of selected images, as centered magnitude/phase images.

## Example of use:

```
$ gmic image.jpg fftpolar ellipse 50%,50%,10,10,0,1,0 ifftpolar
```



[0]: 'image.jpg' (640x427x1x3)

---

# fi

Built-in command

No arguments

## Description:

End a `if...[elif]...[else]...fi` block.

(equivalent to shortcut command `fi`).

This command has a [tutorial page](#).

---

# fibonacci

## Arguments:

- `N>=0`

## Description:

Return the Nth number of the Fibonacci sequence.

## Example of use:

```
$ gmic echo ${"fibonacci 10"}
```

```
[gmic]-0./ Start G'MIC interpreter.  
[gmic]-0./ 55  
[gmic]-0./ End G'MIC interpreter.
```

---

## file\_mv

### Arguments:

- `filename_src, filename_dest`

### Description:

Rename or move a file from a location \$1 to another location \$2.

---

## file\_rand

### No arguments

### Description:

Return a random filename for storing temporary data.

---

## filename

### Arguments:

- `filename,_number1,_number2,...,_numberN`

### Description:

Return a filename numbered with specified indices.

---

Built-in command

# files

## Arguments:

- `_mode, path`

## Description:

Return the list of files and/or subfolders from specified path.

`path` can be eventually a matching pattern.

`mode` can be `{ 0=files only | 1=folders only | 2=files + folders }`.

Add `3` to `mode` to return full paths instead of filenames only.

## Default values:

`mode=5`.

---

# files2img

## Arguments:

- `_mode, path`

## Description:

Insert a new image where each vector-valued pixel is a string encoding the filenames returned by command `files`.

Useful to manage list of filenames containing characters that have a special meaning in the G'MIC language, such as spaces or commas.

---

# files2video

## Arguments:

- `"filename_pattern", _output_filename, _fps>0, _codec`

## Description:

Convert several files into a single video file.

## Default values:

`output_filename=output.mp4`, `fps=25` and `codec=mp4v`.

---

## Arguments:

- `value1, value2, ...` or
- `[image]` or
- `'formula'`

## Description:

Fill selected images with values read from the specified value list, existing image

or mathematical expression. Single quotes may be omitted in `formula`.

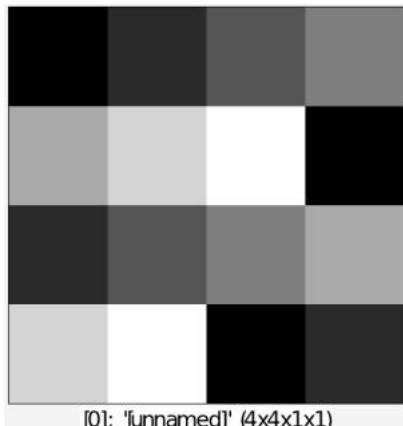
(equivalent to shortcut command `f`).

This command has a [tutorial page](#).

## Examples of use:

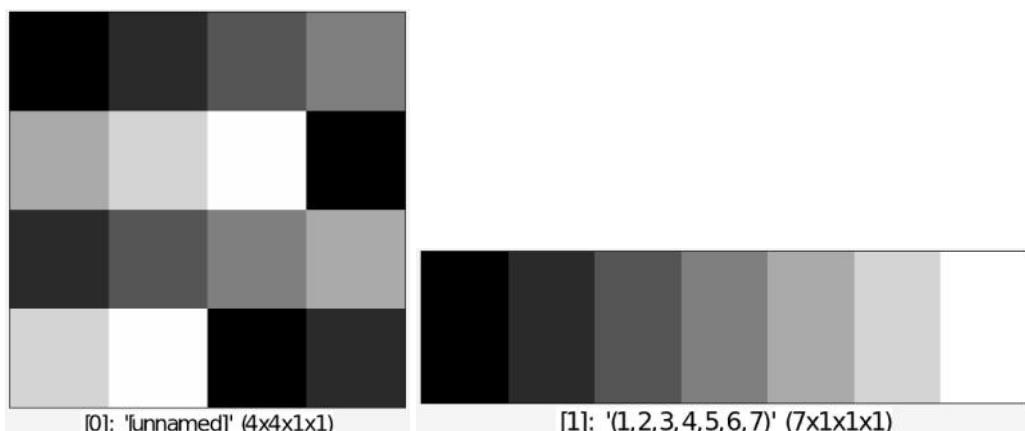
- **Example #1**

```
$ gmic 4,4 fill 1,2,3,4,5,6,7
```



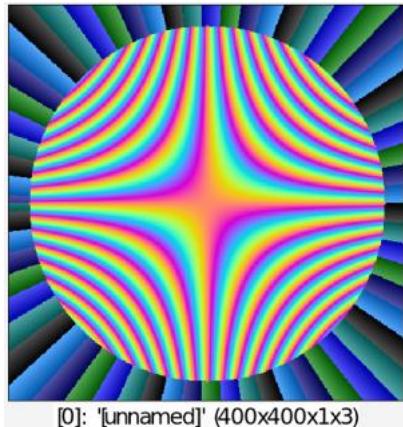
- **Example #2**

```
$ gmic 4,4 (1,2,3,4,5,6,7) fill[-2] [-1]
```



- **Example #3**

```
$ gmic 400,400,1,3 fill "X=x-w/2; Y=y-h/2; R=sqrt(X^2+Y^2);  
a=atan2(Y,X); if(R<=180,255*abs(cos(c+200*(x/w-0.5)*(y/h-0.5))),850*  
(a%(0.1*(c+1))))"
```



---

## fill\_color

### Arguments:

- `col1,...,colN`

### Description:

Fill selected images with specified color.

(equivalent to shortcut command `fc`).

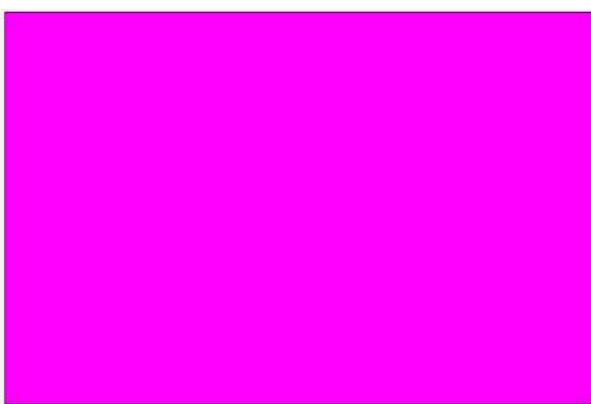
This command has a [tutorial page](#).

### Example of use:

```
$ gmic image.jpg +fill_color 255,0,255
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

# fire\_edges

## Arguments:

- `_edges>=0, 0<=_attenuation<=1, _smoothness>=0, _threshold>=0, _nb_frames>0, _star`

## Description:

Generate fire effect from edges of selected images.

## Default values:

`edges=0.7, attenuation=0.25, smoothness=0.5, threshold=25, nb_frames=1, starting_frame=20` and `frame_skip=0`.

## Example of use:

```
$ gmic image.jpg fire_edges ,
```



---

# fisheye

## Arguments:

- `_center_x, _center_y, 0<=_radius<=100, _amplitude>=0`

## Description:

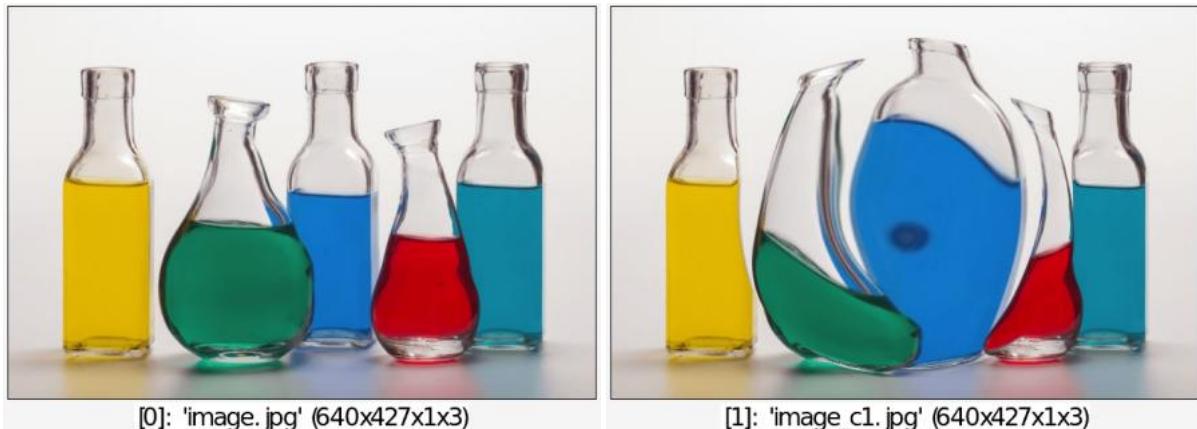
Apply fish-eye deformation on selected images.

## Default values:

`x=y=50, radius=50` and `amplitude=1.2`.

## Example of use:

```
$ gmic image.jpg +fisheye ,
```



---

## fitratio\_wh

### Arguments:

- `min_width, min_height, ratio_wh`

### Description:

Return a 2D size `width, height` which is bigger than `min_width, min_height` and has the specified w/h ratio.

---

## fitscreen

### Arguments:

- `width, height, _depth, _minimal_size[%], _maximal_size[%]` or
- `[image], _minimal_size[%], _maximal_size[%]`

### Description:

Return the `ideal` size WxH for a window intended to display an image of specified size on screen.

### Default values:

`depth=1`, `minimal_size=128` and `maximal_size=85%`.

---

## flood

Built-in command

### Arguments:

- `x[%], _y[%], _z[%], _tolerance>=0, _is_high_connectivity={ 0 | 1 }, _opacity, _color1, ...`

## Description:

Flood-fill selected images using specified value and tolerance.

## Default values:

`y=z=0`, `tolerance=0`, `is_high_connectivity=0`, `opacity=1` and `color1=0`.

## Example of use:

```
$ gmic image.jpg repeat 1000 flood {u(100)}%,{u(100)}%,0,20,0,1,${-rgb} done
```



---

## flower

### Arguments:

- `_amplitude,_frequency,_offset_r[%],_angle,_center_x[%],_center_y[%],_boundary_conditions`  
`{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror}`

## Description:

Apply flower deformation on selected images.

## Default values:

`amplitude=30`, `frequency=6`, `offset_r=0`, `angle=0`, `center_x=center_y=50%` and `boundary_conditions=3`.

## Example of use:

```
$ gmic image.jpg +flower ,
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

## focale3d

Built-in command

### Arguments:

- `focale`

### Description:

Set 3D focale.

(equivalent to shortcut command `f3d`).

Set `focale` to 0 to enable parallel projection (instead of perspective).

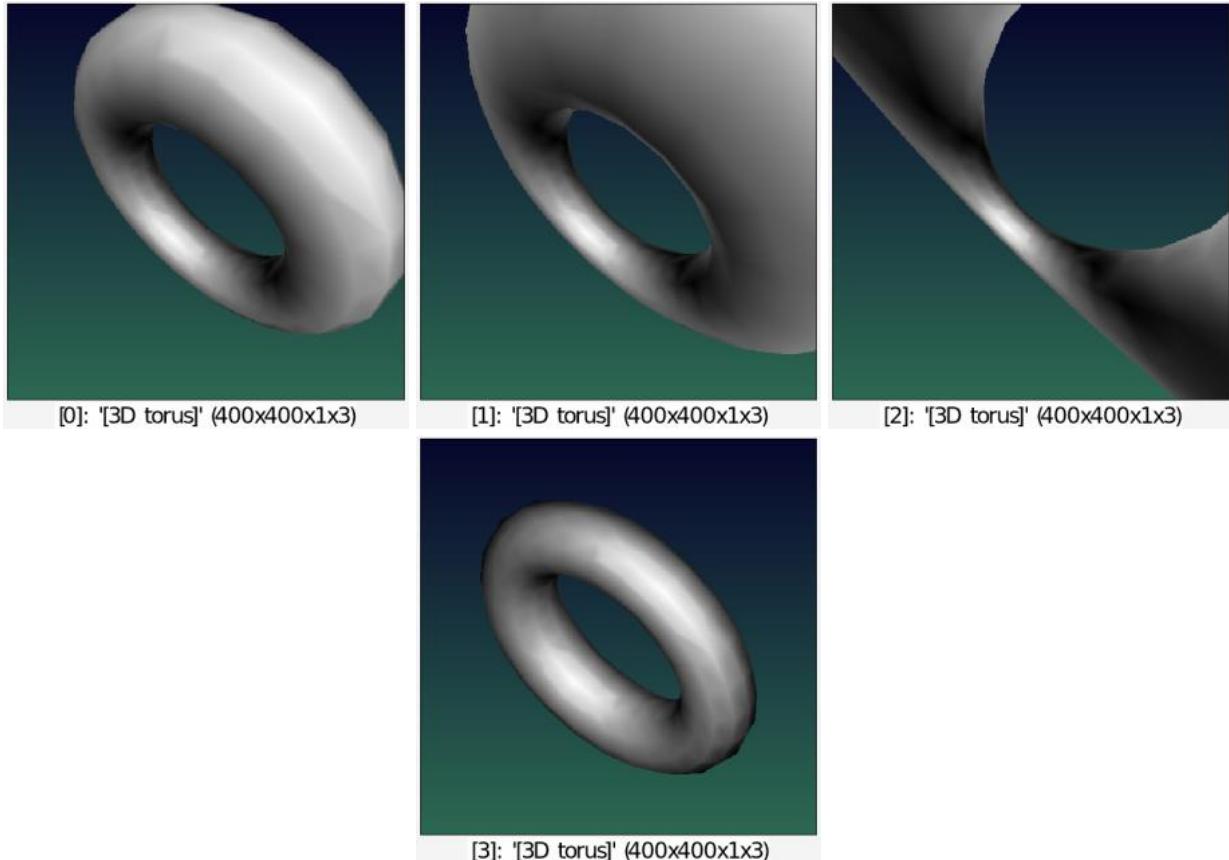
Set negative `focale` will disable 3D sprite zooming.

### Default values:

`focale=700`.

### Example of use:

```
$ gmic repeat 5 torus3d 100,30 rotate3d[-1] 1,1,0,60 focale3d {$<*90}
snapshot3d[-1] 400 done remove[0]
```



# fontchart

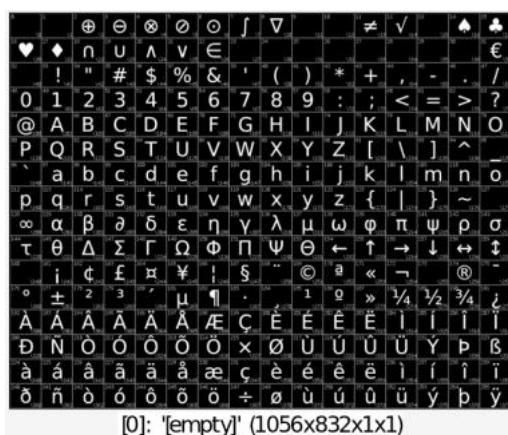
## No arguments

## Description:

Insert G'MIC font chart at the end of the image list.

## Example of use:

```
$ gmic fontchart
```



## Built-in command

# for

## Arguments:

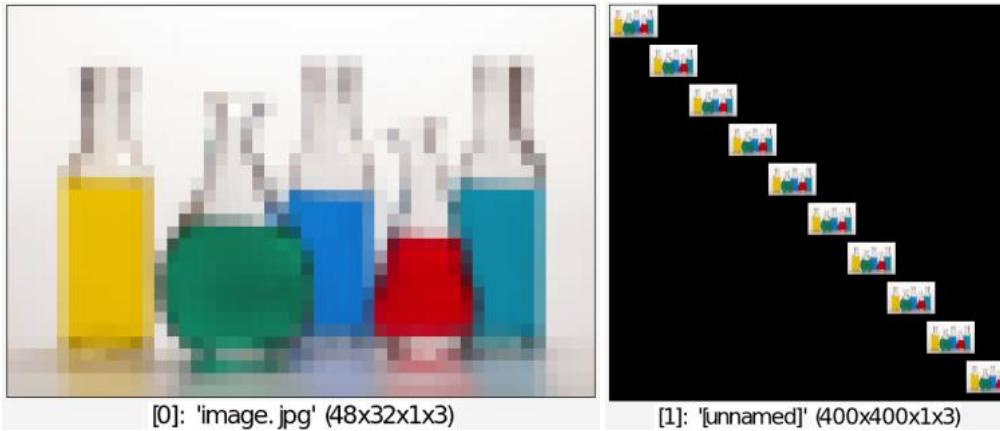
- `condition`

## Description:

Start a `for...done` block.

## Example of use:

```
$ gmic image.jpg resize2dy 32 400,400,1,3 x=0 for $x<400 image[1]  
[0],$x,$x x+=40 done
```



---

# fps

## No arguments

## Description:

Return the number of time this function is called per second, or -1 if this info is not yet available.

Useful to display the framerate when displaying animations.

---

# fractalize

## Arguments:

- `0<=detail_level<=1`

## Description:

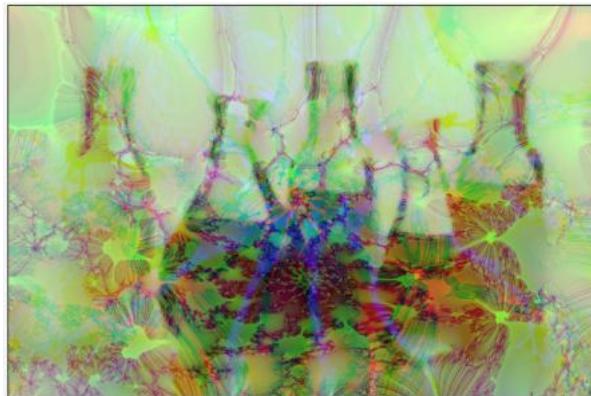
Randomly fractalize selected images.

## Default values:

`detail_level=0.8`

## Example of use:

```
$ gmic image.jpg fractalize ,
```



[0]: 'image.jpg' (640x427x1x3)

---

## frame.blur

### Arguments:

- `_sharpness>0,_size>0,_smoothness,_shading,_blur`

### Description:

Draw RGBA-colored round frame in selected images.

## Default values:

`sharpness=10, size=30, smoothness=0, shading=1` and `blur=3%`.

## Example of use:

```
$ gmic image.jpg frame.blur 3,30,8,10%
```



## frame\_cube

### Arguments:

- `_depth>=0,_centering_x,_centering_y,_left_side={0=normal | 1=mirror-x | 2=mirror-y | 3=mirror-xy},_right_side,_lower_side,_upper_side`

### Description:

Insert 3D frames in selected images.

### Default values:

`depth=1, centering_x=centering_y=0` and  
`left_side=right_side,lower_side=upper_side=0`.

### Example of use:

```
$ gmic image.jpg frame_cube ,
```



## frame\_fuzzy

### Arguments:

- `_size_x[%]>=0, _size_y[%]>=0, _fuzzyness>=0, _smoothness[%]>=0, _R, _G, _B, _A`

## Description:

Draw RGBA-colored fuzzy frame in selected images.

## Default values:

`size_y=size_x`, `fuzzyness=5`, `smoothness=1` and `R=G=B=A=255`.

## Example of use:

```
$ gmic image.jpg frame_fuzzy 20
```



## frame\_painting

### Arguments:

- `_size[%]>=0, 0<=_contrast<=1, _profile_smoothness[%]>=0, _R, _G, _B, _vignette_size`

## Description:

Add a painting frame to selected images.

## Default values:

`size=10%`, `contrast=0.4`, `profile_smoothness=6%`, `R=225`, `G=200`, `B=120`,  
`vignette_size=2%`, `vignette_contrast=400`, `defects_contrast=50`,  
`defects_density=10`, `defects_size=1`, `defects_smoothness=0.5%` and  
`serial_number=123456789`.

## Example of use:

```
$ gmic image.jpg frame_painting ,
```



## frame\_pattern

### Arguments:

- `M>=3,_constrain_size={ 0 | 1 }` or
- `M>=3,_[frame_image],_constrain_size={ 0 | 1 }`

### Description:

Insert selected pattern frame in selected images.

### Default values:

`pattern=0` and `constrain_size=0`.

### Example of use:

```
$ gmic image.jpg frame_pattern 8
```



## frame\_round

### Arguments:

- `_sharpness>0, _size>=0, _smoothness, _shading, _R, _G, _B, _A`

## Description:

Draw RGBA-colored round frame in selected images.

## Default values:

`sharpness=10, size=10, smoothness=0, shading=0` and `R=G=B=A=255`.

## Example of use:

```
$ gmic image.jpg frame_round 10
```



---

## frame\_seamless

### Arguments:

- `frame_size>=0, _patch_size>0, _blend_size>=0, _frame_direction={ 0=inner (preserve image size) | 1=outer }`

## Description:

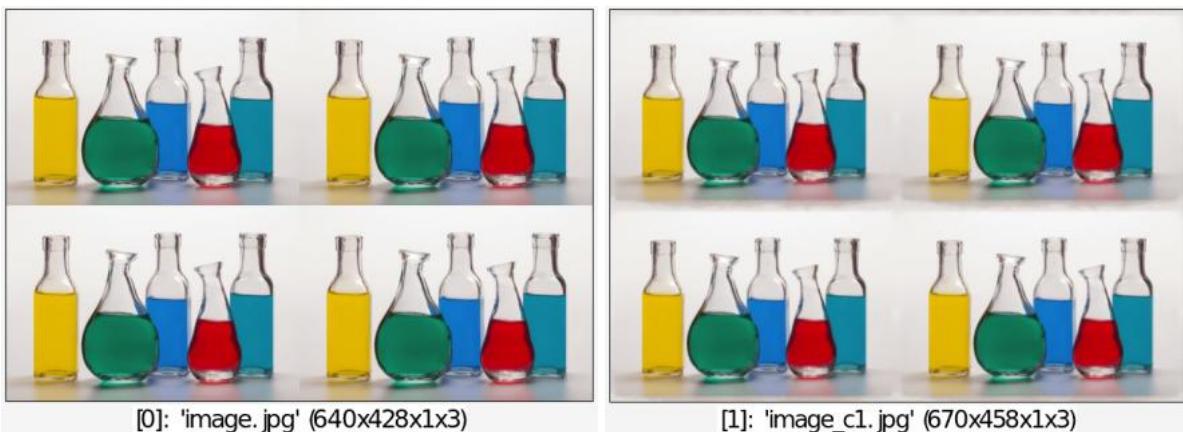
Insert frame in selected images, so that tiling the resulting image makes less visible seams.

## Default values:

`patch_size=7, blend_size=5` and `frame_direction=1`.

## Example of use:

```
$ gmic image.jpg +frame_seamless 30 array 2,2
```



---

## frame\_x

### Arguments:

- `size_x[%],_col1,...,_colN`

### Description:

Insert colored frame along the x-axis in selected images.

### Default values:

`col1=col2=col3=255` and `col4=255`.

### Example of use:

```
$ gmic image.jpg frame_x 20,255,0,255
```



---

## frame\_xy

### Arguments:

- `size_x[%],size_y[%],_col1,...,_colN`

## Description:

Insert colored frame along the x-axis in selected images.

## Default values:

`size_y=size_x`, `col1=col2=col3=255` and `col4=255`.

(equivalent to shortcut command `frame`).

## Example of use:

```
$ gmic image.jpg frame_xy 1,1,0 frame_xy 20,10,255,0,255
```



[0]: 'image.jpg' (682x449x1x3)

---

## frame\_xyz

### Arguments:

- `size_x[%],_size_y[%],_size_z[%]_col1,...,_colN`

## Description:

Insert colored frame along the x-axis in selected images.

## Default values:

`size_y=size_x=size_z`, `col1=col2=col3=255` and `col4=255`.

---

## frame\_y

### Arguments:

- `size_y[%],_col1,...,_colN`

## Description:

Insert colored frame along the y-axis in selected images.

## Default values:

`col1=col2=col3=255` and `col4=255`.

## Example of use:

```
$ gmic image.jpg frame_y 20,255,0,255
```



---

## function1d

### Arguments:

- `0<=smoothness<=1,x0>=0,y0,x1>=0,y1,...,xn>=0,yn`

### Description:

Insert continuous 1D function from specified list of keypoints (xk,yk)

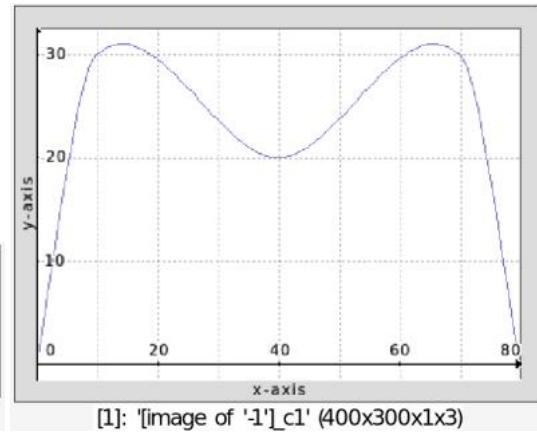
in range [0,max(xk)] (xk are positive integers).

## Example of use:

```
$ gmic function1d 1,0,0,10,30,40,20,70,30,80,0 +display_graph 400,300
```



[0]: '[image of '1']' (81x1x1x1)



[1]: '[image of '1'\_c1]' (400x300x1x3)

## gaussian

### Arguments:

- `_sigma1[%]`, `_sigma2[%]`, `_angle`

### Description:

Draw a centered gaussian on selected images, with specified standard deviations and orientation.

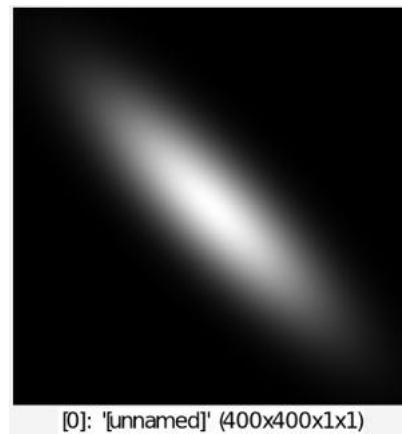
### Default values:

`sigma1=3`, `sigma2=sigma1` and `angle=0`.

This command has a [tutorial page](#).

### Example of use:

```
$ gmic 400,400 gaussian 100,30,45
```



[0]: '[unnamed]' (400x400x1x1)

## gaussians3d

### Arguments:

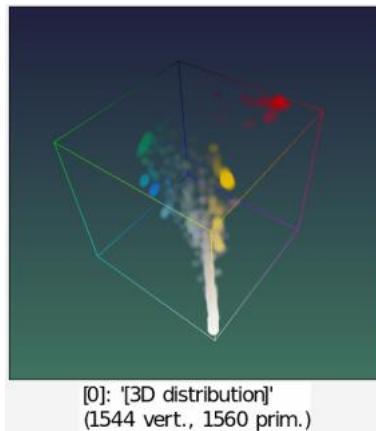
- `_size>0, _opacity`

## Description:

Convert selected 3D objects into set of 3D gaussian-shaped sprites.

## Example of use:

```
$ gmic image.jpg r2dy 32 distribution3d gaussians3d 20 colorcube3d  
primitives3d[-1] 1 +3d
```



---

## gcd

### Arguments:

- `a, b`

## Description:

Return the GCD (greatest common divisor) between a and b.

---

## ge

Built-in command

### Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

## Description:

Compute the boolean 'greater or equal than' of selected images with specified value, image or mathematical expression, or compute the boolean 'greater or equal than' of selected images.

(equivalent to shortcut command `>=`).

## Examples of use:

- Example #1

```
$ gmic image.jpg ge {ia}
```



- Example #2

```
$ gmic image.jpg +mirror x ge
```



---

## glow

### Arguments:

- `_amplitude>=0`

### Description:

Add soft glow on selected images.

### Default values:

`amplitude=1%`.

## Example of use:

```
$ gmic image.jpg glow ,
```



---

## gmd2ascii

### Arguments:

- `_max_line_length>0,_indent_forced_newlines>=0` or
- `(no arg)`

### Description:

Convert selected gmd-formatted text images to ascii format.

### Default values:

`max_line_length=80` and `indent_forced_newline=0`.

---

## gmd2html

### Arguments:

- `_include_default_header_footer={ 0=none | 1=Reference | 2=Tutorial | 3=News }` or
- `(no arg)`

### Description:

Convert selected gmd-formatted text images to html format.

### Default values:

`include_default_header_footer=1`.

---

# gmic3d

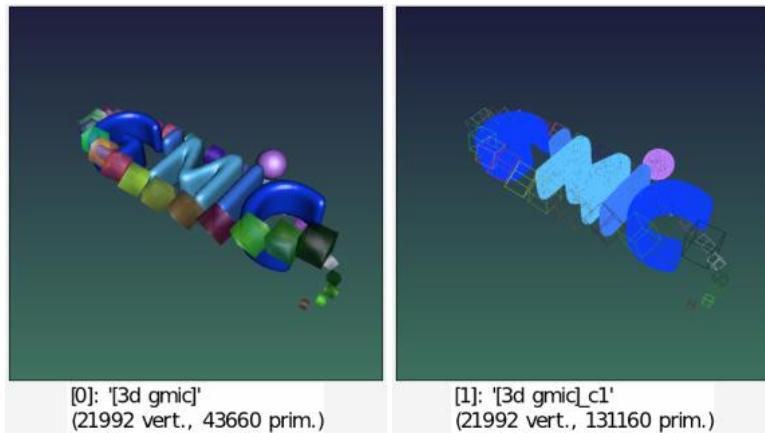
## No arguments

### Description:

Input a 3D G'MIC logo.

### Example of use:

```
$ gmic gmic3d +primitives3d 1
```



---

# gradient

## Arguments:

- `{ x | y | z | c }...{ x | y | z | c },_scheme,_boundary_conditions` or
- `(no arg)`

### Description:

Compute the gradient components (first derivatives) of selected images, along specified axes.

(equivalent to shortcut command `g`).

`scheme` can be `{ -1=backward | 0=centered | 1=forward | 2=sobel | 3=rotation-invariant (default) | 4=deriche | 5=vanvliet }`.

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

(no arg) compute all significant components.

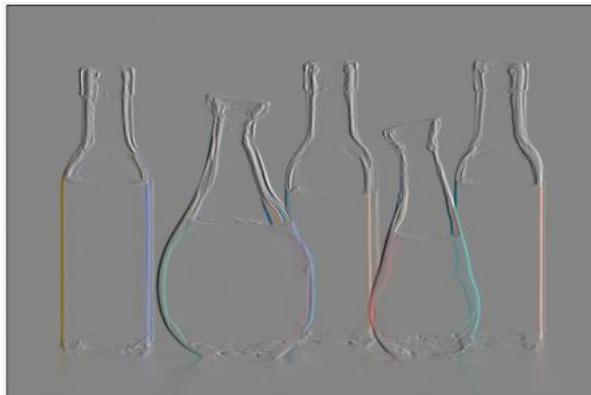
### Default values:

`scheme=0` and `boundary_conditions=1`.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg gradient
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image.jpg' (640x427x1x3)

---

## gradient2rgb

### Arguments:

- `_is_orientation={ 0 | 1 }`

### Description:

Compute RGB representation of 2D gradient of selected images.

### Default values:

`is_orientation=0`.

## Example of use:

```
$ gmic image.jpg +gradient2rgb 0 equalize[-1]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

# gradient\_norm

No arguments

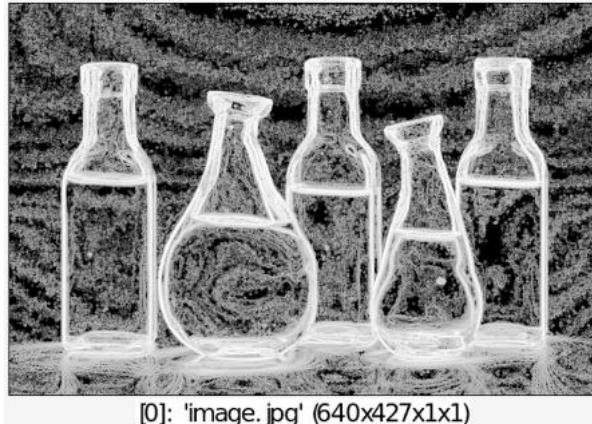
## Description:

Compute gradient norm of selected images.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg gradient_norm equalize
```



---

# gradient\_orientation

## Arguments:

- `_dimension={1,2,3}`

## Description:

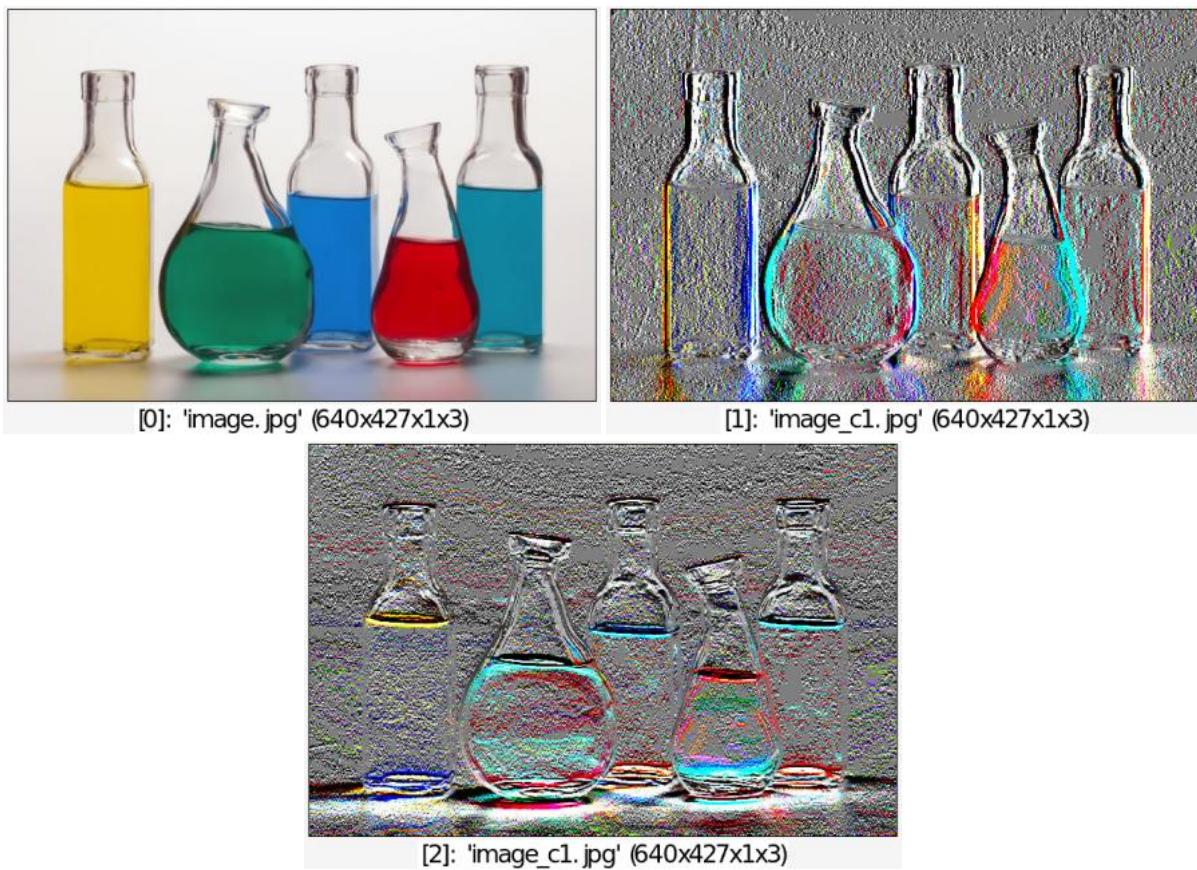
Compute N-d gradient orientation of selected images.

## Default values:

`dimension=3`.

## Example of use:

```
$ gmic image.jpg +gradient_orientation 2
```



# graph

Built-in command

## Arguments:

- `[function_image],_plot_type,_vertex_type,_ymin,_ymax,_opacity,_pattern,_color1`  
or
- `'formula',_resolution>=0,_plot_type,_vertex_type,_xmin,xmax,_ymin,_ymax,_opacity,_color1`

## Description:

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.

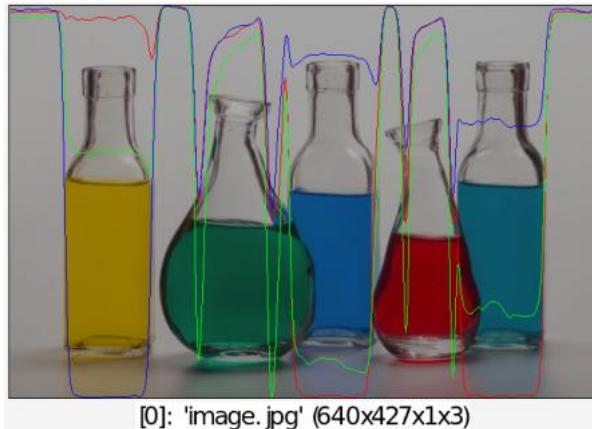
## Default values:

`plot_type=1`, `vertex_type=1`, `'ymin=ymax=0 (auto)'`, `opacity=1`, `pattern=(undefined)`

and `color1=0`.

## Example of use:

```
$ gmic image.jpg +rows 50% blur[-1] 3 split[-1] c div[0] 1.5 graph[0]
[1],2,0,0,0,1,255,0,0 graph[0] [2],2,0,0,0,1,0,255,0 graph[0]
[3],2,0,0,0,1,0,0,255 keep[0]
```



[0]: 'image.jpg' (640x427x1x3)

## grid

### Arguments:

- `size_x[%]>=0, size_y[%]>=0, _offset_x[%], _offset_y[%], _opacity, _pattern, _color`

### Description:

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 values:

`offset_x=offset_y=0`, `opacity=1`, `pattern=(undefined)` and `color1=0`.

### Examples of use:

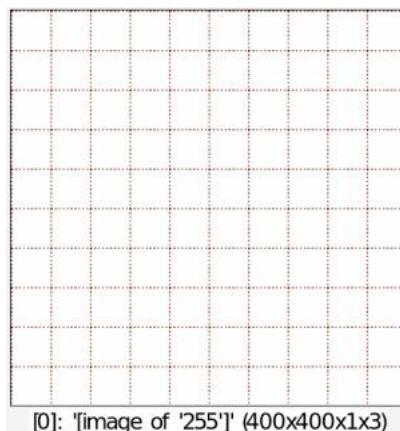
- **Example #1**

```
$ gmic image.jpg grid 10%,10%,0,0,0.5,255
```



- **Example #2**

```
$ gmic 400,400,1,3,255 grid 10%,10%,0,0,0.3,0xCCCCCCCC,128,32,16
```



---

## gt

Built-in command

### Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

### Description:

Compute the boolean 'greater than' of selected images with specified value, image or mathematical expression, or compute the boolean 'greater than' of selected images.

(equivalent to shortcut command `>`).

### Examples of use:

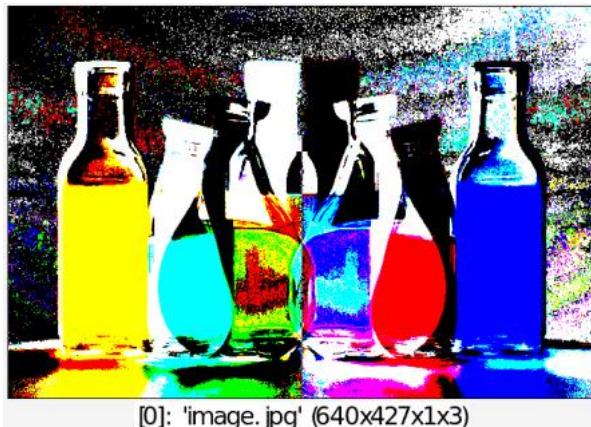
- **Example #1**

```
$ gmic image.jpg gt {ia}
```



- **Example #2**

```
$ gmic image.jpg +mirror x gt
```



---

## guided

Built-in command

### Arguments:

- [guide], radius[%]>=0, regularization[%]>=0 or
- radius[%]>=0, regularization[%]>=0

### Description:

Blur selected images by guided image filtering.

If a guide image is provided, it is used to drive the smoothing process.

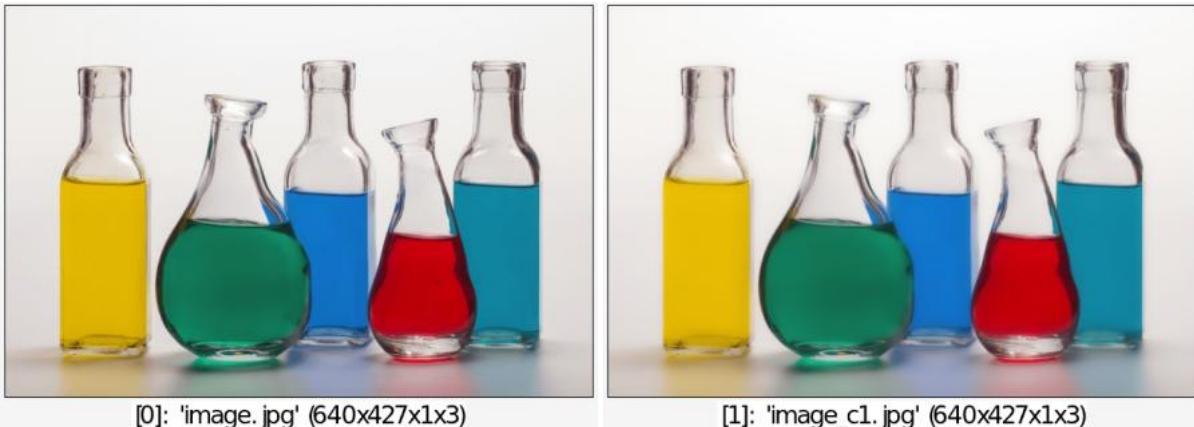
A guide image must be of the same xyz-size as the selected images.

This command implements the filtering algorithm described in:

He, Kaiming; Sun, Jian; Tang, Xiaoou, "Guided Image Filtering",  
IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.35, no.6, pp.1397,1409, June  
2013

### Example of use:

```
$ gmic image.jpg +guided 5,400
```



---

## gyroid3d

### Arguments:

- `_resolution>0, _zoom`

### Description:

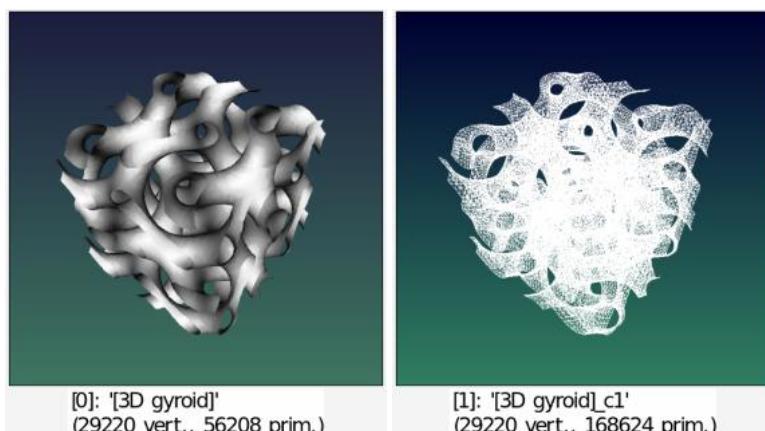
Input 3D gyroid at (0,0,0), with specified resolution.

### Default values:

`resolution=32` and `zoom=5`.

### Example of use:

```
$ gmic gyroid3d 48 +primitives3d 1
```



---

## haar

## Arguments:

- `scale>0`

## Description:

Compute the direct haar multiscale wavelet transform of selected images.

## See also:

`ihaar`.

This command has a [tutorial page](#).

---

# halftone

## Arguments:

- `nb_levels>=2,_size_dark>=2,_size_bright>=2,_shape={ 0=square | 1=diamond | 2=circle | 3=inv-square | 4=inv-diamond | 5=inv-circle },_smoothness[%]>=0`

## Description:

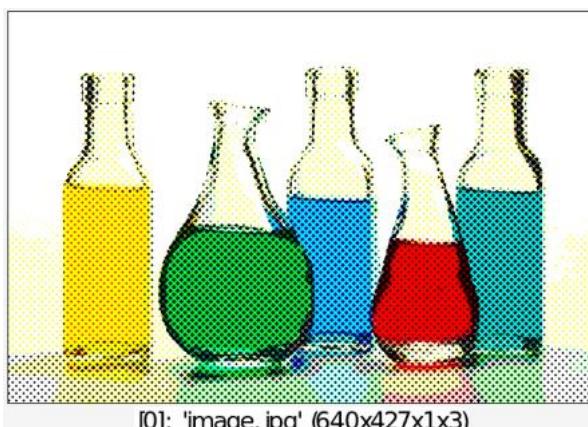
Apply halftone dithering to selected images.

## Default values:

`nb_levels=5`, `size_dark=8`, `size_bright=8`, `shape=5` and `smoothness=0`.

## Example of use:

```
$ gmic image.jpg halftone ,
```



# hardsketchbw

## Arguments:

- `_amplitude>=0,_density>=0,_opacity,0<=_edge_threshold<=100,_is_fast={ 0 | 1 }`

## Description:

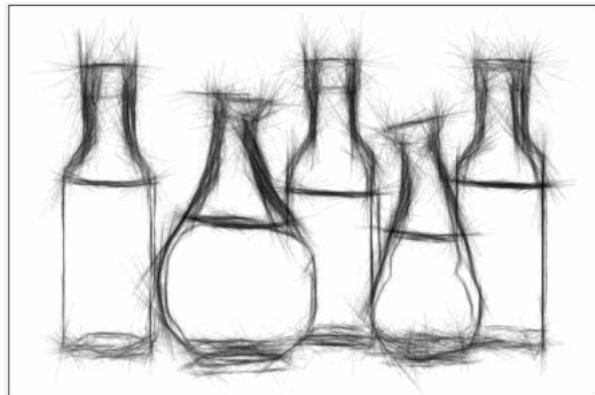
Apply hard B&W sketch effect on selected images.

## Default values:

`amplitude=1000, sampling=3, opacity=0.1, edge_threshold=20` and `is_fast=0`.

## Example of use:

```
$ gmic image.jpg +hardsketchbw 200,70,0.1,10 median[-1] 2 +local  
reverse blur[-1] 3 blend[-2,-1] overlay endlocal
```



---

## hcy2rgb

### No arguments

## Description:

Convert color representation of selected images from HCY to RGB.

---

# hearts

## Arguments:

- `_density>=0`

## Description:

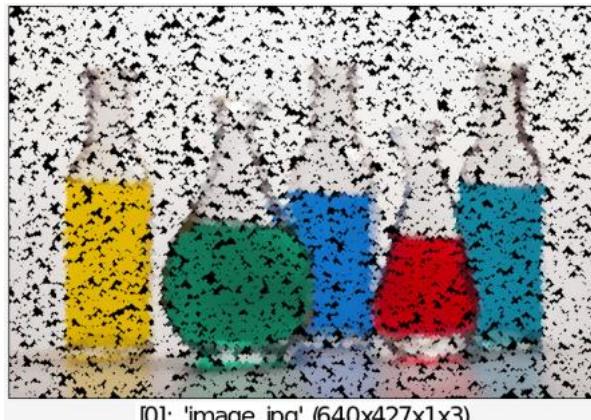
Apply heart effect on selected images.

## Default values:

`density=10`.

## Example of use:

```
$ gmic image.jpg hearts ,
```



# heat\_flow

## Arguments:

- `_nb_iter>=0, _dt, _keep_sequence={ 0 | 1 }`

## Description:

Apply iterations of the heat flow on selected images.

## Default values:

`nb_iter=10`, `dt=30` and `keep_sequence=0`.

## Example of use:

```
$ gmic image.jpg +heat_flow 20
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## help

### Arguments:

- `command` or
- `(no arg)`

### Description:

Display help (optionally for specified command only) and exit.

(equivalent to shortcut command `h`).

---

## hessian

### Arguments:

- `{ xx | xy | xz | yy | yz | zz }...{ xx | xy | xz | yy | yz | zz }`,  
`_boundary_conditions` or
- `(no arg) :`

### Description:

Compute the hessian components (second derivatives) of selected images along specified axes.

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

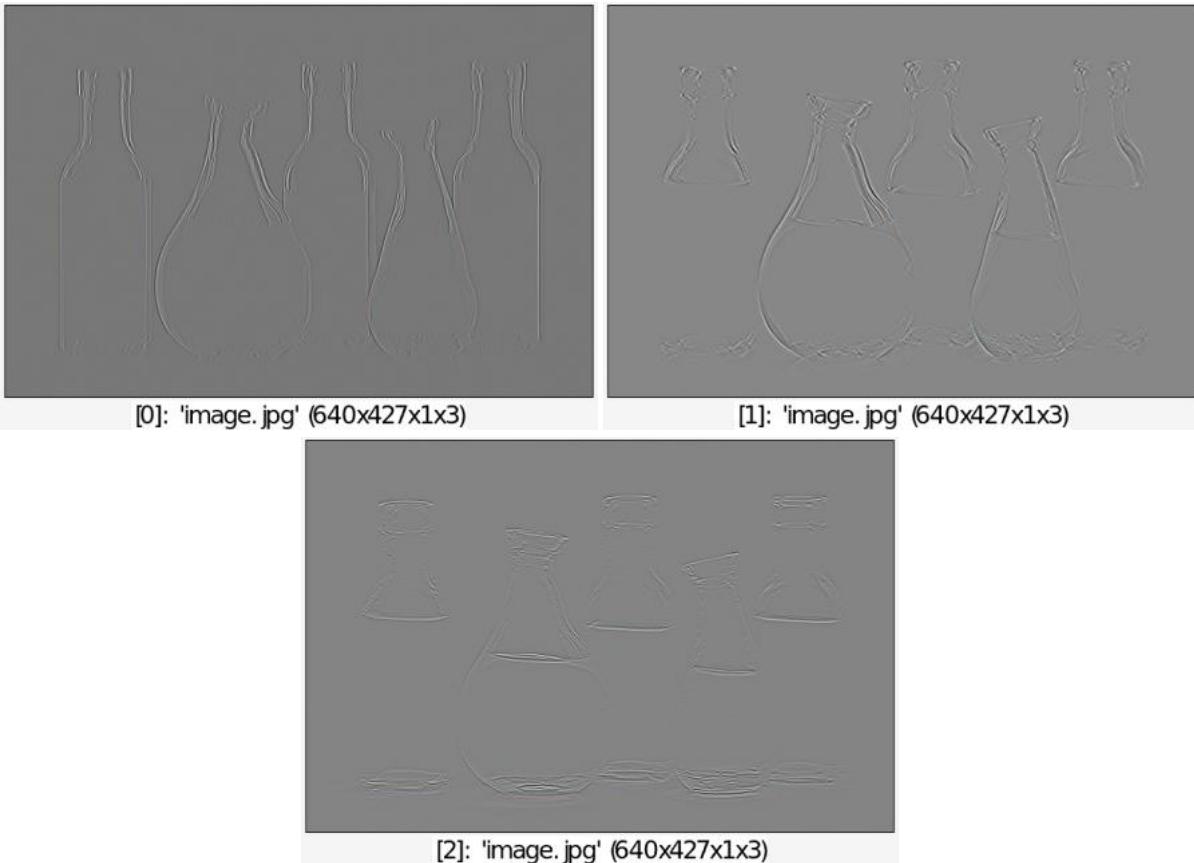
(no arg) compute all significant components.

### Default values:

`boundary_conditions=1`.

### Example of use:

```
$ gmic image.jpg hessian
```



---

## hex

### Arguments:

- `hexadecimal_int1, ...`

### Description:

Print specified hexadecimal integers into their binary, octal, decimal and string representations.

---

## hex2dec

### Arguments:

- `hexadecimal_int1, ...`

### Description:

Convert specified hexadecimal integers into their decimal representations.

---

# hex2img

## Arguments:

- `"hexadecimal_string"`

## Description:

Insert new image 1xN at the end of the list with values specified by the given hexadecimal-encoded string.

---

# hex2img8

## No arguments

## Description:

Convert selected hexadecimal representations (ascii-encoded) into 8bits-valued vectors.

---

# hex2str

## Arguments:

- `hexadecimal_string`

## Description:

Convert specified hexadecimal string into a string.

## See also:

`str2hex`.

---

Built-in command

# histogram

## Arguments:

- `nb_levels>0[%],_min_value[%],_max_value[%]`

## Description:

Compute the histogram of selected images.

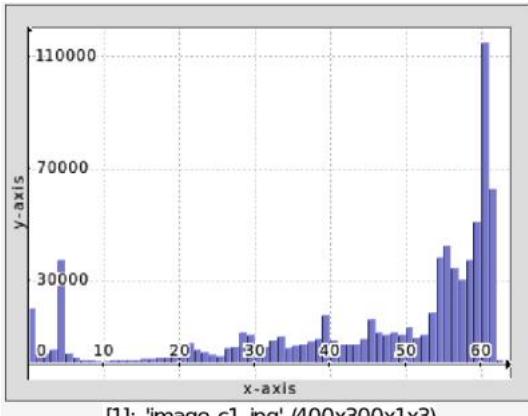
If value range is set, the histogram is estimated only for pixels in the specified value range. Argument `max_value` must be specified if `min_value` is set.

## Default values:

`min_value=0%` and `max_value=100%`.

## Example of use:

```
$ gmic image.jpg +histogram 64 display_graph[-1] 400,300,3
```



---

## histogram3d

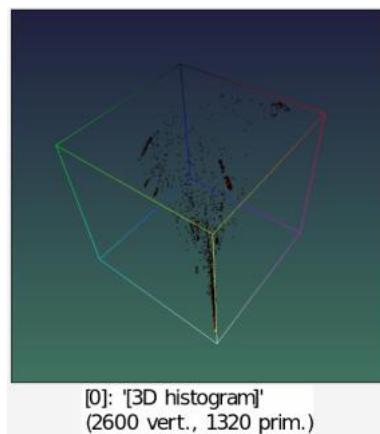
### No arguments

### Description:

Get 3D color histogram of selected images.

### Example of use:

```
$ gmic image.jpg resize2dx 64 histogram3d circles3d 3 opacity3d. 0.75  
colorcube3d primitives3d[-1] 1 add3d
```



---

## histogram\_cumul

## Arguments:

- `_nb_levels>0, _is_normalized={ 0 | 1 }, _val0[%], _val1[%]`

## Description:

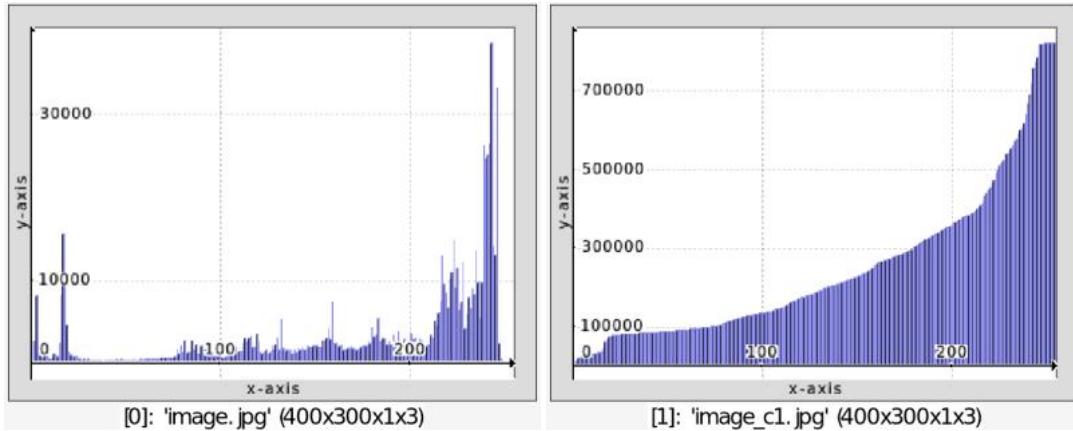
Compute cumulative histogram of selected images.

## Default values:

`nb_levels=256`, `is_normalized=0`, `val0=0%` and `val1=100%`.

## Example of use:

```
$ gmic image.jpg +histogram_cumul 256 histogram[0] 256 display_graph  
400,300,3
```



---

## histogram\_nd

## Arguments:

- `nb_levels>0[%], _value0[%], _value1[%]`

## Description:

Compute the 1D,2D or 3D histogram of selected multi-channels images (having 1,2 or 3 channels).

If value range is set, the histogram is estimated only for pixels in the specified value range.

## Default values:

`value0=0%` and `value1=100%`.

## Example of use:

```
$ gmic image.jpg channels 0,1 +histogram_nd 256
```



---

## histogram\_pointwise

### Arguments:

- `nb_levels>0[%]`, `_value0[%]`, `_value1[%]`

### Description:

Compute the histogram of each vector-valued point of selected images.

If value range is set, the histogram is estimated only for values in the specified value range.

### Default values:

`value0=0%` and `value1=100%`.

---

## hough

### Arguments:

- `_width>0`, `_height>0`, `gradient_norm_voting={ 0 | 1 }`

### Description:

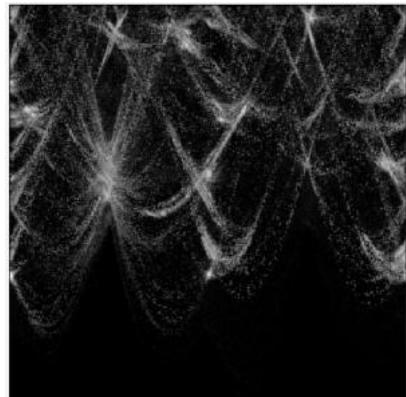
Compute hough transform (theta,rho) of selected images.

### Default values:

`width=512`, `height=width` and `gradient_norm_voting=1`.

### Example of use:

```
$ gmic image.jpg +blur 1.5 hough[-1] 400,400 blur[-1] 0.5 add[-1] 1 log[-1]
```



---

## houghsketchbw

### Arguments:

- `_density>=0, _radius>0, 0<=_threshold<=100, 0<=_opacity<=1, _votesize[%]>0`

### Description:

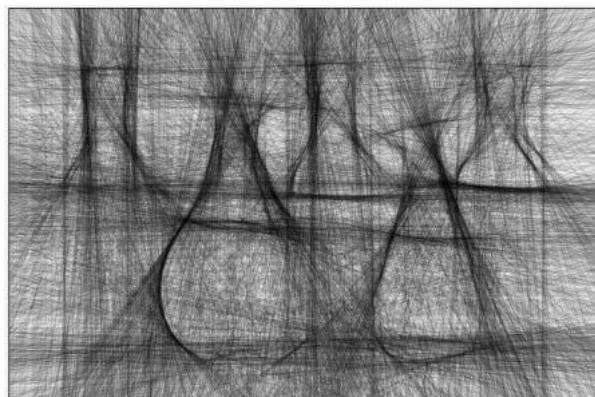
Apply hough B&W sketch effect on selected images.

### Default values:

`density=100`, `radius=3`, `threshold=100`, `opacity=0.1` and `votesize=100%`.

### Example of use:

```
$ gmic image.jpg +houghsketchbw ,
```



---

## hsim2rgb

**No arguments**

**Description:**

Convert color representation of selected images from HSI to RGB.

---

**hs182rgb**

**No arguments**

**Description:**

Convert color representation of selected images from HSI8 to RGB.

---

**hsl2rgb**

**No arguments**

**Description:**

Convert color representation of selected images from HSL to RGB.

---

**hsl82rgb**

**No arguments**

**Description:**

Convert color representation of selected images from HSL8 to RGB.

---

**hsv2rgb**

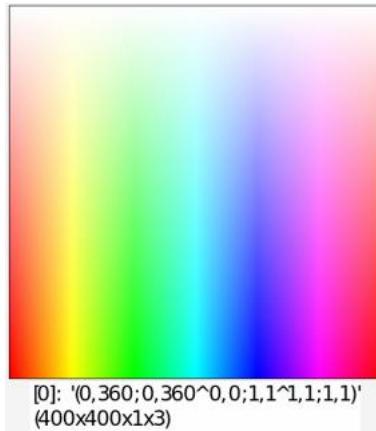
**No arguments**

**Description:**

Convert color representation of selected images from HSV to RGB.

**Example of use:**

```
$ gmic (0,360;0,360^0,0;1,1^1,1;1,1) resize 400,400,1,3,3 hsv2rgb
```



---

## hsv2rgb

**No arguments**

**Description:**

Convert color representation of selected images from HSV8 to RGB.

---

## idct

**Arguments:**

- `_ { x | y | z } ... { x | y | z }` or
- `(no arg)`

**Description:**

Compute the inverse discrete cosine transform of selected images, optionally along the specified axes only.

Output images are always evenly sized, so this command may change the size of the selected images.

(dct images obtained with the `dct` command are evenly sized anyway).

**Default values:**

`(no arg)`

**See also:**

`dct`.

This command has a [tutorial page](#).

---

# iee

## No arguments

### Description:

Compute gradient-orthogonal-directed 2nd derivative of image(s).

### Example of use:

```
$ gmic image.jpg iee
```



---

# if

Built-in command

## Arguments:

- `condition`

### Description:

Start a `if...[elif]...[else]...fi` block and test if specified condition holds.

`condition` is a mathematical expression, whose evaluation is interpreted as `{ 0=false | other=true }`.

This command has a [tutorial page](#).

### Example of use:

```
$ gmic image.jpg if ia<64 add 50% elif ia<128 add 25% elif ia<192 sub 25% else sub 50% fi cut 0,255
```



## ifft

Built-in command

### Arguments:

- `_{{ x | y | z }}...{{ x | y | z }}`

### Description:

Compute the inverse fourier transform (real and imaginary parts) of selected images.

optionally along the specified axes only.

### See also:

[fft](#).

This command has a [tutorial page](#).

## ifftpolar

### No arguments

### Description:

Compute inverse fourier transform of selected images, from centered magnitude/phase images.

## ihaar

### Arguments:

- `scale>0`

### Description:

Compute the inverse haar multiscale wavelet transform of selected images.

## See also:

[haar](#).

This command has a [tutorial page](#).

---

# ilaplacian

## Arguments:

- { nb\_iterations>0 | 0 },\_[initial\_estimate]

## Description:

Invert selected Laplacian images.

If given `nb_iterations` is `0`, inversion is done in Fourier space (single iteration), otherwise, by applying `nb_iterations` of a Laplacian-inversion PDE flow.

Note that the resulting inversions are just estimation of possible/approximated solutions.

## Default values:

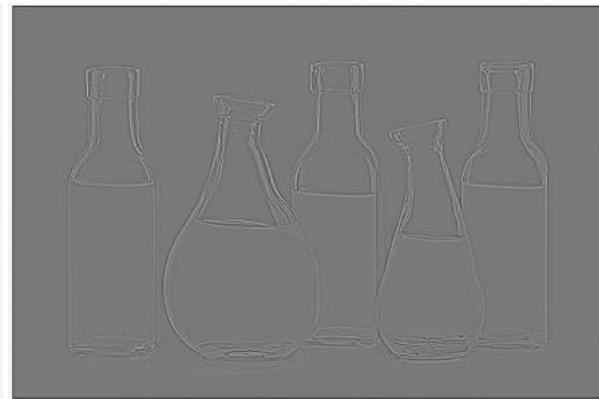
`nb_iterations=0` and `[initial_estimated]=(undefined)`.

## Example of use:

```
$ gmic image.jpg +laplacian +ilaplacian[-1] 0
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)



[2]: 'image\_c2.jpg' (640x427x1x3)

# image

Built-in command

## Arguments:

- `[sprite],_x[%|~],_y[%|~],_z[%|~],_c[%|~],_opacity,_[opacity_mask],_max_opacity`

## Description:

Draw specified sprite image on selected images.

(*equivalent to shortcut command [j](#)*).

If one of the x,y,z or c argument ends with a `~`, its value is expected to be a centering ratio (in [0,1]) rather than a position.

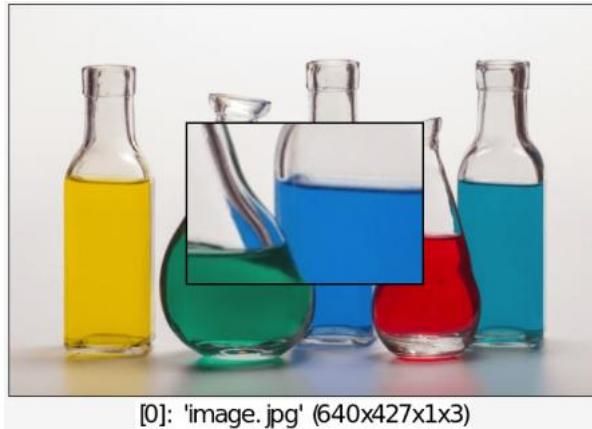
Usual centering ratio are `{ 0=left-justified | 0.5=centered | 1=right-justified }`.

## Default values:

`x=y=z=c=0`, `opacity=1`, `opacity_mask=(undefined)` and `max_opacity_mask=1`.

## Example of use:

```
$ gmic image.jpg +crop 40%,40%,60%,60% resize[-1] 200%,200%,1,3,5  
frame[-1] 2,2,0 image[0] [-1],30%,30% keep[0]
```



---

## image6cube3d

**No arguments**

**Description:**

Generate 3D mapped cubes from 6-sets of selected images.

**Example of use:**

```
$ gmic image.jpg animate flower,"30,0","30,5",6 image6cube3d
```



[0]: '[3D image cube]' (24 vert., 6 prim.)

---

## imageblocks3d

**Arguments:**

- `_maximum_elevation,_smoothness[%]>=0`

**Description:**

Generate 3D blocks from selected images.

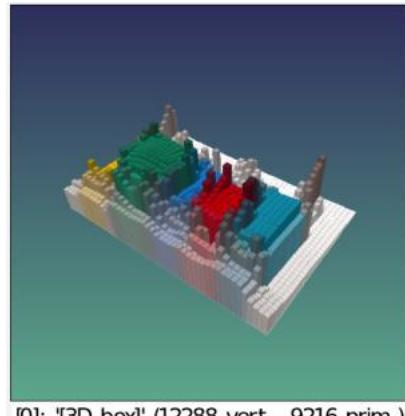
Transparency of selected images is taken into account.

## Default values:

`maximum_elevation=10` and `smoothness=0`.

## Example of use:

```
$ gmic image.jpg resize2dy 32 imageblocks3d -20 mode3d 3
```



---

## imagecube3d

### No arguments

## Description:

Generate 3D mapped cubes from selected images.

## Example of use:

```
$ gmic image.jpg imagecube3d
```



---

## imagegrid

## Arguments:

- `M>0, N>0`

## Description:

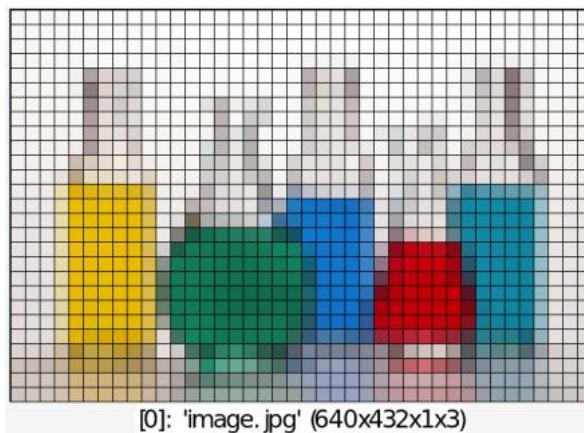
Create MxN image grid from selected images.

## Default values:

`N=M`.

## Example of use:

```
$ gmic image.jpg imagegrid 16
```



---

## imagegrid\_hexagonal

## Arguments:

- `_resolution>0, 0<=_outline<=1`

## Description:

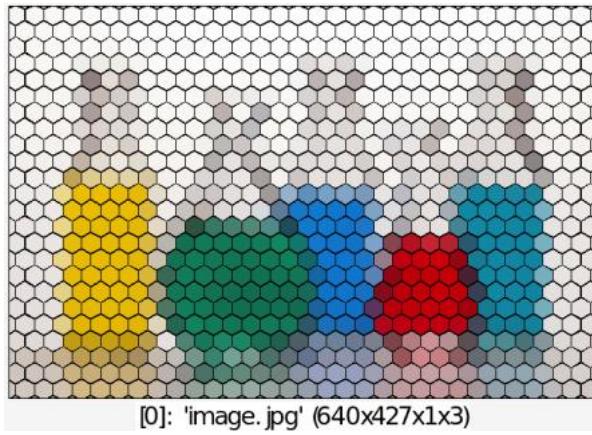
Create hexagonal grids from selected images.

## Default values:

`resolution=32`, `outline=0.1` and `is_antialiased=1`.

## Example of use:

```
$ gmic image.jpg imagegrid_hexagonal 24
```



---

## imagegrid\_triangular

### Arguments:

- `pattern_width>=1, _pattern_height>=1, _pattern_type, 0<=_outline_opacity<=1, _ou`

### Description:

Create triangular grids from selected images.

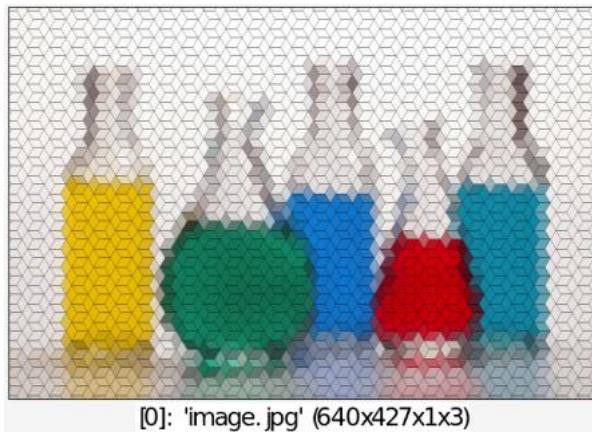
'pattern type' can be `{ 0=horizontal | 1=vertical | 2=crossed | 3=cube | 4=decreasing | 5=increasing }`.

### Default values:

`pattern_width=24, pattern_height=pattern_width, pattern_type=0,`  
`outline_opacity=0.1` and `outline_color1=0`.

### Example of use:

```
$ gmic image.jpg imagegrid_triangular 6,10,3,0.5
```



---

## imageplane3d

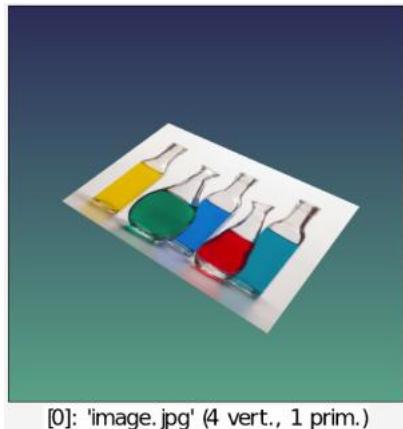
## No arguments

### Description:

Generate 3D mapped planes from selected images.

### Example of use:

```
$ gmic image.jpg imageplane3d
```



---

## imagepyramid3d

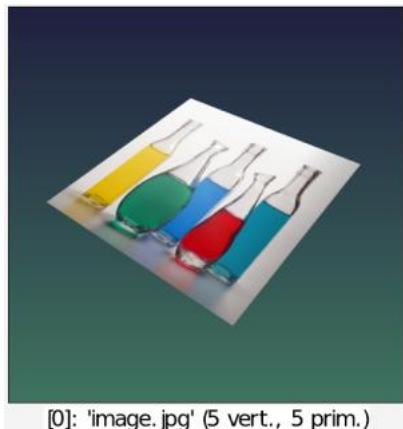
## No arguments

### Description:

Generate 3D mapped pyramids from selected images.

### Example of use:

```
$ gmic image.jpg imagepyramid3d
```



---

## imagerubik3d

## Arguments:

- `_xy_tiles>=1, 0<=xy_shift<=100, 0<=z_shift<=100`

## Description:

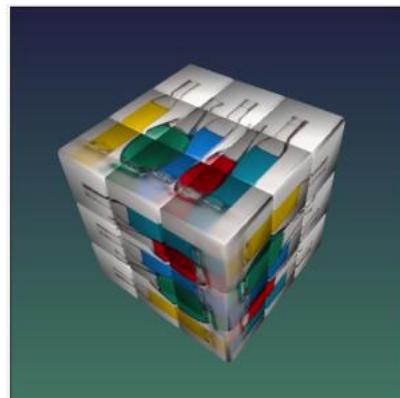
Generate 3D mapped rubik's cubes from selected images.

## Default values:

`xy_tiles=3`, `xy_shift=5` and `z_shift=5`.

## Example of use:

```
$ gmic image.jpg imagerubik3d ,
```



[0]: 'image.jpg' (432 vert., 270 prim.)

---

## imagesphere3d

## Arguments:

- `_resolution1>=3, _resolution2>=3`

## Description:

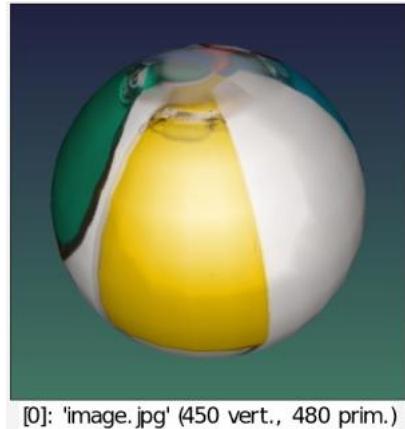
Generate 3D mapped sphere from selected images.

## Default values:

`resolution1=32` and `resolution2=16`.

## Example of use:

```
$ gmic image.jpg imagesphere3d 32,16
```



## img2ascii

### Arguments:

- `_charset,_analysis_scale>0,_analysis_smoothness[%]>=0,_synthesis_scale>0,_ou`

### Description:

Render selected images as binary ascii art.

This command returns the corresponding the list of widths and heights (expressed as a number of characters)  
for each selected image.

### Default values:

'`charset=[ascii charset]', analysis_scale=16, analysis_smoothness=20%,`  
`synthesis_scale=16` and `_output_ascii_filename=[undefined]`'.

### Example of use:

```
$ gmic image.jpg img2ascii ,
```



## img2base64

## Arguments:

- `_encoding={ 0=base64 | 1=base64url },_store_names={ 0 | 1 }`

## Description:

Encode selected images as a base64-encoded string.

The images can be then decoded using command `base642img`.

## Default values:

`encoding=0`.

---

# img2hex

## No arguments

## Description:

Return representation of last image as an hexadecimal-encoded string.

Input image must have values that are integers in [0,255].

---

# img2str

## No arguments

## Description:

Return the content of the latest of the selected images as a special G'MIC input string.

---

# img2text

## Arguments:

- `_line_separator`

## Description:

Return text contained in a multi-line image.

## Default values:

'line\_separator= '.

---

# img82hex

**No arguments**

## Description:

Convert selected 8bits-valued vectors into their hexadecimal representations (ascii-encoded).

## Preamble

- This document is distributed under the **GNU Free Documentation License**, version 1.3.
- A **.pdf version** of this document is available.
- Quick access to the **List of Commands**.

## Version

**G'MIC:** GREYC's Magic for Image Computing

<https://gmic.eu>

Version 3.0.0

Copyright © Since 2008, David Tschumperlé / GREYC / CNRS

<https://www.greyc.fr>

## Table of Contents

- **Usage**
- **Overall Context**
- **Image Definition and Terminology**
- **Items of a Processing Pipeline**
- **Input Data Items**
- **Command Items and Selections**
- **Input/Output Properties**
- **Substitution Rules**
- **Mathematical Expressions**
- **Image and Data Viewers**
- **Adding Custom Commands**
- **List of Commands**
- **Funny Oneliners**
- **G'MIC Markdown**
- **Help Writing Reference Documentation**
- **Installing the G'MIC-Qt Plug-in For 8bf Hosts**
- **Managing 3D Vector Objects**
- **Scientific Publications**
- **Examples of Use**

---

# inn

## No arguments

### Description:

Compute gradient-directed 2nd derivative of image(s).

### Example of use:

```
$ gmic image.jpg inn
```



---

# inpaint

Built-in command

## Arguments:

- `[mask]` or
- `[mask],0,_fast_method` or
- `[mask],_patch_size>=1,_lookup_size>=1,_lookup_factor>=0,_lookup_increment!=0  
{ 0 | 1 }`

### Description:

Inpaint selected images by specified mask.

If no patch size (or 0) is specified, inpainting is done using a fast average or median algorithm.

Otherwise, it used a patch-based reconstruction method, that can be very time consuming.

`fast_method` can be `{ 0=low-connectivity average | 1=high-connectivity average | 2=low-connectivity median | 3=high-connectivity median }`.

### Default values:

```
patch_size=0, fast_method=1, lookup_size=22, lookup_factor=0.5,  
lookup_increment=1, blend_size=0, blend_threshold=0, blend_decay=0.05,
```

`blend_scales=10` and `is_blend_outer=1`.

## Examples of use:

- **Example #1**

```
$ gmic image.jpg 100%,100% ellipse 50%,50%,30,30,0,1,255 ellipse  
20%,20%,30,10,0,1,255 +inpaint[-2] [-1] remove[-2]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

- **Example #2**

```
$ gmic image.jpg 100%,100% circle 30%,30%,30,1,255,0,255 circle  
70%,70%,50,1,255,0,255 +inpaint[0] [1],5,15,0.5,1,9,0 remove[1]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## inpaint\_flow

### Arguments:

- `[mask],_nb_global_iter>=0,_nb_local_iter>=0,_dt>0,_alpha>=0,_sigma>=0`

### Description:

Apply iteration of the inpainting flow on selected images.

### Default values:

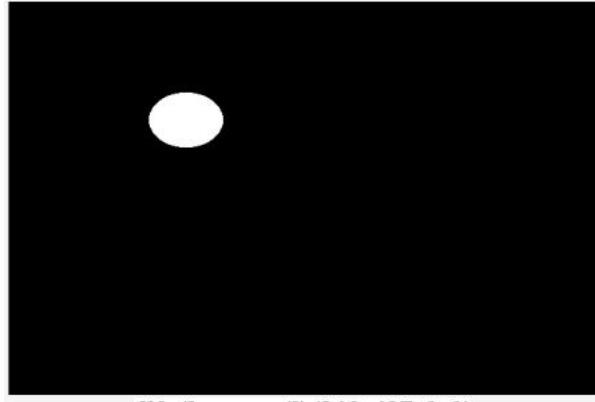
`nb_global_iter=10`, `nb_local_iter=100`, `dt=5`, `alpha=1` and `sigma=3`.

## Example of use:

```
$ gmic image.jpg 100%,100% ellipse[-1] 30%,30%,40,30,0,1,255  
inpaint_flow[0] [1]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: '[unnamed]' (640x427x1x1)

## inpaint\_holes

### Arguments:

- `maximal_area[%]>=0,_tolerance>=0,_is_high_connectivity={ 0 | 1 }`

### Description:

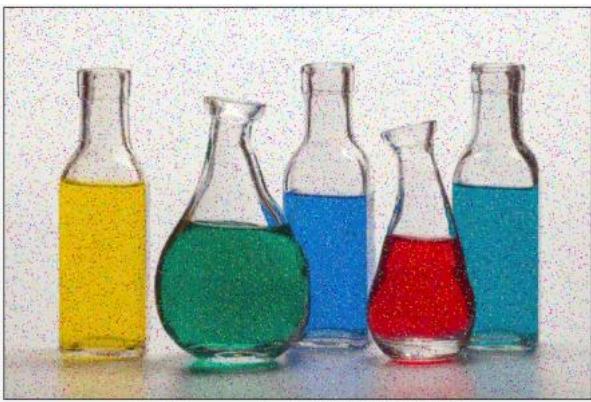
Inpaint all connected regions having an area less than specified value.

### Default values:

`maximal_area=4`, `tolerance=0` and `is_high_connectivity=0`.

## Example of use:

```
$ gmic image.jpg noise 5%,2 +inpaint_holes 8,40
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

# inpaint\_matchpatch

## Arguments:

- `[mask], _nb_scales={ 0=auto | >0 }, _patch_size>0, _nb_iterations_per_scale>0, _blend_size>=0, _allow_outer_blend: { 0 | 1 }, _is_already_initialized={ 0 | 1 }`

## Description:

Inpaint selected images by specified binary mask, using a multi-scale matchpatch algorithm.

## Default values:

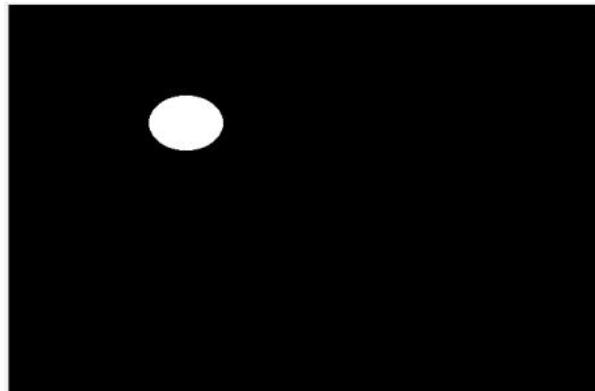
`nb_scales=0, patch_size=9, nb_iterations_per_scale=10, blend_size=5, allow_outer_blending=1` and `is_already_initialized=0`.

