



The Handbook

Version 1.6.5.1 (pre-release #062515)

David Tschumperlé

June 25, 2015

Contents

1	Usage	7
1.1	Overall context	7
1.2	Image definition and terminology	7
1.3	Items of a processing pipeline	8
1.4	Input data items	8
1.5	Command items and selections	9
1.6	Input/output properties	10
1.7	Substitution rules	11
1.8	Mathematical expressions	13
1.9	Image and data viewers	15
1.10	Adding custom commands	16
1.11	List of commands	17
2	List of commands	19
2.1	Global options	19
2.2	Input/output	19
2.3	List manipulation	44
2.4	Mathematical operators	50
2.5	Values manipulation	92
2.6	Colors manipulation	118
2.7	Geometry manipulation	141
2.8	Filtering	171
2.9	Features extraction	227
2.10	Image drawing	247
2.11	Matrix computation	274
2.12	3d rendering	277
2.13	Program control	329
2.14	Arrays, tiles and frames	341
2.15	Artistic	357
2.16	Warpings	378
2.17	Degradations	389
2.18	Blending and fading	396
2.19	Image sequences and videos	405

2.20	PINK-library operators	410
2.21	Convenience functions	418
2.22	Other interactive commands	428
2.23	Commands shortcuts	433
2.24	Examples of use	435

Preamble

License

This document is distributed under the **GNU Free Documentation License**, version 1.3.
Read the full license terms at <http://www.gnu.org/licenses/fdl-1.3.txt>.

An online version of this documentation is available at:
<http://gmick.eu/reference.shtml>.

Motivations

G'MIC is an open and full-featured framework for image processing, providing several different user interfaces to convert/manipulate/filter/visualize generic image datasets, from 1d scalar signals to 3d+t sequences of multi-spectral volumetric images. Technically speaking, what it does is:

- Define a lightweight but powerful script language (the G'MIC language) dedicated to the design of image processing pipelines.
- Provide several user interfaces embedding the corresponding interpreter:
 - A command-line executable 'gmick', to use the G'MIC framework from a shell. In this setting, G'MIC may be seen as a direct (and friendly) competitor of the ImageMagick or GraphicsMagick software suites.
 - A plug-in 'gmick_gimp', to bring G'MIC capabilities to the GIMP image retouching software.
 - A web-service 'G'MIC Online', to allow users applying image processing algorithms directly in a web browser.
 - A Qt-based interface 'ZArt', for real-time manipulation of webcam images.
 - A C++ library 'libgmick', to be linked with third-party applications.

G'MIC is focused on the design of possibly complex pipelines for converting, manipulating, filtering and visualizing generic 1d/2d/3d multi-spectral image datasets. This includes of course color images, but also more complex data as image sequences or 3d(+t) volumetric float-valued

datasets.

G'MIC is an open framework: the default language can be extended with custom G'MIC-written commands, defining thus new available image filters or effects. By the way, G'MIC already contains a substantial set of pre-defined image processing algorithms and pipelines (more than 1000).

G'MIC has been designed with portability in mind and runs on different platforms (Windows, Unix, MacOSX). It is distributed under the CeCILL license (GPL-compatible). Since 2008, it is developed in the Image Team of the GREYC laboratory, in Caen/France, by permanent researchers working in the field of image processing on a daily basis.

Version

gmic: GREYC's Magic for Image Computing.

Version **1.6.5.1 (pre-release #062515)**, Copyright (c) 2008-2015, David Tschumperlé
(<http://gmic.eu>)

Chapter 1

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 image processing pipelines. 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 documentation proposes a technical description of the G'MIC language basics and rules.

1.1 Overall context

- At any time, G'MIC manages one list of numbered (and optionally named) pixel-based images, entirely stored in computer memory.
- The first image of the list has indice '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 indicated by '[name]', if 'name' uses the characters 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 commands '--name' and '--input' for this purpose).
- G'MIC defines a set of various commands and substitution mechanisms to allow the design of complex pipelines managing this list of images, in a very flexible way:
You can insert or remove images in the list, rearrange image indices, process images (individually or as a group), merge image data together and output image files.
- Such a pipeline can be written itself as a custom G'MIC command storable in a user command file, and can be re-used afterwards in another bigger pipeline if necessary.

1.2 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 limitations on the size of each image dimension. For instance, the number of image slices or channels can be of arbitrary size within the limits of 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 '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.
- 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.

1.3 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 output.jpg
```

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

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

1.4 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.
- 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 selection' to learn more about selections).
 - . 'width[%],_height[%],_depth[%],_spectrum[%],_values': Insert a new image 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,if(c==0,x,if(c==1,y,0))' which creates a 256x256 RGB color image with a spatial shading on the red and green channels.

- . '(v1,v2,...)': Insert a new image 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.
 - . '0': Insert a new 'empty' image, containing no pixel data. Empty images are used only in rare occasions.
- Input item 'name=value' declares a new local or global variable 'name', or assign a new value to an existing variable. Variable names use characters 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'). Otherwise, it remains local to the thread that defined it.

1.5 Command items and selections

- A G'MIC item starting by '-' 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[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[0–1:2]': 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 two firsts.
 - . '-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. 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]’.
- 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]’).
- A G’MIC command starting with a double dash ‘–’ (instead of a single dash ‘-’) 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:
 - . Native 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 native command is not permitted, except if all following arguments are also omitted. For instance, call to ‘–plasma 10,,5’ is invalid but ‘–plasma 10’ is correct.
 - . Custom commands, are defined as G’MIC pipelines of native or custom commands.
 - They are interpreted by the G’MIC interpreter, and thus run a bit slower than native commands.
 - But 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 custom commands.
- A user may easily add its own custom commands to the G’MIC interpreter (see section ‘Adding custom commands’ for more details). New native commands cannot be added (unless you modify the G’MIC interpreter source code).

1.6 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, .nii, .pan, .inr, .pnk, ..
 - . video files: .mpeg, .avi, .mov, .ogg, .flv, ..
 - . Generic ascii or binary data files: .gmz, .cimg, .cimgz, .dlm, .asc, .pfm, .raw, .txt, .h.
 - . 3d object files: .off.
- When dealing with color images, G’MIC generally reads, writes and displays data using the usual sRGB color space.
- 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 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 representations. 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 .cimgz or .gmz file extensions to ensure that all data precision are preserved when saving images.
- File options can/must be set for these specific 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={ 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 { uchar | char | ushort | short | uint | int | ulong | long | 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]]]'. 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 { uchar | char | ushort | short | uint | int | ulong | long | float | double }.
- .yuv files: Image dimensions must be specified, and only sub-frames of an image sequence may be loaded, using the input expression 'filename.yuv,width,height,[first_frame],[last_frame],[step]]]'.
- .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]]' can be used to specify the output pixel type, as well as the compression method.
- 'datatype' can be { uchar | char | ushort | short | uint | int | ulong | long | float | double }.
- 'compression' can be { none | lzw | jpeg } and 'force_multipage' can be { 0=no | 1=yes }.
- .gif files: Animated gif files can be saved, using the input expression 'filename.gif,fps,nb_loops'.
- Specify 'nb_loops=0' to get an infinite number of animation loops.
- .jpeg files: The output quality may be specified (in %), using the output expression 'filename.jpg,30' (here, to get a 30% quality output).
- .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 { uchar | char | ushort | short | uint | int | ulong | long | float | double }.
- .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={ 0 | 1 }'.
- . 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.

- Some input/output formats and options may not be supported by your current version of 'gmic', depending on the configuration flags set during the build of the 'gmic' binaries.

1.7 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 indice 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 an empty string).

The following variables are predefined by the G'MIC interpreter:

- . '\$!': The current number of images in the list.
 - . '\$>' and '\$<': The increasing/decreasing indice of the latest (currently running) 'repeat..done' loop.
 - . '\$/': 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.
 - . '\$_pid': The current process identifier, as an integer.
 - . '\$_prerelease': For pre-releases only, the date of the pre-release as mmddyy. For stable releases, this variable is not defined.
 - . '\$_version': A 4-digits number telling about the current version of the G'MIC interpreter. (currently '1.6.5.1').
 - . '\$_vt100': Set to 1 (default value) if colored text output is allowed on the console.
- '\${"command line"}' is substituted by the status value set by the execution of the specified command line (see command '-status' to learn more about status). Expression '\${' thus stands for the current status value.
 - '{string}' (single quotes) is substituted by the sequence of ascii codes that compose the specified string, separated by commas ','. For instance, item '{foo}' is substituted by '102,111,111'.
 - '{value1,...,valueN}' (backquotes) is substituted by the string whose ascii sequence corresponds to the list of values specified between the backquotes. For instance, item '{102,111,111}' is substituted by 'foo'.
 - '{string1=='string2}' is substituted by 0 or 1, whether the two specified strings coincide or not (test is case-sensitive).
 - '{string1!='string2}' is substituted by 0 or a non-null integer, whether the two strings beside the operator are different or not (test is case-sensitive). For instance, item '{foo!='foo}' is substituted by '0' and '{foo!='FOO}' by '32'.
 - '{image,feature}' 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).
 - . 'c': The (x,y,z,c) coordinates of the minimum value, separated by commas ','.
 - . 'C': The (x,y,z,c) coordinates of the maximum value, separated by commas ','.
 - . '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 ascii codes.
 - . 'x': The image extension (i.e last 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 '3.141592653589793').

- ' $\{*\}$ ' is substituted by the visibility state of the instant display window [0] (can be { 0=closed | 1=visible }).
- ' $\{*,feature\}$ ' or ' $\{*indice,feature\}$ ' is substituted by a specific feature of the instant display window #0 (or #indice, if specified). Requested 'feature' can be:
 - . '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.
 - . '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.
 - . 'u': screen width (actually independent on the window size).
 - . 'v': screen height (actually independent on the window size).
 - . 'uv': screen width x screen height.
 - . 'n': current normalization type 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 dash '-' 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 " $3+8 \text{ kg} = \{3+8\} \text{ kg}$ ". 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'.

1.8 Mathematical expressions

- G'MIC has an embedded mathematical parser. It is used to evaluate expressions inside braces ' $\{\}$ ', or formulas in commands that may take one as an argument (e.g. '-fill').
- When used as a command argument, a formula is evaluated for each pixel of the selected images.
- 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).
 - _ Usual functions: sin(), cos(), tan(), asin(), acos(), atan(), sinh(), cosh(), tanh(),

`log()`, `log2()`, `log10()`, `exp()`, `sign()`, `abs()`, `atan2()`, `round()`, `narg()`, `arg()`,
`isval()`, `isnan()`, `isinf()`, `isint()`, `isbool()`, `isdir()`, `isfile()`, `rol()` (left bit rotation),
`ror()` (right bit rotation), `min()`, `max()`, `med()`, `kth()`, `sinc()`, `int()`.
 . Function '`atan2()`' is the version of '`atan()`' with two arguments '`y`' and '`x`' (as in C/C++).
 . Function '`narg()`' returns the number of specified arguments.
 . Function '`arg(i,a_1,...,a_n)`' returns the *i*th argument `a_i`.
 . Functions '`min()`', '`max()`', '`med()`' and '`kth()`' can be called with an arbitrary number of arguments.
 . Functions '`isval()`', '`isnan()`', '`isinf()`', '`isbool()`' can be used to test the type of
 a given number or expression.
 . Function '`isfile()`' (resp. '`isdir()`') returns 0 (false) or 1 (true) whether its argument
 is a valid path to a file (resp. to a directory) or not.
 . Function '`fdate(path,attr)`' 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 }.
 . Function '`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`'.

_ Variable names below are pre-defined. They can be overloaded.

. '`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*height.
 . '`whd`': shortcut for width*height*depth.
 . '`whds`': shortcut for width*height*depth*spectrum (i.e. total number of pixel values).
 . '`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).
 . '`e`': value of *e*, i.e. 2.71828..
 . '`pi`': value of *pi*, i.e. 3.1415926..
 . '`?`' or '`u`': a random value between [0,1], following a uniform distribution.
 . '`g`': a random value, following a gaussian distribution of variance 1
 (roughly in [-5,5]).
 . '`i`': current processed pixel value (i.e. value located at (x,y,z,c)) of the
 associated image, if any (0 otherwise).
 . '`im`', '`iM`', '`ia`', '`iv`', '`is`', '`ip`', '`ic`': Respectively the minimum, maximum, average values,
 variance, sum, product and median value 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).

_ Special operators can be used:

. '`;`': 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. Variable names are case-sensitive. Use this operator in
 conjunction with '`;`' to define 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.

– The following specific functions are also defined:

- . `'if(expr_cond,expr_then,expr_else)'`: return value of `'expr_then'` or `'expr_else'`, depending on the value of `'expr_cond'` (0=false, other=true). For instance, G'MIC command `'–fill if(x%10==0,255,i)'` will draw blank vertical lines on every 10th column of an image.
- . `'?(max)'` or `'?(min,max)'`: return a random value between `[0,max]` or `[min,max]`, following a uniform distribution. `'u(max)'` and `'u(0,max)'` mean the same.
- . `'i(_a,_b,_c,_d,_interpolation,_boundary)'`: return the value of the pixel located at position (a,b,c,d) in the associated image, if any (0 otherwise). Interpolation parameter can be { 0=nearest neighbor | other=linear }. Boundary conditions can be { 0=dirichlet | 1=neumann | 2=periodic }. Omitted coordinates are replaced by their default values which are respectively x, y, z, c and 0.
- . `'j(_dx,_dy,_dz,_dc,_interpolation,_boundary)'`: does the same for the pixel located at position (x+dx,y+dy,z+dz,c+dc).
- . `'i[offset]'`: return the value of the pixel located at specified offset in the associated image buffer.
- . `'j[offset]'`: does the same for an offset relative to the current pixel (x,y,z,c). 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.
- . If specified formula starts with `'>'` or `'<'`, the operators `'i(..)'` and `'j(..)'` will return values of the image currently being modified, in forward (`'>'`) or backward (`'<'`) order.

- The last image of the list is always associated to the evaluations of `'{expressions}'`, e.g. G'MIC sequence `'256,128 –f {w}'` will create a 256x128 image filled with value 256.

1.9 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 objects (command `'–display3d'`). It enables an interactive view of the selected image data.
- The following keyboard shortcuts are available in the interactive viewers:
 - . CTRL+D: Increase window size.
 - . CTRL+C: Decrease window size.
 - . CTRL+R: Reset window size.
 - . CTRL+F: Toggle fullscreen mode.
 - . CTRL+S: Save current window snapshot as numbered file `'gmic_xxxx.bmp'`.
 - . CTRL+O: Save current instance of the viewed data, as numbered file `'gmic_xxxx.cimgz'`.
- Shortcuts specific to the 1d/2d/3d image viewer (command `'–display'`) are:
 - . CTRL+A: Switch cursor mode.
 - . CTRL+P: Play z–stack of frames as a movie (for volumetric 3d images).
 - . CTRL+V: Show/hide 3D view (for volumetric 3d images).
 - . CTRL+(mousewheel): Zoom in/out.
 - . SHIFT+(mousewheel): Go left/right.

- . ALT+(mousewheel): Go up/down.
- . Numeric PAD: Zoom in/out (+/−) and move through zoomed image (digits).
- . BACKSPACE: Reset zoom scale.
- Shortcuts 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.
 - . (mousewheel): Zoom in/out.
 - . CTRL+F1 .. CTRL+F6: Toggle between different 3d rendering modes.
 - . CTRL+Z: Enable/disable z—buffered rendering.
 - . CTRL+A: Show/hide 3d axes.
 - . CTRL+G: Save 3d object, as numbered file `'gmic_xxxx.off'`.
 - . CTRL+T: Switch between single/double—sided 3d modes.

1.10 Adding custom commands

- Custom commands can be defined by a user, through the use of G'MIC custom commands files.
- A command file is a simple ascii 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 `$HOME/.gmic` (on Unix) or `%APPDATA%/gmic` (on Windows), and runs command `'—start'` if defined.
- Custom command names must use characters `[a-zA-Z0-9_]` and cannot start with a number.
- Any `#` comment expression found in a custom commands file is discarded by the G'MIC parser, wherever it is located in a line.
- In a custom command, the following `$`—expressions are 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.
 - . `'$#'` is substituted by the maximum indice of known arguments (either specified by the user or set to a default value in the custom command).
 - . `'$?'` is substituted by a string telling about the command subset restriction (only useful when custom commands need to output descriptive messages).
 - . `'$i'` and `'${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).
 - . `'${subset}'` is substituted by the arguments values (separated by commas `,`) of a specified argument subset. For instance expression `'${2-2}'` is substituted by all specified arguments except the first and the last one. Expression `'${^0}'` is then substituted by all arguments of the invoked command (eq. to `'$*'` if all specified arguments have indeed a value).
 - . `'$=var'` is substituted by the set of instructions that will assign each argument `$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 characters `[a-zA-Z0-9_]` and cannot start with a number.
- These particular `$`—expressions are always substituted, even in double quoted items or when the dollar sign `'$'` 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 does not have a value, an error is thrown by the interpreter.

1.11 List of commands

All available G'MIC commands are listed below, classified by themes.

When several choices of command arguments are possible, they appear separated by '|'.

An argument specified inside '[]' or starting by '' is optional except when standing for an existing image [image], where 'image' can be either an indice number or an image name.

In this case, the '[]' characters are mandatory when writing the item.

A command marked with '(+)' is a native command.

Note also that all images that serve as illustrations in this reference documentation are normalized in [0,255] before being displayed. You may need to do this manually (command '—normalize 0,255') if you want to save image files having the same aspect than those illustrated in the example codes.

Chapter 2

List of commands

2.1 Global options

2.1.1 *-debug* (+)

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 can be useful when debugging the execution of a custom command.

2.1.2 *-help*

Arguments: `command` |
`(no arg)`

Display help (optionally for specified command only) and exit.
(*eq. to ' -h '*).

2.1.3 *-version*

Display current version number.

2.2 Input/output

2.2.1 *-camera* (+)

Arguments: `_camera_index>=0, _nb_frames>0, _skip_frames>=0, _capture_width->=0, _capture_height>=0`

Insert one or several frames from specified camera, with custom delay between frames (in ms).

When `'nb_frames==0'`, the camera stream is released instead of capturing new images.

Default values: `'camera.index=0'` (default camera), `'nb_frames=1'`, `'skip_frames=0'` and `'capture_width=capture_height=0'` (default size).

2.2.2 *-command (+)*

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

Import G'MIC custom commands from specified file, URL or string.
(*eq. to* `'-m'`).

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

Default value: `'add_debug_info=1'`.

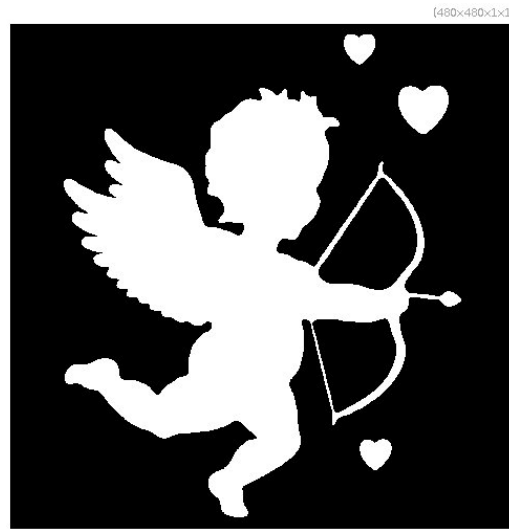


Example 1 : `image.jpg -command "foo : -mirror y -deform $"1" --foo[0] 5 --foo[0] 15`

2.2.3 *-cupid*

Arguments: `_size>0`

Input cupid binary mask with specified size.



Example 2 : `-cupid ,`

2.2.4 `-cursor (+)`

Arguments: `_mode = { 0=hide | 1=show }`

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 value: `'mode=1'` .

2.2.5 `-display (+)`

Arguments: `_X, _Y, _Z`

Display selected images in an interactive viewer (use the instant display window [0] if opened).

Arguments 'X','Y','Z' determine the initial selection view, for 3d volumetric images.
(eq. to `'-d'`).

Tutorial page:

http://gmics.eu/tutorial/_display.shtml

2.2.6 `-display0`

Display selected images without value normalization.
(eq. to `'-d0'`).

2.2.7 *-display3d (+)*

Arguments: `_[background_image]`

Display selected 3d objects in an interactive viewer (use the instant display window [0] if opened).
(eq. to `'-d3d'`).

Default value: `'[background_image]=(default)'`.

2.2.8 *-display_array*

Arguments: `_width>0, _height>0`

Display images in interactive windows where pixel neighborhoods can be explored.

Default values: `'width=13'` and `'height=width'`.

2.2.9 *-display_fft*

Display fourier transform of selected images, with centered log-module and argument.
(eq. to `'-dfft'`).



Example 3 : `image.jpg --display_fft`

2.2.10 *-display_graph*

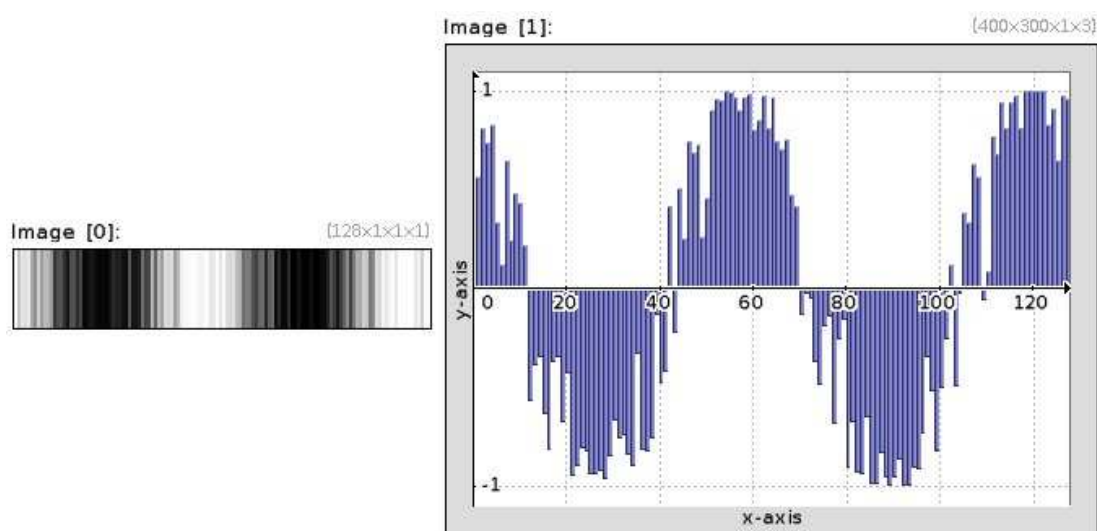
Arguments: `_width>32, _height>32, _plot_type, _vertex_type, _xmin, _xmax, _ymin, _ymax, _xlabel, _ylabel`

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.

Default values: 'width=640', 'height=480', 'plot_type=1',
'vertex.type=1', 'xmin=xmax=ymin=ymax=0 (auto)', 'xlabel="x-axis"'
and 'ylabel="y-axis"'.



Example 4 : 128,1,1,1,1,'cos(x/10+?)' --display_graph 400,300,3

2.2.11 -display_histogram

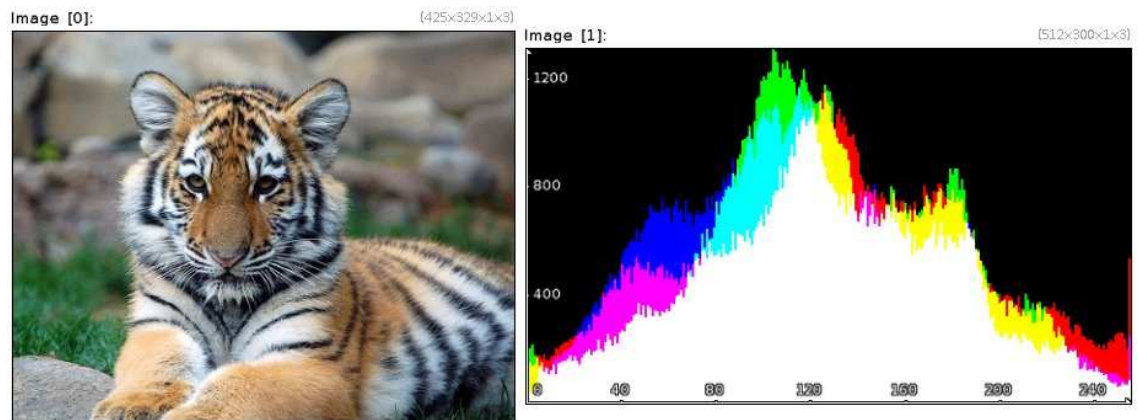
Arguments: _width>0, _height>0, _clusters>0, _min.value[%], _max.value[%],
_show_axes={ 0 | 1 }, _expression.

Render a channel-by-channel histogram.

If selected image has several slices, the rendering is performed
for all input slices.

'expression' is a mathematical expression used to transform the
histogram data for visualization purpose.
(eq. to '-dh').

Default values: 'width=512', 'height=300', 'clusters=256',
'min.value=0%', 'max.value=100%', 'show_axes=1' and 'expression=i'.



Example 5 : `image.jpg --display.histogram 512,300`

2.2.12 *-display_parametric*

Arguments: `_width>0, _height>0, _outline_opacity, _vertex_radius>=0, _is_antialiased={ 0 | 1 }, _is_decorated={ 0 | 1 }, _xlabel, _ylabel`

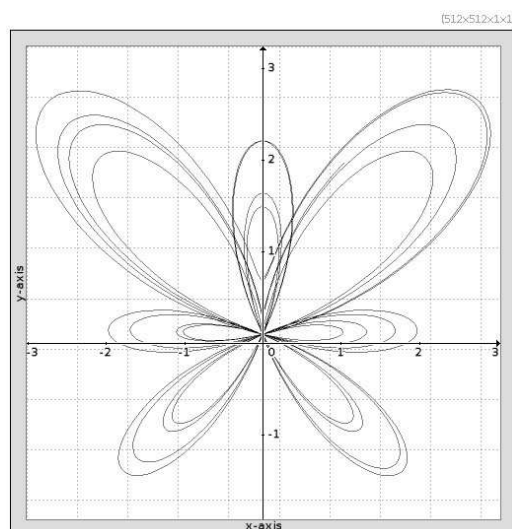
Render 2d or 3d parametric curve or point clouds from selected image data.

Curve points are defined as pixels of a 2 or 3-channel image.

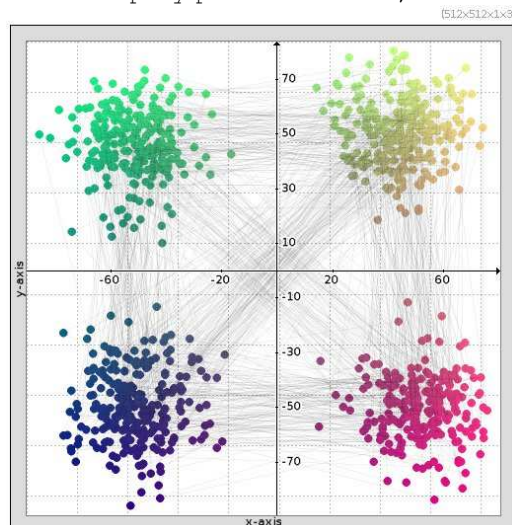
If the point image contains more than 3 channels, additional channels define the (R,G,B) color for each vertex.

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"'.`

**Example 6 :**