## Example of use:

```
$ gmic image.jpg 100%,100% ellipse[-1] 30%,30%,40,30,0,1,255  
+inpaint_matchpatch[0] [1]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: '[unnamed]' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x3)

---

# inpaint\_morpho

## Arguments:

- [mask]

## Description:

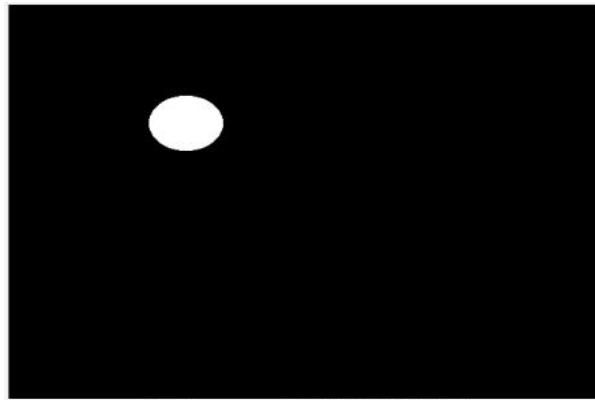
Inpaint selected images by specified mask using morphological operators.

## Example of use:

```
$ gmic image.jpg 100%,100% ellipse[-1] 30%,30%,40,30,0,1,255  
+inpaint_morpho[0] [1]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: '[unnamed]' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x3)

---

## inpaint\_pde

## Arguments:

- [mask],\_nb\_scales[%]>=0,\_diffusion\_type={ 0=isotropic | 1=Delaunay-guided | 2=edge-guided | 3=mask-guided },\_diffusion\_iter>=0

## Description:

Inpaint selected images by specified mask using a multiscale transport-diffusion algorithm.

If 'diffusion type==3', non-zero values of the mask (e.g. a distance function) are used to guide the diffusion process.

## Default values:

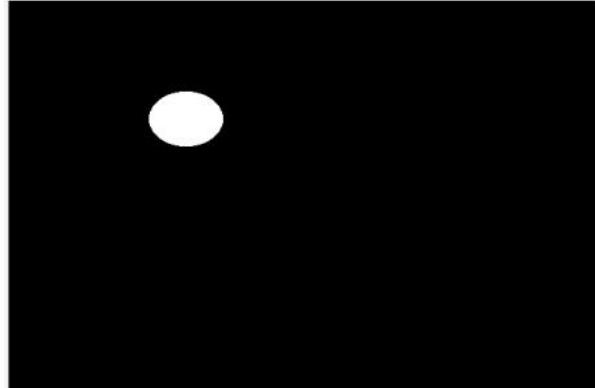
`nb_scales=75%`, `diffusion_type=1` and `diffusion_iter=20`.

## Example of use:

```
$ gmic image.jpg 100%,100% ellipse[-1] 30%,30%,40,30,0,1,255  
+inpaint_pde[0] [1]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: '[unnamed]' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x3)

---

## input

[Built-in command](#)

### Arguments:

- `[type:]filename` or
- `[type:]http://URL` or
- `[selection]x_nb_copies>0` or
- `{ width>0[%] | [image_w] ,{ _height>0[%] | [image_h] },{ _depth>0[%] | [image_d] },{ _spectrum>0[%] | [image_s] },_{ value1,_value2,... | 'formula' }`  or
- `(value1{|;|/|^}value2{|;|/|^}...[:{x|y|z|c}|;|/|^])` or
- `0`

### Description:

Insert a new image taken from a filename or from a copy of an existing image [index],

or insert new image with specified dimensions and values. Single quotes may be omitted in **formula**. Specifying argument **0** inserts an **empty** image.

(equivalent to shortcut command [i](#)).

## Default values:

**nb\_copies=1**, **height=depth=spectrum=1** and **value1=0**.

This command has a [tutorial page](#).

## Examples of use:

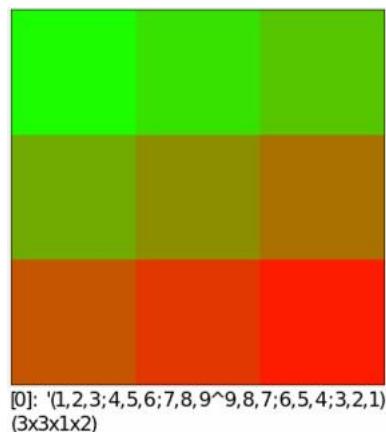
- **Example #1**

```
$ gmic input image.jpg
```



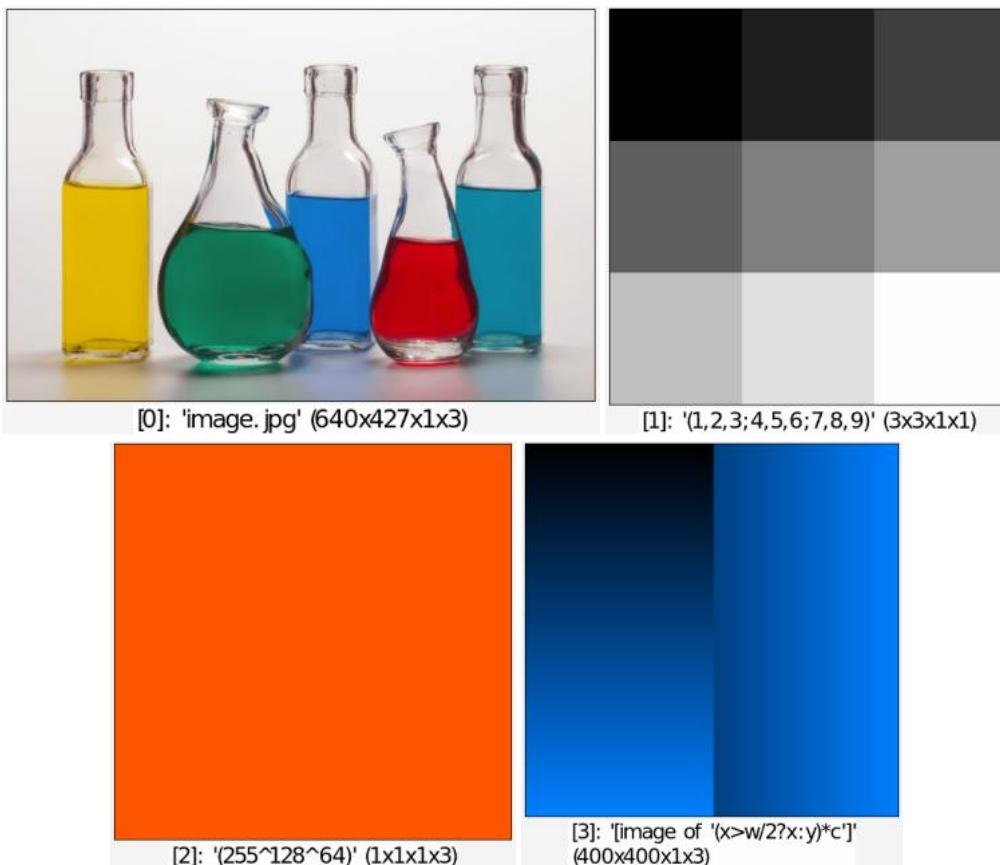
- **Example #2**

```
$ gmic input (1,2,3;4,5,6;7,8,9^9,8,7;6,5,4;3,2,1)
```



- **Example #3**

```
$ gmic image.jpg (1,2,3;4,5,6;7,8,9) (255^128^64)  
400,400,1,3,'(x>w/2?x:y)*c'
```



## input\_565

### Arguments:

- `filename, width>0, height>0, reverse_endianness={ 0 | 1 }`

### Description:

Insert image data from a raw RGB-565 file, at the end of the list.

### Default values:

`reverse_endianness=0`.

## input\_cached

### Arguments:

- `"basename.ext", _try_downloading_from_gmic_server={ 0 | 1 }`

### Description:

Input specified filename, assumed to be stored in one of the G'MIC resource folder.

If file not found and `try_downloading=1`, file is downloaded from the G'MIC server and stored in the  `${-path_cache}` folder.

## Default values:

```
try_downloading_from_gmic_server=1.
```

---

# input\_csv

## Arguments:

- `"filename", _read_data_as={ 0=numbers | 1=strings | _variable_name }`

## Description:

Insert number of string array from specified .csv file.

If `variable_name` is provided, the string of each cell is stored in a numbered variable `_variable_name_x_y`, where `x` and `y` are the indices of the cell column and row respectively (starting from `0`).

Otherwise, a `WxH` image is inserted at the end of the list, with each vector-valued pixel `I(x,y)` encoding the number or the string of each cell.

This command returns the `W,H` dimension of the read array, as the status.

## Default values:

```
read_data_as=1.
```

---

# input\_cube

## Arguments:

- `"filename", _convert_1d_cluts_to_3d={ 0 | 1 }.`

## Description:

Insert CLUT data from a .cube filename (Adobe CLUT file format).

## Default values:

```
convert_1d_cluts_to_3d=1.
```

---

# input\_flo

## **Arguments:**

- `"filename"`

## **Description:**

Insert optical flow data from a .flo filename (vision.middlebury.edu file format).

---

## input\_glob

### **Arguments:**

- `pattern`

### **Description:**

Insert new images from several filenames that match the specified glob pattern.

(*equivalent to shortcut command* `ig`).

---

## input\_gpl

### **Arguments:**

- `filename`

### **Description:**

Input specified filename as a .gpl palette data file.

---

## input\_text

### **Arguments:**

- `filename`

### **Description:**

Input specified text-data filename as a new image.

(*equivalent to shortcut command* `it`).

---

## inrange

## Arguments:

- `min[%],max[%],_include_min_boundary={ 0=no | 1=yes },_include_max_boundary={ 0=no | 1=yes }`

## Description:

Detect pixels whose values are in specified range `[min,max]`, in selected images.

(*equivalent to shortcut command* `ir`).

## Default values:

`_include_min_boundary=_include_max_boundary=1`.

## Example of use:

```
$ gmic image.jpg +inrange 25%,75%
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

## int2rgb

### No arguments

## Description:

Convert color representation of selected images from INT24 to RGB.

## invert

Built-in command

## Arguments:

- `solver={ 0=SVD | 1=LU }`

## Description:

Compute the inverse of the selected matrices.

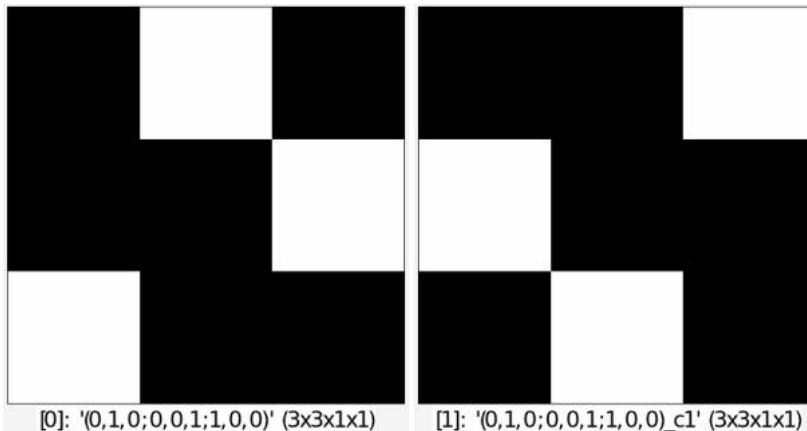
SVD solver is slower but less numerically instable than LU.

## Default values:

`solver=1`.

## Example of use:

```
$ gmic (0,1,0;0,0,1;1,0,0) +invert
```



---

## is\_3d

No arguments

## Description:

Return 1 if all of the selected images are 3D objects, 0 otherwise.

---

## is\_change

## Arguments:

- `_value={ 0=false | 1=true }`

## Description:

Set or unset the `is_change` flag associated to the image list.

This flag tells the interpreter whether or not the image list should be displayed when the pipeline ends.

## Default values:

`value=1`.

---

## is\_ext

### Arguments:

- `filename,_extension`

### Description:

Return 1 if specified filename has a given extensioin.

---

## is\_half

### No arguments

### Description:

Return 1 if the type of image pixels is limited to half-float.

---

## is\_image\_arg

### Arguments:

- `string`

### Description:

Return 1 if specified string looks like `[ind]`.

---

## is\_macos

### No arguments

### Description:

Return 1 if current computer OS is Darwin (MacOS), 0 otherwise.

---

## is\_pattern

## Arguments:

- `string`

## Description:

Return 1 if specified string looks like a drawing pattern `0x.....`.

---

# is\_percent

## Arguments:

- `string`

## Description:

Return 1 if specified string ends with a `%`, 0 otherwise.

---

# is\_videofilename

## No arguments

## Description:

Return 1 if extension of specified filename is typical from video files.

---

# is\_windows

## No arguments

## Description:

Return 1 if current computer OS is Windows, 0 otherwise.

---

# isoline3d

Built-in command

## Arguments:

- `isovalue[%]` or
- `'formula',value,_x0,_y0,_x1,_y1,_size_x>0[%],_size_y>0[%]`

## Description:

Extract 3D isolines with specified value from selected images or from specified formula.

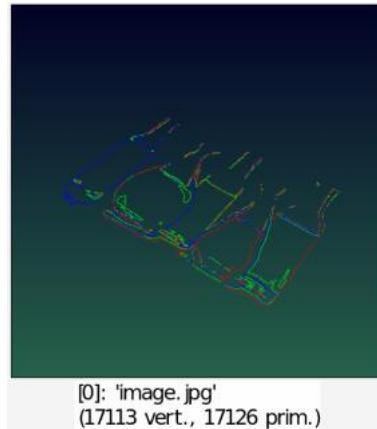
## Default values:

`x0=y0=-3`, `x1=y1=3` and `size_x=size_y=256`.

## Examples of use:

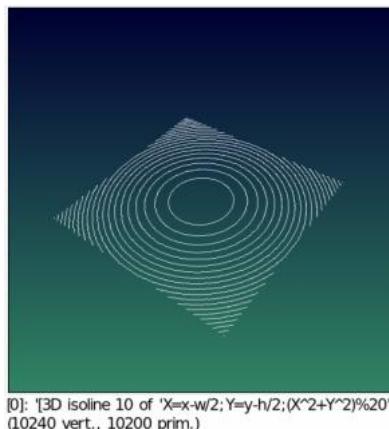
- Example #1

```
$ gmic image.jpg blur 1 isoline3d 50%
```



- Example #2

```
$ gmic isoline3d 'X=x-w/2;Y=y-h/2;(X^2+Y^2)%20',10,-10,-10,10,10
```



---

## isophotes

### Arguments:

- `_nb_levels>0`

### Description:

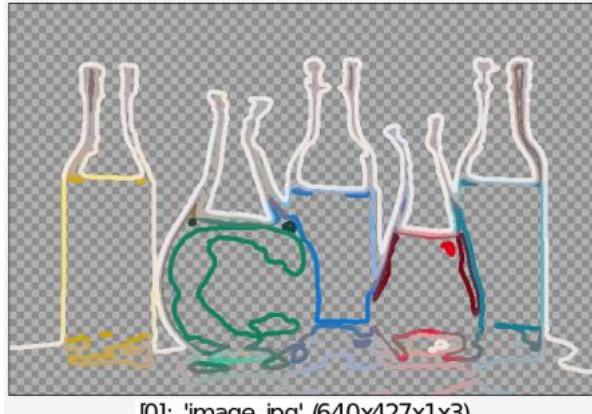
Render isophotes of selected images on a transparent background.

## Default values:

`nb_levels=64`

## Example of use:

```
$ gmic image.jpg blur 2 isophotes 6 dilate_circ 5 display_rgba
```



---

## isosurface3d

Built-in command

### Arguments:

- `isovalue[%]` or
- `'formula',value,_x0,_y0,_z0,_x1,_y1,_z1,_size_x>0[%],_size_y>0[%],_size_z>0[`

### Description:

Extract 3D isosurfaces with specified value from selected images or from specified formula.

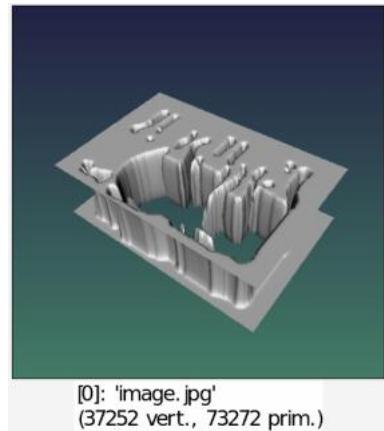
### Default values:

`x0=y0=z0=-3`, `x1=y1=z1=3` and `size_x=size_y=size_z=32`.

## Examples of use:

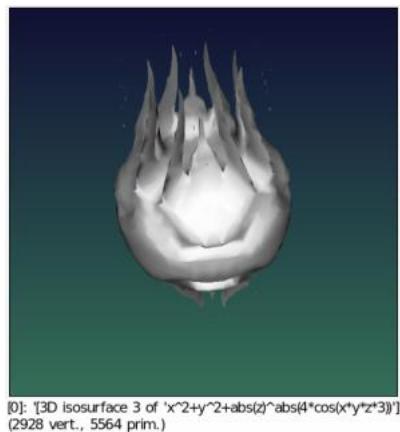
- Example #1

```
$ gmic image.jpg resize2dy 128 luminance threshold 50% expand_z 2,0  
blur 1 isosurface3d 50% mul3d 1,1,30
```



- **Example #2**

```
$ gmic isosurface3d 'x^2+y^2+abs(z)^abs(4*cos(x*y*z*3))',3
```



---

## jzazbz2rgb

### Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

### Description:

Convert color representation of selected images from RGB to Jzazbz.

### Default values:

`illuminant=2.`

---

## jzazbz2xyz

### No arguments

## Description:

Convert color representation of selected images from RGB to XYZ.

---

# kaleidoscope

## Arguments:

- `_center_x[%],_center_y[%],_radius,_angle,_boundary_conditions={0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

## Description:

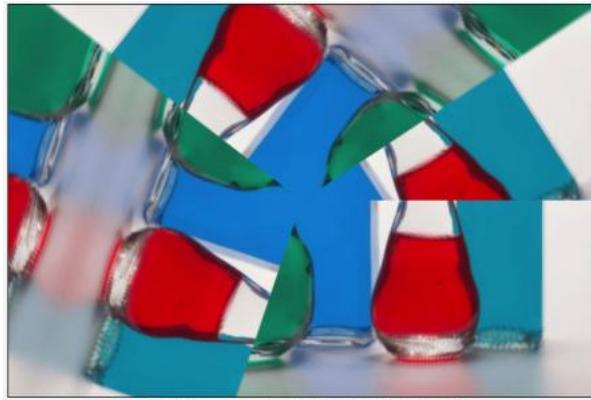
Create kaleidoscope effect from selected images.

## Default values:

`center_x=center_y=50%`, `radius=100`, `angle=30` and `boundary_conditions=3`.

## Example of use:

```
$ gmic image.jpg kaleidoscope ,
```



[0]: 'image.jpg' (640x427x1x3)

---

# keep

Built-in command

## No arguments

## Description:

Keep only selected images.

(equivalent to shortcut command `k`).

## Examples of use:

- Example #1

```
$ gmic image.jpg split x keep[0-50%:2] append x
```



[0]: 'image.jpg'  
(161x427x1x3)

- **Example #2**

```
$ gmic image.jpg split x keep[^30%-70%] append x
```



[0]: 'image.jpg' (384x427x1x3)

---

## kuwahara

### Arguments:

- `size>0`

### Description:

Apply Kuwahara filter of specified size on selected images.

### Example of use:

```
$ gmic image.jpg kuwahara 9
```



## laar

**No arguments**

**Description:**

Extract the largest axis-aligned rectangle in non-zero areas of selected images.

Rectangle coordinates are returned in status, as a sequence of numbers x0,y0,x1,y1.

**Example of use:**

```
$ gmic shape_cupid 256 coords=${-laar} normalize 0,255 to_rgb  
rectangle $coords,0.5,0,128,0
```



## lab2lch

**No arguments**

**Description:**

Convert color representation of selected images from Lab to Lch.

---

## lab2rgb

### Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

### Description:

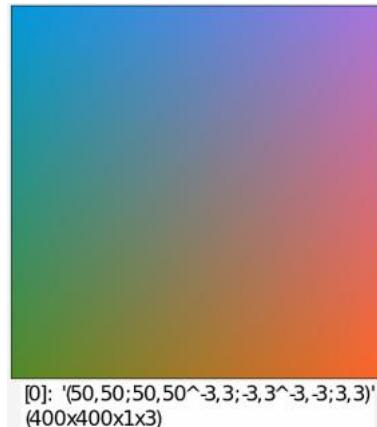
Convert color representation of selected images from Lab to RGB.

### Default values:

`illuminant=2`.

### Example of use:

```
$ gmic (50,50;50,50^-3,3;-3,3^-3,-3;3,3) resize 400,400,1,3,3 lab2rgb
```



## lab2srgb

### Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

### Description:

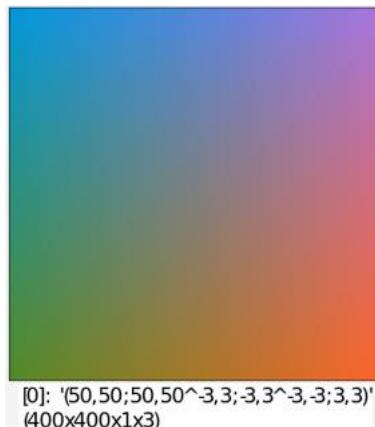
Convert color representation of selected images from Lab to sRGB.

### Default values:

`illuminant=2`.

## Example of use:

```
$ gmic (50,50;50,50^-3,3;-3,3^-3,-3;3,3) resize 400,400,1,3,3 lab2rgb
```



---

## lab2xyz

### Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

### Description:

Convert color representation of selected images from Lab to XYZ.

### Default values:

`illuminant=2`.

---

## lab82rgb

### Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

### Description:

Convert color representation of selected images from Lab8 to RGB.

### Default values:

`illuminant=2`.

---

# lab2srgb

## Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

## Description:

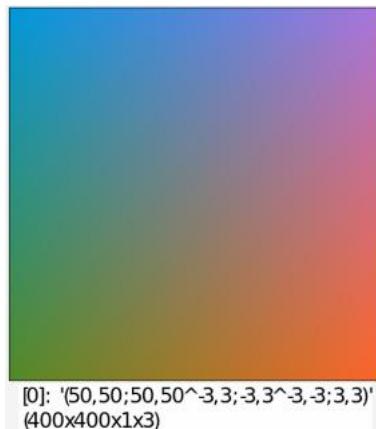
Convert color representation of selected images from Lab8 to sRGB.

## Default values:

`illuminant=2`.

## Example of use:

```
$ gmic (50,50;50,50^-3,3;-3,3^-3,-3;3,3) resize 400,400,1,3,3 lab2rgb
```



---

# label

Built-in command

## Arguments:

- `_tolerance>=0, is_high_connectivity={ 0 | 1 }, _is_L2_norm={ 0 | 1 }`

## Description:

Label connected components in selected images.

## Default values:

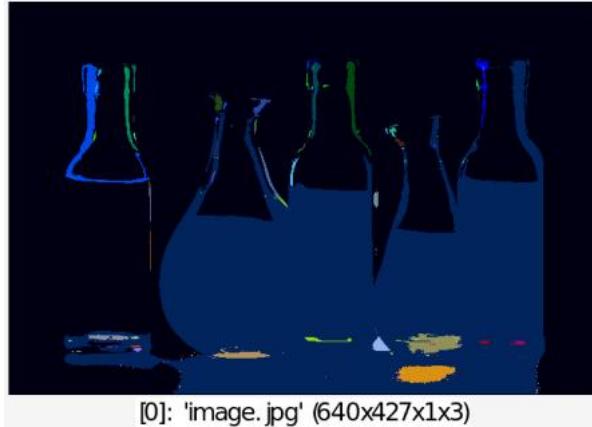
`tolerance=0`, `is_high_connectivity=0` and `is_L2_norm=1`.

This command has a [tutorial page](#).

## Examples of use:

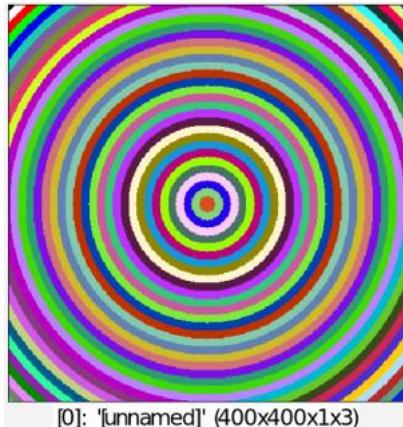
- **Example #1**

```
$ gmic image.jpg luminance threshold 60% label normalize 0,255 map 0
```



- **Example #2**

```
$ gmic 400,400 set 1,50%,50% distance 1 mod 16 threshold 8 label mod 255 map 2
```



---

## label3d

### Arguments:

- "text",  
font\_height>=0,  
\_opacity,  
\_color1,...

### Description:

Generate 3D text label.

### Default values:

`font_height=13`, `opacity=1` and `color=255,255,255`.

---

## label\_fg

### Arguments:

- `tolerance>=0, is_high_connectivity={ 0 | 1 }`

### Description:

Label connected components for non-zero values (foreground) in selected images.

Similar to `label` except that 0-valued pixels are not labeled.

### Default values:

`is_high_connectivity=0`.

---

## label\_points3d

### Arguments:

- `_label_size>0, _opacity`

### Description:

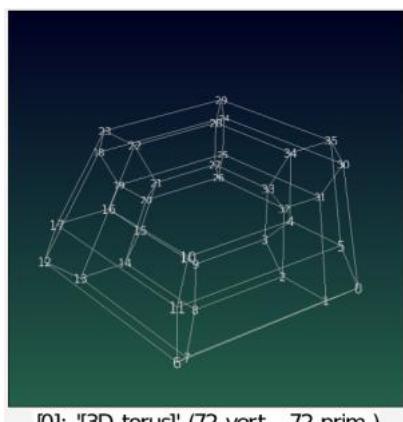
Add a numbered label to all vertices of selected 3D objects.

### Default values:

`label_size=13` and `opacity=0.8`.

### Example of use:

```
$ gmic torus3d 100,40,6,6 label_points3d 23,1 mode3d 1
```



---

# laplacian

**No arguments**

**Description:**

Compute Laplacian of selected images.

**Example of use:**

```
$ gmic image.jpg laplacian
```



---

# lathe3d

**Arguments:**

- `_resolution>0, _smoothness[%]>=0, _max_angle>=0`

**Description:**

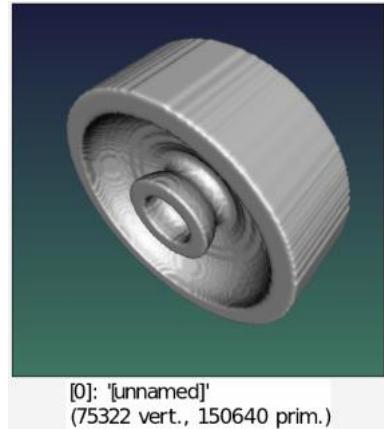
Generate 3D object from selected binary XY-profiles.

**Default values:**

`resolution=128, smoothness=0.5%` and `max_angle=361`.

**Example of use:**

```
$ gmic 300,300 rand -1,1 blur 40 sign normalize 0,255 lathe3d ,
```



---

## lch2lab

**No arguments**

**Description:**

Convert color representation of selected images from Lch to Lab.

---

## lch2rgb

**Arguments:**

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

**Description:**

Convert color representation of selected images from Lch to RGB.

**Default values:**

`illuminant=2`.

---

## lch82rgb

**Arguments:**

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

**Description:**

Convert color representation of selected images from Lch8 to RGB.

## Default values:

`illuminant=2`.

---

le

Built-in command

## Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

## Description:

Compute the boolean 'less or equal than' of selected images with specified value, image or mathematical expression, or compute the boolean 'less or equal than' of selected images.

(equivalent to shortcut command `<=`).

## Examples of use:

- Example #1

```
$ gmic image.jpg le {ia}
```



- Example #2

```
$ gmic image.jpg +mirror x le
```



---

## lic

### Arguments:

- `_amplitude>0, _channels>0`

### Description:

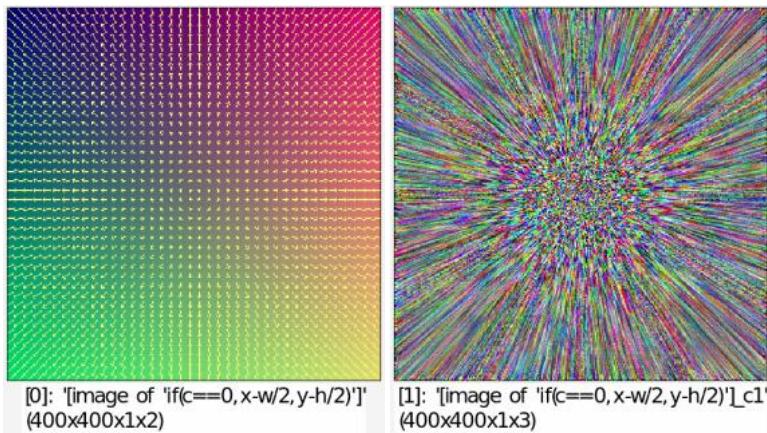
Render LIC representation of selected vector fields.

### Default values:

`amplitude=30` and `channels=1`.

### Example of use:

```
$ gmic 400,400,1,2,'if(c==0,x-w/2,y-h/2)' +lic 200,3 quiver[-2]
[-2],10,1,1,1,255
```



---

## light3d

Built-in command

### Arguments:

- `position_x,position_y,position_z` or
- `[texture]` or
- `(no arg)`

## Description:

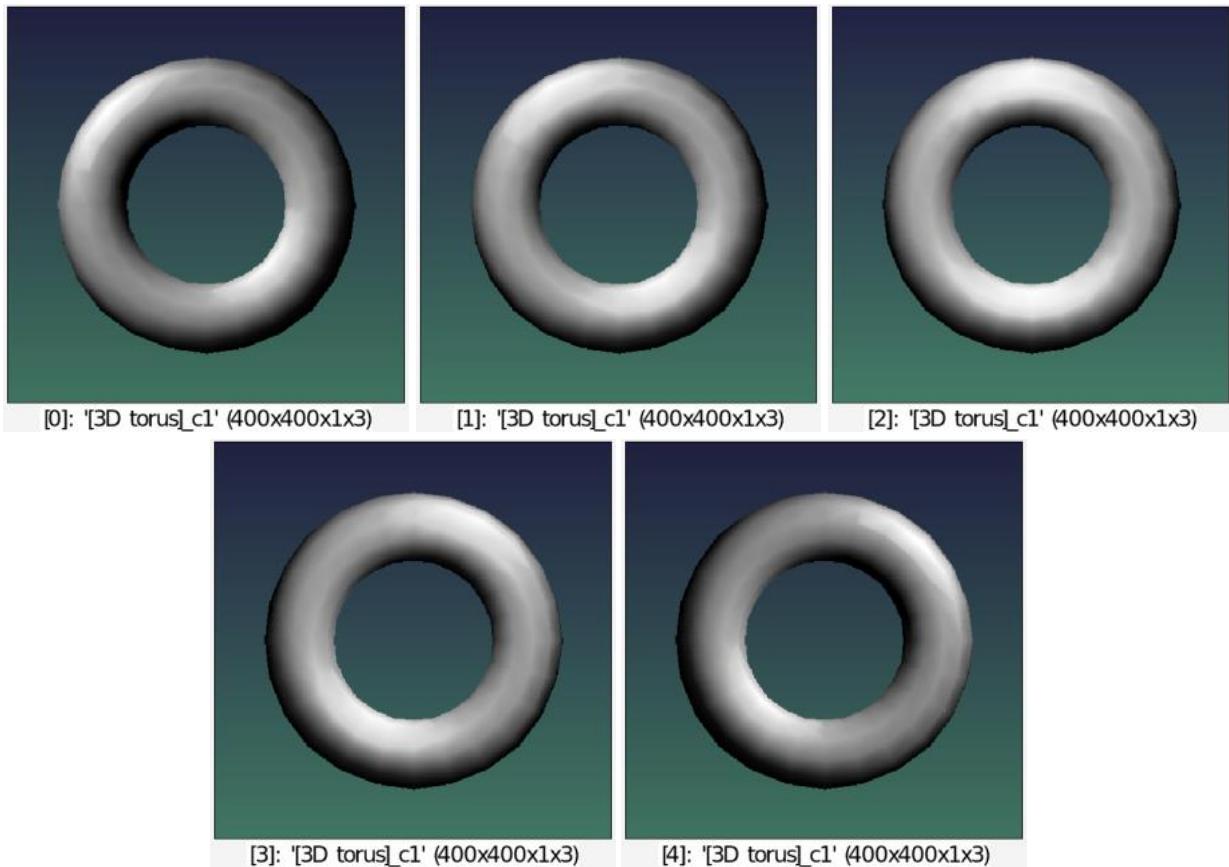
Set the light coordinates or the light texture for 3D rendering.

(*equivalent to shortcut command* `l3d`).

(no arg) resets the 3D light to default.

## Example of use:

```
$ gmic torus3d 100,30 double3d 0 specs3d 1.2 repeat 5 light3d
{$>*100},0,-300 +snapshot3d[0] 400 done remove[0]
```



## light\_patch

### Arguments:

- `_density>0,_darkness>=0,_lightness>=0`

## Description:

Add light patches to selected images.

## Default values:

`density=10`, `darkness=0.9` and `lightness=1.7`.

## Example of use:

```
$ gmic image.jpg +light_patch 20,0.9,4
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## light\_relief

### Arguments:

- `_ambient_light,_specular_lightness,_specular_size,_darkness,_light_smoothnes  
{ 0 | 1 }`

### Description:

Apply relief light to selected images.

Default values(s): `ambient_light=0.3`, `specular_lightness=0.5`, `specular_size=0.2`,  
`darkness=0`, `xl=0.2`, `yl=zl=0.5`,  
`zscale=1`, `opacity=1` and `opacity_is_heightmap=0`.

## Example of use:

```
$ gmic image.jpg blur 2 light_relief 0.3,4,0.1,0
```



---

## lightness

No arguments

### Description:

Compute lightness of selected sRGB images.

### Example of use:

```
$ gmic image.jpg +lightness
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x1)

---

## lightrays

### Arguments:

- `100<=_density<=0,_center_x[%],_center_y[%],_ray_length>=0,_ray_attenuation>=`

### Description:

Generate ray lights from the edges of selected images.

### Default values:

`density=50%`, `center_x=50%`, `center_y=50%`, `ray_length=0.9` and `ray_attenuation=0.5`.

## Example of use:

```
$ gmic image.jpg +lightrays , + cut 0,255
```



[0]: 'image.jpg' (640x427x1x3)

---

# line

Built-in command

## Arguments:

- `x0[%],y0[%],x1[%],y1[%],_opacity,_pattern,_color1,...`

## Description:

Draw specified colored line on selected images.

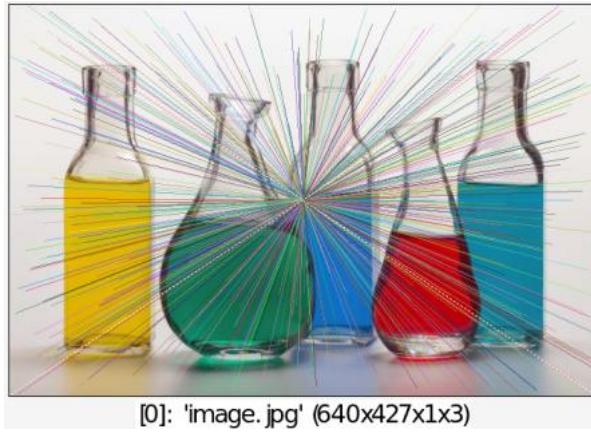
`pattern` is an hexadecimal number starting with `0x` which can be omitted even if a color is specified.

## Default values:

`opacity=1`, `pattern=(undefined)` and `color1=0`.

## Example of use:

```
$ gmic image.jpg repeat 500 line 50%,50%,{u(w)},{u(h)},0.5,${-rgb}
done line 0,0,100%,100%,1,0xFFFFFFFF,255 line
100%,0,0,100%,1,0xFFFFFFFF,255
```



---

## line3d

### Arguments:

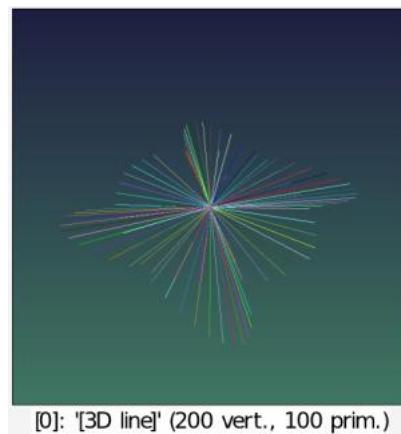
- `x0,y0,z0,x1,y1,z1`

### Description:

Input 3D line at specified coordinates.

### Example of use:

```
$ gmic repeat 100 a={$>*pi/50} line3d 0,0,0,{cos(3*a)},{sin(2*a)},0  
color3d. ${-rgb} done add3d
```



---

## linearize\_tiles

### Arguments:

- `M>0, _N>0`

### Description:

Linearize MxN tiles on selected images.

## Default values:

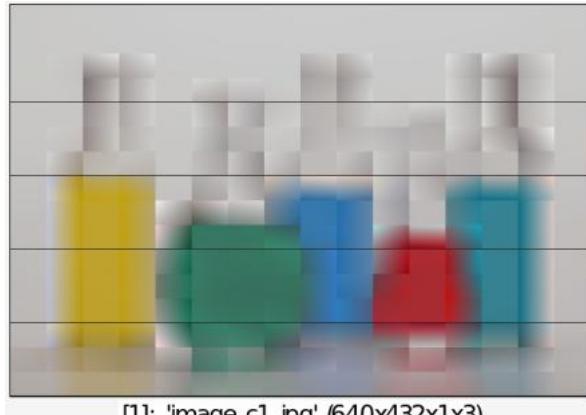
N=M .

## Example of use:

```
$ gmic image.jpg +linearize_tiles 16
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x432x1x3)

---

## linethick

### Arguments:

- `x0[%],y0[%],x1[%],y1[%],_thickness,_opacity,_color1`

### Description:

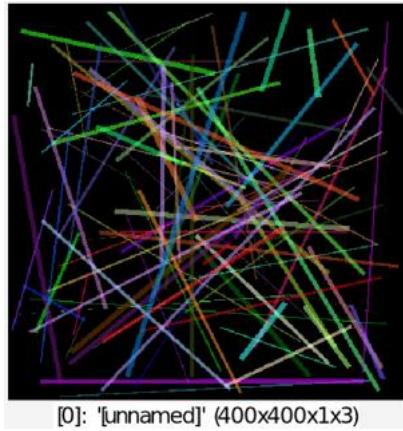
Draw specified colored thick line on selected images.

## Default values:

`thickness=2`, `opacity=1` and `color1=0`.

## Example of use:

```
$ gmic 400,400,1,3 repeat 100 linethick {u([w,h,w,h,5])},0.5,${-rgb}  
done
```



## linify

### Arguments:

- `0<=_density<=100, _spreading>=0, _resolution[%]>0, _line_opacity>=0, _line_precision>0`  
`{ 0=subtractive | 1=additive }`

### Description:

Apply linify effect on selected images.

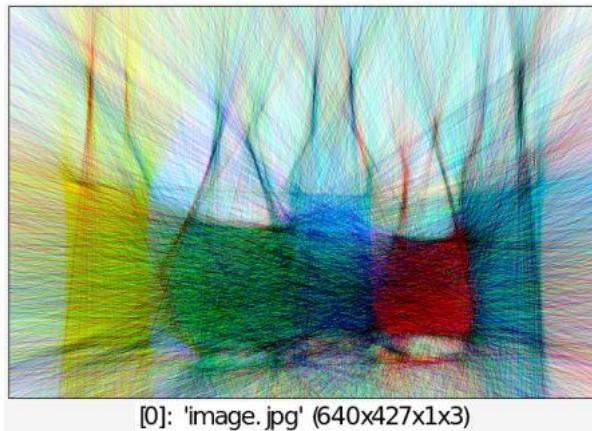
The algorithm is inspired from the one described on the webpage <http://linify.me/about>.

### Default values:

`density=50, spreading=2, resolution=40%, line_opacity=10, line_precision=24`  
and `mode=0`.

### Example of use:

```
$ gmic image.jpg linify 60
```



## lissajous3d

## Arguments:

- `resolution>1,a,A,b,B,c,C`

## Description:

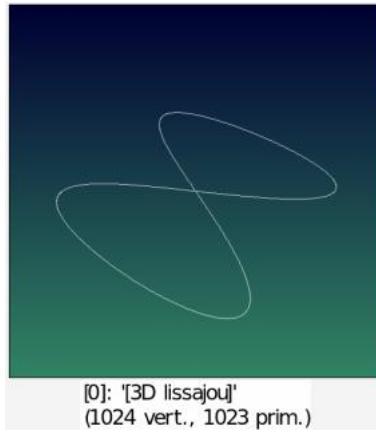
Input 3D lissajous curves ( $x(t)=\sin(at+A*2\pi)$ ,  $y(t)=\sin(bt+B*2\pi)$ ,  $z(t)=\sin(ct+C*2\pi)$ ).

## Default values:

`resolution=1024`, `a=2`, `A=0`, `b=1`, `B=0`, `c=0` and `C=0`.

## Example of use:

```
$ gmic lissajous3d ,
```



---

## local

Built-in command

### No arguments

## Description:

Start a `local...[onfail]...endlocal` block, with selected images.

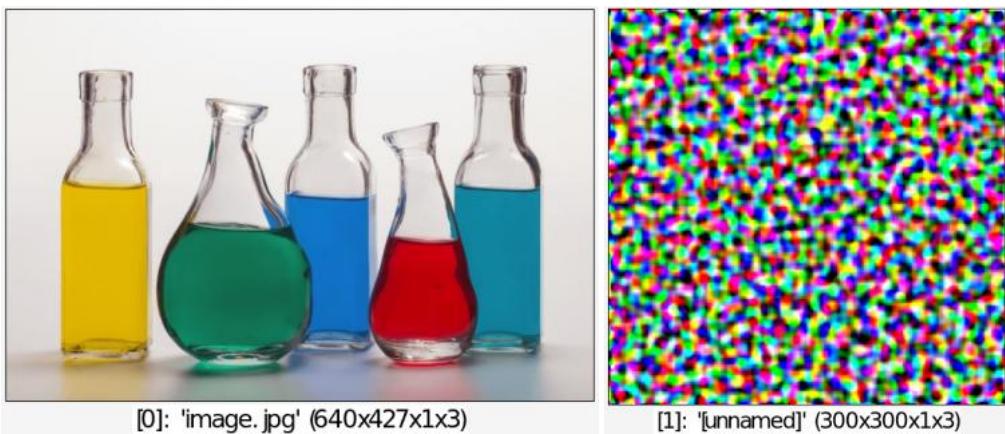
(*equivalent to shortcut command `l`*).

This command has a [tutorial page](#).

## Examples of use:

- **Example #1**

```
$ gmic image.jpg local[] 300,300,1,3 rand[0] 0,255 blur 4 sharpen  
1000 endlocal
```



[0]: 'image.jpg' (640x427x1x3)

[1]: '[unnamed]' (300x300x1x3)

- **Example #2**

```
$ gmic image.jpg +local repeat 3 deform 20 done endlocal
```



[0]: 'image.jpg' (640x427x1x3)

[1]: 'image\_c1.jpg' (640x427x1x3)

---

## log

Built-in command

**No arguments**

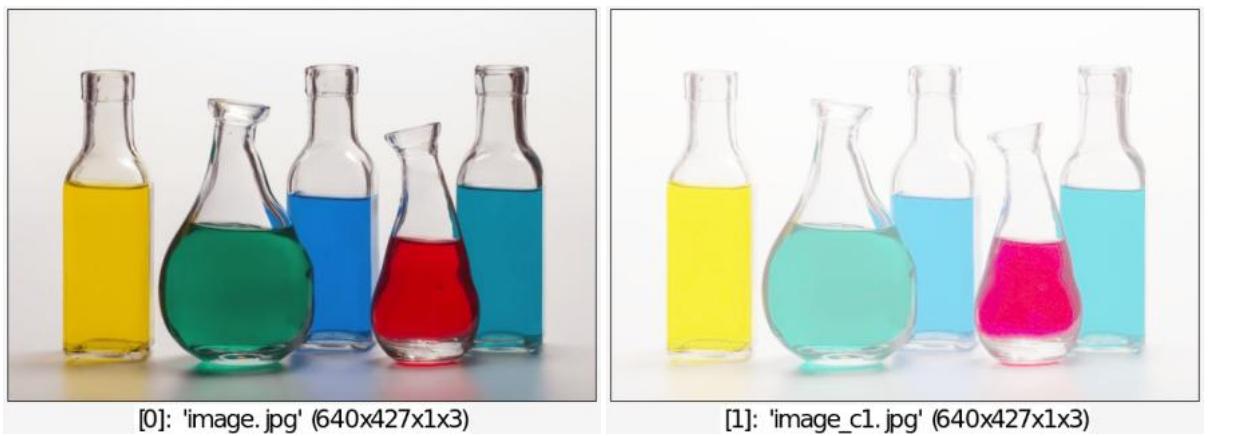
**Description:**

Compute the pointwise base-e logarithm of selected images.

**Examples of use:**

- **Example #1**

```
$ gmic image.jpg +add 1 log[-1]
```

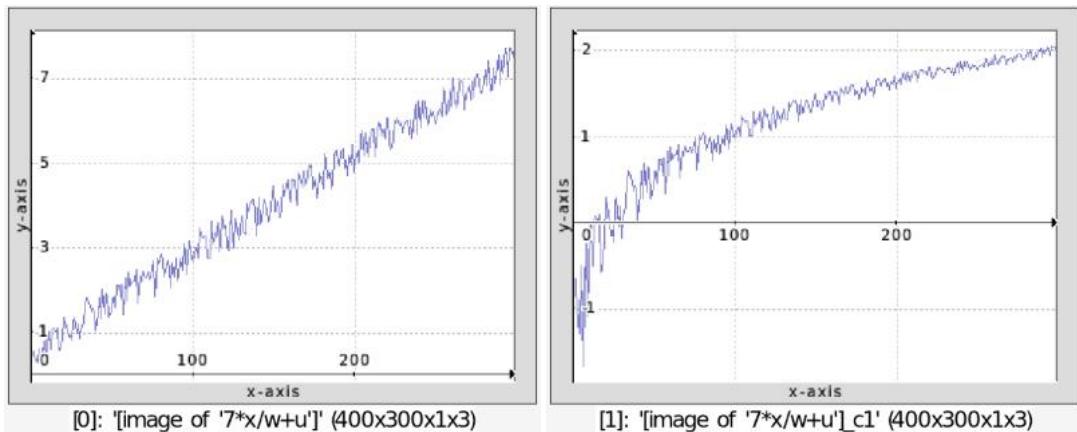


[0]: 'image.jpg' (640x427x1x3)

[1]: 'image\_c1.jpg' (640x427x1x3)

- **Example #2**

```
$ gmic 300,1,1,1,'7*x/w+u' +log display_graph 400,300
```



[0]: '[image of "7\*x/w+u"]' (400x300x1x3)

[1]: '[image of "7\*x/w+u"]\_c1' (400x300x1x3)

## log10

Built-in command

**No arguments**

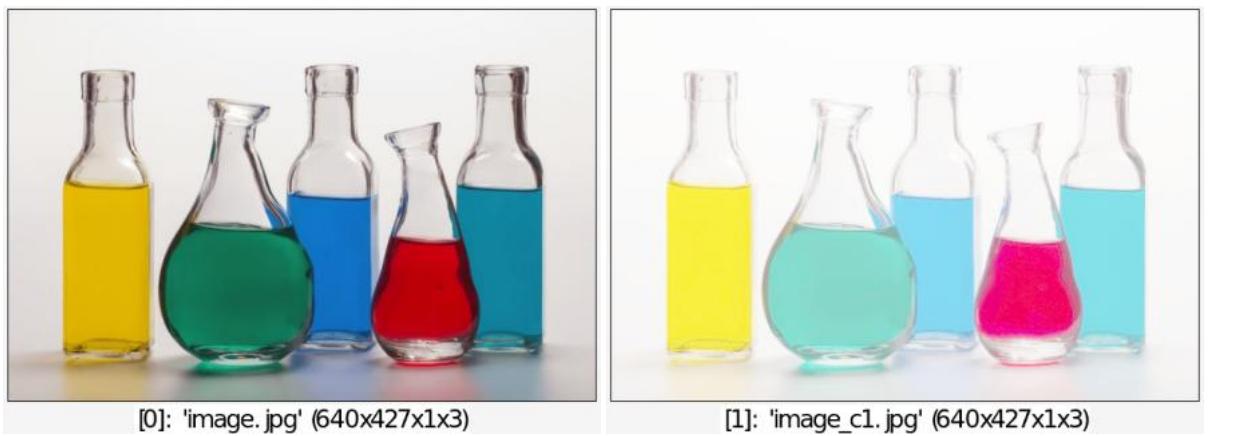
**Description:**

Compute the pointwise base-10 logarithm of selected images.

**Examples of use:**

- **Example #1**

```
$ gmic image.jpg +add 1 log10[-1]
```

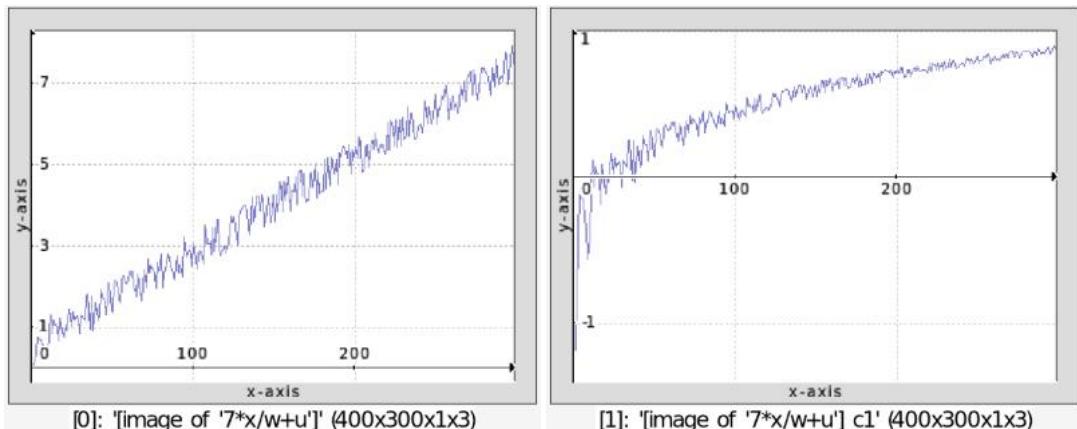


[0]: 'image.jpg' (640x427x1x3)

[1]: 'image\_c1.jpg' (640x427x1x3)

- **Example #2**

```
$ gmic 300,1,1,1,'7*x/w+u' +log10 display_graph 400,300
```



[0]: '[image of "7\*x/w+u"]' (400x300x1x3)

[1]: '[image of "7\*x/w+u"]\_c1' (400x300x1x3)

## log2

Built-in command

**No arguments**

**Description:**

Compute the pointwise base-2 logarithm of selected images

**Examples of use:**

- **Example #1**

```
$ gmic image.jpg +add 1 log2[-1]
```



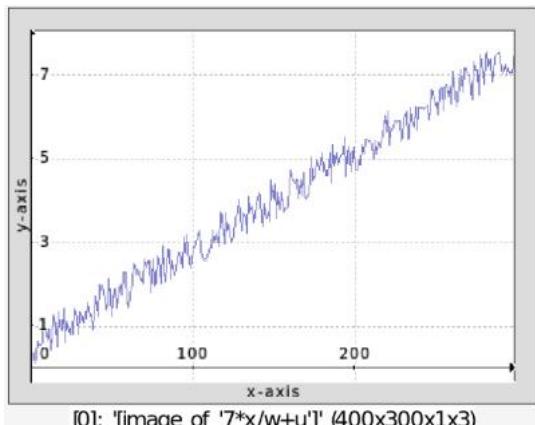
[0]: 'image.jpg' (640x427x1x3)



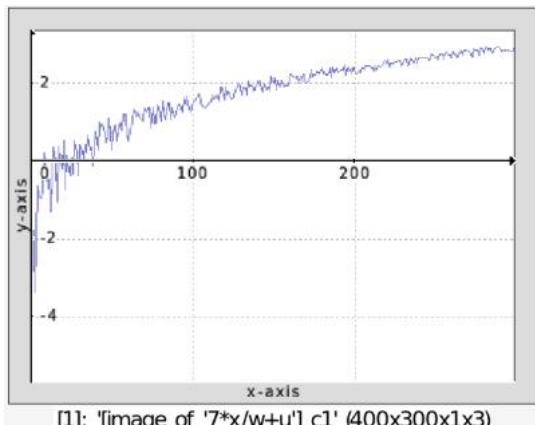
[1]: 'image\_c1.jpg' (640x427x1x3)

## • Example #2

```
$ gmic 300,1,1,1,'7*x/w+u' +log2 display_graph 400,300
```



[0]: '[image of "7\*x/w+u"]' (400x300x1x3)



[1]: '[image of "7\*x/w+u"]\_c1' (400x300x1x3)

---

lorem

## Arguments:

- `_width>0, _height>0`

## Description:

Input random image of specified size, retrieved from Internet.

## Default values:

`width=height=800`.

---

lt

Built-in command

## Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

## Description:

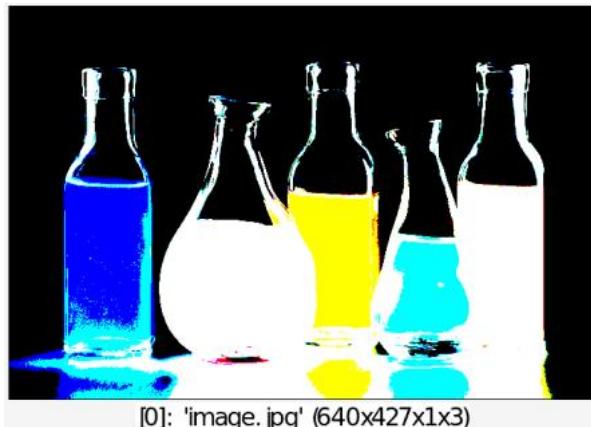
Compute the boolean 'less than' of selected images with specified value, image or mathematical expression, or compute the boolean 'less than' of selected images.

(*equivalent to shortcut command* `<`).

## Examples of use:

- Example #1

```
$ gmic image.jpg lt {ia}
```



- Example #2

```
$ gmic image.jpg +mirror x lt
```



## luminance

### No arguments

## Description:

Compute luminance of selected sRGB images.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg +luminance
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x1)

---

## lut\_contrast

### Arguments:

- `_nb_colors>1,_min_rgb_value`

## Description:

Generate a RGB colormap where consecutive colors have high contrast.

This function performs a specific score maximization to generate the result, so it may take some time when `nb_colors` is high.

## Default values:

`nb_colors=256` and `min_rgb_value=64`.

---

## mad

### No arguments

## Description:

Return the MAD (Maximum Absolute Deviation) of the last selected image.

The MAD is defined as  $MAD = \text{med}_i |x_i - \text{med}_j(x_j)|$

---

## mandelbrot

Built-in command

### Arguments:

- `z0r,z0i,z1r,zli,_iteration_max>=0,_is_julia={ 0 | 1 },_c0r,_c0i,_opacity`

### Description:

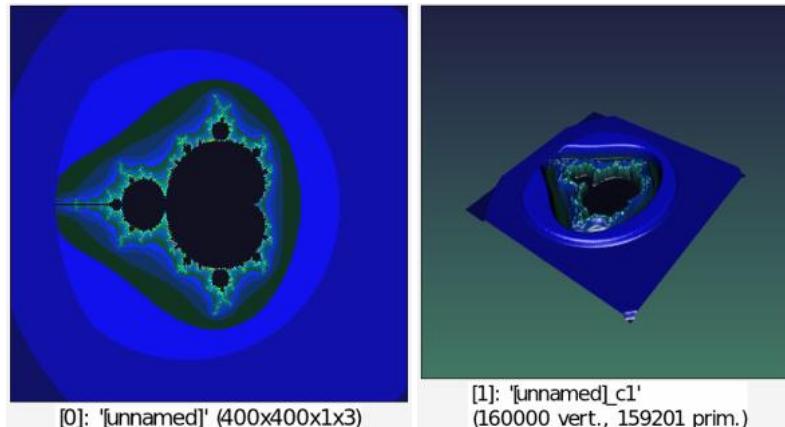
Draw mandelbrot/julia fractal on selected images.

### Default values:

`iteration_max=100`, `is_julia=0`, `c0r=c0i=0` and `opacity=1`.

### Example of use:

```
$ gmic 400,400 mandelbrot -2.5,-2,2,2,1024 map 0 +blur 2  
elevation3d[-1] -0.2
```



---

## map

Built-in command

### Arguments:

- `[palette], _boundary_conditions` or
- `palette_name, _boundary_conditions`

### Description:

Map specified vector-valued palette to selected indexed scalar images.

`palette_name` can be { `default` | `hsv` | `lines` | `hot` | `cool` | `jet` | `flag` | `cube` | `rainbow` | `algae` | `amp` | `balance` | `curl` | `deep` | `delta` | `dense` | `diff` | `gray` | `haline` | `ice` | `matter` | `oxy` | `phase` | `rain` | `solar` | `speed` | `tarn` | `tempo` |

*thermal | topo | turbid | aurora | hocuspocus | srb2 | uzebox }*  
`boundary_conditions` can be { 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }.

## Default values:

`boundary_conditions=0`.

This command has a [tutorial page](#).

## Examples of use:

- Example #1

```
$ gmic image.jpg +luminance map[-1] 3
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

- Example #2

```
$ gmic image.jpg +rgb2ycbcr split[-1] c (0,255,0) resize[-1] 256,1,1,1,3 map[-4] [-1] remove[-1] append[-3--1] c ycbcr2rgb[-1]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## map\_clut

### Arguments:

- `[clut] | "clut_name"`

## Description:

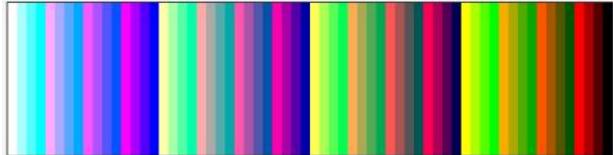
Map specified RGB color LUT to selected images.

## Example of use:

```
$ gmic image.jpg uniform_distribution {2^6},3 mirror[-1] x
+map_clut[0] [1]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: '[empty]' (64x1x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)

## map\_sphere

### Arguments:

- `_width>0, _height>0, _radius, _dilation>0, _fading>=0, _fading_power>=0`

## Description:

Map selected images on a sphere.

## Default values:

`width=height=512`, `radius=100`, `dilation=0.5`, `fading=0` and `fading_power=0.5`.

## Example of use:

```
$ gmic image.jpg map_sphere ,
```



---

## map\_sprites

### Arguments:

- `_nb_sprites>=1, _allow_rotation={ 0=none | 1=90 deg. | 2=180 deg. }`

### Description:

Map set of sprites (defined as the `nb_sprites` latest images of the selection) to other selected images,

according to the luminosity of their pixel values.

### Example of use:

```
$ gmic image.jpg resize2dy 48 repeat 16 ball {8+2*$>}, ${-rgb} mul[-1] {(1+$>)/16} done map_sprites 16
```



---

## map\_tones

## Arguments:

- `_threshold>=0, _gamma>=0, _smoothness>=0, nb_iter>=0`

## Description:

Apply tone mapping operator on selected images, based on Poisson equation.

## Default values:

`threshold=0.1`, `gamma=0.8`, `smoothness=0.5` and `nb_iter=30`.

## Example of use:

```
$ gmic image.jpg +map_tones ,
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## map\_tones\_fast

## Arguments:

- `_radius[%]>=0, _power>=0`

## Description:

Apply fast tone mapping operator on selected images.

## Default values:

`radius=3%` and `power=0.3`.

## Example of use:

```
$ gmic image.jpg +map_tones_fast ,
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## marble

### Arguments:

- `_image_weight, _pattern_weight, _angle, _amplitude, _sharpness>=0, _anisotropy>=0`

### Description:

Render marble like pattern on selected images.

### Default values:

`image_weight=0.2, pattern_weight=0.1, angle=45, amplitude=0, sharpness=0.4`  
and `anisotropy=0.8`,

`alpha=0.6, sigma=1.1` and `cut_low=cut_high=0`.

### Example of use:

```
$ gmic image.jpg +marble ,
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## matchpatch

Built-in command

## Arguments:

- `[patch_image],_patch_width>=1,_patch_height>=1,_patch_depth>=1,_nb_iterations { 0 | 1 },_[guide]`

## Description:

Estimate correspondence map between selected images and specified patch image, using a patch-matching algorithm.

Each pixel of the returned correspondence map gives the location (p,q) of the closest patch in the specified patch image. If `output_score=1`, the third channel also gives the corresponding matching score for each patch as well.

If `patch_penalization` is  $>=0$ , SSD is penalized with patch occurrences.

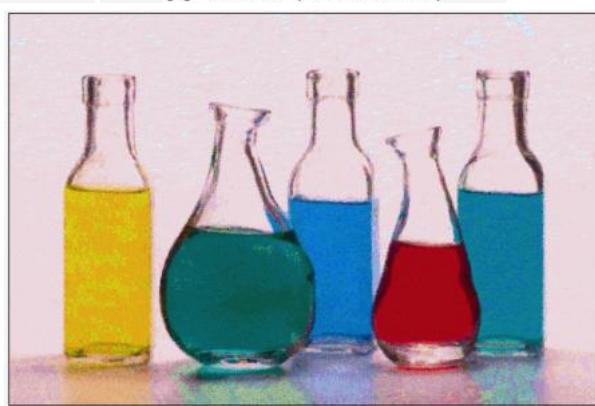
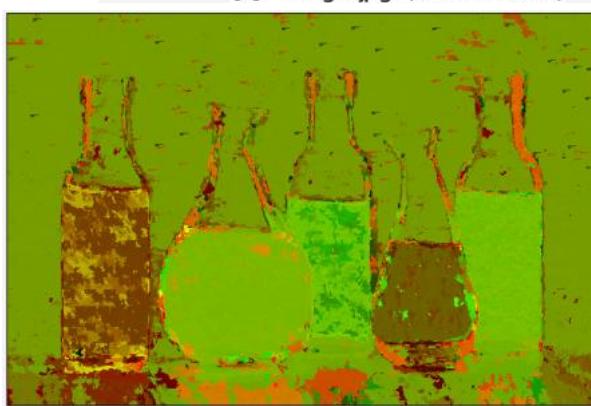
If `patch_penalization` is  $<0$ , SSD is inf-penalized when distance between patches are less than `-patch_penalization`.

## Default values:

`patch_height=patch_width`, `patch_depth=1`, `nb_iterations=5`, `nb_randoms=5`,  
`patch_penalization=0`, `output_score=0` and `guide=(undefined)`.

## Example of use:

```
$ gmic image.jpg sample colorful +matchpatch[0] [1],3 +warp[-2]  
[-1],0
```



# math\_lib

No arguments

## Description:

Return string that defines a set of several useful macros for the embedded math evaluator.

---

## max

Built-in command

### Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

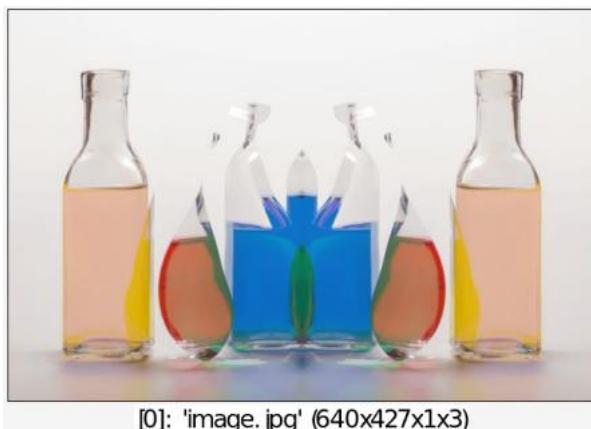
## Description:

Compute the maximum between selected images and specified value, image or mathematical expression, or compute the pointwise maxima between selected images.

## Examples of use:

- **Example #1**

```
$ gmic image.jpg +mirror x max
```



- **Example #2**

```
$ gmic image.jpg max 'R=((x/w-0.5)^2+(y/h-0.5)^2)^0.5;255*R'
```



---

## max\_d

**No arguments**

**Description:**

Return the maximal depth between selected images.

---

## max\_h

**No arguments**

**Description:**

Return the maximal height between selected images.

---

## max\_patch

**Arguments:**

- `_patch_size>=1`

**Description:**

Return locations of maximal values in local patch-based neighborhood of given size for selected images.

**Default values:**

`patch_size=16`.

**Example of use:**

```
$ gmic image.jpg norm +max_patch 16
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)

---

## max\_s

**No arguments**

**Description:**

Return the maximal spectrum between selected images.

---

## max\_w

**No arguments**

**Description:**

Return the maximal width between selected images.

---

## max\_wh

**No arguments**

**Description:**

Return the maximal w size of selected images.

---

## max\_whd

**No arguments**

**Description:**

Return the maximal wxhxd size of selected images.

---

## max\_whds

**No arguments**

### Description:

Return the maximal wxhxdxs size of selected images.

---

## maxabs

Built-in command

### Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

### Description:

Compute the maxabs between selected images and specified value, image or mathematical expression, or compute the pointwise maxabs between selected images.

---

## maze

### Arguments:

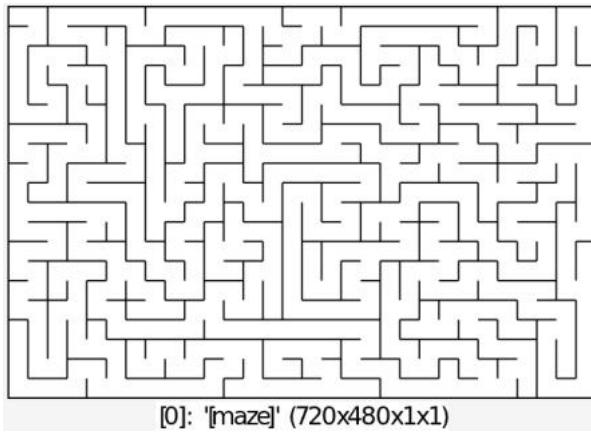
- `_width>0,_height>0,_cell_size>0`

### Description:

Input maze with specified size.

### Example of use:

```
$ gmic maze 30,20 negate normalize 0,255
```



---

## maze\_mask

### Arguments:

- `_cellsize>0`

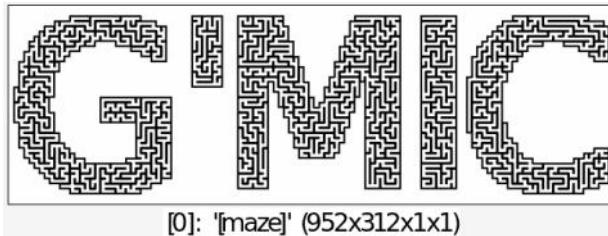
### Description:

Input maze according to size and shape of selected mask images.

Mask may contain disconnected shapes.

### Example of use:

```
$ gmic 0 text "G'MIC",0,0,53,1,1 dilate 3 autocrop 0 frame 1,1,0  
maze_mask 8 dilate 3 negate mul 255
```



---

## mdiv

Built-in command

### Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

### Description:

Compute the matrix division of selected matrices/vectors by specified value, image or mathematical expression, or compute the matrix division of selected images.

(equivalent to shortcut command `m/`).

---

## meancurvature\_flow

### Arguments:

- `_nb_iter>=0, _dt, _keep_sequence={ 0 | 1 }`

### Description:

Apply iterations of the mean curvature flow on selected images.

### Default values:

`nb_iter=10`, `dt=30` and `keep_sequence=0`.

### Example of use:

```
$ gmic image.jpg +meancurvature_flow 20
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

## med

### No arguments

### Description:

Compute the median of selected images.

### Example of use:

```
$ gmic image.jpg sample lena,lion,square +med
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'lena' (512x512x1x3)



[2]: 'lion' (640x600x1x3)



[3]: 'square' (750x500x1x3)



[4]: '[med]\_c1' (750x600x1x3)

# median

Built-in command

## Arguments:

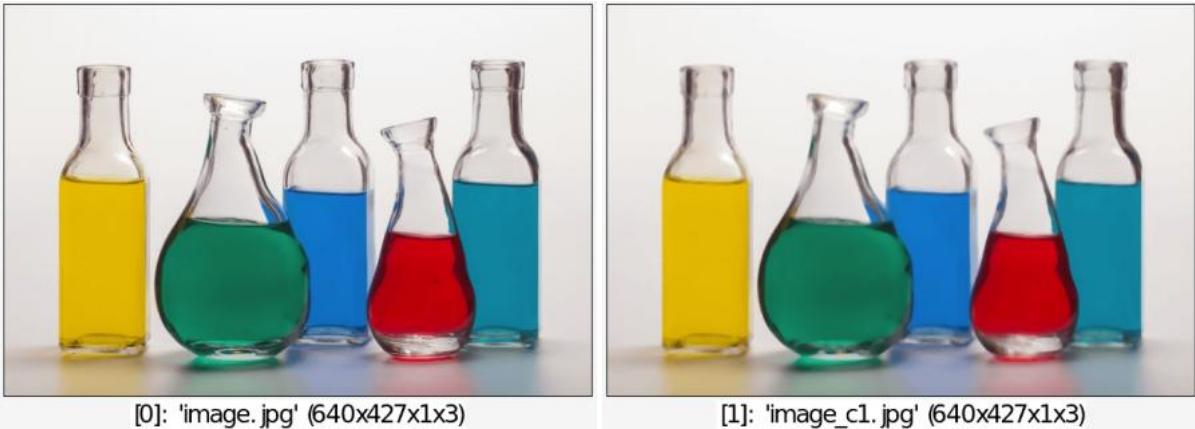
- `size>=0, _threshold>0`

## Description:

Apply (opt. thresholded) median filter on selected images with structuring element size x size.

## Example of use:

```
$ gmic image.jpg +median 5
```



---

## median\_color

**No arguments**

**Description:**

Return the median color value of the last selected image.

---

## median\_files

**Arguments:**

- `"filename_pattern", _first_frame>=0, _last_frame={ >=0 | -1=last } , _frame_step>=1, _frame_rows[%]>=1, _is_fast_approximation={ 0 | 1 }`

**Description:**

Compute the median frame of specified input image files, in a streamed way.

If a display window is opened, rendered frame is displayed in it during processing.

**Default values:**

`first_frame=0, last_frame=-1, frame_step=1, frame_rows=20%` and  
`is_fast_approximation=0`.

---

## median\_video

**Arguments:**

- `video_filename, _first_frame>=0, _last_frame={ >=0 | -1=last } , _frame_step>=1, _frame_rows[%]>=1, _is_fast_approximation={ 0 | 1 }`

## Description:

Compute the median of all frames of an input video file, in a streamed way.

If a display window is opened, rendered frame is displayed in it during processing.

This command requires features from the OpenCV library (not enabled in G'MIC by default).

## Default values:

`first_frame=0`, `last_frame=-1`, `frame_step=1`, `frame_rows=100%` and  
`is_fast_approximation=1`.

---

# meigen

## Arguments:

- `m>=1`

## Description:

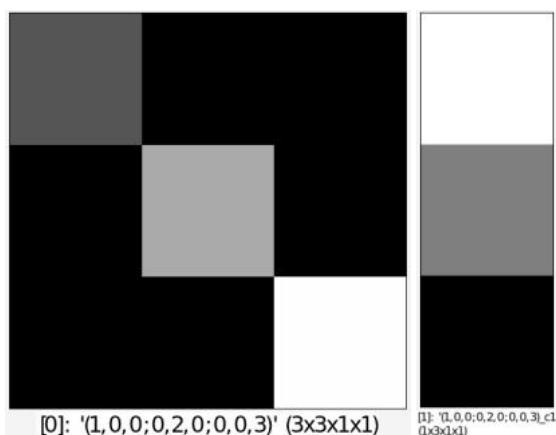
Compute an approximation of the `m` largest eigenvalues and eigenvectors of selected symmetric matrices,

using the Arnoldi iteration method ([https://en.wikipedia.org/wiki/Arnoldi\\_iteration](https://en.wikipedia.org/wiki/Arnoldi_iteration)).

A larger `m` goes with better numerical precision.

## Example of use:

```
$ gmic (1,0,0;0,2,0;0,0,3) +meigen 3
```



# min

Built-in command

## Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

## Description:

Compute the minimum between selected images and specified value, image or mathematical expression, or compute the pointwise minima between selected images.

## Examples of use:

- Example #1

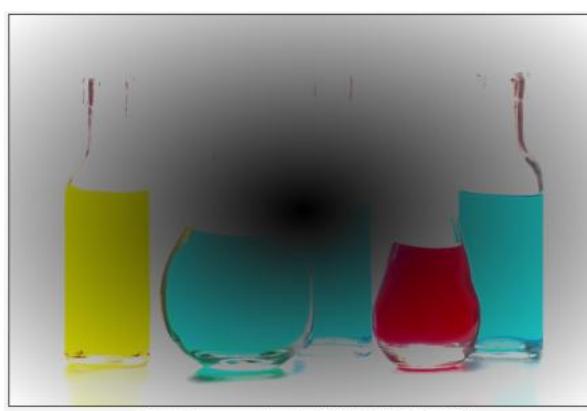
```
$ gmic image.jpg +mirror x min
```



[0]: 'image.jpg' (640x427x1x3)

- Example #2

```
$ gmic image.jpg min 'R=((x/w-0.5)^2+(y/h-0.5)^2)^0.5;255*R'
```



[0]: 'image.jpg' (640x427x1x3)

## min\_d

### No arguments

## Description:

Return the minimal depth between selected images.

---

## min\_h

### No arguments

## Description:

Return the minimal height between selected images.

---

## min\_patch

### Arguments:

- `_patch_size>=1`

## Description:

Return locations of minimal values in local patch-based neighborhood of given size for selected images.

### Default values:

`patch_size=16`.

### Example of use:

```
$ gmic image.jpg norm +min_patch 16
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)

## min\_s

**No arguments**

**Description:**

Return the minimal s size of selected images.

---

**min\_w**

**No arguments**

**Description:**

Return the minimal width between selected images.

---

**min\_wh**

**No arguments**

**Description:**

Return the minimal wxh size of selected images.

---

**min\_whd**

**No arguments**

**Description:**

Return the minimal wxhxd size of selected images.

---

**min\_whds**

**No arguments**

**Description:**

Return the minimal wxhxdxs size of selected images.

---

**minabs**

**Built-in command**

**Arguments:**

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

## Description:

Compute the minabs between selected images and specified value, image or mathematical expression, or compute the pointwise minabs between selected images.

---

# minimal\_path

## Arguments:

- `x0[%]>=0,y0[%]>=0,z0[%]>=0,x1[%]>=0,y1[%]>=0,z1[%]>=0,_is_high_connectivity={ 0 | 1 }`

## Description:

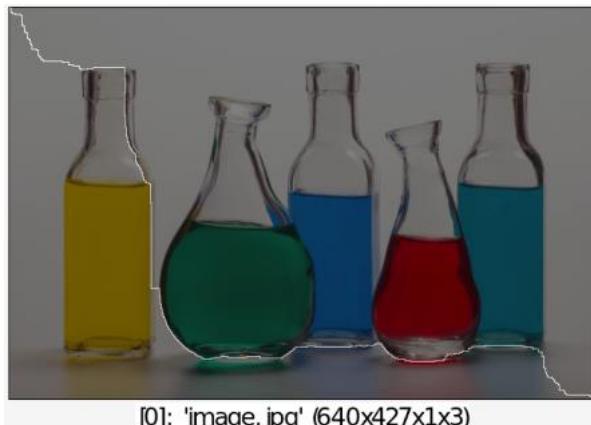
Compute minimal path between two points on selected potential maps.

## Default values:

`is_high_connectivity=0`.

## Example of use:

```
$ gmic image.jpg +gradient_norm fill[-1] 1/(1+i) minimal_path[-1]
0,0,0,100%,100%,0 pointcloud[-1] 0 *[-1] 280 to_rgb[-1] ri[-1] [-2],0
or
```



[0]: 'image.jpg' (640x427x1x3)

---

# mirror

Built-in command

## Arguments:

- `{ x | y | z }...{ x | y | z }`

## Description:

Mirror selected images along specified axes.

## Examples of use:

- Example #1

```
$ gmic image.jpg +mirror y +mirror[0] c
```



[0]: 'image.jpg' (640x427x1x3)



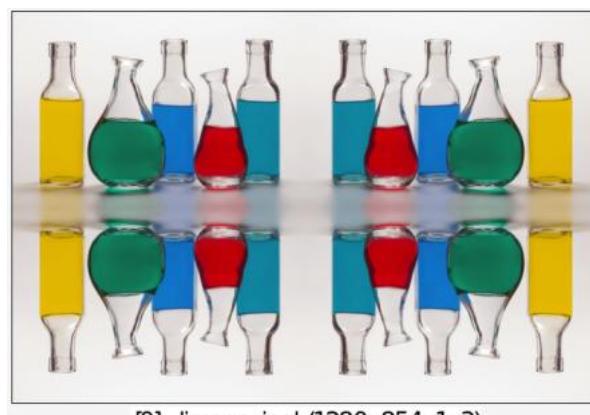
[1]: 'image\_c1.jpg' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)

- Example #2

```
$ gmic image.jpg +mirror x +mirror y append_tiles 2,2
```



[0]: 'image.jpg' (1280x854x1x3)

---

## mix\_channels

### Arguments:

- `(a00,...,aMN)` or
- `[matrix]`

### Description:

Apply specified matrix to channels of selected images.

### Example of use:

```
$ gmic image.jpg +mix_channels (0,1,0;1,0,0;0,0,1)
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## mix\_rgb

### Arguments:

- `a11,a12,a13,a21,a22,a23,a31,a32,a33`

### Description:

Apply 3x3 specified matrix to RGB colors of selected images.

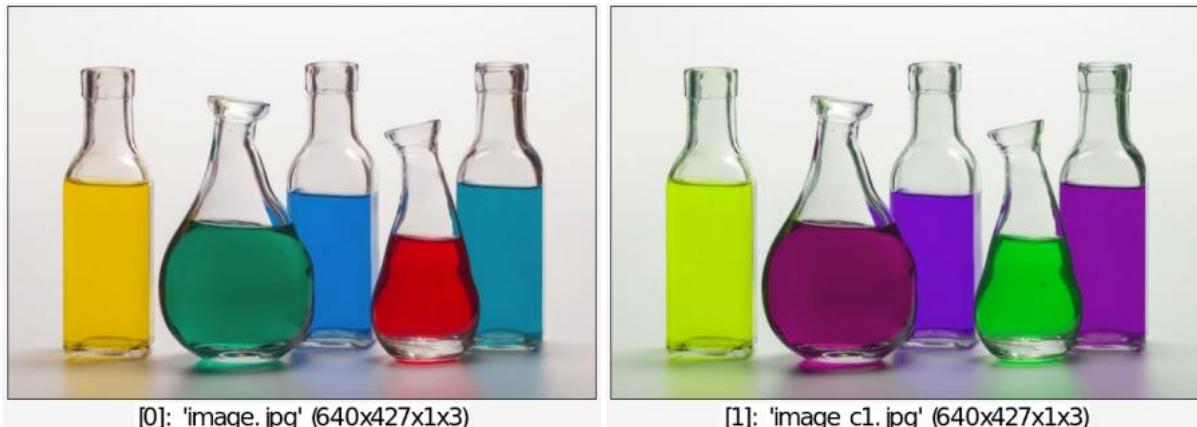
### Default values:

`a11=1`, `a12=a13=a21=0`, `a22=1`, `a23=a31=a32=0` and `a33=1`.

This command has a [tutorial page](#).

### Example of use:

```
$ gmic image.jpg +mix_rgb 0,1,0,1,0,0,0,0,1
```



[0]: 'image.jpg' (640x427x1x3)

[1]: 'image\_c1.jpg' (640x427x1x3)

## mmul

Built-in command

### Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

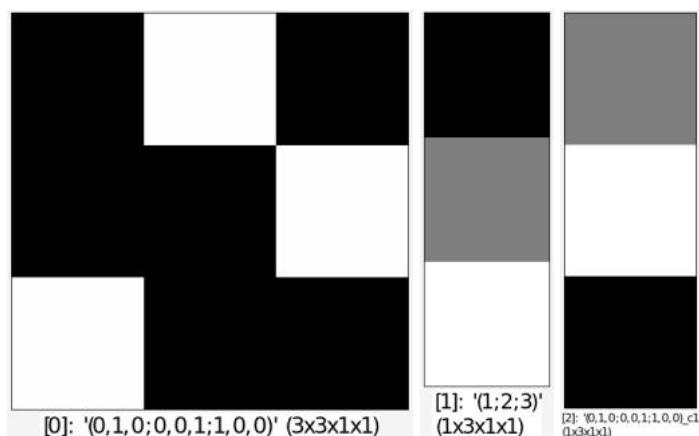
### Description:

Compute the matrix right multiplication of selected matrices/vectors by specified value, image or mathematical expression, or compute the matrix right multiplication of selected images.

(equivalent to shortcut command `m*`).

### Example of use:

```
$ gmic (0,1,0;0,0,1;1,0,0) (1;2;3) +mmul
```



[0]: '(0,1,0;0,0,1;1,0,0)' (3x3x1x1)

[1]: '(1;2;3)'  
(1x3x1x1)[2]: '(0,1,0;0,0,1;1,0,0)\_c1'  
(1x3x1x1)

## mod

Built-in command

## Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

## Description:

Compute the modulo of selected images with specified value, image or mathematical expression, or compute the pointwise sequential modulo of selected images.

(equivalent to shortcut command `%`).

## Examples of use:

- Example #1

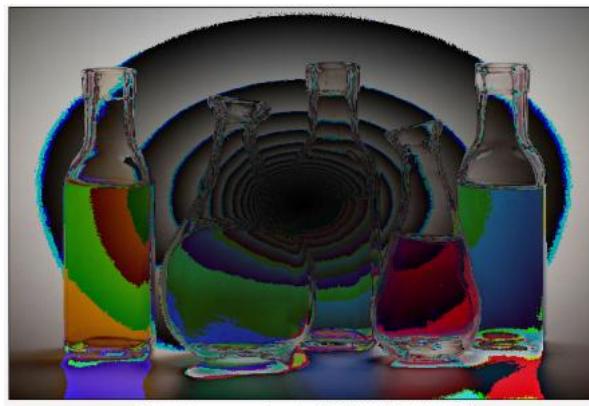
```
$ gmic image.jpg +mirror x mod
```



[0]: 'image.jpg' (640x427x1x3)

- Example #2

```
$ gmic image.jpg mod 'R=((x/w-0.5)^2+(y/h-0.5)^2)^0.5;255*R'
```



[0]: 'image.jpg' (640x427x1x3)

## Arguments:

- `_mode`

## Description:

Set static 3D rendering mode.

(equivalent to shortcut command `m3d`).

`mode` can be `{ -1=bounding-box | 0=dots | 1=wireframe | 2=flat | 3=flat-shaded | 4=gouraud-shaded | 5=phong-shaded }.`");

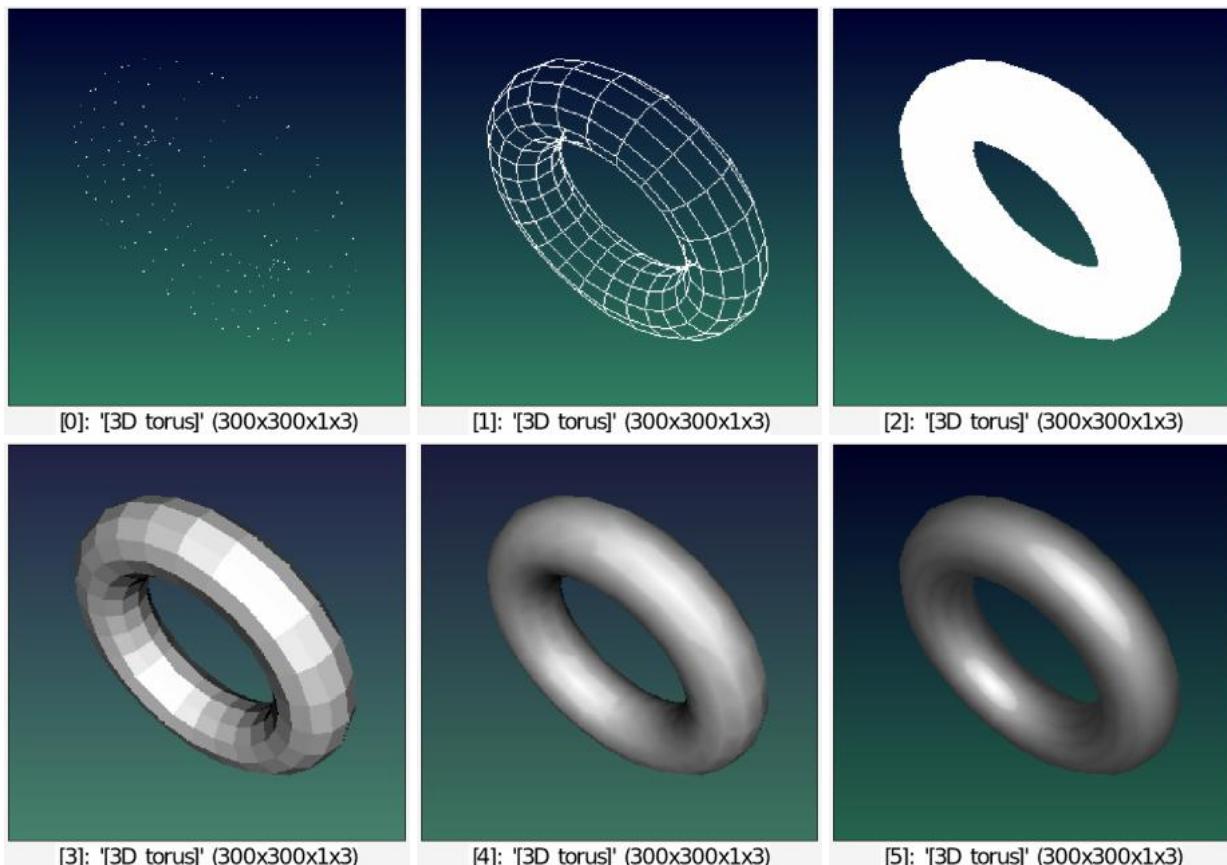
Bounding-box mode (`mode== -1`) is active only for the interactive 3D viewer.

## Default values:

`mode=4`.

## Example of use:

```
$ gmic (0,1,2,3,4,5) double3d 0 repeat w torus3d 100,30 rotate3d[-1] 1,1,0,60 mode3d {0,@$>} snapshot3d[-1] 300 done remove[0]
```



---

## moded3d

[Built-in command](#)

## Arguments:

- `_mode`

## Description:

Set dynamic 3D rendering mode for interactive 3D viewer.

(equivalent to shortcut command `md3d`).

`mode` can be `{ -1=bounding-box | 0=dots | 1=wireframe | 2=flat | 3=flat-shaded | 4=gouraud-shaded | 5=phong-shaded }`.

## Default values:

`mode=-1`.

---

# montage

## Arguments:

- `"_layout_code", _montage_mode={ 0<=centering<=1 | 2<=scale+2<=3 }, _output_mode={ 0=single layer | 1=multiple layers }, "_processing_command"`

## Description:

Create a single image montage from selected images, according to specified layout code :

- `X` to assemble all images using an automatically estimated layout.
- `H` to assemble all images horizontally.
- `V` to assemble all images vertically.
- `A` to assemble all images as an horizontal array.
- `B` to assemble all images as a vertical array.
- `Ha:b` to assemble two blocks `a` and `b` horizontally.
- `Va:b` to assemble two blocks `a` and `b` vertically.
- `Ra` to rotate a block `a` by 90 deg. (`RRa` for 180 deg. and `RRRa` for 270 deg.).
- `Ma` to mirror a block `a` along the X-axis (`MRRa` for the Y-axis).

A block `a` can be an image index (treated periodically) or a nested layout expression

`Hb:c, Vb:c, Rb` or

`Mb` itself.

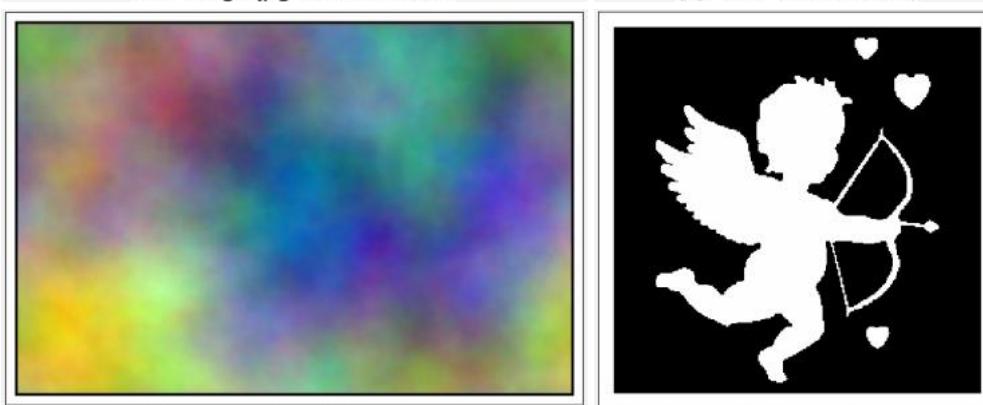
For example, layout code `H0:V1:2` creates an image where image [0] is on the left, and images [1] and [2] vertically packed on the right.

## Default values:

`layout_code=X`, `montage_mode=2`, `output_mode='0'` and `processing_command=""`.

## Example of use:

```
$ gmic image.jpg sample ? +plasma[0] shape_cupid 256 normalize 0,255  
frame 3,3,0 frame 10,10,255 to_rgb +montage A +montage[^-1]  
H1:V0:VH2:1H0:3
```



## morph

### Arguments:

- `nb_inner_frames>=1,_smoothness>=0,_precision>=0`

### Description:

Create morphing sequence between selected images.

## Default values:

`smoothness=0.1` and `precision=4`.

## Example of use:

```
$ gmic image.jpg +rotate 20,1,1,50%,50% morph 9
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image.jpg' (640x427x1x3)



[2]: 'image.jpg' (640x427x1x3)



[3]: 'image.jpg' (640x427x1x3)



[4]: 'image.jpg' (640x427x1x3)



[5]: 'image.jpg' (640x427x1x3)



[6]: 'image.jpg' (640x427x1x3)



[7]: 'image.jpg' (640x427x1x3)



[8]: 'image.jpg' (640x427x1x3)



[9]: 'image.jpg' (640x427x1x3)



[10]: 'image.jpg' (640x427x1x3)

---

## morph\_files

### Arguments:

- `"filename_pattern", _nb_inner_frames>=0, _smoothness>=0, _precision>=0, _first_fr { >=0 | -1=last }, _frame_step>=1, _output_filename`

### Description:

Generate a temporal morphing from specified input image files, in a streamed way.

If a display window is opened, rendered frames are displayed in it during processing.  
The output filename may have extension `.avi` or `.mp4` (saved as a video), or any other usual image file extension (saved as a sequence of images).

## Default values:

```
nb_inner_frames=10, smoothness=0.1, precision=4, first_frame=0,  
last_frame=-1, frame_step=1 and output_filename=(undefined).
```

---

## morph\_rbf

### Arguments:

- `nb_inner_frames>=1, xs0[%], ys0[%], xt0[%], yt0[%], ..., xsN[%], ysN[%], xtN[%], ytN[%]`

### Description:

Create morphing sequence between selected images, using RBF-based interpolation.

Each argument (xsk,ysk)-(xtk,ytk) corresponds to the coordinates of a keypoint respectively on the source and target images. The set of all keypoints define the overall image deformation.

---

## morph\_video

### Arguments:

- `video_filename, _nb_inner_frames>0, _smoothness>=0, _precision>=0, _first_frame>{ >=0 | -1=last }, _frame_step>=1, _output_filename`

### Description:

Generate a temporal morphing from specified input video file, in a streamed way.

If a display window is opened, rendered frames are displayed in it during processing.  
The output filename may have extension `.avi` or `.mp4` (saved as a video), or any other usual image file extension (saved as a sequence of images).  
This command requires features from the OpenCV library (not enabled in G'MIC by default).

## Default values:

```
nb_inner_frames=10, smoothness=0.1, precision=4, first_frame=0,  
last_frame=-1, frame_step=1 and output_filename=(undefined).
```

---

## mosaic

### Arguments:

- `0<=_density<=100`

## Description:

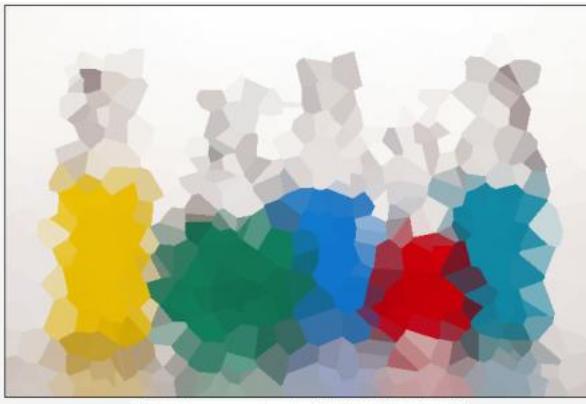
Create random mosaic from selected images.

## Default values:

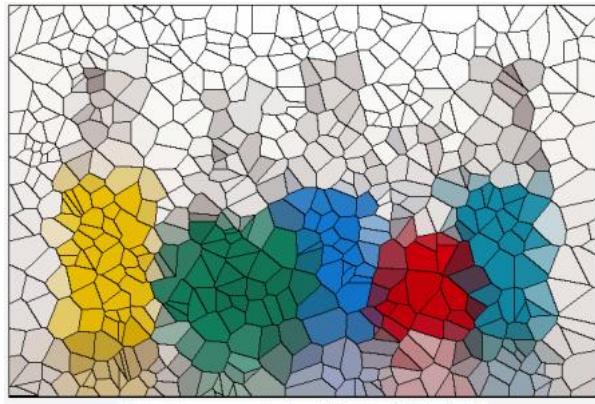
`density=30`.

## Example of use:

```
$ gmic image.jpg mosaic , +fill "I!=J(1) || I!=J(0,1)?[0,0,0]:I"
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

## move

Built-in command

## Arguments:

- `position[%]`

## Description:

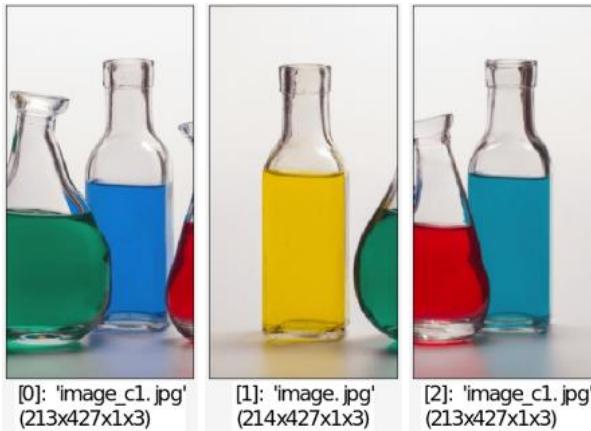
Move selected images at specified position.

(equivalent to shortcut command `mv`).

## Examples of use:

- Example #1

```
$ gmic image.jpg split x,3 move[1] 0
```



- **Example #2**

```
$ gmic image.jpg split x move[50%--1:2] 0 append x
```



## mproj

Built-in command

### Arguments:

- `[dictionary],_method,_max_iter={ 0=auto | >0 },_max_residual>=0`

### Description:

Find best matching projection of selected matrices onto the span of an over-complete

dictionary D, using the orthogonal projection or Matching Pursuit algorithm.

Selected images are 2D-matrices in which each column represent a signal to project.

`[dictionary]` is a matrix in which each column is an element of the dictionary D.

`method` tells what projection algorithm must be applied. It can be:

- 0 = orthogonal projection (least-squares solution using LU-based solver).
- 1 = matching pursuit.
- 2 = matching pursuit, with a single orthogonal projection step at the end.
- >=3 = orthogonal matching pursuit where an orthogonal projection step is performed every `method-2` iterations.

`max_iter` sets the max number of iterations processed for each signal.

If set to `0` (default), `max_iter` is equal to the number of columns in D.

(only meaningful for matching pursuit and its variants).

`max_residual` gives a stopping criterion on signal reconstruction accuracy.

(only meaningful for matching pursuit and its variants).

For each selected image, the result is returned as a matrix W

whose columns correspond to the weights associated to each column of D,  
such that the matrix product D\*W is an approximation of the input matrix.

## Default values:

`method=0`, `max_iter=0` and `max_residual=1e-6`.

---

# mse

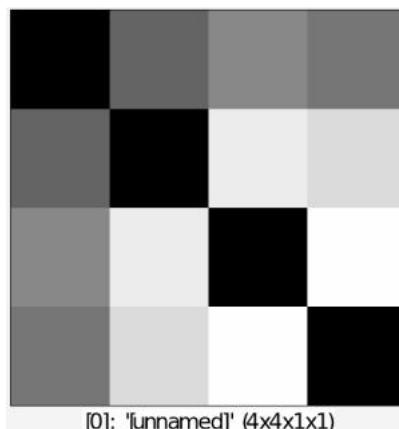
## No arguments

### Description:

Compute MSE (Mean-Squared Error) matrix between selected images.

### Example of use:

```
$ gmic image.jpg +noise 30 +noise[0] 35 +noise[0] 38 cut. 0,255 mse
```



# mul

Built-in command

---

## Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

### Description:

Multiply selected images by specified value, image or mathematical expression, or compute the pointwise product of selected images.

(equivalent to shortcut command `*`).

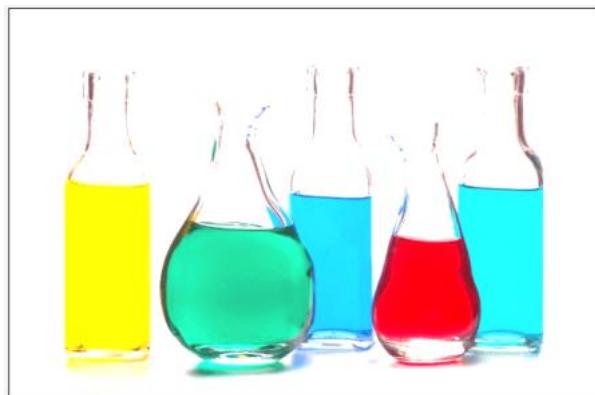
## See also:

`add`, `sub`, `div`.

## Examples of use:

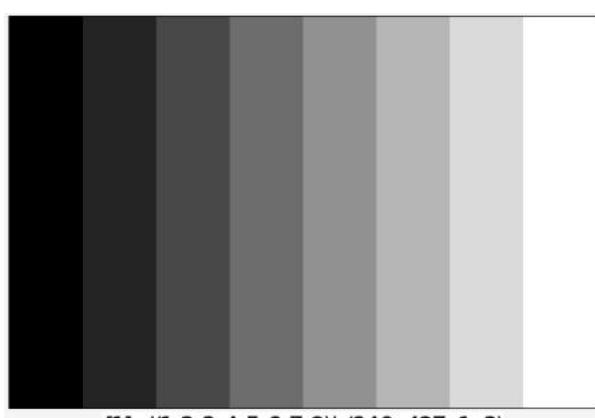
- Example #1

```
$ gmic image.jpg +mul 2 cut 0,255
```



- Example #2

```
$ gmic image.jpg (1,2,3,4,5,6,7,8) ri[-1] [0] mul[0] [-1]
```



- Example #3

```
$ gmic image.jpg mul '1-3*abs(x/w-0.5)' cut 0,255
```



[0]: 'image.jpg' (640x427x1x3)

- **Example #4**

```
$ gmic image.jpg +luminance negate[-1] +mul
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x3)

---

## mul3d

Built-in command

### Arguments:

- `factor` or
- `factor_x, factor_y, factor_z`

### Description:

Scale selected 3D objects isotropically or anisotropically, with specified factors.

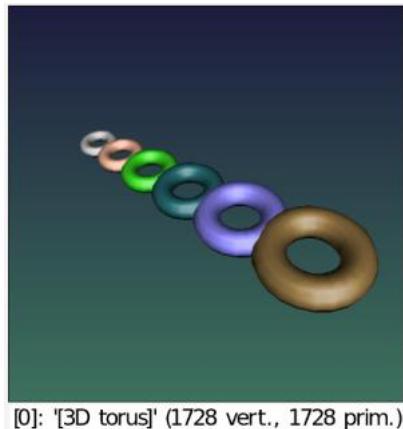
(equivalent to shortcut command `*3d`).

## Default values:

`factor_z=0`.

## Example of use:

```
$ gmic torus3d 5,2 repeat 5 +add3d[-1] 10,0,0 mul3d[-1] 1.2  
color3d[-1] ${-rgb} done add3d
```



---

## mul\_channels

### Arguments:

- `value1,_value2,...,_valueN`

### Description:

Multiply channels of selected images by specified sequence of values.

## Example of use:

```
$ gmic image.jpg +mul_channels 1,0.5,0.8
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## mul\_complex

### Arguments:

- `[multiplier_real,multiplier_imag]`

### Description:

Perform multiplication of the selected complex pairs (real1,imag1,...,realN,imagN) of images by specified complex pair of images (multiplier\_real,multiplier\_imag). In complex pairs, the real image must be always located before the imaginary image in the image list.

---

## mutex

Built-in command

### Arguments:

- `index,_action={ 0=unlock | 1=lock }`

### Description:

Lock or unlock specified mutex for multi-threaded programming.

A locked mutex can be unlocked only by the same thread. All mutexes are unlocked by default. `index` designates the mutex index, in [0,255].

### Default values:

`action=1`.

---

## nadirzenith2equirectangular

### No arguments

### Description:

Transform selected nadir/zenith rectilinear projections to equirectangular images.

---

## name

Built-in command

### Arguments:

- `"name1", "name2", ...`

## Description:

Set names of selected images.

- If the selection contains a single image, then it is assumed the command has a single name argument (possibly containing multiple commas).
- If the selection contains more than one image, each command argument defines a single image name for each image of the selection.

(equivalent to shortcut command `nm`).

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg name image blur[image] 2
```



## named

[Built-in command](#)

## Arguments:

- `_mode, "name1", "name2", ...`

## Description:

Return the set of indices corresponding to images of the selection with specified names.

After this command returns, the status contains a list of indices (unsigned integers), separated by commas (or an empty string if no images with those names have been found).

(equivalent to shortcut command `nmd`).

`mode` can be `{ 0=all indices (default) | 1=lowest index | 2=highest index | 3 = all indices (case insensitive) | 4 = lowest index (case insensitive) | 5 = highest index (case insensitive)}`

# negate

## Arguments:

- `base_value` or
- `(no arg)`

## Description:

Negate image values.

## Default values:

`base_value=(undefined)`.

## Example of use:

```
$ gmic image.jpg +negate
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

# neq

Built-in command

## Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

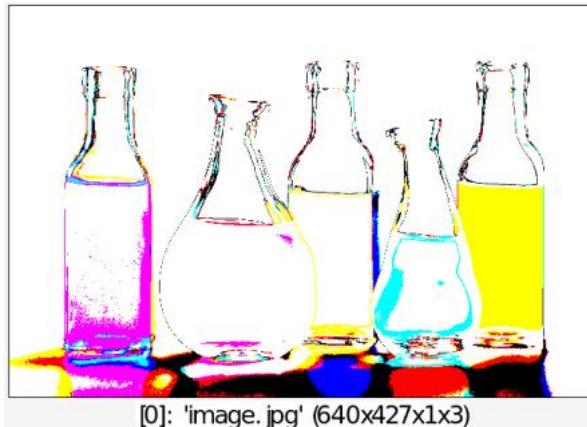
## Description:

Compute the boolean inequality of selected images with specified value, image or mathematical expression, or compute the boolean inequality of selected images.

(equivalent to shortcut command `!=`).

## Example of use:

```
$ gmic image.jpg round 40 neq {round(ia,40)}
```



---

## network

Built-in command

### Arguments:

- mode={ -1=disabled | 0=enabled w/o timeout | >0=enabled w/ specified timeout in seconds }

### Description:

Enable/disable load-from-network and set corresponding timeout.

(Default mode is 'enabled w/o timeout').

---

## newton\_fractal

### Arguments:

- z0r,z0i,z1r,z1i,\_angle,\_descent\_method<=2,\_iteration\_max>=0,\_convergence\_

### Description:

Draw newton fractal on selected images, for complex numbers in range (z0r,z0i) - (z1r,z1i).

Resulting images have 3 channels whose meaning is [ last\_zr, last\_zi, nb\_iter\_used\_for\_convergence ].

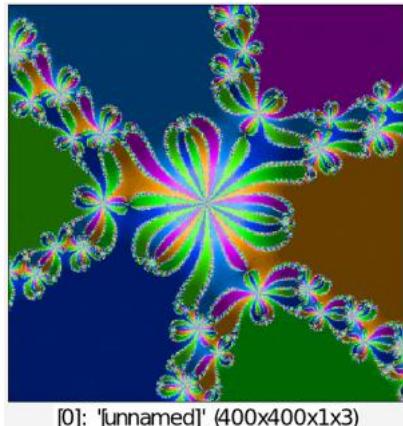
`descent_method` can be { 0=secant | 1=newton | 2=householder }.

### Default values:

`angle=0`, `descent_method=1`, `iteration_max=200`, `convergence_precision=0.01`,  
`expr_p(z)=z^^3-1`, `expr_dp(z)=3*z^^2` and `expr_d2z(z)=6*z`.

### Example of use:

```
$ gmic 400,400 newton_fractal -1.5,-1.5,1.5,1.5,0,2,200,0.01,"z^^6 + z^^3 - 1","6*z^^5 + 3*z^^2","30*z^^4 + 6*z" f "[ atan2(i1,i0)*90+20,1,cut(i2/30,0.2,0.7) ]" hsl2rgb
```



## nlmeans

### Arguments:

- `[guide],_patch_radius>0,_spatial_bandwidth>0,_tonal_bandwidth>0,_patch_measur`  
or
- `_patch_radius>0,_spatial_bandwidth>0,_tonal_bandwidth>0,_patch_measure_comma`

### Description:

Apply non local means denoising of Buades et al, 2005. on selected images.

The patch is a gaussian function of `std_patch_radius`.

The spatial kernel is a rectangle of radius `spatial_bandwidth`.

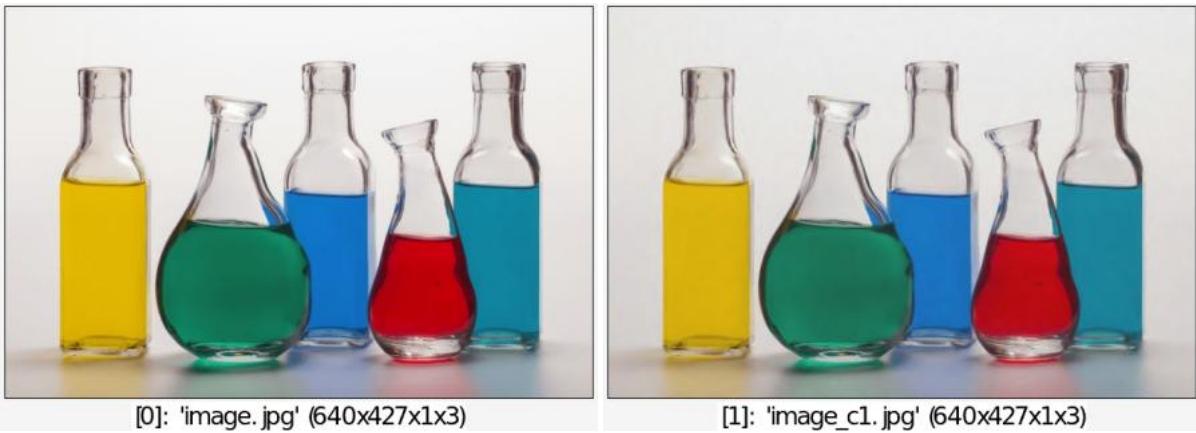
The tonal kernel is exponential (`exp( -d^2/_tonal_bandwidth^2 )`)  
with `d` the euclidean distance between image patches.

### Default values:

`patch_radius=4`, `spatial_bandwidth=4`, `tonal_bandwidth=10` and  
`patch_measure_command=-norm`.

### Example of use:

```
$ gmic image.jpg +noise 10 nlmeans[-1] 4,4,{0.6*${-std_noise}}
```



---

## nlmeans\_core

### Arguments:

- `_reference_image, _scaling_map, _patch_radius>0, _spatial_bandwidth>0`

### Description:

Apply non local means denoising using a image for weight and a map for scaling

---

## nn\_check\_layer

### Arguments:

- `name`

### Description:

Check that the layer with specified name exists in the network.

---

## nn\_init

### No arguments

### Description:

Initialize a new network.

---

## nn\_layer\_add

### Arguments:

- `name,in0,in1`

## Description:

Add an `add` layer to the network.

---

# nn\_layer\_append

## Arguments:

- `name,in0,in1`

## Description:

Add an `append` layer to the network.

---

# nn\_layer\_avgpool2d

## Arguments:

- `name,in`

## Description:

Add a `avgpool2d` layer (2d average pooling) to the network.

---

# nn\_layer\_batchnorm

## Arguments:

- `name,in,_learning_mode.`

## Description:

Add a `batchnorm` layer to the network.

`learning_mode` can be `{ 0=learn no parameters | 1=learn gamma | 2=learn beta | 3=learn gamma and beta}`.

## Default values:

`learning_mode=3`.

---

# nn\_layer\_clone

## Arguments:

- `name0, name1, in`

## Description:

Add a `clone` layer to the network.

---

# nn\_layer\_conv2d

## Arguments:

- `name, in, nb_channels>0, _kernel_size>0, _stride>0, _dilation, _is_updated={ 0 | 1 }`

## Description:

Add a `conv2d` layer (2D convolutional layer) to the network.

## Default values:

`kernel_size=3, stride=1, dilation=1` and `is_updated=1`.

---

# nn\_layer\_conv2dbnnl

## Arguments:

- `name, in, nb_channels>0, _kernel_size>0, _stride>0, _dilation>0, _activation, _is_u { 0 | 1 }`

## Description:

Add a `con2dbnnl` (2D convolutional layer followed by a batchnorm, then a non-linearity), to the network.

## Default values:

`kernel_size=3, stride=1, dilation=1, activation=leakyrelu` and `is_updated=1`.

---

# nn\_layer\_conv2dnl

## Arguments:

- `name, in, nb_channels>0, _kernel_size>0, _stride>0, _dilation>0, _activation, _is_updated={ 0 | 1 }`

## Description:

Add a `con2dn1` (2D convolutional layer followed by a non-linearity), to the network.

## Default values:

`kernel_size=3, stride=1, dilation=1, activation=leakyrelu` and `is_updated=1`.

---

# nn\_layer\_crop

## Arguments:

- `name, in, x0, y0, z0, x1, y1, z1`

## Description:

Add a `crop` layer to the network.

---

# nn\_layer\_fc

## Arguments:

- `name, in, nb_channels>0, _is_updated={ 0 | 1 }`

## Description:

Add a `fc` layer (fully connected layer) to the network.

## Default values:

`is_updated=1`.

---

# nn\_layer\_fcbnnl

## Arguments:

- `name, in, nb_neurons>0, _activation, _is_updated={ 0 | 1 }`

## Description:

Add a `fcbnnl` layer (fully connected layer followed by batchnorm, then a non-linearity), to the network.

## Default values:

`activation=leakyrelu` and `is_updated=1`.

---

## nn\_layer\_fcnl

### Arguments:

- `name,in,nb_neurons>0,_activation,_is_updated={ 0 | 1 }`

### Description:

Add a `fcnl` layer (fully connected layer followed by a non-linearity), to the network.

## Default values:

`activation=leakyrelu` and `is_updated=1`.

---

## nn\_layer\_input

### Arguments:

- `name,width,_height,_depth,_spectrum`

### Description:

Add an `input` layer to the network.

## Default values:

`height=1`, `depth=1` and `spectrum=1`.

---

## nn\_layer\_maxpool2d

### Arguments:

- `name,in`

### Description:

Add a `maxpool2d` layer (2d max pooling) to the network.

---

## nn\_layer\_nl

### Arguments:

- `name,in,_activation`

### Description:

Add a `nl` layer (non-linearity) to the network.

### Default values:

`activation=leakyrelu`.

---

## nn\_layer\_rename

### Arguments:

- `name,in`

### Description:

Add a `rename` layer to the network.

---

## nn\_layer\_reshape

### Arguments:

- `name,in,width>0,height>0,depth>0,spectrum>0`

### Description:

Add a `reshape` layer to the network.

---

## nn\_layer\_resize

### Arguments:

- `name,in,width[%]>0,_height[%]>0,_depth[%]>0,_spectrum[%]>0,_interpolation`

### Description:

Add a `resize` layer to the network.

### Default values:

---

```
height=depth=spectrum=100%.
```

## nn\_layer\_run

### Arguments:

- `name,in,"command",_width[%]>0,_height[%]>0,_depth[%]>0,_spectrum[%]>0`

### Description:

Add a `run` layer to the network.

### Default values:

```
width=height=depth=spectrum=100%.
```

---

## nn\_layer\_split

### Arguments:

- `name0,name1,in,nb_channels0`

### Description:

Add a `split` layer to the network.

---

## nn\_lib

### No arguments

### Description:

Return the list of library functions that has to be included in a math expression,in order to use the neural network library.

---

## nn\_load

### Arguments:

- `'filename.gmz'`

### Description:

Load and initialize network saved as a .gmz file.

Neural network files can be only loaded in .gmz format.

---

## nn\_loss\_mse

### Arguments:

- `name, in, ground_truth`

### Description:

Add a `mse` loss to the network.

### Default values:

`learning_rate=1e-6` and `learning_rate_scheduler=adaptive`.

---

## nn\_save

### Arguments:

- `'filename.gmz'`

### Description:

Save current network as a .gmz file.

Neural network files can be only saved in .gmz format.

---

## nn\_trainer

### Arguments:

- `name, loss, _learning_rate>0, _optimizer, _scheduler`

### Description:

Add a network trainer to the network.

`optimizer` can be `{ sgd | rmsprop | adam | adamax }`.

`scheduler` can be `{ constant | linear | exponential | adaptive }`.

### Default values:

`learning_rate=1e-6`, `optimizer=adam` and `scheduler=constant`.

---

## noarg

Built-in command

No arguments

### Description:

Used in a custom command, `noarg` tells the command that its argument list have not been used finally, and so they must be evaluated next in the G'MIC pipeline, just as if the custom command takes no arguments at all.  
Use this command to write a custom command which can decide if it takes arguments or not.

---

## noise

Built-in command

### Arguments:

- `std_deviation>=0[%]` , `_noise_type`

### Description:

Add random noise to selected images.

`noise_type` can be `{ 0=gaussian | 1=uniform | 2=salt&pepper | 3=poisson | 4=rice }`.

### Default values:

`noise_type=0`.

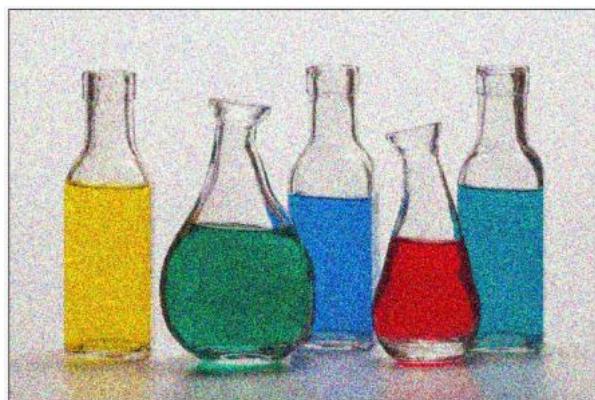
### Examples of use:

- **Example #1**

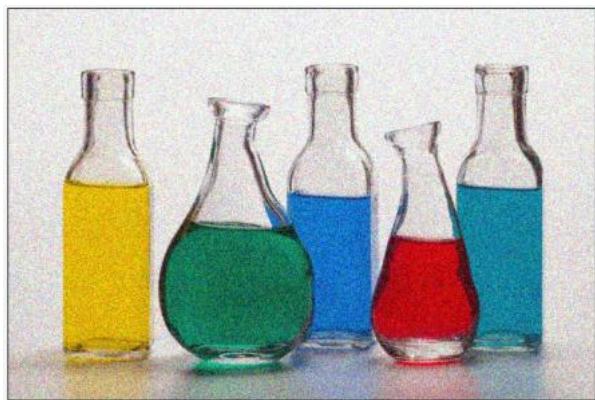
```
$ gmic image.jpg +noise[0] 50,0 +noise[0] 50,1 +noise[0] 10,2 cut  
0,255
```



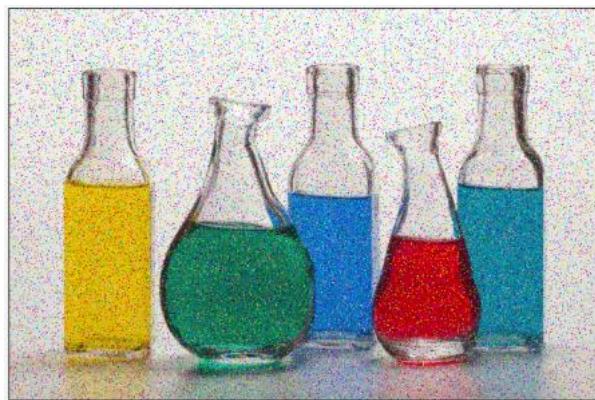
[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)



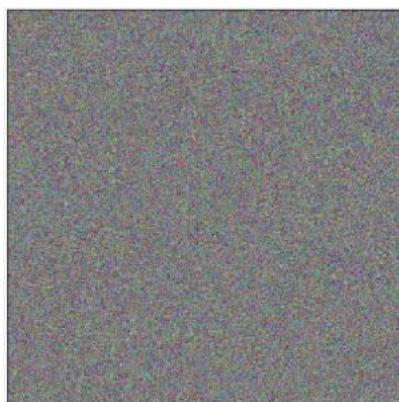
[2]: 'image\_c1.jpg' (640x427x1x3)



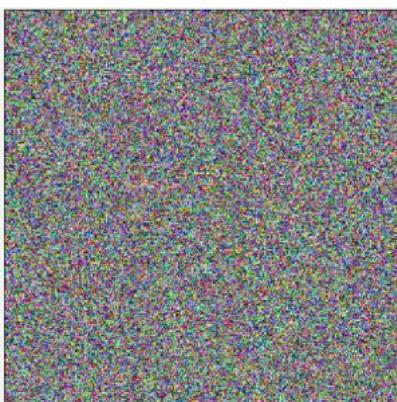
[3]: 'image\_c1.jpg' (640x427x1x3)

## • Example #2

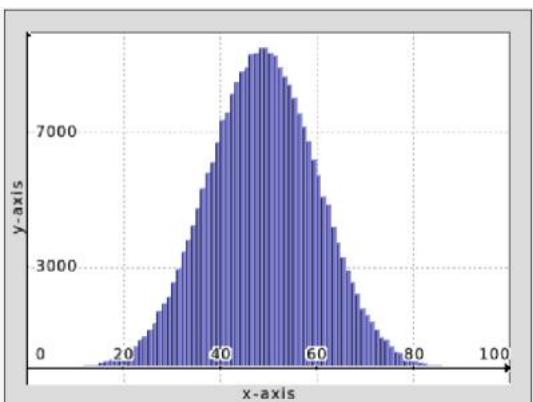
```
$ gmic 300,300,1,3 [0] noise[0] 20,0 noise[1] 20,1 +histogram 100  
display_graph[-2,-1] 400,300,3
```



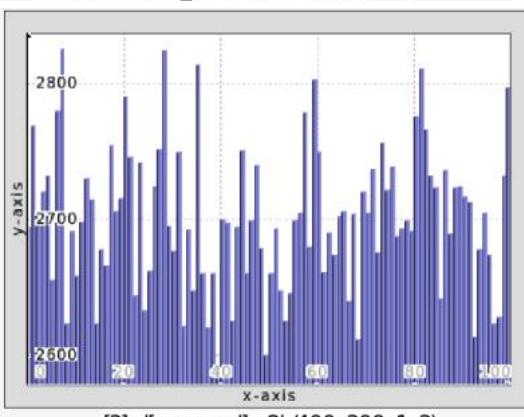
[0]: '[unnamed]' (300x300x1x3)



[1]: '[unnamed]\_c1' (300x300x1x3)



[2]: '[unnamed]\_c1' (400x300x1x3)



[3]: '[unnamed]\_c2' (400x300x1x3)

# noise\_hurl

## Arguments:

- `_amplitude>=0`

## Description:

Add hurl noise to selected images.

## Default values:

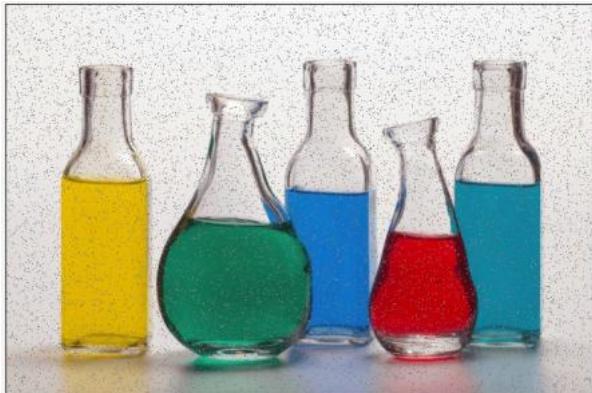
`amplitude=10`.

## Example of use:

```
$ gmic image.jpg +noise_hurl ,
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

# noise\_perlin

## Arguments:

- `_scale_x[%]>0,_scale_y[%]>0,_scale_z[%]>0,_seed_x,_seed_y,_seed_z`

## Description:

Render 2D or 3D Perlin noise on selected images, from specified coordinates.

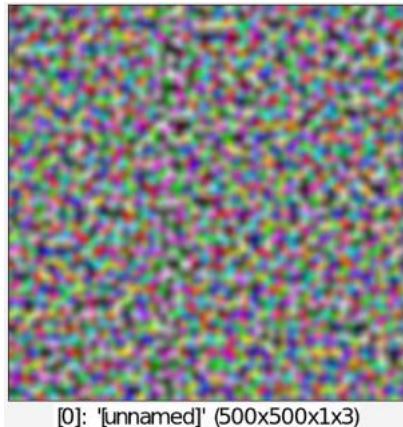
The Perlin noise is a specific type of smooth noise,  
described here : [https://en.wikipedia.org/wiki/Perlin\\_noise](https://en.wikipedia.org/wiki/Perlin_noise).

## Default values:

`scale_x=scale_y=scale_z=16` and `seed_x=seed_y=seed_z=0`.

## Example of use:

```
$ gmic 500,500,1,3 noise_perlin ,
```



---

## noise\_poissondisk

### Arguments:

- `_radius[%]>0, _max_sample_attempts>0`

### Description:

Add poisson disk sampling noise to selected images.

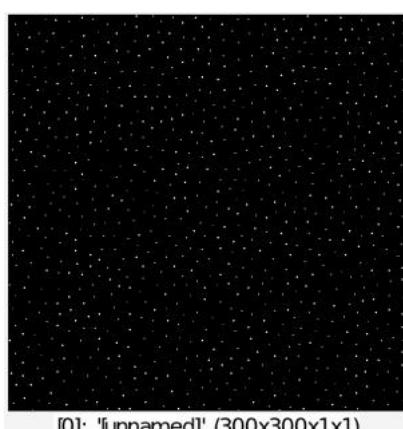
Implements the algorithm from the article "Fast Poisson Disk Sampling in Arbitrary Dimensions", by Robert Bridson (SIGGRAPH'2007).

### Default values:

`radius=8` and `max_sample_attempts=30`.

### Example of use:

```
$ gmic 300,300 noise_poissondisk 8
```



# norm

No arguments

## Description:

Compute the pointwise euclidean norm of vector-valued pixels in selected images.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg +norm
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x1)

---

# normalize

[Built-in command](#)

## Arguments:

- `{ value0[%] | [image0] }, { value1[%] | [image1] }, _constant_case_ratio` or
- `[image]`

## Description:

Linearly normalize values of selected images in specified range.

(equivalent to shortcut command [n](#)).

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg split x,2 normalize[-1] 64,196 append x
```



---

## normalize3d

**No arguments**

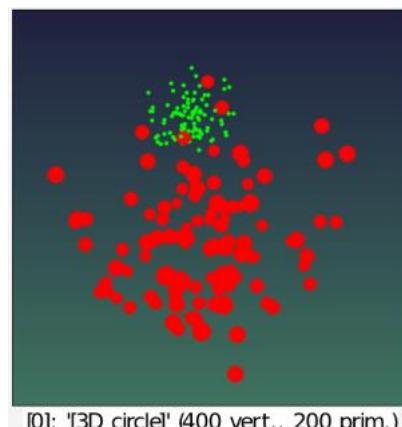
**Description:**

Normalize selected 3D objects to unit size.

(equivalent to shortcut command `n3d`).

**Example of use:**

```
$ gmic repeat 100 circle3d {u(3)},{u(3)},{u(3)},0.1 done add3d  
color3d[-1] 255,0,0 +normalize3d[-1] color3d[-1] 0,255,0 add3d
```



---

## normalize\_filename

**Arguments:**

- `filename`

**Description:**

Return a "normalized" version of the specified filename, without spaces and capital letters.

---

## normalize\_l2

**No arguments**

### Description:

Normalize selected images such that they have a unit L2 norm.

---

## normalize\_local

### Arguments:

- `_amplitude>=0, _radius>0, _n_smooth>=0[%], _a_smooth>=0[%], _is_cut={ 0 | 1 }, _min=0, _max=255`

### Description:

Normalize selected images locally.

### Default values:

`amplitude=3, radius=16, n_smooth=4%, a_smooth=2%, is_cut=1, min=0` and `max=255`.

### Example of use:

```
$ gmic image.jpg normalize_local 8,10
```



---

## normalize\_sum

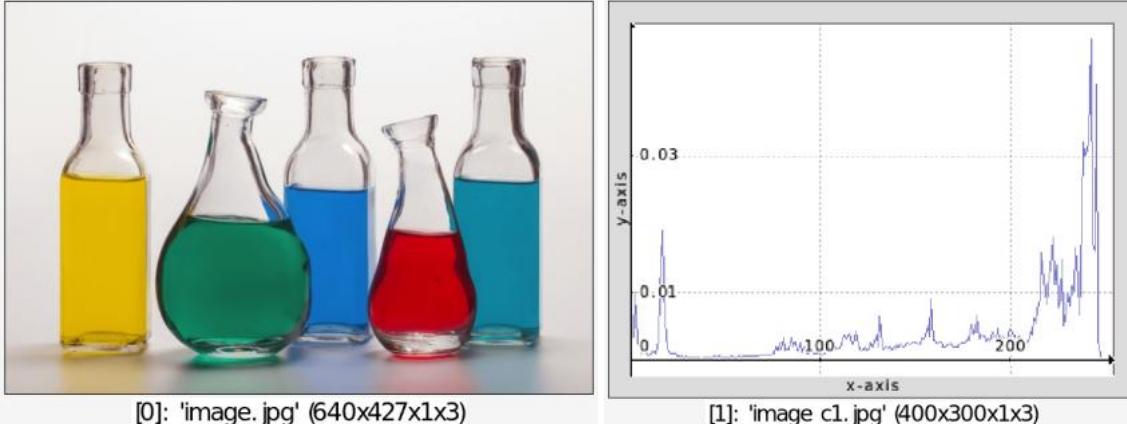
**No arguments**

## Description:

Normalize selected images such that they have a unit sum.

## Example of use:

```
$ gmic image.jpg +histogram 256 normalize_sum[-1] display_graph[-1]  
400,300
```



---

## normalized\_cross\_correlation

### Arguments:

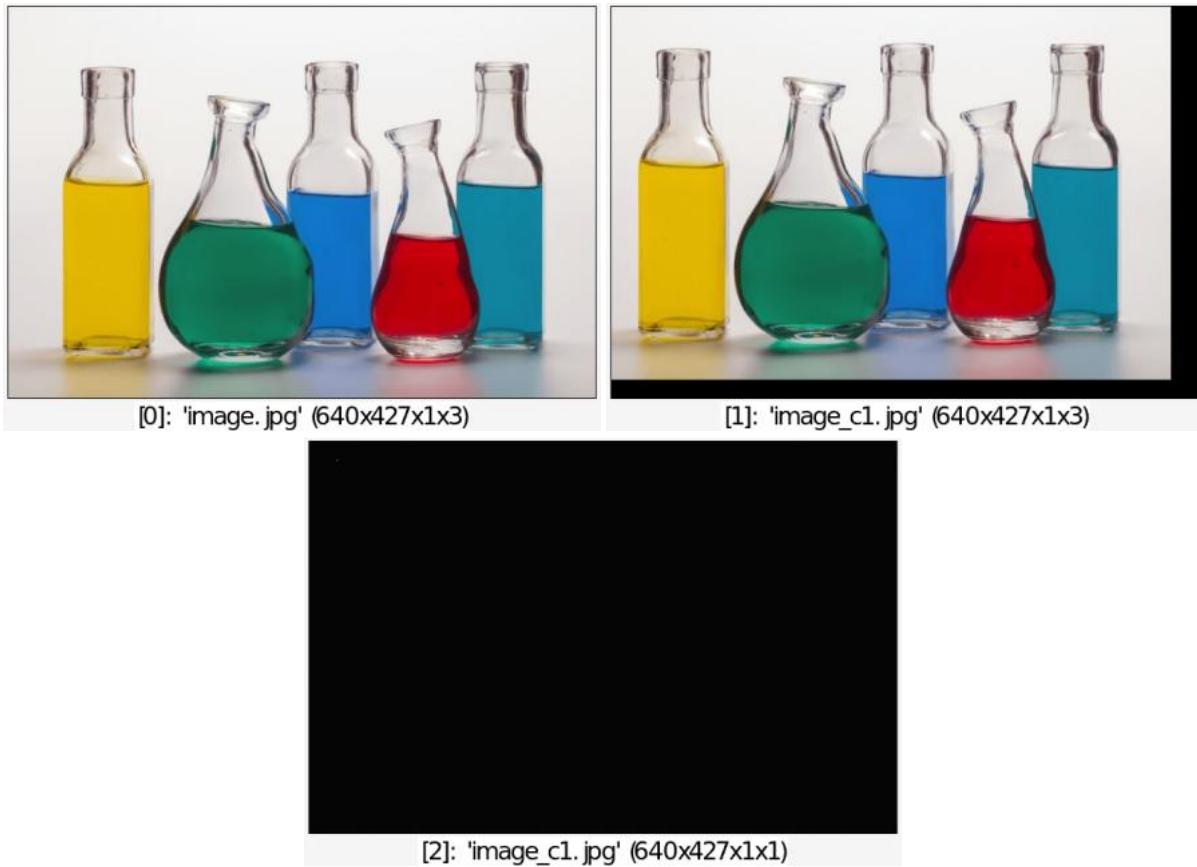
- [mask]

## Description:

Compute normalized cross-correlation of selected images with specified mask.

## Example of use:

```
$ gmic image.jpg +shift -30,-20 +normalized_cross_correlation[0] [1]
```



---

## normp

### Arguments:

- `p>=0`

### Description:

Compute the pointwise L<sub>p</sub>-norm norm of vector-valued pixels in selected images.

### Default values:

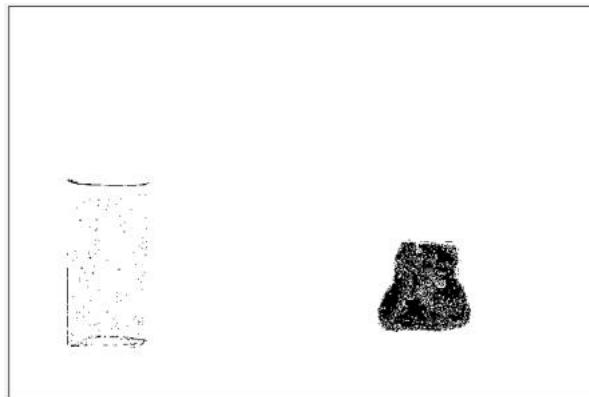
`p=2`.

### Example of use:

```
$ gmic image.jpg +normp[0] 0 +normp[0] 1 +normp[0] 2 +normp[0] inf
```



[0]: 'image.jpg' (640x427x1x3)



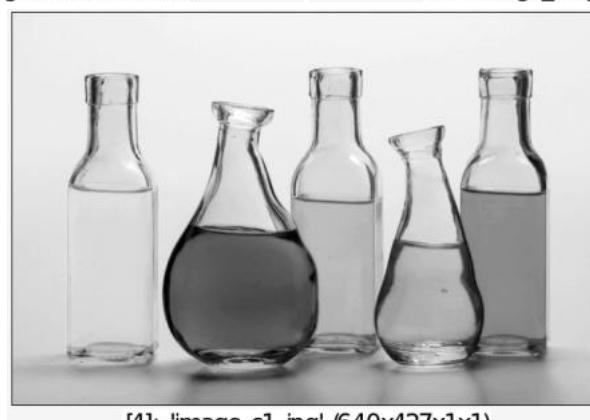
[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x1)



[3]: 'image\_c1.jpg' (640x427x1x1)



[4]: 'image\_c1.jpg' (640x427x1x1)

---

not

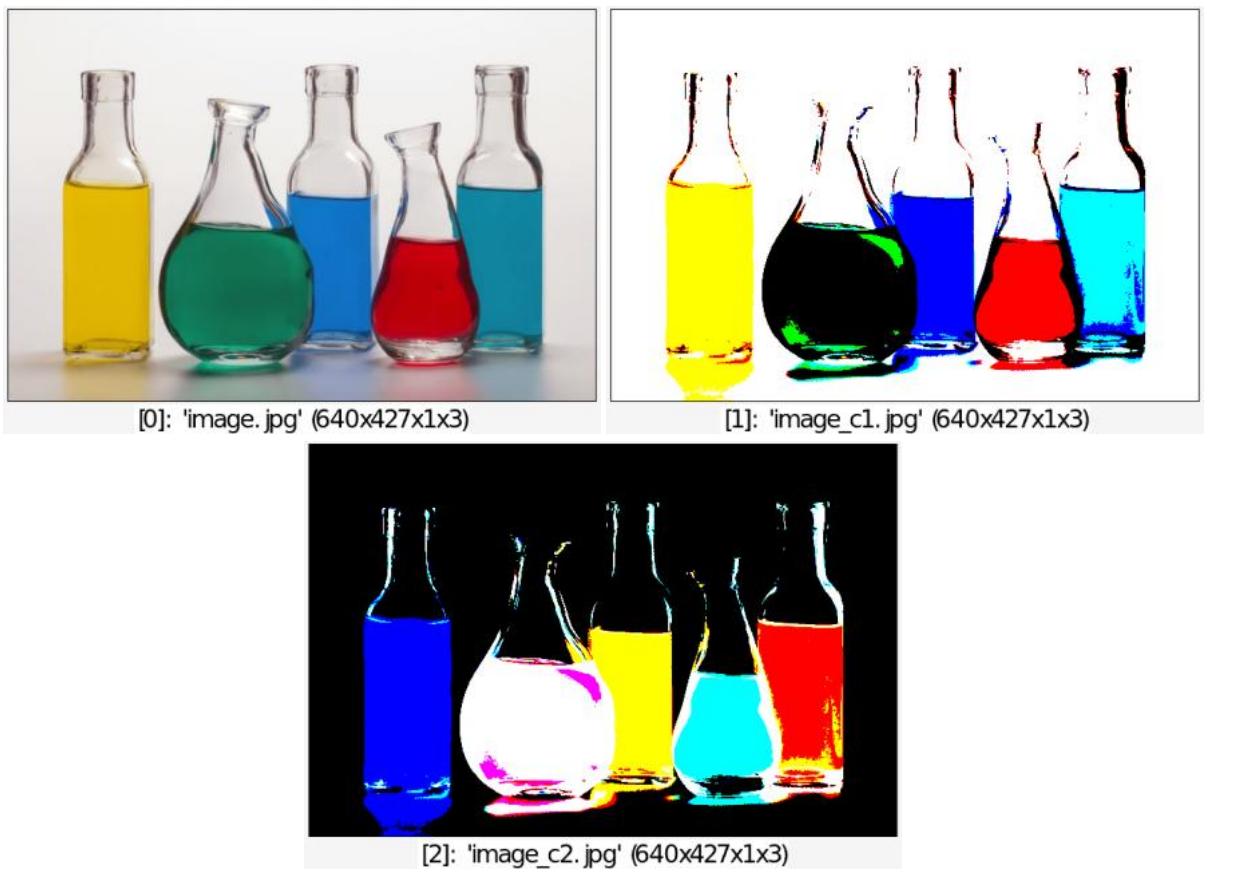
No arguments

**Description:**

Apply boolean not operation on selected images.

**Example of use:**

```
$ gmic image.jpg +ge 50% +not[-1]
```



## object3d

Built-in command

### Arguments:

- `[object3d],_x[%],_y[%],_z,_opacity,_rendering_mode,_is_double_sided={ 0 | 1 },_is_zbuffer={ 0 | 1 }`  
`,_focale,_light_x,_light_y,_light_z,_specular_lightness,_specular_shininess`

### Description:

Draw specified 3D object on selected images.

(equivalent to shortcut command `j3d`).

`rendering_mode` can be `{ 0=dots | 1=wireframe | 2=flat | 3=flat-shaded | 4=gouraud-shaded | 5=phong-shaded }`.

### Default values:

`x=y=z=0`, `opacity=1` and `is_zbuffer=1`. All other arguments take their default values from the 3D environment variables.

### Example of use:

```
$ gmic image.jpg torus3d 100,10 cone3d 30,-120 add3d[-2,-1] rotate3d.  
1,1,0,60 object3d[0] [-1],50%,50% keep[0]
```



## oct

### Arguments:

- `octal_int1,...`

### Description:

Print specified octal integers into their binary, decimal, hexadecimal and string representations.

---

## oct2dec

### Arguments:

- `octal_int1,...`

### Description:

Convert specified octal integers into their decimal representations.

---

## oklab2rgb

### No arguments

### Description:

Convert color representation of selected images from OKlab to RGB.

(see colorspace definition at: <https://bottosson.github.io/posts/oklab/> ).

### See also:

---

## old\_photo

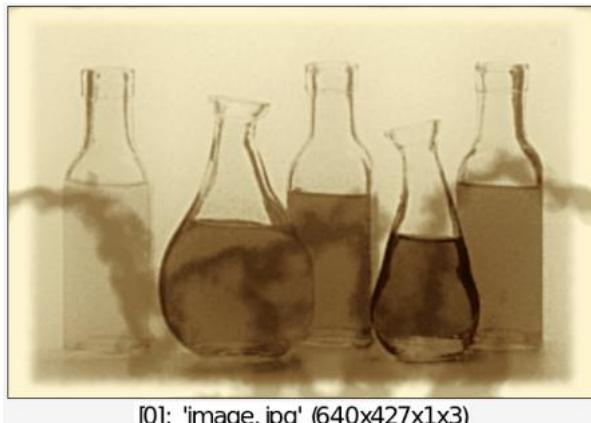
**No arguments**

**Description:**

Apply old photo effect on selected images.

**Example of use:**

```
$ gmic image.jpg old_photo
```



## oneminus

**No arguments**

**Description:**

For each selected image, compute one minus image.

**Example of use:**

```
$ gmic image.jpg normalize 0,1 +oneminus
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

## onfail

Built-in command

No arguments

Description:

Execute following commands when an error is encountered in the body of the `local...endlocal` block.

The status value is set with the corresponding error message.

Example of use:

```
$ gmic image.jpg +local blur -3 onfail mirror x endlocal
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

## opacity3d

Built-in command

Arguments:

- `_opacity`

Description:

Set opacity of selected 3D objects.

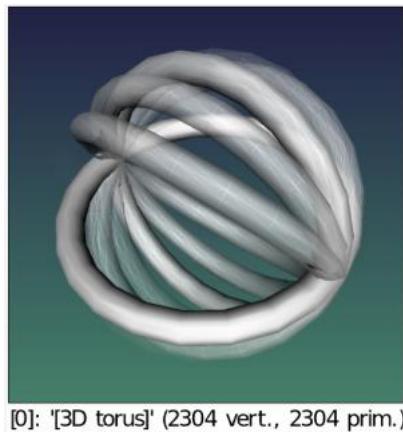
(equivalent to shortcut command `o3d`).

## Default values:

`opacity=1`.

## Example of use:

```
$ gmic torus3d 100,10 double3d 0 repeat 7 +rotate3d[-1] 1,0,0,20  
opacity3d[-1] {u} done add3d
```



---

or

Built-in command

## Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

## Description:

Compute the bitwise OR of selected images with specified value, image or mathematical expression, or compute the pointwise sequential bitwise OR of selected images.

(equivalent to shortcut command `|`).

## Examples of use:

- Example #1

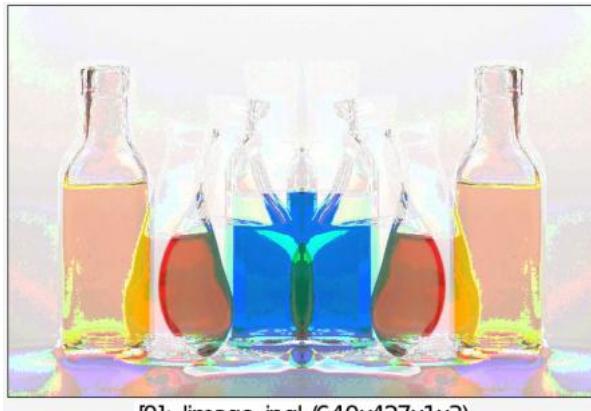
```
$ gmic image.jpg or 128
```



[0]: 'image.jpg' (640x427x1x3)

- **Example #2**

```
$ gmic image.jpg +mirror x or
```



[0]: 'image.jpg' (640x427x1x3)

---

## orientation

### No arguments

### Description:

Compute the pointwise orientation of vector-valued pixels in selected images.

This command has a [tutorial page](#).

### Example of use:

```
$ gmic image.jpg +orientation +norm[-2] negate[-1] mul[-2] [-1]  
reverse[-2,-1]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x3)

## orthogonalize

### Arguments:

- `_mode = { 0=orthogonalize | 1=orthonormalize }`

### Description:

Orthogonalize or orthonormalize selected matrices, using Modified Gram-Schmidt process.

### Default values:

`mode=0`.

## otsu

### Arguments:

- `_nb_levels>0`

### Description:

Hard-threshold selected images using Otsu's method.

The computed thresholds are returned as a list of values in the status.

## Default values:

`nb_levels=256`.

## Example of use:

```
$ gmic image.jpg luminance +otsu ,
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)

---

# output

Built-in command

## Arguments:

- `[type:]filename,_format_options`

## Description:

Output selected images as one or several numbered file(s).

(equivalent to shortcut command `o`).

## Default values:

'format\_options'=(undefined).

---

# output\_565

## Arguments:

- `"filename",reverse_endianness={ 0=false | 1=true }`

## Description:

Output selected images as raw RGB-565 files.

## **Default values:**

`reverse_endianness=0`.

---

## **output\_cube**

### **Arguments:**

- `"filename"`

### **Description:**

Output selected CLUTs as a .cube file (Adobe CLUT format).

---

## **output\_flo**

### **Arguments:**

- `"filename"`

### **Description:**

Output selected optical flow as a .flo file (vision.middlebury.edu file format).

---

## **output\_ggr**

### **Arguments:**

- `filename,_gradient_name`

### **Description:**

Output selected images as .ggr gradient files (GIMP).

If no gradient name is specified, it is deduced from the filename.

---

## **output\_gmz**

### **Arguments:**

- `filename,_datatype`

### **Description:**

Output selected images as .gmz files (G'MIC native file format).

**datatype** can be { `bool` | `uchar` | `char` | `ushort` | `short` | `uint` | `int` | `uint64` | `int64` | `float` | `double` }.

---

## output\_obj

### Arguments:

- `filename, _save_materials={ 0=no | 1=yes }`

### Description:

Output selected 3D objects as Wavefront 3D object files.

Set `save_materials` to `1` to produce a corresponding material file (`.mtl`) and eventually texture files.

Beware, the export to `.obj` files may be quite slow for large 3D objects.

### Default values:

`save_materials=1`.

---

## output\_text

### Arguments:

- `filename`

### Description:

Output selected images as text-data filenames.

(equivalent to shortcut command `ot`).

---

## outputn

### Arguments:

- `filename, _index`

### Description:

Output selected images as automatically numbered filenames in repeat...done loops.

(equivalent to shortcut command `on`).

---

## outputp

### Arguments:

- `prefix`

### Description:

Output selected images as prefixed versions of their original filenames.

(equivalent to shortcut command `op`).

### Default values:

`prefix=_`.

---

## outputw

### No arguments

### Description:

Output selected images by overwriting their original location.

(equivalent to shortcut command `ow`).

---

## outputx

### Arguments:

- `extension1,_extension2,_...,_extensionN,_output_at_same_location={ 0 | 1 }`

### Description:

Output selected images with same base filenames but for N different extensions.

(equivalent to shortcut command `ox`).

### Default values:

`output_at_same_location=0`.

---

# pack

## Arguments:

- `is_ratio_constraint={ 0 | 1 },_sort_criterion`

## Description:

Pack selected images into a single image.

The returned status contains the list of new (x,y) offsets for each input image.

Parameter `is_ratio_constraint` tells if the resulting image must tend to a square image.

## Default values:

`is_ratio_constraint=0` and `sort_criterion=max(w,h)`.

## Example of use:

```
$ gmic image.jpg repeat 10 +resize2dx[-1] 75% balance_gamma[-1] ${-rgb} done pack 0
```



---

# pack\_sprites

## Arguments:

- `_nb_scales>=0,0<=_min_scale<=100,_allow_rotation={ 0=0 deg. | 1=180 deg. | 2=90 deg. | 3=any },_spacing,_precision>=0,max_iterations>=0`

## Description:

Try to randomly pack as many sprites as possible onto the `empty` areas of an image.

Sprites can be eventually rotated and scaled during the packing process.

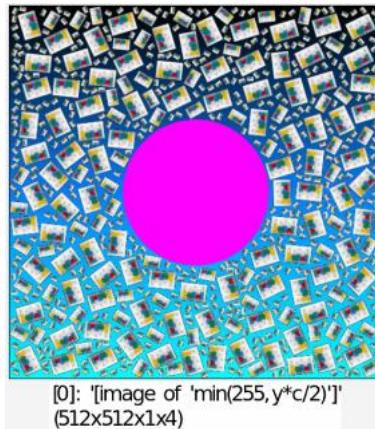
First selected image is the canvas that will be filled with the sprites.  
 Its last channel must be a binary mask whose zero values represent potential locations for drawing the sprites.  
 All other selected images represent the sprites considered for packing.  
 Their last channel must be a binary mask that represents the sprite shape (i.e. a 8-connected component).  
 The order of sprite packing follows the order of specified sprites in the image list.  
 Sprite packing is done on random locations and iteratively with decreasing scales.  
`nb_scales` sets the number of decreasing scales considered for all specified sprites to be packed.  
`min_scale` (in %) sets the minimal size considered for packing (specified as a percentage of the original sprite size).  
`spacing` can be positive or negative.  
`precision` tells about the desired number of failed trials before ending the filling process.

## Default values:

`nb_scales=5`, `min_scale=25`, `allow_rotation=3`, `spacing=1`, `precision=7` and `max_iterations=256`.

## Example of use:

```
$ gmic 512,512,1,3,"min(255,y*c/2)" 100%,100% circle  
50%,50%,100,1,255 append c image.jpg resize2dy[-1] 24 to_rgba  
pack_sprites 3,25
```



## padint

### Arguments:

- `number,_size>0`

### Description:

Return a integer with `size` digits (eventually left-padded with `0`).

## palette

## Arguments:

- `palette_name | palette_number`

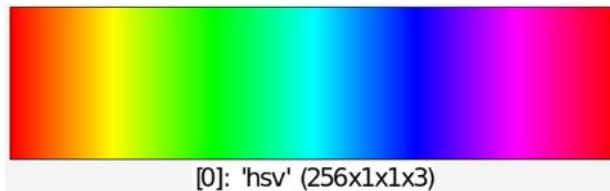
## Description:

Input specified color palette at the end of the image list.

`palette_name` can be { *default* | *hsv* | *lines* | *hot* | *cool* | *jet* | *flag* | *cube* | *rainbow* | *parula* | *spring* | *summer* | *autumn* | *winter* | *bone* | *copper* | *pink* | *vga* | *algae* | *amp* | *balance* | *curl* | *deep* | *delta* | *dense* | *diff* | *gray* | *haline* | *ice* | *matter* | *oxy* | *phase* | *rain* | *solar* | *speed* | *tarn* | *tempo* | *thermal* | *topo* | *turbid* | *aurora* | *hocuspocus* | *sr2* | *uzebox* | *amiga7800* | *amiga7800mess* | *fornaxvoid1* }

## Example of use:

```
$ gmic palette hsv
```



---

## parallel

[Built-in command](#)

## Arguments:

- `_wait_threads,"command1","command2",...`

## Description:

Execute specified commands in parallel, each in a different thread.

Parallel threads share the list of images.

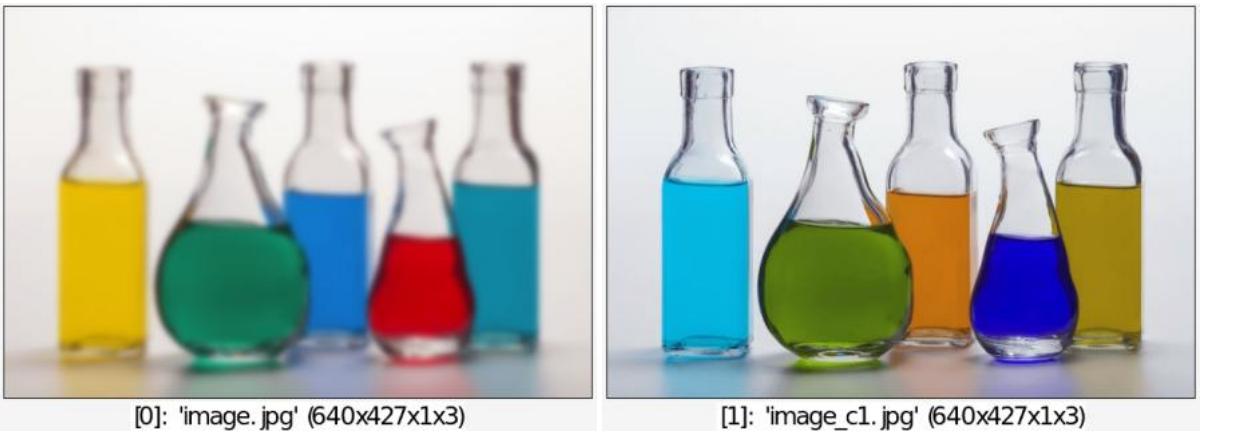
`wait_threads` can be { *0=when current environment ends* | *1=immediately* }.

## Default values:

`wait_threads=1`.

## Example of use:

```
$ gmic image.jpg [0] parallel "blur[0] 3","mirror[1] c"
```



---

## parametric3d

### Arguments:

- `_x(a,b),_y(a,b),_z(a,b),_amin,_amax,_bmin,_bmax,_res_a>0,_res_b>0,_res_x>0,...`

### Description:

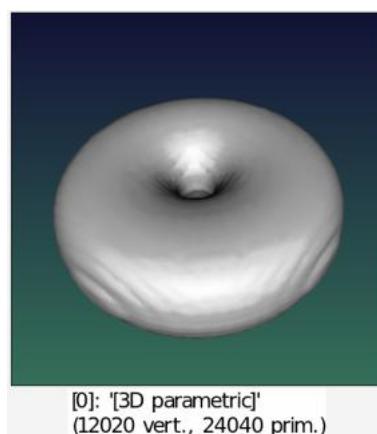
Input 3D object from specified parametric surface  $(a,b) \rightarrow (x(a,b), y(a,b), z(a,b))$ .

### Default values:

```
x=(2+cos(b))*sin(a), y=(2+cos(b))*cos(a), c=sin(b), amin=-pi, amax=pi, bmin=-pi, bmax=pi, res_a=512, res_b=res_a, res_x=64, res_y=res_x, res_z=res_y, smoothness=2% and isovalue=10%.
```

### Example of use:

```
$ gmic parametric3d ,
```



---

## parse\_cli

### Arguments:

- `_output_mode, _{ * | command_name }`

## Description:

Parse definition of "-documented commands and output info about them in specified output mode.

`output_mode` can be `{ ascii | bashcompletion | html | images | print }`.

## Default values:

`output_mode=print` and `command_name=*`.

---

# parse\_gmd

## No arguments

## Description:

Parse and tokenize selected images, viewed as text strings formatted with the G'MIC markdown syntax.

---

# parse\_gui

## Arguments:

- `_outputmode, _{ * | filter_name }`

## Description:

Parse selected filter definitions and generate info about filters in selected output mode.

`outputmode` can be `{ gmicol | json | list | print | ts | update | zart }`.

It is possible to define a custom output mode, by implementing the following commands (`outputmode` must be replaced by the name of the custom user output mode):

. `parse_gui_outputmode` : A command that outputs the parsing information with a custom format.

. `parse_gui_parseparams_outputmode` (optional): A simple command that returns 0 or 1. It tells the parser whether parameters of matching filter must be analyzed (slower) or not.

. `parse_gui_trigger_outputmode` (optional): A command that is called by the parser just before parsing the set of each matching filters.

Here is the list of global variables set by the parser, accessible in command

`parse_gui_outputmode`:

`$_nbfilters`: Number of matching filters.

`$_nongui` (stored as an image): All merged lines in the file that do not correspond to `#@gui` lines.

For each filter #F (#F in range [0, \$\_nbfilters-1]):

- `$_fF_name` : Filter name.
- `$_fF_path` : Full path.
- `$_fF_locale` : Filter locale (empty, if not specified).
- `$_fF_command` : Filter command.
- `$_fF_commandpreview` : Filter preview command (empty, if not specified).
- `$_fF_zoomfactor` : Default zoom factor (empty, if not specified).
- `$_fF_zoomaccurate` : Is preview accurate when zoom changes ? (can be `{ 0=false | 1=true }`).
- `$_fF_inputmode` : Default preferred input mode (empty, if not specified).
- `$_fF_hide` : Path of filter hid by current filter (for localized filters, empty if not specified).
- `$_fF_nbparams` : Number of parameters.

For each parameter #P of the filter #F (#P in range [0, \$\_fF\_nbparams-1]):

- `$_fF_pP_name` : Parameter name.
- `$_fF_pP_type` : Parameter type.
- `$_fF_pP_responsivity` : Parameter responsivity (can be `{ 0 | 1 }`).
- `$_fF_pP_visibility` : Parameter visibility.
- `$_fF_pP_propagation` : Propagation of the parameter visibility.
- `$_fF_pP_nbargs` : Number of parameter arguments.

For each argument #A of the parameter #P (#A in range [0, \$\_fF\_pP\_nbargs-1]):

- `$_fF_pP_aA` : Argument value

Default parameters: `filter_name=*` and `output_format=print`.

---

## pass

Built-in command

### Arguments:

- `_shared_state={ -1=status only | 0=non-shared (copy) | 1=shared | 2=adaptive }`

### Description:

Insert images from parent context of a custom command or a local environment.

Command selection (if any) stands for a selection of images in the parent context.

By default (adaptive shared state), selected images are inserted in a shared state if they do not belong

to the context (selection) of the current custom command or local environment as well. Typical use of command `pass` concerns the design of custom commands that take images as arguments.  
This command returns the list of corresponding indices in the status.

## Default values:

`shared_state=2`.

## Example of use:

```
$ gmic command "average : pass""1 add[^-1] [-1] remove[-1] div 2"
sample ? +mirror y +average[0] [1]
```



[0]: 'cameraman' (256x256x1x1)



[1]: 'cameraman\_c1' (256x256x1x1)



[2]: 'cameraman\_c1' (256x256x1x1)

---

## patches

### Arguments:

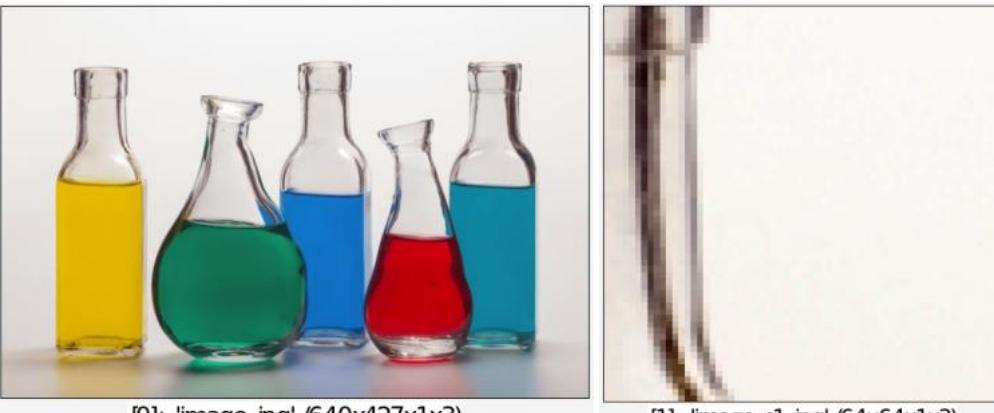
- `patch_width>0,patch_height>0,patch_depth>0,x0,y0,z0,_x1,_y1,_z1,...,_xN,_yN,`

### Description:

Extract N+1 patches from selected images, centered at specified locations.

## Example of use:

```
$ gmic image.jpg +patches
64,64,1,153,124,0,184,240,0,217,126,0,275,38,0
```



[0]: 'image.jpg' (640x427x1x3)



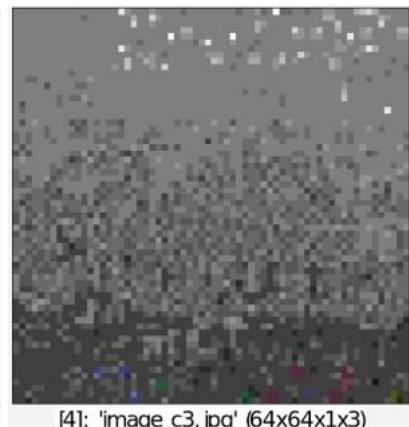
[1]: 'image\_c1.jpg' (64x64x1x3)



[2]: 'image\_c2.jpg' (64x64x1x3)



[3]: 'image\_c2.jpg' (64x64x1x3)



[4]: 'image\_c3.jpg' (64x64x1x3)

## path\_cache

**No arguments**

**Description:**

Return a path to store G'MIC data files for one user (whose value is OS-dependent).

## path\_current

**No arguments**

**Description:**

Return current folder from where G'MIC has been run.

## path\_gimp

**No arguments**

**Description:**

Return a path to store GIMP configuration files for one user (whose value is OS-dependent).

---

## path\_tmp

**No arguments**

### Description:

Return a path to store temporary files (whose value is OS-dependent).

---

## pca\_patch3d

### Arguments:

- `_patch_size>0, _M>0, _N>0, _normalize_input={ 0 | 1 }, _normalize_output={ 0 | 1 }, _lambda_xy`

### Description:

Get 3D patch-pca representation of selected images.

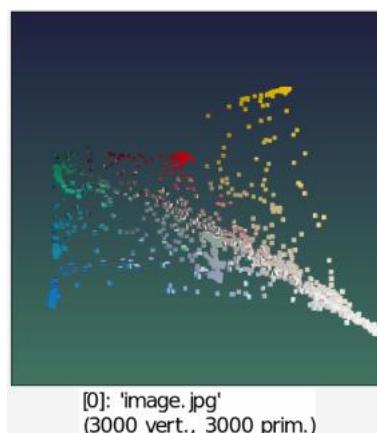
The 3D patch-pca is estimated from M patches on the input image, and displayed as a cloud of N 3D points.

### Default values:

`patch_size=7, M=1000, N=3000, normalize_input=1, normalize_output=0, and lambda_xy=0.`

### Example of use:

```
$ gmic image.jpg pca_patch3d 7
```



## pde\_flow

### Arguments:

- `_nb_iter>=0, _dt, _velocity_command, _keep_sequence={ 0 | 1 }`

### Description:

Apply iterations of a generic PDE flow on selected images.

### Default values:

`nb_iter=10`, `dt=30`, `velocity_command=laplacian` and `keep_sequence=0`.