```
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
```



Example 7 : 1000,1,1,2,?(-100,100) -quantize 4,1 -noise 12 -channels 0,2
--normalize 0,255 -append c -display_parametric 512,512,0.1,8

2.2.13 -display_polar

Arguments: _width>32, _height>32, _outline_type, _fill_R, _fill_G, _fill_B, _theta_start, _theta_end, _xlabel, _ylabel

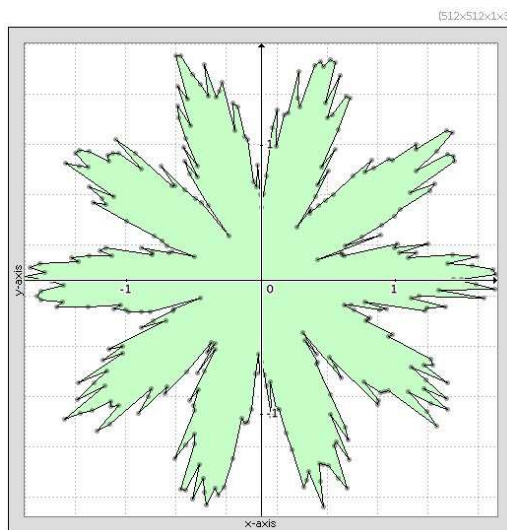
Render polar curve from selected image data.

(eq. to '-dp').

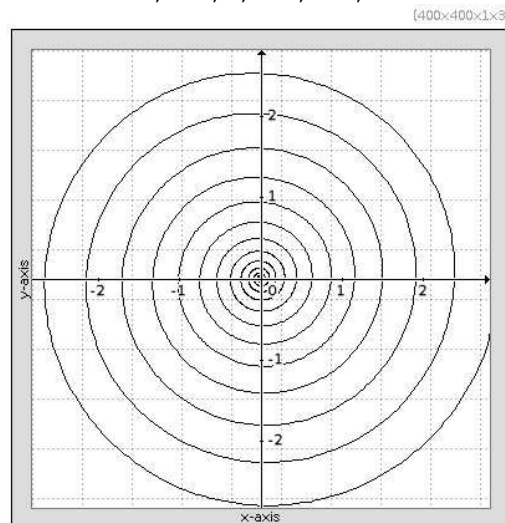
'outline.type' can be { r<0=dots with radius -r | 0=no outline

| $r > 0 = \text{lines+dots with radius } r$ }.
 '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"'.



Example 8 : `300,1,1,1,'0.3+abs(cos(10*pi*x/w))+?(0.4)'` -display_polar
 512,512,4,200,255,200



Example 9 : `3000,1,1,1,'x^3/1e10'` -display_polar 400,400,1,-1,,,0,{15*360}

2.2.14 *-display_rgba*

Render selected RGBA images over a checkerboard background.
(eq. to `'-drgba'`).



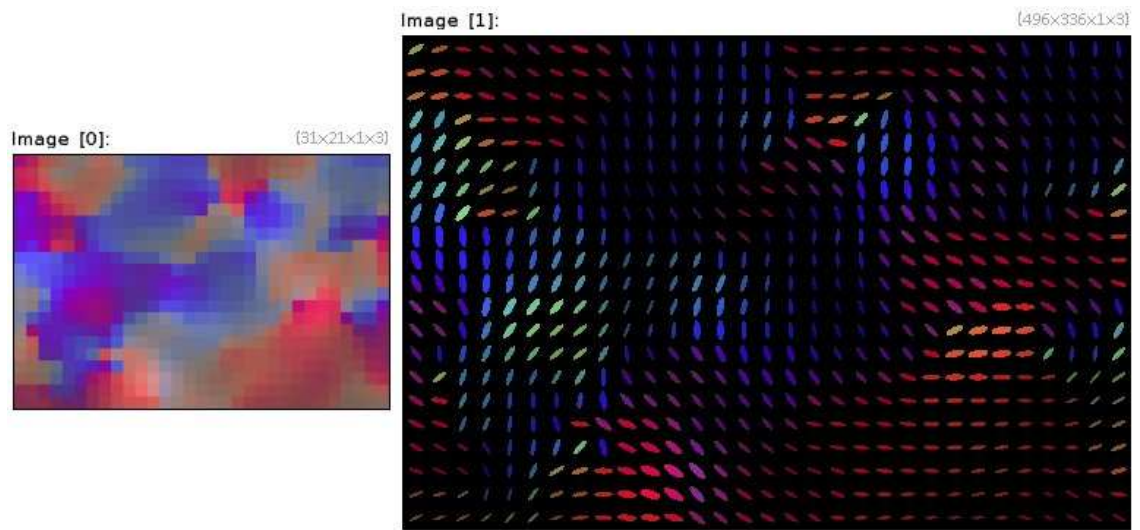
Example 10 : `image.jpg --norm -threshold[-1] 40% -blur[-1] 3 -normalize[-1] 0,255 -append c -display_rgba`

2.2.15 *-display_tensors*

Arguments: `_size_factor>0, _ellipse_factor>=0, _colored_mode={ 0 | 1 }`

Render selected mask field of 2x2 tensors with ellipses.
(eq. to `'-dt'`).

Default values: `'size_factor=16', 'ellipse_factor=0.92', 'color_mode=1'`.



Example 11 : `image.jpg -diffusontensors 0.7,0.6 -crop 60,10,90,30
--display_tensors ,`

Tutorial page:

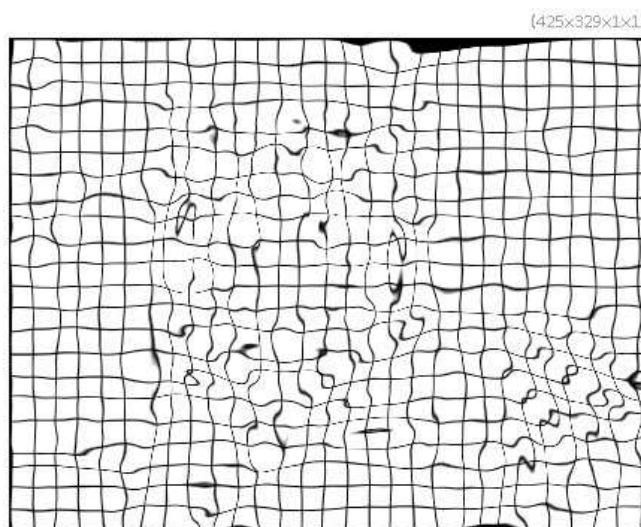
http://gmic.eu/tutorial/_display_tensors.shtml

2.2.16 *-display_warp*

Arguments: `_cell_size>0`

Render selected 2d warping fields.
(eq. to `'-dw'`).

Default value: `'cell_size=15'`.



Example 12 : `image.jpg -luminance -blur 5 -gradient -append c -display-warp ,`

2.2.17 *-document gmic*

Arguments: `_format={ ascii | bash | html | images | latex
,_image_path,_write_wrapper={ 0 | 1 }`

Create documentation of .gmic command files (loaded as raw 'uchar' images), in specified format.

Default values: `'format=ascii', 'image_path=""` and `'write_wrapper=1'.\n`

Example(s) : `raw:filename.gmic,char -document-gmic html,img`

2.2.18 *-echo (+)*

Arguments: `message`

Output specified message, on the error output.
(eq. to '-e').

Command selection (if any) stands for displayed call stack subset instead of image indices.

2.2.19 *-echo file*

Arguments: `filename,message`

Output specified message, appending it to specified output

file.
(similar to `'-echo'` for specified output file stream).

2.2.20 `-echo_stdout`

Arguments: message

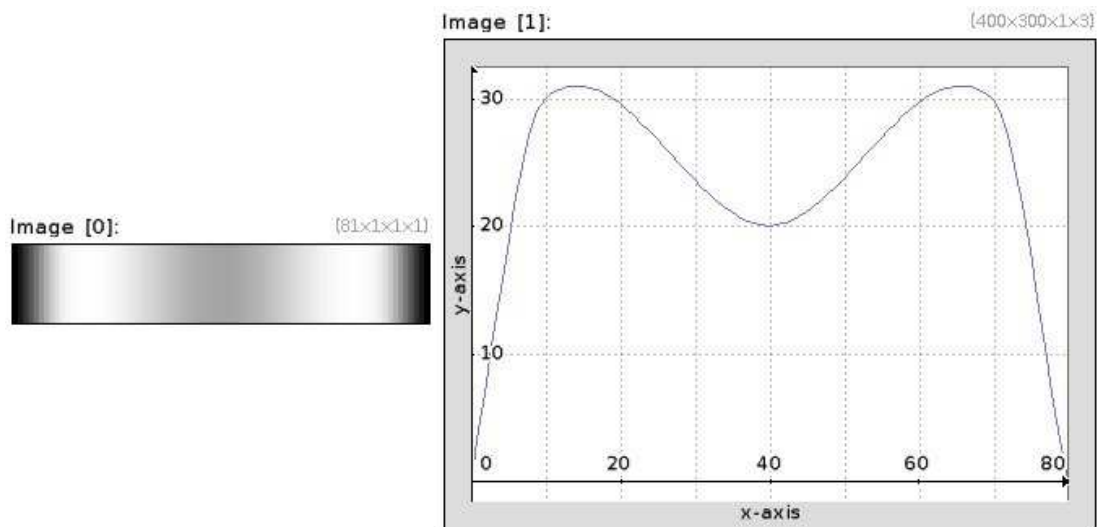
Output specified message, on the standard output (stdout).
(similar to `'-echo'` for output on standard output instead of standard error).

2.2.21 `-function1d`

Arguments: $0 \leq \text{smoothness} \leq 1, x_0 \geq 0, y_0, x_1 \geq 0, y_1, \dots, x_n \geq 0, y_n$

Input continuous 1d function from specified list of keypoints (x_k, y_k) in range $[0, \max(x_k)]$ (x_k are positive integers).

Default values: `'smoothness=1'` and `'x0=y0=0'`.



Example 13 : `-function1d 1,0,0,10,30,40,20,70,30,80,0 --display-graph 400,300`

2.2.22 `-gmicky`

Load a new image of the G'MIC mascot 'Gmicky'.



Example 14 : `-gmicky`

2.2.23 `-gmicky_wilber`

Load a new image of the G'MIC mascot 'Gmicky' together with GIMP mascot 'Wilber'.

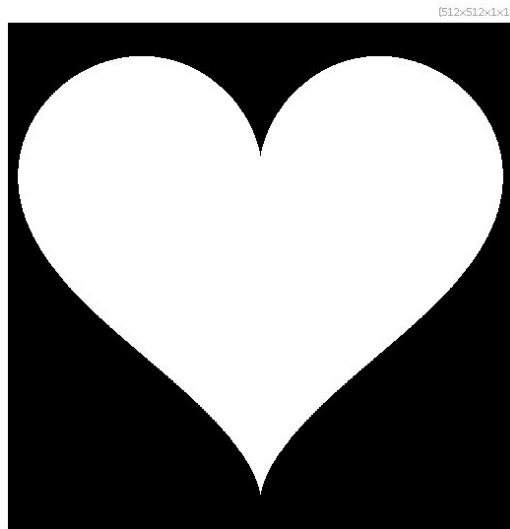


Example 15 : `-gmicky_wilber`

2.2.24 `-heart`

Arguments: `_width>0, _height>0`

Input heart binary mask with specified size.



Example 16 : `-heart ,`

2.2.25 `-input (+)`

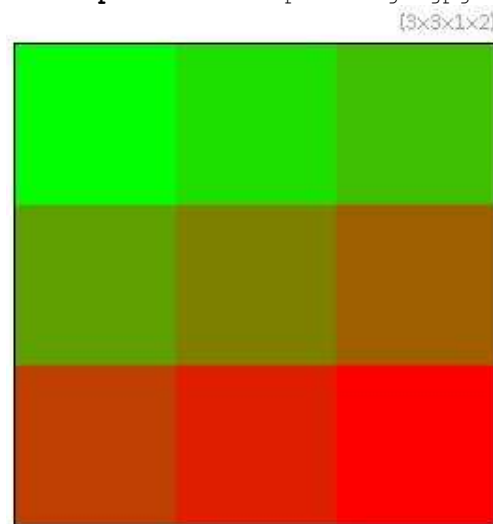
Arguments: `[type:]filename` |
`[type:]http://URL` |
`[selection]x_nb_copies>0` |
`{ width>0[%] | [image_w] },{ _height>0[%] | [image_h]`
`},{ _depth>0[%] | [image_d] },{ _spectrum>0[%] | [image_s] },-{ va-`
`lue1,_value2,.. | 'formula' }` |
`(value1{, | ; | / | ^}value2{, | ; | / | ^}..) |`
`0`

Insert a new image taken from a filename or from a copy of an existing image `['indice']`," or insert new image with specified dimensions and values. Single quotes may be omitted in `'formula'`. Specifying argument `'0'` inserts an `'empty'` image. (eq. to `'-i'` | (no arg)).

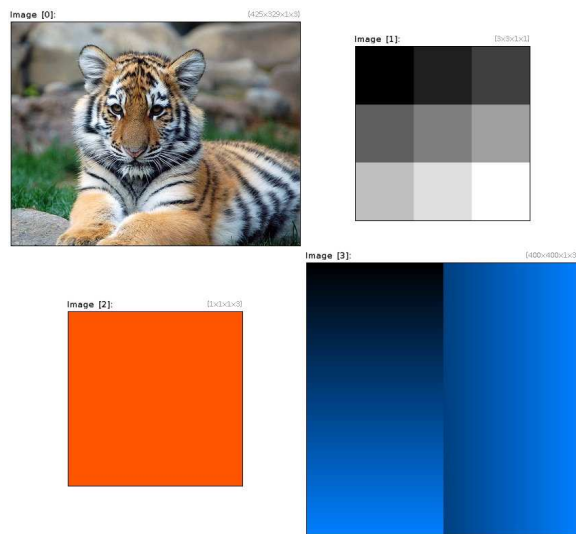
Default values: `'nb_copies=1'`, `'height=depth=spectrum=1'` and `'value1=0'`.



Example 17 : -input image.jpg



Example 18 : -i (1,2,3;4,5,6;7,8,9^9,8,7;6,5,4;3,2,1)



Example 19 : `image.jpg (1,2,3;4,5,6;7,8,9) (255^128^64)`
`400,400,1,3,'if(x>w/2,x,y)*c'`

Tutorial page:

http://gmic.eu/tutorial/_input.shtml

2.2.26 -input_gpl

Arguments: `filename`

Input specified filename as a GIMP palette data file.

2.2.27 -output (+)

Arguments: `[type:]filename,_format_options`

Output selected images as one or several numbered file(s).
 (eq. to `'-o'`).

Default value: `'format_options'=(undefined)`.

2.2.28 -output_ggr

Arguments: `filename,_gradient_name`

Output selected images as GIMP gradient files.
 If no gradient name is specified, it is deduced from the filename.

2.2.29 *-outputn*

Arguments: filename

Output selected images as automatically numbered filenames in repeat..done loops.
(eq. to '-on').

2.2.30 *-outputp*

Arguments: prefix

Output selected images as prefixed versions of their original filenames.
(eq. to '-op').

Default value: 'prefix=-'.

2.2.31 *-outputw*

Output selected images by overwriting their original location.
(eq. to '-ow').

2.2.32 *-outputx*

Arguments: extension1,extension2,...,extensionN,output_at_same_location={ 0 | 1 }

Output selected images with same base filenames but for N different extensions.
(eq. to '-ox').

Default value: 'output_at_same_location=0'.

2.2.33 *-pass (+)*

Arguments: _shared_state={ 0=non-shared (copy) | 1=shared
 | 2=adaptive }

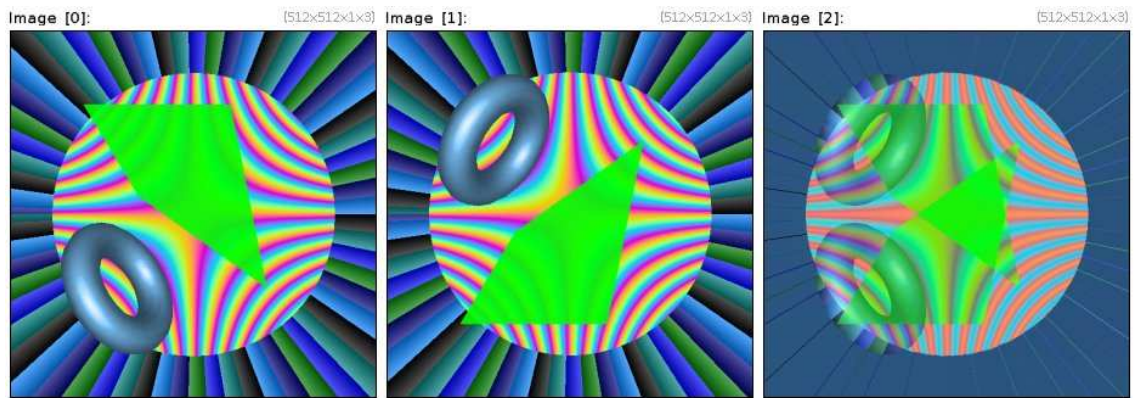
Insert images from parent context of a custom command or a local environment.
Command selection (if any) stands for a selection of images in

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.

Default value: `'shared.state=2'`.



Example 20 : `-command "average : -pass$""1 -add[^-1] [-1] -remove[-1] -div 2" -testimage2d 512 --mirror y --average[0] [1]`

2.2.34 `-plot (+)`

Arguments: `_plot_type, _vertex_type, _xmin, _xmax, _ymin, _ymax` | `'formula', _resolution>=0, _plot_type, _vertex_type, _xmin, _xmax, _ymin, _ymax`

Display selected image or formula in an interactive viewer (use the instant 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.

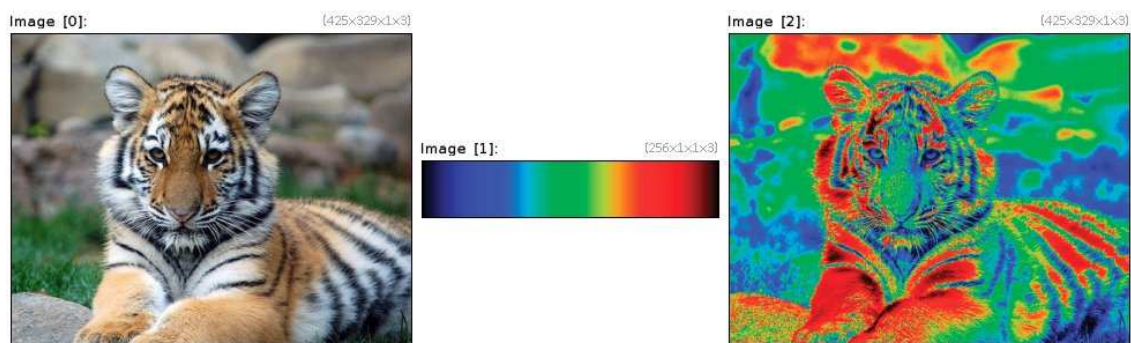
Default values: `'plot_type=1', 'vertex_type=1'` and `'xmin=xmax=ymin=ymax=0 (auto)'`.

2.2.35 **-print (+)**

Output information on selected images, on the standard error (stderr).
(eq. to '-p').

2.2.36 **-rainbow_lut**

Input a 256-entries RGB colormap of rainbow colors.



Example 21 : `image.jpg -rainbow.lut --luminance[-2] -map[-1] [-2]`

2.2.37 **-rodgy**

Load a new image of the G'MIC Rodilius mascot 'Roddy'.



Example 22 : `-rodgy`

2.2.38 **-select (+)**

Arguments: `feature_type, _X, _Y, _Z`

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 or 6d vector containing the feature coordinates.

2.2.39 -serialize (+)

Arguments: _datatype, _is_compressed={ 0 | 1 }, _store_names={ 0 | 1 }

Serialize selected list of images into a single image, optionnally in a compressed form.

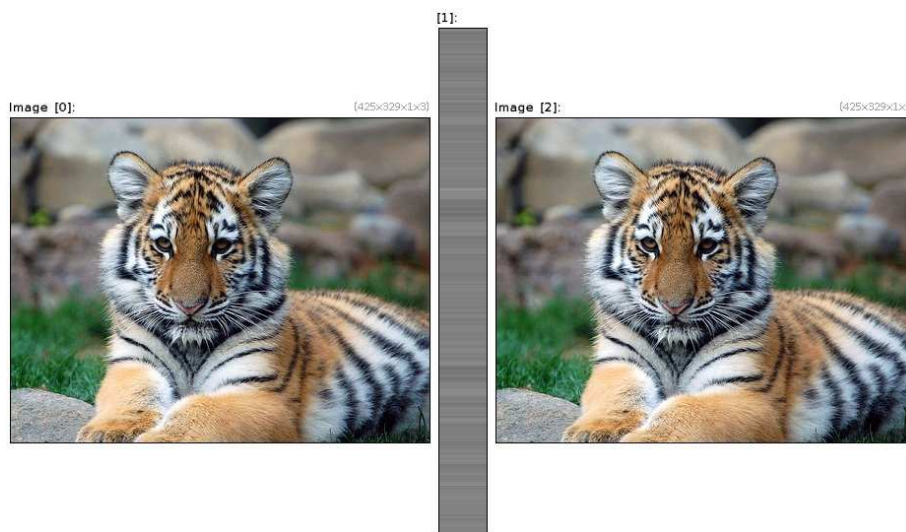
'datatype' can be { uchar | char | ushort | short | uint | int | ulong | long | float | double }.

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=float', 'is_compressed=1' and 'store_names=1'.



Example 23 : `image.jpg --serialize uchar --unserialize[-1]`

2.2.40 *-shared (+)*

Arguments: `x0[%],x1[%],y[%],z[%],v[%]` |
 `y0[%],y1[%],z[%],v[%]` |
 `z0[%],z1[%],v[%]` |
 `v0[%],v1[%]` |
 (no arg)

Insert shared buffers from (opt. points/rows/planes/channels of) selected images.

Shared buffers cannot be returned by a command, nor a local environment.

(eq. to `'-sh'`).



Example 24 : `image.jpg -shared 1,1 -blur[-1] 3 -remove[-1]`
(425x329x1x3)



Example 25 : `image.jpg -repeat {s} -shared 25%,75%,0,$> -mirror[-1] x
-remove[-1] -done`

Tutorial page:

http://gmic.eu/tutorial/_shared.shtml

2.2.41 *-srand* (+)

Arguments: value |
 (no arg)

Set random generator seed.

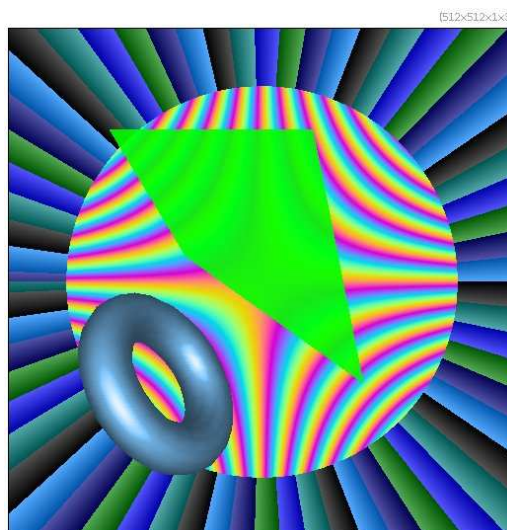
If no argument is specified, a random value is used as the random generator seed.

2.2.42 *-testimage2d*

Arguments: `_width>0, _height>0, _spectrum>0`

Input a 2d synthetic image.

Default values: `'width=512', 'height=width' and 'spectrum=3'`.



Example 26 : `-testimage2d 512`

2.2.43 *-uncommand (+)*

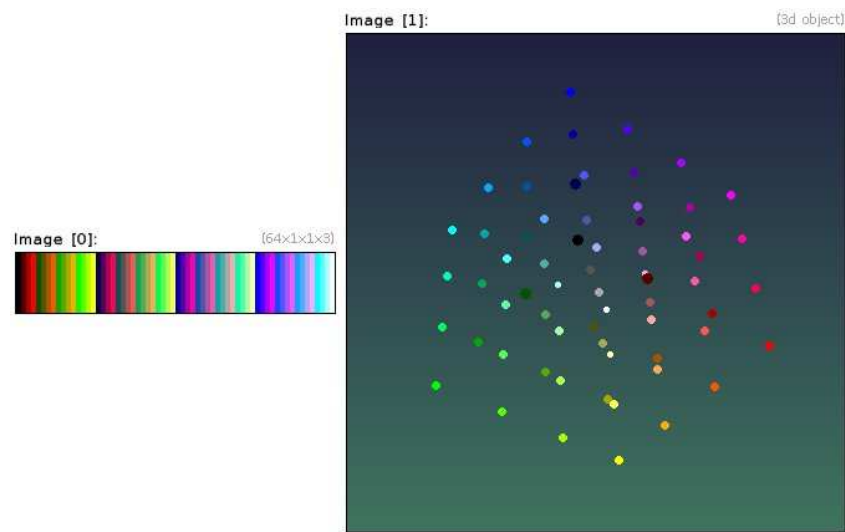
Arguments: `command_name[, _command_name2, ...]` |
*

Discard last definition of specified custom commands.
Set argument to '*' for discarding all existing custom commands.

2.2.44 *-uniform_distribution*

Arguments: `nb_levels>=1, spectrum>=1`

Input set of uniformly distributed N-d points in $[0,1]^N$.



Example 27 : `-uniform_distribution 64,3 -* 255 --distribution3d -circles3d[-1]`
10

2.2.45 *-unserialize (+)*

Recreate lists of images from serialized image buffers, obtained with command '`-serialize`'.

2.2.46 *-update*

Update commands from the latest definition file on the G'MIC server.

This requires an active Internet connection and an access to the external tools '`curl`' or '`wget`'.

Once the update has been downloaded, running '`gmic`' makes it use automatically.

(eq. to '`-up`').

2.2.47 *-verbose (+)*

Arguments: `level` |
 { + | - }

Set or increment/decrement the verbosity level. Default level is 0.

(eq. to '`-v`').

When '`level`'>=0, G'MIC log messages are displayed on the standard error (stderr).

Default value: '`level=0`'.

2.2.48 -wait (+)

Arguments: delay |
 (no arg)

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 value: 'delay=0'.

2.2.49 -warn (+)

Arguments: _force_visible={ 0 | 1 },_message

Print specified warning message, on the standard error (stderr).

Command selection (if any) stands for displayed call stack subset instead of image indices.

2.2.50 -window (+)

Arguments: _width[%]>=-1,_height[%]>=-1,_normalization,_fullscreen,_pos-
_x[%],_pos-y[%],_title

Display selected images into an instant display window with specified size, normalization type, fullscreen mode and title. (eq. to '-w').

If 'width' or 'height' is set to -1, the corresponding dimension is adjusted to the window or image size.

When arguments 'pos_x' and 'pos_y' are both different than -1, the window is moved to the specified coordinates.

'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'`.