### Example of use:

```
$ gmic image.jpg +pde_flow 20
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## pencilbw

### Arguments:

- `_size>=0, _amplitude>=0`

### Description:

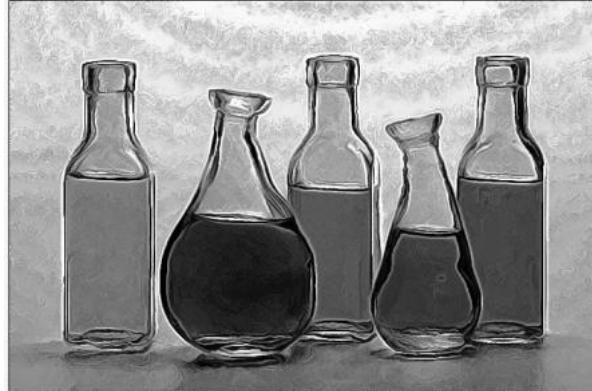
Apply B&W pencil effect on selected images.

### Default values:

`size=0.3` and `amplitude=60`.

### Example of use:

```
$ gmic image.jpg pencilbw ,
```



[0]: 'image.jpg' (640x427x1x1)

---

## percentile

### Arguments:

- `[mask], 0<=_min_percentile[%]<=100, 0<=_max_percentile[%]<=100.`

### Description:

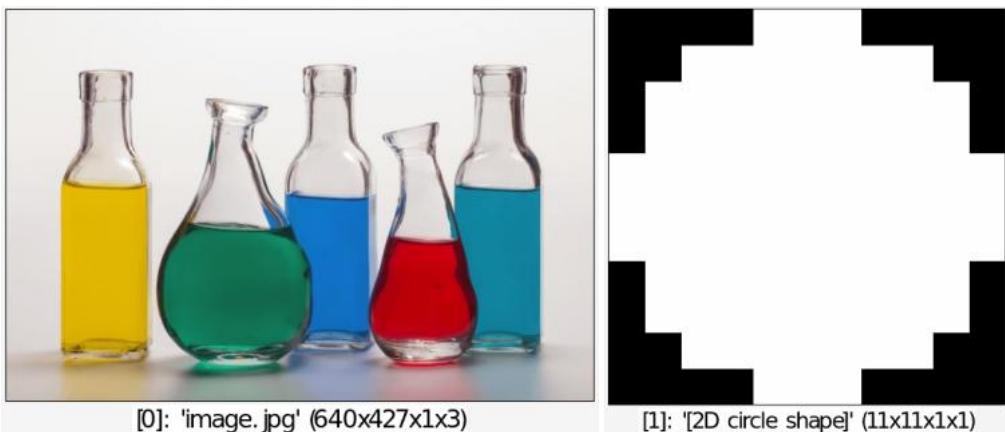
Apply percentile averaging filter to selected images.

### Default values:

`min_percentile=0` and `max_percentile=100`.

### Example of use:

```
$ gmic image.jpg shape_circle 11,11 +percentile[0] [1],25,75
```



## periodize\_poisson

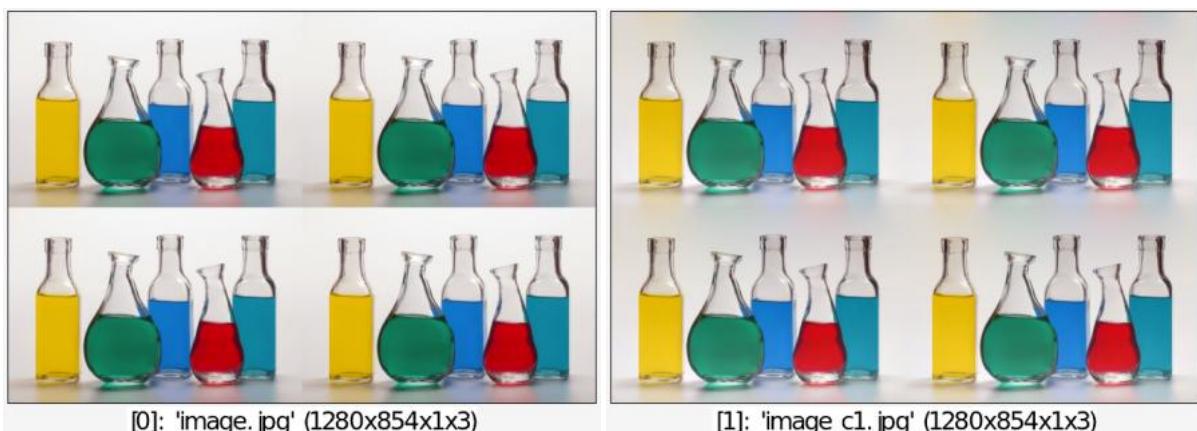
**No arguments**

**Description:**

Periodize selected images using a Poisson solver in Fourier space.

**Example of use:**

```
$ gmic image.jpg +periodize_poisson array 2,2,2
```



Built-in command

# permute

## Arguments:

- `permutation_string`

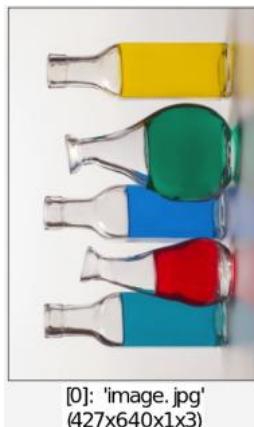
## Description:

Permute selected image axes by specified permutation.

`permutation` is a combination of the character set `{x|y|z|c}`,  
e.g. `xycz`, `cxyz`, ...

## Example of use:

```
$ gmic image.jpg permute yxzc
```



---

# peronamalik\_flow

## Arguments:

- `K_factor>0, _nb_iter>=0, _dt, _keep_sequence={ 0 | 1 }`

## Description:

Apply iterations of the Perona-Malik flow on selected images.

## Default values:

`K_factor=20`, `nb_iter=5`, `dt=5` and `keep_sequence=0`.

## Example of use:

```
$ gmic image.jpg +heat_flow 20
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## phase\_correlation

### Arguments:

- [destination]

### Description:

Estimate translation vector between selected source images and specified destination.

### Example of use:

```
$ gmic image.jpg +shift -30,-20 +phase_correlation[0] [1] unroll[-1]  
y
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

[2]: 'phase correlation\_Lc1'  
(1x3x1x1)

---

## piechart

### Arguments:

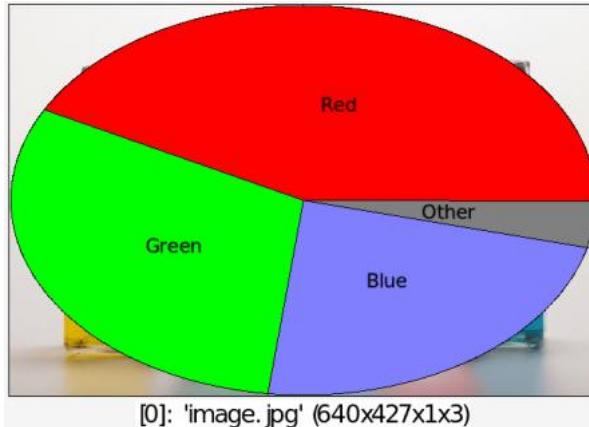
- `label_height>=0, label_R, label_G, label_B, "label1", value1, R1, G1, B1, ..., "labelN"`

### Description:

Draw pie chart on selected (RGB) images.

## Example of use:

```
$ gmic image.jpg piechart  
25,0,0,0,"Red",55,255,0,0,"Green",40,0,255,0,"Blue",30,128,128,255,"Other",5
```



---

## pixelize

### Arguments:

- `_scale_x>0,_scale_y>0,_scale_z>0`

### Description:

Pixelize selected images with specified scales.

### Default values:

`scale_x=20` and `scale_y=scale_z=scale_x`.

## Example of use:

```
$ gmic image.jpg +pixelize ,
```



---

# pixelsort

## Arguments:

- `_ordering={ + | - },_axis={ x | y | z | xy | yx }`,  
`_,[_sorting_criterion],_,[_mask]`

## Description:

Apply a 'pixel sorting' algorithm on selected images, as described in the page :

<http://satyarth.me/articles/pixel-sorting/>.

## Default values:

`ordering=+`, `axis=x` and `sorting_criterion=mask=(undefined)`.

## Example of use:

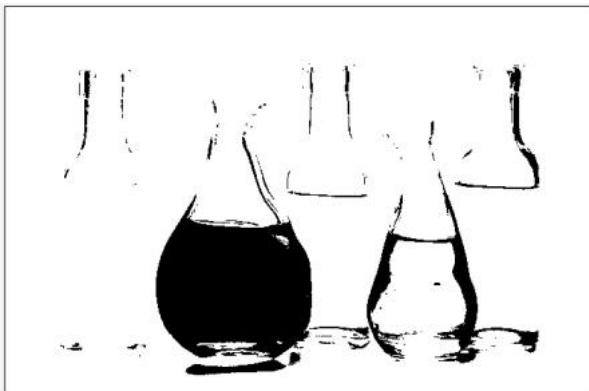
```
$ gmic image.jpg +norm +ge[-1] 30% +pixelsort[0] +,y,[1],[2]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c2.jpg' (640x427x1x1)



[3]: 'image\_c1.jpg' (640x427x1x3)

---

# plane3d

## Arguments:

- `_size_x,_size_y,_nb_subdivisions_x>0,_nb_subdivisions_y>0`

## Description:

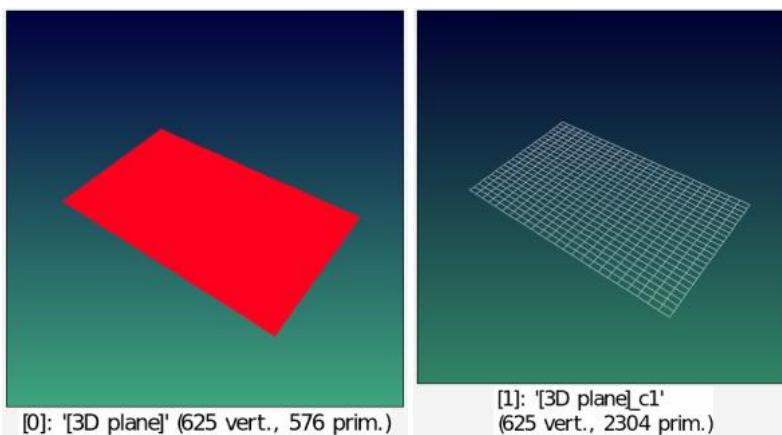
Input 3D plane at (0,0,0), with specified geometry.

## Default values:

`size_x=1`, `size_y=size_x` and `nb_subdivisions_x=nb_subdivisions_y=24`.

## Example of use:

```
$ gmic plane3d 50,30 +primitives3d 1 color3d[-2] ${-rgb}
```



---

# plasma

Built-in command

## Arguments:

- `_alpha,_beta,_scale>=0`

## Description:

Draw a random colored plasma fractal on selected images.

This command implements the so-called [Diamond-Square](#) algorithm.

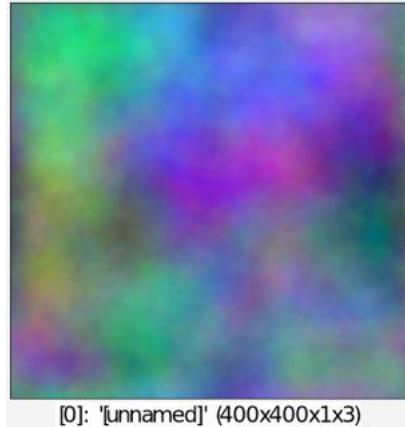
## Default values:

`alpha=1`, `beta=1` and `scale=8`.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic 400,400,1,3 plasma
```



---

## plot

Built-in command

### Arguments:

- `_plot_type,_vertex_type,_xmin,_xmax,_ymin,_ymax,_exit_on_anykey={ 0 | 1 }` or
- `'formula',_resolution>=0,_plot_type,_vertex_type,_xmin,xmax,_ymin,_ymax,_exit_on_anykey={ 0 | 1 }`

### Description:

Display selected images or formula in an interactive viewer (use the instant display 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 displayed xy-axes.

### Default values:

`plot_type=1`, `vertex_type=1`, `'xmin=xmax=ymin=ymax=0 (auto)'` and `exit_on_anykey=0`.

---

## plot2value

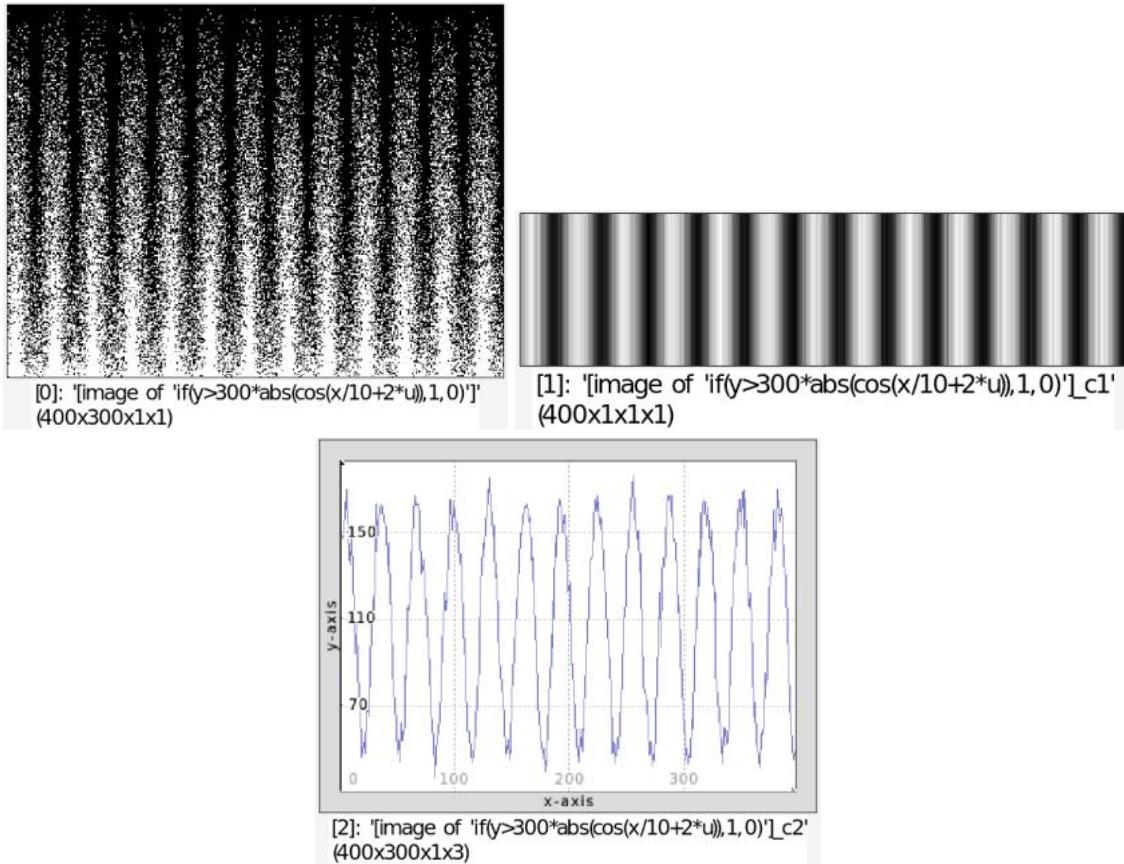
### No arguments

### Description:

Retrieve values from selected 2D graph plots.

### Example of use:

```
$ gmic 400,300,1,1,'if(y>300*abs(cos(x/10+2*u)),1,0)' +plot2value  
+display_graph[-1] 400,300
```



# point

Built-in command

## Arguments:

- `x[%],_y[%],_z[%],_opacity,_color1,...`

## Description:

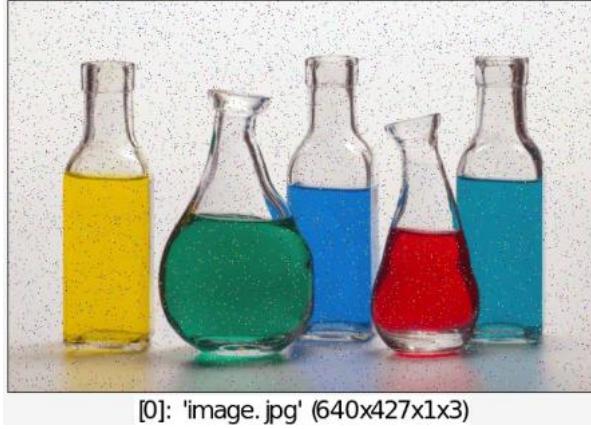
Set specified colored pixel on selected images.

## Default values:

`z=0`, `opacity=1` and `color1=0`.

## Example of use:

```
$ gmic image.jpg repeat 10000 point {u(100)}%,{u(100)}%,0,1,${-rgb}  
done
```



## point3d

### Arguments:

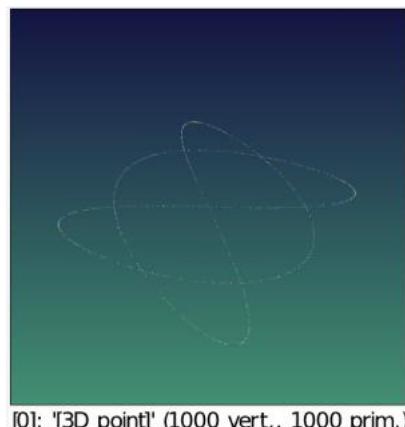
- `x0,y0,z0`

### Description:

Input 3D point at specified coordinates.

### Example of use:

```
$ gmic repeat 1000 a={$>*pi/500} point3d {cos(3*a)},{sin(2*a)},0  
color3d[-1] ${-rgb} done add3d
```



## pointcloud

### Arguments:

- `_type = { -X=-X-opacity | 0=binary | 1=cumulative | 2=label | 3=retrieve  
coordinates },_width,_height>0,_depth>0`

### Description:

Render a set of point coordinates, as a point cloud in a 1D/2D or 3D binary image

(or do the reverse, i.e. retrieve coordinates of non-zero points from a rendered point cloud).

Input point coordinates can be a  $N \times M \times 1 \times 1$ ,  $N \times 1 \times 1 \times M$  or  $1 \times N \times 1 \times M$  image, where  $N$  is the number of points,

and  $M$  the point coordinates.

If ' $M > 3$ ', the 3-to- $M$  components sets the  $(M-3)$ -dimensional color at each point.

Parameters `width`, `height` and `depth` are related to the size of the final image :

- If set to 0, the size is automatically set along the specified axis.
- If set to  $N > 0$ , the size along the specified axis is  $N$ .
- If set to  $N < 0$ , the size along the specified axis is at most  $N$ .

Points with coordinates that are negative or higher than specified (`width`, `height`, `depth`) are not plotted.

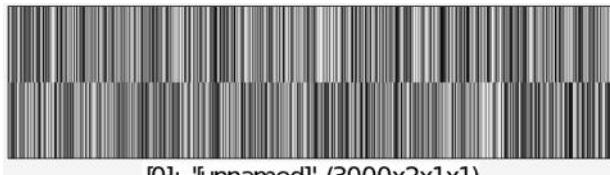
## Default values:

`type=0` and `max_width=max_height=max_depth=0`.

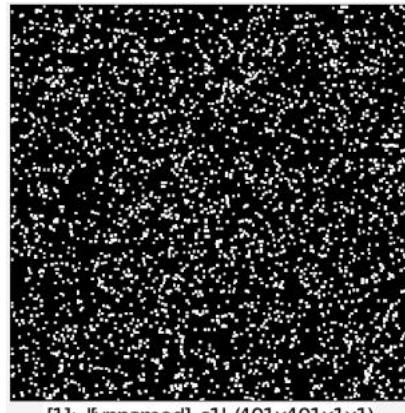
## Examples of use:

### • Example #1

```
$ gmic 3000,2 rand 0,400 +pointcloud 0 dilate[-1] 3
```



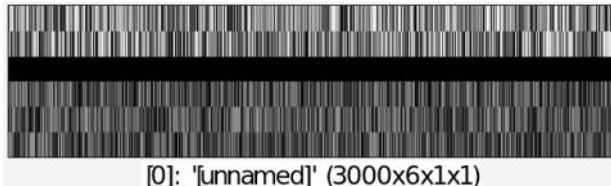
[0]: '[unnamed]' (3000x2x1x1)



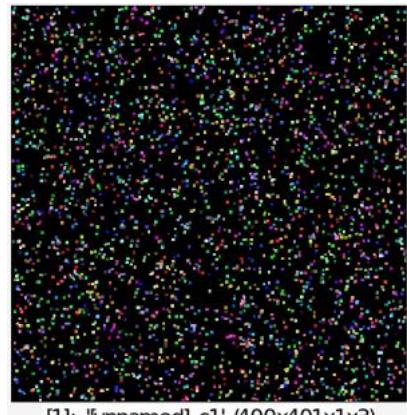
[1]: '[unnamed]\_c1' (401x401x1x1)

### • Example #2

```
$ gmic 3000,2 rand 0,400 {w} {w},3 rand[-1] 0,255 append y  
+pointcloud 0 dilate[-1] 3
```



[0]: '[unnamed]' (3000x6x1x1)



[1]: '[unnamed]\_c1' (400x401x1x3)

---

## pointcloud3d

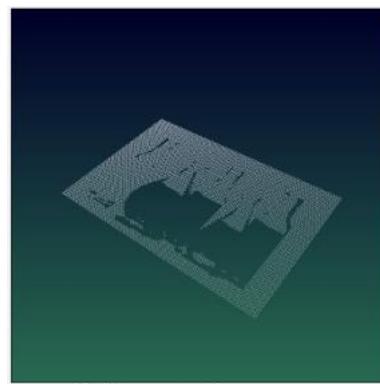
**No arguments**

**Description:**

Convert selected planar or volumetric images to 3D point clouds.

**Example of use:**

```
$ gmic image.jpg luminance resize2dy 100 threshold 50% mul 255  
pointcloud3d color3d[-1] 255,255,255
```



[0]: 'image.jpg'  
(10975 vert., 10975 prim.)

---

## polar2complex

**No arguments**

**Description:**

Compute polar to complex transforms of selected images.

# polar2euclidean

## Arguments:

- `_center_x[%],_center_y[%],_stretch_factor>0,_boundary_conditions={0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

## Description:

Apply euclidean to polar transform on selected images.

## Default values:

`center_x=center_y=50%`, `stretch_factor=1` and `boundary_conditions=1`.

## Example of use:

```
$ gmic image.jpg +euclidean2polar ,
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

# polaroid

## Arguments:

- `_size1>=0,_size2>=0`

## Description:

Create polaroid effect in selected images.

## Default values:

`size1=10` and `size2=20`.

## Example of use:

```
$ gmic image.jpg to_rgba polaroid 5,30 rotate 20 drop_shadow , drgba
```



[0]: 'image.jpg' (860x806x1x3)

---

## polka\_dots

### Arguments:

- `diameter>=0, _density, _offset1, _offset2, _angle, _aliasing, _shading, _opacity, _c`

### Description:

Draw dots pattern on selected images.

### Default values:

`density=20, offset1=offset2=50, angle=0, aliasing=10, shading=1, opacity=1`  
and `color=255`.

### Example of use:

```
$ gmic image.jpg polka_dots 10,15,0,0,20,10,1,0.5,0,128,255
```



[0]: 'image.jpg' (640x427x1x3)

---

## polygon

[Built-in command](#)

### Arguments:

- `N>=1,x1[%],y1[%],...,xN[%],yN[%],_opacity,_pattern,_color1,...`

## Description:

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 values:

`opacity=1`, `pattern=(undefined)` and `color1=0`.

## Examples of use:

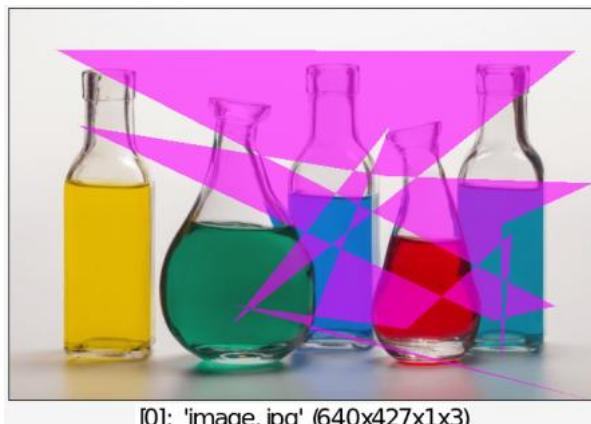
- Example #1

```
$ gmic image.jpg polygon  
4,20%,20%,80%,30%,80%,70%,20%,80%,0.3,0,255,0 polygon  
4,20%,20%,80%,30%,80%,70%,20%,80%,1,0xFFFFFFFF,255
```



- Example #2

```
$ gmic image.jpg 2,16,1,1,'u(if(x,{h},{w}))' polygon[-2] {h},  
{^},0.6,255,0,255 remove[-1]
```



# polygonize

## Arguments:

- `_warp_amplitude>=0, _smoothness[%]>=0, _min_area[%]>=0, _resolution_x[%]>0, _res`

## Description:

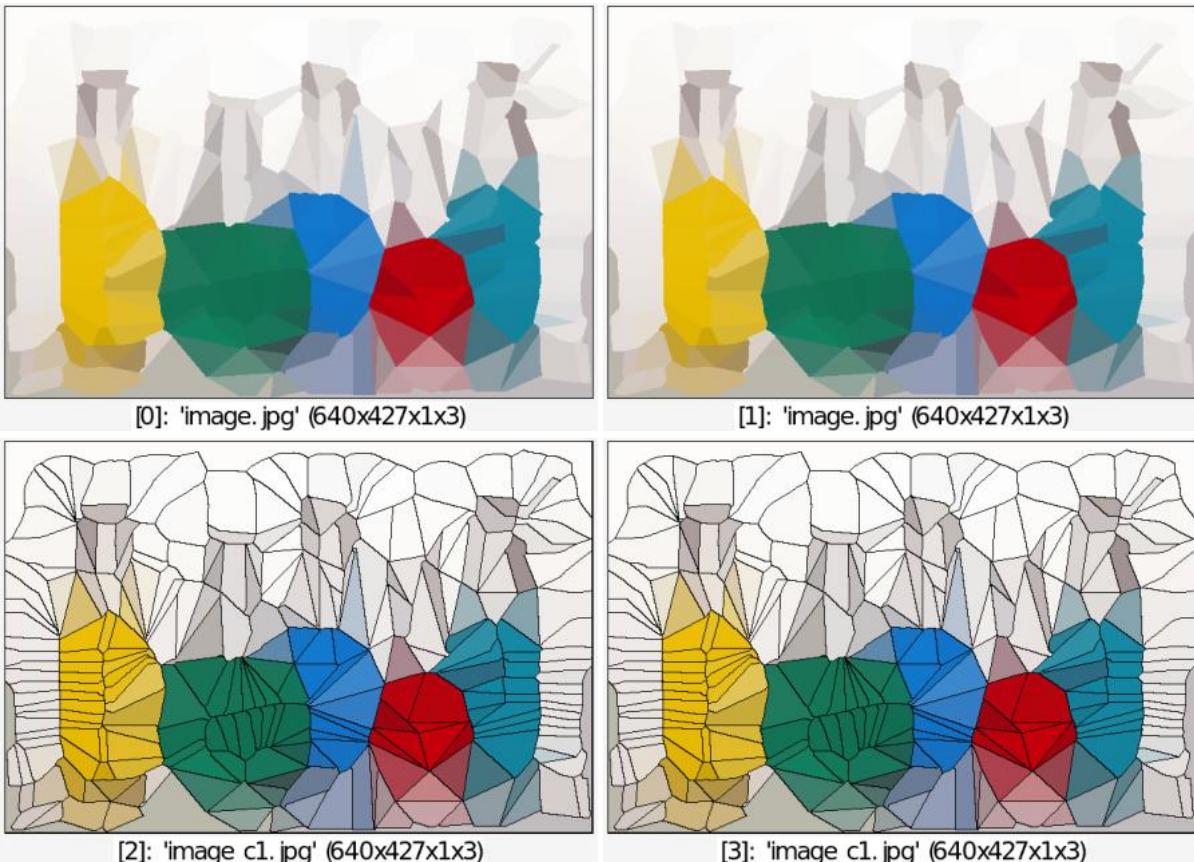
Apply polygon effect on selected images.

## Default values:

```
warp_amplitude=300, smoothness=2%, min_area=0.1%,
resolution_x=resolution_y=10%.
```

## Example of use:

```
$ gmic image.jpg image.jpg polygonize 100,10 +fill "I!=J(1) ||
I!=J(0,1)?[0,0,0]:I"
```



# pose3d

## Arguments:

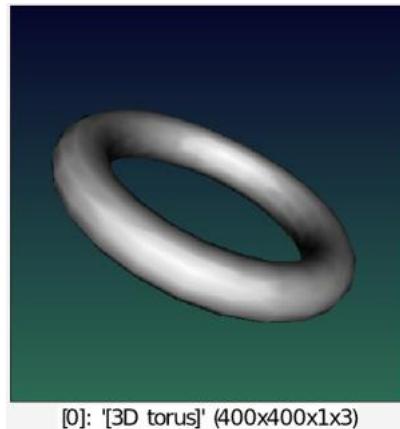
- `p1,...,p12`

## Description:

Apply 3D pose matrix to selected 3D objects.

## Example of use:

```
$ gmic torus3d 100,20 pose3d  
0.152437,1.20666,-0.546366,0,-0.535962,0.559129,1.08531,0,1.21132,0.0955431,  
snapshot3d 400
```



---

## poster\_edges

### Arguments:

- `0<=_edge_threshold<=100,0<=_edge_shade<=100,_edge_thickness>=0,_edge_antiali`

## Description:

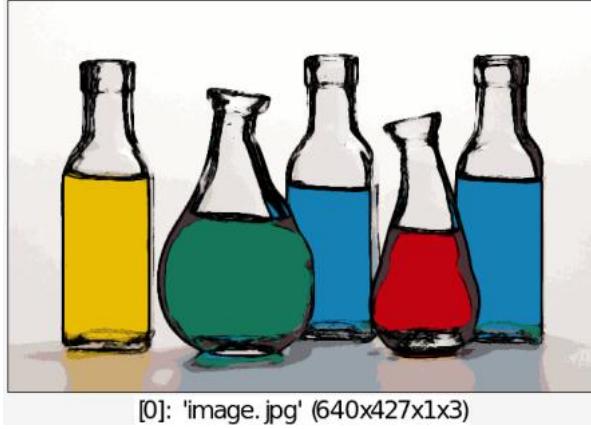
Apply poster edges effect on selected images.

## Default values:

`edge_threshold=40, edge_shade=5, edge_thickness=0.5, edge_antialiasing=10,`  
`posterization_level=12` and `posterization_antialiasing=0`.

## Example of use:

```
$ gmic image.jpg poster_edges ,
```



---

## poster\_hope

### Arguments:

- `_smoothness>=0`

### Description:

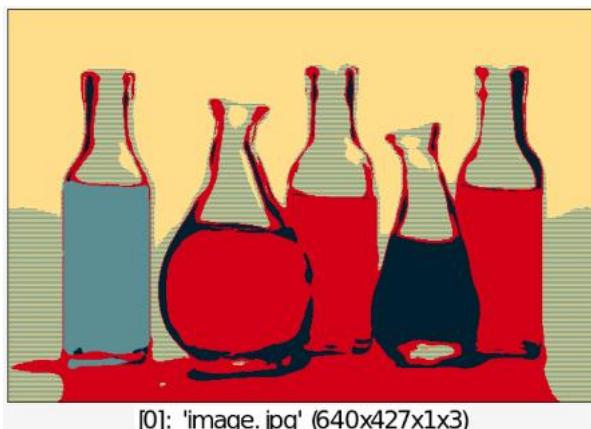
Apply Hope stencil poster effect on selected images.

### Default values:

`smoothness=3`.

### Example of use:

```
$ gmic image.jpg poster_hope ,
```



---

## pow

Built-in command

### Arguments:

- `value[%]` or

- [image] or
- 'formula' or
- (no arg)

## Description:

Raise selected images to the power of specified value, image or mathematical expression, or compute the pointwise sequential powers of selected images.

(equivalent to shortcut command  $\wedge$ ).

## Examples of use:

- Example #1

```
$ gmic image.jpg div 255 +pow 0.5 mul 255
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

- Example #2

```
$ gmic image.jpg gradient pow 2 add pow 0.2
```



[0]: 'image.jpg' (640x427x1x3)

## primitives3d

### Arguments:

- mode

## Description:

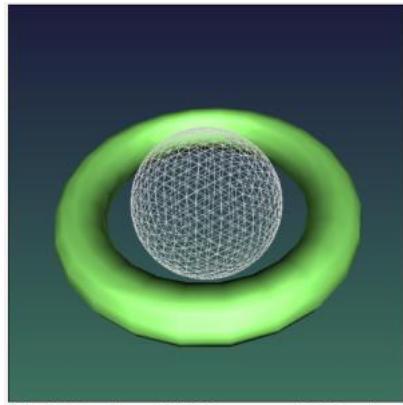
Convert primitives of selected 3D objects.

(equivalent to shortcut command `p3d`).

`mode` can be `{ 0=points | 1=outlines | 2=non-textured }`.

## Example of use:

```
$ gmic sphere3d 30 primitives3d 1 torus3d 50,10 color3d[-1] ${-rgb}
add3d
```



[0]: '[3D sphere]' (930 vert., 4128 prim.)

# print

Built-in command

No arguments

## Description:

Output information on selected images, on the standard error (stderr).

(equivalent to shortcut command `p`).

When invoked with a `+` prefix (i.e. `+print`), the command output its message on stdout rather than stderr.

# progress

Built-in command

## Arguments:

- `0<=value<=100` or
- `-1`

## Description:

Set the progress index of the current processing pipeline.

This command is useful only when G'MIC is used by an embedding application.

---

# projections3d

## Arguments:

- `_x[%],_y[%],_z[%],_is_bounding_box={ 0 | 1 }`

## Description:

Generate 3D xy,xz,yz projection planes from specified volumetric images.

---

# pseudogray

## Arguments:

- `_max_increment>=0,_JND_threshold>=0,_bits_depth>0`

## Description:

Generate pseudogray colormap with specified increment and perceptual threshold.

If `JND_threshold` is 0, no perceptual constraints are applied.

## Default values:

`max_increment=5`, `JND_threshold=2.3` and `bits_depth=8`.

## Example of use:

```
$ gmic pseudogray 5
```



# psnr

## Arguments:

- [max\\_value](#)

## Description:

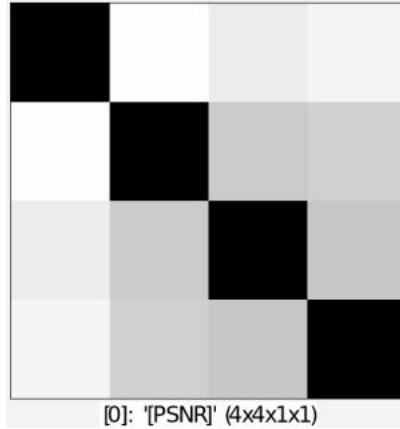
Compute PSNR (Peak Signal-to-Noise Ratio) matrix between selected images.

## Default values:

`max_value=255`.

## Example of use:

```
$ gmic image.jpg +noise 30 +noise[0] 35 +noise[0] 38 cut[-1] 0,255
psnr 255 replace_inf 0
```



## puzzle

### Arguments:

- [width>0, height>0, M>=1, N>=1, curvature, centering, connectors\\_variability](#)

## Description:

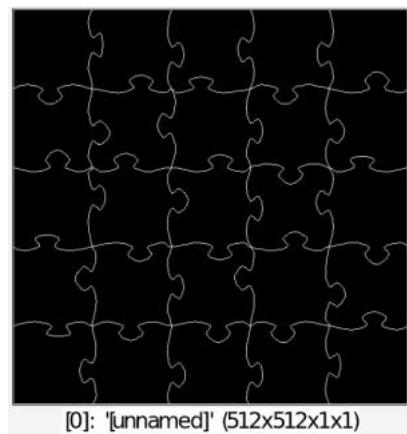
Input puzzle binary mask with specified size and geometry.

## Default values:

`width=height=512, M=N=5, curvature=0.5, centering=0.5, connectors_variability=0.5` and `resolution=64`.

## Example of use:

```
$ gmic puzzle ,
```



---

## pyramid3d

### Arguments:

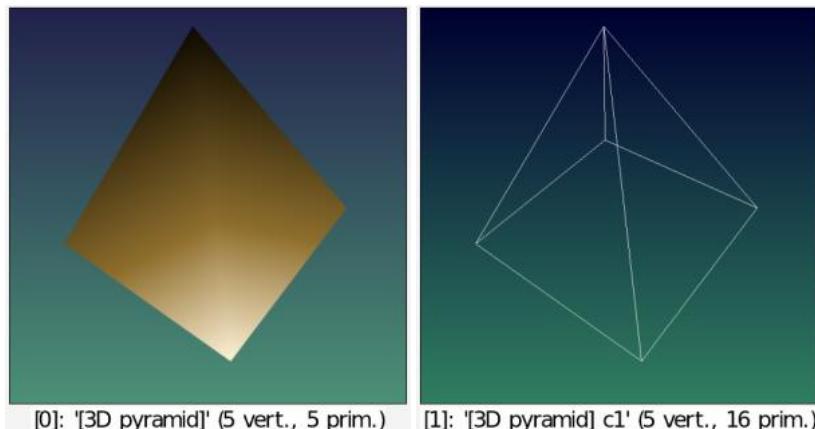
- `width, height`

### Description:

Input 3D pyramid at (0,0,0), with specified geometry.

### Example of use:

```
$ gmic pyramid3d 100,-100 +primitives3d 1 color3d[-2] ${-rgb}
```



---

## quadrangle3d

### Arguments:

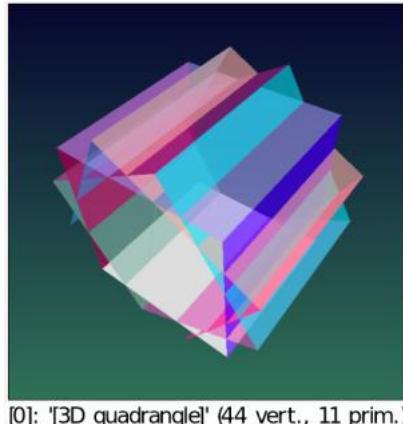
- `x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3`

### Description:

Input 3D quadrangle at specified coordinates.

## Example of use:

```
$ gmic quadrangle3d -10,-10,10,10,-10,10,10,10,10,-10,10,10 repeat 10  
+rotate3d[-1] 0,1,0,30 color3d[-1] ${-rgb},0.6 done add3d mode3d 2
```



---

## quadratize\_tiles

### Arguments:

- `M>0, _N>0`

### Description:

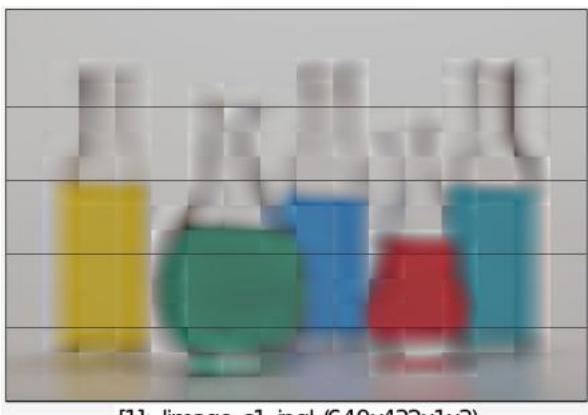
Quadratize MxN tiles on selected images.

### Default values:

`N=M`.

### Example of use:

```
$ gmic image.jpg +quadratize_tiles 16
```



# quantize

## Arguments:

- `nb_levels>=1,_keep_values={ 0 | 1 },_quantization_type={ -1=median-cut | 0=k-means | 1=uniform }`

## Description:

Quantize selected images.

## Default values:

`keep_values=1` and `quantization_type=0`.

## Examples of use:

- Example #1

```
$ gmic image.jpg luminance +quantize 3
```



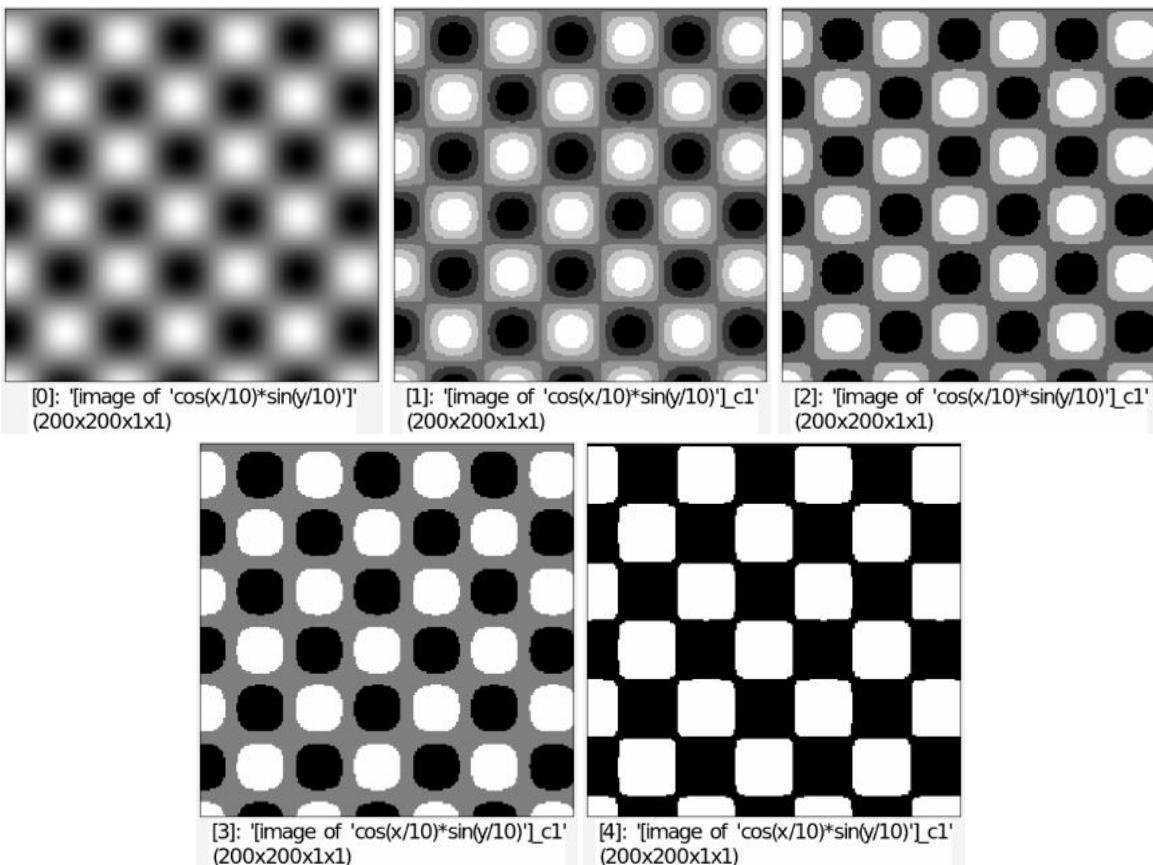
[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)

- Example #2

```
$ gmic 200,200,1,1,'cos(x/10)*sin(y/10)' +quantize[0] 6 +quantize[0]
4 +quantize[0] 3 +quantize[0] 2
```



## quantize\_area

### Arguments:

- `_min_area>0`

### Description:

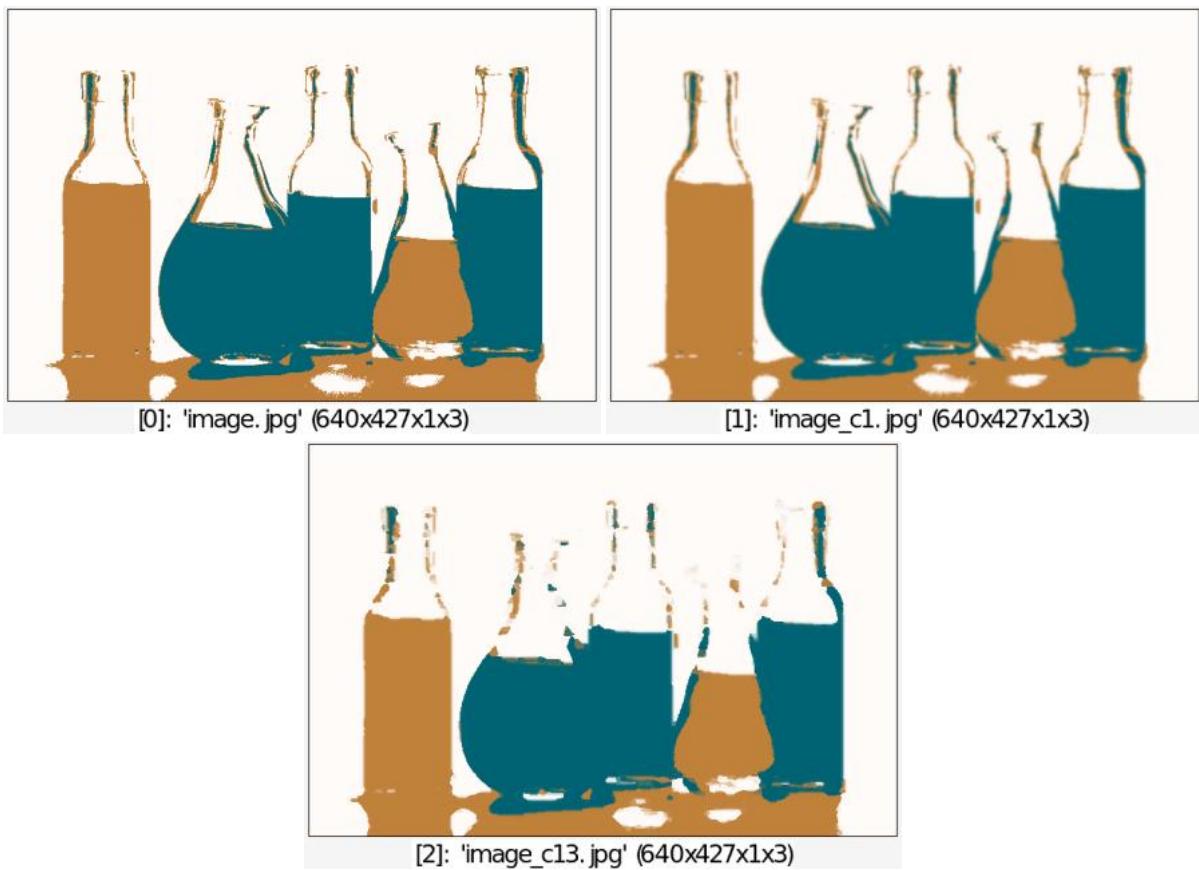
Quantize selected images such that each flat region has an area greater or equal to `min_area`.

### Default values:

`min_area=10`.

### Example of use:

```
$ gmic image.jpg quantize 3 +blur 1 round[-1] +quantize_area[-1] 2
```



---

## quit

Built-in command

No arguments

**Description:**

Quit G'MIC interpreter.

(equivalent to shortcut command `q`).

---

## quiver

**Arguments:**

- `[function_image],_sampling[%]>0,_factor>=0,_is_arrow={ 0 | 1 },_opacity,_color1,...`

**Description:**

Draw specified 2D vector/orientation field on selected images.

**Default values:**

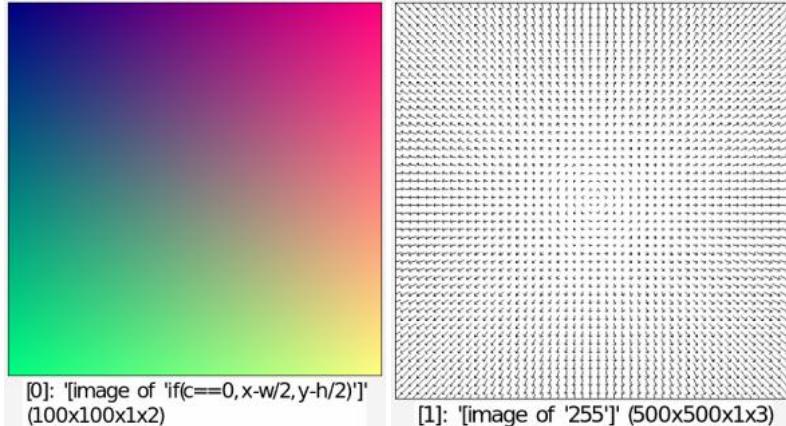
`sampling=5%, factor=1, is_arrow=1, opacity=1, pattern=(undefined)`

and `color1=0`.

## Examples of use:

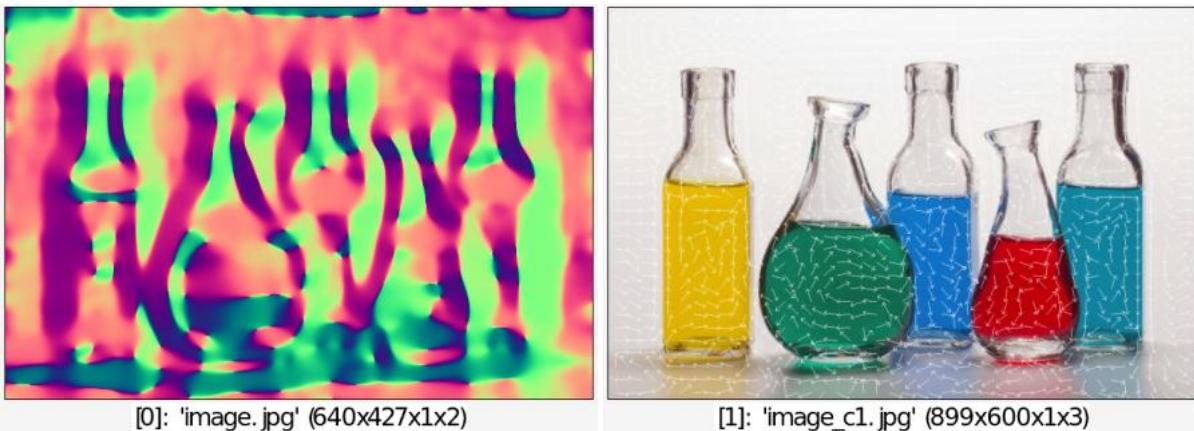
- **Example #1**

```
$ gmic 100,100,1,2,'if(c==0,x-w/2,y-h/2)' 500,500,1,3,255 quiver[-1]
[-2],10
```



- **Example #2**

```
$ gmic image.jpg +resize2dy 600 luminance[0] gradient[0] mul[1] -1
reverse[0,1] append[0,1] c blur[0] 8 orientation[0] quiver[1]
[0],20,1,1,0.8,255
```



---

## rad2deg

**No arguments**

### Description:

Convert pointwise angle values of selected images, from radians to degrees (apply `i*180/pi`).

---

# raindrops

## Arguments:

- `_amplitude,_density>=0,_wavelength>=0,_merging_steps>=0`

## Description:

Apply raindrops deformation on selected images.

## Default values:

`amplitude=80,density=0.1,wavelength=1` and `merging_steps=0`.

## Example of use:

```
$ gmic image.jpg +raindrops ,
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

# rand

Built-in command

## Arguments:

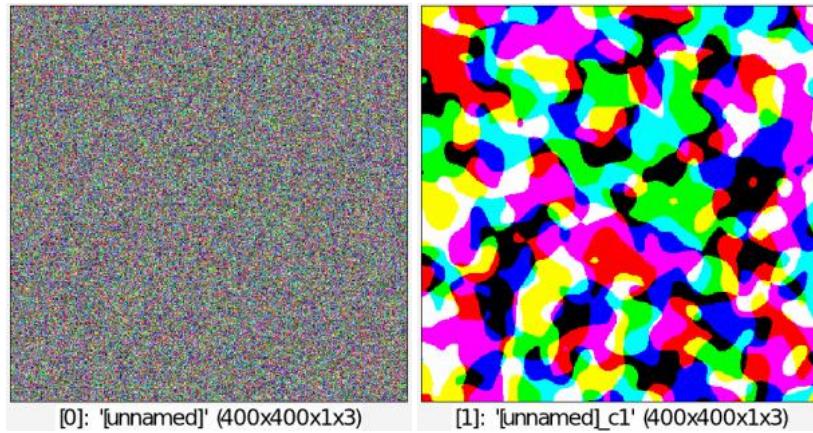
- `{ value0[%] | [image0] },_{ value1[%] | [image1] }` or
- `[image]`

## Description:

Fill selected images with random values uniformly distributed in the specified range.

## Example of use:

```
$ gmic 400,400,1,3 rand -10,10 +blur 10 sign[-1]
```



---

## random3d

### Arguments:

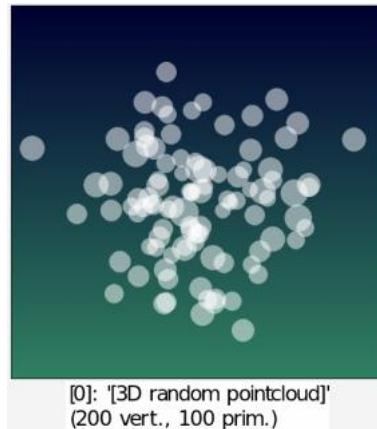
- `nb_points>=0`

### Description:

Input random 3D point cloud in  $[0,1]^3$ .

### Example of use:

```
$ gmic random3d 100 circles3d 0.1 opacity3d 0.5
```



---

## random\_pattern

### Arguments:

- `_width>0,_height>0,_min_detail_level>=0`

### Description:

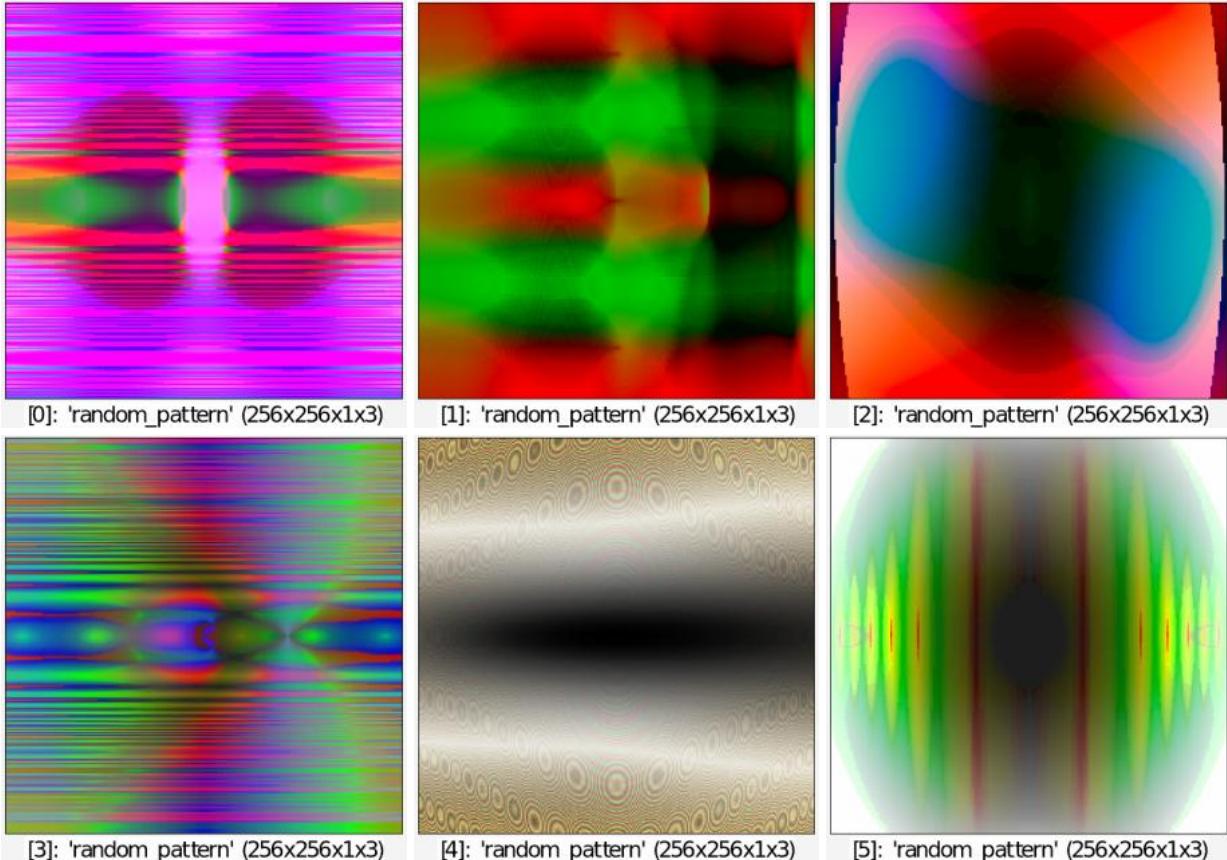
Insert a new RGB image of specified size at the end of the image list, rendered with a random pattern.

## Default values:

`width=height=512` and `min_detail_level=2`.

## Example of use:

```
$ gmic repeat 6 random_pattern 256 done
```



---

## rbf

### Arguments:

- `dx,_x0,_x1,_phi(r)` or
- `dx,dy,_x0,_y0,_x1,_y1,_phi(r)` or
- `dx,dy,dz,x0,y0,z0,x1,y1,z1,phi(r)`

### Description:

Reconstruct 1D/2D or 3D image from selected sets of keypoints, by RBF-interpolation.

A set of keypoints is represented by a vector-valued image, where each pixel represents a single keypoint.

Vector components of a keypoint have the following meaning:

- For 1D reconstruction: [  $x_k$ ,  $f_1(k)$ , ...,  $f_N(k)$  ].
- For 2D reconstruction: [  $x_k, y_k$ ,  $f_1(k)$ , ...,  $f_N(k)$  ].
- For 3D reconstruction: [  $x_k, y_k, z_k$ ,  $f_1(k)$ , ...,  $f_N(k)$  ].

Values  $x_k$ ,  $y_k$  and  $z_k$  are the spatial coordinates of keypoint  $k$ .

Values  $f_1(k), \dots, f_N(k)$  are the  $N$  components of the vector value of keypoint  $k$ .

The command reconstructs an image with specified size  $dx \times dy \times dz$ , with  $N$  channels.

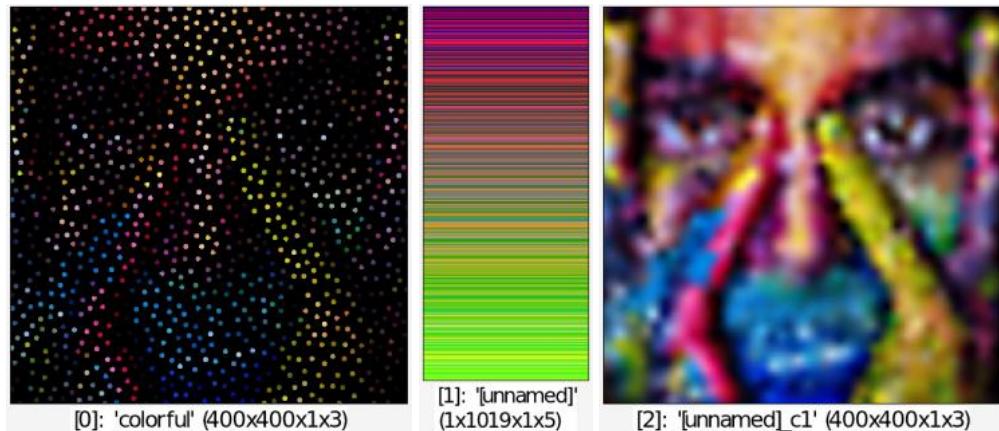
## Default values:

$x0=y0=z0=0$ ,  $x1=dx-1$ ,  $y1=dy-1$ ,  $z1=dz-1$ ,  $\phi(r)=r^2 \log(1e-5+r)$ .

## Examples of use:

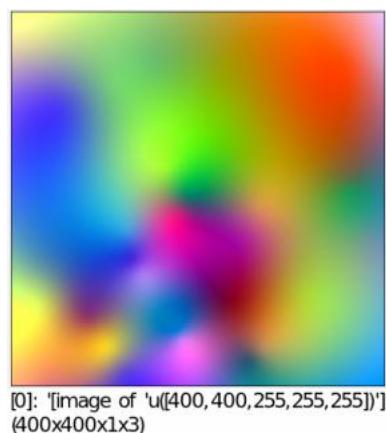
- Example #1

```
$ gmic sp colorful r2dx 400 100%,100% noise_poissondisk. 10 1,
{is},1,5 eval[-2] "begin(p=0);i?(I[#-1,p++]=[x,y,I(#0)])" to_rgb[1]
mul[0,1] dilate_circ[0] 5 +rbf[-1] {0,[w,h]} c[-1] 0,255
```



- Example #2

```
$ gmic 32,1,1,5,u([400,400,255,255,255]) rbf 400,400 c 0,255
```



## Arguments:

- `x0[%],y0[%],x1[%],y1[%],_opacity,_pattern,_color1,...`

## Description:

Draw specified colored rectangle on selected images.

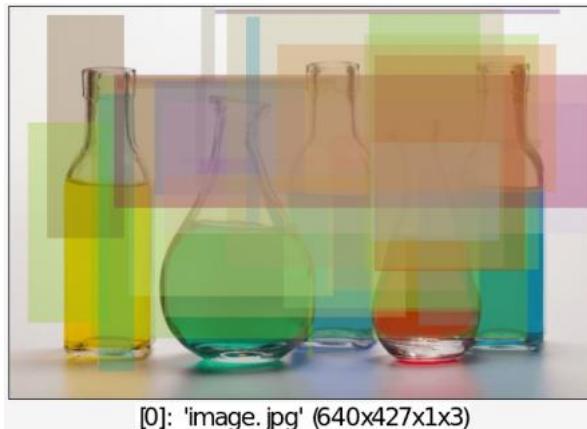
`pattern` is an hexadecimal number starting with `0x` which can be omitted even if a color is specified. If a pattern is specified, the rectangle is drawn outlined instead of filled.

## Default values:

`opacity=1`, `pattern=(undefined)` and `color1=0`.

## Example of use:

```
$ gmic image.jpg repeat 30 rectangle {u(100)}%,{u(100)}%,{u(100)}%,
{u(100)}%,0.3,${-rgb} done
```



---

## red\_eye

## Arguments:

- `0<=_threshold<=100,_smoothness>=0,0<=attenuation<=1`

## Description:

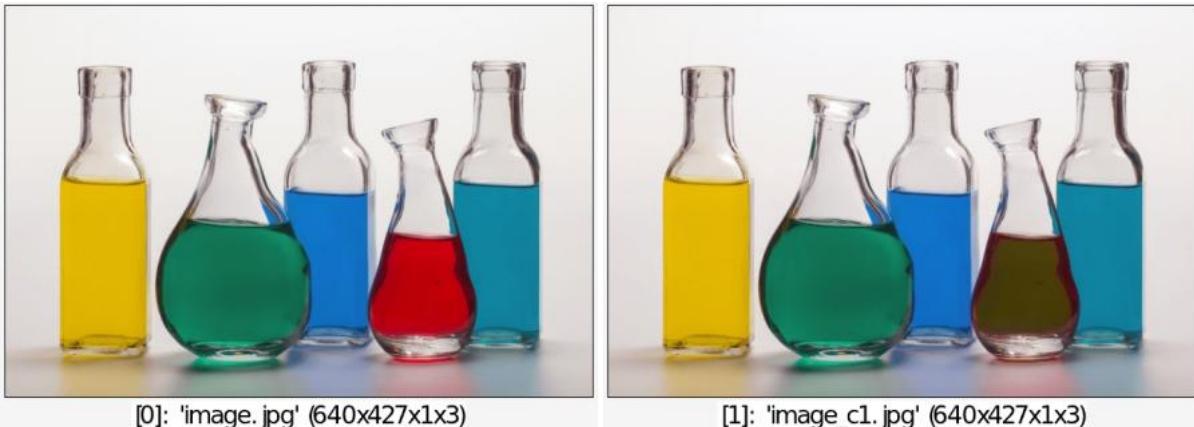
Attenuate red-eye effect in selected images.

## Default values:

`threshold=75`, `smoothness=3.5` and `attenuation=0.1`.

## Example of use:

```
$ gmic image.jpg +red_eye ,
```



---

## register\_nonrigid

### Arguments:

- `[destination],_smoothness>=0,_precision>0,_nb_scale>=0`

### Description:

Register selected source images with specified destination image, using non-rigid warp.

### Default values:

`smoothness=0.2`, `precision=6` and `nb_scale=0(auto)`.

### Example of use:

```
$ gmic image.jpg +rotate 20,1,1,50%,50% +register_nonrigid[0] [1]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)

---

## register\_rigid

### Arguments:

- `[destination],_smoothness>=0,_boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

### Description:

Register selected source images with specified destination image, using rigid warp (shift).

### Default values:

`smoothness=0.1%` and `boundary_conditions=0`.

### Example of use:

```
$ gmic image.jpg +shift 30,20 +register_rigid[0] [1]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)

---

## remove

Built-in command

No arguments

**Description:**

Remove selected images.

(equivalent to shortcut command `rm`).

**Examples of use:**

- **Example #1**

```
$ gmic image.jpg split x remove[30%-70%] append x
```



[0]: 'image.jpg' (384x427x1x3)

- **Example #2**

```
$ gmic image.jpg split x remove[0-50%:2] append x
```



---

## remove\_copymark

### Arguments:

- "image\_name"

### Description:

Remove copy mark from names of selected images.

---

## remove\_duplicates

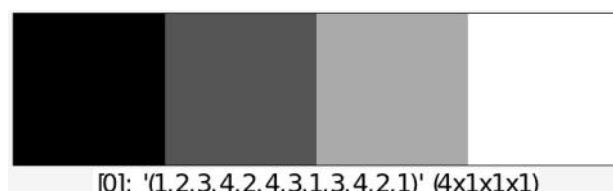
### No arguments

### Description:

Remove duplicates images in the selected images list.

### Example of use:

```
$ gmic (1,2,3,4,2,4,3,1,3,4,2,1) split x remove_duplicates append x
```



## remove\_empty

No arguments

### Description:

Remove empty images in the selected image list.

---

## remove\_hotpixels

### Arguments:

- `_mask_size>0, _threshold[%]>0`

### Description:

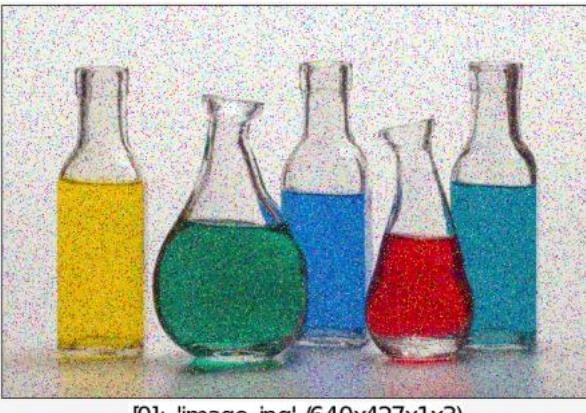
Remove hot pixels in selected images.

### Default values:

`mask_size=3` and `threshold=10%`.

### Example of use:

```
$ gmic image.jpg noise 10,2 +remove_hotpixels ,
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## remove\_named

### Arguments:

- `"name1", "name2", ...`

### Description:

Remove all images with specified names from the list of images.

Does nothing if no images with those names exist.

(equivalent to shortcut command `rmn`).

---

## remove\_opacity

**No arguments**

**Description:**

Remove opacity channel of selected images.

---

## remove\_pixels

**Arguments:**

- `number_of_pixels[%]>=0`

**Description:**

Remove specified number of pixels (i.e. set them to 0) from the set of non-zero pixels in selected images.

**Example of use:**

```
$ gmic image.jpg +remove_pixels 50%
```



---

## repeat

Built-in command

**Arguments:**

- `nb_iterations,_variable_name`

## Description:

Start `nb_iterations` iterations of a `repeat...done` block.

`nb_iterations` is a mathematical expression that will be evaluated.

This command has a [tutorial page](#).

## Examples of use:

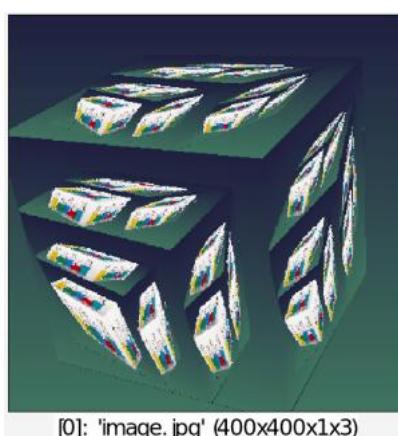
- Example #1

```
$ gmic image.jpg split y repeat $!,n shift[$n] $<,0,0,0,2 done append  
y
```



- Example #2

```
$ gmic image.jpg mode3d 2 repeat 4 imagecube3d rotate3d 1,1,0,40  
snapshot3d 400,1.4 done
```



---

replace

Arguments:

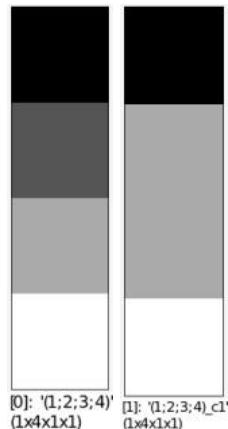
- source, target

## Description:

Replace pixel values in selected images.

## **Example of use:**

```
$ gmic (1;2;3;4) +replace 2,3
```



## replace\_color

## Arguments:

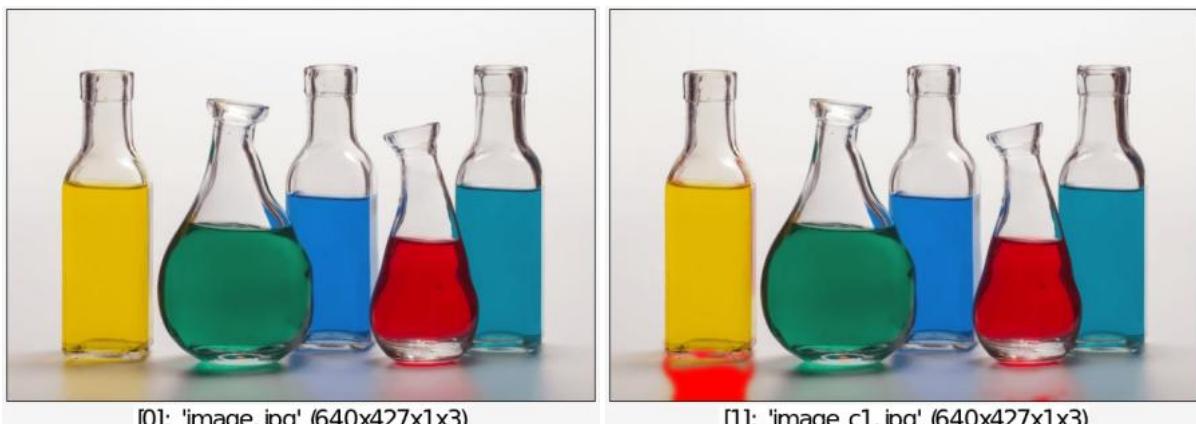
- `tolerance[%]>=0, smoothness[%]>=0, src1, src2, ..., dest1, dest2, ...`

## Description:

Replace pixels from/to specified colors in selected images.

## **Example of use:**

```
$ gmic image.jpg +replace_color 40,3,204,153,110,255,0,0
```



# replace\_inf

## Arguments:

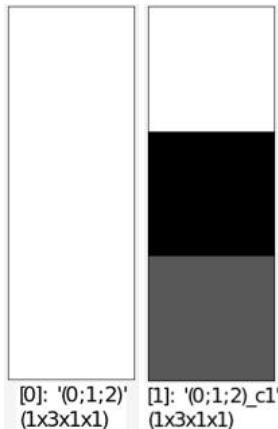
- `_expression`

## Description:

Replace all infinite values in selected images by specified expression.

## Example of use:

```
$ gmic (0;1;2) log +replace_inf 2
```



---

# replace\_nan

## Arguments:

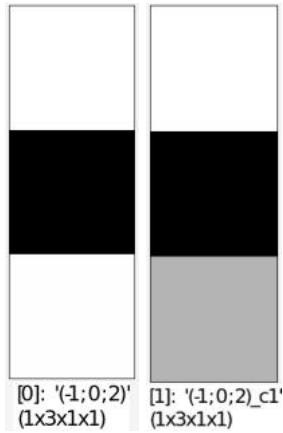
- `_expression`

## Description:

Replace all NaN values in selected images by specified expression.

## Example of use:

```
$ gmic (-1;0;2) sqrt +replace_nan 2
```



---

## replace\_naninf

### Arguments:

- `_expression`

### Description:

Replace all NaN and infinite values in selected images by specified expression.

---

## replace\_seq

### Arguments:

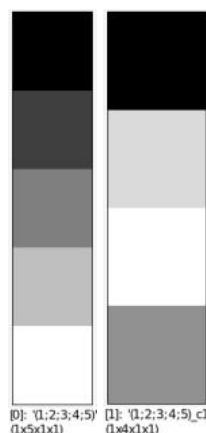
- `"search_seq", "replace_seq"`

### Description:

Search and replace a sequence of values in selected images.

### Example of use:

```
$ gmic (1;2;3;4;5) +replace_seq "2,3,4","7,8"
```



---

# replace\_str

## Arguments:

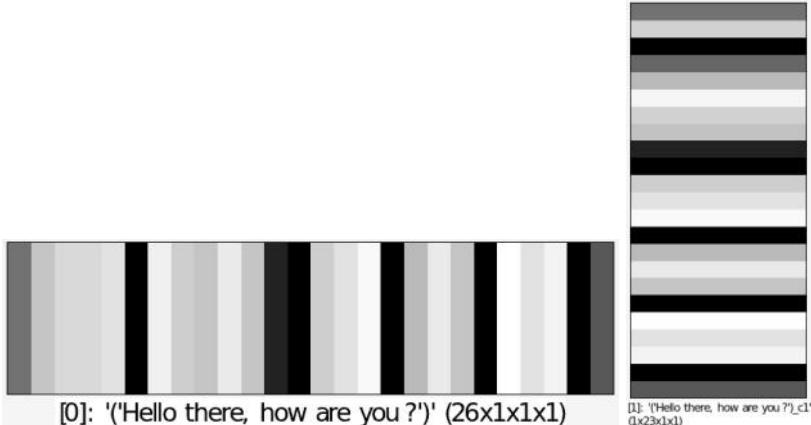
- `"search_str","replace_str"`

## Description:

Search and replace a string in selected images (viewed as strings, i.e. sequences of character codes).

## Example of use:

```
$ gmic ('"Hello there, how are you ?"') +replace_str "Hello  
there","Hi David"
```



---

# reset

## No arguments

## Description:

Reset global parameters of the interpreter environment.

---

# resize

Built-in command

## Arguments:

- `[[image_w] | width>0[%]],_{{[image_h] | height>0[%]}},_{{[image_d] | depth>0[%]}},_{{[image_s] | spectrum>0[%]}},_{{[interpolation]}},_{{[boundary_conditions]}},_{{[ax]}},_{{[ay]}},_{{[az]}},_{{[ac]}}`

## Description:

Resize selected images with specified geometry.

(equivalent to shortcut command `r`).

`interpolation` can be `{ -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }`.

`boundary_conditions` has different meanings, according to the chosen `interpolation` mode :

. When 'interpolation=={ -1 | 1 | 2 | 4 }', `boundary_conditions` is meaningless.

. When `interpolation==0`, `boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

. When 'interpolation=={ 3 | 5 | 6 }', `boundary_conditions` can be `{ 0=none | 1=neumann }`.

`ax,ay,az,ac` set the centering along each axis when 'interpolation=0 or 4' (set to `0` by default, must be defined in range [0,1]).

## Default values:

`interpolation=1`, `boundary_conditions=0` and `ax=ay=az=ac=0`.

## Example of use:

```
$ gmic image.jpg +resize[-1] 256,128,1,3,2 +resize[-1]
120%,120%,1,3,0,1,0.5,0.5 +resize[-1] 120%,120%,1,3,0,0,0.2,0.2
+resize[-1] [0],[0],1,3,4
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (256x128x1x3)



[2]: 'image\_c2.jpg' (307x154x1x3)



[3]: 'image\_c3.jpg' (368x185x1x3)



[4]: 'image\_c4.jpg' (640x427x1x3)

---

## resize2din

### Arguments:

- `width[%]>0,_height[%]>0,_interpolation,_boundary_conditions,_ax,_ay,_az,_ac`

### Description:

Resize selected images so the size is not larger than `width 'x' height` while preserving 2D ratio.

(equivalent to shortcut command `(unknown)`).

`interpolation` can be `{ -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }`.

`boundary_conditions` has different meanings, according to the chosen `interpolation` mode :

. When 'interpolation=={ -1 | 1 | 2 | 4 }', `boundary_conditions` is meaningless.

. When `interpolation==0`, `boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

. When 'interpolation=={ 3 | 5 | 6 }', `boundary_conditions` can be `{ 0=none | 1=neumann }`.

`ax,ay,az,ac` set the centering along each axis when `interpolation=0` (set to `0` by default, must be defined in range [0,1]).

### Default values:

`height=100%`, `interpolation=3`, `boundary_conditions=0` and `ax=ay=az=ac=0`.

### Example of use:

```
$ gmic image.jpg +resize2din 100,100 append x
```



---

## resize2dout

### Arguments:

- `width[%]>0, _height[%]>0, _interpolation, _boundary_conditions, _ax, _ay, _az, _ac`

### Description:

Resize selected images so the size is not smaller than `width'x'height` while preserving 2D ratio.

(*equivalent to shortcut command* `(unknown)`).

`interpolation` can be `{ -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }`.

`boundary_conditions` has different meanings, according to the chosen `interpolation` mode :

. When 'interpolation=={ -1 | 1 | 2 | 4 }', `boundary_conditions` is meaningless.

. When `interpolation==0`, `boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

. When 'interpolation=={ 3 | 5 | 6 }', `boundary_conditions` can be `{ 0=none | 1=neumann }`.  
`ax,ay,az,ac` set the centering along each axis when `interpolation=0` (set to `0` by default, must be defined in range [0,1]).

### Default values:

`height=100%`, `interpolation=3`, `boundary_conditions=0` and `ax=ay=az=ac=0`.

### Example of use:

```
$ gmic image.jpg +resize2dout 100,100 append x
```



---

## resize2dx

### Arguments:

- `width[%]>0, _interpolation, _boundary_conditions, _ax, _ay, _az, _ac`

### Description:

Resize selected images along the x-axis, while preserving 2D ratio.

(*equivalent to shortcut command `r2dx`*).

`interpolation` can be `{ -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }`.

`boundary_conditions` has different meanings, according to the chosen `interpolation` mode :

- . When 'interpolation=={-1 | 1 | 2 | 4}', `boundary_conditions` is meaningless.
  - . When `interpolation==0`, `boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.
  - . When 'interpolation=={3 | 5 | 6}', `boundary_conditions` can be `{ 0=none | 1=neumann }`.
- `ax,ay,az,ac` set the centering along each axis when `interpolation=0` (set to `0` by default, must be defined in range [0,1]).

### Default values:

`interpolation=3`, `boundary_conditions=0` and `ax=ay=az=ac=0`.

### Example of use:

```
$ gmic image.jpg +resize2dx 100,2 append x
```



[0]: 'image.jpg' (740x427x1x3)

---

## resize2dy

### Arguments:

- `height[%]>=0, _interpolation, _boundary_conditions, _ax, _ay, _az, _ac`

### Description:

Resize selected images along the y-axis, while preserving 2D ratio.

(equivalent to shortcut command `r2dy`).

`interpolation` can be `{ -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }`.

`boundary_conditions` has different meanings, according to the chosen `interpolation` mode :

. When 'interpolation=={ -1 | 1 | 2 | 4 }', `boundary_conditions` is meaningless.

. When `interpolation==0`, `boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

. When 'interpolation=={ 3 | 5 | 6 }', `boundary_conditions` can be `{ 0=none | 1=neumann }`.

`ax,ay,az,ac` set the centering along each axis when `interpolation=0` (set to `0` by default, must be defined in range [0,1]).

### Default values:

`interpolation=3, boundary_conditions=0` and `ax=ay=az=ac=0`.

### Example of use:

```
$ gmic image.jpg +resize2dy 100,2 append x
```



## resize3din

### Arguments:

- `width[%]>0,_height[%]>0,_depth[%]>0,_interpolation,_boundary_conditions,_ax,`

### Description:

Resize selected images so the size is not larger than `width'x'height'x'depth` while preserving 3D ratio.

(equivalent to shortcut command `(unknown)`).

`interpolation` can be `{ -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }`.

`boundary_conditions` has different meanings, according to the chosen `interpolation` mode :

. When 'interpolation=={-1 | 1 | 2 | 4}', `boundary_conditions` is meaningless.

. When `interpolation==0`, `boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

. When 'interpolation=={3 | 5 | 6}', `boundary_conditions` can be `{ 0=none | 1=neumann }`.

`ax,ay,az,ac` set the centering along each axis when `interpolation=0` (set to `0` by default, must be defined in range [0,1]).

### Default values:

`height=100%, depth=100%, interpolation=3, boundary_conditions=0` and `ax=ay=az=ac=0`.

---

## resize3dout

### Arguments:

- `width[%]>0,_height[%]>0,_depth[%]>0,_interpolation,_boundary_conditions,_ax,`

### Description:

Resize selected images so the size is not smaller than `width'x'height'x'depth` while preserving 3D ratio.

(equivalent to shortcut command `(unknown)`).

`interpolation` can be `{ -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }`.

`boundary_conditions` has different meanings, according to the chosen `interpolation` mode

:

. When 'interpolation=={-1 | 1 | 2 | 4}', `boundary_conditions` is meaningless.

. When `interpolation==0`, `boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

. When 'interpolation=={3 | 5 | 6}', `boundary_conditions` can be `{ 0=none | 1=neumann }`.

`ax,ay,az,ac` set the centering along each axis when `interpolation=0`

(set to `0` by default, must be defined in range [0,1]).

## Default values:

`height=100%`, `depth=100%`, `interpolation=3`, `boundary_conditions=0` and `ax=ay=az=ac=0`.

---

# resize3dx

## Arguments:

- `width[%]>0, _interpolation, _boundary_conditions, _ax, _ay, _az, _ac`

## Description:

Resize selected images along the x-axis, while preserving 3D ratio.

(equivalent to shortcut command `r3dx`).

`interpolation` can be `{ -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }`.

`boundary_conditions` has different meanings, according to the chosen `interpolation` mode

:

. When 'interpolation=={-1 | 1 | 2 | 4}', `boundary_conditions` is meaningless.

. When `interpolation==0`, `boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

. When 'interpolation=={3 | 5 | 6}', `boundary_conditions` can be `{ 0=none | 1=neumann }`.

`ax,ay,az,ac` set the centering along each axis when `interpolation=0`

(set to `0` by default, must be defined in range [0,1]).

## Default values:

`interpolation=3`, `boundary_conditions=0` and `ax=ay=az=ac=0`.

---

# resize3dy

## Arguments:

- `height[%]>0, _interpolation, _boundary_conditions, _ax, _ay, _az, _ac`

## Description:

Resize selected images along the y-axis, while preserving 3D ratio.

(equivalent to shortcut command `r3dy`).

`interpolation` can be `{ -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }`.

`boundary_conditions` has different meanings, according to the chosen `interpolation` mode :

- . When 'interpolation=={ -1 | 1 | 2 | 4 }', `boundary_conditions` is meaningless.
  - . When `interpolation==0`, `boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.
  - . When 'interpolation=={ 3 | 5 | 6 }', `boundary_conditions` can be `{ 0=none | 1=neumann }`.
- `ax,ay,az,ac` set the centering along each axis when `interpolation=0` (set to `0` by default, must be defined in range [0,1]).

## Default values:

`interpolation=3, boundary_conditions=0` and `ax=ay=az=ac=0`.

---

# resize3dz

## Arguments:

- `depth[%]>0, _interpolation, _boundary_conditions, _ax, _ay, _az, _ac`

## Description:

Resize selected images along the z-axis, while preserving 3D ratio.

(equivalent to shortcut command `r3dz`).

`interpolation` can be `{ -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }`.

`boundary_conditions` has different meanings, according to the chosen `interpolation` mode :

- . When 'interpolation=={ -1 | 1 | 2 | 4 }', `boundary_conditions` is meaningless.
- . When `interpolation==0`, `boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

`2=periodic | 3=mirror }.`

. When 'interpolation=={ 3 | 5 | 6 }', `boundary_conditions` can be `{ 0=none | 1=neumann }`.  
`ax,ay,az,ac` set the centering along each axis when `interpolation=0` (set to `0` by default, must be defined in range [0,1]).

## Default values:

`interpolation=3`, `boundary_conditions=0` and `ax=ay=az=ac=0`.

---

# resize\_as\_image

## Arguments:

- `[reference],_interpolation,_boundary_conditions,_ax,_ay,_az,_ac`

## Description:

Resize selected images to the geometry of specified [reference] image.

(equivalent to shortcut command `ri`).

## Default values:

`interpolation=1`, `boundary_conditions=0` and `ax=ay=az=ac=0`.

## Example of use:

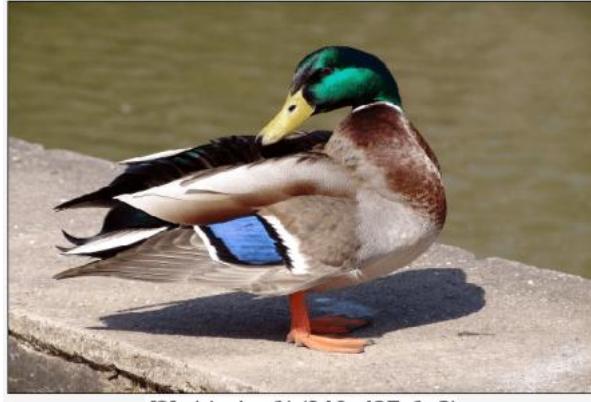
```
$ gmic image.jpg sample duck +resize_as_image[-1] [-2]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'duck' (640x480x1x3)



[2]: 'duck\_c1' (640x427x1x3)

## resize\_mn

### Arguments:

- `width[%]>=0, _height[%]>=0, _depth[%]>=0, _B_value, _C_value`

### Description:

Resize selected images with Mitchell-Netravali filter (cubic).

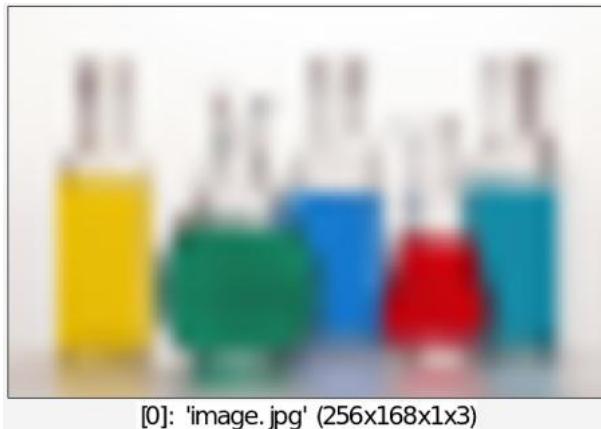
For details about the method, see: <https://de.wikipedia.org/wiki/Mitchell-Netravali-Filter>.

### Default values:

`height=100%`, `depth=100%`, `B=0.3333` and `C=0.3333`.

### Example of use:

```
$ gmic image.jpg resize2dx 32 resize_mn 800%,800%
```



---

## resize\_pow2

### Arguments:

- `_interpolation, _boundary_conditions, _ax, _ay, _az, _ac`

### Description:

Resize selected images so that each dimension is a power of 2.

`interpolation` can be `{ -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }`.

`boundary_conditions` has different meanings, according to the chosen `interpolation` mode :

. When 'interpolation=={ -1 | 1 | 2 | 4 }', `boundary_conditions` is meaningless.

. When `interpolation==0`, `boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

. When 'interpolation=={ 3 | 5 | 6 }', `boundary_conditions` can be `{ 0=none | 1=neumann }`.

`ax, ay, az, ac` set the centering along each axis when `interpolation=0` (set to `0` by default, must be defined in range [0,1]).

### Default values:

`interpolation=0`, `boundary_conditions=0` and `ax=ay=az=ac=0`.

### Example of use:

```
$ gmic image.jpg +resize_pow2[-1] 0
```



---

## resize\_ratio2d

### Arguments:

- `width>0, height>0, mode={ 0=inside | 1=outside | 2=padded }, 0=<_interpolation<=6`

### Description:

Resize selected images while preserving their aspect ratio.

(equivalent to shortcut command `rr2d`).

### Default values:

`mode=0` and `interpolation=6`.

---

## retinex

### Arguments:

- `_value_offset>0, _colorspace={ hsi | hsv | lab | lrgb | rgb | ycbcr }, 0=<_min_cut<=100, 0=<_max_cut<=100, _sigma_low>0, _sigma_mid>0, _sigma_high>0`

### Description:

Apply multi-scale retinex algorithm on selected images to improve color consistency.

(as described in the page <http://www.ipol.im/pub/art/2014/107/>).

### Default values:

`offset=1, colorspace=hsv, min_cut=1, max_cut=1, sigma_low=15, sigma_mid=80` and `sigma_high=250`.

# return

Built-in command

No arguments

## Description:

Return from current custom command.

# reverse

Built-in command

No arguments

## Description:

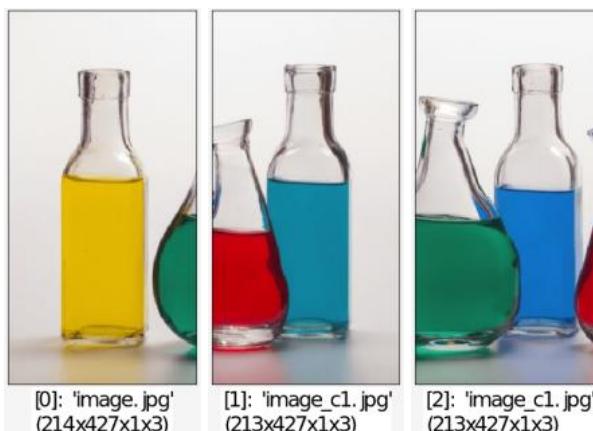
Reverse positions of selected images.

(equivalent to shortcut command `rv`).

## Examples of use:

- Example #1

```
$ gmic image.jpg split x,3 reverse[-2,-1]
```



- Example #2

```
$ gmic image.jpg split x,-16 reverse[50%-100%] append x
```



## reverse3d

Built-in command

No arguments

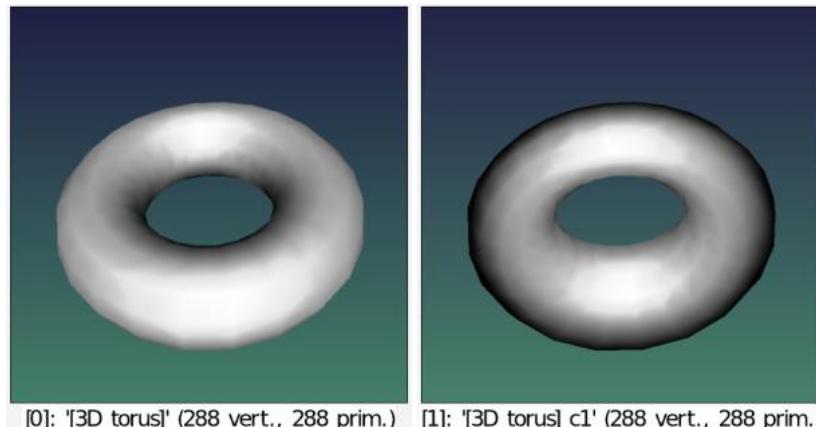
**Description:**

Reverse primitive orientations of selected 3D objects.

(equivalent to shortcut command `rv3d`).

**Example of use:**

```
$ gmic torus3d 100,40 double3d 0 +reverse3d
```



## rgb

No arguments

**Description:**

Return a random int-valued RGB color.

# rgb2bayer

## Arguments:

- `_start_pattern=0, _color_grid=0`

## Description:

Transform selected color images to RGB-Bayer sampled images.

## Default values:

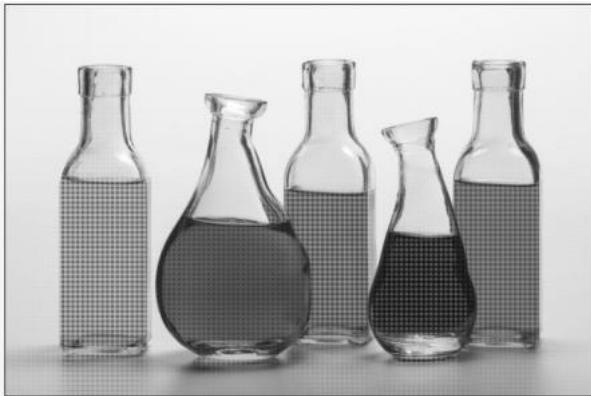
`start_pattern=0` and `color_grid=0`.

## Example of use:

```
$ gmic image.jpg +rgb2bayer 0
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x1)

---

# rgb2cmy

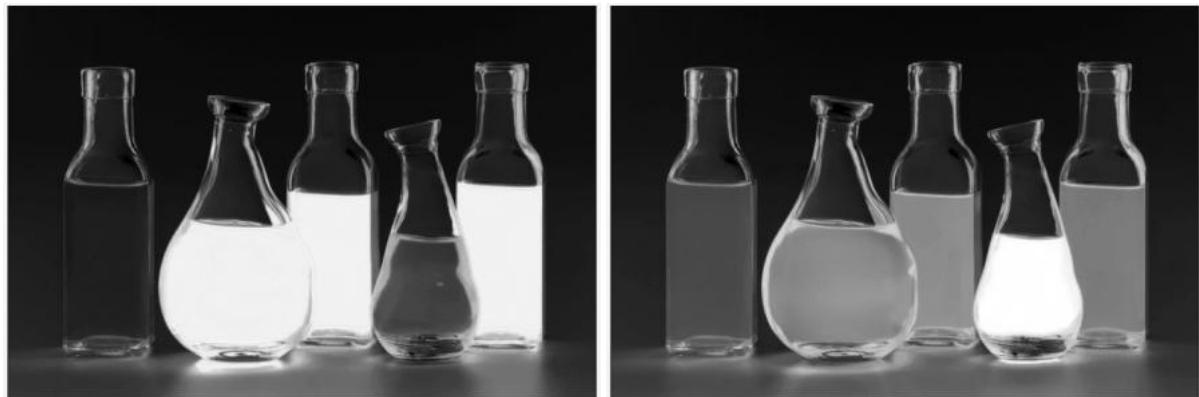
## No arguments

## Description:

Convert color representation of selected images from RGB to CMY.

## Example of use:

```
$ gmic image.jpg rgb2cmy split c
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x1)

---

## rgb2cmyk

**No arguments**

**Description:**

Convert color representation of selected images from RGB to CMYK.

**Examples of use:**

- **Example #1**

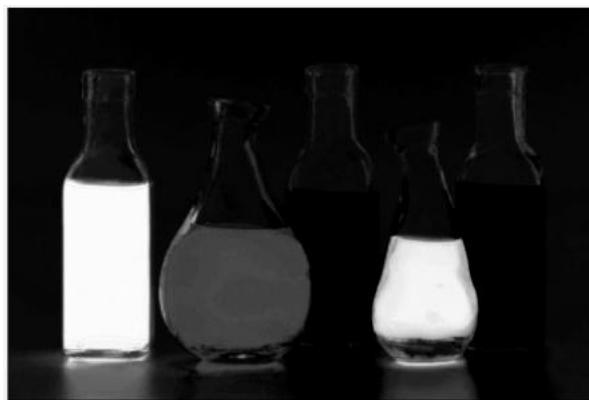
```
$ gmic image.jpg rgb2cmyk split c
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x1)



[3]: 'image\_c1.jpg' (640x427x1x1)

- **Example #2**

```
$ gmic image.jpg rgb2cmyk split c fill[3] 0 append c cmyk2rgb
```



[0]: 'image.jpg' (640x427x1x3)

---

## rgb2hcy

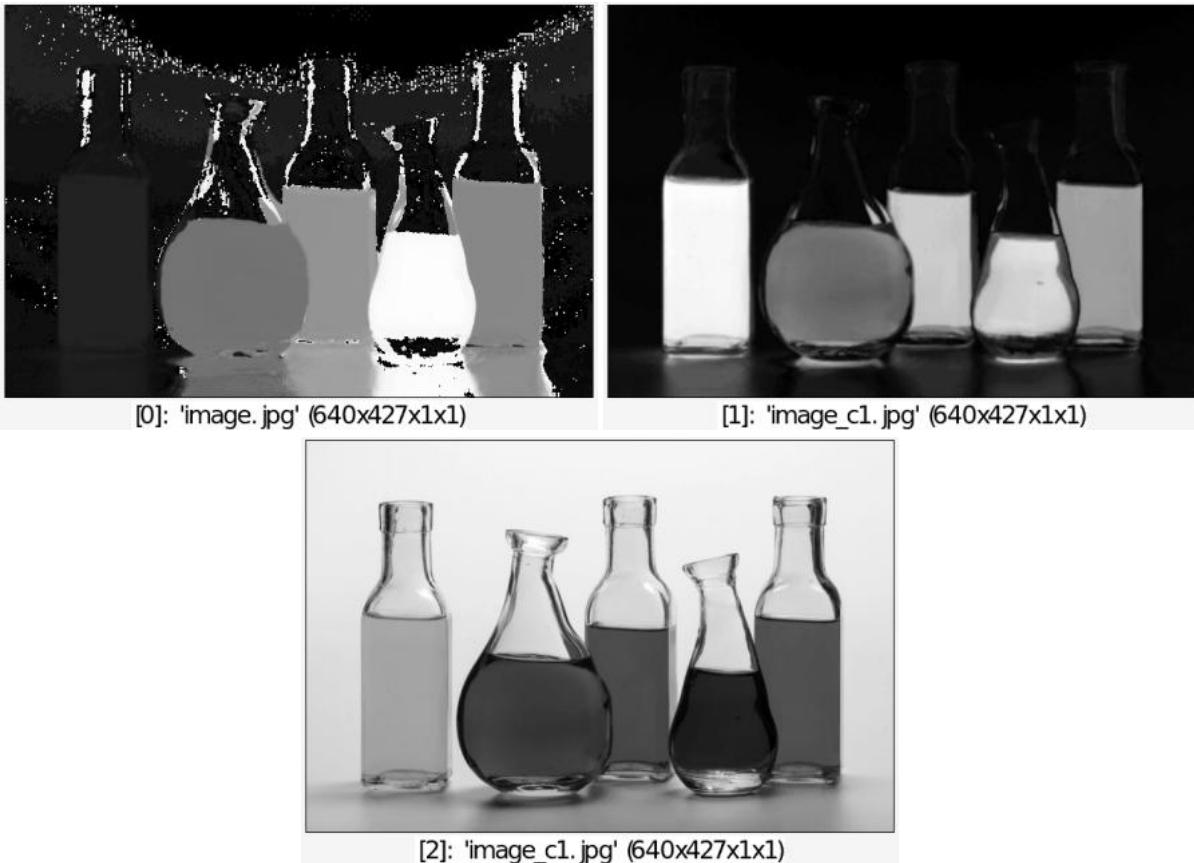
### No arguments

### Description:

Convert color representation of selected images from RGB to HCY.

### Example of use:

```
$ gmic image.jpg rgb2hcy split c
```



---

## rgb2hsd

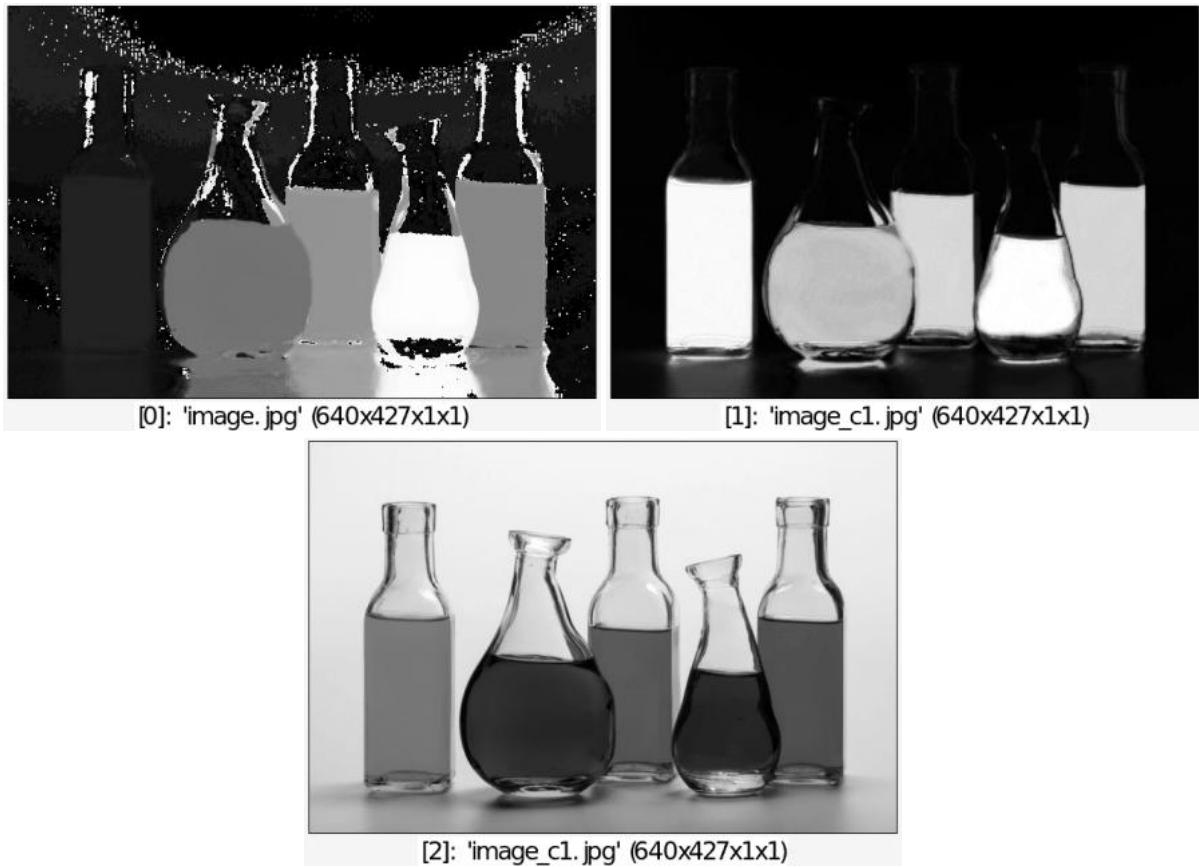
**No arguments**

**Description:**

Convert color representation of selected images from RGB to HSI.

**Example of use:**

```
$ gmic image.jpg rgb2hsd split c
```



---

## rgb2hsi8

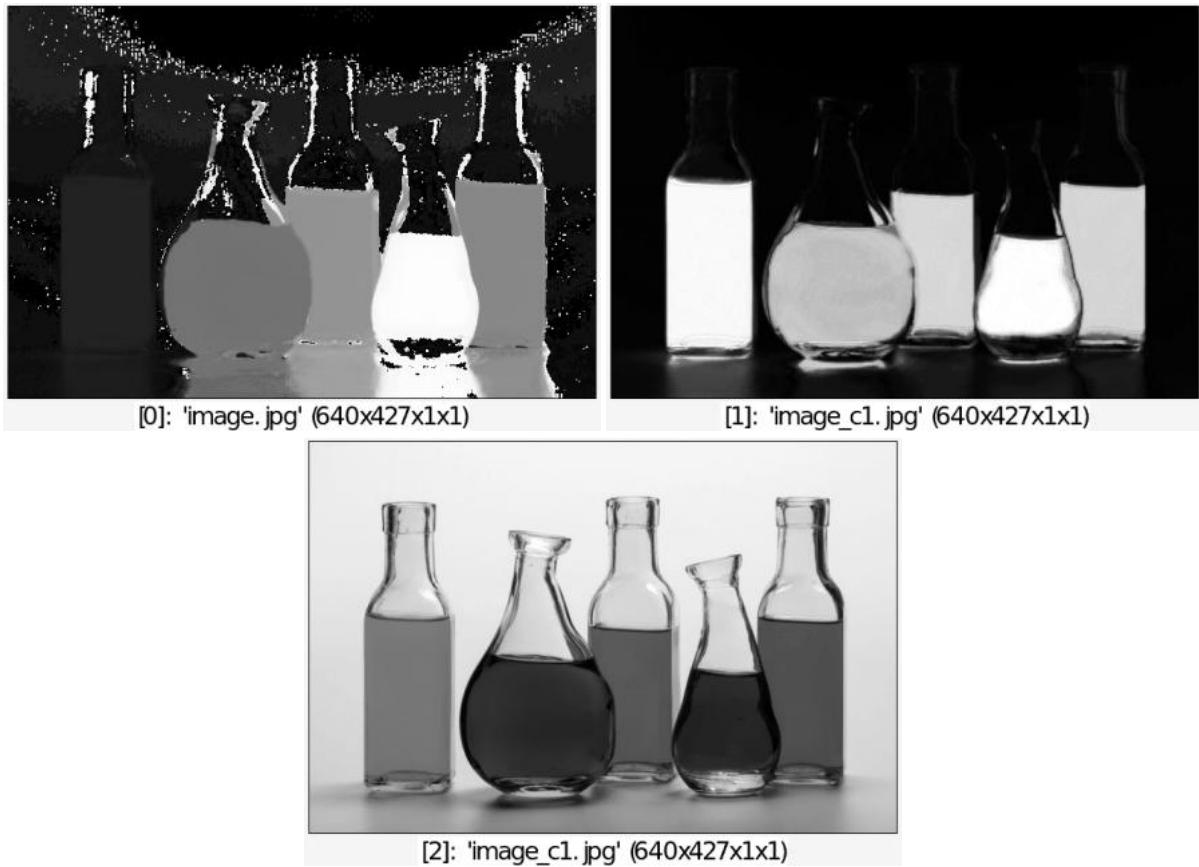
**No arguments**

**Description:**

Convert color representation of selected images from RGB to HSI8.

**Example of use:**

```
$ gmic image.jpg rgb2hsi8 split c
```



---

## rgb2hsl

**No arguments**

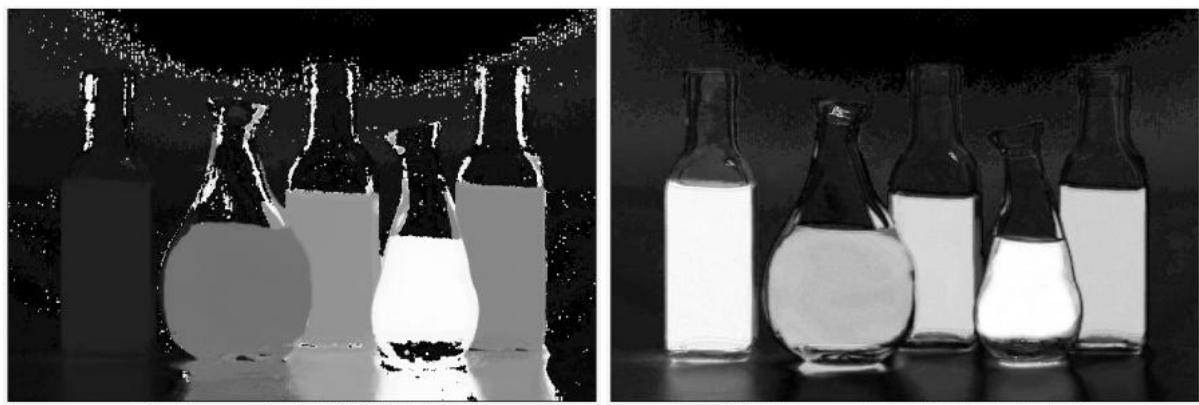
**Description:**

Convert color representation of selected images from RGB to HSL.

**Examples of use:**

- **Example #1**

```
$ gmic image.jpg rgb2hsl split c
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x1)

- **Example #2**

```
$ gmic image.jpg rgb2hsl +split c add[-3] 100 mod[-3] 360 append[-3-  
-1] c hsl2rgb
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## rgb2hsl8

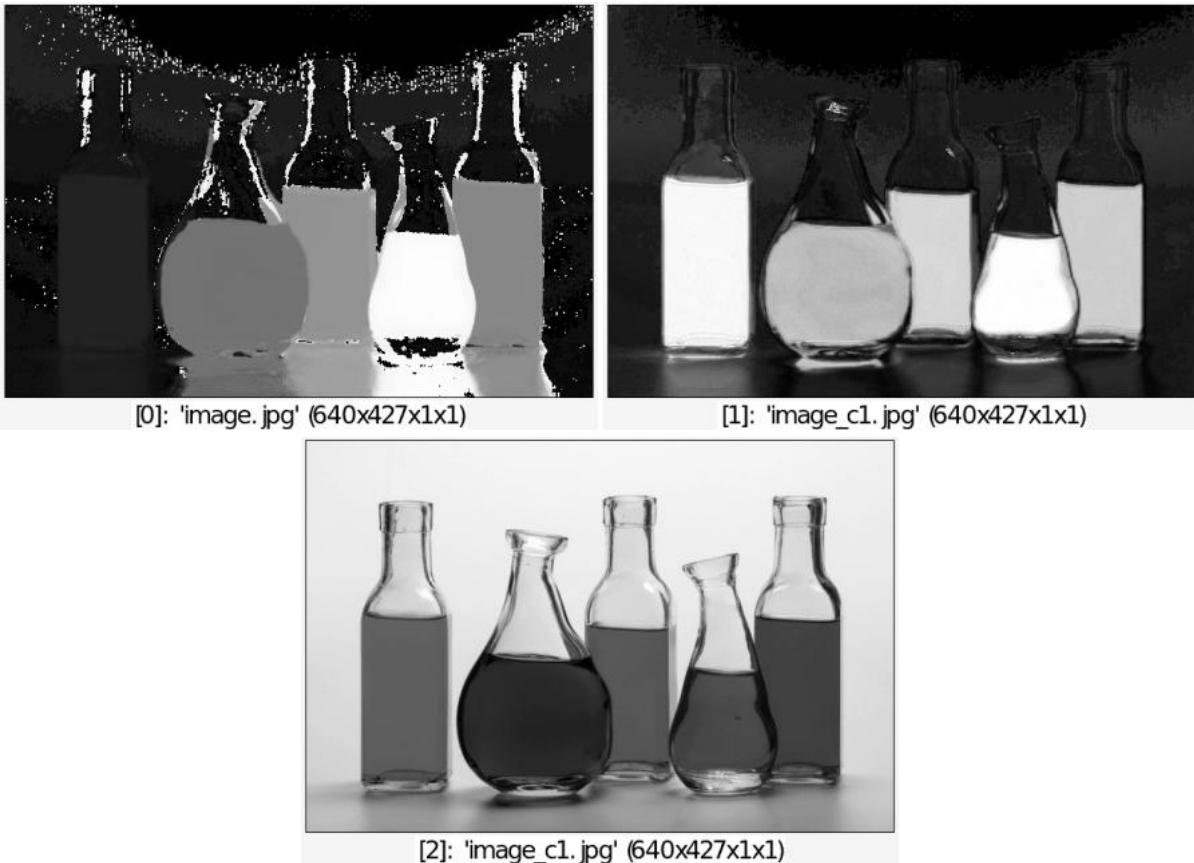
### No arguments

### Description:

Convert color representation of selected images from RGB to HSL8.

### Example of use:

```
$ gmic image.jpg rgb2hsl8 split c
```



---

## rgb2hsv

**No arguments**

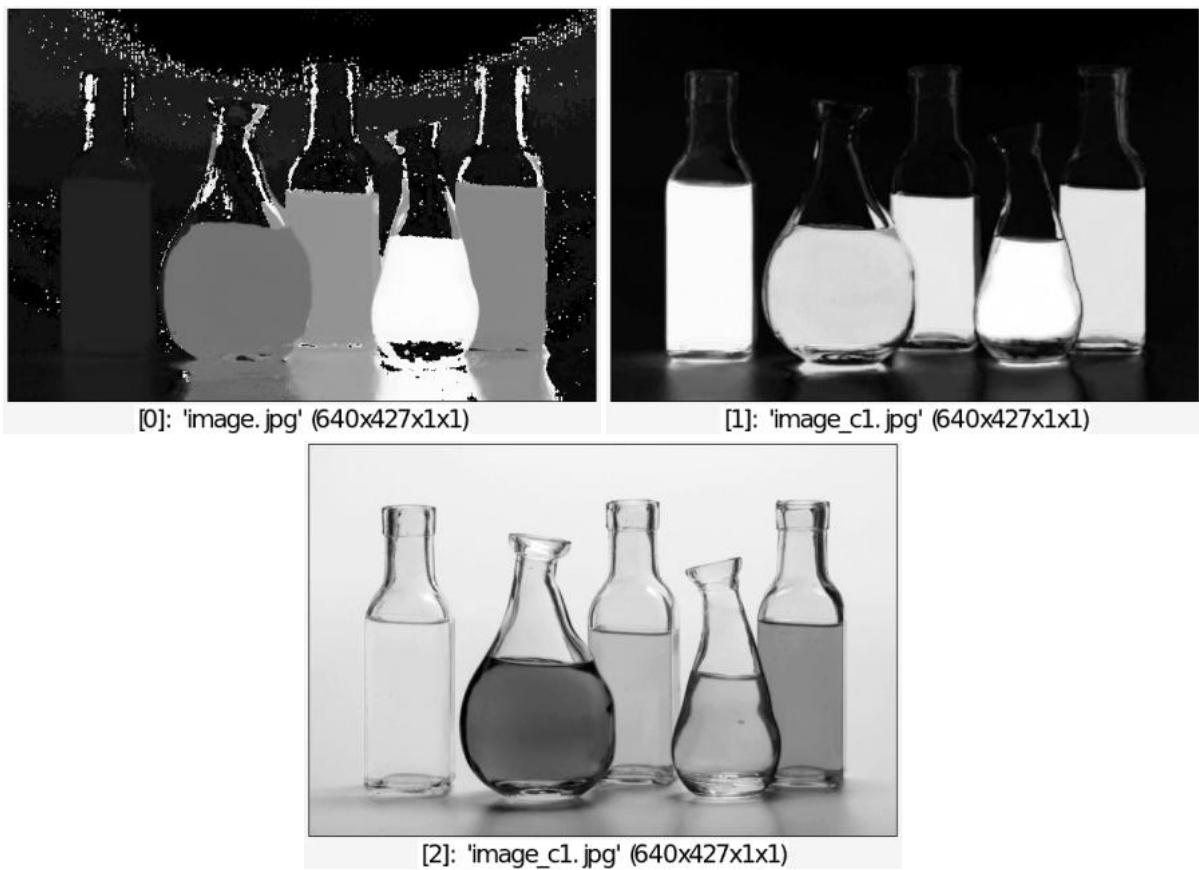
**Description:**

Convert color representation of selected images from RGB to HSV.

**Examples of use:**

- **Example #1**

```
$ gmic image.jpg rgb2hsv split c
```



- **Example #2**

```
$ gmic image.jpg rgb2hsv +split c add[-2] 0.3 cut[-2] 0,1 append[-3-  
-1] c hsv2rgb
```



## rgb2hsv8

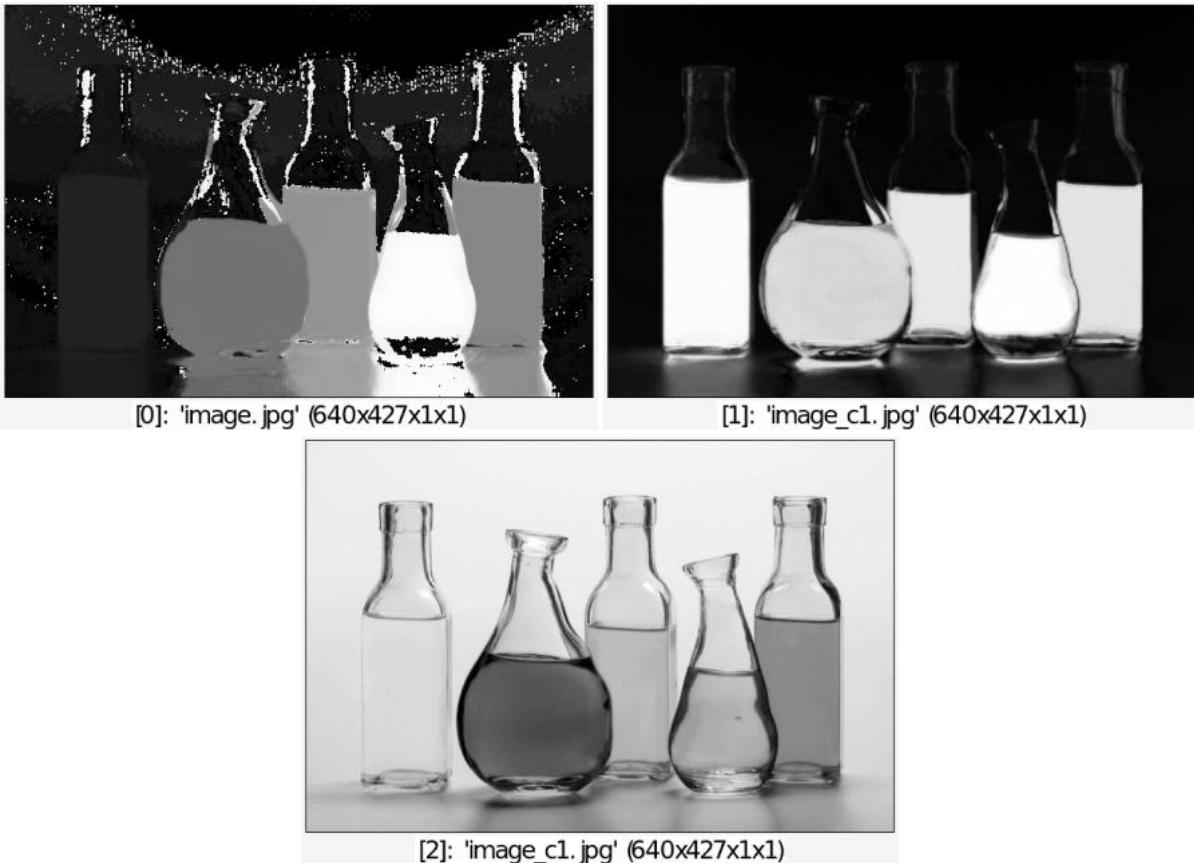
**No arguments**

**Description:**

Convert color representation of selected images from RGB to HSV8.

**Example of use:**

```
$ gmic image.jpg rgb2HSV8 split c
```



---

## rgb2int

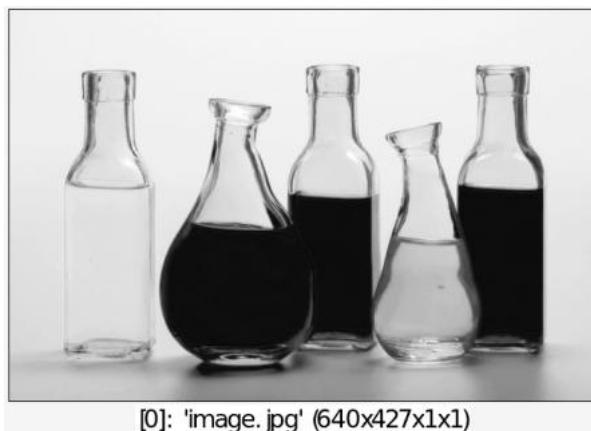
**No arguments**

**Description:**

Convert color representation of selected images from RGB to INT24 scalars.

**Example of use:**

```
$ gmic image.jpg rgb2int
```



---

# rgb2jzazbz

## Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

## Description:

Convert color representation of selected images from RGB to Jzazbz.

## Default values:

`illuminant=2`.

---

# rgb2lab

## Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

## Description:

Convert color representation of selected images from RGB to Lab.

## Default values:

`illuminant=2`.

---

# rgb2lab8

## Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

## Description:

Convert color representation of selected images from RGB to Lab8.

## Default values:

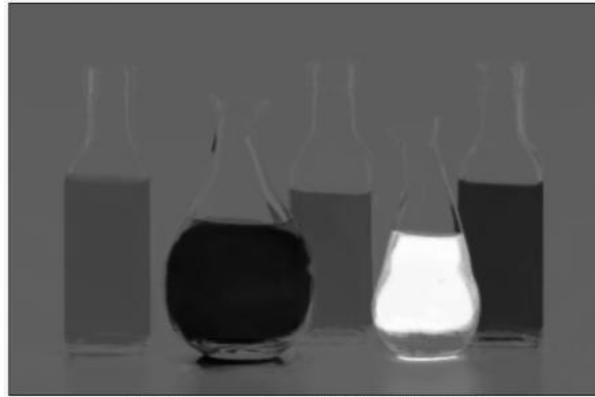
`illuminant=2`.

## Example of use:

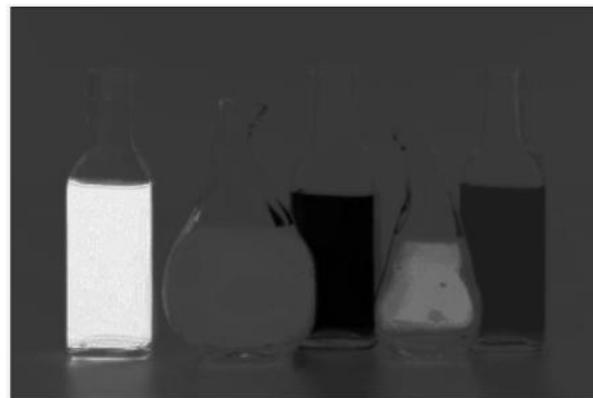
```
$ gmic image.jpg rgb2lab8 split c
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x1)

## rgb2lch

### Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

### Description:

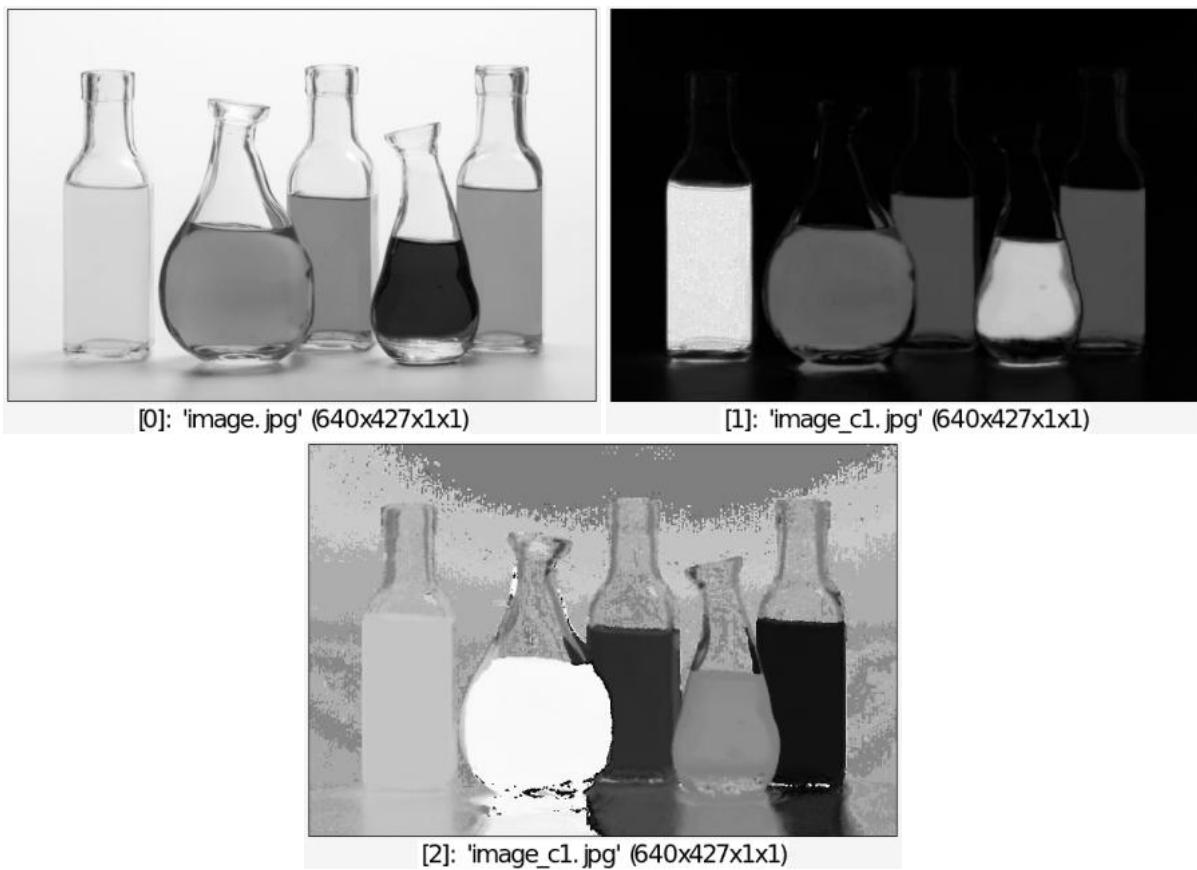
Convert color representation of selected images from RGB to Lch.

### Default values:

`illuminant=2`.

## Example of use:

```
$ gmic image.jpg rgb2lch split c
```



---

## rgb2lch8

### Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

### Description:

Convert color representation of selected images from RGB to Lch8.

### Default values:

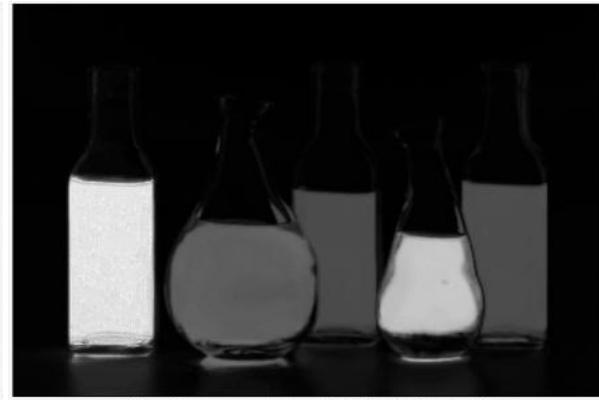
`illuminant=2`.

### Example of use:

```
$ gmic image.jpg rgb2lch8 split c
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x1)

---

## rgb2luv

**No arguments**

**Description:**

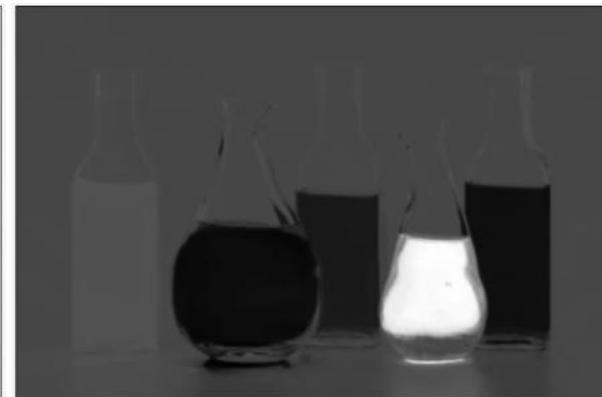
Convert color representation of selected images from RGB to LUV.

**Example of use:**

```
$ gmic image.jpg rgb2luv split c
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x1)

## rgb2oklab

**No arguments**

**Description:**

Convert color representation of selected images from RGB to Oklab.

(see colorspace definition at: <https://bottosson.github.io/posts/oklab/> ).

**See also:**

[oklab2rgb](#).

## rgb2ryb

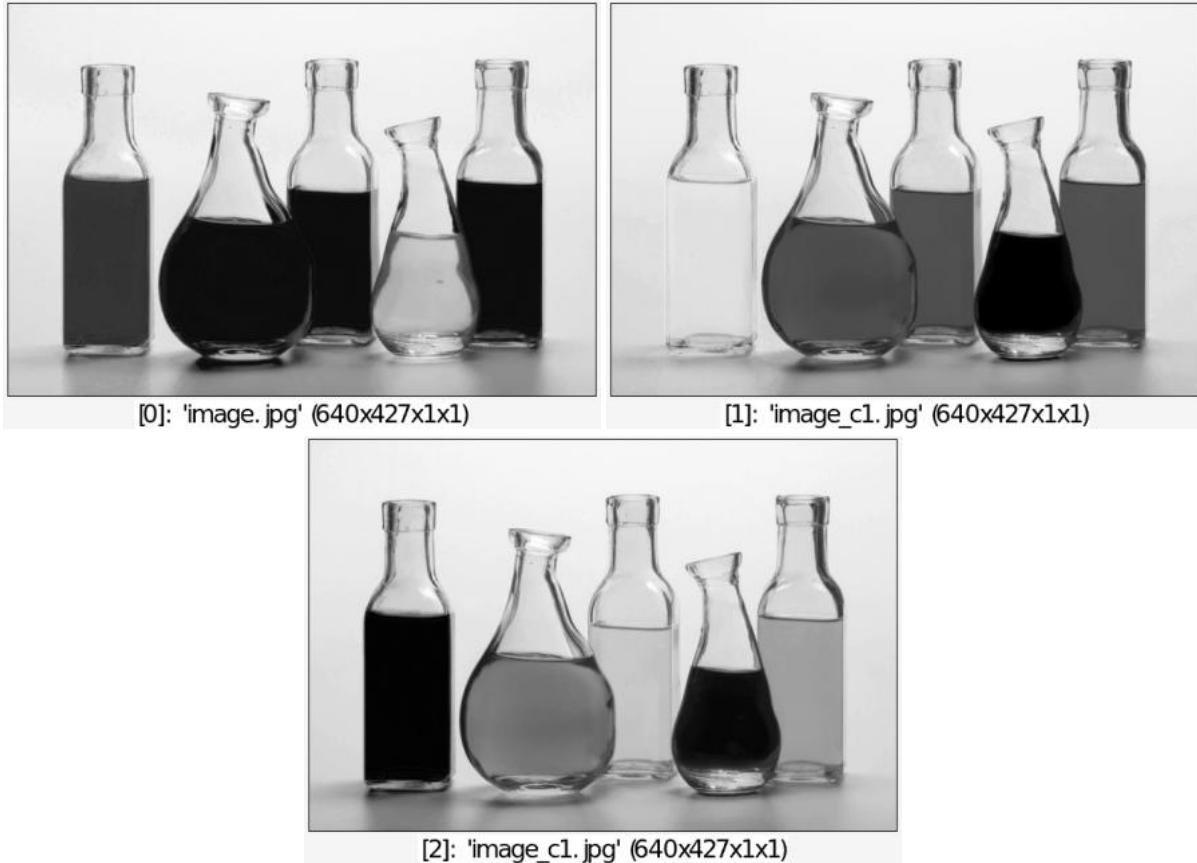
**No arguments**

**Description:**

Convert color representation of selected images from RGB to RYB.

**Example of use:**

```
$ gmic image.jpg rgb2ryb split c
```



## rgb2srgb

**No arguments**

**Description:**

Convert color representation of selected images from linear RGB to sRGB.

## rgb2xyz

**Arguments:**

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

**Description:**

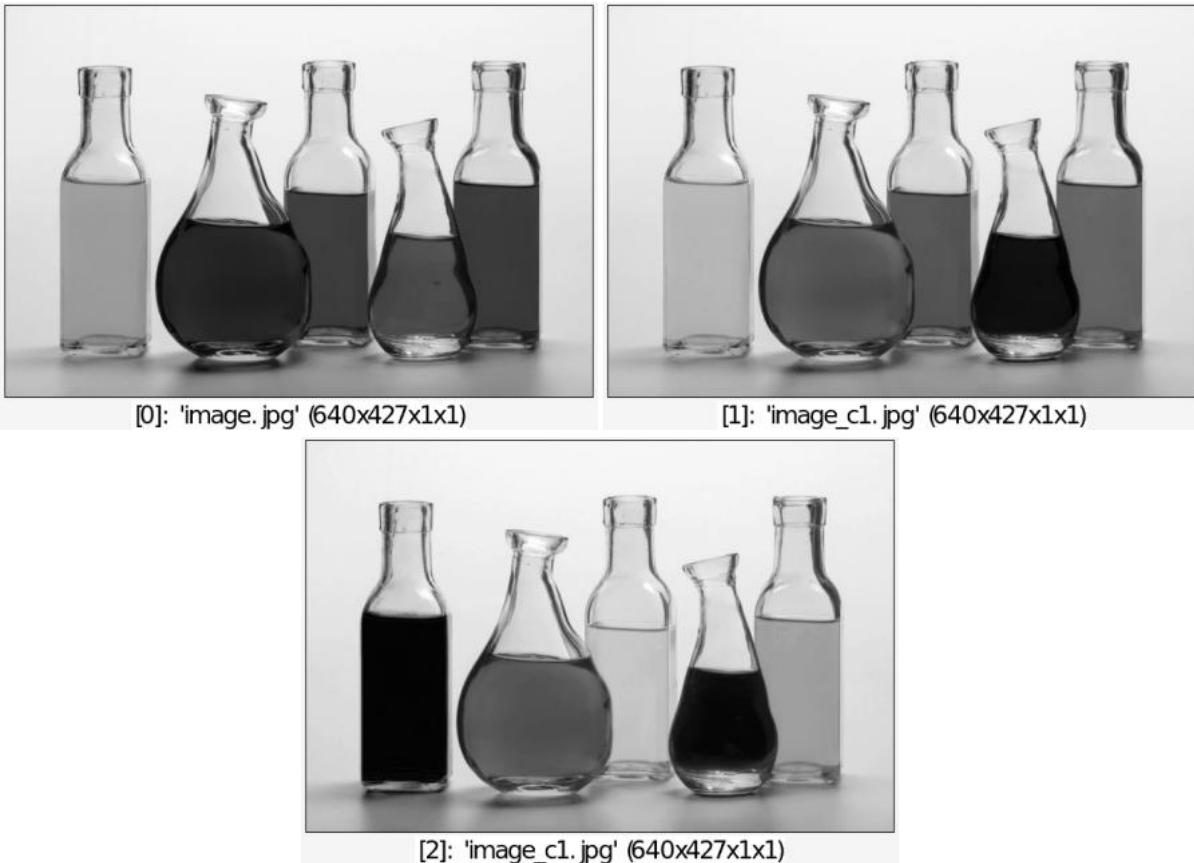
Convert color representation of selected images from RGB to XYZ.

**Default values:**

`illuminant=2`.

**Example of use:**

```
$ gmic image.jpg rgb2xyz split c
```



---

## rgb2xyz8

### Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

### Description:

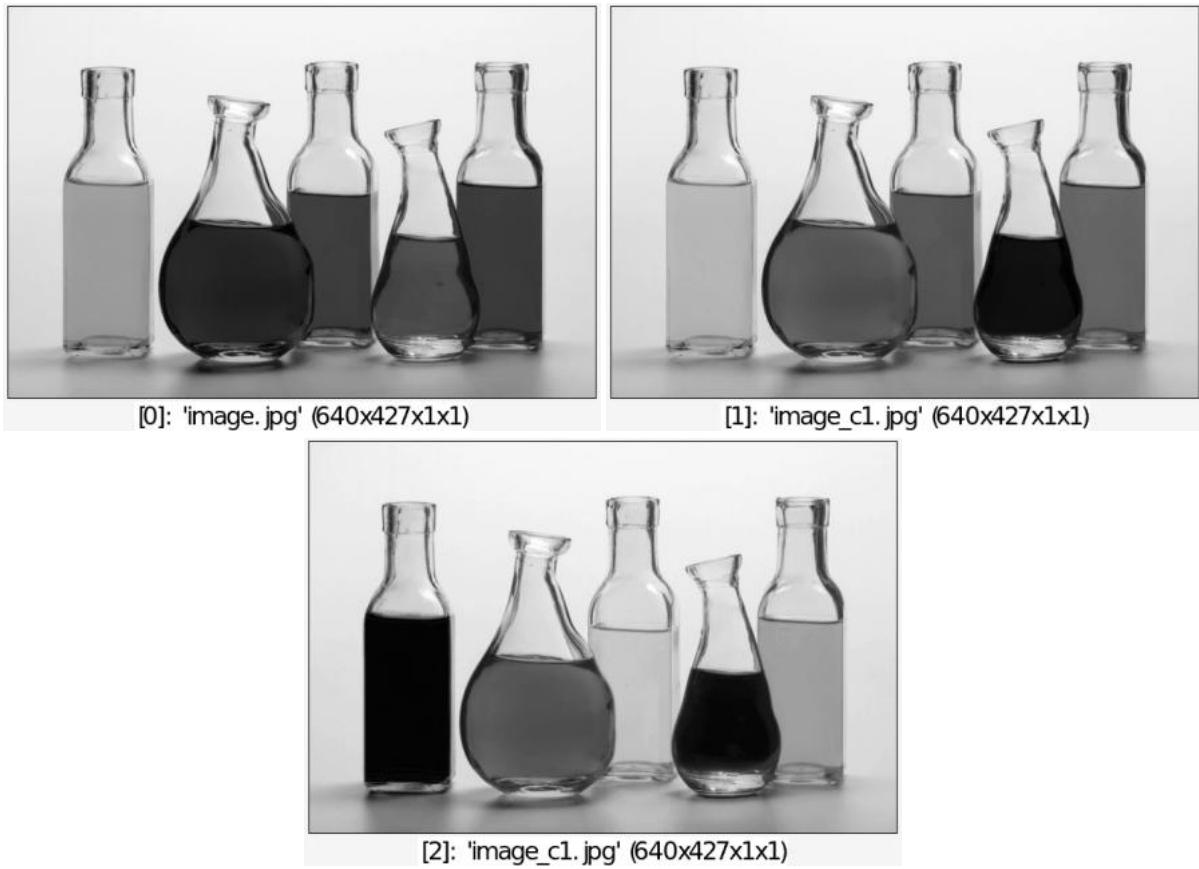
Convert color representation of selected images from RGB to XYZ8.

### Default values:

`illuminant=2`.

### Example of use:

```
$ gmic image.jpg rgb2xyz8 split c
```



---

## rgb2ycbcr

**No arguments**

**Description:**

Convert color representation of selected images from RGB to YCbCr.

**Example of use:**

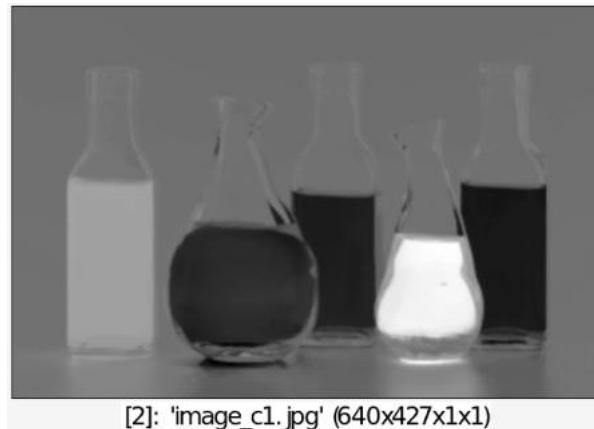
```
$ gmic image.jpg rgb2ycbcr split c
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x1)

---

## rgb2yiq

**No arguments**

**Description:**

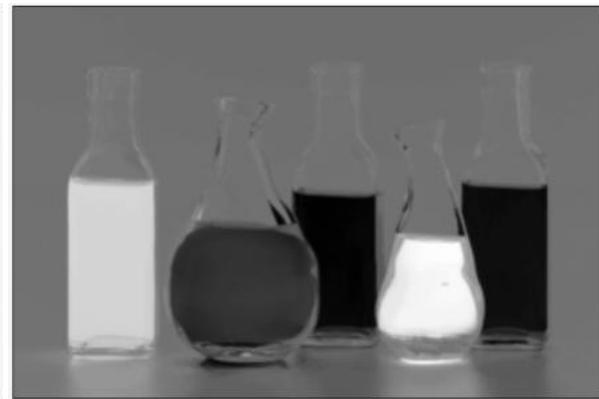
Convert color representation of selected images from RGB to YIQ.

**Example of use:**

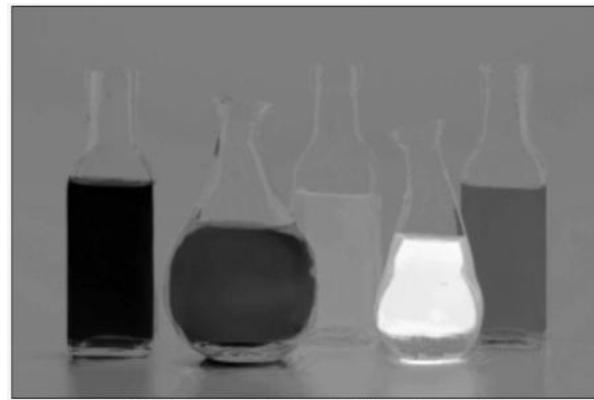
```
$ gmic image.jpg rgb2yiq split c
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x1)

---

## rgb2yiq8

**No arguments**

**Description:**

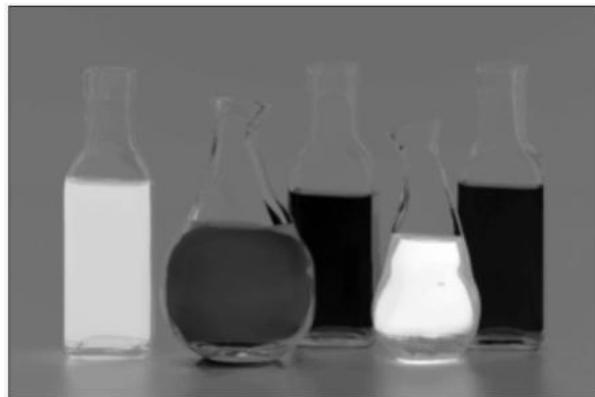
Convert color representation of selected images from RGB to YIQ8.

**Example of use:**

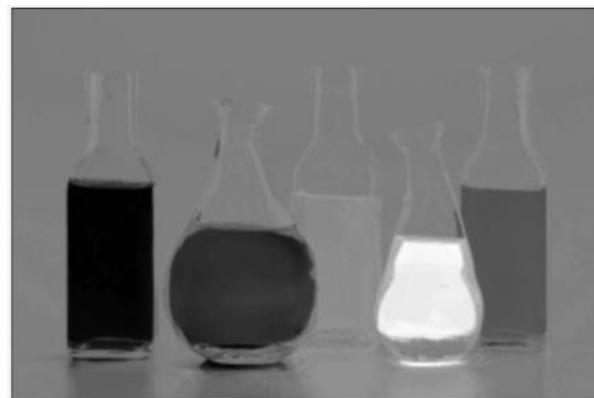
```
$ gmic image.jpg rgb2yiq8 split c
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x1)

---

## rgb2yuv

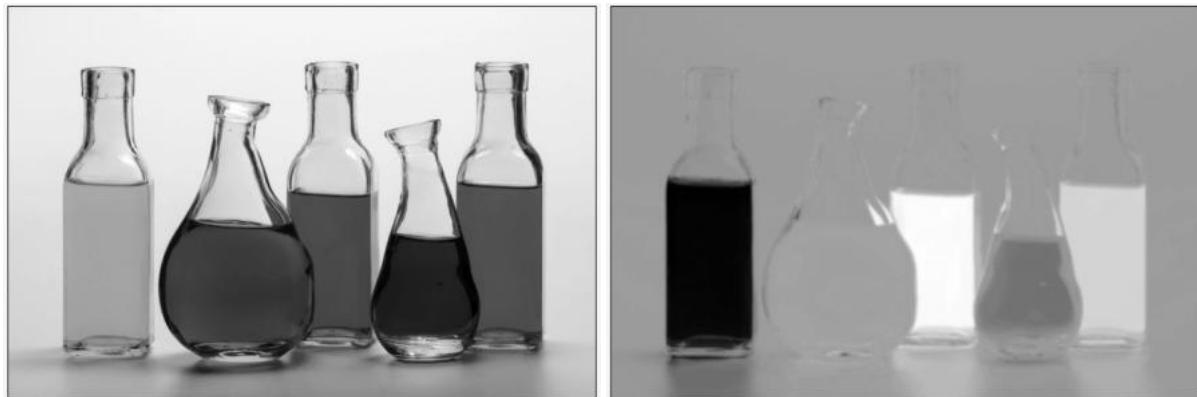
**No arguments**

**Description:**

Convert color representation of selected images from RGB to YUV.

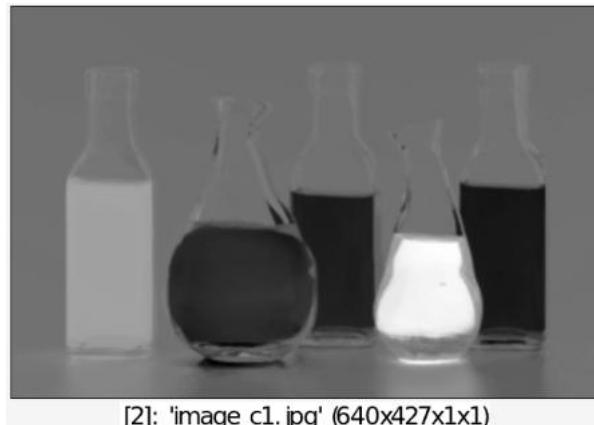
**Example of use:**

```
$ gmic image.jpg rgb2yuv split c
```



[0]: 'image.jpg' (640x427x1x1)

[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x1)

---

## rgb2yuv8

**No arguments**

**Description:**

Convert color representation of selected images from RGB to YUV8.

**Example of use:**

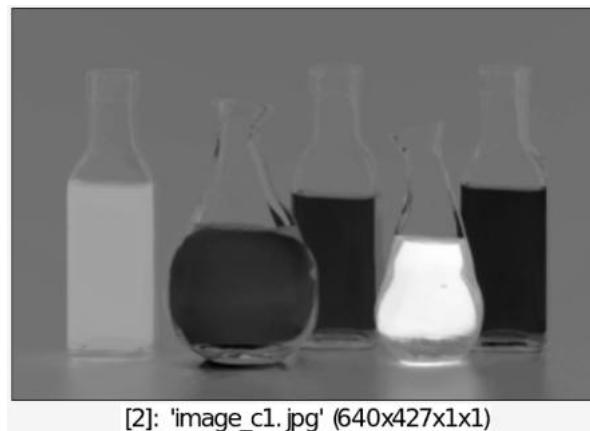
```
$ gmic image.jpg rgb2yuv8 split c
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x1)

## rgba

**No arguments**

### Description:

Return a random int-valued RGBA color.

## ripple

### Arguments:

- `_amplitude, _bandwidth, _shape={ 0=bloc | 1=triangle | 2=sine | 3=sine+ | 4=random }, _angle, _offset`

### Description:

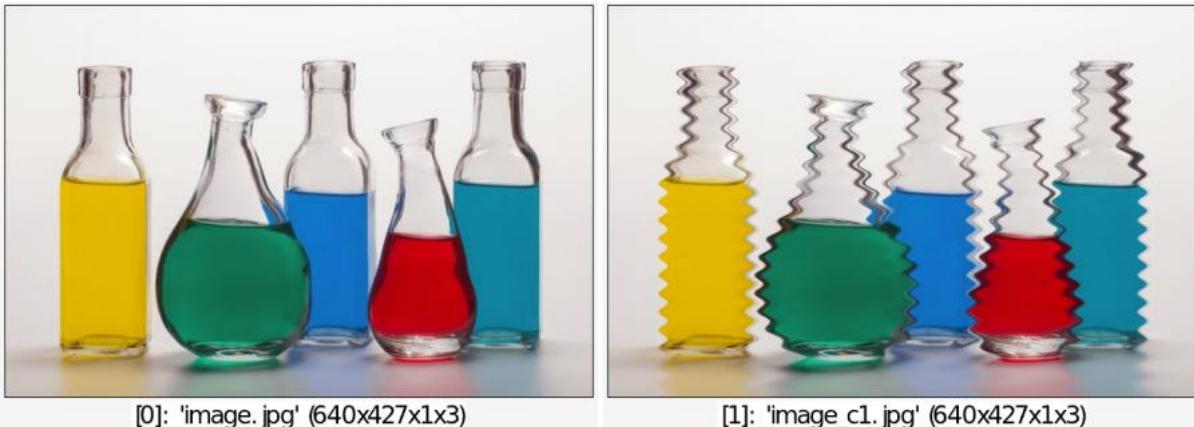
Apply ripple deformation on selected images.

### Default values:

`amplitude=10, bandwidth=10, shape=2, angle=0` and `offset=0`.

### Example of use:

```
$ gmic image.jpg +ripple ,
```



## rodilius

### Arguments:

- `0<=_amplitude<=100, _0<=thickness<=100, _sharpness>=0, _nb_orientations>0, _offs { 0=darker | 1=brighter }`

### Description:

Apply rodilius (fractalius-like) filter on selected images.

### Default values:

`amplitude=10, thickness=10, sharpness=400, nb_orientations=7, offset=0` and `color_mode=1`.

### Examples of use:

- **Example #1**

```
$ gmic image.jpg rodilius 12,10,300,10 normalize_local 10,6
```



- **Example #2**

```
$ gmic image.jpg normalize_local 10,16 rodilius 10,4,400,16 smooth  
60,0,1,1,4 normalize_local 10,16
```



[0]: 'image.jpg' (640x427x1x3)

---

## rol

Built-in command

### Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

### Description:

Compute the bitwise left rotation of selected images with specified value, image or mathematical expression, or compute the pointwise sequential bitwise left rotation of selected images.

### Example of use:

```
$ gmic image.jpg rol 'round(3*x/w,0)' cut 0,255
```



[0]: 'image.jpg' (640x427x1x3)

# rolling\_guidance

## Arguments:

- `std_deviation_s[%]>=0, std_deviation_r[%]>=0, _precision>=0`

## Description:

Apply the rolling guidance filter on selected image.

Rolling guidance filter is a fast image abstraction filter, described in:  
"Rolling Guidance Filter", Qi Zhang Xiaoyong, Shen Li, Xu Jiaya Jia, ECCV'2014.

## Default values:

`std_deviation_s=4`, `std_deviation_r=10` and `precision=0.5`.

## Example of use:

```
$ gmic image.jpg +rolling_guidance , +-
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c2.jpg' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)

---

ror

Built-in command

## Arguments:

- `value[%]` or

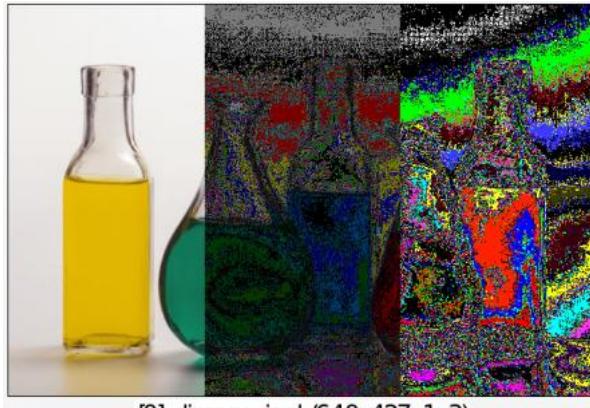
- [image] or
- 'formula' or
- (no arg)

## Description:

Compute the bitwise right rotation of selected images with specified value, image or mathematical expression, or compute the pointwise sequential bitwise right rotation of selected images.

## Example of use:

```
$ gmic image.jpg ror 'round(3*x/w,0)' cut 0,255
```



## rorschach

### Arguments:

- 'smoothness[%]>=0', 'mirroring={ 0=none | 1=x | 2=y | 3=xy }

## Description:

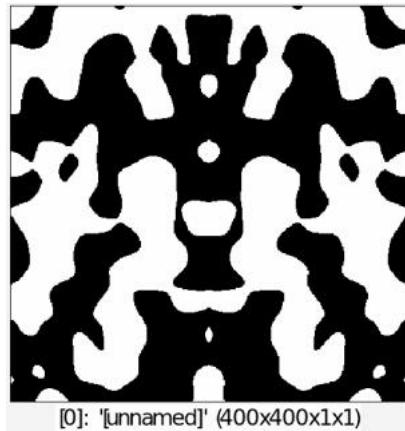
Render rorschach-like inkblots on selected images.

## Default values:

`smoothness=5%` and `mirroring=1`.

## Example of use:

```
$ gmic 400,400 rorschach 3%
```



## rotate

Built-in command

### Arguments:

- `angle,_interpolation,_boundary_conditions,_center_x[%],_center_y[%]` or
- `u,v,w,angle,interpolation,boundary_conditions,_center_x[%],_center_y[%],_cen`

### Description:

Rotate selected images with specified angle (in deg.), and optionally 3D axis (u,v,w).

`interpolation` can be `{ 0=none | 1=linear | 2=bicubic }`.

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

When a rotation center (cx,cy,\_cz) is specified, the size of the image is preserved.

### Default values:

`interpolation=1`, `boundary_conditions=0` and `center_x=center_y=(undefined)`.

### Example of use:

```
$ gmic image.jpg +rotate -25,1,2,50%,50% rotate[0] 25
```



[0]: 'image.jpg' (760x657x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

# rotate3d

Built-in command

## Arguments:

- `u,v,w,angle`

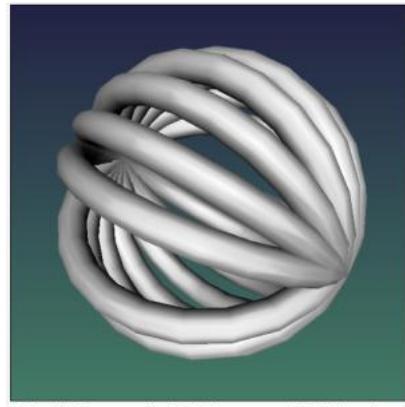
## Description:

Rotate selected 3D objects around specified axis with specified angle (in deg.).

(equivalent to shortcut command `r3d`).

## Example of use:

```
$ gmic torus3d 100,10 double3d 0 repeat 7 +rotate3d[-1] 1,0,0,20 done  
add3d
```



[0]: '[3D torus]' (2304 vert., 2304 prim.)

---

# rotate\_tileable

## Arguments:

- `angle,_max_size_factor>=0`

## Description:

Rotate selected images by specified angle and make them tileable.

If resulting size of an image is too big, the image is replaced by a 1x1 image.

## Default values:

`max_size_factor=8`.

---

# rotate\_tiles

## Arguments:

- `angle, _M>0, N>0`

## Description:

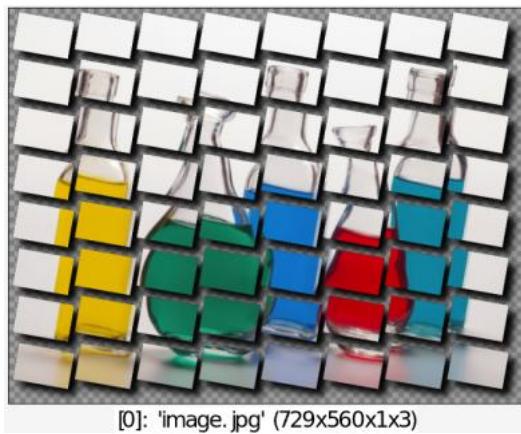
Apply MxN tiled-rotation effect on selected images.

## Default values:

`M=8` and `N=M`.

## Example of use:

```
$ gmic image.jpg to_rgba rotate_tiles 10,8 drop_shadow 10,10  
display_rgba
```



---

## rotation3d

### Arguments:

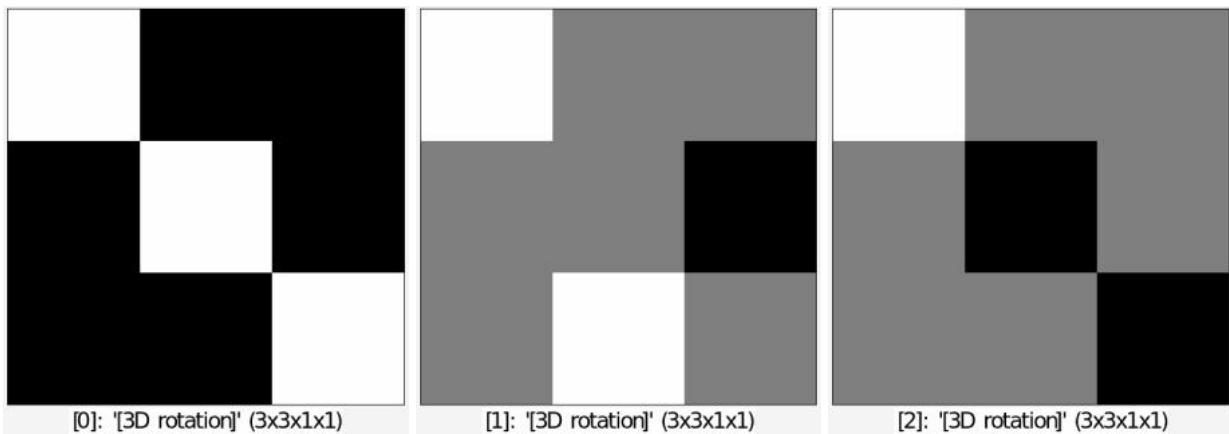
- `u,v,w,angle`

## Description:

Input 3x3 rotation matrix with specified axis and angle (in deg).

## Example of use:

```
$ gmic rotation3d 1,0,0,0 rotation3d 1,0,0,90 rotation3d 1,0,0,180
```



## rotoidoscope

### Arguments:

- `_center_x[%],_center_y[%],_tiles>0,_smoothness[%]>=0,_boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

### Description:

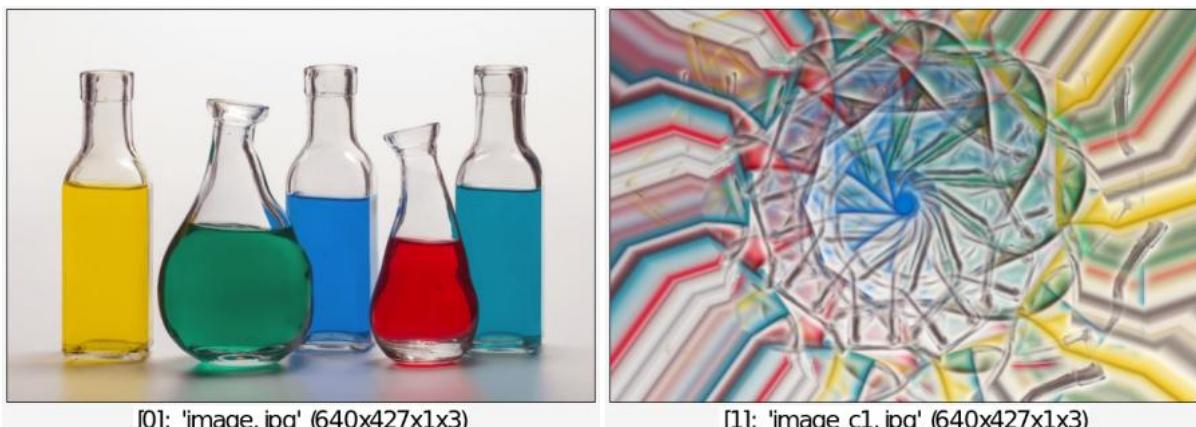
Create rotational kaleidoscope effect from selected images.

### Default values:

`center_x=center_y=50%`, `tiles=10`, `smoothness=1` and `boundary_conditions=3`.

### Example of use:

```
$ gmic image.jpg +rotoidoscope ,
```



## round

Built-in command

### Arguments:

- `rounding_value>=0, _rounding_type` or
- `(no arg)`

## Description:

Round values of selected images.

`rounding_type` can be `{ -1=backward | 0=nearest | 1=forward }`.

## Default values:

`rounding_type=0`.

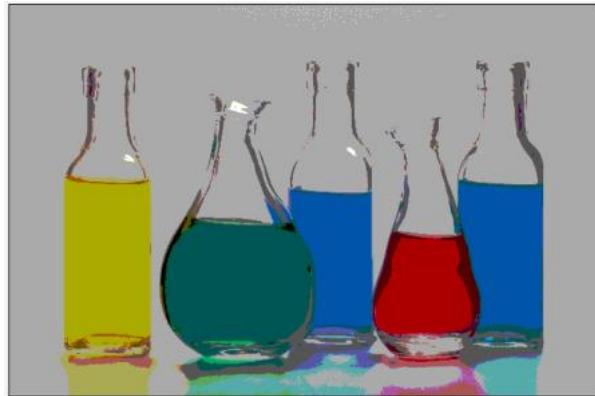
## Examples of use:

- Example #1

```
$ gmic image.jpg +round 100
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

- Example #2

```
$ gmic image.jpg mul {pi/180} sin +round
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

roundify

## Arguments:

- `gamma>=0`

## Description:

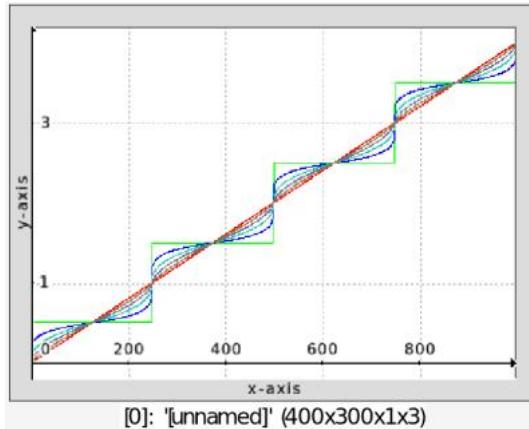
Apply roundify transformation on float-valued data, with specified gamma.

## Default values:

`gamma=0`.

## Example of use:

```
$ gmic 1000 fill '4*x/w' repeat 5 +roundify[0] {$>*0.2} done append c  
display_graph 400,300
```



---

## ROWS

## Arguments:

- `y0[%],_y1[%]`

## Description:

Keep only specified rows of selected images.

Dirichlet boundary conditions are used when specified rows are out of range.

## Default values:

`y1=y0`.

## Example of use:

```
$ gmic image.jpg rows -25%,50%
```



---

## rprogress

### Arguments:

- `0<=value<=100 | -1 | "command", 0<=value_min<=100, 0<=value_max<=100`

### Description:

Set the progress index of the current processing pipeline (relatively to

previously defined progress bounds), or call the specified command with specified progress bounds.