Default values: `'width=height=normalization=fullscreen=-1'` and
`'title=(undefined)'`.

2.3 List manipulation

2.3.1 *-keep* (+)

Keep only selected images.
 (eq. to `'-k'`).



Example 28 : `image.jpg -split x -keep[0-50%:2] -append x`

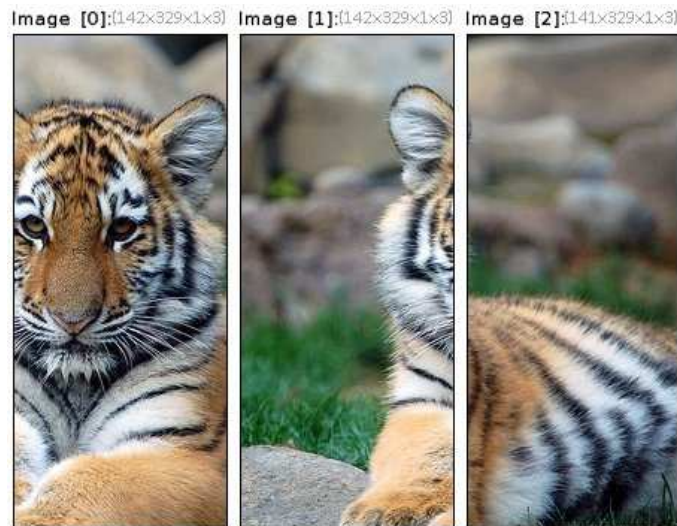


Example 29 : `image.jpg -split x -keep[^30%-70%] -append x`

2.3.2 *-move (+)*

Arguments: `position[%]`

Move selected images at specified position.
(eq. to `'-mv'`).



Example 30 : `image.jpg -split x,3 -move[1] 0`
(425x329x1x3)

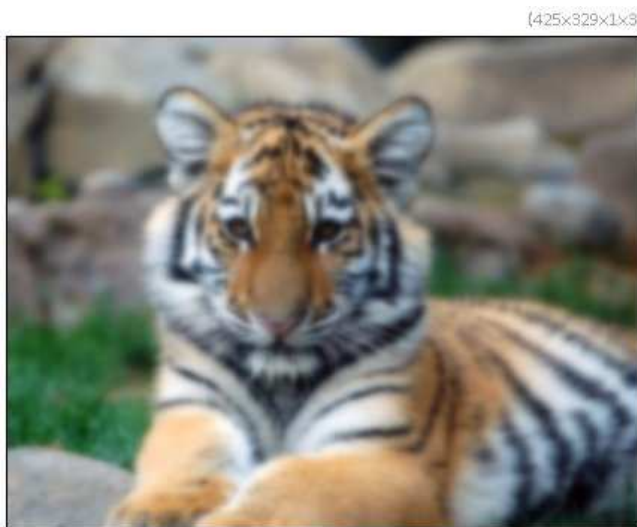


Example 31 : `image.jpg -split x -move[50%--1:2] 0 -append x`

2.3.3 *-name (+)*

Arguments: " *name* "

Set name of selected images.
(eq. to '*-nm*').



Example 32 : `image.jpg -name image -blur[image] 2`

2.3.4 *-names*

Arguments: *name1, name2, ..., nameN*

Set each name of (multiple) selected images from the sequence of the provided arguments.
(eq. to '*-nms*').

2.3.5 *-remove (+)*

Remove selected images.
(eq. to '*-rm*').



Example 33 : `image.jpg -split x -remove[30%-70%] -append x`



Example 34 : `image.jpg -split x -remove[0-50%:2] -append x`

2.3.6 *-remove_duplicates*

Remove duplicates images in the selected images list.



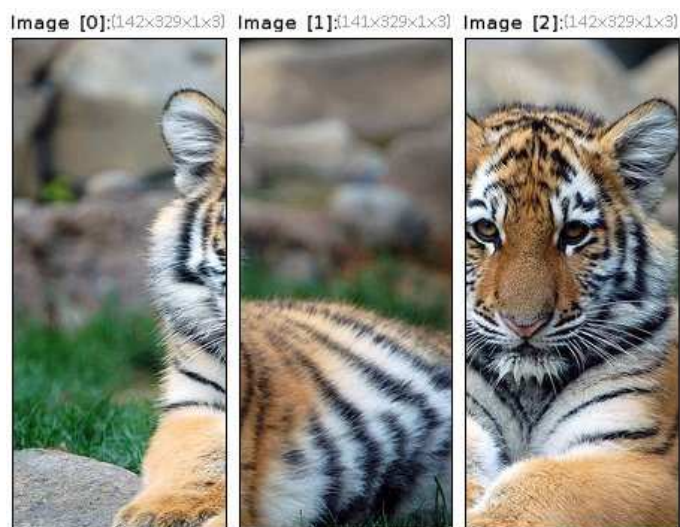
Example 35 : `(1,2,3,4,2,4,3,1,3,4,2,1) -split x -remove_duplicates -append x`

2.3.7 *-remove_empty*

Remove empty images in the selected image list.

2.3.8 *-reverse (+)*

Reverse positions of selected images.
(eq. to `'-rv'`).



Example 36 : `image.jpg -split x,3 -reverse[-2,-1]`



Example 37 : `image.jpg -split x,-16 -reverse[50%-100%] -append x`

2.3.9 *-sort_list*

Arguments: `_ordering={ + | - },_criterion`

Sort list of selected images according to the specified image criterion.

Default values: `'ordering=+', 'criterion=i'`.



Example 38 : `(1;4;7;3;9;2;4;7;6;3;9;1;0;3;3;2) -split y -sort_list + -append y`

2.3.10 `-sort_str`

Sort selected images (viewed as a list of strings) in lexicographic order.

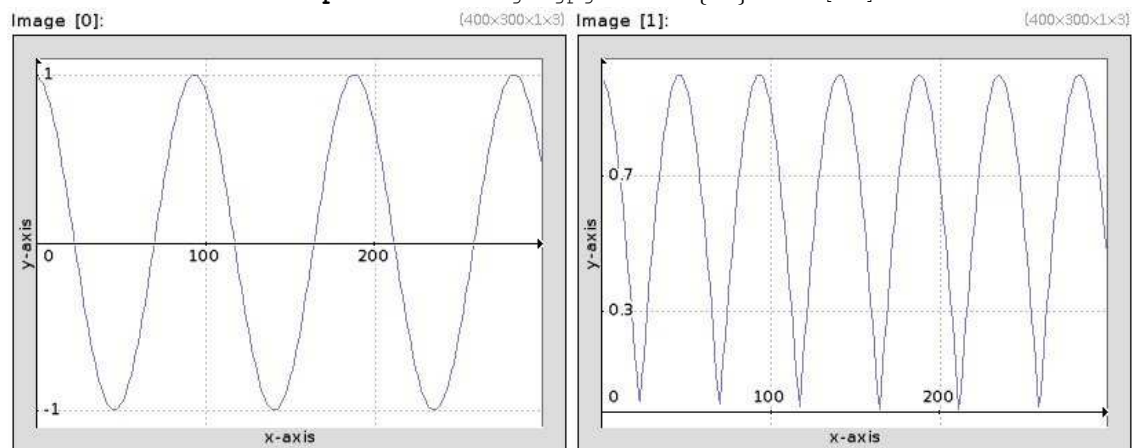
2.4 Mathematical operators

2.4.1 `-abs (+)`

Compute the pointwise absolute values of selected images.



Example 39 : `image.jpg --sub {ia} -abs[-1]`



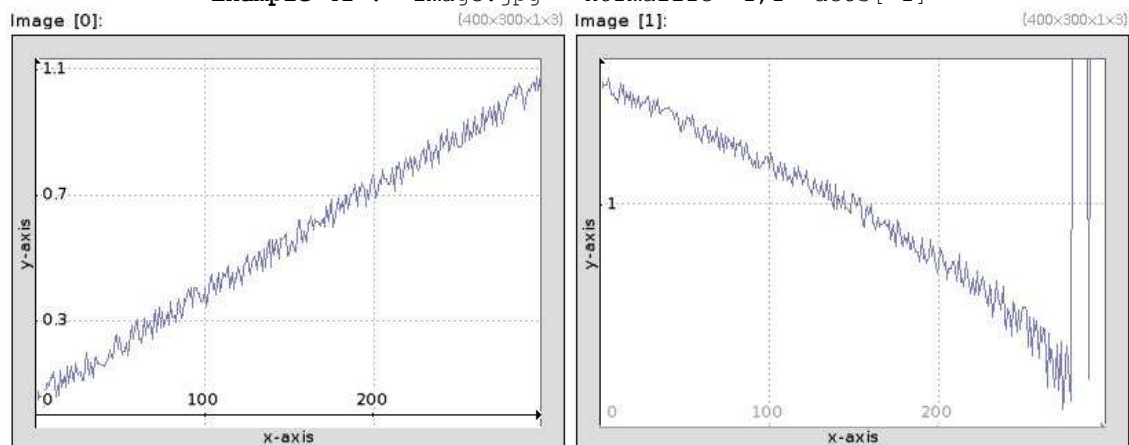
Example 40 : `300,1,1,1,'cos(20*x/w)'` `--abs -display-graph 400,300`

2.4.2 `-acos (+)`

Compute the pointwise arc-cosine of selected images.



Example 41 : `image.jpg --normalize -1,1 -acos[-1]`



Example 42 : `300,1,1,1, 'x/w+0.1*u' --acos -display_graph 400,300`

Tutorial page:

<http://gmic.eu/tutorial/trigometric-and-inverse-trigometric-commands.shtml>

2.4.3 -add (+)

Arguments: value[%] |
 [image] |
 'formula' |
 (no arg)

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

(eq. to '-+').



Example 43 : `image.jpg --add 30% -cut 0,255`



Example 44 : `image.jpg --blur 5 -normalize 0,255 -add[1] [0]`
(425x329x1x3)



Example 45 : `image.jpg -add '80*cos(80*(x/w-0.5)*(y/w-0.5)+c)' -cut 0,255`



Example 46 : `image.jpg -repeat 9 --rotate[0] {>*36},1,0,50%,50% -done -add -div 10`

2.4.4 *-and* (+)

Arguments: value[%] |
 [image] |
 'formula' |
 (no arg)

Compute the bitwise AND of selected images with specified value, image or mathematical expression, or compute the pointwise sequential bitwise AND of selected images.
 (eq. to `'-&'`).



Example 47 : `image.jpg -and {128+64}`



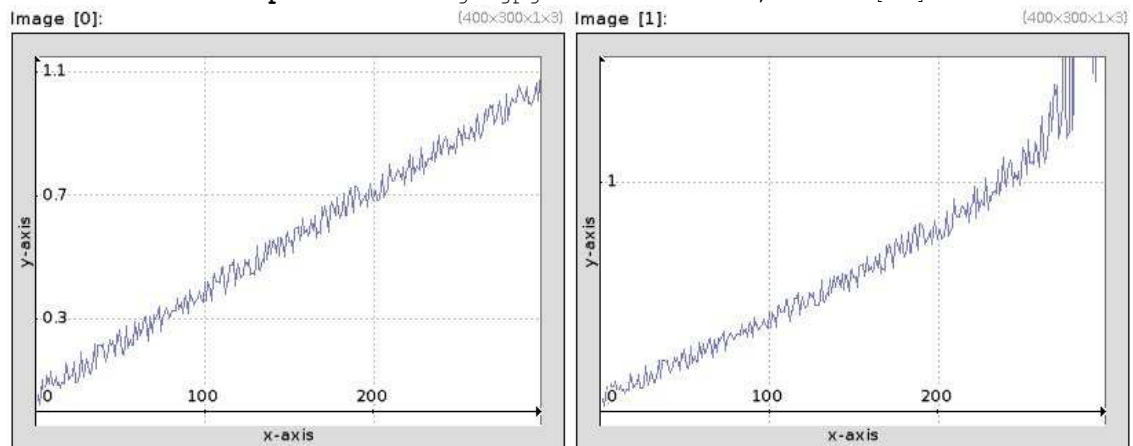
Example 48 : `image.jpg --mirror x -and`

2.4.5 *-asin* (+)

Compute the pointwise arc-sine of selected images.



Example 49 : `image.jpg --normalize -1,1 -asin[-1]`



Example 50 : `300,1,1,1,1,'x/w+0.1*u' --asin -display_graph 400,300`

Tutorial page:

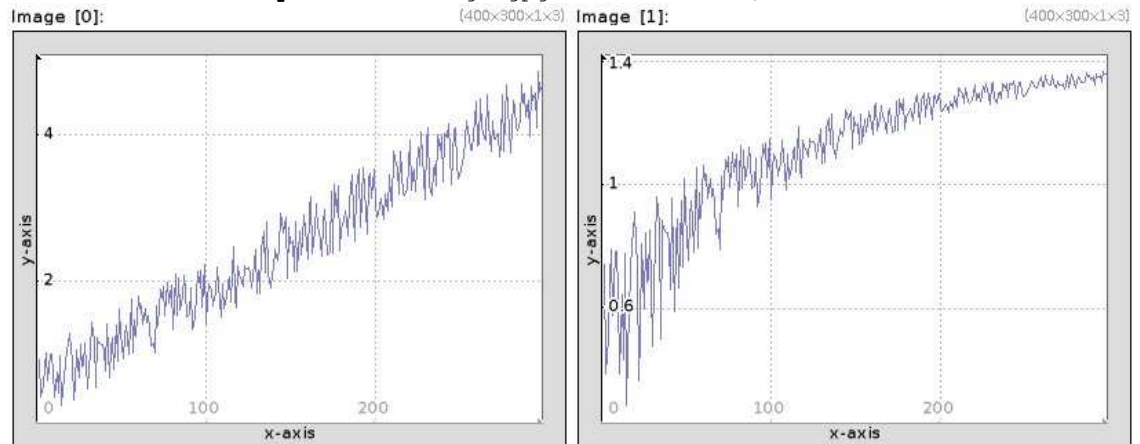
<http://gmics.eu/tutorial/trigometric-and-inverse-trigometric-commands.shtml>

2.4.6 `-atan (+)`

Compute the pointwise arc-tangent of selected images.



Example 51 : `image.jpg --normalize 0,8 -atan[-1]`



Example 52 : `300,1,1,1,'4*x/w+u' --atan -display-graph 400,300`

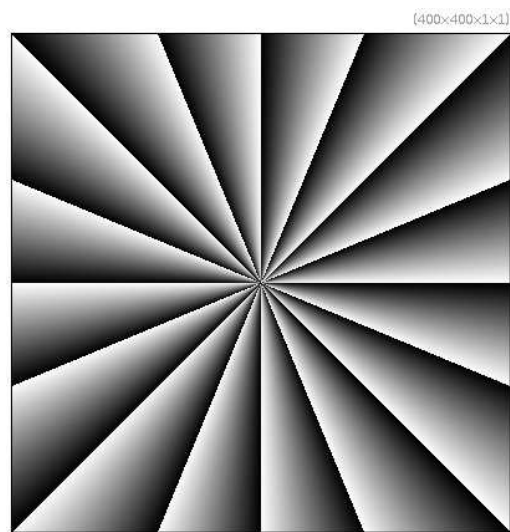
Tutorial page:

<http://gmic.eu/tutorial/trigometric-and-inverse-trigometric-commands.shtml>

2.4.7 *-atan2* (+)

Arguments: `[x_argument]`

Compute the pointwise oriented arc-tangent of selected images. Each selected image is regarded as the y-argument of the arc-tangent function, while the specified image gives the corresponding x-argument.



Example 53 : `(-1,1) (-1;1) -resize 400,400,1,1,3 -atan2[1] [0] -keep[1] -mod {pi/8}`

Tutorial page:

<http://gmic.eu/tutorial/trigometric-and-inverse-trigometric-commands.shtml>

2.4.8 *-bsl* (+)

Arguments: value[%] |
 [image] |
 'formula' |
 (no arg)

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. (eq. to ' \ll ').



Example 54 : `image.jpg -bsl 'round(3*x/w,0)' -cut 0,255`

2.4.9 *-bsr (+)*

Arguments: value[%] |
 [image] |
 'formula' |
 (no arg)

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.

(eq. to ' \rightarrow ').



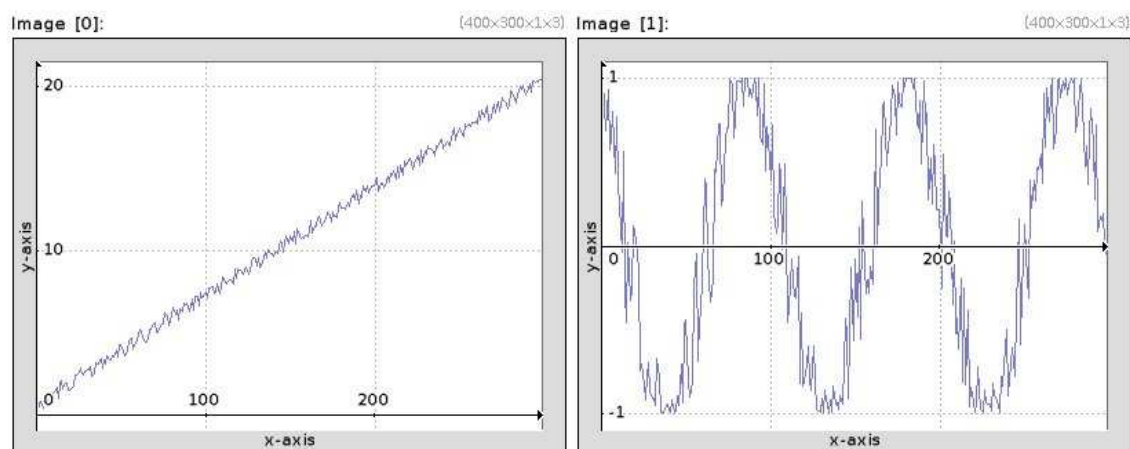
Example 55 : `image.jpg -bsr 'round(3*x/w,0)' -cut 0,255`

2.4.10 *-cos (+)*

Compute the pointwise cosine of selected images.



Example 56 : `image.jpg --normalize 0,{2*pi} -cos[-1]`



Example 57 : `300,1,1,1,'20*x/w+u' --cos -display_graph 400,300`

Tutorial page:

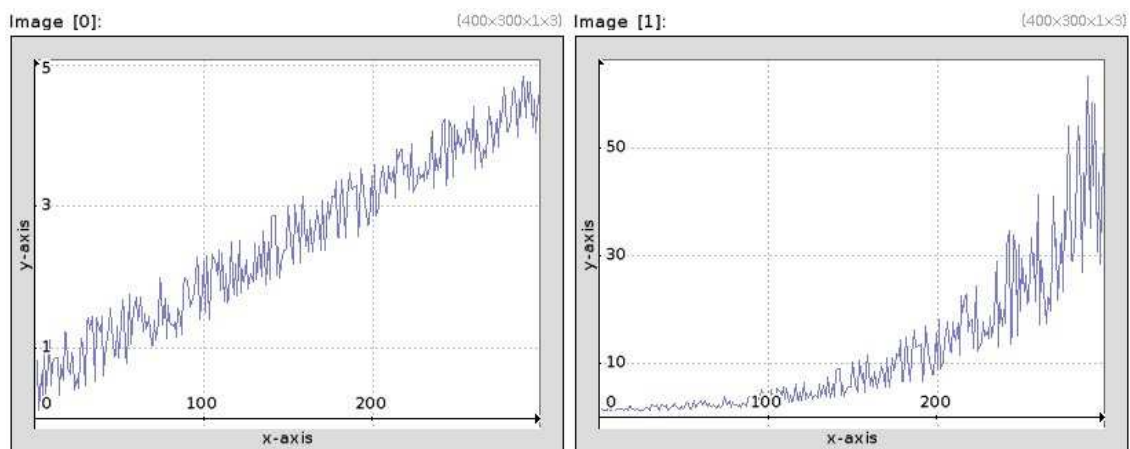
<http://gmick.eu/tutorial/trigometric-and-inverse-trigometric-commands.shtml>

2.4.11 *-cosh (+)*

Compute the pointwise hyperbolic cosine of selected images.



Example 58 : `image.jpg --normalize -3,3 -cosh[-1]`



Example 59 : `300,1,1,1,'4*x/w+u' --cosh -display_graph 400,300`

2.4.12 *-div (+)*

Arguments: `value[%]` |
 `[image]` |
 `'formula'` |
 `(no arg)`

Divide selected image by specified value, image or mathematical expression, or compute the pointwise quotient of selected images.
 (eq. to `'-/'`).



Example 60 : `image.jpg -div '1+abs(cos(x/10)*sin(y/10))'`



Example 61 : `image.jpg --norm -add[-1] 1 --div`

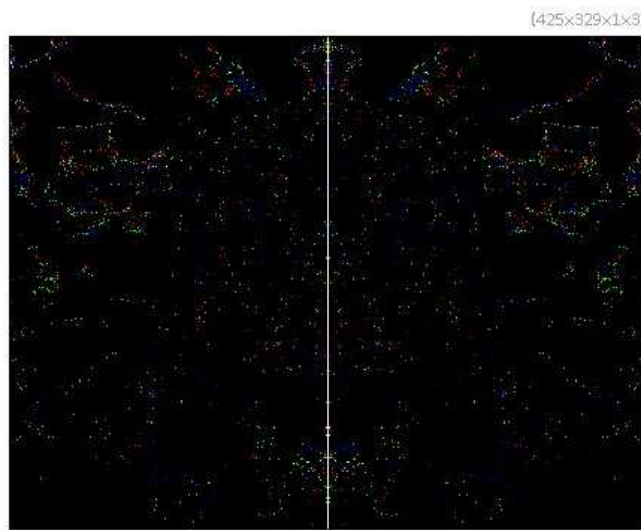
2.4.13 `-eq (+)`

Arguments: value[%] |
 [image] |
 'formula' |
 (no arg)

Compute the boolean equality of selected images with specified value, image or mathematical expression, or compute the boolean equality of selected images.
 (eq. to '==').



Example 62 : `image.jpg -round 40 -eq {round(ia,40)}`



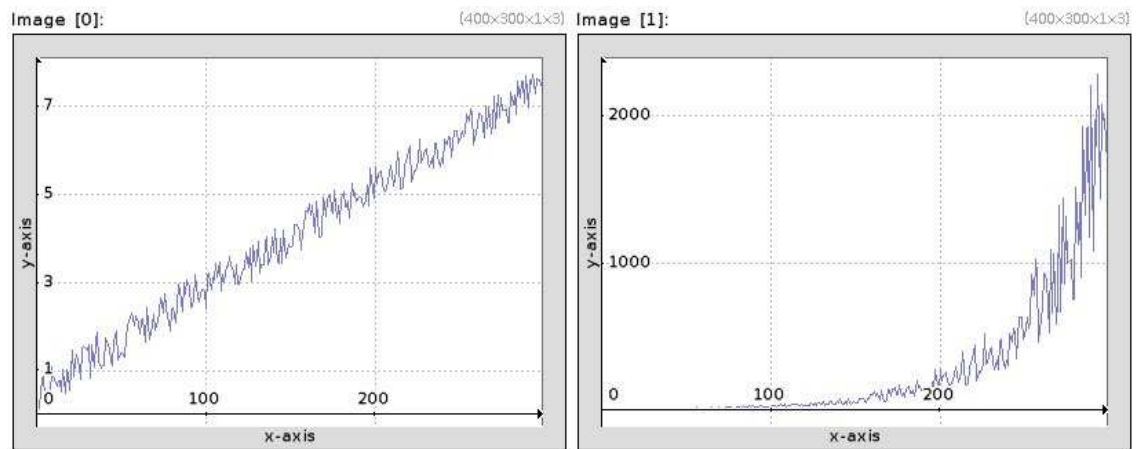
Example 63 : `image.jpg --mirror x -eq`

2.4.14 *-exp (+)*

Compute the pointwise exponential of selected images.



Example 64 : `image.jpg --normalize 0,2 -exp[-1]`



Example 65 : `300,1,1,1,'7*x/w+u' --exp -display_graph 400,300`

2.4.15 -ge (+)

Arguments: value[%] |
 [image] |
 'formula' |
 (no arg)

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.

(eq. to ' \geq ').



Example 66 : `image.jpg -ge {ia}`



Example 67 : `image.jpg --mirror x -ge`

2.4.16 `-gt (+)`

Arguments: `value[%]` |
 `[image]` |
 `'formula'` |
 `(no arg)`

Compute the boolean 'greater than' of selected images with specified value, image or mathematical expression, or compute the boolean 'greater than' of selected images.
 (eq. to '`->`').



Example 68 : `image.jpg -gt {ia}`

(425x329x1x3)



Example 69 : `image.jpg --mirror x -gt`

2.4.17 *-le (+)*

Arguments: value[%] |
 [image] |
 'formula' |
 (no arg)

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. (eq. to '`-<=`').



Example 70 : `image.jpg -le {ia}`



Example 71 : `image.jpg --mirror x -le`

2.4.18 `-lt (+)`

Arguments: `value[%]` |
 `[image]` |
 `'formula'` |
 `(no arg)`

Compute the boolean 'less than' of selected images with specified value, image or mathematical expression, or compute the boolean 'less than' of selected images.
 (eq. to '`-<`').



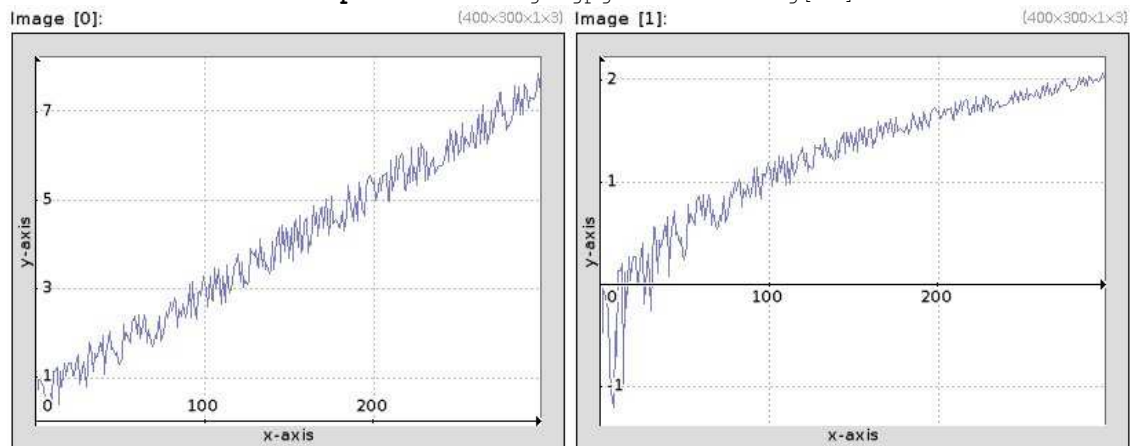
Example 72 : `image.jpg -lt {ia}`



Example 73 : `image.jpg --mirror x -lt`

2.4.19 *-log (+)*

Compute the pointwise base-e logarithm of selected images.

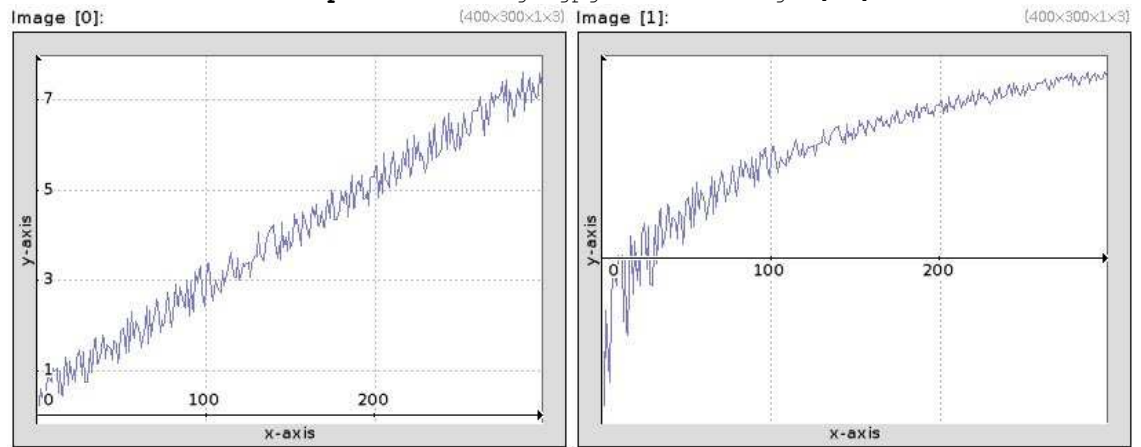
**Example 74 :** `image.jpg --add 1 -log[-1]`**Example 75 :** `300,1,1,1,'7*x/w+u' --log -display_graph 400,300`

2.4.20 $-\log_{10} (+)$

Compute the pointwise base-10 logarithm of selected images.



Example 76 : `image.jpg --add 1 -log10[-1]`



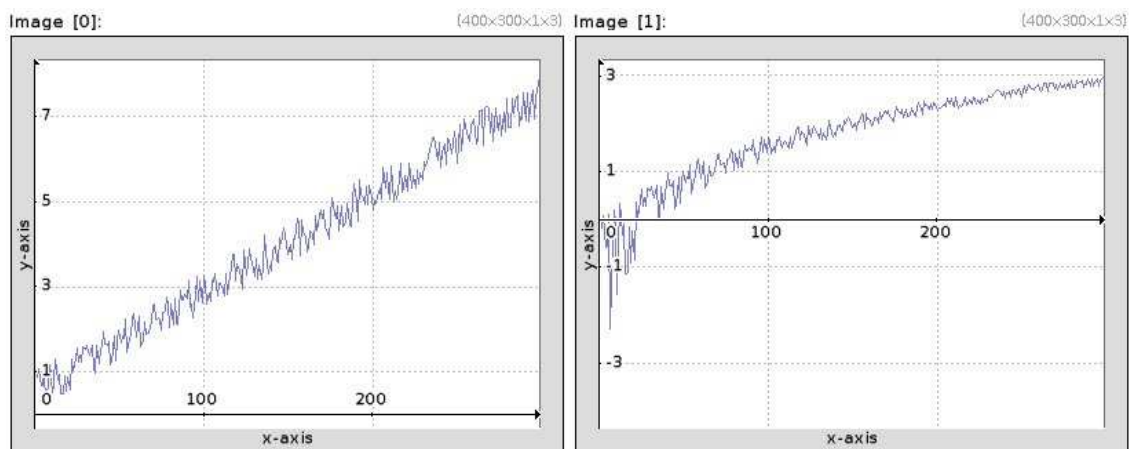
Example 77 : `300,1,1,1,'7*x/w+u' --log10 -display-graph 400,300`

2.4.21 $-\log_2 (+)$

Compute the pointwise base-2 logarithm of selected images



Example 78 : `image.jpg --add 1 -log2[-1]`



Example 79 : `300,1,1,1,'7*x/w+u' --log2 -display_graph 400,300`

2.4.22 *-max (+)*

Arguments: value[%] |
 [image] |
 'formula' |
 (no arg)

Compute the maximum between selected images and specified value, image or mathematical expression, or compute the pointwise maxima between selected images.



Example 80 : `image.jpg --mirror x -max`



Example 81 : `image.jpg -max 'R=((x/w-0.5)^2+(y/h-0.5)^2)^0.5;255*R'`

2.4.23 *-mdiv (+)*

Arguments: value[%] |
 [image] |
 'formula' |
 (no arg)

Compute the matrix division of selected matrices/vectors by specified value, image or mathematical expression, or compute the matrix division of selected images.
 (eq. to '-//').

2.4.24 *-min (+)*

Arguments: value[%] |
 [image] |
 'formula' |
 (no arg)

Compute the minimum between selected images and specified value, image or mathematical expression, or compute the pointwise minima between selected images.



Example 82 : `image.jpg --mirror x -min`
(425x329x1x3)



Example 83 : `image.jpg -min 'R=((x/w-0.5)^2+(y/h-0.5)^2)^0.5;255*R'`

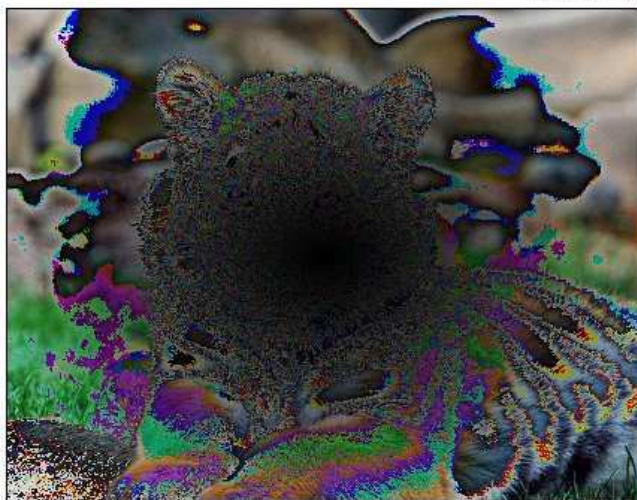
2.4.25 *-mod (+)*

Arguments: value[%] |
 [image] |
 'formula' |
 (no arg)

Compute the modulo of selected images with specified value, image or mathematical expression, or compute the pointwise sequential modulo of selected images. (eq. to `'-%'`).



Example 84 : `image.jpg --mirror x -mod`
(425x329x1x3)



Example 85 : `image.jpg -mod 'R=((x/w-0.5)^2+(y/h-0.5)^2)^0.5;255*R'`

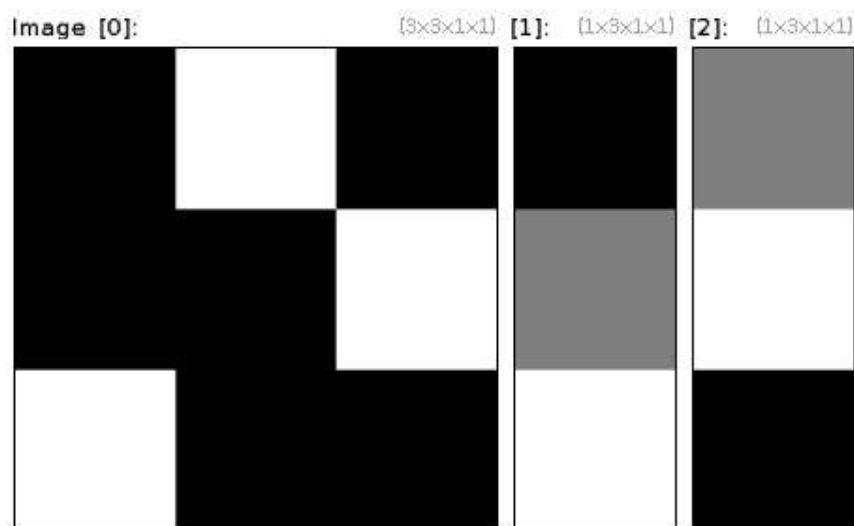
2.4.26 *-mmul* (+)

Arguments:

<code>value[%]</code>	
<code>[image]</code>	
<code>'formula'</code>	
<code>(no arg)</code>	

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.

(eq. to `'-**'`).



Example 86 : `(0,1,0;0,0,1;1,0,0) (1;2;3) --mmul`

2.4.27 `-mul (+)`

Arguments: `value[%]` |
 `[image]` |
 `'formula'` |
 `(no arg)`

Multiply selected images by specified value, image or mathematical expression, or compute the pointwise product of selected images.

(eq. to `'-**'`).



Example 87 : `image.jpg --mul 2 -cut 0,255`



Example 88 : `image.jpg (1,2,3,4,5,6,7,8) -resize[-1] [0] -mul[0] [-1]`
 (425x329x1x3)



Example 89 : `image.jpg -mul '1-3*abs(x/w-0.5)' -cut 0,255`