---

## run

### Arguments:

- `"G'MIC pipeline"`

### Description:

Run specified G'MIC pipeline.

This is only useful when used from a shell, e.g. to avoid shell substitutions to happen in argument.

---

## ryb2rgb

### No arguments

### Description:

Convert color representation of selected images from RYB to RGB.

---

## sample

## Arguments:

- `_name1={ ? | apples | balloons | barbara | boats | bottles | butterfly | cameraman | car | cat | cliff | chick | colorful | david | dog | duck | eagle | elephant | earth | flower | fruits | gmicky | gmicky_mahvin | gmicky_wilber | greece | gummy | house | inside | landscape | leaf | lena | leno | lion | mandrill | monalisa | monkey | parrots | pencils | peppers | portrait0 | portrait1 | portrait2 | portrait3 | portrait4 | portrait5 | portrait6 | portrait7 | portrait8 | portrait9 | roddy | rooster | rose | square | swan | teddy | tiger | tulips | wall | waterfall | zelda } ,_name2,...,_nameN,_width={ >=0 | 0 (auto) },_height = { >=0 | 0 (auto) } } or`
- `(no arg)`

## Description:

Input a new sample RGB image (opt. with specified size).

(equivalent to shortcut command `sp`).

Argument `name` can be replaced by an integer which serves as a sample index.

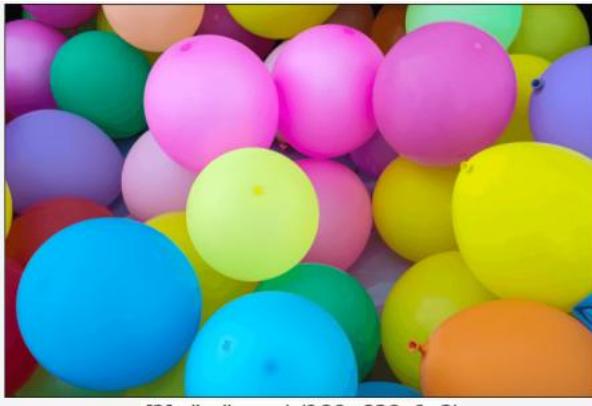
## Example of use:

```
$ gmic repeat 6 sample done
```



[0]: 'portrait6' (800x800x1x3)

[1]: 'monalisa' (700x800x1x3)



[2]: 'balloons' (960x638x1x3)



[3]: 'portrait3' (800x800x1x3)



[4]: 'gmicky\_mahvin' (600x800x1x3)



---

## scale2x

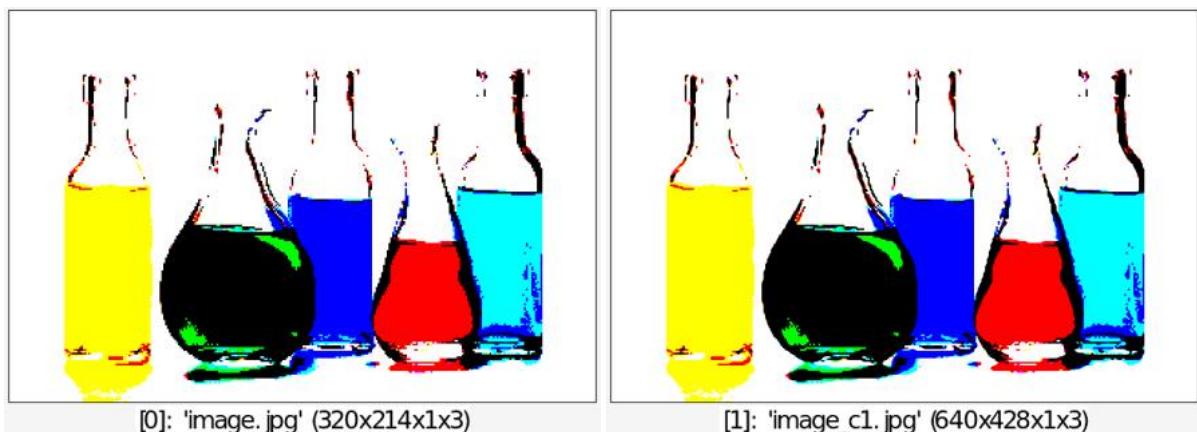
**No arguments**

**Description:**

Resize selected images using the Scale2x algorithm.

**Example of use:**

```
$ gmic image.jpg threshold 50% resize 50%,50% +scale2x
```



---

## scale3x

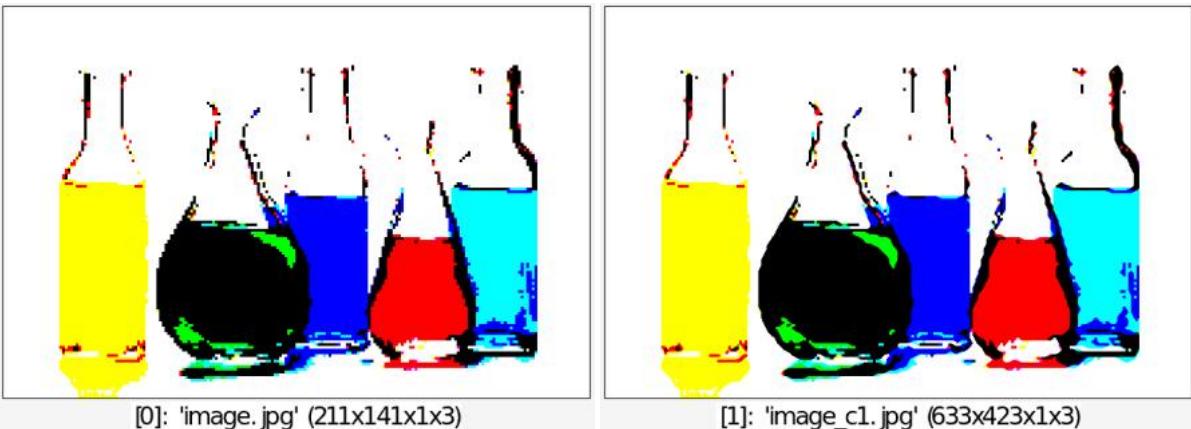
**No arguments**

**Description:**

Resize selected images using the Scale3x algorithm.

**Example of use:**

```
$ gmic image.jpg threshold 50% resize 33%,33% +scale3x
```



## scale\_dcci2x

### Arguments:

- `_edge_threshold>=0, _exponent>0, _extend_1px={ 0=false | 1=true }`

### Description:

Double image size using directional cubic convolution interpolation,

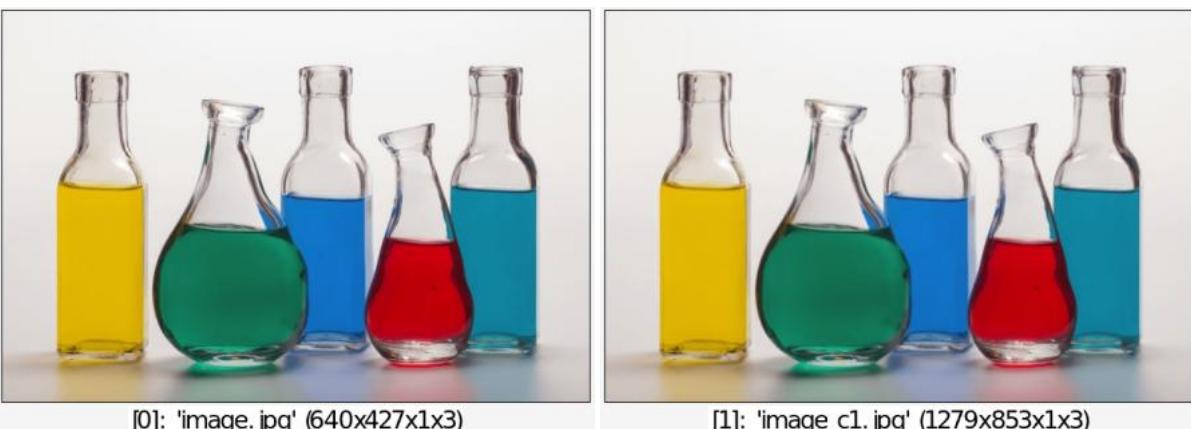
as described in [https://en.wikipedia.org/wiki/Directional\\_Cubic\\_Convolution\\_Interpolation](https://en.wikipedia.org/wiki/Directional_Cubic_Convolution_Interpolation).

### Default values:

`edge_threshold=1.15`, `exponent=5` and `extend_1px=0`.

### Example of use:

```
$ gmic image.jpg +scale_dcci2x ,
```



## scanlines

## Arguments:

- `_amplitude`, `_bandwidth`, `_shape={ 0=bloc | 1=triangle | 2=sine | 3=sine+ | 4=random }`, `_angle`, `_offset`

## Description:

Apply ripple deformation on selected images.

## Default values:

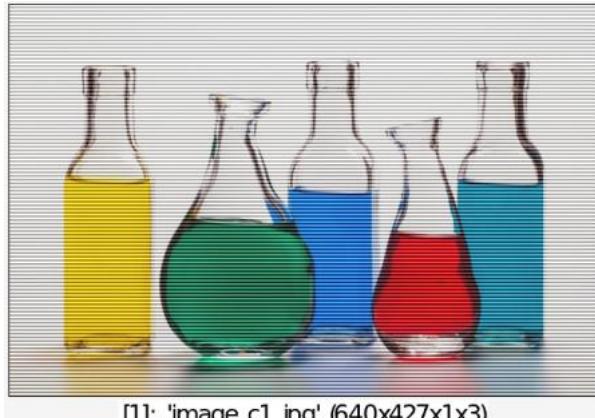
`amplitude=60`, `bandwidth=2`, `shape=0`, `angle=0` and `offset=0`.

## Example of use:

```
$ gmic image.jpg +scanlines ,
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## screen

Built-in command

## Arguments:

- `_x0[%]`, `_y0[%]`, `_x1[%]`, `_y1[%]`

## Description:

Take screenshot, optionally grabbed with specified coordinates, and insert it at the end of the image list.

---

## seamcarve

## Arguments:

- `_width[%]>=0`, `_height[%]>=0`, `_is_priority_channel={ 0 | 1 }`, `_is_antialiasing={ 0 | 1 }`, `_maximum_seams[%]>=0`

## Description:

Resize selected images with specified 2D geometry, using the seam-carving algorithm.

## Default values:

`height=100%`, `is_priority_channel=0`, `is_antialiasing=1` and `maximum_seams=25%`.

## Example of use:

```
$ gmic image.jpg seamcarve 60%
```



---

## segment\_watershed

### Arguments:

- `_threshold>=0`

## Description:

Apply watershed segmentation on selected images.

## Default values:

`threshold=2`.

## Example of use:

```
$ gmic image.jpg segment_watershed 2
```



## select

Built-in command

### Arguments:

- `feature_type, _X[%]>=0, _Y[%]>=0, _Z[%]>=0, _exit_on_anykey={ 0 | 1 }, _is_deep_selection={ 0 | 1 }`

### Description:

Interactively select a feature from selected images (use the instant display window [0] if opened).

`feature_type` can be `{ 0=point | 1=segment | 2=rectangle | 3=ellipse }`.

Arguments `X, Y, Z` determine the initial selection view, for 3D volumetric images.

The retrieved feature is returned as a 3D vector (if `feature_type==0`) or as a 6d vector (if `feature_type!=0`) containing the feature coordinates.

### Default values:

`X=Y=Z=(undefined), exit_on_anykey=0` and `is_deep_selection=0`.

## select\_color

### Arguments:

- `tolerance[%]>=0, col1, ..., colN`

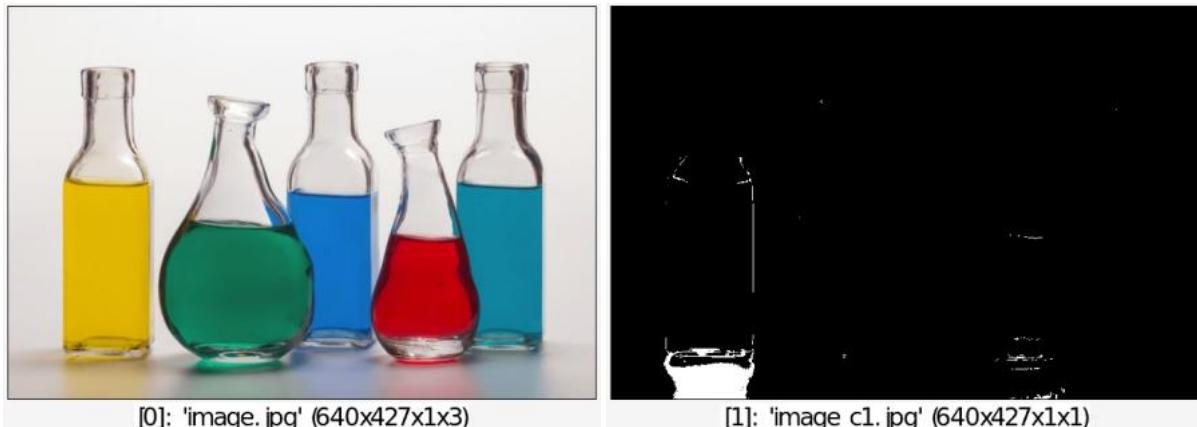
### Description:

Select pixels with specified color in selected images.

This command has a [tutorial page](#).

### Example of use:

```
$ gmic image.jpg +select_color 40,204,153,110
```



---

## sepia

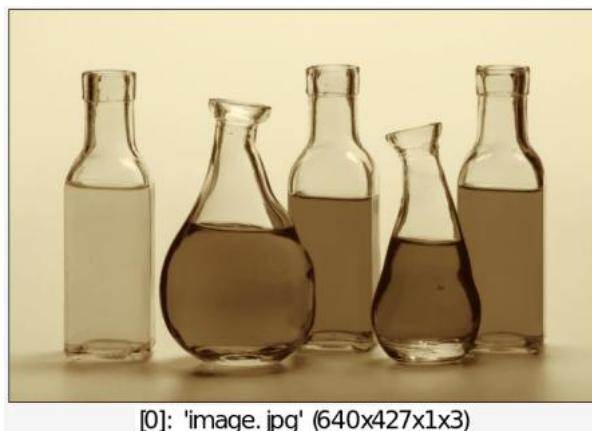
**No arguments**

**Description:**

Apply sepia tones effect on selected images.

**Example of use:**

```
$ gmic image.jpg sepia
```



---

## serialize

Built-in command

**Arguments:**

- `_datatype,_is_compressed={ 0 | 1 },_store_names={ 0 | 1 }`

**Description:**

Serialize selected list of images into a single image, optionnally in a compressed form.

`datatype` can be `{ auto | uchar | char | ushort | short | uint | int | uint64 | int64 | float | double }`.

Specify `datatype` if all selected images have a range of values constrained to a particular datatype,

in order to minimize the memory footprint.

The resulting image has only integers values in [0,255] and can then be saved as a raw image of unsigned chars (doing so will output a valid .cimg[z] or .gmz file).

If `store_names` is set to `1`, serialization uses the .gmz format to store data in memory (otherwise the .cimg[z] format).

## Default values:

`datatype=auto`, `is_compressed=1` and `store_names=1`.

## Example of use:

```
$ gmic image.jpg +serialize uchar +unserialize[-1]
```



## set

Built-in command

### Arguments:

- `value,_x[%],_y[%],_z[%],_c[%]`

### Description:

Set pixel value in selected images, at specified coordinates.

(equivalent to shortcut command `=`).

If specified coordinates are outside the image bounds, no action is performed.

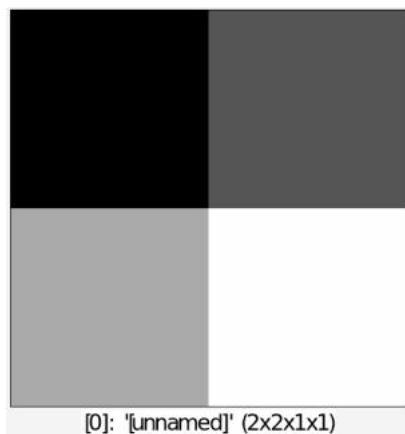
## Default values:

`x=y=z=c=0`.

## Examples of use:

- Example #1

```
$ gmic 2,2 set 1,0,0 set 2,1,0 set 3,0,1 set 4,1,1
```



- Example #2

```
$ gmic image.jpg repeat 10000 set 255,{u(100)}%,{u(100)}%,0,{u(100)}%
done
```



---

## shade\_stripes

### Arguments:

- `_frequency>=0, _direction={ 0=horizontal | 1=vertical }`, `_darkness>=0, _lightness>=0`

## Description:

Add shade stripes to selected images.

## Default values:

`frequency=5`, `direction=1`, `darkness=0.8` and `lightness=2`.

## Example of use:

```
$ gmic image.jpg +shade_stripes 30
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## shadow\_patch

### Arguments:

- `_opacity>=0`

## Description:

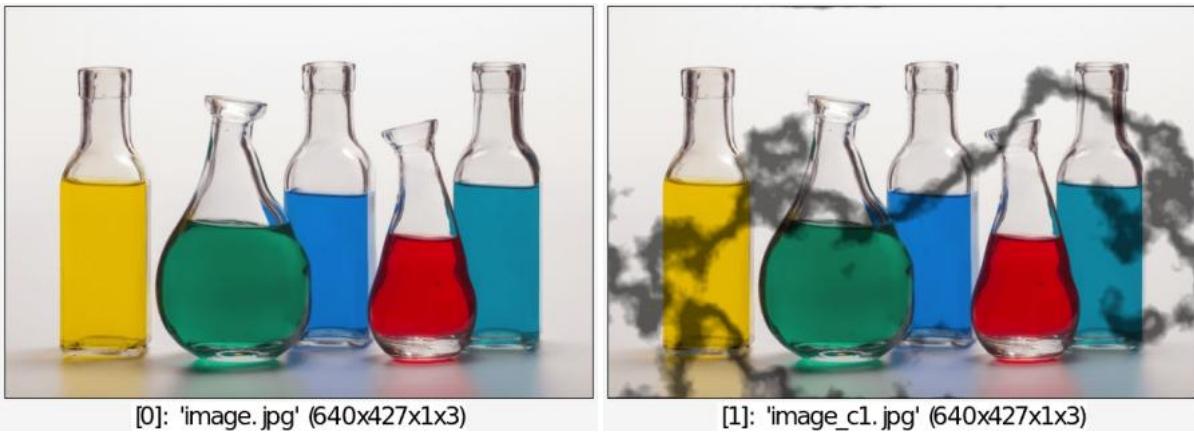
Add shadow patches to selected images.

## Default values:

`opacity=0.7`.

## Example of use:

```
$ gmic image.jpg +shadow_patch 0.4
```



---

## shape2bump

### Arguments:

- `_resolution>=0, 0<=weight_avg_max_avg<=1, _dilation, _smoothness>=0`

### Description:

Estimate bumpmap from binary shape in selected images.

### Default values:

`resolution=256, weight_avg_max=0.75, dilation=0` and `smoothness=100`.

---

## shape\_circle

### Arguments:

- `_size>=0`

### Description:

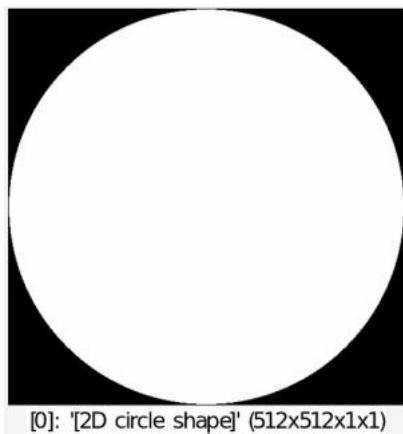
Input a 2D circle binary shape with specified size.

### Default values:

`size=512`.

### Example of use:

```
$ gmic shape_circle ,
```



---

## shape\_cupid

### Arguments:

- `_size>=0`

### Description:

Input a 2D cupid binary shape with specified size.

### Default values:

`size=512`.

### Example of use:

```
$ gmic shape_cupid ,
```



---

## shape\_diamond

### Arguments:

- `_size>=0`

## Description:

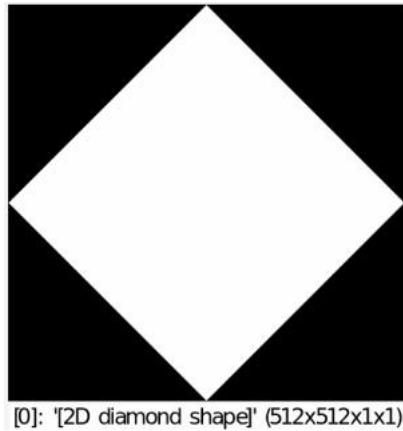
Input a 2D diamond binary shape with specified size.

## Default values:

`size=512`.

## Example of use:

```
$ gmic shape_diamond ,
```



---

## shape\_dragon

### Arguments:

- `_size>=0,_recursion_level>=0,_angle`

## Description:

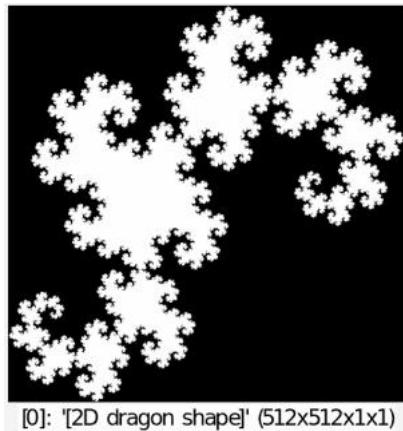
Input a 2D Dragon curve with specified size.

## Default values:

`size=512`, `recursion_level=18` and `angle=0`.

## Example of use:

```
$ gmic shape_dragon ,
```



---

## shape\_fern

### Arguments:

- `_size>=0, _density[%]>=0, _angle, 0<=_opacity<=1, _type={ 0=Asplenium adiantum-nigrum | 1=Thelypteridaceae }`

### Description:

Input a 2D Barnsley fern with specified size.

### Default values:

`size=512`, `density=50%`, `angle=30`, `opacity=0.3` and `type=0`.

### Example of use:

```
$ gmic shape_fern ,
```



---

## shape\_gear

### Arguments:

- `_size>=0, _nb_teeth>0, 0<=_height_teeth<=100, 0<=_offset_teeth<=100, 0<=_inner_r`

## Description:

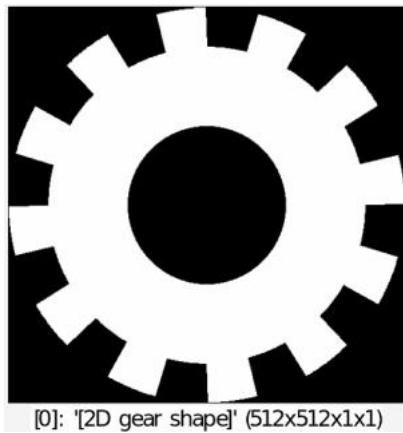
Input a 2D gear binary shape with specified size.

## Default values:

`size=512, nb_teeth=12, height_teeth=20, offset_teeth=0` and `inner_radius=40`.

## Example of use:

```
$ gmic shape_gear ,
```



---

## shape\_heart

### Arguments:

- `_size>=0`

## Description:

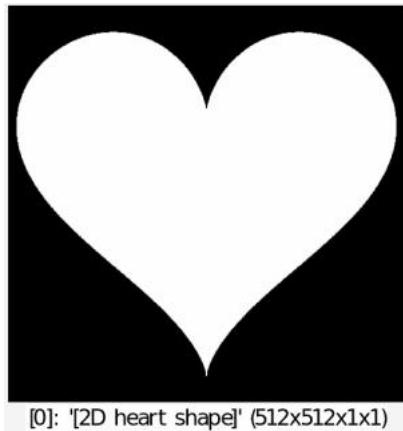
Input a 2D heart binary shape with specified size.

## Default values:

`size=512`.

## Example of use:

```
$ gmic shape_heart ,
```



---

## shape\_polygon

### Arguments:

- `_size>=0, _nb_vertices>=3, _angle`

### Description:

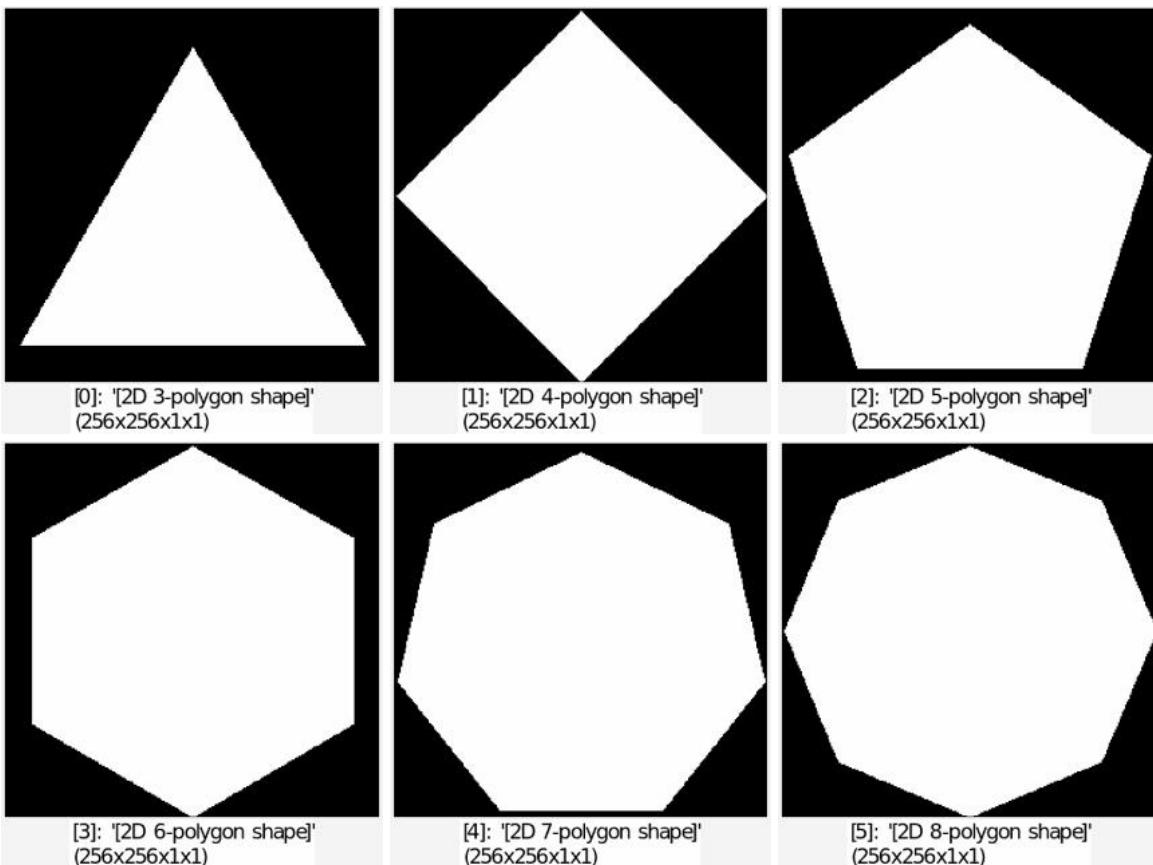
Input a 2D polygonal binary shape with specified geometry.

### Default values:

`size=512`, `nb_vertices=5` and `angle=0`.

### Example of use:

```
$ gmic repeat 6 shape_polygon 256,{3+$>} done
```



---

## shape\_snowflake

### Arguments:

- `size>=0, 0<=_nb_recursions<=6`

### Description:

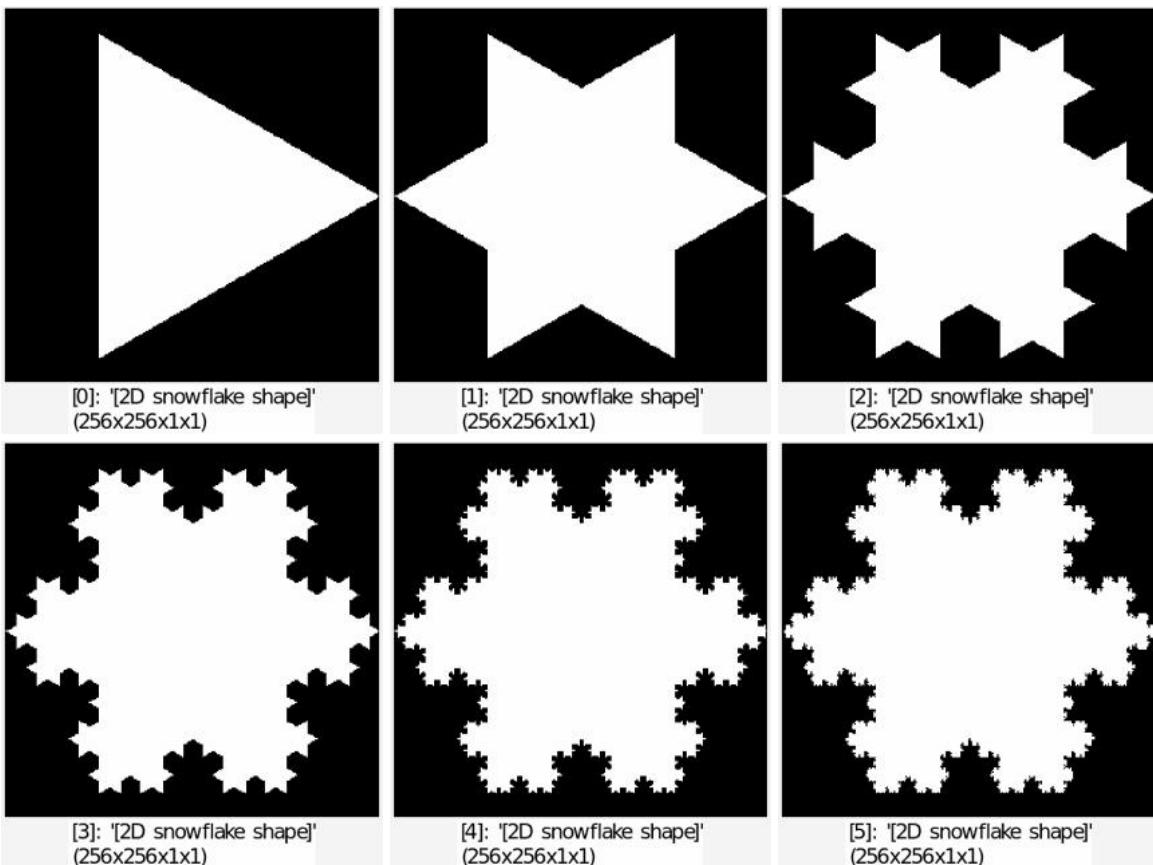
Input a 2D snowflake binary shape with specified size.

### Default values:

`size=512` and `nb_recursions=5`.

### Example of use:

```
$ gmic repeat 6 shape_snowflake 256,> done
```



## shape\_star

### Arguments:

- `_size>=0, _nb_branches>0, 0<=_thickness<=1`

### Description:

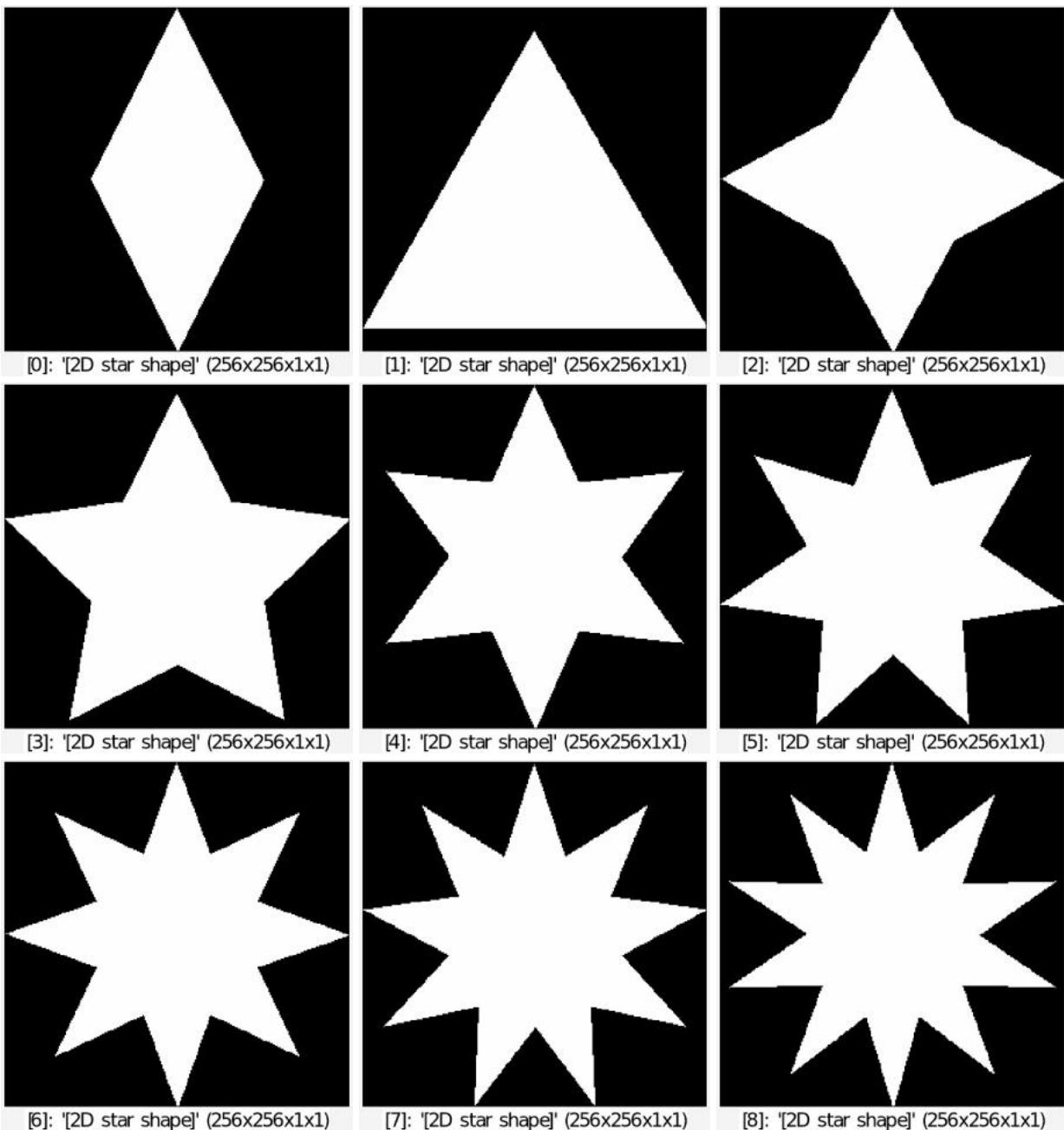
Input a 2D star binary shape with specified size.

### Default values:

`size=512`, `nb_branches=5` and `thickness=0.38`.

### Example of use:

```
$ gmic repeat 9 shape_star 256,{$>+2} done
```



## shared

Built-in command

### Arguments:

- `x0[%],x1[%],y[%],z[%],c[%]` or
- `y0[%],y1[%],z[%],c[%]` or
- `z0[%],z1[%],c[%]` or
- `c0[%],c1[%]` or
- `c0[%]` or
- `(no arg)`

### Description:

Insert shared buffers from (opt. points/rows/planes/channels of) selected images.

Shared buffers cannot be returned by a command, nor a local environment.

(equivalent to shortcut command `sh`).

This command has a [tutorial page](#).

## Examples of use:

- **Example #1**

```
$ gmic image.jpg shared 1 blur[-1] 3 remove[-1]
```



[0]: 'image.jpg' (640x427x1x3)

- **Example #2**

```
$ gmic image.jpg repeat s shared 25%,75%,0,$> mirror[-1] x remove[-1]
done
```



[0]: 'image.jpg' (640x427x1x3)

---

## sharpen

### Arguments:

- `amplitude>=0` or
- `amplitude>=0,edge>=0,_alpha[%],_sigma[%]`

### Description:

Sharpen selected images by inverse diffusion or shock filters methods.

**edge** must be specified to enable shock-filter method.

## Default values:

`edge=0`, `alpha=0` and `sigma=0`.

## Examples of use:

- **Example #1**

```
$ gmic image.jpg sharpen 300
```



- **Example #2**

```
$ gmic image.jpg blur 5 sharpen 300,1
```



---

## shell\_cols

### No arguments

### Description:

Return the estimated number of columns of the current shell.

# shift

Built-in command

## Arguments:

- `vx[%],_vy[%],_vz[%],_vc[%],_boundary_conditions,_interpolation={0=nearest_neighbor | 1=linear }`

## Description:

Shift selected images by specified displacement vector.

Displacement vector can be non-integer in which case linear interpolation should be chosen.

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

## Default values:

`boundary_conditions=0` and `interpolation=0`.

## Example of use:

```
$ gmic image.jpg +shift[0] 50%,50%,0,0,0 +shift[0] 50%,50%,0,0,1  
+shift[0] 50%,50%,0,0,2
```



# shift\_tiles

## Arguments:

- `M>0, N>0, amplitude`

## Description:

Apply MxN tiled-shift effect on selected images.

## Default values:

`N=M` and `amplitude=20`.

## Example of use:

```
$ gmic image.jpg +shift_tiles 8,8,10
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

# shrink\_x

## Arguments:

- `size_x>=0`

## Description:

Shrink selected images along the x-axis.

## Example of use:

```
$ gmic image.jpg shrink_x 30
```



[0]: 'image.jpg' (580x427x1x3)

---

## shrink\_xy

### Arguments:

- `size>=0`

### Description:

Shrink selected images along the xy-axes.

### Example of use:

```
$ gmic image.jpg shrink_xy 30
```



[0]: 'image.jpg' (580x367x1x3)

---

## shrink\_xyz

### Arguments:

- `size>=0`

### Description:

Shrink selected images along the xyz-axes.

---

## shrink\_y

### Arguments:

- `size_y>=0`

### Description:

Shrink selected images along the y-axis.

### Example of use:

```
$ gmic image.jpg shrink_y 30
```



## shrink\_z

### Arguments:

- `size_z>=0`

### Description:

Shrink selected images along the z-axis.

---

## sierpinsk

### Arguments:

- `recursion_level>=0`

### Description:

Draw Sierpinski triangle on selected images.

## Default values:

`recursion_level=7`.

## Example of use:

```
$ gmic image.jpg sierpinski 7
```



---

## sierpinski3d

### Arguments:

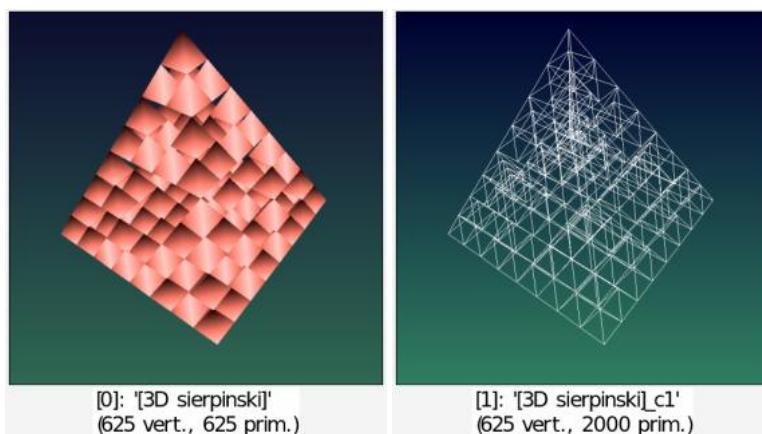
- `_recursion_level>=0,_width,_height`

### Description:

Input 3d Sierpinski pyramid.

## Example of use:

```
$ gmic sierpinski3d 3,100,-100 +primitives3d 1 color3d[-2] ${-rgb}
```



# sign

Built-in command

No arguments

## Description:

Compute the pointwise sign of selected images.

## Examples of use:

- Example #1

```
$ gmic image.jpg +sub {ia} sign[-1]
```



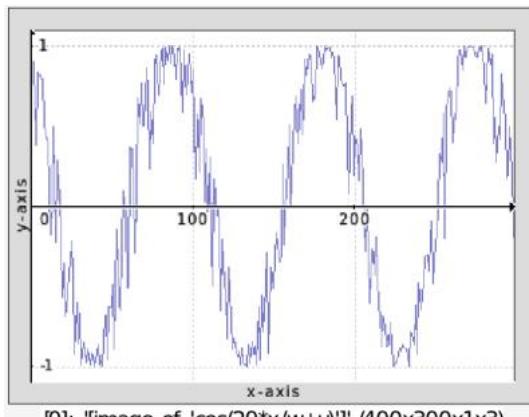
[0]: 'image.jpg' (640x427x1x3)



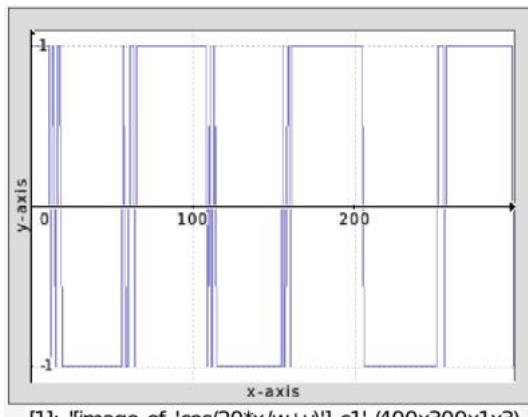
[1]: 'image\_c1.jpg' (640x427x1x3)

- Example #2

```
$ gmic 300,1,1,1,'cos(20*x/w+u)' +sign display_graph 400,300
```



[0]: '[image of 'cos(20\*x/w+u)']' (400x300x1x3)



[1]: '[image of 'cos(20\*x/w+u)']\_c1' (400x300x1x3)

---

# sin

Built-in command

No arguments

## Description:

Compute the pointwise sine of selected images.

This command has a [tutorial page](#).

## Examples of use:

- Example #1

```
$ gmic image.jpg +normalize 0,{2*pi} sin[-1]
```



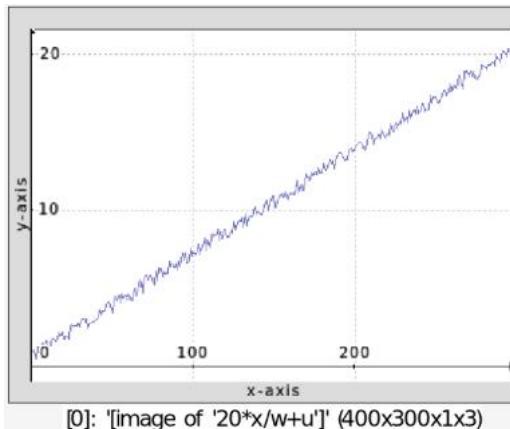
[0]: 'image.jpg' (640x427x1x3)



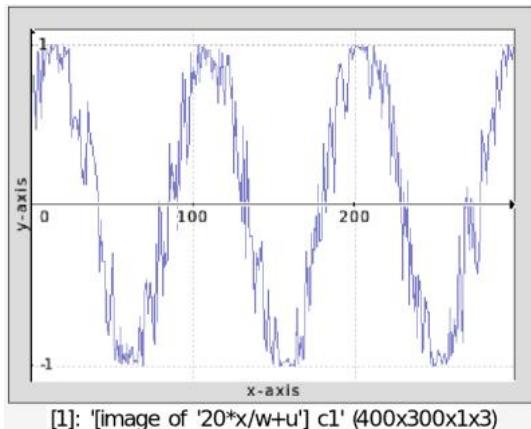
[1]: 'image\_c1.jpg' (640x427x1x3)

- Example #2

```
$ gmic 300,1,1,1,'20*x/w+u' +sin display_graph 400,300
```



[0]: '[image of '20\*x/w+u']' (400x300x1x3)



[1]: '[image of '20\*x/w+u']\_c1' (400x300x1x3)

---

## sinc

[Built-in command](#)

No arguments

## Description:

Compute the pointwise sinc function of selected images.

## Examples of use:

- Example #1

```
$ gmic image.jpg +normalize {-2*pi},{2*pi} sinc[-1]
```



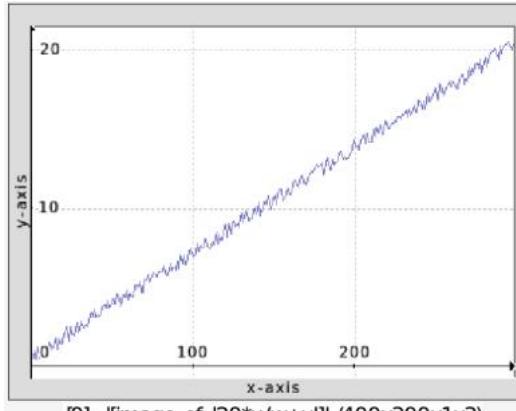
[0]: 'image.jpg' (640x427x1x3)



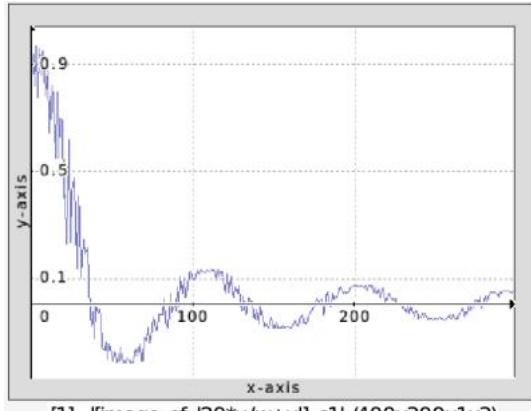
[1]: 'image\_c1.jpg' (640x427x1x3)

- **Example #2**

```
$ gmic 300,1,1,1,'20*x/w+u' +sinc display_graph 400,300
```



[0]: '[image of "20\*x/w+u"]' (400x300x1x3)



[1]: '[image of "20\*x/w+u"]\_c1' (400x300x1x3)

## sinh

Built-in command

No arguments

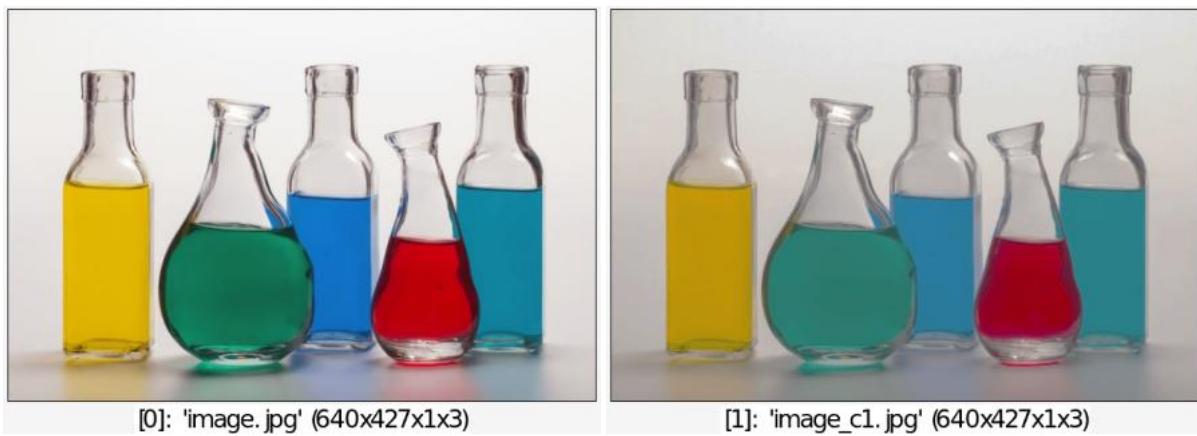
**Description:**

Compute the pointwise hyperbolic sine of selected images.

**Examples of use:**

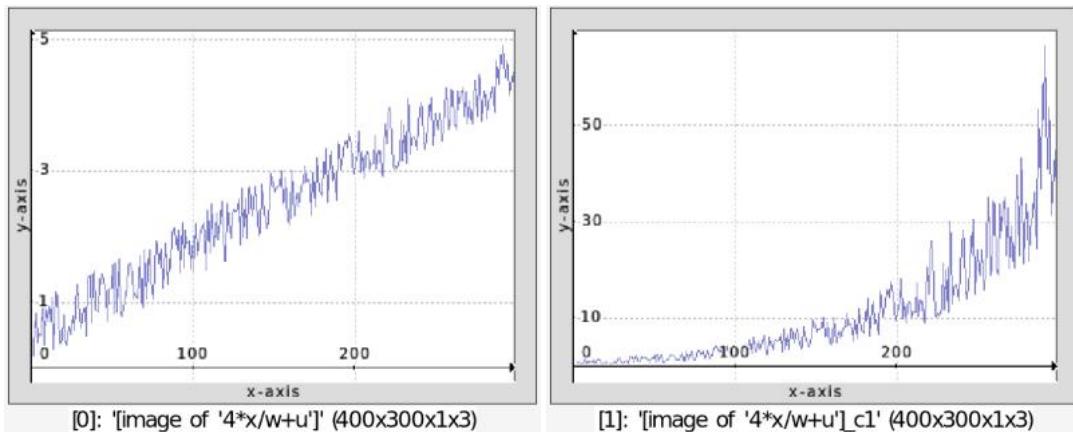
- **Example #1**

```
$ gmic image.jpg +normalize -3,3 sinh[-1]
```



- **Example #2**

```
$ gmic 300,1,1,1,'4*x/w+u' +sinh display_graph 400,300
```



## size3d

### No arguments

#### Description:

Return bounding box size of the last selected 3D object.

## size\_value

### No arguments

#### Description:

Return the size (in bytes) of an image value.

## skeleton

## Arguments:

- `_boundary_conditions={ 0=dirichlet | 1=neumann }`

## Description:

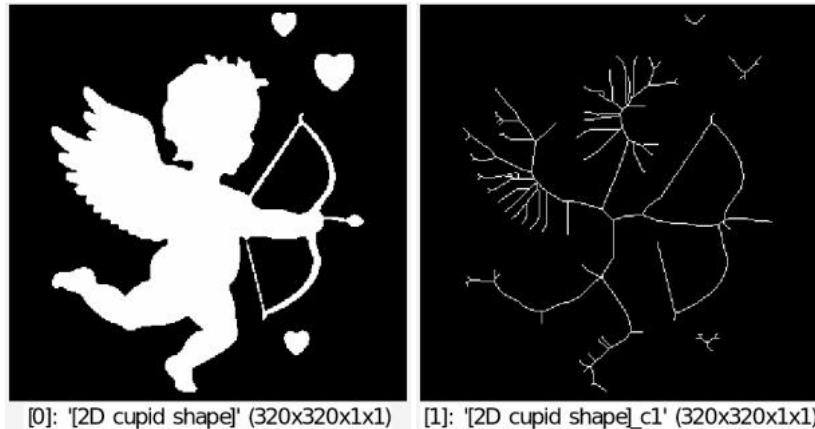
Compute skeleton of binary shapes using distance transform and constrained thinning.

## Default values:

`boundary_conditions=1`.

## Example of use:

```
$ gmic shape_cupid 320 +skeleton 0
```



---

## skeleton3d

### Arguments:

- `_metric,_frame_type={ 0=squares | 1=diamonds | 2=circles | 3=auto }`, `_skeleton_opacity,_frame_opacity,_is_frame_wireframe={ 0 | 1 }`

### Description:

Build 3D skeletal structure object from 2d binary shapes located in selected images.

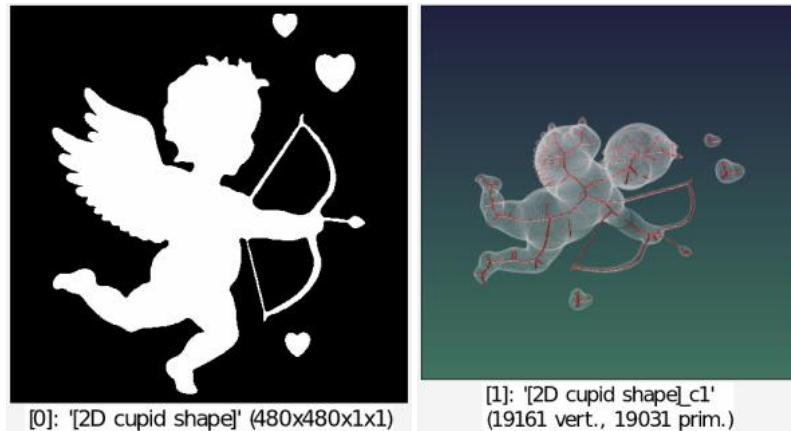
`metric` can be `{ 0=chebyshev | 1=manhattan | 2=euclidean }`.

### Default values:

`metric=2`, `bones_type=3`, `skeleton_opacity=1` and `frame_opacity=0.1`.

### Example of use:

```
$ gmic shape_cupid 480 +skeleton3d ,
```



---

## sketchbw

### Arguments:

- `_nb_angles>0, _start_angle, _angle_range>=0, _length>=0, _threshold>=0, _opacity, { 0 | 1 }, _is_curved={ 0 | 1 }`

### Description:

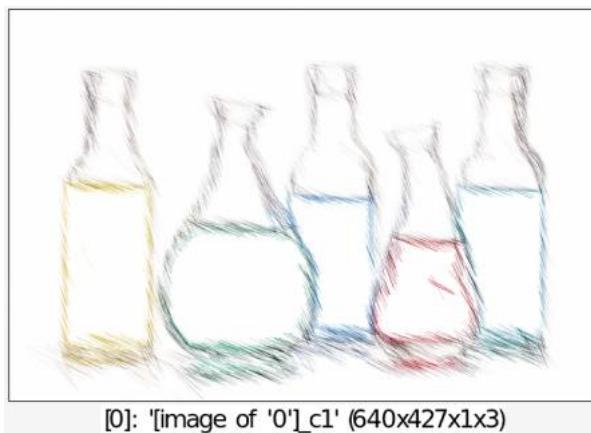
Apply sketch effect to selected images.

### Default values:

`nb_angles=2, start_angle=45, angle_range=180, length=30, threshold=3, opacity=0.03, bgfactor=0, density=0.6, sharpness=0.1, anisotropy=0.6, smoothness=0.25, coherence=1, is_boost=0` and `is_curved=1`.

### Example of use:

```
$ gmic image.jpg +sketchbw 1 reverse blur[-1] 3 blend[-2, -1] overlay
```



---

## skip

Built-in command

## Arguments:

- `item`

## Description:

Do nothing but skip specified item.

---

# slic

## Arguments:

- `size>0, _regularity>=0, _nb_iterations>0`

## Description:

Segment selected 2D images with superpixels, using the SLIC algorithm (Simple Linear Iterative Clustering).

Scalar images of increasingly labeled pixels are returned.

Reference paper: Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Susstrunk, S. (2010). SLIC Superpixels (No. EPFL-REPORT-149300).

## Default values:

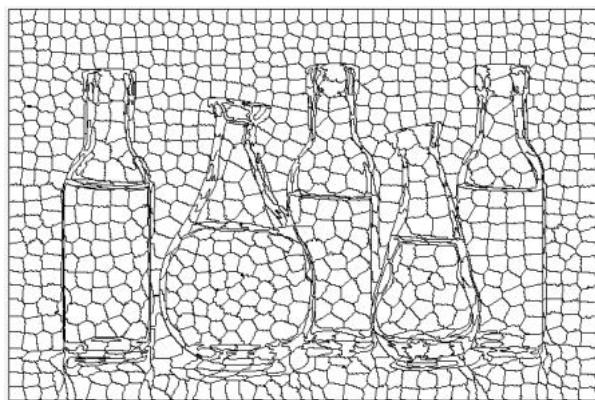
`size=16`, `regularity=10` and `nb_iterations=10`.

## Example of use:

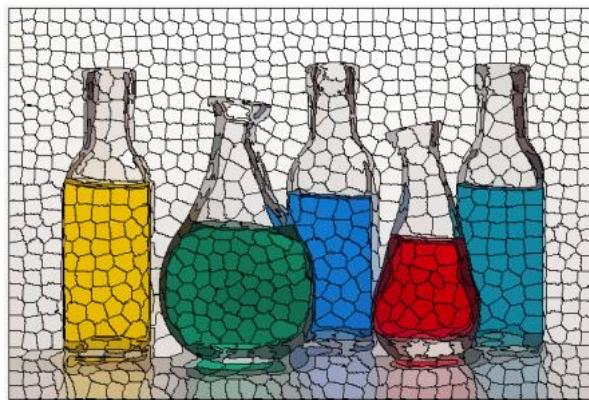
```
$ gmic image.jpg +srgb2lab slic[-1] 16 +blend shapeaverage f[-2]
"j(1,0)==i && j(0,1)==i" *[-1] [-2]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: '[unnamed]' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x3)

## slices

### Arguments:

- `z0[%],_z1[%]`

### Description:

Keep only specified slices of selected images.

Dirichlet boundary conditions are used when specified slices are out of range.

### Default values:

`z1=z0`.

## smooth

Built-in command

### Arguments:

- `amplitude[%]>=0,_sharpness>=0,0<=_anisotropy<=1,_alpha[%],_sigma[%],_dl>0,_d{ 0 | 1 } or`
- `nb_iterations>=0,_sharpness>=0,_anisotropy,_alpha,_sigma,_dt>0,0 or`

- `[tensor_field],_amplitude>=0,_dl>0,_da>0,_precision>0,_interpolation,_fast_a { 0 | 1 }` or
- `[tensor_field],_nb_iters>=0,_dt>0,0`

## Description:

Smooth selected images anisotropically using diffusion PDE's, with specified field of diffusion tensors.

`interpolation` can be `{ 0=nearest | 1=linear | 2=runga-kutta }`.

## Default values:

`sharpness=0.7, anisotropy=0.3, alpha=0.6, sigma=1.1, dl=0.8, da=30, precision=2, interpolation=0` and `fast_approx=1`.

This command has a [tutorial page](#).

## Examples of use:

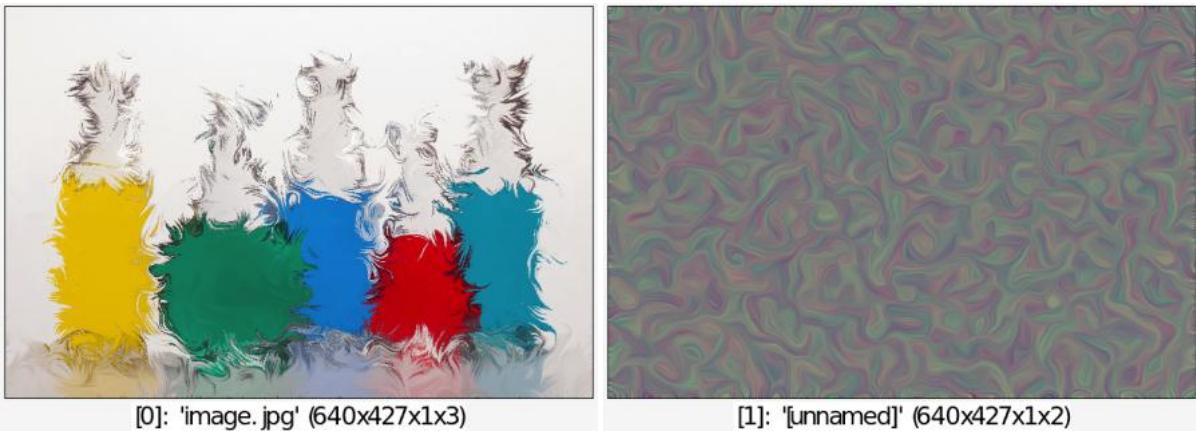
- Example #1

```
$ gmic image.jpg repeat 3 smooth 40,0,1,1,2 done
```



- Example #2

```
$ gmic image.jpg 100%,100%,1,2 rand[-1] -100,100 repeat 2 smooth[-1]
100,0.2,1,4,4 done warp[0] [-1],1,1
```



---

## snapshot3d

### Arguments:

- `_size>0, _zoom>=0, _backgroundR, _backgroundG, _backgroundB, _backgroundA` or
- `[background_image], zoom>=0`

### Description:

Take 2d snapshots of selected 3D objects.

Set `zoom` to 0 to disable object auto-scaling.

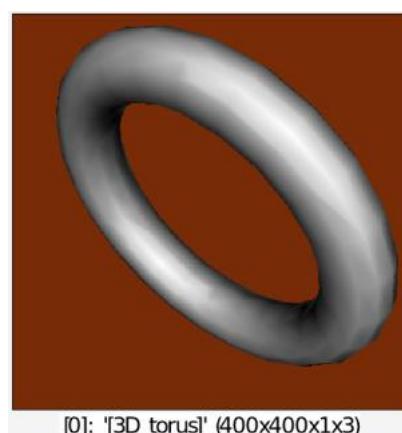
### Default values:

`size=512, zoom=1` and `[background_image]=(default)`.

### Examples of use:

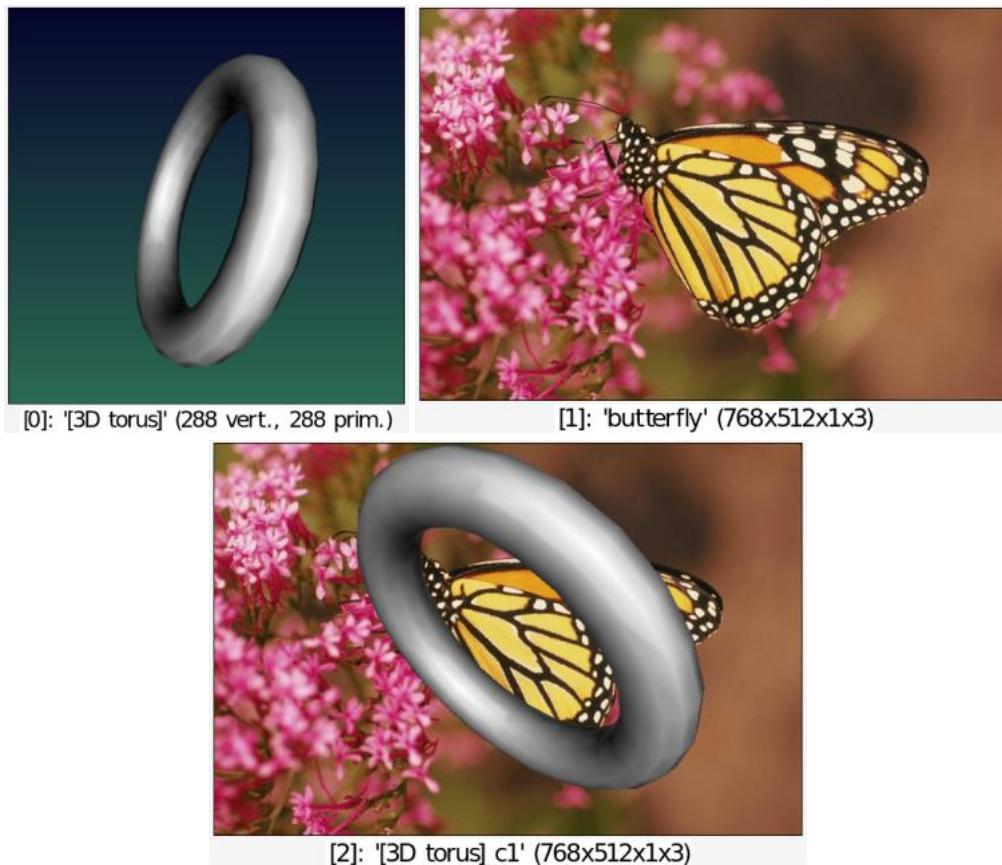
- **Example #1**

```
$ gmic torus3d 100,20 rotate3d 1,1,0,60 snapshot3d 400,1.2,128,64,32
```



- **Example #2**

```
$ gmic torus3d 100,20 rotate3d 1,1,0,60 sample ? +snapshot3d[0]  
[1],1.2
```



---

## solarize

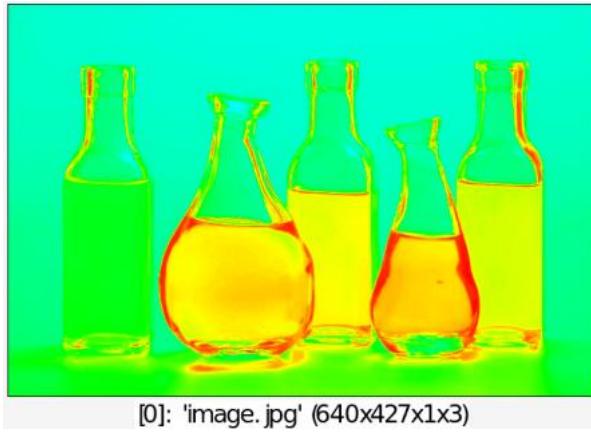
**No arguments**

**Description:**

Solarize selected images.

**Example of use:**

```
$ gmic image.jpg solarize
```



---

## solidify

### Arguments:

- `_smoothness[%]>=0, _diffusion_type={ 0=isotropic | 1=Delaunay-guided | 2=edge-oriented }, _diffusion_iter>=0`

### Description:

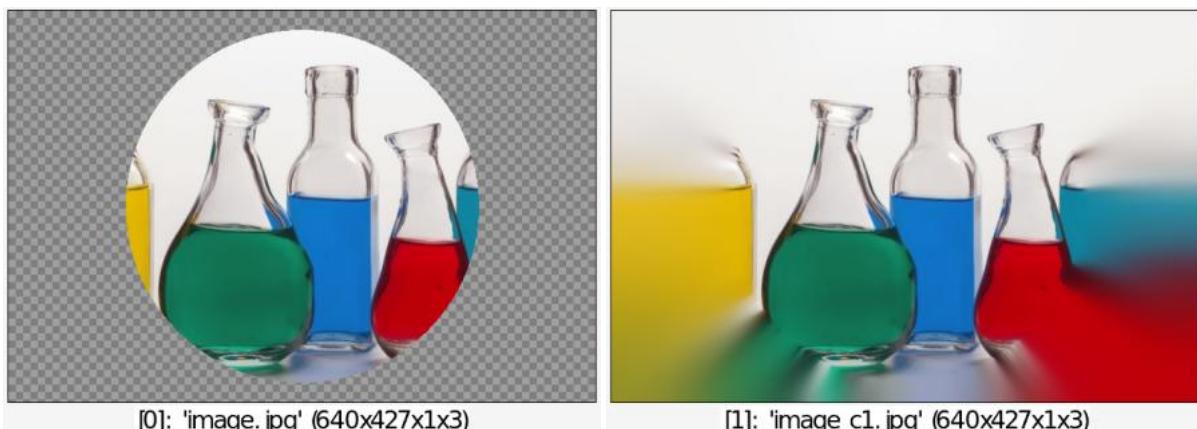
Solidify selected transparent images.

### Default values:

`smoothness=75%`, `diffusion_type=1` and `diffusion_iter=20`.

### Example of use:

```
$ gmic image.jpg 100%,100% circle[-1] 50%,50%,25%,1,255 append c  
+solidify , display_rgba
```



---

## solve

[Built-in command](#)

### Arguments:

- [image]

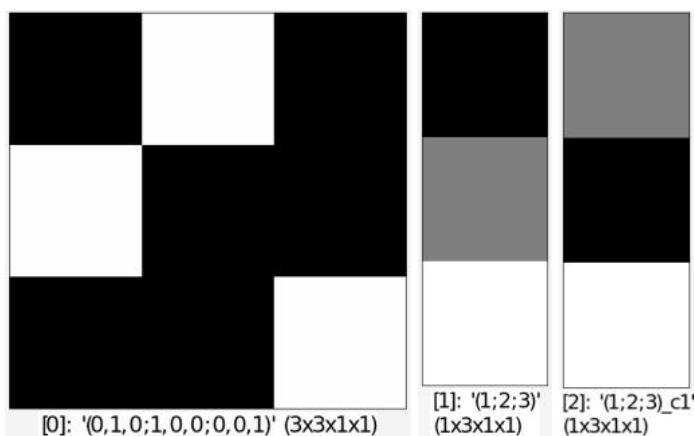
## Description:

Solve linear system  $AX = B$  for selected B-matrices and specified A-matrix.

If the system is under- or over-determined, the least squares solution is returned (using SVD-based solver).

## Example of use:

```
$ gmic (0,1,0;1,0,0;0,0,1) (1;2;3) +solve[-1] [-2]
```



## solve\_poisson

### Arguments:

- "laplacian\_command", \_nb\_iterations>=0, \_time\_step>0, \_nb\_scales>=0

## Description:

Solve Poisson equation so that applying `laplacian[n]` is close to the result of `laplacian_command[n]`.

Solving is performed using a multi-scale gradient descent algorithm.  
If `nb_scales=0`, the number of scales is automatically determined.

## Default values:

`nb_iterations=60`, `dt=5` and `nb_scales=0`.

## Example of use:

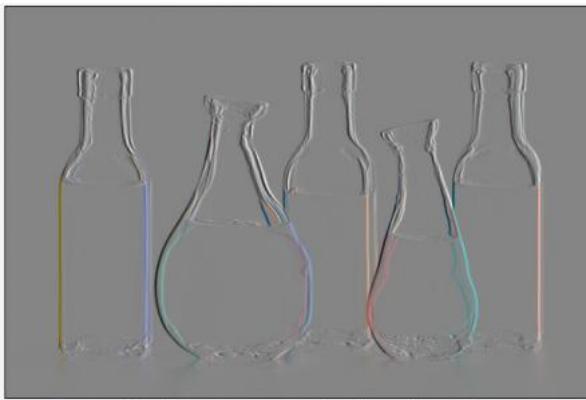
```
$ gmic image.jpg command "foo : gradient x" +solve_poisson foo
+foo[0] +laplacian[1]
```



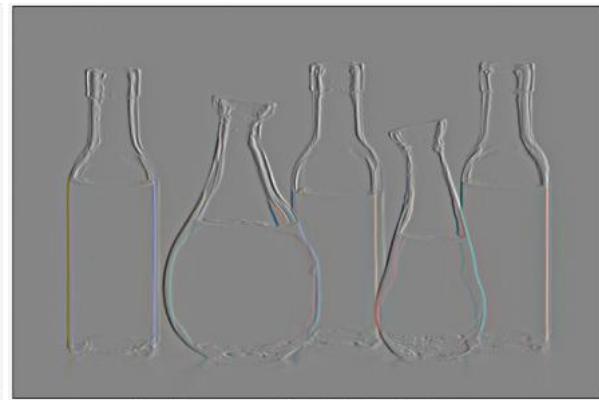
[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c2.jpg' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)



[3]: 'image\_c3.jpg' (640x427x1x3)

## sort

Built-in command

### Arguments:

- `_ordering={ + | - },_axis={ x | y | z | c }`

### Description:

Sort pixel values of selected images.

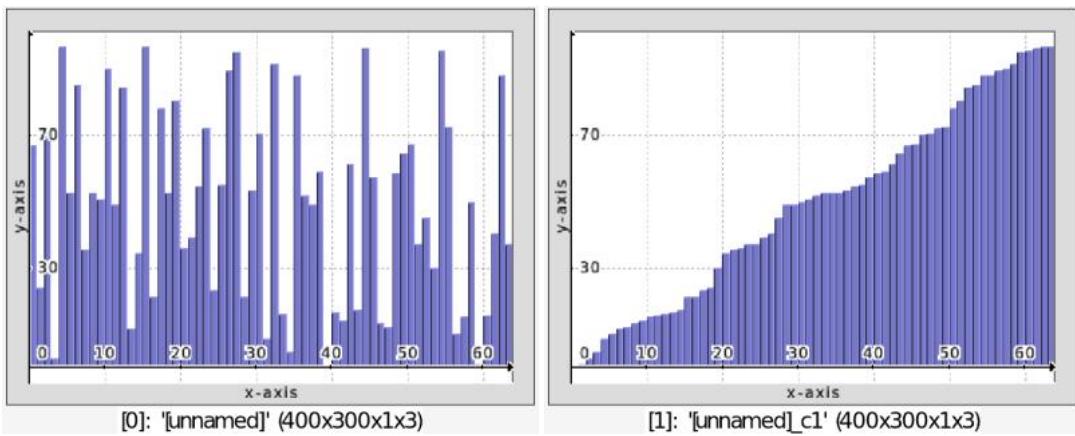
If `axis` is specified, the sorting is done according to the data of the first column/row/slice/channel of selected images.

### Default values:

`ordering=+` and `axis=(undefined)`.

### Example of use:

```
$ gmic 64 rand 0,100 +sort display_graph 400,300,3
```



## sort\_list

### Arguments:

- `_ordering={ + | - },_criterion`

### Description:

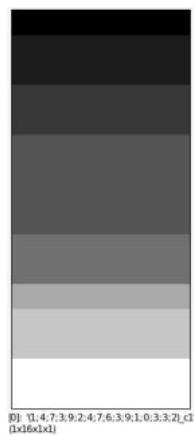
Sort list of selected images according to the specified image criterion.

### Default values:

`ordering=+, criterion=i.`

### Example of use:

```
$ gmic (1;4;7;3;9;2;4;7;6;3;9;1;0;3;3;2) split y sort_list +,i append  
y
```



## spec13d

Built-in command

### Arguments:

- `value>=0`

## Description:

Set lightness of 3D specular light.

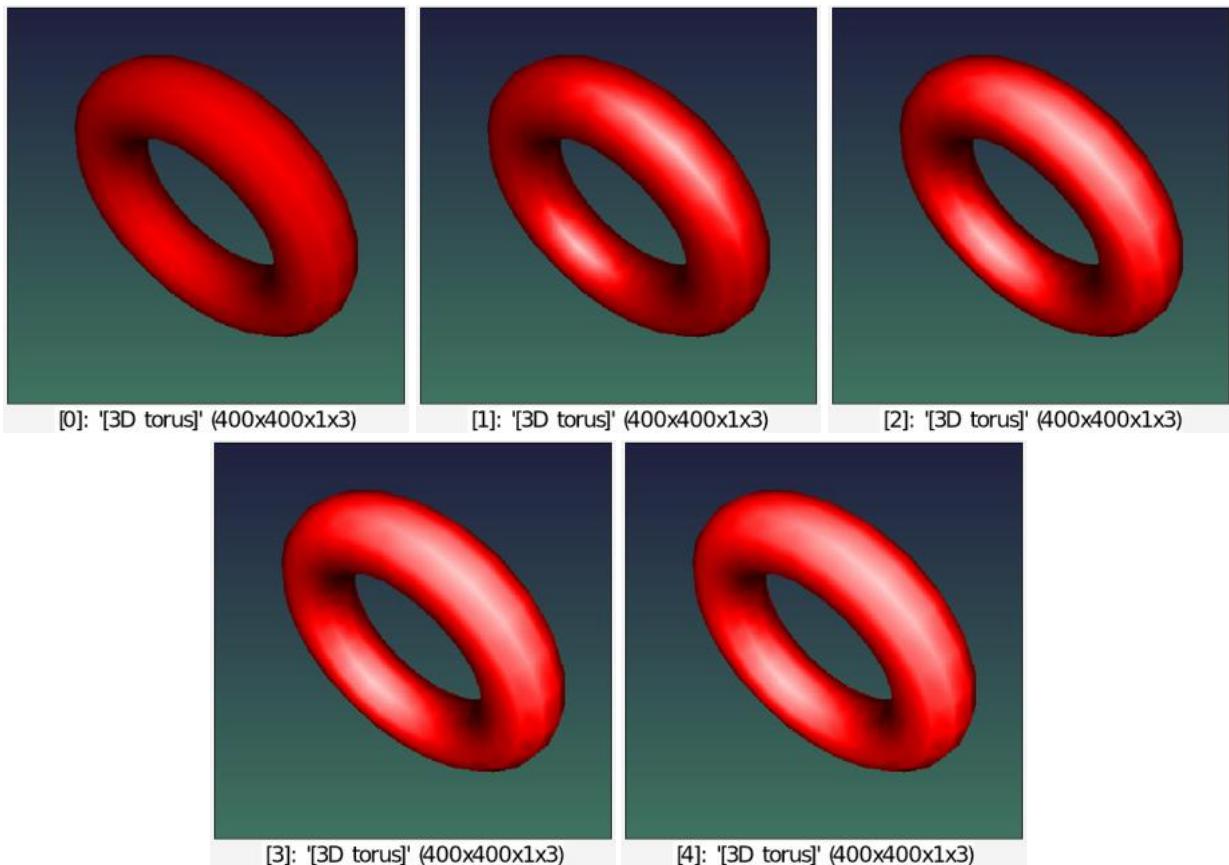
(equivalent to shortcut command `sl3d`).

## Default values:

`value=0.15`.

## Example of use:

```
$ gmic (0,0.3,0.6,0.9,1.2) repeat w torus3d 100,30 rotate3d[-1]
1,1,0,60 color3d[-1] 255,0,0 spec3d {0,@$>} snapshot3d[-1] 400 done
remove[0]
```



## specs3d

Built-in command

## Arguments:

- `value>=0`

## Description:

Set shininess of 3D specular light.

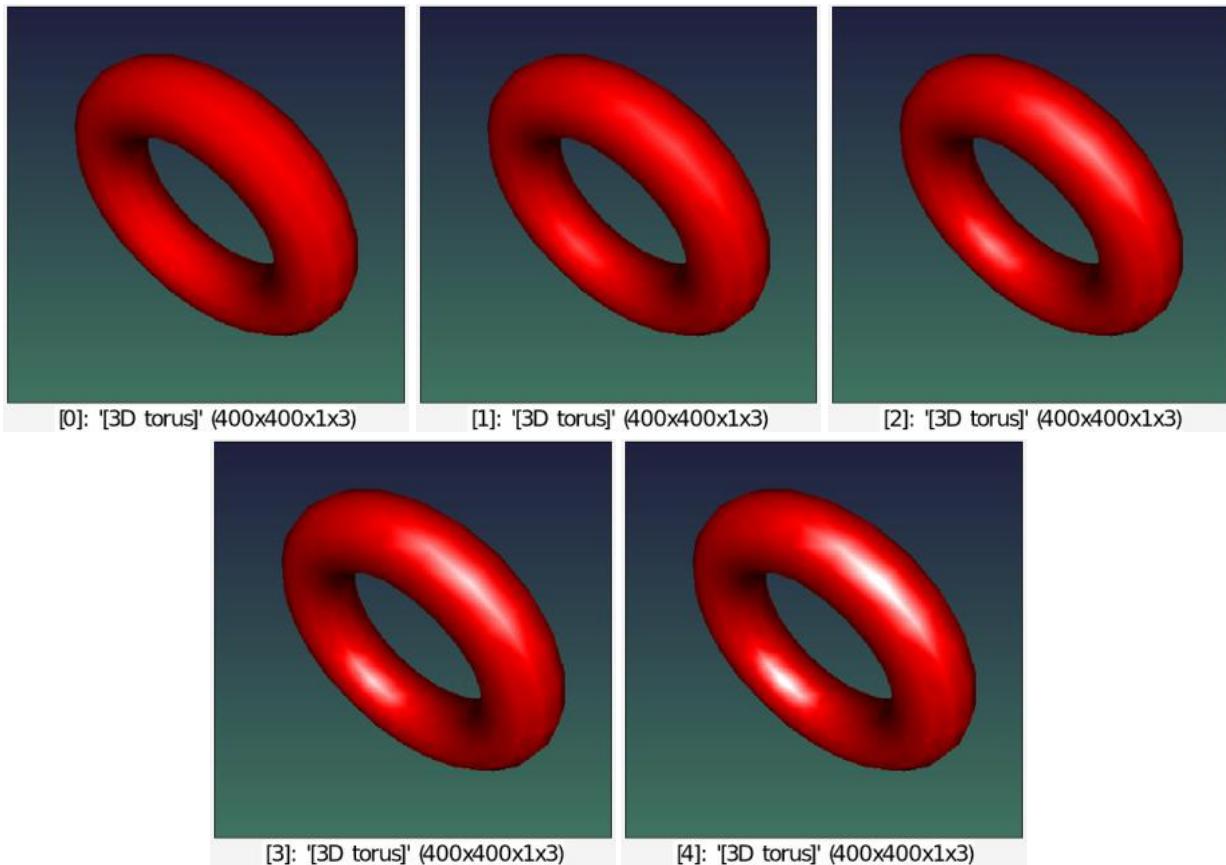
(equivalent to shortcut command `ss3d`).

## Default values:

`value=0.8`.

## Example of use:

```
$ gmic (0,0.3,0.6,0.9,1.2) repeat w torus3d 100,30 rotate3d[-1]  
1,1,0,60 color3d[-1] 255,0,0 specs3d {0,@$>} snapshot3d[-1] 400 done  
remove[0]
```



---

## sphere3d

Built-in command

### Arguments:

- `radius,_nb_recursions>=0`

### Description:

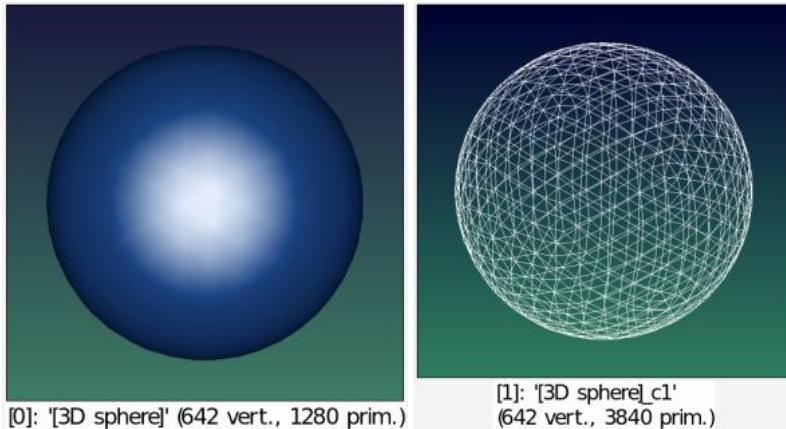
Input 3D sphere at (0,0,0), with specified geometry.