Example 90 : `image.jpg --luminance -negative[-1] --mul`

2.4.28 *-mul_channels*

Arguments: `value1,value2,...,valueN`

Multiply channels of selected images by specified sequence of values.



Example 91 : `image.jpg --mul-channels 1,0.5,0.8`

2.4.29 *-neq* (+)

Arguments: value[%] |
 [image] |
 'formula' |
 (no arg)

Compute the boolean inequality of selected images with specified value, image or mathematical expression, or compute the boolean inequality of selected images.
 (eq. to '-!=').



Example 92 : `image.jpg -round 40 -neg {round(ia,40)}`

2.4.30 -or (+)

Arguments: value[%] |
 [image] |
 'formula' |
 (no arg)

Compute the bitwise OR of selected images with specified value, image or mathematical expression, or compute the pointwise sequential bitwise OR of selected images. (eq. to '- | ').



Example 93 : `image.jpg -or 128`

(425x329x1x3)



Example 94 : `image.jpg --mirror x -or`

2.4.31 *-pow (+)*

Arguments: `value[%]` |
 `[image]` |
 `'formula'` |
 `(no arg)`

Raise selected image to the power of specified value, image or mathematical expression, or compute the pointwise sequential powers of selected images.
 (eq. to `'-^'`).



Example 95 : `image.jpg -div 255 --pow 0.5 -mul 255`



Example 96 : `image.jpg -gradient -pow 2 -add -pow 0.2`

2.4.32 *-rol (+)*

Arguments: `value[%]` |
 `[image]` |
 `'formula'` |
 `(no arg)`

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 97 : `image.jpg -rol 'round(3*x/w,0)' -cut 0,255`

2.4.33 *-ror (+)*

Arguments: value[%] |
 [image] |
 'formula' |
 (no arg)

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.



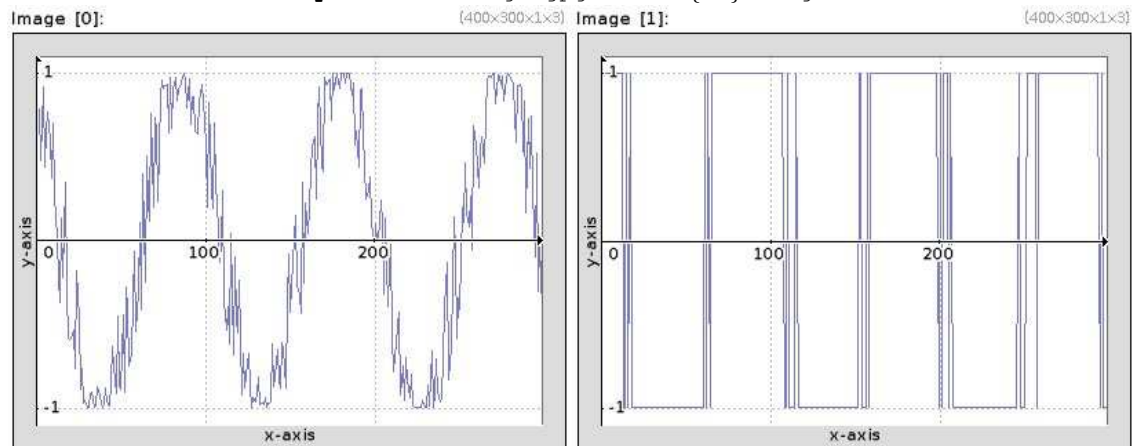
Example 98 : `image.jpg -ror 'round(3*x/w,0)' -cut 0,255`

2.4.34 *-sign (+)*

Compute the pointwise sign of selected images.



Example 99 : `image.jpg --sub {ia} -sign[-1]`



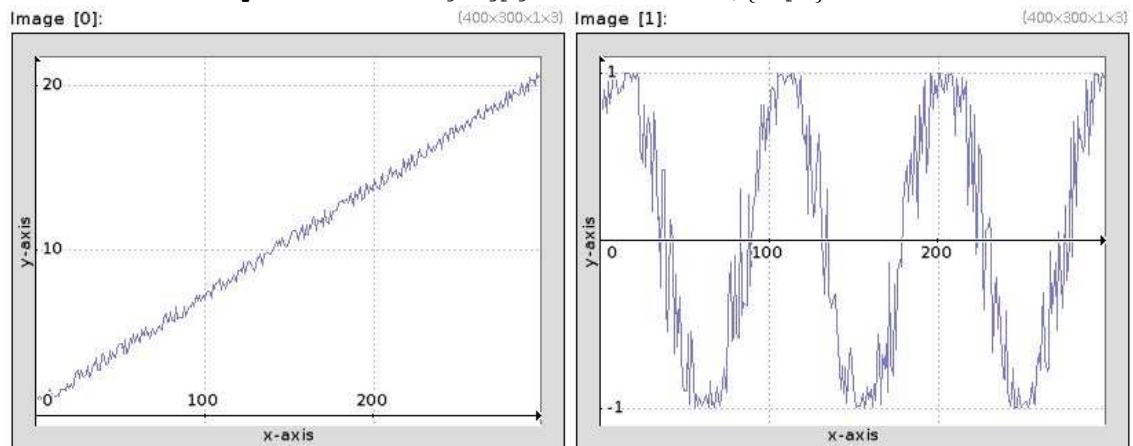
Example 100 : `300,1,1,1,'cos(20*x/w+u)' --sign -display_graph 400,300`

2.4.35 *-sin (+)*

Compute the pointwise sine of selected images.



Example 101 : `image.jpg --normalize 0,{2*pi} -sin[-1]`



Example 102 : `300,1,1,1,'20*x/w+u' --sin -display-graph 400,300`

Tutorial page:

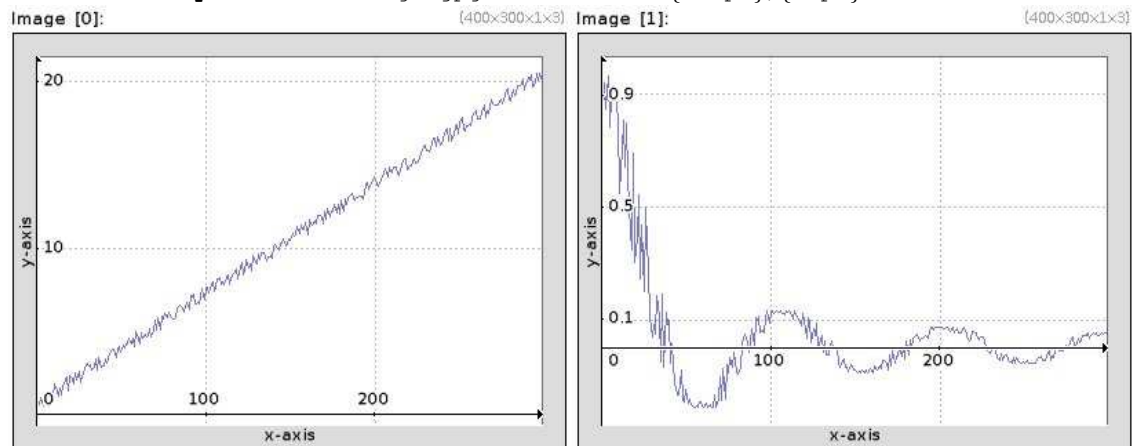
<http://gmic.eu/tutorial/trigometric-and-inverse-trigometric-commands.shtml>

2.4.36 *-sinc (+)*

Compute the pointwise sinc function of selected images.



Example 103 : `image.jpg --normalize {-2*pi},{2*pi} -sinc[-1]`



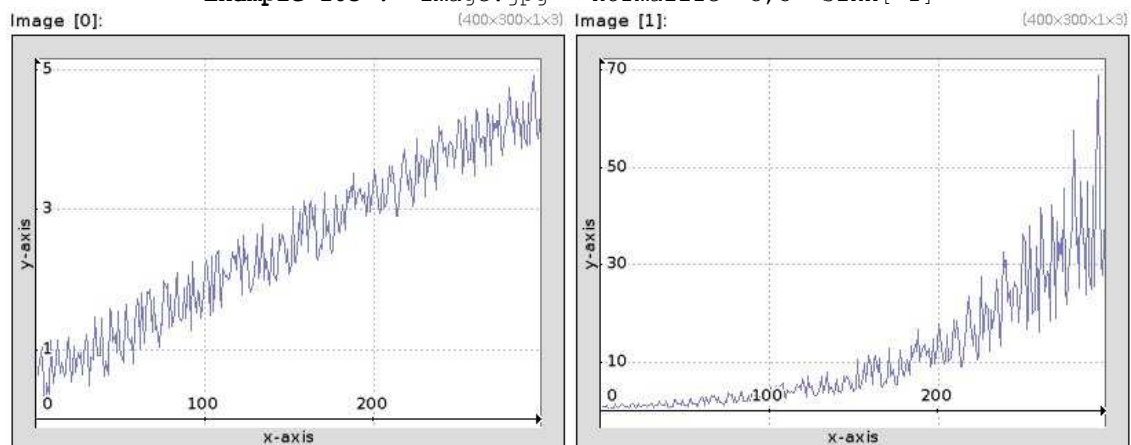
Example 104 : `300,1,1,1,'20*x/w+u' --sinc -display-graph 400,300`

2.4.37 *-sinh (+)*

Compute the pointwise hyperbolic sine of selected images.



Example 105 : `image.jpg --normalize -3,3 -sinh[-1]`



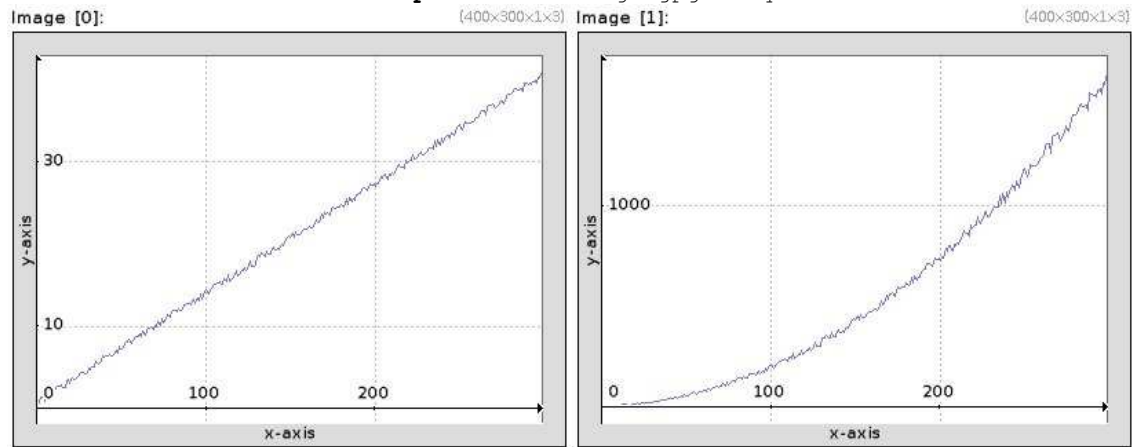
Example 106 : `300,1,1,1,'4*x/w+u' --sinh -display_graph 400,300`

2.4.38 -sqr (+)

Compute the pointwise square function of selected images.



Example 107 : `image.jpg --sqr`



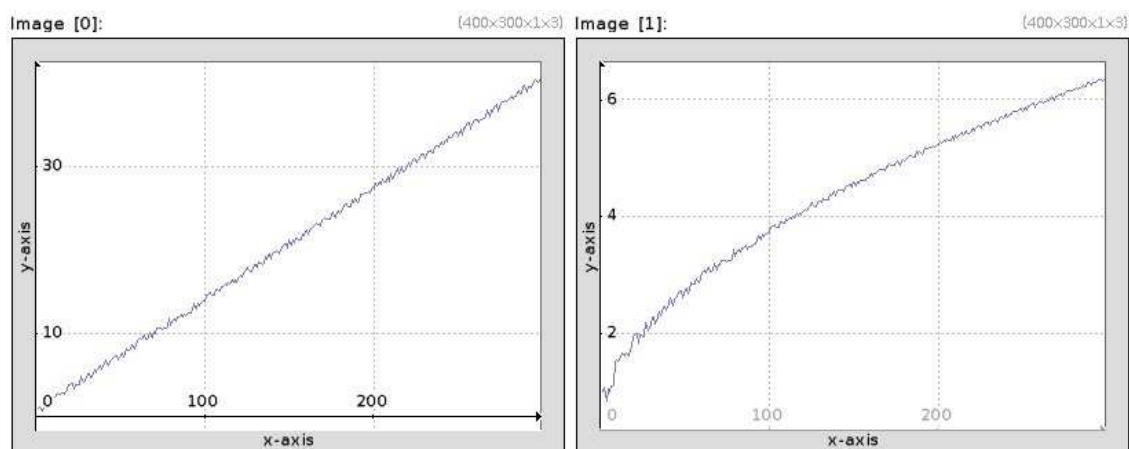
Example 108 : `300,1,1,1,'40*x/w+u' --sqr -display-graph 400,300`

2.4.39 `-sqr (+)`

Compute the pointwise square root of selected images.



Example 109 : `image.jpg --sqr`



Example 110 : `300,1,1,1,'40*x/w+u' --sqrt -display_graph 400,300`

2.4.40 *-sub* (+)

Arguments: value[%] |
 [image] |
 'formula' |
 (no arg)

Subtract specified value, image or mathematical expression to selected images, or compute the pointwise difference of selected images.

(eq. to '--').



Example 111 : `image.jpg --sub 30% -cut 0,255`



Example 112 : `image.jpg --mirror x -sub[-1] [0]`
 {425x329x1x3}



Example 113 : `image.jpg -sub 'i(w/2+0.9*(x-w/2),y)'`



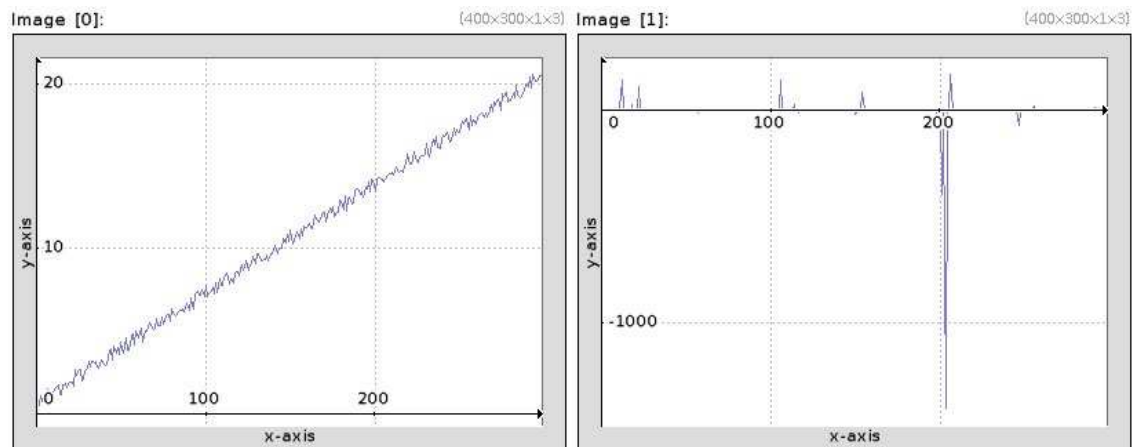
Example 114 : `image.jpg --mirror x -sub`

2.4.41 *-tan (+)*

Compute the pointwise tangent of selected images.



Example 115 : `image.jpg --normalize {-0.47*pi},{0.47*pi} -tan[-1]`



Example 116 : `300,1,1,1, '20*x/w+u' --tan -display_graph 400,300`

Tutorial page:

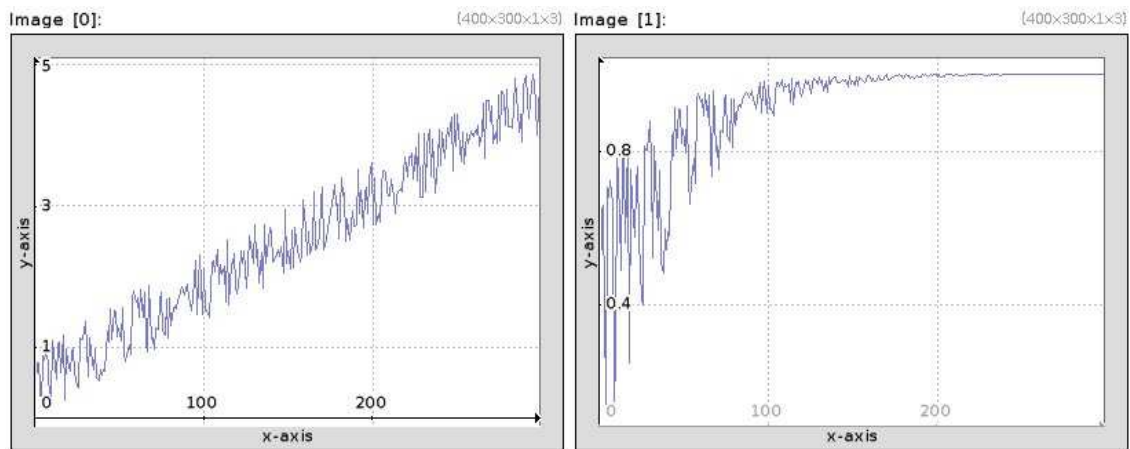
<http://gmics.eu/tutorial/trigometric-and-inverse-trigometric-commands.shtml>

2.4.42 -tanh (+)

Compute the pointwise hyperbolic tangent of selected images.



Example 117 : `image.jpg --normalize -3,3 -tanh[-1]`

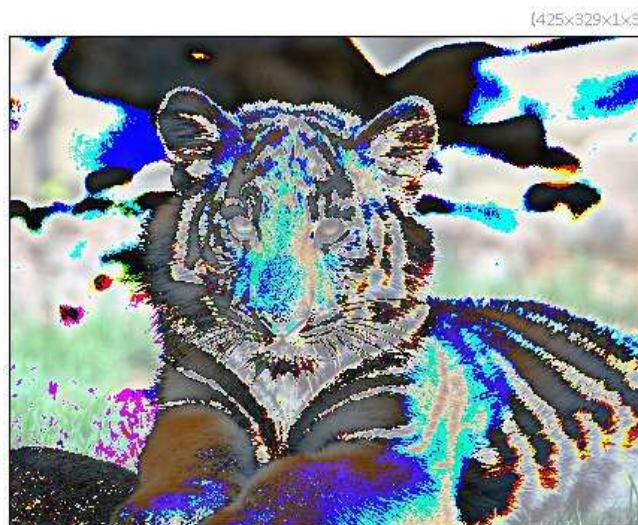


Example 118 : `300,1,1,1,'4*x/w+u' --tanh -display_graph 400,300`

2.4.43 `-xor (+)`

Arguments: value[%] |
 [image] |
 'formula' |
 (no arg)

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



Example 119 : `image.jpg -xor 128`



Example 120 : `image.jpg --mirror x -xor`

2.5 Values manipulation

2.5.1 *-apply_curve*

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

Apply curve transformation to image values.

Default values: `'smoothness=1', 'x0=0', 'y0=100'.`



Example 121 : `image.jpg --apply_curve 1,0,0,128,255,255,0`

2.5.2 *-apply_gamma*

Arguments: `gamma>=0`

Apply gamma correction to selected images.



Example 122 : `image.jpg --apply-gamma 2`

2.5.3 *-balance_gamma*

Arguments: `_ref_color1,...`

Apply color balance transformation on selected image, with respect to specified reference color.

Default value: `'ref_color1=128'`.



Example 123 : `image.jpg --balance_gamma 128,64,64`

2.5.4 *-complex2polar*

Compute complex to polar transforms of selected images.



Example 124 : `image.jpg --fft -complex2polar[-2,-1] -log[-2] -shift[-2] 50%,50%,0,0,2 -remove[-1]`

2.5.5 *-compress_rle*

Arguments: `_is_binary_data={ 0 | 1 },_maximum_sequence_length>=0`

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`'.

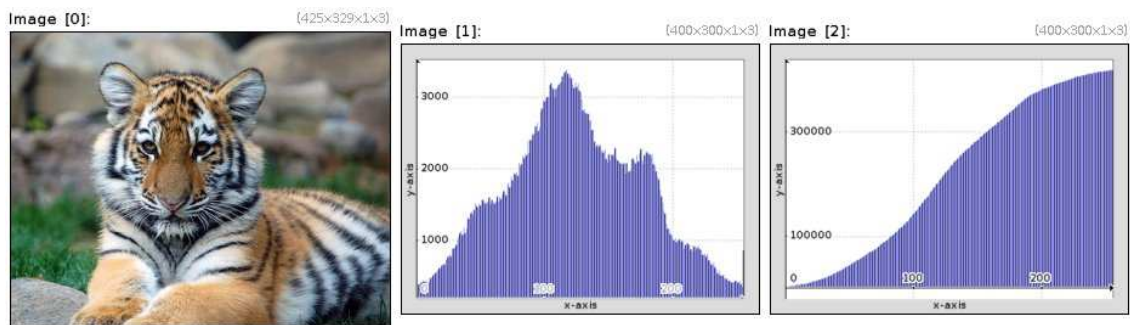


Example 125 : `image.jpg -resize2dy 100 -quantize 4 -round --compress_rle ,
--uncompress_rle[-1]`

2.5.6 -cumulate (+)

Arguments: { x | y | z | c } .. { x | y | z | c } |
(no arg)

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



Example 126 : `image.jpg --histogram --cumulate[-1] -display_graph[-2,-1]
400,300,3`

2.5.7 -cut (+)

Arguments: { value0[%] | [image0] }, { value1[%] | [image1] } |
[image] |
(no arg)

Cut values of selected images in specified range.

(eq. to '-c').

(no arg) runs interactive mode (uses the instant display window
[0] if opened).



Example 127 : `image.jpg --add 30% -cut[-1] 0,255`



Example 128 : `image.jpg --cut 25%,75%`

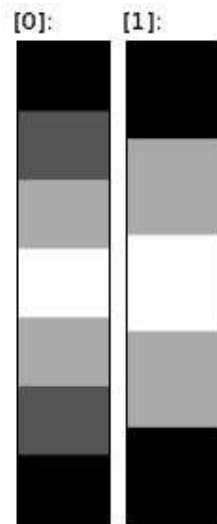
2.5.8 *-discard (+)*

Arguments: `_value1,_value2,...` |
`{ x | y | z | c}..{ x | y | z | c},_value1,_value2,...` |
 (no args)

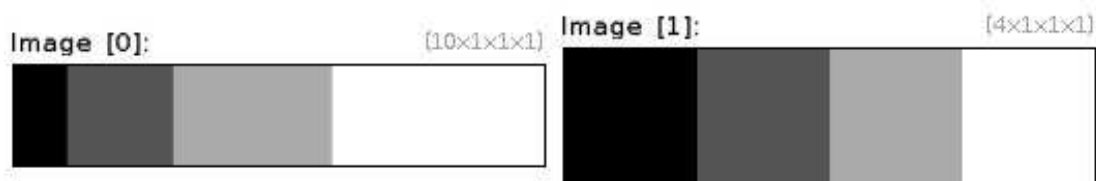
Discard specified values in selected images or discard neighboring duplicate values, optionally only for the values along the first of a specified axis.

If no values are specified, neighboring duplicate values are discarded.

If all pixels of a selected image are discarded, an empty image is returned.



Example 129 : (1;2;3;4;3;2;1) --discard 2



Example 130 : (1,2,2,3,3,3,4,4,4,4) --discard x

2.5.9 *-eigen2tensor*

Recompose selected pairs of eigenvalues/eigenvectors as 2x2 or 3x3 tensor fields.

Tutorial page:

http://gmics.eu/tutorial/_eigen2tensor.shtml

2.5.10 *-endian (+)*

Arguments: _datatype

Reverse data endianness of selected images, eventually considering the pixel being of the specified datatype. 'datatype' can be { uchar | char | ushort | short | uint | int | ulong | long | float | double }.

2.5.11 *-equalize* (+)

Arguments: `_nb_levels>0[%], _value_min[%], _value_max[%]`

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%'.



Example 131 : `image.jpg --equalize`

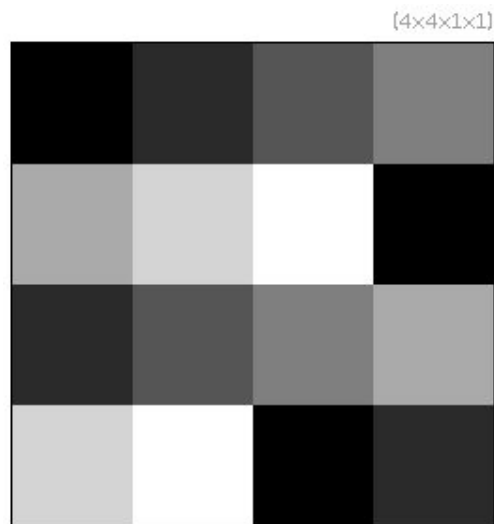


Example 132 : `image.jpg --equalize 4,0,128`

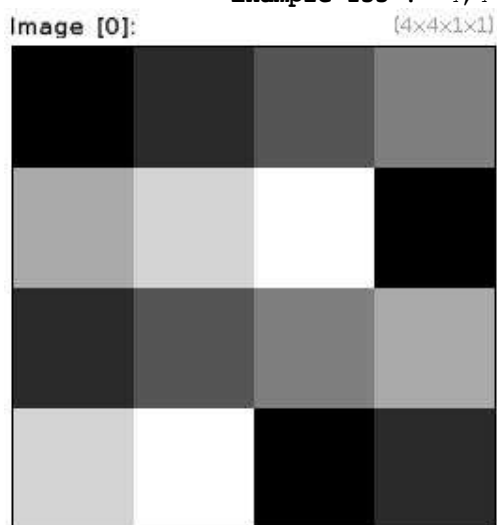
2.5.12 -fill (+)

Arguments: value1, _value2, .. |
 [image] |
 'formula'

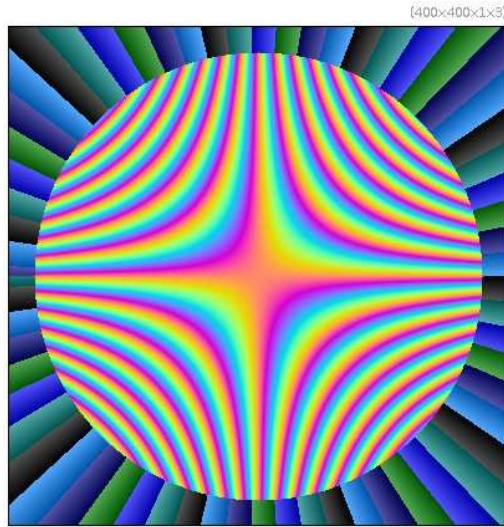
Fill selected images with values read from the specified value list, existing image or mathematical expression. Single quotes may be omitted in 'formula'.
 (eq. to '-f').



Example 133 : 4,4 -fill 1,2,3,4,5,6,7



Example 134 : 4,4 (1,2,3,4,5,6,7) -fill[-2] [-1]



Example 135 : `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))))"`

2.5.13 *-float2int8*

Convert selected float-valued images to 8bits integer representations.

2.5.14 *-int82float*

Convert selected 8bits integer representations to float-valued images.

2.5.15 *-index (+)*

Arguments: `{ [palette] | predefined.palette
,0<=_dithering<=1, map.palette={ 0 | 1 }`

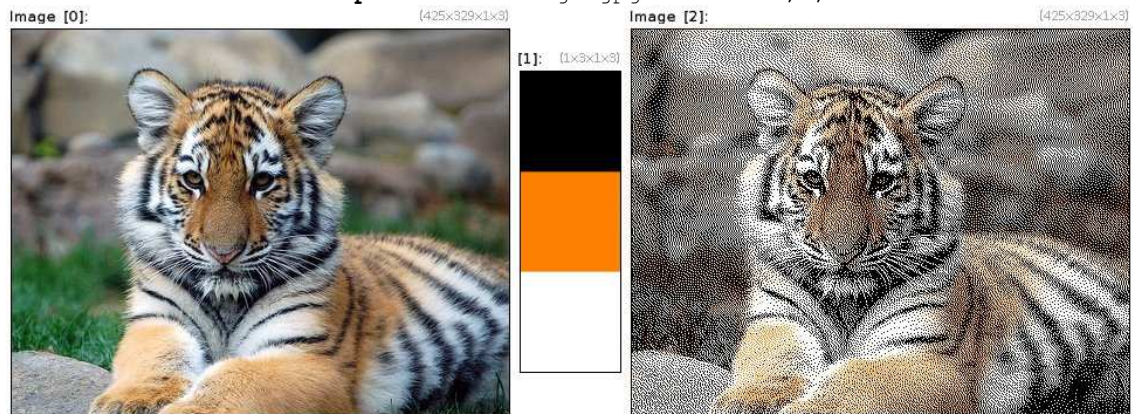
Index selected vector-valued images by specified vector-valued palette.

'predefined.palette' can be { 0=default | 1=HSV | 2=lines
| 3=hot | 4=cool | 5=jet | 6=flag | 7=cube }.

Default values: 'dithering=0' and 'map.palette=0'.



Example 136 : `image.jpg --index 1,1,1`



Example 137 : `image.jpg (0;255;255^0;128;255^0;0;255) --index[-2] [-1],1,1`

2.5.16 *-inrange*

Arguments: `min[%],max[%]`

Detect pixels whose values are in specified range [min,max],
in selected images.
(eq. to '-ir').



Example 138 : `image.jpg --inrange 25%,75%`

2.5.17 `-map (+)`

Arguments: `[palette],_boundary` | `predefined_palette,_boundary`

Map specified vector-valued palette to selected indexed scalar images.

'predefined_palette' can be { 0=default | 1=HSV | 2=lines | 3=hot | 4=cool | 5=jet | 6=flag | 7=cube }.

'boundary' can be { 0=dirichlet | 1=neumann | 2=periodic }.

Default value: 'boundary=0'.



Example 139 : `image.jpg --luminance -map[-1] 3`



Example 140 : `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]`

2.5.18 *-map_clut*

Arguments: `[clut]`

Map specified RGB color LUT to selected images.



Example 141 : `image.jpg -uniform.distribution {2^6},3 -mirror[-1] x --map_clut[0] [1]`

2.5.19 *-mix_channels*

Arguments: `(a00,...,aMN)`

Apply specified matrix to channels of selected images.



Example 142 : `image.jpg --mix_channels (0,1,0;1,0,0;0,0,1)`

2.5.20 *-negative*

Compute negative of selected images.



Example 143 : `image.jpg --negative`

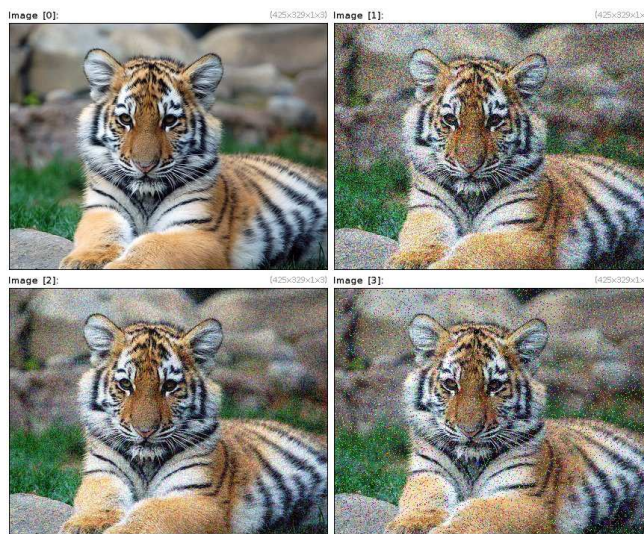
2.5.21 *-noise (+)*

Arguments: `std_variation>=0[%],_noise_type`

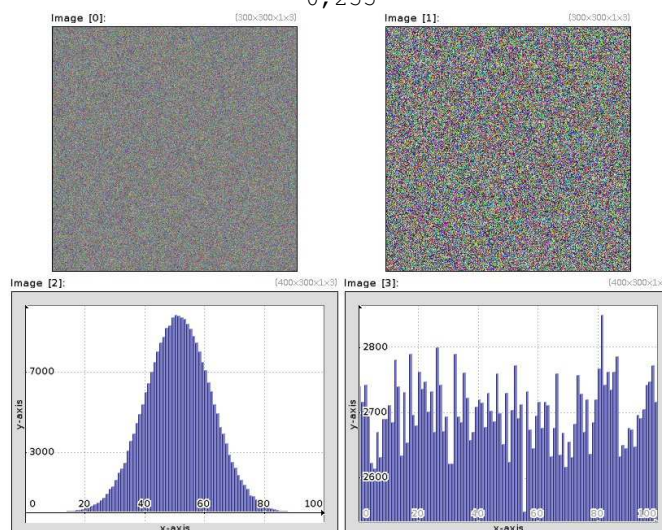
Add random noise to selected images.

'noise_type' can be { 0=gaussian | 1=uniform | 2=salt&pepper
| 3=poisson | 4=rice }.

Default value: `'noise_type=0'`.



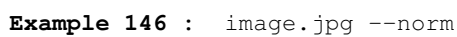
Example 144 : `image.jpg --noise[0] 50,0 --noise[0] 50,1 --noise[0] 10,2 -cut 0,255`



Example 145 : `300,300,1,3 [0] -noise[0] 20,0 -noise[1] 20,1 --histogram 100 -display_graph[-2,-1] 400,300,3`

2.5.22 -norm

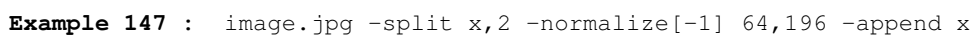
Compute the pointwise euclidean norm of vector-valued pixels in selected images.



http://gmics.eu/tutorial/_norm.shtml