## Default values:

`nb_recursions=3`.

## Example of use:

```
$ gmic sphere3d 100 +primitives3d 1 color3d[-2] ${-rgb}
```



---

## spherical3d

### Arguments:

- `_nb_azimuth>=3,_nb_zenith>=3,_radius_function(phi,theta)`

### Description:

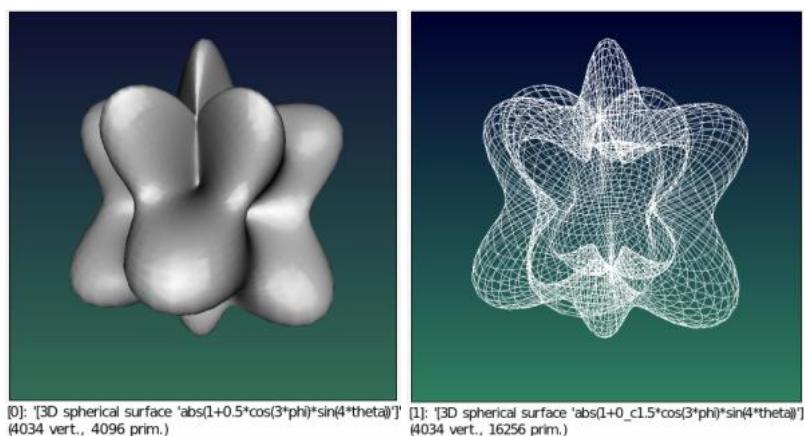
Input 3D spherical object at (0,0,0), with specified geometry.

### Default values:

`nb_zenith=nb_azimut=64` and  
`radius_function="abs(1+0.5*cos(3*phi)*sin(4*theta))".`

### Example of use:

```
$ gmic spherical3d 64 +primitives3d 1
```



---

## spherize

## Arguments:

- `_radius[%]>=0,_strength,_smoothness[%]>=0,_center_x[%],_center_y[%],_ratio_x`

## Description:

Apply spherize effect on selected images.

## Default values:

`radius=50%, strength=1, smoothness=0, center_x=center_y=50%, ratio_x/y=1,`  
`angle=0` and `interpolation=1`.

## Example of use:

```
$ gmic image.jpg grid 5%,5%,0,0,0.6,255 spherize ,
```



---

## spiralbw

## Arguments:

- `width>0,_height>0,_is_2dcoords={ 0 | 1 }`

## Description:

Input a 2D rectangular spiral image with specified size.

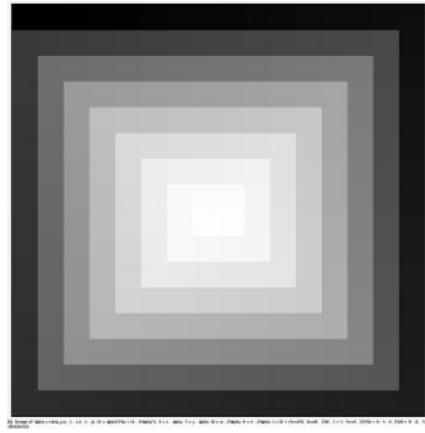
## Default values:

`height=width` and `is_2dcoords=0`.

## Examples of use:

- Example #1

```
$ gmic spiralbw 16
```

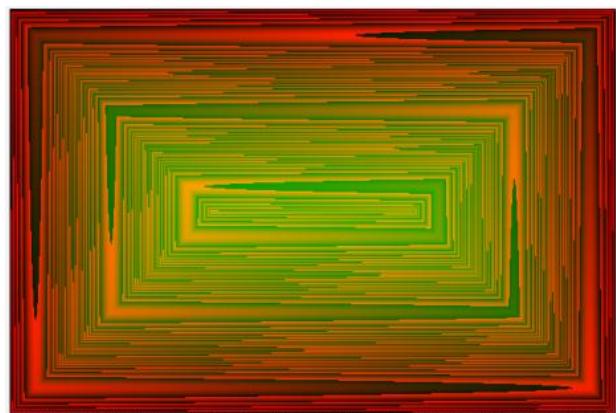


- **Example #2**

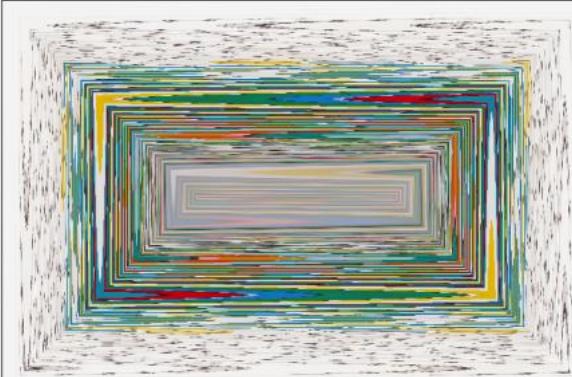
```
$ gmic image.jpg spiralbw {[w,h]},1 +warp[0] [1],0 +warp[2] [1],2
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)



[3]: 'image\_c2.jpg' (640x427x1x3)

---

## spline

### Arguments:

- `x0[%],y0[%],u0[%],v0[%],x1[%],y1[%],u1[%],v1[%],_opacity,_color1,...`

### Description:

Draw specified colored spline curve on selected images (cubic hermite spline).

## Default values:

`opacity=1` and `color1=0`.

## Example of use:

```
$ gmic image.jpg repeat 30 spline {u(100)}%,{u(100)}%,{u(-600,600)},  
{u(-600,600)},{u(100)}%,{u(100)}%,{u(-600,600)},{u(-600,600)},0.6,255  
done
```



---

## spline3d

### Arguments:

- `x0[%],y0[%],z0[%],u0[%],v0[%],w0[%],x1[%],y1[%],z1[%],u1[%],v1[%],w1[%],_nb_`

### Description:

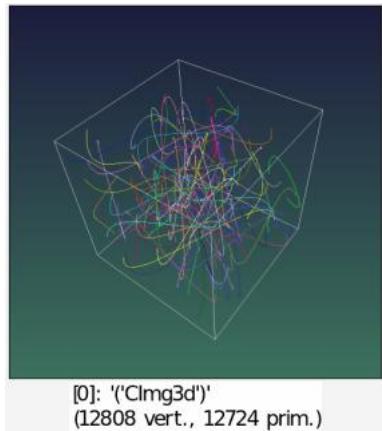
Input 3D spline with specified geometry.

## Default values:

`nb_vertices=128`.

## Example of use:

```
$ gmic repeat 100 spline3d {u},{u},{u},{u},{u},{u},{u},{u},{u},{u},  
{u},{u},128 color3d[-1] ${-rgb} done box3d 1 primitives3d[-1] 1 add3d
```



# split

Built-in command

## Arguments:

- `{ x | y | z | c }...{ x | y | z | c },_split_mode` or
- `keep_splitting_values={ + | - },_{ x | y | z | c }...{ x | y | z | c }`,  
`_value1,_value2,...` or
- `(no arg)`

## Description:

Split selected images along specified axes, or regarding to a sequence of scalar values (optionally along specified axes too).

(equivalent to shortcut command `s`).

`split_mode` can be `{ 0=split according to constant values | >0=split in N parts | <0=split in parts of size -N }`.

## Default values:

`split_mode=-1`.

## Examples of use:

- **Example #1**

```
$ gmic image.jpg split c
```



[0]: 'image.jpg' (640x427x1x1)



[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c1.jpg' (640x427x1x1)

- **Example #2**

```
$ gmic image.jpg split y,3
```



[0]: 'image.jpg' (640x143x1x3)



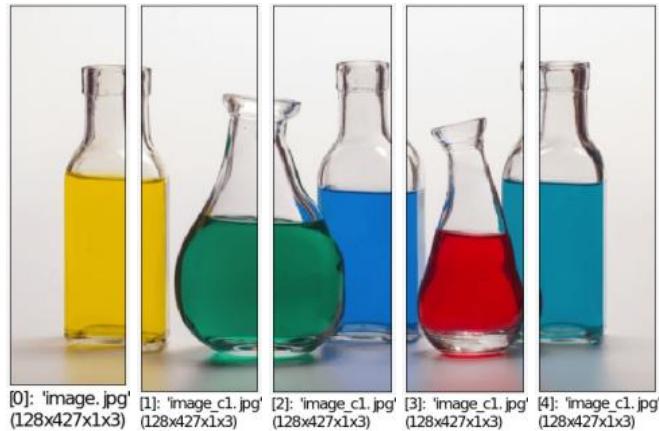
[1]: 'image\_c1.jpg' (640x142x1x3)



[2]: 'image\_c1.jpg' (640x142x1x3)

- **Example #3**

```
$ gmic image.jpg split x,-128
```



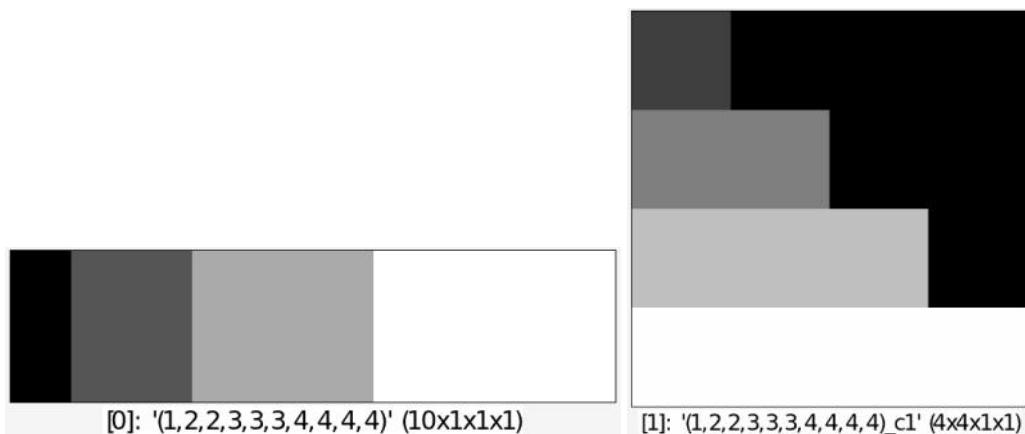
- **Example #4**

```
$ gmic 1,20,1,1,"1,2,3,4" +split -,2,3 append[1--1] y
```



- **Example #5**

```
$ gmic (1,2,2,3,3,3,4,4,4,4) +split x,0 append[1--1] y
```



## split3d

[Built-in command](#)

### Arguments:

- `_full_split={ 0 | 1 }`

## Description:

Split selected 3D objects into feature vectors :

- If `full_split==0`, { `header, sizes, vertices, primitives, colors, opacities` }.
- If `full_split==1`, { `header, sizes, vertices, p0,...,pP, c0,...,cP, o0,...,oP` }.

(equivalent to shortcut command `s3d`).

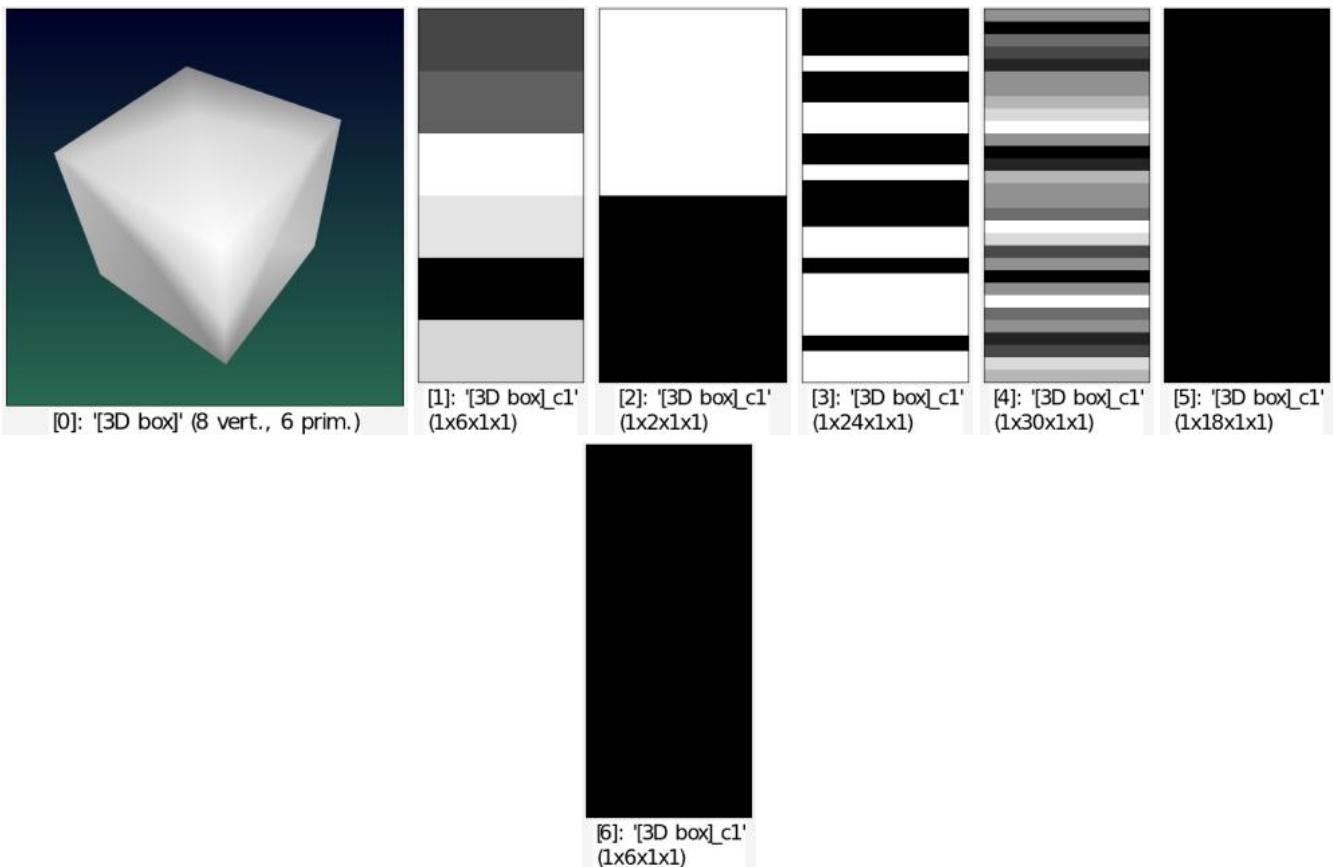
To recreate the 3D object, append all produced images along the y-axis.

## Default values:

`full_split=0`.

## Example of use:

```
$ gmic box3d 100 +split3d
```



## split\_colors

### Arguments:

- `_tolerance>=0, _max_nb_outputs>0, _min_area>0`

## Description:

Split selected images as several image containing a single color.

One selected image can be split as at most `max_nb_outputs` images.

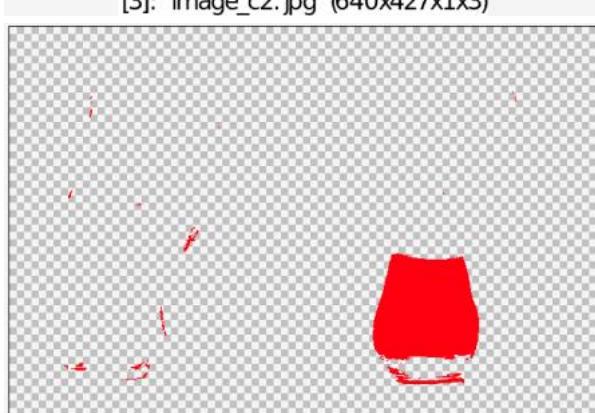
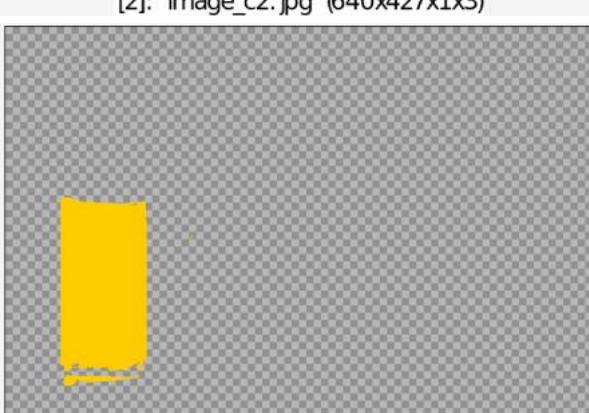
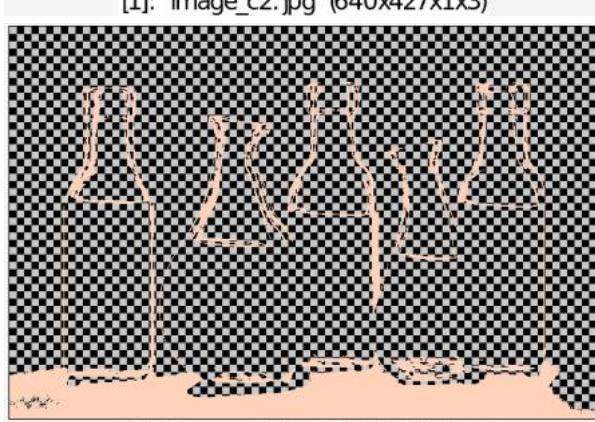
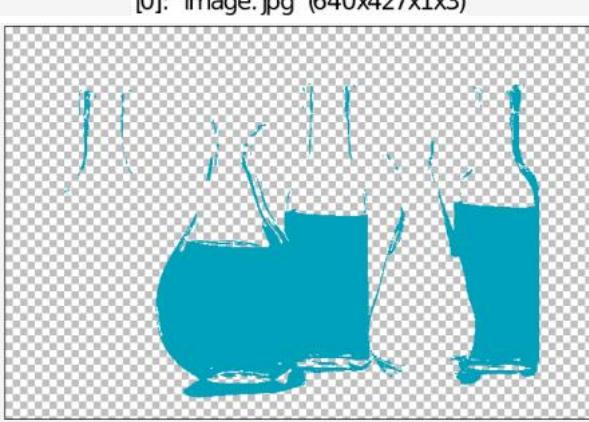
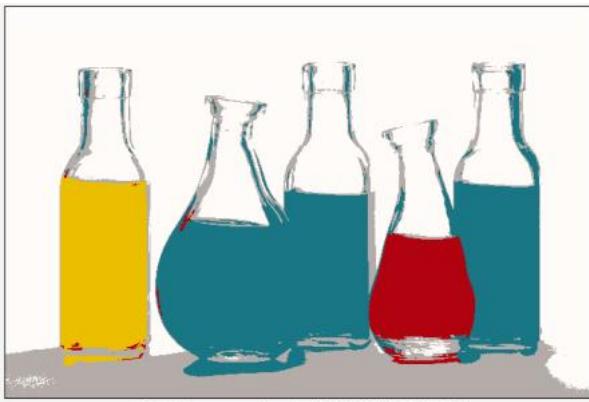
Output images are sorted by decreasing area of extracted color regions and have an additional alpha-channel.

## Default values:

`tolerance=0`, `max_nb_outputs=256` and `min_area=8`.

## Example of use:

```
$ gmic image.jpg quantize 5 +split_colors , display_rgba
```



# split\_details

## Arguments:

- `_nb_scales>0, _base_scale[%]>=0, _detail_scale[%]>=0`

## Description:

Split selected images into `nb_scales` detail scales.

If 'base\_scale'<`detail_scale`'0, the image decomposition is done with 'a trous' wavelets. Otherwise, it uses laplacian pyramids with linear standard deviations.

## Default values:

`nb_scales=4`, `base_scale=0` and `detail_scale=0`.

## Example of use:

```
$ gmic image.jpg split_details ,
```



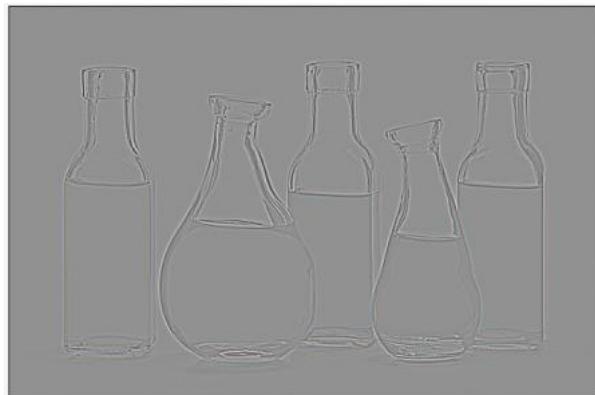
[0]: 'image\_c3.jpg' (640x427x1x3)



[1]: 'image\_c2.jpg' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)



[3]: 'image.jpg' (640x427x1x3)

# split\_freq

## Arguments:

- `smoothness>0[%]`

## Description:

Split selected images into low and high frequency parts.

## Example of use:

```
$ gmic image.jpg split_freq 2%
```



[0]: 'image\_c1.jpg' (640x427x1x3)



[1]: 'image.jpg' (640x427x1x3)

---

## split\_opacity

### No arguments

## Description:

Split color and opacity parts of selected images.

---

## split\_tiles

### Arguments:

- `M!=0, _N!=0, _is_homogeneous={ 0 | 1 }`

## Description:

Split selected images as a MxN array of tiles.

If M or N is negative, it stands for the tile size instead.

## Default values:

`N=M` and `is_homogeneous=0`.

## Example of use:

```
$ gmic image.jpg +local split_tiles 5,4 blur 3,0 sharpen 700  
append_tiles 4,5 endlocal
```



---

## sponge

### Arguments:

- `_size>0`

### Description:

Apply sponge effect on selected images.

### Default values:

`size=13`.

### Example of use:

```
$ gmic image.jpg sponge ,
```



---

## spread

## Arguments:

- `_dx>=0, _dy>=0, _dz>=0`

## Description:

Spread pixel values of selected images randomly along x,y and z.

## Default values:

`dx=3`, `dy=dx` and `dz=0`.

## Example of use:

```
$ gmic image.jpg +spread 3
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

---

## sprite3d

### No arguments

## Description:

Convert selected images as 3D sprites.

Selected images with alpha channels are managed.

## Example of use:

```
$ gmic image.jpg sprite3d
```



---

## sprites3d

### Arguments:

- `[sprite],_sprite_has_alpha_channel={ 0 | 1 }`

### Description:

Convert selected 3D objects as a sprite cloud.

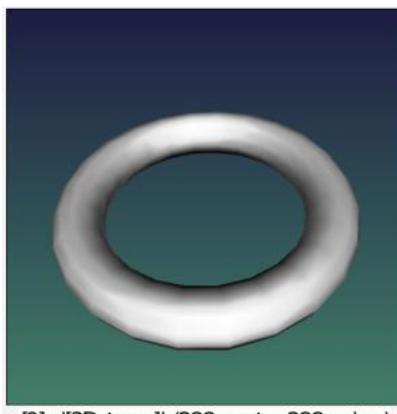
Set `sprite_has_alpha_channel` to 1 to make the last channel of the selected sprite be a transparency mask.

### Default values:

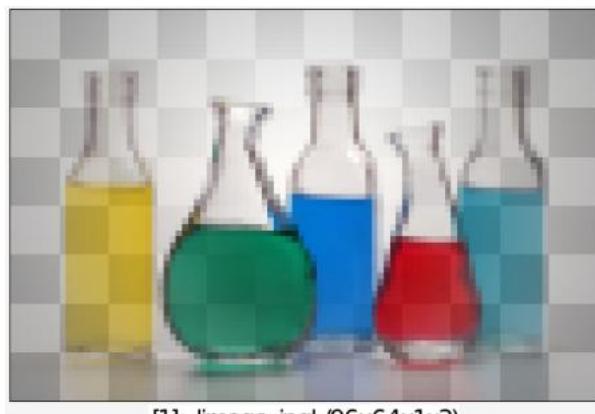
`mask_has_alpha_channel=0`.

### Example of use:

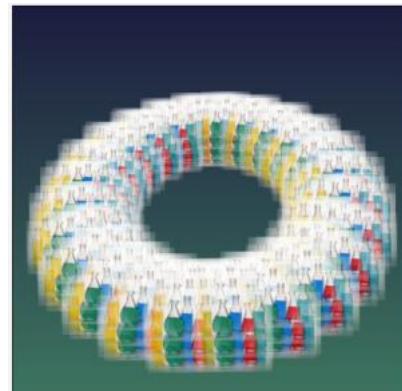
```
$ gmic torus3d 100,20 image.jpg resize2dy[-1] 64 100%,100%
gaussian[-1] 30%,30% *[-1] 255 append[-2,-1] c +sprites3d[0] [1],1
display_rgba[-2]
```



[0]: '[3D torus]' (288 vert., 288 prim.)



[1]: 'image.jpg' (96x64x1x3)



[2]: '[3D torus]\_c1' (288 vert., 288 prim.)

## sqr

Built-in command

No arguments

Description:

Compute the pointwise square function of selected images.

Examples of use:

- Example #1

```
$ gmic image.jpg +sqr
```



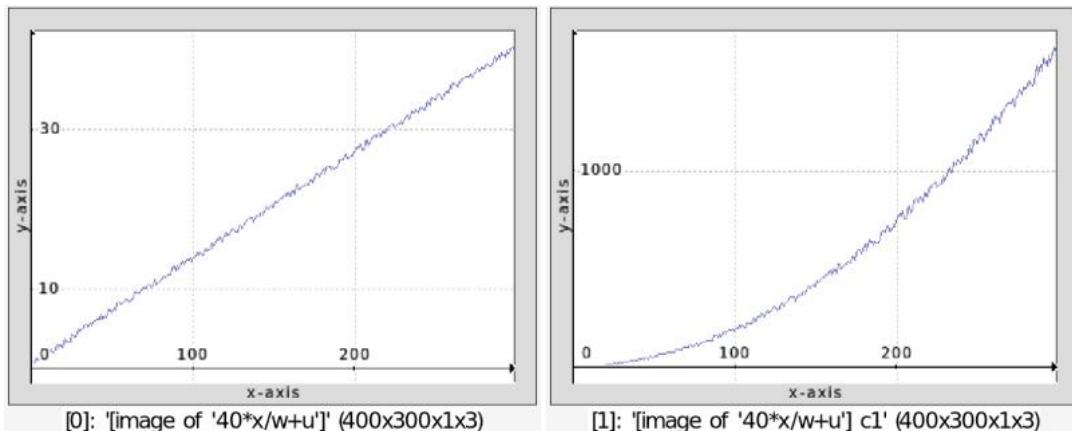
[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

- **Example #2**

```
$ gmic 300,1,1,1,'40*x/w+u' +sqrt display_graph 400,300
```



---

## sqrt

Built-in command

No arguments

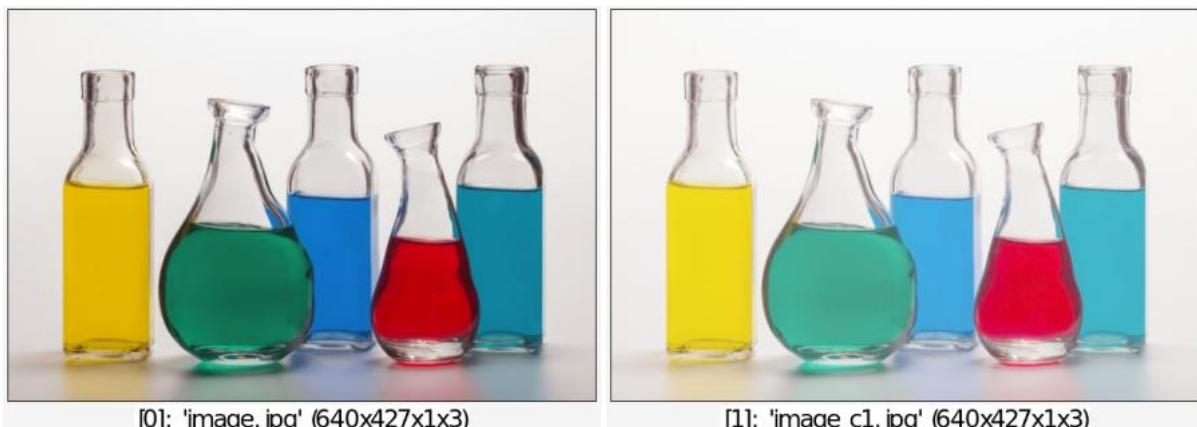
**Description:**

Compute the pointwise square root of selected images.

**Examples of use:**

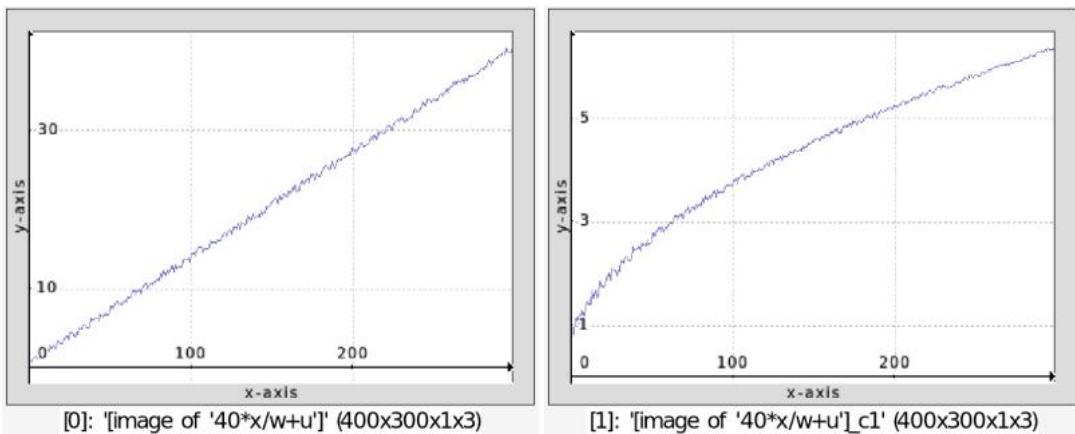
- **Example #1**

```
$ gmic image.jpg +sqrt
```



- **Example #2**

```
$ gmic 300,1,1,1,'40*x/w+u' +sqrt display_graph 400,300
```



## srand

**Built-in command**

### Arguments:

- `value` or
- `(no arg)`

### Description:

Set random generator seed.

If no argument is specified, a random value is used as the random generator seed.

## srgb2lab

### Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

### Description:

Convert color representation of selected images from sRGB to Lab.

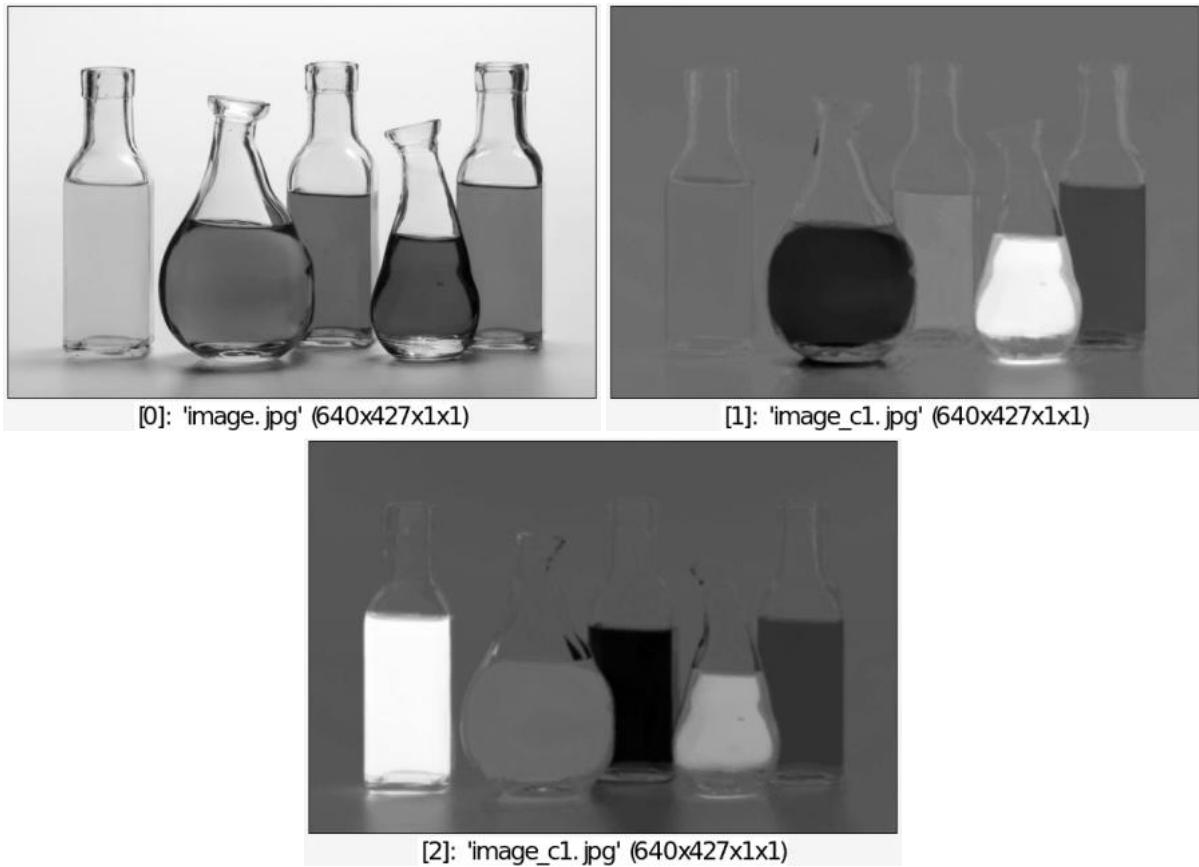
### Default values:

`illuminant=2`.

### Examples of use:

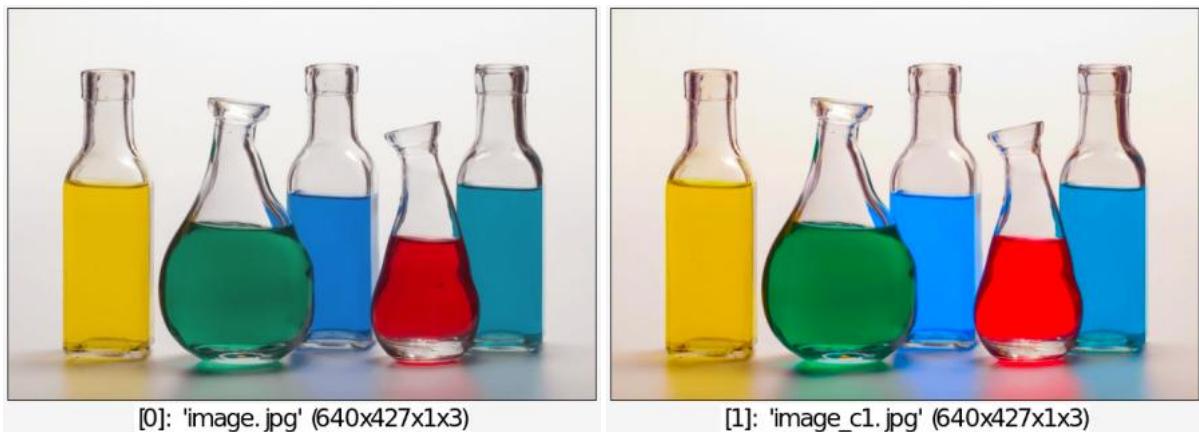
- **Example #1**

```
$ gmic image.jpg srgb2lab split c
```



- **Example #2**

```
$ gmic image.jpg srgb2lab +split c mul[-2,-1] 2.5 append[-3--1] c  
lab2srgb
```



## srgb2lab8

### Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

### Description:

Convert color representation of selected images from sRGB to Lab8.

## Default values:

`illuminant=2`.

---

# srgb2rgb

## No arguments

## Description:

Convert color representation of selected images from sRGB to linear RGB.

---

# ssd\_patch

## Arguments:

- `[patch],_use_fourier={ 0 | 1 },_boundary_conditions`

## Description:

Compute fields of SSD between selected images and specified patch.

Argument `boundary_conditions` is valid only when `use_fourier=0`.

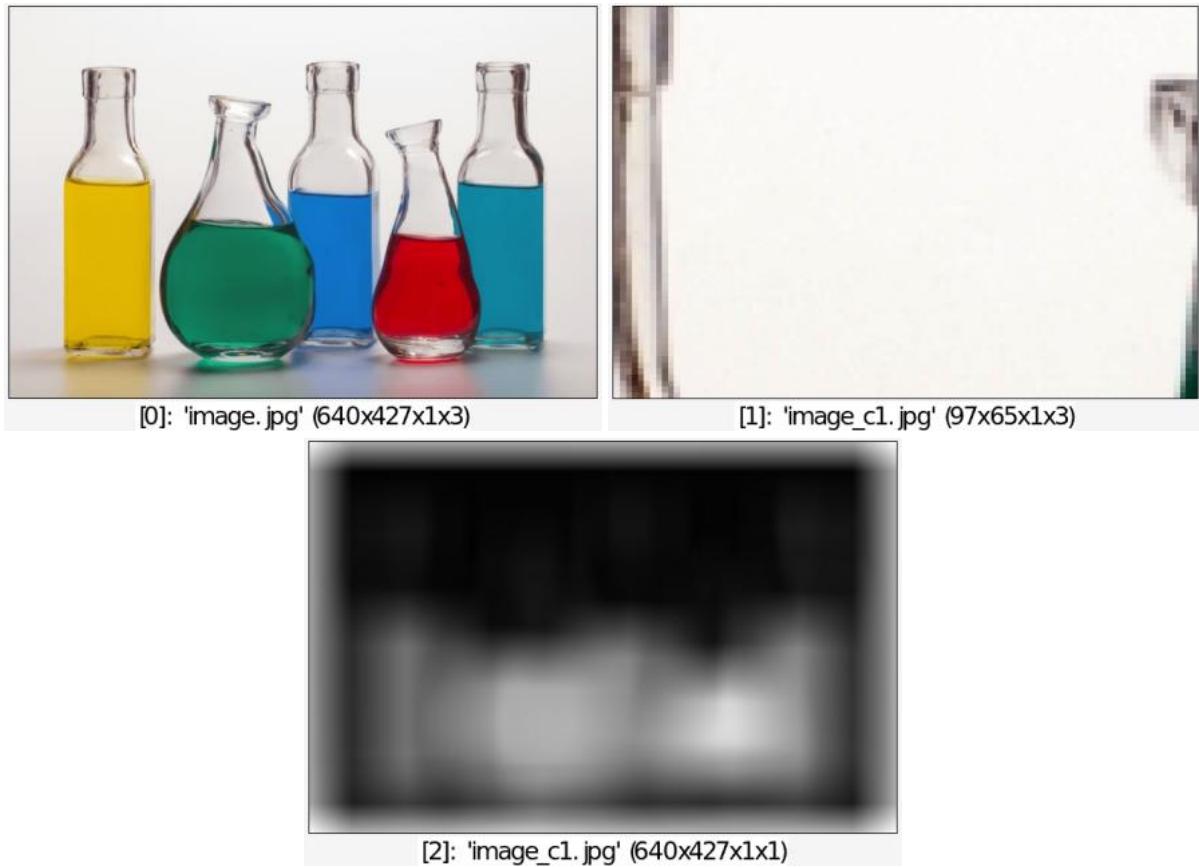
`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

## Default values:

`use_fourier=0` and `boundary_conditions=0`.

## Example of use:

```
$ gmic image.jpg +crop 20%,20%,35%,35% +ssd_patch[0] [1],0,0
```



---

## stained\_glass

### Arguments:

- `_edges[%]>=0, shading>=0, is_thin_separators={ 0 | 1 }`

### Description:

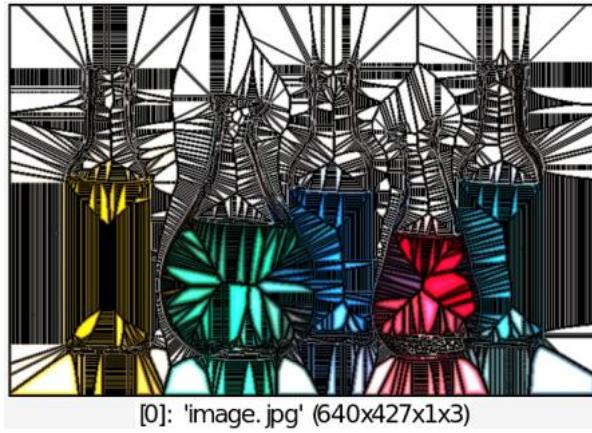
Generate stained glass from selected images.

### Default values:

`edges=40%`, `shading=0.2` and `is_precise=0`.

### Example of use:

```
$ gmic image.jpg stained_glass 20%,1 cut 0,20
```



---

## star3d

### Arguments:

- `_nb_branches>0, 0<=_thickness<=1`

### Description:

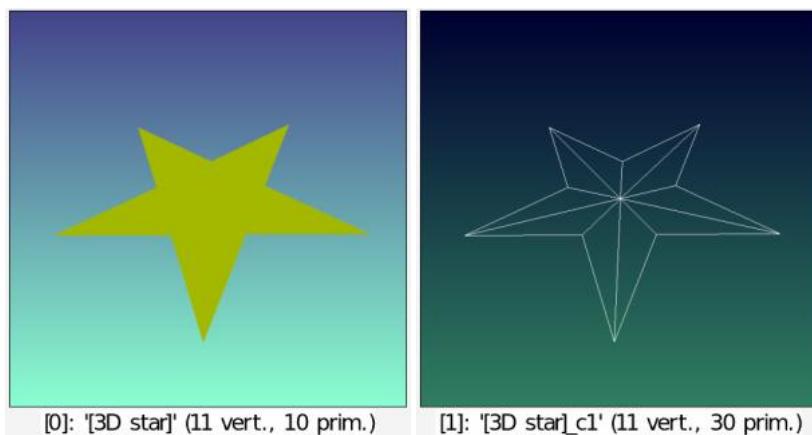
Input 3D star at position `(0,0,0)`, with specified geometry.

### Default values:

`nb_branches=5` and `thickness=0.38`.

### Example of use:

```
$ gmic star3d , +primitives3d 1 color3d[-2] ${-rgb}
```



---

## stars

### Arguments:

- `_density[%]>=0, _depth>=0, _size>0, _nb_branches>=1, 0<=_thickness<=1, _smoothnes`

## Description:

Add random stars to selected images.

## Default values:

`density=10%`, `depth=1`, `size=32`, `nb_branches=5`, `thickness=0.38`, `smoothness=0.5`, `R=G=B=200` and `opacity=1`.

## Example of use:

```
$ gmic image.jpg stars ,
```



---

## status

Built-in command

## Arguments:

- `status_string`

## Description:

Set the current status. Used to define a returning value from a function.

(equivalent to shortcut command `u`).

## Example of use:

```
$ gmic image.jpg command "foo : u0=Dark u1=Bright status ${u{ia}>=128}" text_outline ${-foo},2,2,23,2,1,255
```



---

## std\_noise

**No arguments**

**Description:**

Return the estimated noise standard deviation of the last selected image.

---

## stencil

**Arguments:**

- `_radius[%]>=0, _smoothness>=0, _iterations>=0`

**Description:**

Apply stencil filter on selected images.

**Default values:**

`radius=3, smoothness=1` and `iterations=8`.

**Example of use:**

```
$ gmic image.jpg +norm stencil. 2,1,4 +mul rm[0]
```



---

## stencilbw

### Arguments:

- `_edges>=0, _smoothness>=0`

### Description:

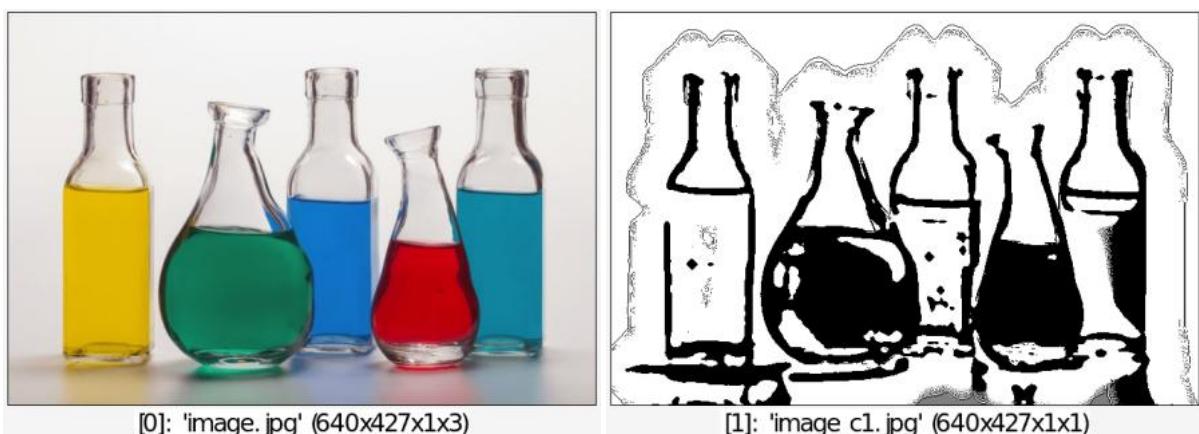
Apply B&W stencil effect on selected images.

### Default values:

`edges=15` and `smoothness=10`.

### Example of use:

```
$ gmic image.jpg +stencilbw 40,4
```



---

## store

Built-in command

### Arguments:

- `_is_compressed={ 0 | 1 },variable_name1,_variable_name2,...`

## Description:

Store selected images into one or several named variables.

Selected images are transferred to the variables, and are so removed from the image list.  
(except if the prepended variant of the command `+store[selection]` is used).

If a single variable name is specified, all images of the selection are assigned  
to the named variable. Otherwise, there must be as many variable names as images  
in the selection, and each selected image is assigned to each specified named variable.  
Use command `input $variable_name` to bring the stored images back in the list.

## Default values:

`is_compressed=0`.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic sample eagle,earth store img1,img2 input $img2 $img1
```



---

## str

### Arguments:

- `string`

## Description:

Print specified string into its binary, octal, decimal and hexadecimal representations.

---

## str2hex

### Arguments:

- `"string"`

## Description:

Convert specified string argument into a sequence of hexadecimal values (returned as a string).

## See also:

[hex2str](#).

## Example of use:

```
$ gmic hex={"str2hex \"Hello my friends\""} echo $hex
```

```
[gmic]-0./ Start G'MIC interpreter.  
[gmic]-0./ 48656c6c6f206d7920667269656e6473  
[gmic]-0./ End G'MIC interpreter.
```

# strcapitalize

## Arguments:

- `string`

## Description:

Capitalize specified string.

# strcasevar

## Arguments:

- `"string"`

## Description:

Return a simplified version of the specified string, that can be used as a variable name.

(version that keeps original case of specified string, no longer than 128 chars).

# strcontains

## Arguments:

- `string1, string2`

## Description:

Return 1 if the first string contains the second one.

---

## streamline3d

Built-in command

## Arguments:

- `x[%],y[%],z[%],_L>=0,_dl>0,_interpolation,_is_backward={ 0 | 1 },_is_oriented={ 0 | 1 }` or
- `'formula',x,y,z,_L>=0,_dl>0,_interpolation,_is_backward={ 0 | 1 },_is_oriented={ 0 | 1 }`

## Description:

Extract 3D streamlines from selected vector fields or from specified formula.

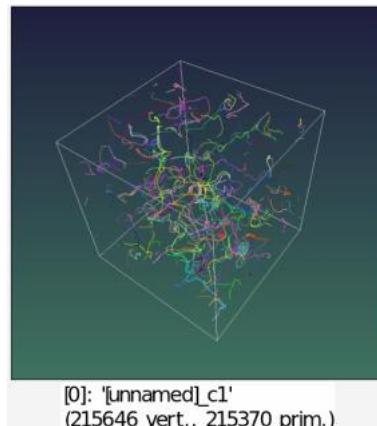
`interpolation` can be `{ 0=nearest integer | 1=1st-order | 2=2nd-order | 3=4th-order }`.

## Default values:

`dl=0.1`, `interpolation=2`, `is_backward=0` and `is_oriented=0`.

## Example of use:

```
$ gmic 100,100,100,3 rand -10,10 blur 3 repeat 300 +streamline3d[0]
{u(100)},{u(100)},{u(100)},1000,1,1 color3d[-1] ${-rgb} done
remove[0] box3d 100 primitives3d[-1] 1 add3d
```



---

## stripes\_y

## Arguments:

- `_frequency>=0`

## Description:

Add vertical stripes to selected images.

## Default values:

`frequency=10`.

## Example of use:

```
$ gmic image.jpg +stripes_y ,
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

## strlen

## Arguments:

- `string1`

## Description:

Return the length of specified string argument.

## strlowercase

## Arguments:

- `string`

## Description:

Return a lower-case version of the specified string.

---

## strreplace

### Arguments:

- `string,search,replace`

### Description:

Search and replace substrings in an input string.

---

## structuretensors

Built-in command

### Arguments:

- `_scheme={ 0=centered | 1=forward/backward }`

### Description:

Compute the structure tensor field of selected images.

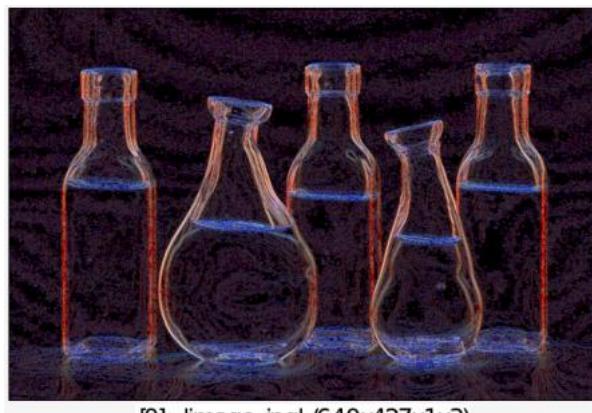
### Default values:

`scheme=1`.

This command has a [tutorial page](#).

### Example of use:

```
$ gmic image.jpg structuretensors abs pow 0.2
```



[0]: 'image.jpg' (640x427x1x3)

---

## strupercase

## **Arguments:**

- `string`

## **Description:**

Return an upper-case version of the specified string.

---

# strvar

## **Arguments:**

- `"string"`

## **Description:**

Return a simplified version of the specified string, that can be used as a variable name.

(version that creates a lowercase result, no longer than 128 chars).

---

# strver

## **Arguments:**

- `_version`

## **Description:**

Return the specified version number of the G'MIC interpreter, as a string.

## **Default values:**

`version=$_version`.

---

# stylize

## **Arguments:**

- `[style_image],_fidelity_finest,_fidelity_coarsest,_fidelity_smoothness_fines`

## **Description:**

Transfer colors and textures from specified style image to selected images, using a multi-scale patch-matching algorithm.

If instant display window[0] is opened, the steps of the image synthesis are displayed on it.

`init_type` can be `{ 0=best-match | 1=identity | 2=randomized }`.

## Default values:

```
fidelity_finest=0.5, fidelity_coarsest=2, fidelity_smoothness_finest=3,  
fidelity_smoothness_coarsest=0.5, fidelity_chroma=0.1, init_type=0,  
init_resolution=16, init_max_gradient=0, patchsize_analysis=5,  
patchsize_synthesis=5, patchsize_synthesis_final=5, nb_matches_fine=2,  
nb_matchesc_coarse=30, penalize_repetitions=2, matching_precision=2,  
scale_factor=1.85, skip_fine_scales=0 and 'image_matching_command'='s c,-3  
transfer_pca[0] [2] b[0,2] xy,0.7 n[0,2] 0,255 n[1,2] 0,200 a[0,1] c a[1,2] c"'.
```

---

# sub

Built-in command

## Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

## Description:

Subtract specified value, image or mathematical expression to selected images, or compute the pointwise difference of selected images.

(equivalent to shortcut command `-`).

## Examples of use:

- **Example #1**

```
$ gmic image.jpg +sub 30% cut 0,255
```



- **Example #2**

```
$ gmic image.jpg +mirror x sub[-1] [0]
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

- **Example #3**

```
$ gmic image.jpg sub 'i(w/2+0.9*(x-w/2),y)'
```



[0]: 'image.jpg' (640x427x1x3)

- **Example #4**

```
$ gmic image.jpg +mirror x sub
```



[0]: 'image.jpg' (640x427x1x3)

## Arguments:

- `tx,_ty,_tz`

## Description:

Shift selected 3D objects with the opposite of specified displacement vector.

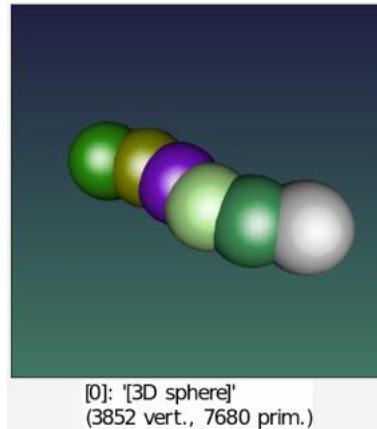
(*equivalent to shortcut command* `3d`).

## Default values:

`ty=tz=0`.

## Example of use:

```
$ gmic sphere3d 10 repeat 5 +sub3d[-1] 10,{u(-10,10)},0 color3d[-1]
 ${-rgb} done add3d
```



---

## sub\_alpha

### Arguments:

- `[base_image],_opacity_gain>=1`

## Description:

Compute the minimal alpha-channel difference (opposite of alpha blending) between the selected images

and the specified base image.

The alpha difference A-B is defined as the image having minimal opacity, such that  $\text{alpha\_blend}(B,A-B) = A$ .

## Default values:

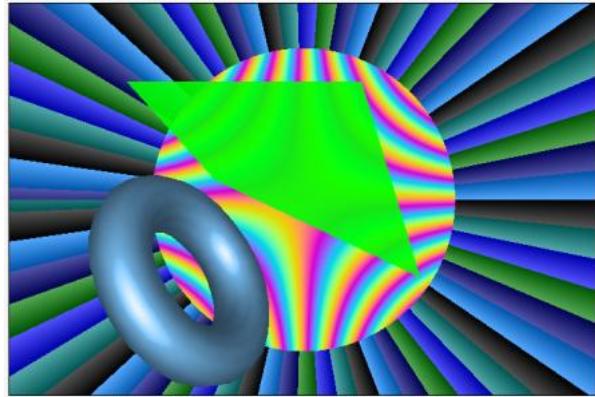
`opacity_gain=1`.

## Example of use:

```
$ gmic image.jpg testimage2d {w},{h} +sub_alpha[0] [1] display_rgba
```



[0]: 'image.jpg' (640x427x1x3)



[1]: '[2D test image]' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)

## superformula3d

### Arguments:

- `resolution>1, m>=1, n1, n2, n3`

### Description:

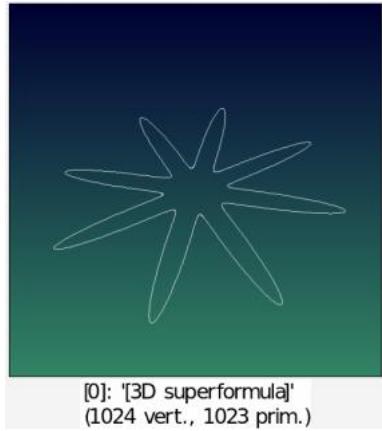
Input 2D superformula curve as a 3D object.

### Default values:

`resolution=1024, m=8, n1=1, n2=5` and `n3=8`.

## Example of use:

```
$ gmic superformula3d ,
```



## svd

Built-in command

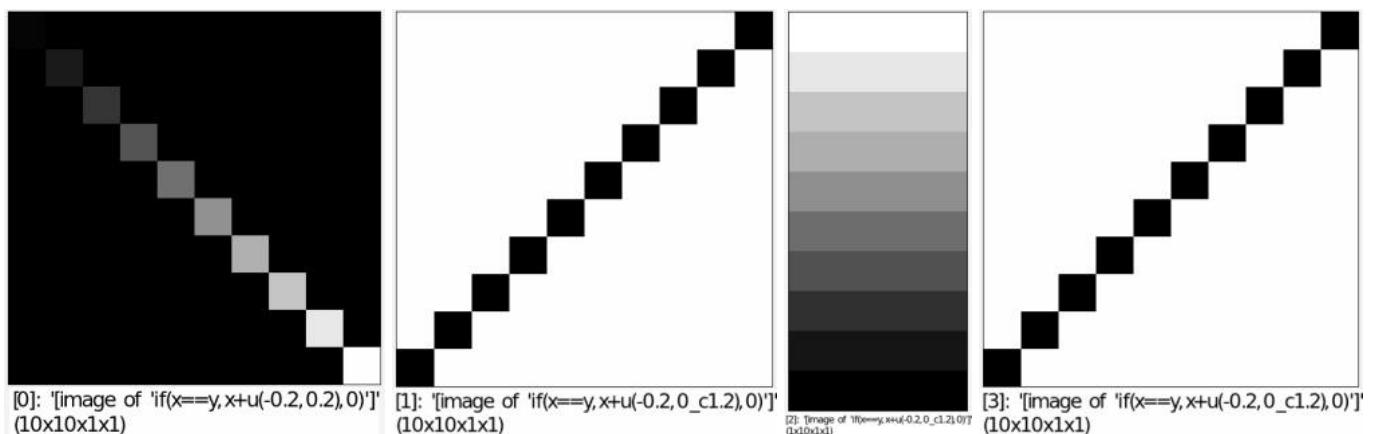
No arguments

**Description:**

Compute SVD decomposition of selected matrices.

**Example of use:**

```
$ gmic 10,10,1,1,'if(x==y,x+u(-0.2,0.2),0)' +svd
```



## symmetrize

**Arguments:**

- `_x[%],_y[%],_angle,_boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror },_is_antisymmetry={ 0 | 1 },_swap_sides={ 0 | 1 }`

**Description:**

Symmetrize selected images regarding specified axis.

## Default values:

`x=y=50%, angle=90, boundary_conditions=3, is_antisymmetry=0` and `swap_sides=0`.

## Example of use:

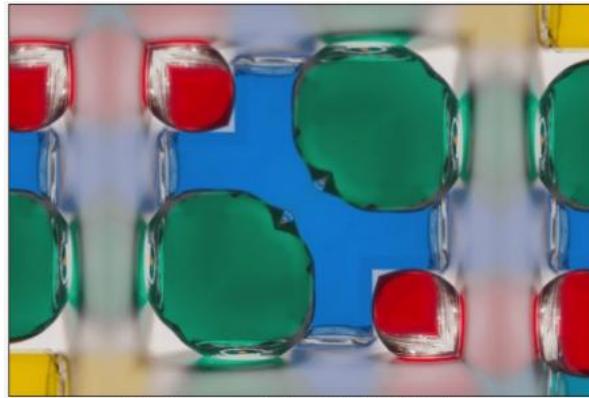
```
$ gmic image.jpg +symmetrize 50%,50%,45 +symmetrize[-1] 50%,50%, -45
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)



[2]: 'image\_c2.jpg' (640x427x1x3)

## syntexturize

### Arguments:

- `_width[%]>0, _height[%]>0`

### Description:

Resynthesize `width'x'height` versions of selected micro-textures by phase randomization.

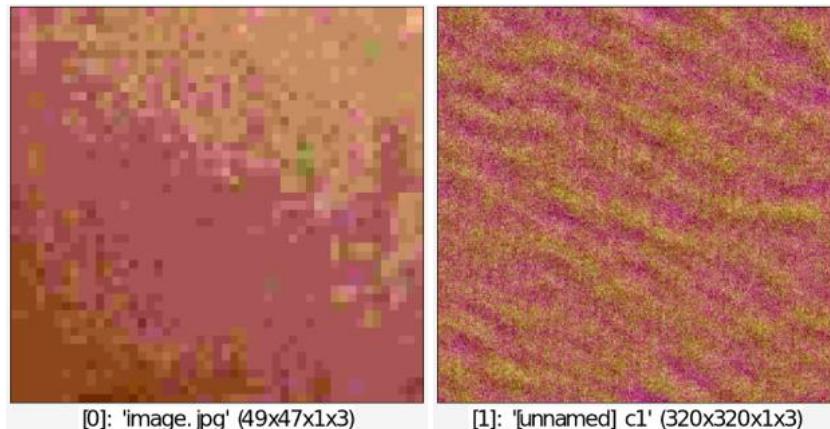
The texture synthesis algorithm is a straightforward implementation of the method described in :  
[http://www.ipol.im/pub/art/2011/ggm\\_rpn/](http://www.ipol.im/pub/art/2011/ggm_rpn/).

## Default values:

`width=height=100%`.

## Example of use:

```
$ gmic image.jpg crop 2,282,50,328 +syntexturize 320,320
```



---

## syntexturize\_matchpatch

### Arguments:

- `_width[%]>0,_height[%]>0,_nb_scales>=0,_patch_size>0,_blending_size>=0,_prec`

### Description:

Resyntheticize `width'x'height` versions of selected micro-textures using a patch-matching algorithm.

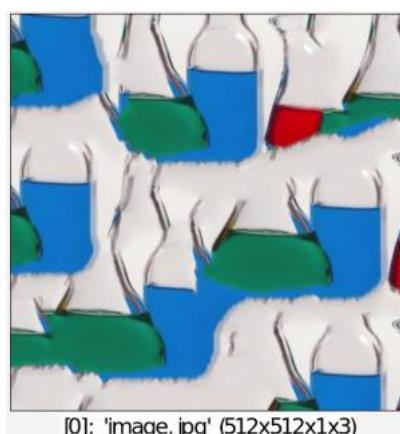
If `nbscales==0`, the number of scales used is estimated from the image size.

### Default values:

`width=height=100%`, `nb_scales=0`, `patch_size=7`, `blending_size=5` and `precision=1`.

### Example of use:

```
$ gmic image.jpg crop 25%,25%,75%,75% syntexturize_matchpatch 512,512
```



# tan

Built-in command

## No arguments

### Description:

Compute the pointwise tangent of selected images.

This command has a [tutorial page](#).

### Examples of use:

- Example #1

```
$ gmic image.jpg +normalize {-0.47*pi},{0.47*pi} tan[-1]
```



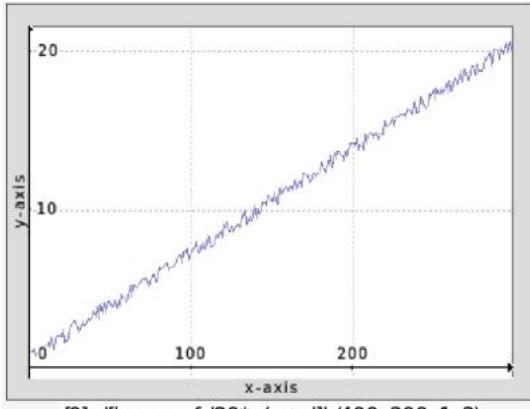
[0]: 'image.jpg' (640x427x1x3)



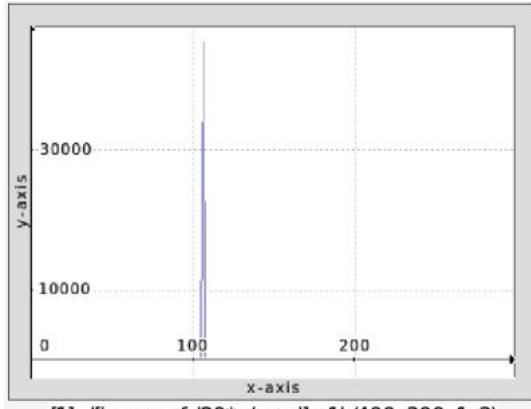
[1]: 'image\_c1.jpg' (640x427x1x3)

- Example #2

```
$ gmic 300,1,1,1,'20*x/w+u' +tan display_graph 400,300
```



[0]: '[image of '20\*x/w+u']' (400x300x1x3)



[1]: '[image of '20\*x/w+u']\_c1' (400x300x1x3)

# tanh

Built-in command

## No arguments

### Description:

Compute the pointwise hyperbolic tangent of selected images.

### Examples of use:

- Example #1

```
$ gmic image.jpg +normalize -3,3 tanh[-1]
```



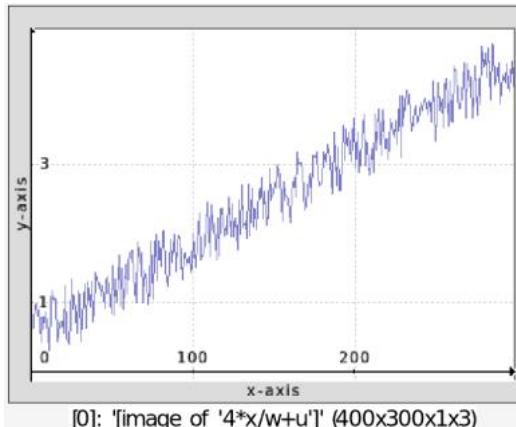
[0]: 'image.jpg' (640x427x1x3)



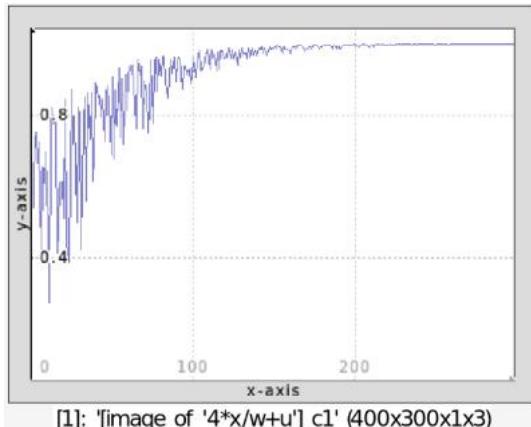
[1]: 'image\_c1.jpg' (640x427x1x3)

- Example #2

```
$ gmic 300,1,1,1,'4*x/w+u' +tanh display_graph 400,300
```



[0]: '[image of '4\*x/w+u']' (400x300x1x3)



[1]: '[image of '4\*x/w+u']\_c1' (400x300x1x3)

---

## taquin

### Arguments:

- `M>0, _N>0, _remove_tile={ 0=none | 1=first | 2=last | 3=random }`, `_relief`, `_border_thickness[%]`, `_border_outline[%]`, `_outline_color`

### Description:

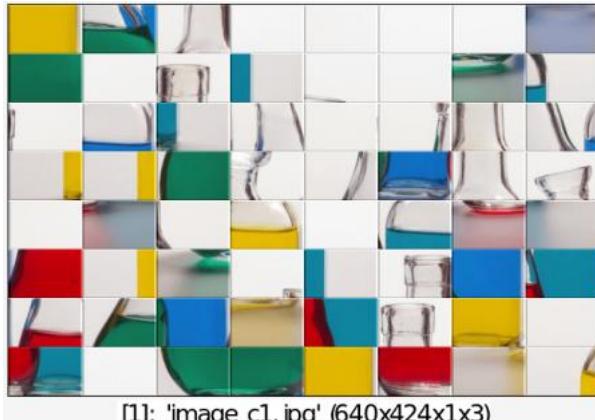
Create MxN taquin puzzle from selected images.

## Default values:

`N=M`, `relief=50`, `border_thickness=5`, `border_outline=0` and `remove_tile=0`.

## Example of use:

```
$ gmic image.jpg +taquin 8
```



---

## tensors3d

### Arguments:

- `_radius_factor>=0`, `_shape={ 0=box | >=N=ellipsoid }`, `_radius_min>=0`

### Description:

Generate 3D tensor fields from selected images.

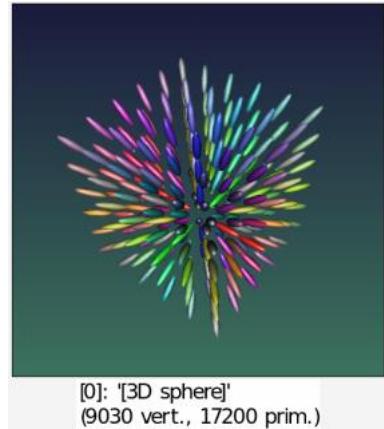
when 'shape'>0, it gives the ellipsoid shape precision.

## Default values:

`radius_factor=1`, `shape=2` and `radius_min=0.05`.

## Example of use:

```
$ gmic 6,6,6,9,"U = [x,y,z] - [w,h,d]/2; U/=norm(U); mul(U,U,3) + 0.3*eye(3)" tensors3d 0.8
```



## testimage2d

### Arguments:

- `_width>0, _height>0, _spectrum>0`

### Description:

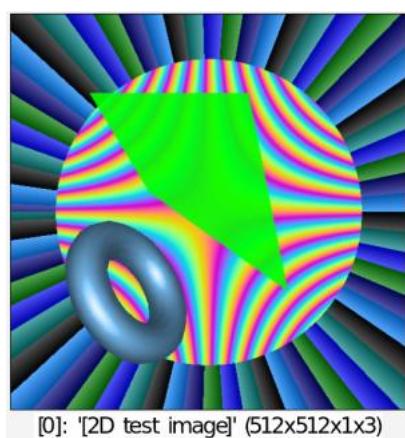
Input a 2D synthetic image.

### Default values:

`width=512`, `height=width` and `spectrum=3`.

### Example of use:

```
$ gmic testimage2d 512
```



## tetraedron\_shade

### Arguments:

- `x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3,R0,G0,B0,...,R1,G1,B1,...,R2,G2,B2,...,R`

## Description:

Draw tetraedron with interpolated colors on selected (volumetric) images.

---

# tetris

## Arguments:

- `_scale>0`

## Description:

Apply tetris effect on selected images.

## Default values:

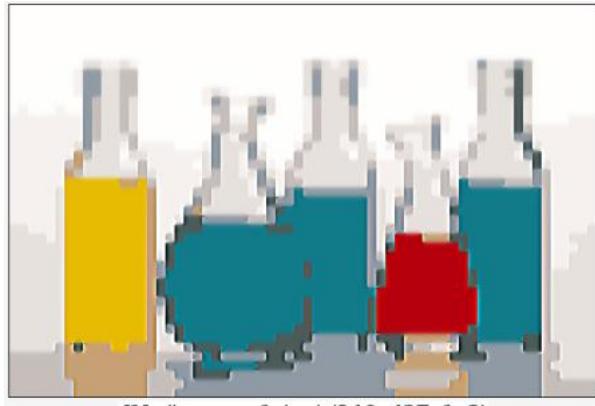
`scale=10`.

## Example of use:

```
$ gmic image.jpg +tetris 10
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

# text

Built-in command

---

## Arguments:

- `text,_x[%|~],_y[%|~],_font_height[%]>=0,_opacity,_color1,...`

## Description:

Draw specified colored text string on selected images.

(equivalent to shortcut command `t`).

If one of the x or y argument ends with a `~`, its value is expected to be

a centering ratio (in [0,1]) rather than a position.

Usual centering ratio are `{ 0=left-justified | 0.5=centered | 1=right-justified }`.

Sizes `13` and `128` are special and correspond to binary fonts (no-antialiasing).

Any other font size is rendered with anti-aliasing.

Specifying an empty target image resizes it to new dimensions such that the image contains the entire text string.

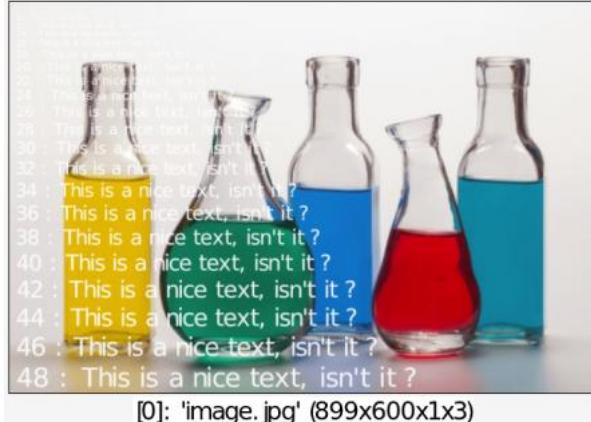
## Default values:

`x=y=0.01~`, `font_height=16`, `opacity=1` and `color1=0`.

## Examples of use:

- Example #1

```
$ gmic image.jpg resize2dy 600 y=0 repeat 30 text {2*$>}": This is a nice text, isn't it?",10,$y,{2*$>},0.9,255 y+={2*$>} done
```



- Example #2

```
$ gmic 0 text "G'MIC",0,0,23,1,255
```



---

## text3d

### Arguments:

- `text,_font_height>0,_depth>0,_smoothness`

## Description:

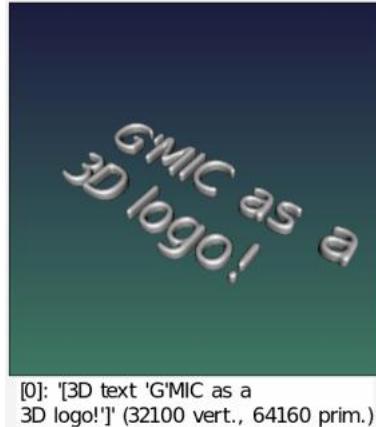
Input a 3D text object from specified text.

## Default values:

`font_height=53`, `depth=10` and `smoothness=1.5`.

## Example of use:

```
$ gmic text3d "G'MIC as a\n3D logo!"
```



---

## text\_outline

### Arguments:

- `text`, `_x[%|~]`, `_y[%|~]`, `_font_height[%]>0`, `_outline>=0`, `_opacity`, `_color1`, ...

## Description:

Draw specified colored and outlined text string on selected images.

If one of the x or y argument ends with a `~`, its value is expected to be a centering ratio (in [0,1]) rather than a position.

Usual centering ratio are `{ 0=left-justified | 0.5=centered | 1=right-justified }`.

## Default values:

`x=y=0.01~`, `font_height=7.5%`, `outline=2`, `opacity=1`, `color1=color2=color3=255` and `color4=255`.

## Example of use:

```
$ gmic image.jpg text_outline "Hi there!",10,10,63,3
```



---

## text\_pointcloud3d

### Arguments:

- `_text1`, `_text2`, `_smoothness`

### Description:

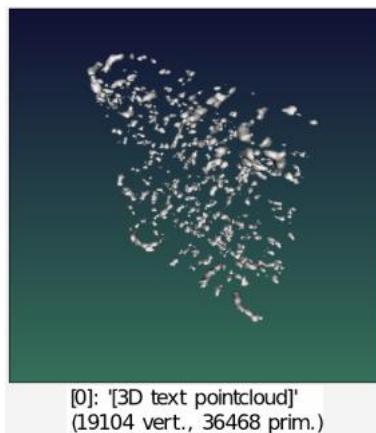
Input 3D text pointcloud from the two specified strings.

### Default values:

`text1="text1"`, `text2="text2"` and `smoothness=1`.

### Example of use:

```
$ gmic text_pointcloud3d "G'MIC","Rocks!"
```



---

## texturize3d

### Arguments:

- `[ind_texture]`, `[ind_coords]`

## Description:

Texturize selected 3D objects with specified texture and coordinates.

(equivalent to shortcut command `t3d`).

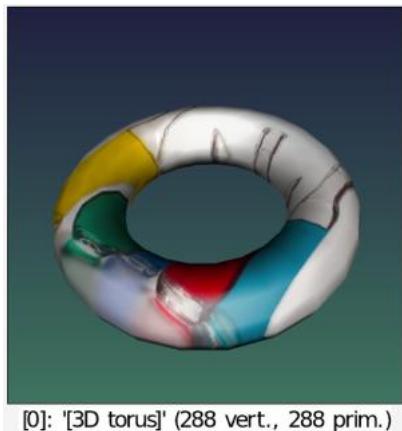
When `[ind_coords]` is omitted, default XY texture projection is performed.

## Default values:

`ind_coords=(undefined)`.

## Example of use:

```
$ gmic image.jpg torus3d 100,30 texturize3d[-1] [-2] keep[-1]
```



[0]: '[3D torus]' (288 vert., 288 prim.)

---

## texturize\_canvas

### Arguments:

- `_amplitude>=0, _fibrousness>=0, _emboss_level>=0`

## Description:

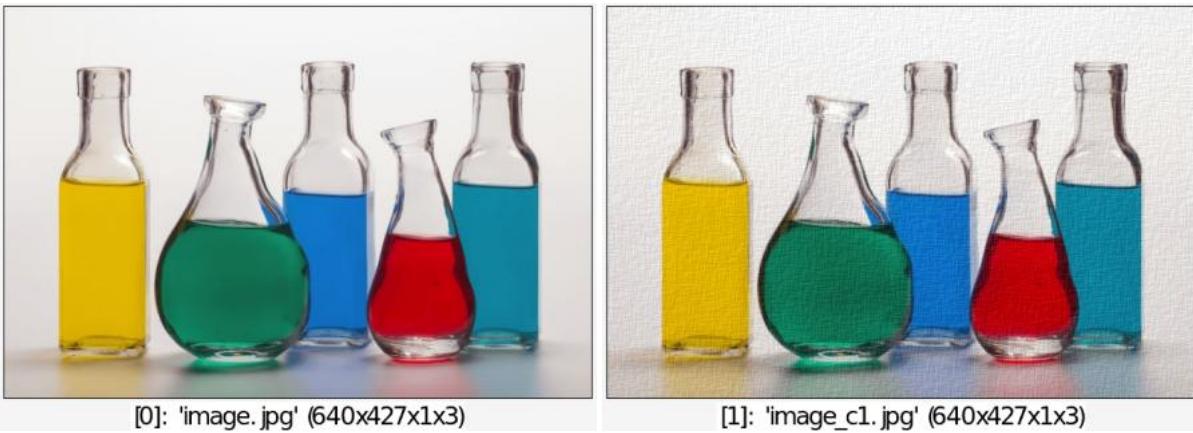
Add paint canvas texture to selected images.

## Default values:

`amplitude=20, fibrousness=3` and `emboss_level=0.6`.

## Example of use:

```
$ gmic image.jpg +texturize_canvas ,
```



---

## texturize\_paper

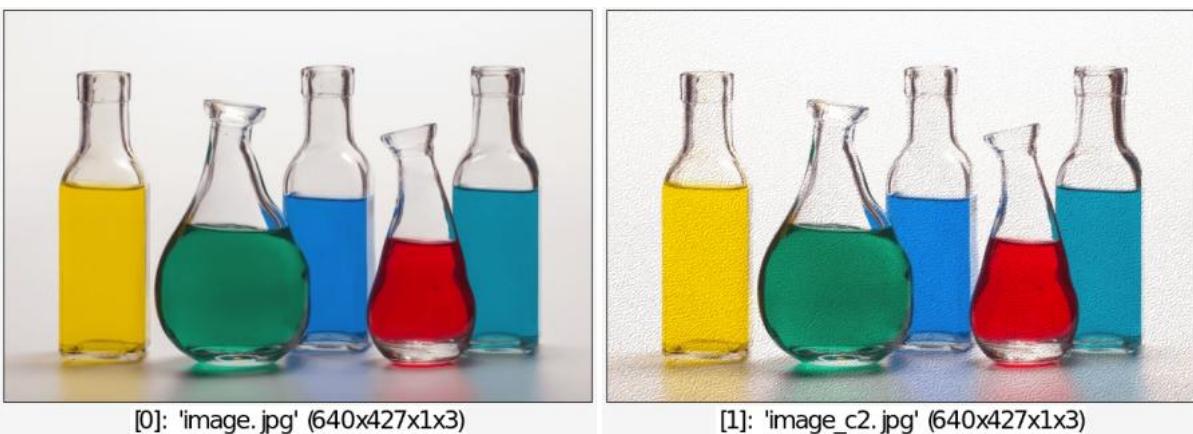
**No arguments**

**Description:**

Add paper texture to selected images.

**Example of use:**

```
$ gmic image.jpg +texturize_paper
```



---

## thinning

**Arguments:**

- `_boundary_conditions={ 0=dirichlet | 1=neumann }`

**Description:**

Compute skeleton of binary shapes using morphological thinning

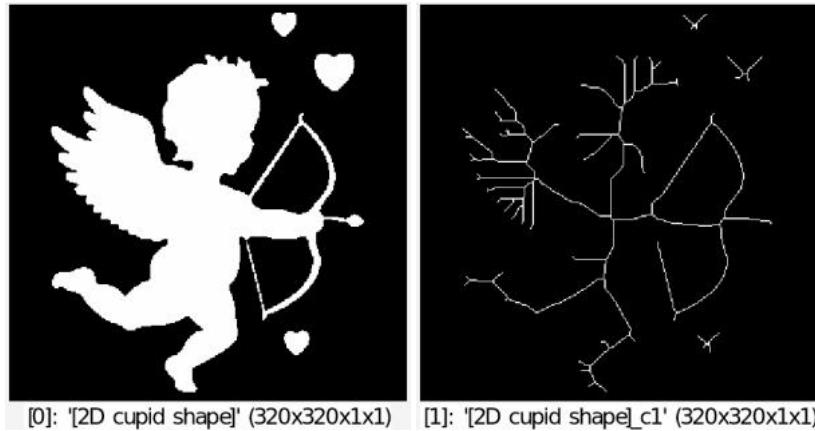
(beware, this is a quite slow iterative process)

## Default values:

`boundary_conditions=1`.

## Example of use:

```
$ gmic shape_cupid 320 +thinning
```



---

# threshold

## Arguments:

- `value[%]`, `_is_soft={ 0 | 1 }` :

## Description:

Threshold values of selected images.

`soft` can be `{ 0=hard-thresholding | 1=soft-thresholding }`.

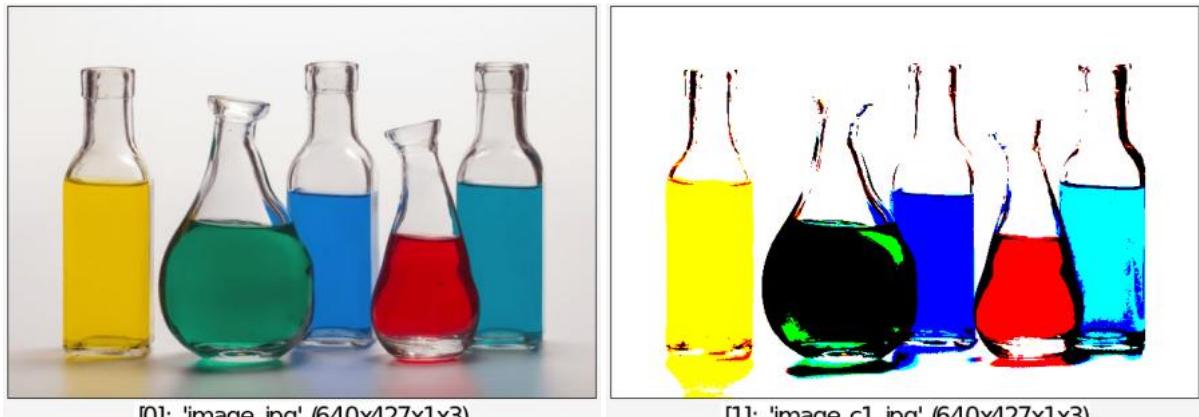
## Default values:

`is_soft=0`.

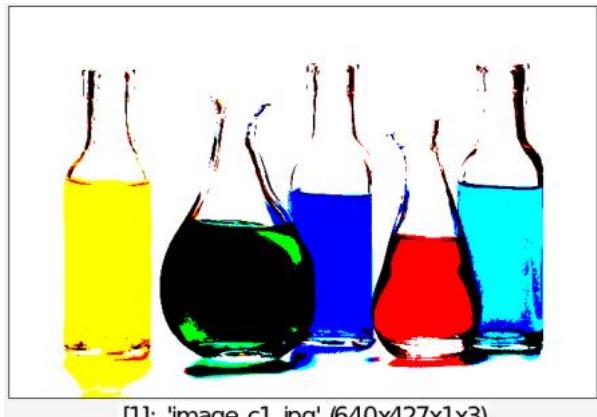
This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg +threshold[0] 50% +threshold[0] 50%,1
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)

## tic

**No arguments**

**Description:**

Initialize tic-toc timer.

Use it in conjunction with `toc`.

## tixy

**Arguments:**

- `"expression"`

**Description:**

Animate specified mathematical expression with a 16x16 grid of circles, using the rules described at <https://tixy.land>.

## to\_a

**No arguments**

## **Description:**

Force selected images to have an alpha channel.

---

# to\_clutname

## **Arguments:**

- `"string"`

## **Description:**

Return simplified name that can be used as a CLUT name, from specified input string.

---

# to\_color

**No arguments**

## **Description:**

Force selected images to be in color mode (RGB or RGBA).

---

# to\_colormode

## **Arguments:**

- `mode={ 0=adaptive | 1=G | 2=GA | 3=RGB | 4=RGBA }`

## **Description:**

Force selected images to be in a given color mode.

## **Default values:**

`mode=0`.

---

# to\_gray

**No arguments**

## **Description:**

Force selected images to be in GRAY mode.

## Example of use:

```
$ gmic image.jpg +to_gray
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x1)

---

## to\_graya

No arguments

### Description:

Force selected images to be in GRAYA mode.