```
Arguments:      { value0[%] | [image0] }, { value1[%] | [image1] } |
                  [image]
```

(eq. to $'-n'$).

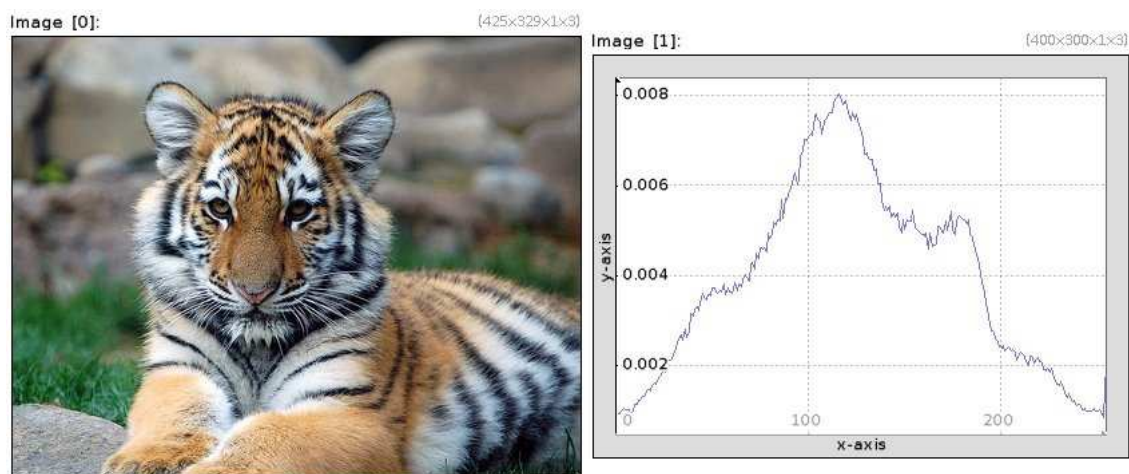


Tutorial page:

http://gmic.eu/tutorial/_normalize.shtml

2.5.24 -normalize_sum

Normalize selected images with a unitary sum.



Example 148 : `image.jpg --histogram[-1] -normalize_sum[-1] -display_graph[-1]
400,300`

2.5.25 -not

Apply boolean not operation on selected images.



Example 149 : `image.jpg --ge 50% --not[-1]`

2.5.26 -orientation

Compute the pointwise orientation of vector-valued pixels in selected images.



Example 150 : `image.jpg --orientation --norm[-2] -negative[-1] -mul[-2] [-1] -reverse[-2,-1]`

Tutorial page:

http://gmic.eu/tutorial/_orientation.shtml

2.5.27 *-oneminus*

For each selected image, compute one minus image.



Example 151 : `image.jpg -n 0,1 --oneminus`

2.5.28 *-otsu*

Arguments: `_nb_levels>0`

Hard-threshold selected images using Otsu's method.

The computed thresholds are returned as a list of values in the status.

Default value: `'nb_levels=256'`.



Example 152 : `image.jpg -luminance --otsu ,`

2.5.29 *-polar2complex*

Compute polar to complex transforms of selected images.

2.5.30 *-quantize*

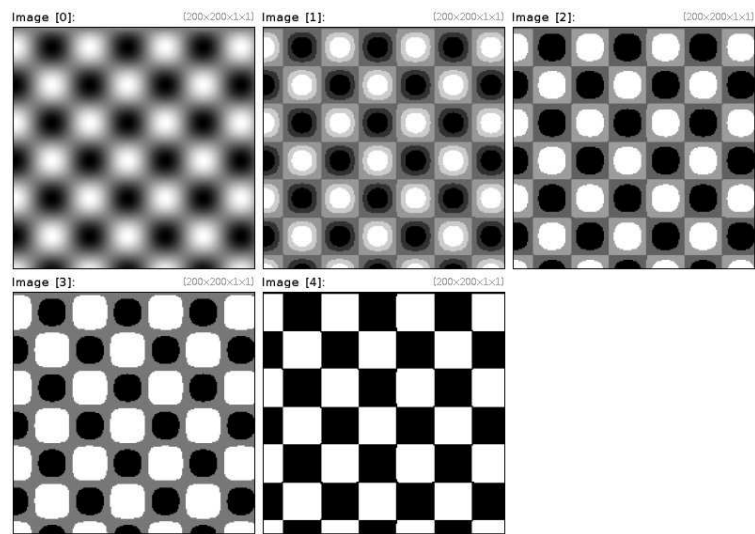
Arguments: `nb_levels>=1,keep_values={ 0 | 1 },_is_uniform={ 0 | 1 }`

Quantize selected images.

Default value: `'keep_values=1'` and `'is_uniform=0'`.



Example 153 : `image.jpg -luminance --quantize 3`

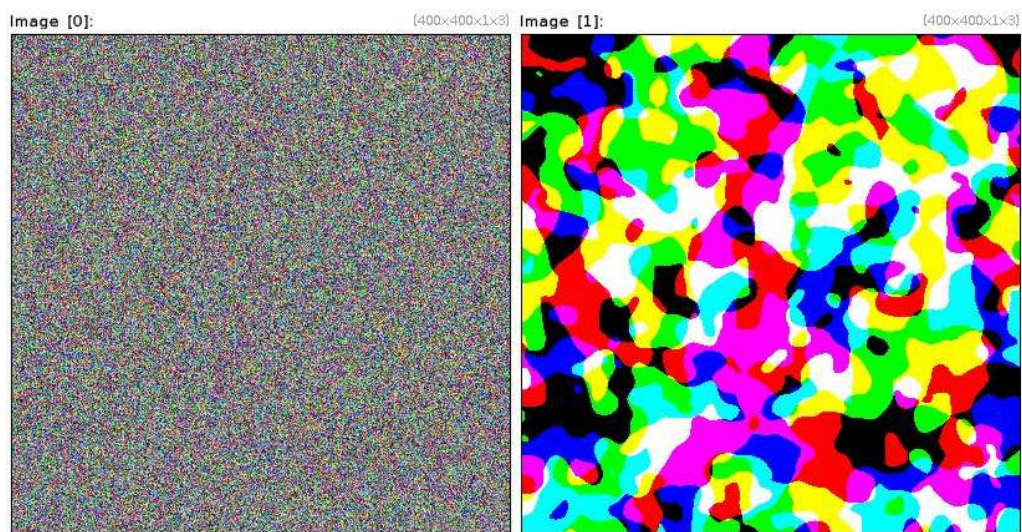


Example 154 : `200,200,1,1,'cos(x/10)*sin(y/10)' --quantize[0] 6 --quantize[0] 4 --quantize[0] 3 --quantize[0] 2`

2.5.31 *-rand (+)*

Arguments: `{ value0[%] | [image0] }, { value1[%] | [image1] } | [image]`

Fill selected images with random values uniformly distributed in the specified range.

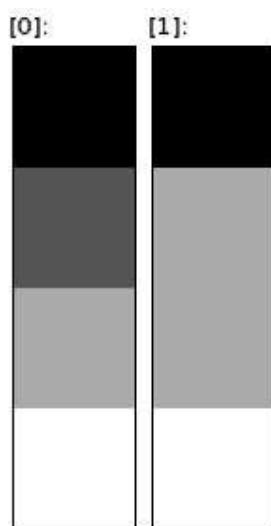


Example 155 : `400,400,1,3 -rand -10,10 --blur 10 -sign[-1]`

2.5.32 *-replace*

Arguments: `value_src, value_dest`

Replace pixel values in selected images.

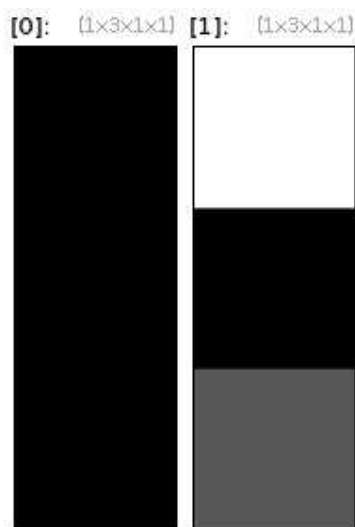


Example 156 : `(1;2;3;4) --replace 2,3`

2.5.33 *-replace_inf*

Arguments: `_expression`

Replace all infinite values in selected images by specified expression.

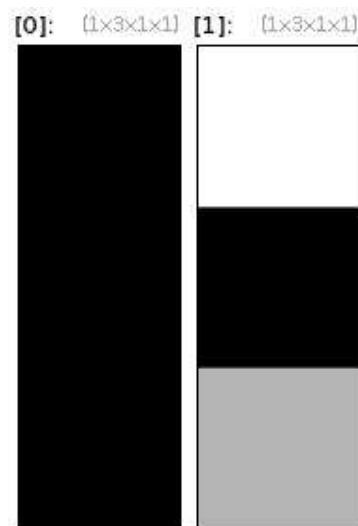


Example 157 : `(0;1;2) -log --replace.inf 2`

2.5.34 *-replace nan*

Arguments: `_expression`

Replace all NaN values in selected images by specified expression.



Example 158 : `(-1;0;2) -sqrt --replace.nan 2`

2.5.35 *-replace seq*

Arguments: `"search_seq", "replace_seq"`

Search and replace a sequence of values in selected images.

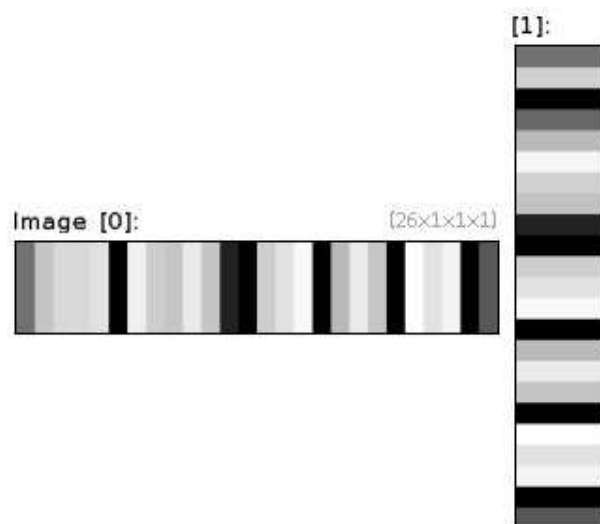


Example 159 : `(1;2;3;4;5) --replace_seq "2,3,4","7,8"`

2.5.36 *-replace_str*

Arguments: `"search_str", "replace_str"`

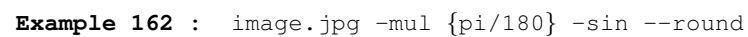
Search and replace a string in selected images (viewed as strings, i.e. sequences of ascii codes).



Example 160 : `{{"Hello there, how are you ?"}} --replace_str "Hello there", "Hi David"`

Arguments: rounding_value>=0, _rounding_type
 (no arg)

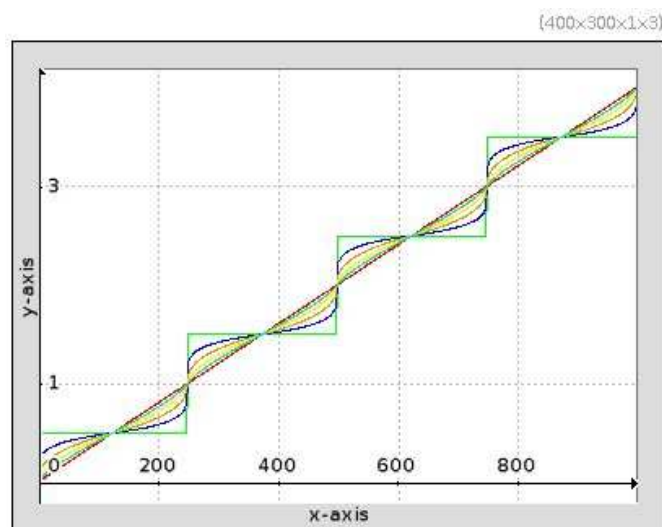
Default value: `'rounding_type=0'`.



Arguments: `gamma` ≥ 0

Apply roundify transformation on float-valued data, with specified gamma.

Default value: 'gamma=0'.



Example 163 : `1000 -fill '4*x/w' -repeat 5 --roundify[0] {$>*0.2} -done
-append c -display-graph 400,300`

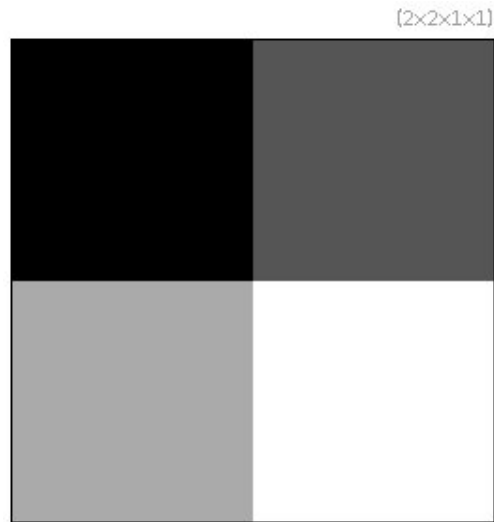
2.5.39 -set (+)

Arguments: value, x[%], y[%], z[%], c[%]

Set pixel value in selected images, at specified coordinates.
(eq. to '--').

If specified coordinates are outside the image bounds, no action is performed.

Default values: 'x=y=z=c=0'.



Example 164 : `2,2 -set 1,0,0 -set 2,1,0 -set 3,0,1 -set 4,1,1`
(425x329x1x3)



Example 165 : `image.jpg -repeat 10000 -set 255,{?(100)}%,{?(100)}%,0,{?(100)}%`
-done

2.5.40 *-threshold (+)*

Arguments: `value[%],_is_soft={ 0 | 1 } |`
(no arg)

Threshold values of selected images.
 'soft' can be { 0=hard-thresholding | 1=soft-thresholding }.
 (no arg) runs interactive mode (uses the instant display window
 [0] if opened).

Default value: `'is-soft=0'`.



Example 166 : `image.jpg --threshold[0] 50% --threshold[0] 50%,1`

Tutorial page:

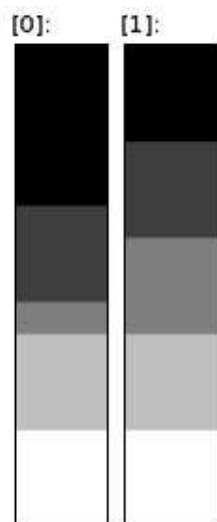
http://gmic.eu/tutorial/_threshold.shtml

2.5.41 *-uncompress rle*

Uncompress selected 2xN data matrices, using RLE algorithm.

2.5.42 *-unrepeat*

Remove repetition of adjacent values in selected images.



Example 167 : `(1;1;1;1;1;2;2;2;3;4;4;4;5;5;5) --unrepeat`

2.5.43 *-vector2tensor*

Convert selected vector fields to corresponding tensor fields.

2.6 Colors manipulation

2.6.1 *-apply_channels*

Arguments: `"command", channels, _value_action={ 0=none | 1=cut
| 2=normalize }`

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

(eq. to `'-ac'`).

Argument `'channels'` refers to a colorspace, and can be basically one of { `all` | `rgba` | `rgb` | `lrgb` | `ycbcr` | `lab` | `lch` | `hsv` | `hsi` | `hsl` | `cmv` | `cmk` }.

You can also focus the processing on one particular channel of this colorspace, by setting `'channels'` as `'colorspace-channel'` (e.g. `'hsv_h'` for the hue).

All channel values are considered to be in the `[0,255]` range.

Default value: `'value_action=0'`.



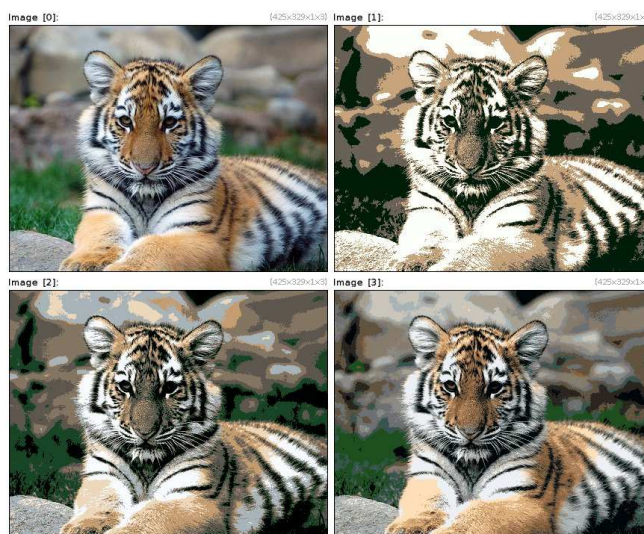
Example 168 : `image.jpg --apply-channels "-equalize -blur 2",ycbcr_cbc`

2.6.2 *-autoindex*

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

Index selected vector-valued images by adapted colormaps.

Default values: `'dithering=0'` and `'method=0'`.



Example 169 : `image.jpg --autoindex[0] 4 --autoindex[0] 8 --autoindex[0] 16`

2.6.3 *-bayer2rgb*

Arguments: `_GM_smoothness, _RB_smoothness1, _RB_smoothness2`

Transform selected RGB-Bayer sampled images to color images.

Default values: `'GM_smoothness=RB_smoothness=1'` and `'RB_smoothness2=0.5'`.



Example 170 : `image.jpg -rgb2bayer 0 --bayer2rgb 1,1,0.5`

2.6.4 *-cmy2rgb*

Convert selected images from CMY to RGB colorbases.

2.6.5 *-cmyk2rgb*

Convert selected images from CMYK to RGB colorbases.

2.6.6 *-colormap*

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

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

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

Default value: 'method=0' and 'sort_vectors=1'.



Example 171 : `image.jpg --colormap[0] 4 --colormap[0] 8 --colormap[0] 16`

2.6.7 *-compose_channels*

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

Default value: '1=+'.



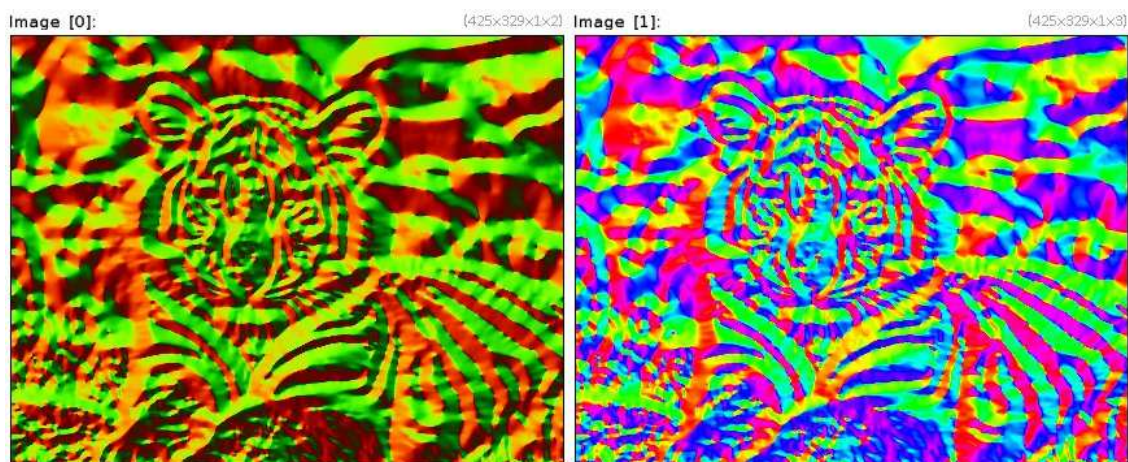
Example 172 : `image.jpg --compose_channels` and

Tutorial page:

http://gmic.eu/tutorial/_compose_channels.shtml

2.6.8 *-direction2rgb*

Compute RGB representation of selected 2d direction fields.



Example 173 : `image.jpg -luminance -gradient -append c -blur 2 -orientation --direction2rgb`

2.6.9 *-ditheredbw*

Create dithered B&W version of selected images.



Example 174 : `image.jpg --equalize -ditheredbw[-1]`

2.6.10 *-fill_color*

Arguments: `col1,...,colN`

Fill selected images with specified color.
(eq. to `'-fc'`).



Example 175 : `image.jpg --fill_color 255,0,255`

2.6.11 *-gradient2rgb*

Arguments: `_is_orientation={ 0 | 1 }`

Compute RGB representation of 2d gradient of selected images.

Default value: `'is_orientation=0'`.



Example 176 : `image.jpg --gradient2rgb 0 -equalize[-1]`

2.6.12 *-hsi2rgb (+)*

Convert selected images from HSI to RGB colorbases.

2.6.13 *-hsi82rgb*

Convert selected images from HSI8 to RGB color bases.

2.6.14 *-hsl2rgb (+)*

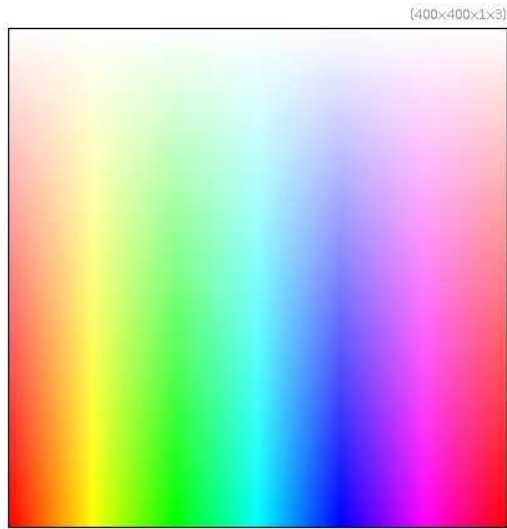
Convert selected images from HSL to RGB colorbases.

2.6.15 *-hsl82rgb*

Convert selected images from HSL8 to RGB color bases.

2.6.16 *-hsv2rgb (+)*

Convert selected images from HSV to RGB colorbases.



Example 177 : `(0,360;0,360^0,0;1,1^1,1;1,1) -resize 400,400,1,3,3 -hsv2rgb`

2.6.17 *-hsv82rgb*

Convert selected images from HSV8 to RGB color bases.

2.6.18 *-int2rgb*

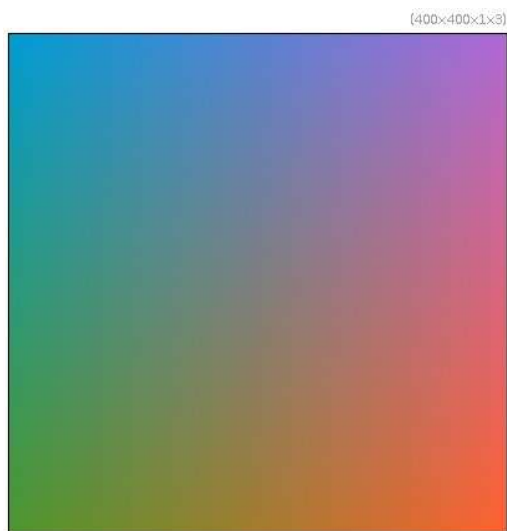
Convert selected images from INT24 scalars to RGB.

2.6.19 *-lab2lch*

Convert selected images from Lab to Lch color bases.

2.6.20 *-lab2rgb (+)*

Convert selected images from Lab to RGB colorbases.



Example 178 : `(50,50;50,50^-3,3;-3,3^-3,-3;3,3) -resize 400,400,1,3,3
-lab2rgb`

2.6.21 *-lab82rgb*

Convert selected images from Lab8 to RGB color bases.

2.6.22 *-lch2lab*

Convert selected images from Lch to Lab color bases.

2.6.23 *-lch2rgb*

Convert selected images from Lch to RGB color bases.

2.6.24 *-lch82rgb*

Convert selected images from Lch8 to RGB color bases.

2.6.25 *-luminance*

Compute luminance of selected sRGB images.



Example 179 : `image.jpg --luminance`

2.6.26 *-mix_rgb*

Arguments: `a11,a12,a13,a21,a22,a23,a31,a32,a33`

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'`.



Example 180 : `image.jpg --mix-rgb 0,1,0,1,0,0,0,0,1`

Tutorial page:

http://gmic.eu/tutorial/_mix-rgb.shtml

2.6.27 *-pseudogray*

Arguments: `_max_increment>=0, _JND_threshold>=0, _bits_depth>0`

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 181 : `-pseudogray 5`

2.6.28 *-replace_color*

Arguments: `tolerance[%]>=0,smoothness[%]>=0,src1,src2,...,dest1,dest2,..`
`..`

Replace pixels from/to specified colors in selected images.



Example 182 : `image.jpg --replace-color 40,3,204,153,110,255,0,0`

2.6.29 *-rgb2bayer*

Arguments: `_start_pattern=0,_color_grid=0`

Transform selected color images to RGB-Bayer sampled images.

Default values: `'start_pattern=0'` and `'color_grid=0'`.



Example 183 : `image.jpg --rgb2bayer 0`

2.6.30 *-rgb2cmy*

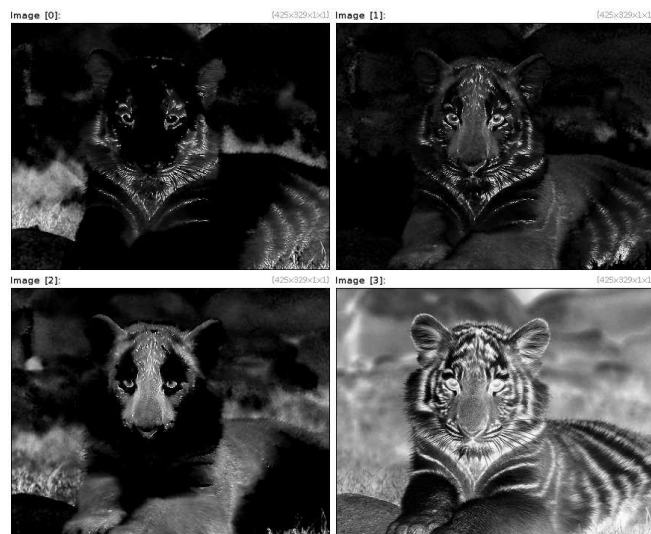
Convert selected images from RGB to CMY colorbases.



Example 184 : `image.jpg -rgb2cmy -split c`

2.6.31 *-rgb2cmyk*

Convert selected images from RGB to CMYK colorbases.



Example 185 : `image.jpg -rgb2cmyk -split c`
(425x329x1x3)



Example 186 : `image.jpg -rgb2cmyk -split c -fill[3] 0 -append c -cmyk2rgb`

2.6.32 -rgb2hsi (+)

Convert selected images from RGB to HSI colorbases.



Example 187 : `image.jpg -rgb2hsi -split c`

2.6.33 *-rgb2hsi8*

Convert selected images from RGB to HSI8 color bases.



Example 188 : `image.jpg -rgb2hsi8 -split c`

2.6.34 *-rgb2hsl (+)*

Convert selected images from RGB to HSL colorbases.



Example 189 : `image.jpg -rgb2hsl -split c`



Example 190 : `image.jpg -rgb2hsl --split c -add[-3] 100 -mod[-3] 360
-append[-3--1] c -hsl2rgb`

2.6.35 -rgb2hsl8

Convert selected images from RGB to HSL8 color bases.



Example 191 : `image.jpg -rgb2hsl8 -split c`

2.6.36 -rgb2hsv (+)

Convert selected images from RGB to HSV colorbases.



Example 192 : `image.jpg -rgb2hsv -split c`



Example 193 : `image.jpg -rgb2hsv --split c -add[-2] 0.3 -cut[-2] 0,1
-append[-3--1] c -hsv2rgb`

2.6.37 -rgb2hsv8

Convert selected images from RGB to HSV8 color bases.



Example 194 : `image.jpg -rgb2hsv8 -split c`

2.6.38 `-rgb2lab (+)`

Convert selected images from RGB to Lab colorbases.



Example 195 : `image.jpg -rgb2lab -split c`



Example 196 : `image.jpg -rgb2lab --split c -mul[-2,-1] 2.5 -append[-3--1] c -lab2rgb`

2.6.39 `-rgb2lab8`

Convert selected images from RGB to Lab8 color bases.



Example 197 : `image.jpg -rgb2lab8 -split c`

2.6.40 *-rgb2lch*

Convert selected images from RGB to Lch color bases.



Example 198 : `image.jpg -rgb2lch -split c`

2.6.41 *-rgb2lch8*

Convert selected images from RGB to Lch8 color bases.



Example 199 : `image.jpg -rgb2lch8 -split c`

2.6.42 *-rgb2luv*

Convert selected images from RGB to LUV color bases.



Example 200 : `image.jpg -rgb2luv -split c`

2.6.43 *-rgb2int*

Convert selected images from RGB to INT24 scalars.



Example 201 : `image.jpg -rgb2int`

2.6.44 *-rgb2srgb (+)*

Convert selected images from RGB to sRGB colorbases.

2.6.45 *-rgb2xyz*

Convert selected images from RGB to XYZ colorbases. (the D65 illuminant is used as the white point).



Example 202 : `image.jpg -rgb2xyz -split c`

2.6.46 *-rgb2xyz8*

Convert selected images from RGB to XYZ8 color bases.



Example 203 : `image.jpg -rgb2xyz8 -split c`

2.6.47 *-rgb2ycbcr*

Convert selected images from RGB to YCbCr colorbases.



Example 204 : `image.jpg -rgb2ycbcr -split c`

2.6.48 *-rgb2yuv*

Convert selected images from RGB to YUV colorbases.



Example 205 : `image.jpg -rgb2yuv -split c`

2.6.49 *-rgb2yuv8*

Convert selected images from RGB to YUV8 color bases.



Example 206 : `image.jpg -rgb2yuv8 -split c`

2.6.50 *-remove_opacity*

Remove opacity channel of selected images.

2.6.51 *-select_color*

Arguments: `tolerance[%]>=0,col1,...,colN`

Select pixels with specified color in selected images.



Example 207 : `image.jpg --select_color 40,204,153,110`

Tutorial page:

http://gmic.eu/tutorial/_select_color.shtml

2.6.52 *-sepia*

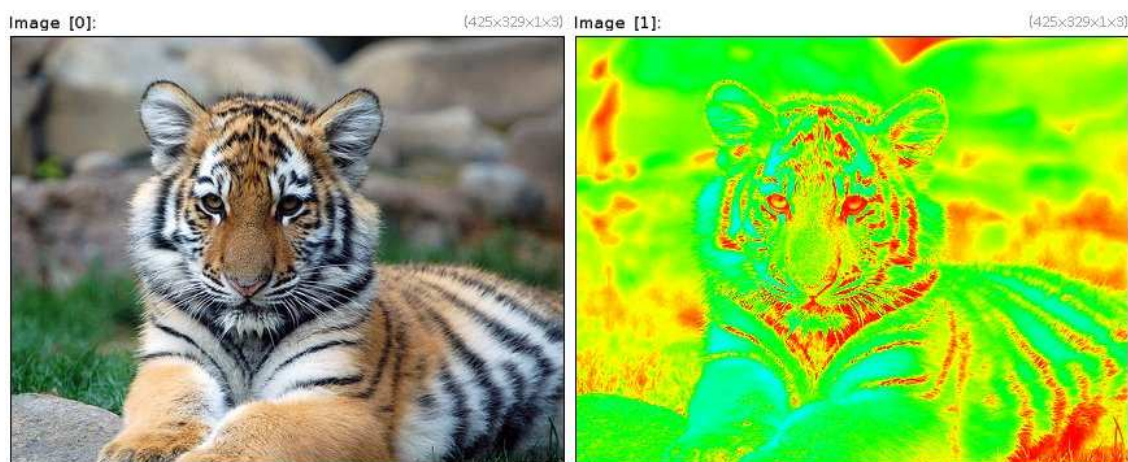
Apply sepia tones effect on selected images.



Example 208 : `image.jpg --sepia`

2.6.53 *-solarize*

Solarize selected images.



Example 209 : `image.jpg --solarize`

2.6.54 *-split_colors*

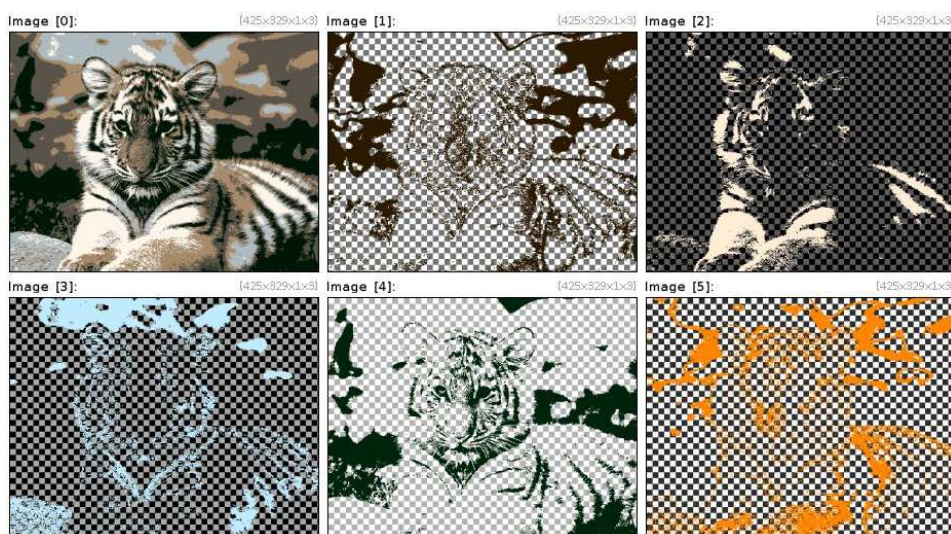
Arguments: `_tolerance>=0, _max.nb.outputs>0, _min.area>0`

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 210 : `image.jpg -quantize 5 --split_colors , -display_rgba`

2.6.55 -split_opacity

Split color and opacity parts of selected images.

2.6.56 -srgb2rgb (+)

Convert selected images from sRGB to RGB colorbases.

2.6.57 -to_a

Force selected images to have an alpha channel.

2.6.58 -to_color

Force selected images to be in color mode (RGB or RGBA).

2.6.59 -to_colormode

Arguments: mode={ 0=adaptive | 1=G | 2=GA | 3=RGB | 4=RGBA }

Force selected images to be in a given color mode.

Default value: 'mode=0'.

2.6.60 -to_gray

Force selected images to be in GRAY mode.



Example 211 : image.jpg --to-gray

2.6.61 -to_graya

Force selected images to be in GRAYA mode.

2.6.62 -to_pseudogray

Arguments: `_max_step>=0, _is_perceptual_constraint={ 0 | 1 }, _bits_depth>0`

Convert selected scalar images ([0-255]-valued) to pseudo-gray color images.

Default parameters : `'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 Tschumperle.

2.6.63 -to_rgb

Force selected images to be in RGB mode.

2.6.64 -to_rgba

Force selected images to be in RGBA mode.

2.6.65 -transfer_colors

Arguments: `[reference_image], _transfer_brightness={ 0 | 1 }`

Transfer colors of the specified reference image to selected images.

Default value: `'transfer_brightness=0'`.



Example 212 : `image.jpg --rand 0,255 --transfer_colors[0] [1],1`

2.6.66 -xyz2rgb

Convert selected images from XYZ to RGB colorbases.

2.6.67 -xyz82rgb

Convert selected images from XYZ8 to RGB color bases.

2.6.68 -ycbcr2rgb

Convert selected images from YCbCr to RGB colorbases.

2.6.69 -yuv2rgb

Convert selected images from YUV to RGB colorbases.

2.6.70 -yuv82rgb

Convert selected images from YUV8 to RGB color bases.

2.7 Geometry manipulation

2.7.1 -append (+)

Arguments: [image],axis,_centering |
 axis,_centering

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

(eq. to '-a').

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

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

Default value: 'centering=0'.



Example 213 : `image.jpg -split y,10 -reverse -append y`



Example 214 : `image.jpg -repeat 5 --rows[0] 0,{10+18*$>}% -done -remove[0]
-append x,0.5`



Example 215 : `image.jpg -append[0] [0],y`

2.7.2 *-append tiles*

Arguments: `_M>=0, _N>=0, 0<=_centering_x<=1, 0<=_centering_y<=1`

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'.

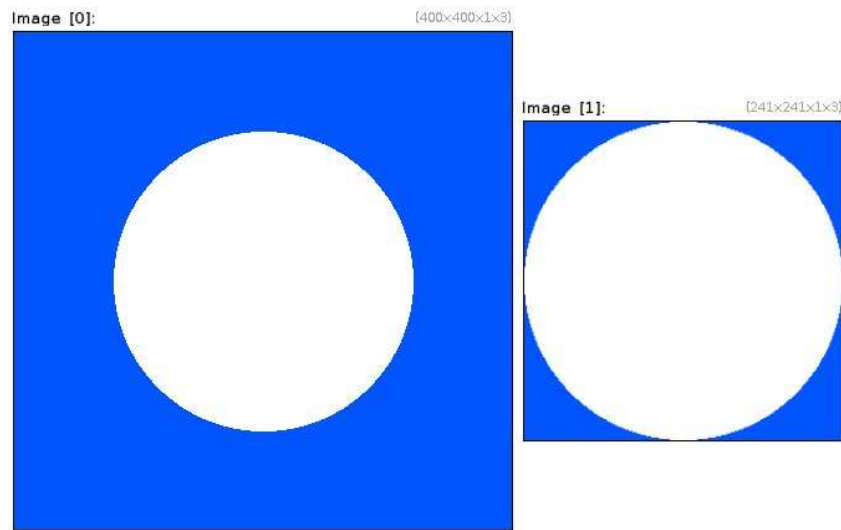


Example 216 : `image.jpg -split xy,4 -appendtiles ,`

2.7.3 -autocrop (+)

Arguments: value1,value2,... |
 (no arg)

Autocrop selected images by specified vector-valued intensity. If no arguments are provided, cropping value is guessed.



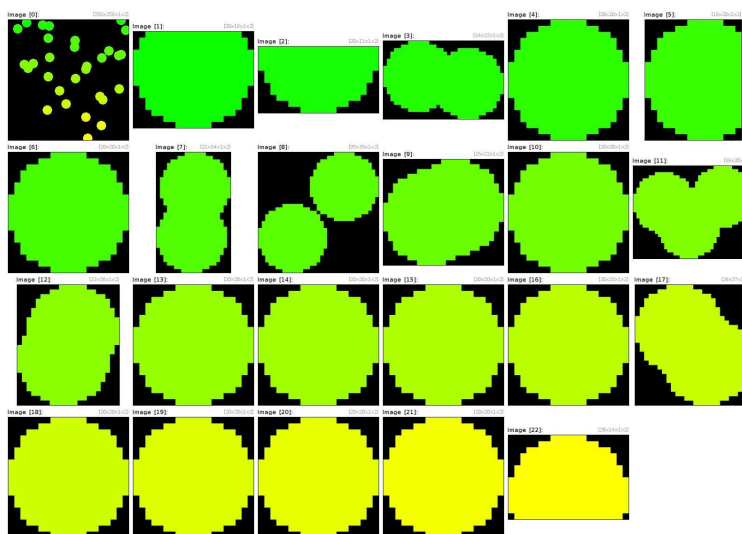
Example 217 : `400,400,1,3 -fill_color 64,128,255 -ellipse
50%,50%,120,120,0,1,255 --autocrop`

2.7.4 *-autocrop_components*

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

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'.`



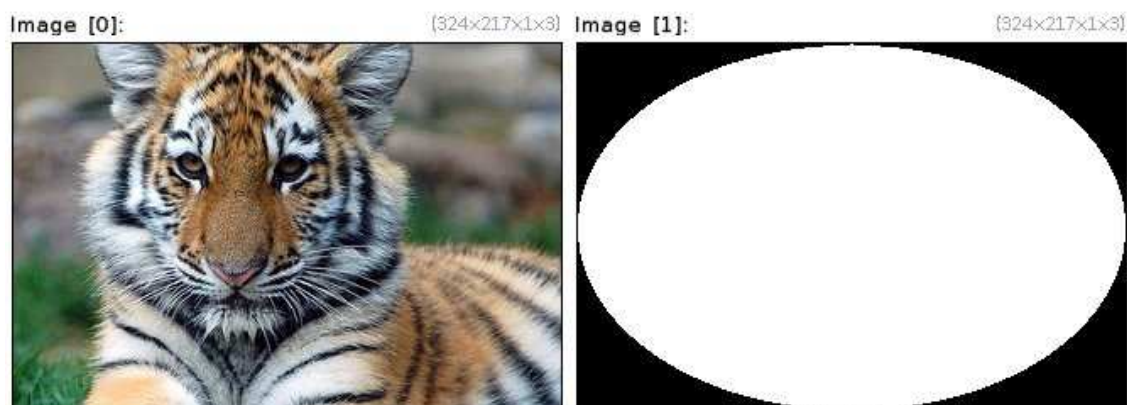
Example 218 : `256,256 -noise 0.1,2 -dilate-circ 20 -label-fg 0,1 -normalize 0,255 --neg 0 --*[-1] 255 -append c --autocrop-components ,`

2.7.5 -autocrop_seq

Arguments: `value1,value2,...` | `auto`

Autocrop selected images using the crop geometry of the last one by specified vector-valued intensity, or by automatic guessing the cropping value.

Default value: `auto mode.`



Example 219 : `image.jpg --fill[-1] 0 -ellipse[-1] 50%,50%,30%,20%,0,1,1 -autocrop_seq 0`

2.7.6 *-channels* (+)

Arguments: { [image0] | c0[%] },-{ [image1] | c1[%] }

Keep only specified channels of selected images.
Dirichlet boundary is used when specified channels are out of range.



Example 220 : image.jpg -channels 0,1
(425x329x1x3)



Example 221 : image.jpg -luminance -channels 0,2

2.7.7 *-columns* (+)

Arguments: { [image0] | x0[%] },-{ [image1] | x1[%] }

Keep only specified columns of selected images.
Dirichlet boundary is used when specified columns are out of range.



Example 222 : `image.jpg -columns -25%,50%`

2.7.8 *-crop* (+)

Arguments: `x0[%],x1[%],_boundary` |
`x0[%],y0[%],x1[%],y1[%],_boundary` |
`x0[%],y0[%],z0[%],x1[%],y1[%],z1[%],_boundary` |
`x0[%],y0[%],z0[%],c0[%],x1[%],y1[%],z1[%],c1[%],_boundary` |
 (no arg)

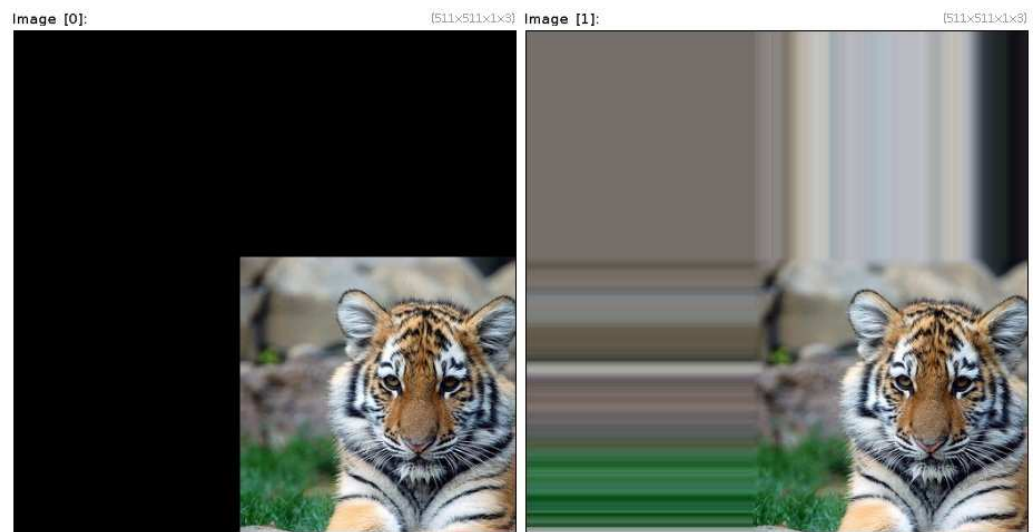
Crop selected images with specified region coordinates.

(eq. to '-z').

'boundary' can be { 0=dirichlet | 1=neumann }.

(no arg) runs interactive mode (uses the instant display window [0] if opened).

Default value: 'boundary=0'.



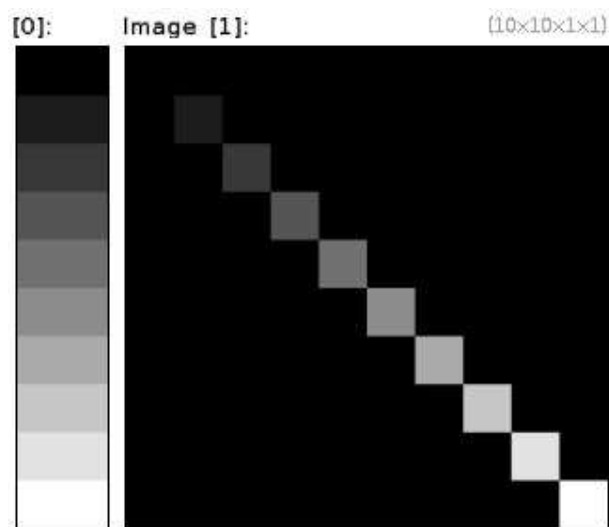
Example 223 : `image.jpg --crop -230,-230,280,280,1 -crop[0]
-230,-230,280,280,0`



Example 224 : `image.jpg -crop 25%,25%,75%,75%`

2.7.9 *-diagonal*

Transform selected vectors as diagonal matrices.



Example 225 : `1,10,1,1,'y' --diagonal`

2.7.10 *-elevate*

Arguments: `_depth, _is_plain={ 0 | 1 }, _is_colored={ 0 | 1 }`

Elevate selected 2d images into 3d volumes.

Default values: `'depth=64', 'is_plain=1' and 'is_colored=1'.`

2.7.11 *-expand_x*

Arguments: `size_x>=0, boundary={ 0=dirichlet | 1=neumann
| 2=periodic }`

Expand selected images along the x-axis.

Default value: `'border=1'.`



Example 226 : `image.jpg -expand_x 30,0`

2.7.12 *-expand_xy*

Arguments: `size>=0, boundary={ 0=dirichlet | 1=neumann
| 2=periodic }`

Expand selected images along the xy-axes.

Default value: `'border=1'`.



Example 227 : `image.jpg -expand_xy 30,0`

2.7.13 *-expand xyz*

Arguments: `size>=0, boundary={ 0=dirichlet | 1=neumann
| 2=periodic }`

Expand selected images along the xyz-axes.

Default value: `'border=1'`.

2.7.14 *-expand y*

Arguments: `size-y>=0, boundary={ 0=dirichlet | 1=neumann
| 2=periodic }`

Expand selected images along the y-axis.

Default value: `'border=1'`.



Example 228 : `image.jpg -expand_y 30,0`

2.7.15 *-expand z*

Arguments: `size-z>=0, boundary={ 0=dirichlet | 1=neumann
| 2=periodic }`

Expand selected images along the z-axis.

Default value: `'border=1'`.

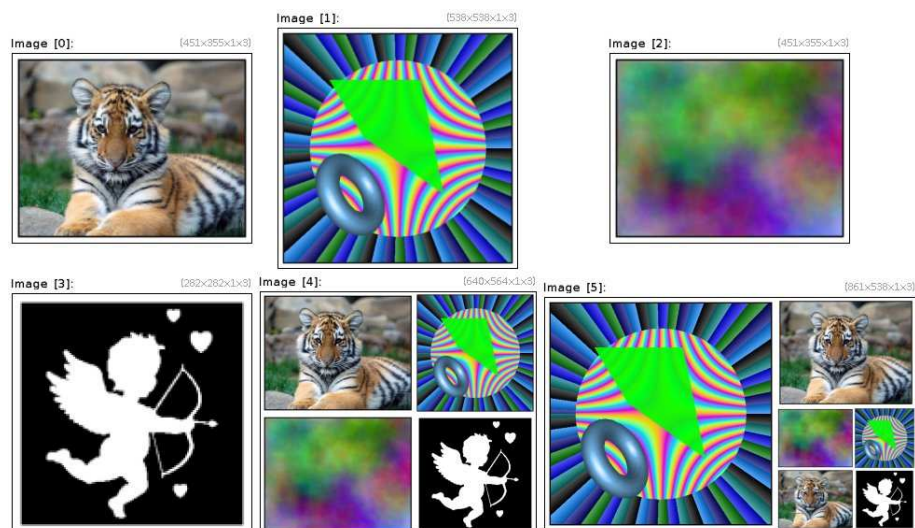
2.7.16 *-montage*

Arguments: `"_layout_code",_montage_mode={ 0<=centering<=1
| 2<=scale+2<=3 },_output_mode={ 0=single layer | 1=multiple layers
},"_processing_command"`

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 indice (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 229 : `image.jpg -testimage2d 512 --plasma[0] -cupid 256 -normalize
0,255 -frame 3,3,0 -frame 10,10,255 -to-rgb --montage A --montage[^-1]
H1:V0:VH2:1H0:3`

2.7.17 -mirror (+)

Arguments: { x | y | z }..{ x | y | z }

Mirror selected images along specified axes.



Example 230 : `image.jpg --mirror y --mirror[0] c`
 (850x658x1x3)



Example 231 : `image.jpg --mirror x --mirror y -appendtiles 2,2`

2.7.18 -permute (+)

Arguments: permutation_string

Permute selected image axes by specified permutation.
 'permutation' is a combination of the character set
 {x | y | z | c}, e.g. 'xycz', 'cxyz', ..



Example 232 : `image.jpg -permute yxzc`

2.7.19 *-resize (+)*

Arguments: `[image],_interpolation,_boundary,_ax,_ay,_az,_ac` |
`{[image_w] | width>0[%]},-[image_h] | height>0[%]},-[image-`
`_d] | depth>0[%]},-[image_s] | spectrum>0[%]},_interpolatio-`
`n,_boundary,_ax,_ay,_az,_ac` |
 (no arg)

Resize selected images with specified geometry.

(eq. to `'-r'`).

`'interpolation'` can be { `-1=none` (memory content) | `0=none`
 | `1=nearest` | `2=average` | `3=linear` | `4=grid` | `5=bicubic`
 | `6=lanczos` }.

`'boundary'` has different meanings, according to the chosen

`'interpolation'` mode : . When `'interpolation'=={ -1 | 1 | 2`
 | `4 }`, `'boundary'` is meaningless. . When `'interpolation'==0`,

`'boundary'` can be { `0=dirichlet` | `1=neumann` | `2=periodic` }.

. When `'interpolation'=={ 3 | 5 | 6 }`, `'boundary'` can be {
`0=none` | `1=neumann` }.

`'ax,ay,az,ac'` set the centering along each axis when

`'interpolation=0 or 4'`

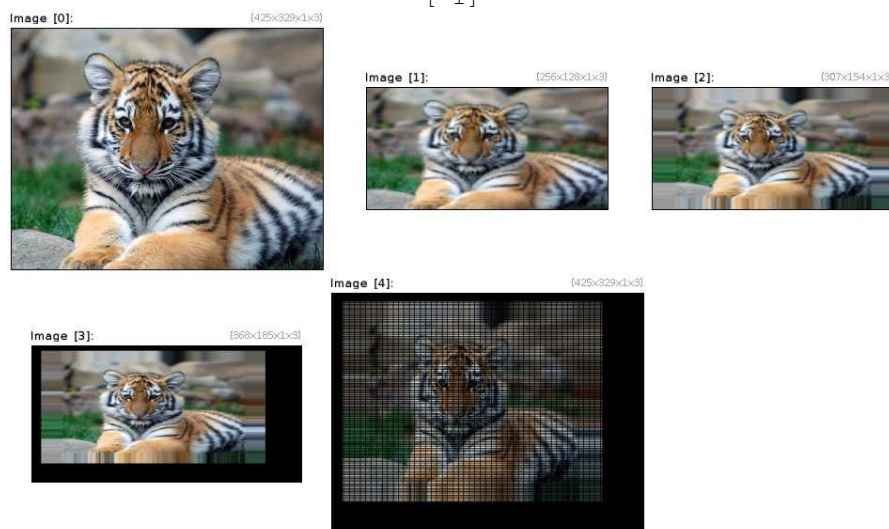
(set to `'0'` by default, must be defined in range `[0,1]`).

(no arg) runs interactive mode (uses the instant display window
`[0]` if opened).

Default values: `'interpolation=1'`, `'boundary=0'` and `'ax=ay=az=ac=0'`.



Example 233 : `image.jpg (0,1;0,1^0,0;1,1^1,1;1,1) -resize[-1] [-2],3 -mul[-2] [-1]`



Example 234 : `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`

2.7.20 -resize_pow2

Arguments: `_interpolation, _boundary, _ax, _ay, _az, _ac`

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' has different meanings, according to the chosen

'interpolation' mode : . When 'interpolation'=={ -1 | 1 | 2

| 4 }', 'boundary' is meaningless. . When 'interpolation==0', 'boundary' can be { 0=dirichlet | 1=neumann | 2=periodic }. . When 'interpolation=={ 3 | 5 | 6 }', 'boundary' 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=0' and 'ax=ay=az=ac=0'.



Example 235 : image.jpg --resize_pow2[-1] 0

2.7.21 -resize_ratio2d

Arguments: width>0,height>0,_mode={ 0=inside | 1=outside | 2=padded },0=<_interpolation<=6

Resize selected images while preserving their aspect ratio. (eq. to '-rr2d').

Default values: 'mode=0' and 'interpolation=6'.

2.7.22 -resize2dx

Arguments: width[%]>0,_interpolation,_boundary,_ax,_ay,_az,_ac

Resize selected images along the x-axis, preserving 2d ratio. (eq. to '-r2dx').

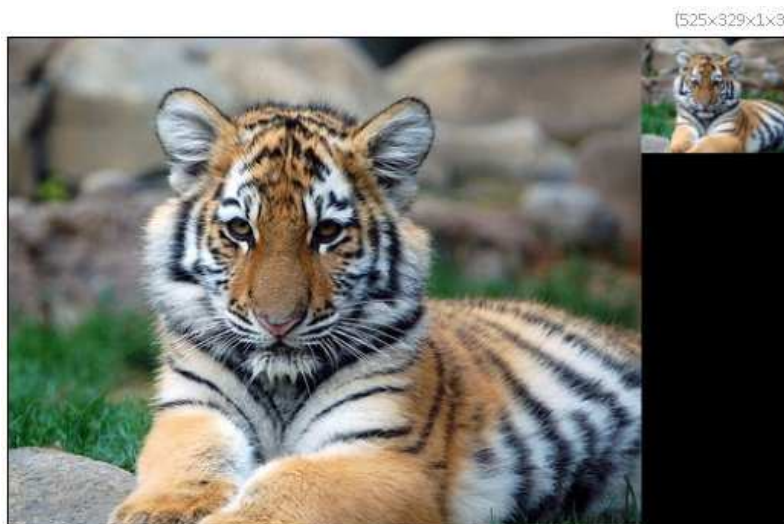
'interpolation' can be { -1=none (memory content) | 0=none