---

## to\_pseudogray

### Arguments:

- `_max_step>=0, _is_perceptual_constraint={ 0 | 1 }, _bits_depth>0`

### Description:

Convert selected scalar images ([0-255]-valued) to pseudo-gray color images.

### Default values:

`max_step=5`, `is_perceptual_constraint=1` and `bits_depth=8`.

The original pseudo-gray technique has been introduced by Rich Franzen  
<http://r0k.us/graphics/pseudoGrey.html>.

Extension of this technique to arbitrary increments for more tones, has been done by David Tschumperlé.

---

# to\_rgb

**No arguments**

## Description:

Force selected images to be in RGB mode.

---

# to\_rgba

**No arguments**

## Description:

Force selected images to be in RGBA mode.

---

# toc

**No arguments**

## Description:

Display elapsed time of the tic-toc timer since the last call to `tic`.

This command returns the elapsed time in the status value.

Use it in conjunction with `tic`.

---

# tones

## Arguments:

- `N>0`

## Description:

Get N tones masks from selected images.

## Example of use:

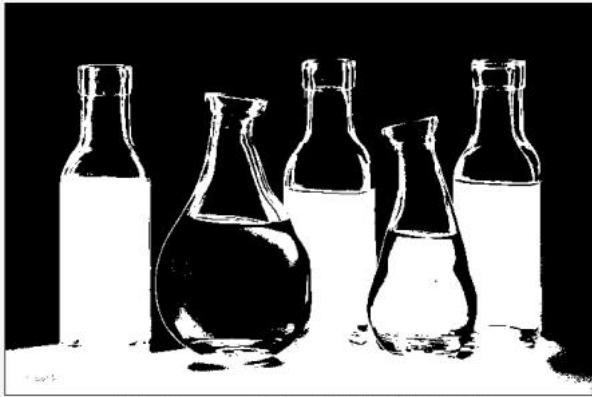
```
$ gmic image.jpg +tones 3
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x1)



[2]: 'image\_c2.jpg' (640x427x1x1)



[3]: 'image\_c2.jpg' (640x427x1x1)

---

## topographic\_map

### Arguments:

- `_nb_levels>0, _smoothness`

### Description:

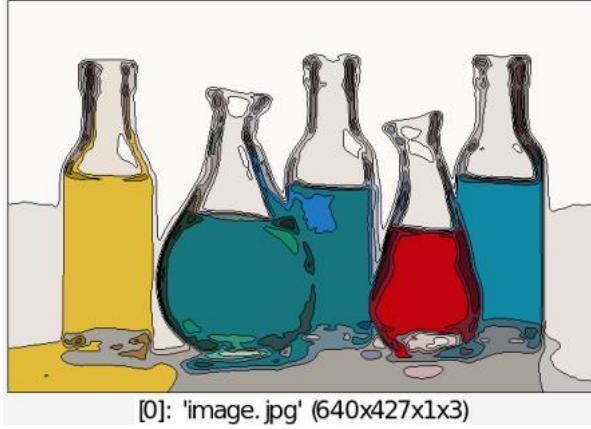
Render selected images as topographic maps.

### Default values:

`nb_levels=16` and `smoothness=2`.

### Example of use:

```
$ gmic image.jpg topographic_map 10
```



---

## torus3d

### Arguments:

- `_radius1,_radius2,_nb_subdivisions1>2,_nb_subdivisions2>2`

### Description:

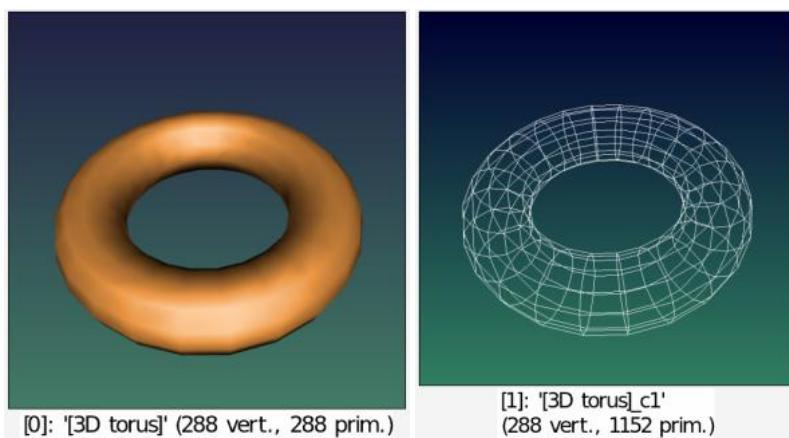
Input 3D torus at (0,0,0), with specified geometry.

### Default values:

`radius1=1`, `radius2=0.3`, `nb_subdivisions1=24` and `nb_subdivisions2=12`.

### Example of use:

```
$ gmic torus3d 10,3 +primitives3d 1 color3d[-2] ${-rgb}
```



---

## transfer\_histogram

### Arguments:

- `[reference_image],_nb_levels>0,_color_channels`

## Description:

Transfer histogram of the specified reference image to selected images.

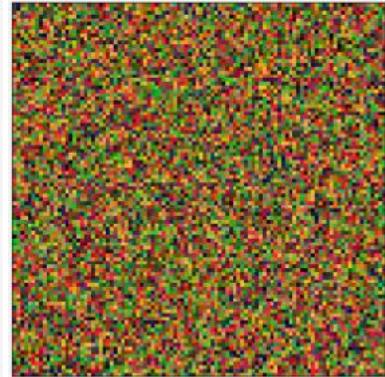
Argument 'color channels' is the same as with command [apply\\_channels](#).

## Default values:

`nb_levels=256` and `color_channels=all`.

## Example of use:

```
$ gmic image.jpg 100,100,1,3,"u([256,200,100])"  
+transfer_histogram[0] [1]
```



---

## transfer\_pca

### Arguments:

- `[reference_image],_color_channels`

## Description:

Transfer mean and covariance matrix of specified vector-valued reference image to selected images.

Argument 'color channels' is the same as with command [apply\\_channels](#).

## Default values:

`color_channels=all`.

## Example of use:

```
$ gmic sample lena,earth +transfer_pca[0] [1]
```



---

## transfer\_rgb

### Arguments:

- `[target],_gamma>=0,_regularization>=0,_luminosity_constraints>=0,_rgb_resolution { 0 | 1 }`

### Description:

Transfer colors from selected source images to selected reference image (given as argument).

`gamma` determines the importance of color occurrences in the matching process (0=none to 1=huge).

`regularization` determines the number of guided filter iterations to remove quantization effects.

`luminosity_constraints` tells if luminosity constraints must be applied on non-confident matched colors.

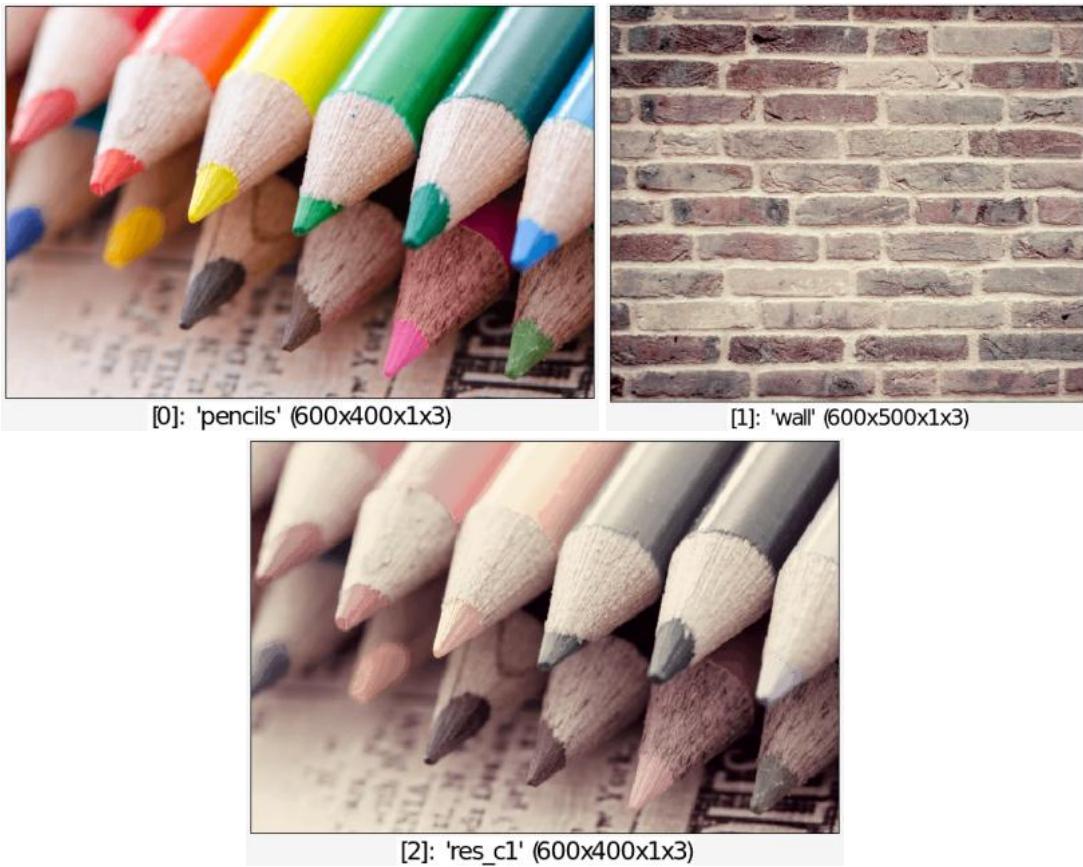
`is_constraints` tells if additional hard color constraints must be set (opens an interactive window).

## Default values:

`gamma=0.3,regularization=8,luminosity_constraints=0.1,rgb_resolution=64` and `is_constraints=0`.

## Example of use:

```
$ gmic sample pencils,wall +transfer_rgb[0] [1],0,0.01
```



## transform\_polar

### Arguments:

- `"expr_radius", "expr_angle", _center_x[%], _center_y[%], _boundary_conditions= { 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

### Description:

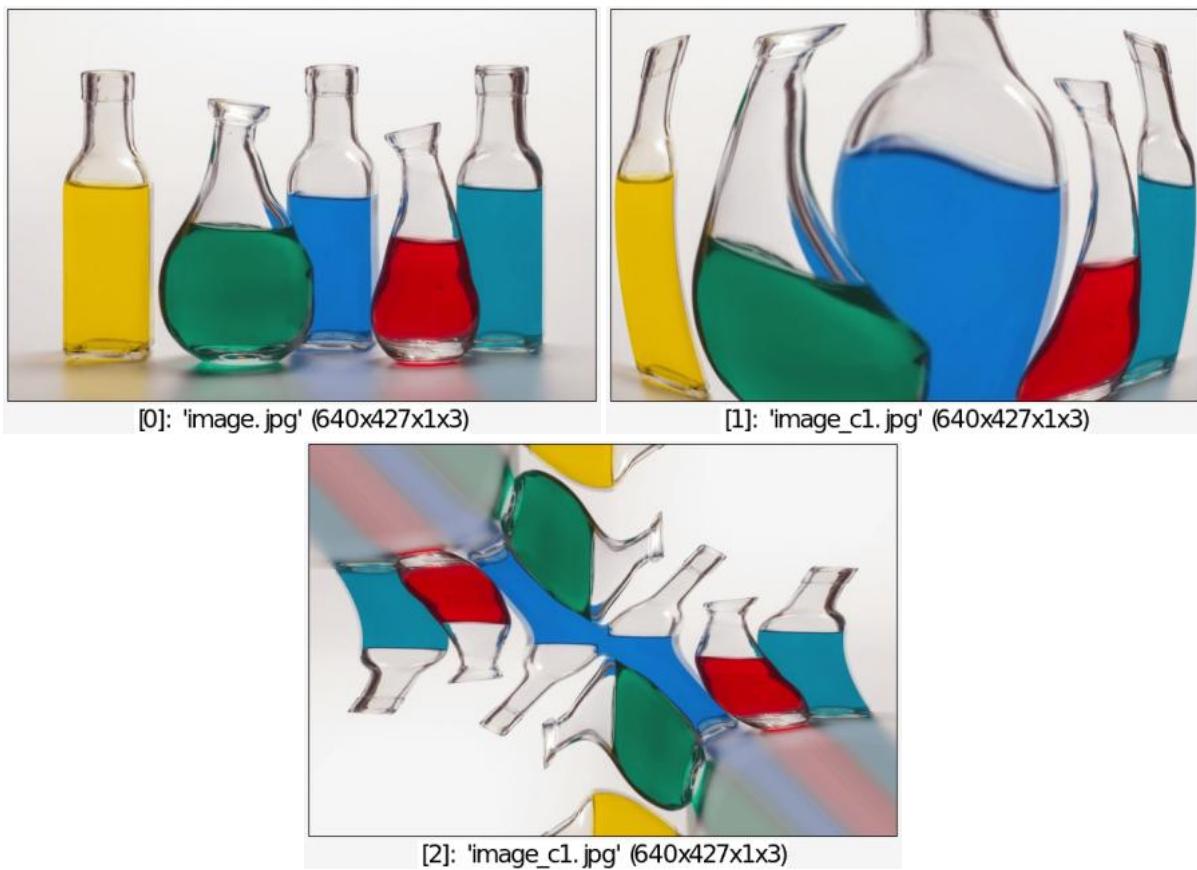
Apply user-defined transform on polar representation of selected images.

### Default values:

`expr_radius=R-r`, `expr_rangle=a`, `center_x=center_y=50%` and `boundary_conditions=1`.

### Example of use:

```
$ gmic image.jpg +transform_polar[0] R*(r/R)^2,a +transform_polar[0]
r,2*a
```



## transition

### Arguments:

- `[transition_shape],nb_added_frames>=0,100>=shading>=0,_single_frame_only={-1=disabled | >=0 }`

### Description:

Generate a transition sequence between selected images.

### Default values:

`shading=0` and `single_frame_only=-1`.

### Example of use:

```
$ gmic image.jpg +mirror c 100%,100% plasma[-1] 1,1,6 transition[0,1]
[2],5
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image.jpg #1' (640x427x1x3)



[2]: 'image.jpg #2' (640x427x1x3)



[3]: 'image.jpg #3' (640x427x1x3)



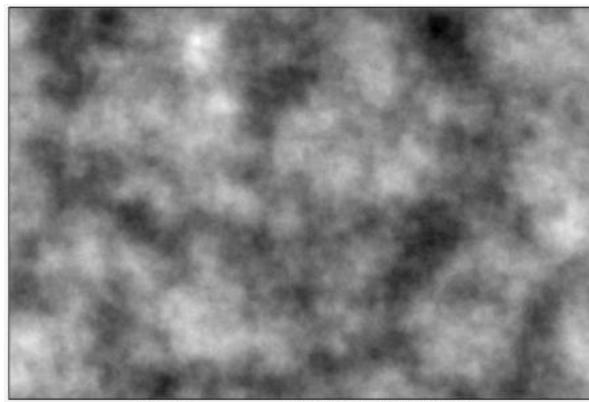
[4]: 'image.jpg #4' (640x427x1x3)



[5]: 'image.jpg #5' (640x427x1x3)



[6]: 'image\_c1.jpg' (640x427x1x3)



[7]: '[unnamed]' (640x427x1x1)

---

transition3d

**Arguments:**

- `nb_frames>=2, nb_xtiles>0, nb_ytiles>0, axis_x, axis_y, axis_z, is_antialias`  
`{ 0 | 1 }`

## Description:

Create 3D transition sequence between selected consecutive images.

`axis_x`, `axis_y` and `axis_z` can be set as mathematical expressions, depending on `x` and `y`.

## Default values:

`nb_frames=10, nb_xtiles=nb_ytiles=3, axis_x=1, axis_y=1, axis_z=0` and `is_antialias=1`.

## Example of use:

```
$ gmic image.jpg +blur 5 transition3d 9 display_rgba
```



[0]: 'image.jpg' (640x427x1x3)



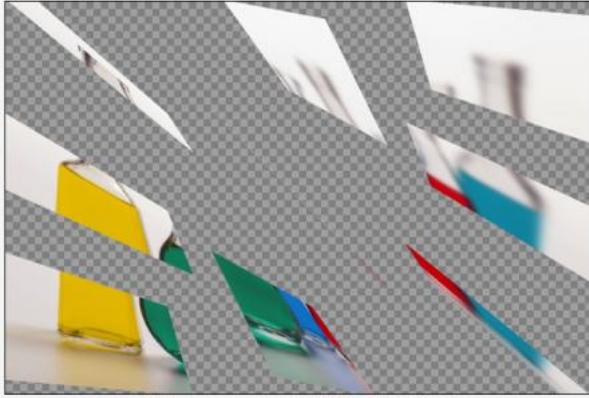
[1]: 'image.jpg' (640x427x1x3)



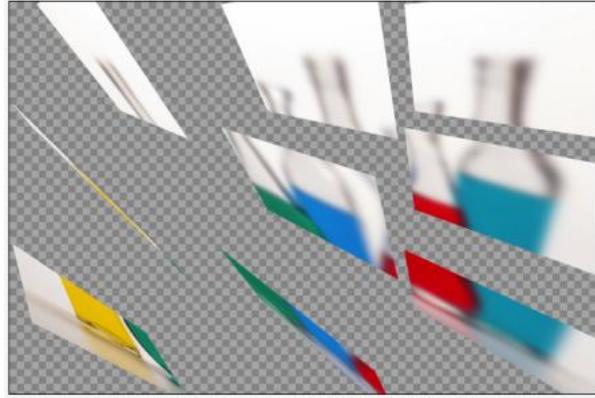
[2]: 'image.jpg' (640x427x1x3)



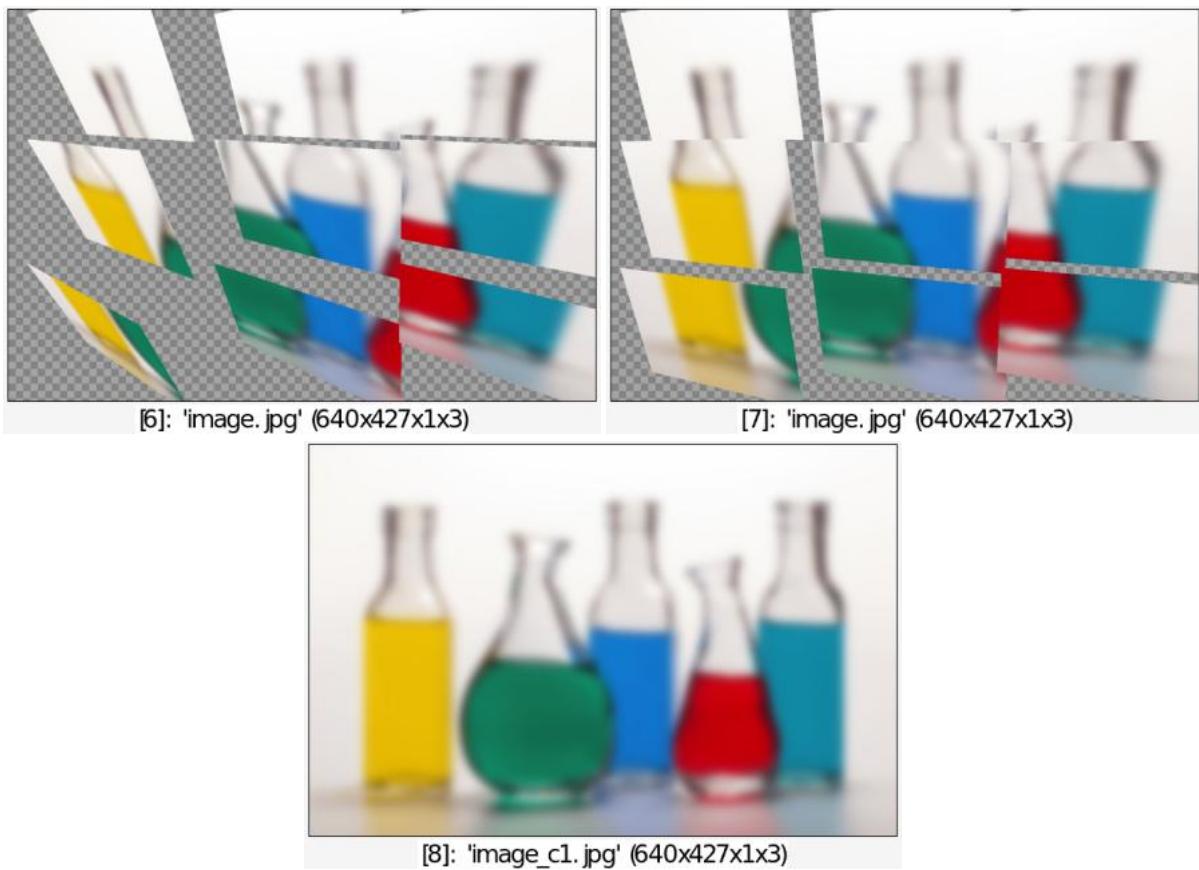
[3]: 'image.jpg' (640x427x1x3)



[4]: 'image.jpg' (640x427x1x3)



[5]: 'image.jpg' (640x427x1x3)



---

## transpose

No arguments

**Description:**

Transpose selected matrices.

**Example of use:**

```
$ gmic image.jpg +transpose
```



# triangle3d

## Arguments:

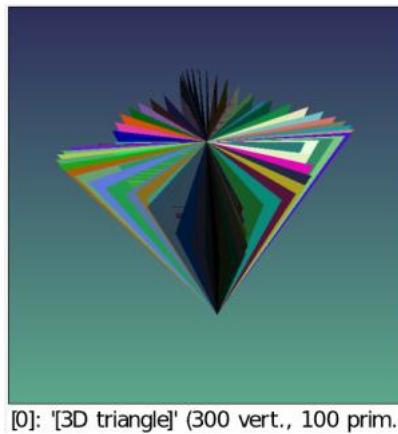
- `x0,y0,z0,x1,y1,z1,x2,y2,z2`

## Description:

Input 3D triangle at specified coordinates.

## Example of use:

```
$ gmic repeat 100 a={$>*pi/50} triangle3d 0,0,0,0,0,0,3,{cos(3*$a)}, {sin(2*$a)},0 color3d[-1] ${-rgb} done add3d
```



---

# triangle\_shade

## Arguments:

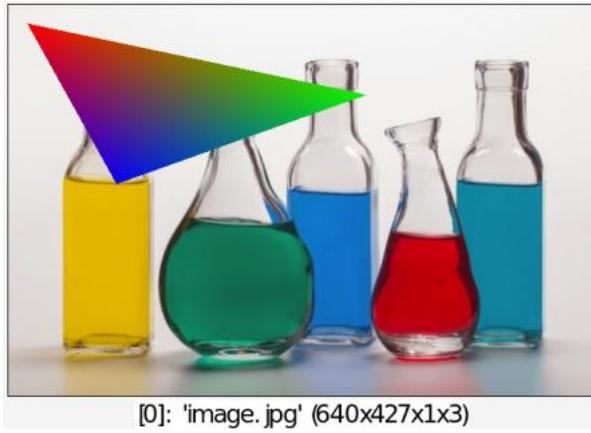
- `x0,y0,x1,y1,x2,y2,R0,G0,B0,...,R1,G1,B1,...,R2,G2,B2,...`

## Description:

Draw triangle with interpolated colors on selected images.

## Example of use:

```
$ gmic image.jpg triangle_shade  
20,20,400,100,120,200,255,0,0,0,255,0,0,0,255
```



[0]: 'image.jpg' (640x427x1x3)

---

## trisolve

Built-in command

### Arguments:

- [image]

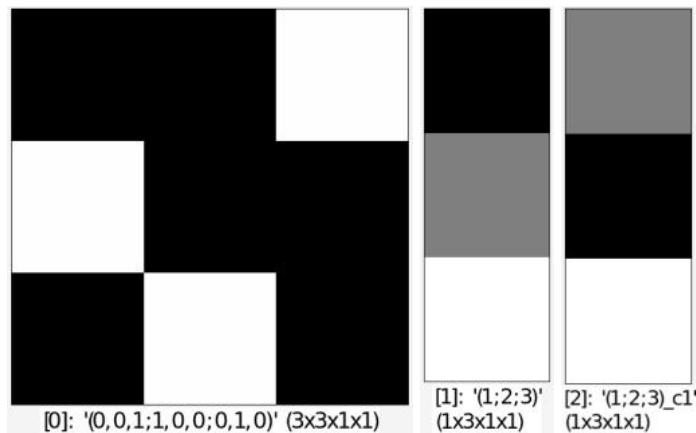
### Description:

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.

### Example of use:

```
$ gmic (0,0,1;1,0,0;0,1,0) (1;2;3) +trisolve[-1] [-2]
```



---

## truchet

### Arguments:

- \_scale>0,\_radius>=0,\_pattern\_type={ 0=straight | 1=curved }

## Description:

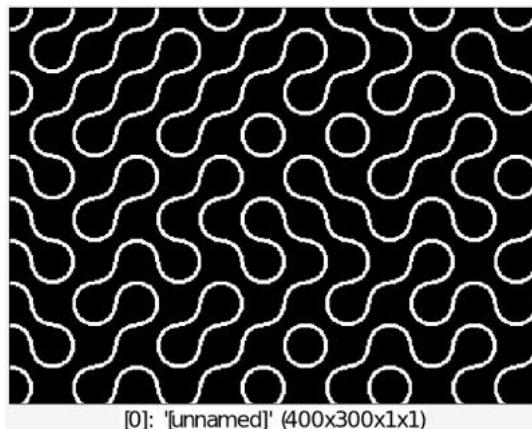
Fill selected images with random truchet patterns.

## Default values:

`scale=32`, `radius=5` and `pattern_type=1`.

## Example of use:

```
$ gmic 400,300 truchet ,
```



---

## tsp

### Arguments:

- `_precision>=0`

## Description:

Try to solve the 'travelling salesman' problem, using a combination of greedy search and 2-opt algorithms.

Selected images must have dimensions Nx1x1xC to represent N cities each with C-dimensional coordinates.

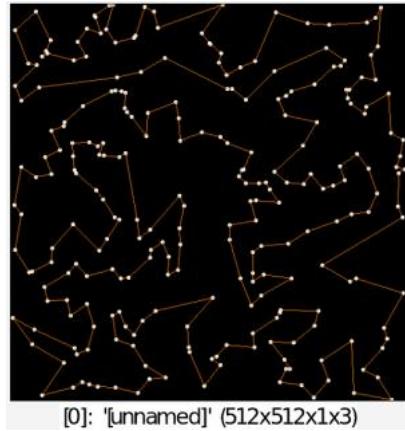
This command re-order the selected data along the x-axis so that the point sequence becomes a shortest path.

## Default values:

`precision=256`.

## Example of use:

```
$ gmic 256,1,1,2 rand 0,512 tsp , 512,512,1,3 repeat w#0 circle[-1]
{0,I[$>]},2,1,255,255,255 line[-1] {0,boundary=2;
[I[$>],I[$>+1]]},1,255,128,0 done keep[-1]
```



## tunnel

### Arguments:

- `_level>=0, _factor>0, _centering_x, _centering_y, _opacity, _angle`

### Description:

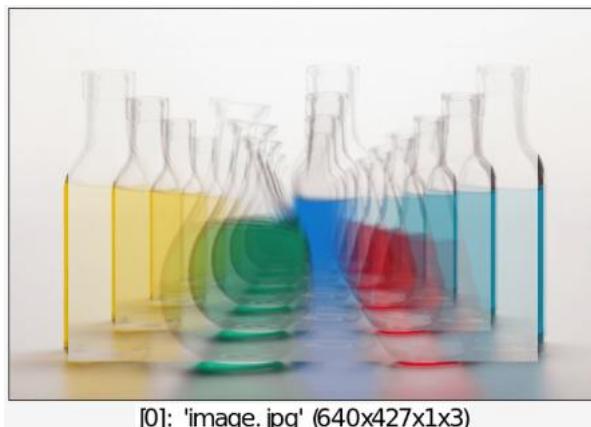
Apply tunnel effect on selected images.

### Default values:

`level=9, factor=80%, centering_x=centering_y=0.5, opacity=1` and `angle=0`

### Example of use:

```
$ gmic image.jpg tunnel 20
```



## turbulence

### Arguments:

- `_radius>0, _octaves={1,2,3...,12}, _alpha>0, _difference={-10,10}, _mode={0,1,2,3}`

## Description:

Render fractal noise or turbulence on selected images.

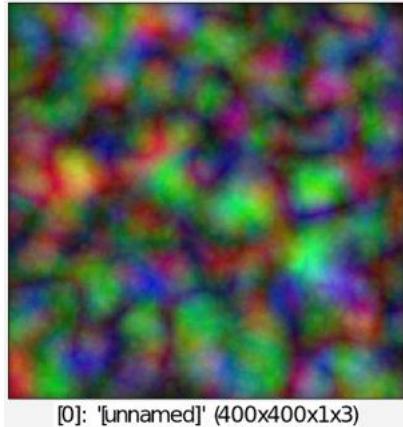
## Default values:

`radius=32`, `octaves=6`, `alpha=3`, `difference=0` and `mode=0`.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic 400,400,1,3 turbulence 16
```



---

## tv\_flow

### Arguments:

- `_nb_iter>=0, _dt, _keep_sequence={ 0 | 1 }`

## Description:

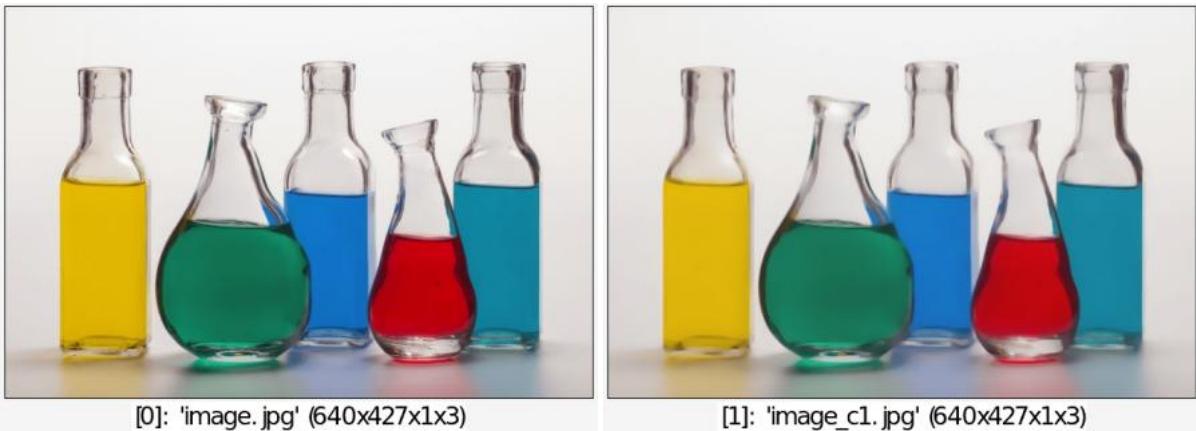
Apply iterations of the total variation flow on selected images.

## Default values:

`nb_iter=10`, `dt=30` and `keep_sequence=0`.

## Example of use:

```
$ gmic image.jpg +tv_flow 40
```



---

## twirl

### Arguments:

- `_amplitude, _center_x[%], _center_y[%], _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

### Description:

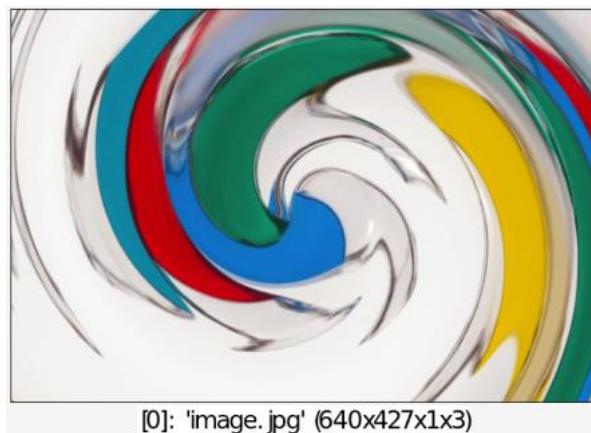
Apply twirl deformation on selected images.

### Default values:

`amplitude=1`, `center_x=center_y=50%` and `boundary_conditions=3`.

### Example of use:

```
$ gmic image.jpg twirl 0.6
```



---

## uchar2base64

### Arguments:

- `_encoding={ 0=base64 | 1=base64url }`

## Description:

Encode the values of the latest of the selected images as a base64-encoded string.

The string can be decoded using command `base642uchar`.

Selected images must have values that are integers in [0,255].

## Default values:

`encoding=0`.

---

# uncommand

Built-in command

## Arguments:

- `command_name[, _command_name2, ...]` or
- `*`

## Description:

Discard definition of specified custom commands.

Set argument to `*` for discarding all existing custom commands.

(equivalent to shortcut command `um`).

---

# undistort

## Arguments:

- `-1<=_amplitude<=1, _aspect_ratio, _zoom, _center_x[%], _center_y[%], _boundary_co`

## Description:

Correct barrel/pincushion distortions occurring with wide-angle lens.

### References:

[1] Zhang Z. (1999). Flexible camera calibration by viewing a plane from unknown orientation.

[2] Andrew W. Fitzgibbon (2001). Simultaneous linear estimation of multiple view geometry and lens distortion.

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

## Default values:

`amplitude=0.25`, `aspect_ratio=0`, `zoom=0`, `center_x=center_y=50%` and  
`boundary_conditions=0`.

---

## uniform\_distribution

### Arguments:

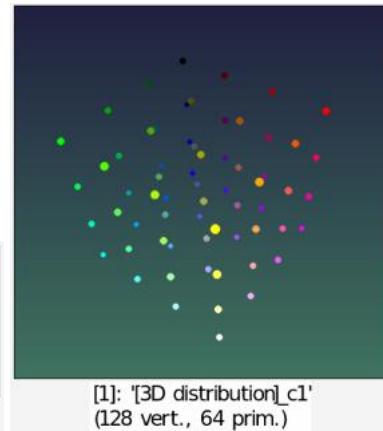
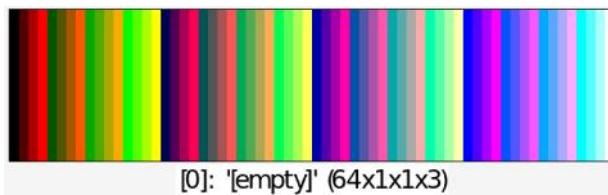
- `nb_levels>=1, spectrum>=1`

### Description:

Input set of uniformly distributed spectrum-d points in  $[0,1]^{\text{spectrum}}$ .

### Example of use:

```
$ gmic uniform_distribution 64,3 * 255 +distribution3d circles3d[-1]  
10
```



## unroll

Built-in command

### Arguments:

- `_axis={ x | y | z | c }`

### Description:

Unroll selected images along specified axis.

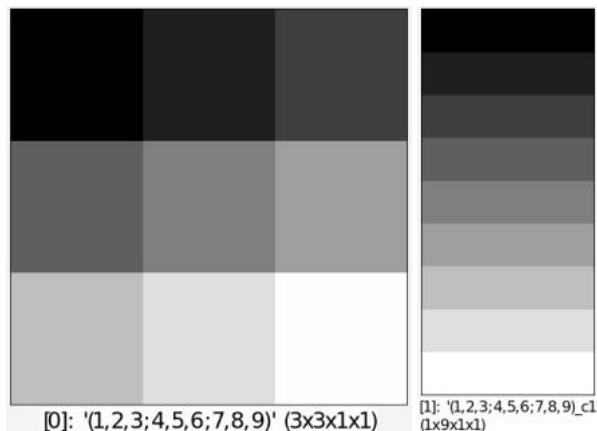
(equivalent to shortcut command `y`).

### Default values:

`axis=y`.

### Example of use:

```
$ gmic (1,2,3;4,5,6;7,8,9) +unroll y
```



## unserialize

Built-in command

**No arguments**

**Description:**

Recreate lists of images from serialized image buffers, obtained with command `serialize`.

## unsharp

**Arguments:**

- `radius[%]>=0, _amount>=0, _threshold[%]>=0`

**Description:**

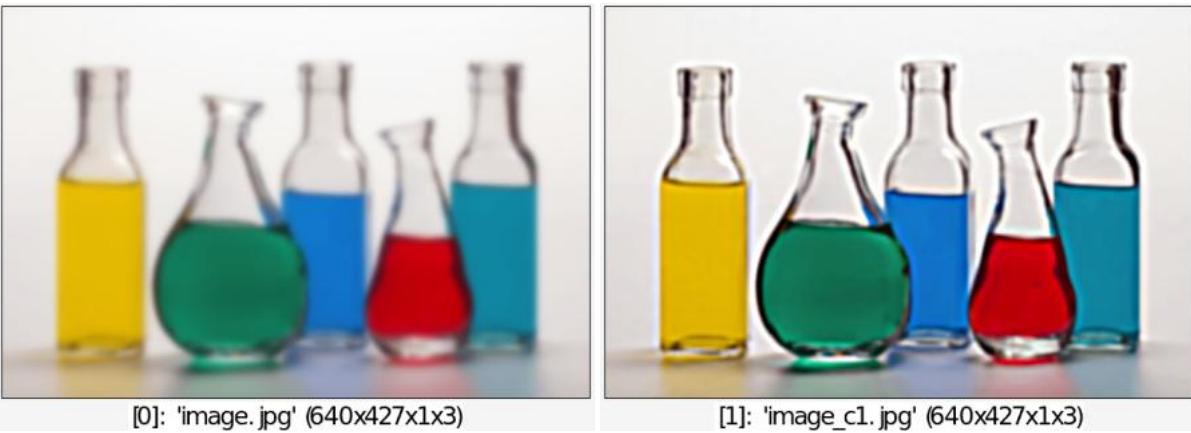
Apply unsharp mask on selected images.

**Default values:**

`amount=2` and `threshold=0`.

**Example of use:**

```
$ gmic image.jpg blur 3 +unsharp 1.5,15 cut 0,255
```



## unsharp\_octave

### Arguments:

- `_nb_scales>0, _radius[%]>=0, _amount>=0, threshold[%]>=0`

### Description:

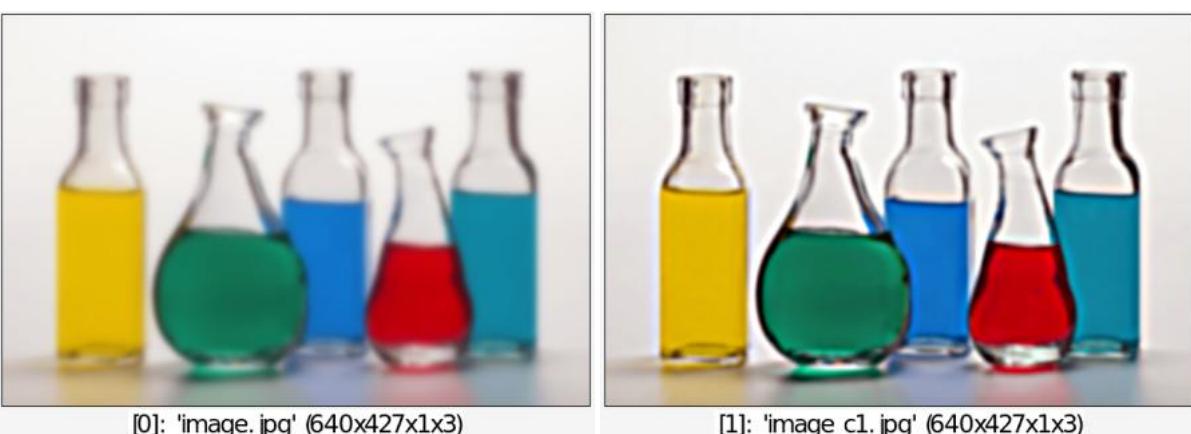
Apply octave sharpening on selected images.

### Default values:

`nb_scales=4, radius=1, amount=2` and `threshold=0`.

### Example of use:

```
$ gmic image.jpg blur 3 +unsharp_octave 4,5,15 cut 0,255
```



## update

### No arguments

### Description:

Update commands from the latest definition file on the G'MIC server.

(equivalent to shortcut command `up`).

---

## upscale\_smart

### Arguments:

- `width[%],_height[%],_depth,_smoothness>=0,_anisotropy=[0,1],sharpening>=0`

### Description:

Upscale selected images with an edge-preserving algorithm.

### Default values:

`height=100%`, `depth=100%`, `smoothness=2`, `anisotropy=0.4` and `sharpening=10`.

### Example of use:

```
$ gmic image.jpg resize2dy 100 +upscale_smart 500%,500% append x
```



## vanvliet

Built-in command

### Arguments:

- `std_deviation>=0[%],order={ 0 | 1 | 2 | 3 },axis={ x | y | z | c },_boundary_conditions`

### Description:

Apply Vanvliet recursive filter on selected images, along specified axis and with

specified standard deviation, order and boundary conditions.

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

## Default values:

`boundary_conditions=1`.

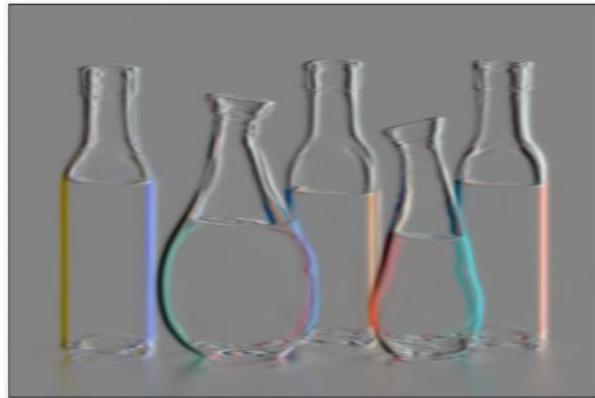
## Examples of use:

- Example #1

```
$ gmic image.jpg +vanvliet 3,1,x
```



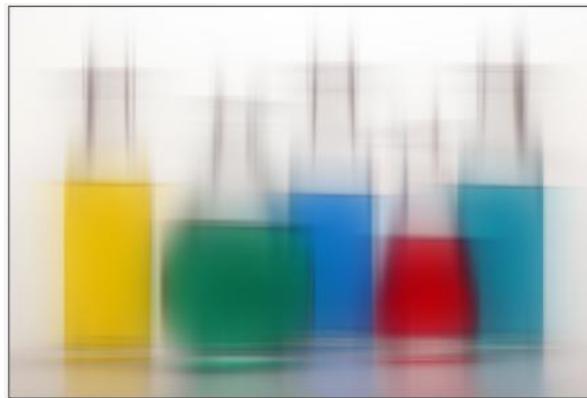
[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

- Example #2

```
$ gmic image.jpg +vanvliet 30,0,x vanvliet[-2] 30,0,y add
```



[0]: 'image.jpg' (640x427x1x3)

---

## variance\_patch

### Arguments:

- `_patch_size>=1`

### Description:

Compute variance of each images patch centered at (x,y), in selected images.

### Default values:

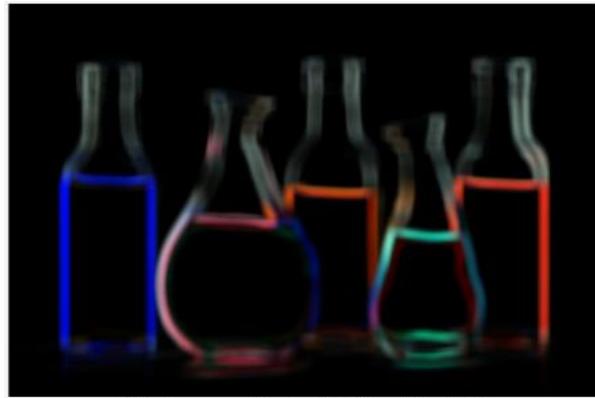
`patch_size=16`

## Example of use:

```
$ gmic image.jpg +variance_patch
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c2.jpg' (640x427x1x3)

---

## vector2tensor

### No arguments

### Description:

Convert selected vector fields to corresponding tensor fields.

---

## verbose

Built-in command

### Arguments:

- `level` or
- `{ + | - }`

### Description:

Set or increment/decrement the verbosity level. Default level is 0.

(equivalent to shortcut command `v`).

When 'level'>0, G'MIC log messages are displayed on the standard error (stderr).

### Default values:

`level=1`.

---

# version

## No arguments

### Description:

Display current version number on stdout.

---

# video2files

## Arguments:

- `_input_filename,_output_filename,_first_frame>=0,_last_frame={ >=0 | -1=last },_frame_step>=1`

### Description:

Split specified input video file into image files, one for each frame.

First and last frames as well as step between frames can be specified.

## Default values:

`output_filename=frame.png`, `first_frame=0`, `last_frame=-1` and `frame_step=1`.

---

# vignette

## Arguments:

- `_strength>=0,0<=_radius_min<=100,0<=_radius_max<=100`

### Description:

Add vignette effect to selected images.

## Default values:

`strength=100`, `radius_min=70` and `radius_max=90`.

## Example of use:

```
$ gmic image.jpg vignette ,
```



## volume3d

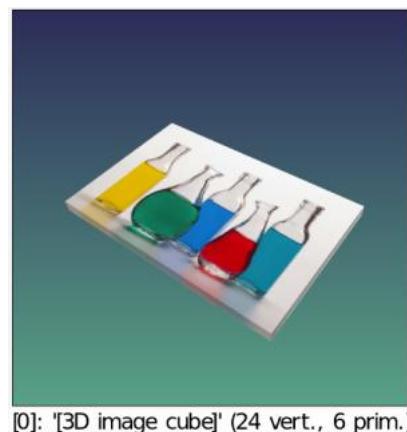
**No arguments**

**Description:**

Transform selected 3D volumetric images as 3D parallelepipedic objects.

**Example of use:**

```
$ gmic image.jpg animate blur,0,5,30 append z volume3d
```



## voronoi

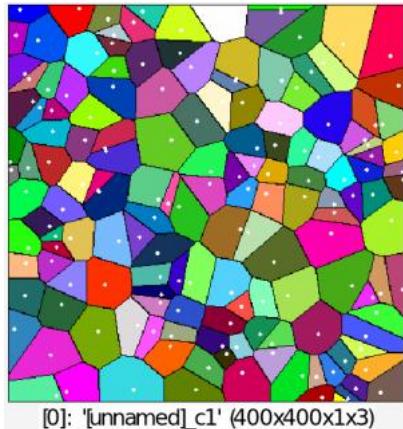
**No arguments**

**Description:**

Compute the discrete Voronoi diagram of non-zero pixels in selected images.

**Example of use:**

```
$ gmic 400,400 noise 0.2,2 eq 1 +label_fg 0 voronoi[-1] +gradient[-1]
xy,1 append[-2,-1] c norm[-1] ==[-1] 0 map[-2] 2,2 mul[-2,-1]
normalize[-2] 0,255 dilate_circ[-2] 4 reverse max
```



## wait

Built-in command

### Arguments:

- `delay` or
- `(no arg)`

### Description:

Wait for a given delay (in ms), optionally since the last call to `wait`.

or wait for a user event occurring on the selected instant display windows.

`delay` can be `{ <0=delay+flush events | 0=event | >0=delay }`.

Command selection (if any) stands for instant display window indices instead of image indices. If no window indices are specified and if `delay` is positive, the command results in a `hard` sleep during specified delay.

### Default values:

`delay=0`.

## warhol

### Arguments:

- `_M>0, _N>0, _smoothness>=0, _color>=0`

### Description:

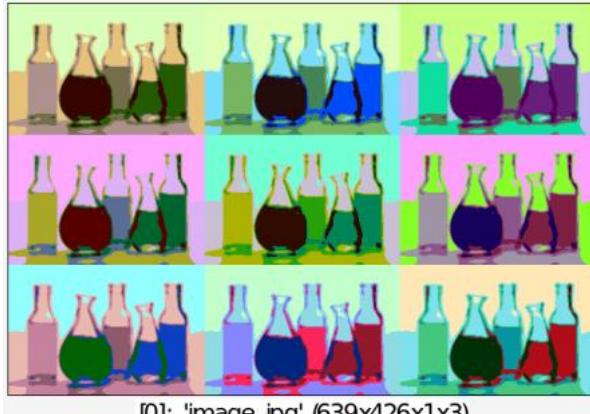
Create MxN Andy Warhol-like artwork from selected images.

## Default values:

`M=3` , `N=M` , `smoothness=2` and `color=20` .

## Example of use:

```
$ gmic image.jpg warhol 3,3,3,40
```



---

## warn

Built-in command

### Arguments:

- `_force_visible={ 0 | 1 },_message`

### Description:

Print specified warning message, on the standard error (stderr).

Command selection (if any) stands for displayed call stack subset instead of image indices.

---

## warp

Built-in command

### Arguments:

- `[warping_field],_mode,_interpolation,_boundary_conditions,_nb_frames>0`

### Description:

Warp selected images with specified displacement field.

`mode` can be `{ 0=backward-absolute | 1=backward-relative | 2=forward-absolute | 3=forward-relative }`.

`interpolation` can be `{ 0=nearest-neighbor | 1=linear | 2=cubic }`.

`boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.

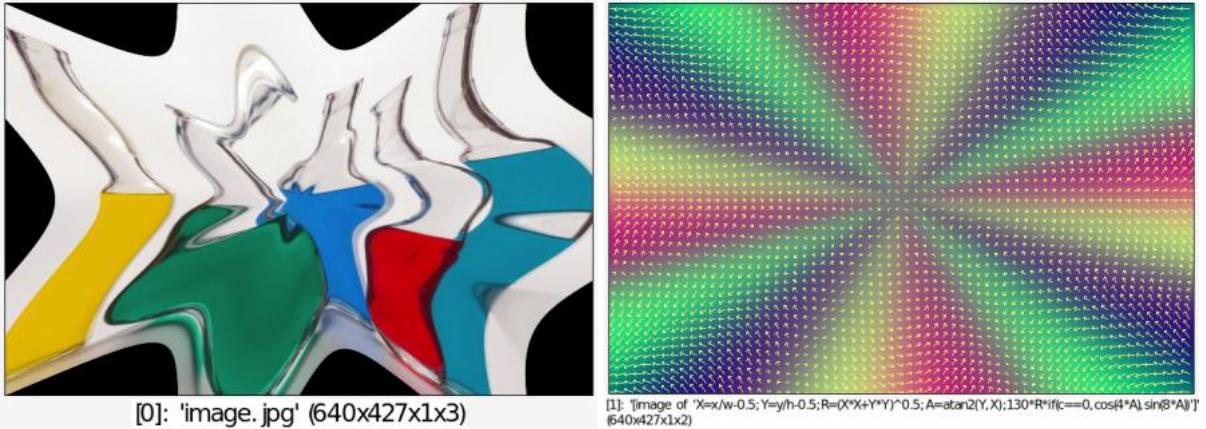
## Default values:

`mode=0`, `interpolation=1`, `boundary_conditions=1` and `nb_frames=1`.

This command has a [tutorial page](#).

## Example of use:

```
$ gmic image.jpg 100%,100%,1,2,'X=x/w-0.5;Y=y/h-0.5;R=(X*X+Y*Y)^0.5;A=atan2(Y,X);130*R*if(c==0,cos(4*A),sin(8*A))' warp[-2][-1],1,1,0 quiver[-1] [-1],10,1,1,1,100
```



---

## warp\_patch

### Arguments:

- `[warping_field],_patch_width>=1,_patch_height>=1,_patch_depth>=1,_std_factor>`

### Description:

Patch-warp selected images, with specified 2D or 3D displacement field (in backward-absolute mode).

Argument `std_factor` sets the std of the gaussian weights for the patch overlap, equal to 'std = std\_factor\*patch\_size'.

`boundary_conditions` can be { `0=dirichlet` | `1=neumann` | `2=periodic` | `3=mirror` }.

### Default values:

`std_factor=0.3` and `boundary_conditions=3`.

---

## warp\_perspective

### Arguments:

- `_x-angle,_y-angle,_zoom>0,_x-center,_y-center,_boundary_conditions={  
0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

## Description:

Warp selected images with perspective deformation.

## Default values:

`x-angle=1.5, y-angle=0, zoom=1, x-center=y-center=50` and  
`boundary_conditions=2`.

## Example of use:

```
$ gmic image.jpg warp_perspective ,
```



## warp\_rbf

### Arguments:

- `xs0[%],ys0[%],xt0[%],yt0[%],...,xsN[%],ysN[%],xtN[%],ytN[%]`

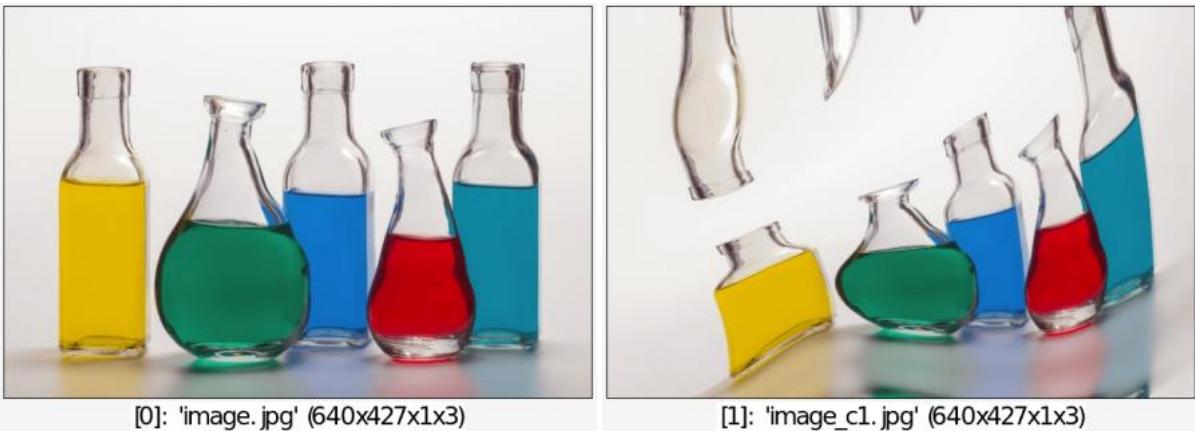
## Description:

Warp selected images using RBF-based interpolation.

Each argument (xsk,ysk)-(xtk,ytk) corresponds to the coordinates of a keypoint respectively on the source and target images. The set of all keypoints define the overall image deformation.

## Example of use:

```
$ gmic image.jpg +warp_rbf  
0,0,0,0,100%,0,100%,0,100%,100%,100%,100%,0,100%,0,100%,50%,50%,70%,50%,25%,
```



[0]: 'image.jpg' (640x427x1x3)

[1]: 'image\_c1.jpg' (640x427x1x3)

---

## water

### Arguments:

- `_amplitude, _smoothness>=0, _angle`

### Description:

Apply water deformation on selected images.

### Default values:

`amplitude=30`, `smoothness=1.5` and `angle=45`.

### Example of use:

```
$ gmic image.jpg water ,
```



[0]: 'image.jpg' (640x427x1x3)

---

## watermark\_fourier

### Arguments:

- `text, _size>0`

## Description:

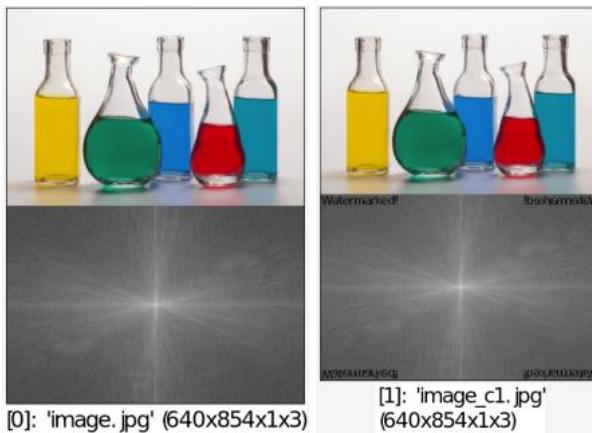
Add a textual watermark in the frequency domain of selected images.

## Default values:

`size=33`.

## Example of use:

```
$ gmic image.jpg +watermark_fourier "Watermarked!" +display_fft  
remove[-3,-1] normalize 0,255 append[-4,-2] y append[-2,-1] y
```



---

## watermark\_visible

### Arguments:

- `_text,0<_opacity<1,_size>0,_angle,_mode={ 0=remove | 1=add },_smoothness>=0`

## Description:

Add or remove a visible watermark on selected images (value range must be [0,255]).

## Default values:

'text=(c) G'MIC', `opacity=0.3`, `size=53`, `angle=25`, `mode=1` and `smoothness=0`.

## Example of use:

```
$ gmic image.jpg watermark_visible ,0.7
```



## watershed

Built-in command

### Arguments:

- `[priority_image], _is_high_connectivity={ 0 | 1 }`

### Description:

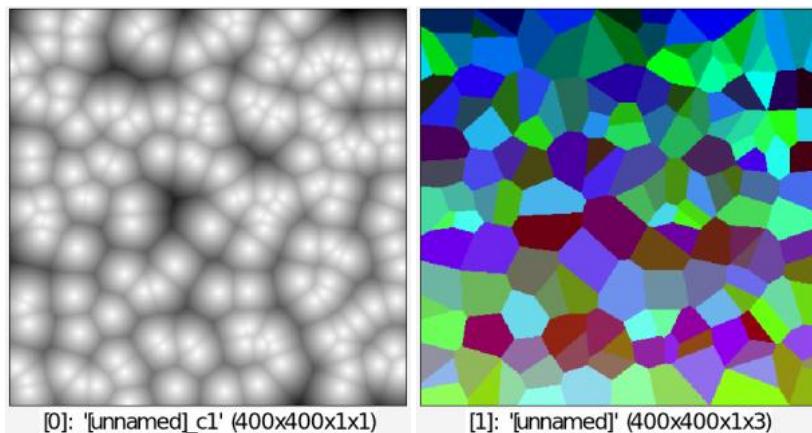
Compute the watershed transform of selected images.

### Default values:

`is_high_connectivity=1`.

### Example of use:

```
$ gmic 400,400 noise 0.2,2 eq 1 +distance 1 mul[-1] -1 label[-2]
watershed[-2] [-1] mod[-2] 256 map[-2] 0 reverse
```



## wave

### Arguments:

- `_amplitude>=0, _frequency>=0, _center_x, _center_y`

## Description:

Apply wave deformation on selected images.

## Default values:

`amplitude=4`, `frequency=0.4` and `center_x=center_y=50`.

## Example of use:

```
$ gmic image.jpg wave ,
```



## weave

### Arguments:

- `_density>=0, 0<=_thickness<=100, 0<=_shadow<=100, _shading>=0, _fibers_amplitude`

## Description:

Apply weave effect to the selected images.

`angle` can be `{ 0=0 deg. | 1=22.5 deg. | 2=45 deg. | 3=67.5 deg. }`.

## Default values:

`density=6`, `thickness=65`, `shadow=40`, `shading=0.5`, `fibers_amplitude=0`,  
`'fibers_smoothness=0'`, `angle=0` and `curvature_x=curvature_y=0`

## Example of use:

```
$ gmic image.jpg weave ,
```



---

## weird3d

### Arguments:

- `_resolution>0`

### Description:

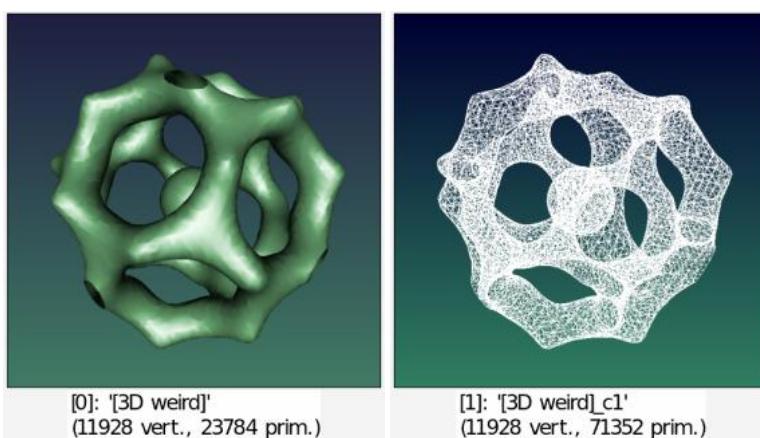
Input 3D weird object at (0,0,0), with specified resolution.

### Default values:

`resolution=32`.

### Example of use:

```
$ gmic weird3d 48 +primitives3d 1 color3d[-2] ${-rgb}
```



---

## while

Built-in command

### Arguments:

- `condition`

## Description:

End a `do...while` block and go back to associated `do` if specified condition holds.

`condition` is a mathematical expression, whose evaluation is interpreted as `{ 0=false | other=true }`.

---

# whirls

## Arguments:

- `_texture>=0, _smoothness>=0, _darkness>=0, _lightness>=0`

## Description:

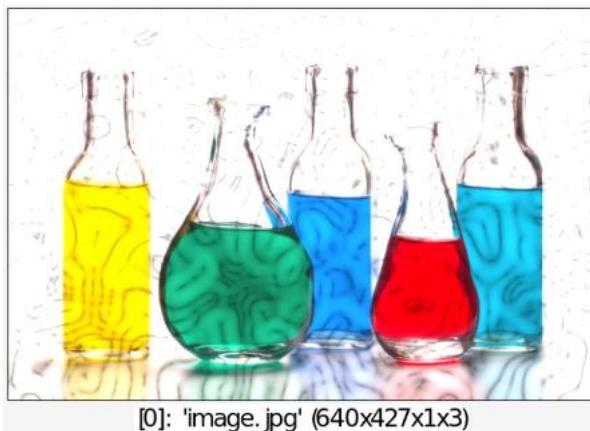
Add random whirl texture to selected images.

## Default values:

`texture=3, smoothness=6, darkness=0.5` and `lightness=1.8`.

## Example of use:

```
$ gmic image.jpg whirls ,
```



---

# wind

## Arguments:

- `_amplitude>=0, _angle, 0<=_attenuation<=1, _threshold`

## Description:

Apply wind effect on selected images.

## Default values:

`amplitude=20`, `angle=0`, `attenuation=0.7` and `threshold=20`.

## Example of use:

```
$ gmic image.jpg +wind ,
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)

## window

Built-in command

### Arguments:

- `_width[%]>=-1,_height[%]>=-1,_normalization,_fullscreen,_pos_x[%],_pos_y[%]`,

### Description:

Display selected images into an instant display window with specified size, normalization type, fullscreen mode and title.

(equivalent to shortcut command `w`).

If `width` or `height` is set to -1, the corresponding dimension is adjusted to the window or image size.

Specify `pos_x` and `pos_y` arguments only if the window has to be moved to the specified coordinates. Otherwise, they can be avoided.

'width'=0 or 'height'=0 closes the instant display 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 display windows by using the numbered variants `w0` (default, eq. to `w`), `w1`..., `w9` of the command `w`.

Invoke `window` with no selection to make the window visible, if it has been closed by the user.

## Default values:

`width=height=normalization=fullscreen=-1` and `title=(undefined)`.

---

## x\_2048

**No arguments**

**Description:**

Launch the 2048 game.

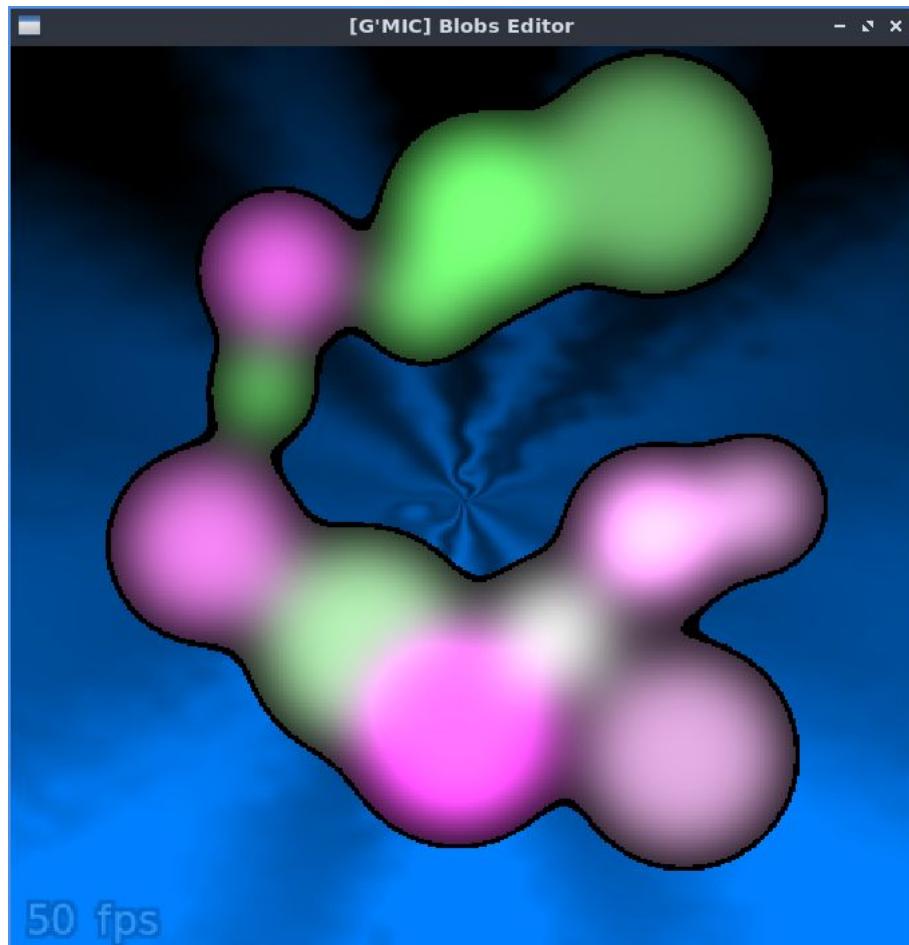
---

## x\_blobs

**No arguments**

**Description:**

Launch the blobs editor.



---

## x\_bouncing

**No arguments**

## Description:

Launch the bouncing balls demo.

---

# x\_color\_curves

## Arguments:

- `_colorspace={ rgb | cmy | cmyk | hsi | hsl | hsv | lab | lch | ycbcr | last }`

## Description:

Apply color curves on selected RGB[A] images, using an interactive window.

Set `colorspace` to `last` to apply last defined color curves without opening interactive windows.

## Default values:

`colorspace=rgb`.

---

# x\_colorize

## Arguments:

- `_is_lineart={ 0 | 1 },_max_resolution={ 0 | >=128 },_multichannels_output={ 0 | 1 },_[palette1],_[palette2],_[grabber1]`

## Description:

Colorized selected B&W images, using an interactive window.

When >0, argument `max_resolution` defines the maximal image resolution used in the interactive window.

## Default values:

`is_lineart=1`, `max_resolution=1024` and `multichannels_output=0`.

---

# x\_connect4

## No arguments

## Description:

Launch the Connect Four game.

---

## x\_crop

**No arguments**

### Description:

Crop selected images interactively.

(equivalent to shortcut command [xz](#)).

---

## x\_cut

**No arguments**

### Description:

Cut selected images interactively.

---

## x\_fire

**No arguments**

### Description:

Launch the fire effect demo.

---

## x\_fireworks

**No arguments**

### Description:

Launch the fireworks demo.

---

## x\_fisheye

**No arguments**

## Description:

Launch the fish-eye effect demo.

---

## x\_fourier

No arguments

## Description:

Launch the fourier filtering demo.

---

## x\_grab\_color

### Arguments:

- `_variable_name`

## Description:

Open a color grabber widget from the first selected image.

Argument `variable_name` specifies the variable that contains the selected color values at any time.

Assigning `-1` to it forces the interactive window to close.

### Default values:

`variable_name=xgc_variable`.

---

## x\_hanoi

No arguments

## Description:

Launch the Tower of Hanoi game.

---

## x\_histogram

No arguments

## Description:

---

Launch the histogram demo.

## x\_hough

**No arguments**

**Description:**

Launch the hough transform demo.

---

## x\_jawbreaker

**Arguments:**

- `0<_width<20,0<_height<20,0<_balls<=8`

**Description:**

Launch the Jawbreaker game.

---

## x\_landscape

**No arguments**

**Description:**

Launch the virtual landscape demo.

---

## x\_life

**No arguments**

**Description:**

Launch the game of life.

---

## x\_light

**No arguments**

**Description:**

Launch the light effect demo.

---

## x\_mandelbrot

### Arguments:

- `_julia={ 0 | 1 },_c0r,_c0i`

### Description:

Launch Mandelbrot/Julia explorer.

---

## x\_mask\_color

### Arguments:

- `_colorspace={ all | rgb | lrgb | ycbcr | lab | lch | hsv | hsi | hsl | cmy | cmyk | yiq },_spatial_tolerance>=0,_color_tolerance>=0`

### Description:

Interactively select a color, and add an alpha channel containing the corresponding color mask.

Argument `colorspace` refers to the color metric used to compute color similarities, and can be basically one of `{ rgb | lrgb | ycbcr | lab | lch | hsv | hsi | hsl | cmy | cmyk | yiq }`. You can also select one particular channel of this colorspace, by setting `colorspace` as `colorspace_channel` (e.g. `hsv_h` for the hue).

### Default values:

`colorspace=all`, `spatial_tolerance=5` and `color_tolerance=5`.

---

## x\_metaballs3d

### No arguments

### Description:

Launch the 3D metaballs demo.

---

## x\_minesweeper

## **Arguments:**

- `8<=_width=<20,8<=_height=<20`

## **Description:**

Launch the Minesweeper game.

---

## x\_minimal\_path

### No arguments

## **Description:**

Launch the minimal path demo.

---

## x\_morph

## **Arguments:**

- `_nb_frames>=2,_preview_fidelity={ 0=coarsest | 1=coarse | 2=normal | 3=fine | 4=finest }`

## **Description:**

Launch the interactive image morpher.

## **Default values:**

`nb_frames=16` and `preview_fidelity=3`.

---

## x\_pacman

### No arguments

## **Description:**

Launch pacman game.

---

## x\_paint

### No arguments

## **Description:**

Launch the interactive painter.

---

## x\_plasma

**No arguments**

## **Description:**

Launch the plasma effect demo.

---

## x\_quantize\_rgb

**Arguments:**

- `_nbcolors>=2`

## **Description:**

Launch the RGB color quantization demo.

---

## x\_reflection3d

**No arguments**

## **Description:**

Launch the 3D reflection demo.

---

## x\_rubber3d

**No arguments**

## **Description:**

Launch the 3D rubber object demo.

---

## x\_segment

**Arguments:**

- `_max_resolution={ 0 | >=128 }`

## Description:

Segment foreground from background in selected opaque RGB images, interactively.

Return RGBA images with binary alpha-channels.

## Default values:

`max_resolution=1024`.

---

# x\_select\_color

## Arguments:

- `_variable_name`

## Description:

Display a RGB or RGBA color selector.

Argument `variable_name` specifies the variable that contains the selected color values (as R,G,B, [A])

at any time.

Its value specifies the initial selected color. Assigning `-1` to it forces the interactive window to close.

## Default values:

`variable_name=xsc_variable`.

---

# x\_select\_function1d

## Arguments:

- `_variable_name,_background_curve_R,_background_curve_G,_background_curve_B`

## Description:

Open an interactive window, where the user can defined its own 1D function.

If an image is selected, it is used to display additional information :

- The first row defines the values of a background curve displayed on the window (e.g. an histogram).
- The 2nd, 3rd and 4th rows define the R,G,B color components displayed beside the X and Y axes.

Argument `variable_name` specifies the variable that contains the selected function keypoints at any time.

Assigning `-1` to it forces the interactive window to close.

## Default values:

```
variable_name=xsf_variable, background_curve_R=220,  
background_curve_G=background_curve_B=background_curve_T.
```

---

# x\_select\_palette

## Arguments:

- `_variable_name, _number_of_columns={ 0=auto | >0 }`

## Description:

Open a RGB or RGBA color selector widget from a palette.

The palette is given as a selected image.

Argument `variable_name` specifies the variable that contains the selected color values (as R,G,B, [A])

at any time.

Assigning `-1` to it forces the interactive window to close.

## Default values:

```
variable_name=xsp_variable and number_of_columns=2.
```

---

# x\_shadebobs

## No arguments

## Description:

Launch the shade bobs demo.

---

# x\_spline

## No arguments

## Description:

Launch spline curve editor.

---

## x\_starfield3d

**No arguments**

**Description:**

Launch the 3D starfield demo.

---

## x\_tetris

**No arguments**

**Description:**

Launch tetris game.

---

## x\_threshold

**No arguments**

**Description:**

Threshold selected images interactively.

---

## x\_tictactoe

**No arguments**

**Description:**

Launch tic-tac-toe game.

---

## x\_warp

**Arguments:**

- `_nb_keypoints_xgrid>=2,_nb_keypoints_ygrid>=2,_nb_keypoints_contours>=0,_pre { 0=coarsest | 1=coarse | 2=normal | 3=fine | 4=finest } ,_[background_image],0<=_background_opacity<=1`

## Description:

Launch the interactive image warper.

## Default values:

`nb_keypoints_xgrid=nb_keypoints_ygrid=2`, `nb_keypoints_contours=0` and  
`preview_fidelity=1`.

---

## x\_waves

### No arguments

## Description:

Launch the image waves demo.

---

## x\_whirl

### Arguments:

- `_opacity>=0`

## Description:

Launch the fractal whirls demo.

## Default values:

`opacity=0.2`.

---

## xor

Built-in command

### Arguments:

- `value[%]` or
- `[image]` or
- `'formula'` or
- `(no arg)`

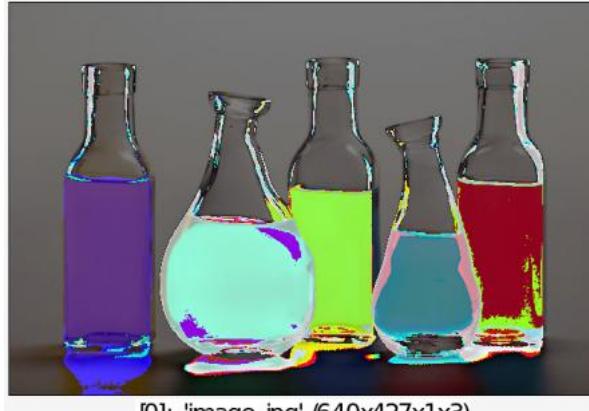
## Description:

Compute the bitwise XOR of selected images with specified value, image or mathematical expression, or compute the pointwise sequential bitwise XOR of selected images.

## Examples of use:

- Example #1

```
$ gmic image.jpg xor 128
```



[0]: 'image.jpg' (640x427x1x3)

- Example #2

```
$ gmic image.jpg +mirror x xor
```



[0]: 'image.jpg' (640x427x1x3)

---

## xyz2jzazbz

### No arguments

### Description:

Convert color representation of selected images from XYZ to RGB.

---

## xyz2lab

### Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

## Description:

Convert color representation of selected images from XYZ to Lab.

## Default values:

`illuminant=2`.

---

# xyz2rgb

## Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

## Description:

Convert color representation of selected images from XYZ to RGB.

## Default values:

`illuminant=2`.

---

# xyz82rgb

## Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }` or
- `(no arg)`

## Description:

Convert color representation of selected images from XYZ8 to RGB.

## Default values:

`illuminant=2`.

---

# ycbcr2rgb

## No arguments

## Description:

Convert color representation of selected images from YCbCr to RGB.

---

# yinyang

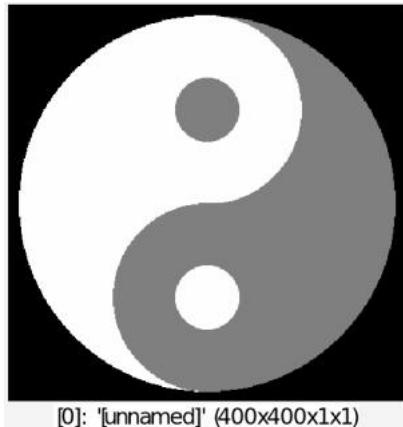
No arguments

## Description:

Draw a yin-yang symbol on selected images.

## Example of use:

```
$ gmic 400,400 yinyang
```



# yiq2rgb

No arguments

## Description:

Convert color representation of selected images from YIQ to RGB.

---

# yiq82rgb

No arguments

## Description:

Convert color representation of selected images from YIQ8 to RGB.

---

## yuv2rgb

**No arguments**

### Description:

Convert color representation of selected images from YUV to RGB.

---

## yuv82rgb

**No arguments**

### Description:

Convert selected images from YUV8 to RGB color bases.

---

## zoom

### Arguments:

- `_factor, _cx, _cy, _cz, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

### Description:

Apply zoom factor to selected images.

### Default values:

`factor=1`, `cx=cy=cz=0.5` and `boundary_conditions=0`.

### Example of use:

```
$ gmic image.jpg +zoom[0] 0.6 +zoom[0] 1.5
```



[0]: 'image.jpg' (640x427x1x3)



[1]: 'image\_c1.jpg' (640x427x1x3)



[2]: 'image\_c1.jpg' (640x427x1x3)

---

□ End of document