```
| 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic
| 6=lanczos }.
```

'boundary' has different meanings, according to the chosen 'interpolation' mode : . When 'interpolation=={ -1 | 1 | 2 | 4 }', 'boundary' is meaningless. . When 'interpolation==0', 'boundary' can be { 0=dirichlet | 1=neumann | 2=periodic }. . When 'interpolation=={ 3 | 5 | 6 }', 'boundary' 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=0' and 'ax=ay=az=ac=0'.



Example 236 : `image.jpg --resize2dx 100,2 -append x`

2.7.23 -resize2dy

Arguments: `height[%]>=0, _interpolation, _boundary, _ax, _ay, _az, _ac`

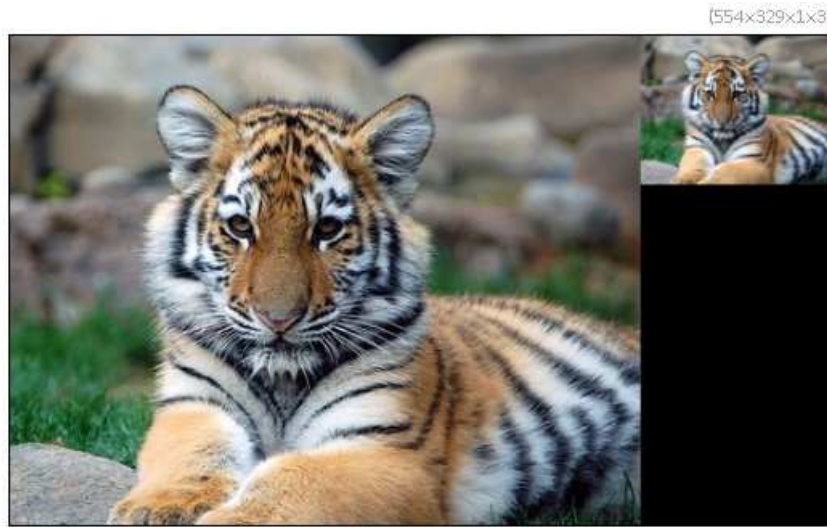
Resize selected images along the y-axis, preserving 2d ratio. (eq. to '-r2dy').

```
'interpolation' can be { -1=none (memory content) | 0=none
| 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic
| 6=lanczos }.
```

'boundary' has different meanings, according to the chosen 'interpolation' mode : . When 'interpolation=={ -1 | 1 | 2 | 4 }', 'boundary' is meaningless. . When 'interpolation==0', 'boundary' can be { 0=dirichlet | 1=neumann | 2=periodic }.

. When 'interpolation'=={ 3 | 5 | 6 }, 'boundary' 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=0' and 'ax=ay=az=ac=0'.



Example 237 : image.jpg --resize2dy 100,2 -append x

2.7.24 -resize3dx

Arguments: width[%]>0, interpolation, boundary, ax, ay, az, ac

Resize selected images along the x-axis, preserving 3d ratio.
 (eq. to '-r3dx').

'interpolation' can be { -1=none (memory content) | 0=none
 | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic
 | 6=lanczos }.

'boundary' has different meanings, according to the chosen
 'interpolation' mode : . When 'interpolation'=={ -1 | 1 | 2
 | 4 }, 'boundary' is meaningless. . When 'interpolation'==0',
 'boundary' can be { 0=dirichlet | 1=neumann | 2=periodic }.
 . When 'interpolation'=={ 3 | 5 | 6 }, 'boundary' 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=0' and 'ax=ay=az=ac=0'.

2.7.25 *-resize3dy*

Arguments: height[%]>0, _interpolation, _boundary, _ax, _ay, _az, _ac

Resize selected images along the y-axis, preserving 3d ratio.
(eq. to '-r3dy').

'interpolation' can be { -1=none (memory content) | 0=none
| 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic
| 6=lanczos }.

'boundary' has different meanings, according to the chosen
'interpolation' mode : . When 'interpolation=={ -1 | 1 | 2
| 4 }', 'boundary' is meaningless. . When 'interpolation==0',
'boundary' can be { 0=dirichlet | 1=neumann | 2=periodic }.
. When 'interpolation=={ 3 | 5 | 6 }', 'boundary' 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=0' and 'ax=ay=az=ac=0'.

2.7.26 *-resize3dz*

Arguments: depth[%]>0, _interpolation, _boundary, _ax, _ay, _az, _ac

Resize selected images along the z-axis, preserving 3d ratio.
(eq. to '-r3dz').

'interpolation' can be { -1=none (memory content) | 0=none
| 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic
| 6=lanczos }.

'boundary' has different meanings, according to the chosen
'interpolation' mode : . When 'interpolation=={ -1 | 1 | 2
| 4 }', 'boundary' is meaningless. . When 'interpolation==0',
'boundary' can be { 0=dirichlet | 1=neumann | 2=periodic }.
. When 'interpolation=={ 3 | 5 | 6 }', 'boundary' 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=0' and 'ax=ay=az=ac=0'.

2.7.27 -rotate (+)

Arguments: `angle, _interpolation, _boundary, _center_x[_], _center_y[_], _zoom`

Rotate selected images with specified angle (in deg.).
 'interpolation' can be { 0=none | 1=linear | 2=bicubic }.
 'boundary' can be { 0=dirichlet | 1=neumann | 2=periodic }.
 When rotation center is specified, the size of the image is preserved.

Default values: 'boundary=0', 'interpolation=1', 'cx=cy=(undefined)' and 'zoom=1'.



Example 238 : `image.jpg --rotate -25,1,2,50%,50%,0.6 -rotate[0] 25`

2.7.28 -rotate_tileable

Arguments: `angle, _max_size_factor>=0`

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'.

2.7.29 -rows (+)

Arguments: { [image0] | y0[%] },-{ [image1] | y1[%] }

Keep only specified rows of selected images.
Dirichlet boundary is used when specified rows are out of range.



Example 239 : `image.jpg -rows -25%,50%`

2.7.30 -scale2x

Resize selected images using the Scale2x algorithm.



Example 240 : `image.jpg -threshold 50% -resize 50%,50% --scale2x`

2.7.31 *-scale3x*

Resize selected images using the Scale3x algorithm.



Example 241 : `image.jpg -threshold 50% -resize 33%,33% --scale3x`

2.7.32 *-seamcarve*

Arguments: `_width[%]>=0, _height[%]>=0, _is_priority_channel={ 0 | 1 }, _is_antialiasing={ 0 | 1 }, _maximum_seams[%]>=0`

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 242 : `image.jpg --seamcarve 60%`

2.7.33 *-shift (+)*

Arguments: `vx[%], -vy[%], -vz[%], -vc[%], -boundary`

Shift selected images by specified displacement vector.

'boundary' can be { 0=dirichlet | 1=neumann | 2=periodic }.

Default value: `'boundary=0'`.



Example 243 : `image.jpg --shift[0] 50%,50%,0,0,0 --shift[0] 50%,50%,0,0,1
--shift[0] 50%,50%,0,0,2`

2.7.34 *-shrink_x*

Arguments: `size_x` ≥ 0

Shrink selected images along the x-axis.



Example 244 : `image.jpg -shrink_x 30`

2.7.35 *-shrink_xy*

Arguments: `size` ≥ 0

Shrink selected images along the xy-axes.



Example 245 : `image.jpg -shrink_xy 30`

2.7.36 *-shrink_xyz*

Arguments: `size>=0`

Shrink selected images along the xyz-axes.

2.7.37 *-shrink_y*

Arguments: `size_y>=0`

Shrink selected images along the y-axis.



Example 246 : `image.jpg -shrink_y 30`

2.7.38 *-shrink_z*

Arguments: `size_z>=0`

Shrink selected images along the z-axis.

2.7.39 *-slices (+)*

Arguments: `{ [image0] | z0[%] },-{ [image1] | z1[%] }`

Keep only specified slices of selected images.
Dirichlet boundary is used when specified slices are out of range.

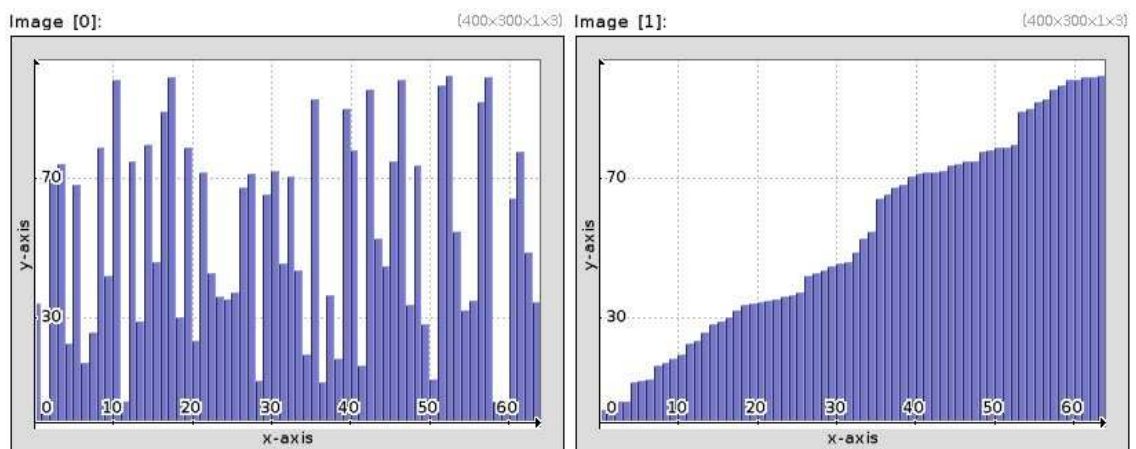
2.7.40 -sort (+)

Arguments: `_ordering={ + | - },_axis={ x | y | z | c }`

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 247 : `64 -rand 0,100 --sort -display_graph 400,300,3`

2.7.41 -split (+)

Arguments: `{ x | y | z | c }..{ x | y | z | c }`
 `},_split_mode |`
 `keep_splitting_values={ + | - },-{ x | y | z | c }..{ x`
 `| y | z | c },value1,_value2,... |`
 `(no args)`

Split selected images along specified axes, or regarding to a sequence of scalar values (optionally along specified axes too).

(eq. to '-s').

'split_mode' can be { 0=split according to constant values
 | >0=split in N parts | <0=split in parts of size -N }.

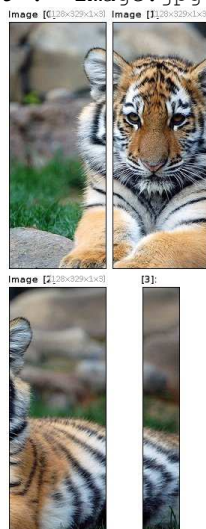
Default value: 'split_mode=-1'.



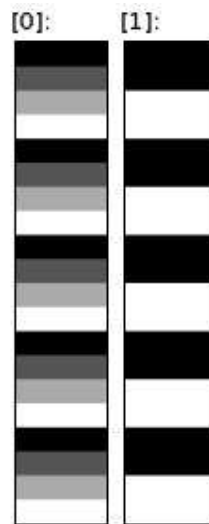
Example 248 : `image.jpg -split c`



Example 249 : `image.jpg -split y,3`



Example 250 : `image.jpg -split x,-128`



Example 251 : `1,20,1,1,"1,2,3,4" --split -,2,3 -append[1--1] y`



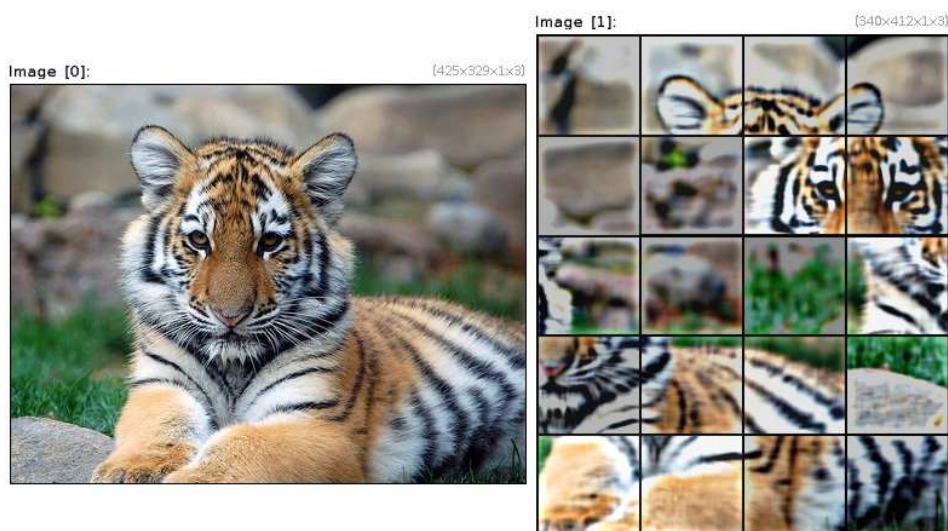
Example 252 : `(1,2,2,3,3,3,4,4,4,4) --split x,0 -append[1--1] y`

2.7.42 *-split_tiles*

Arguments: `M!=0, N!=0, _is_homogeneous={ 0 | 1 }`

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'`.



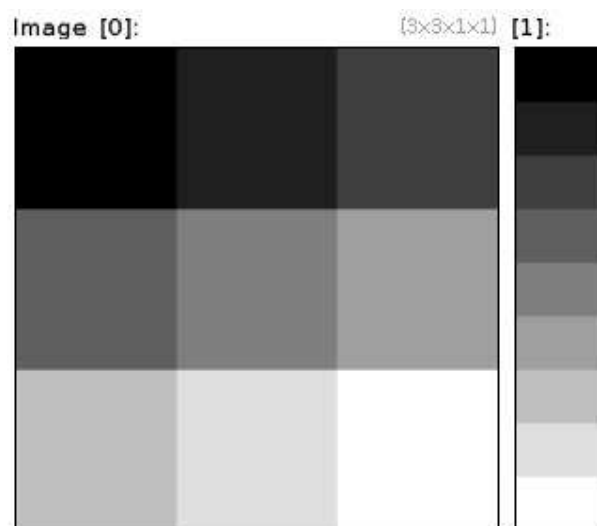
Example 253 : `image.jpg --local -split-tiles 5,4 -blur 3,0 -sharpen 700
-append-tiles 4,5 -endlocal`

2.7.43 -unroll (+)

Arguments: `-axis={ x | y | z | c }`

Unroll selected images along specified axis.
(eq. to '-y').

Default value: 'axis=y'.



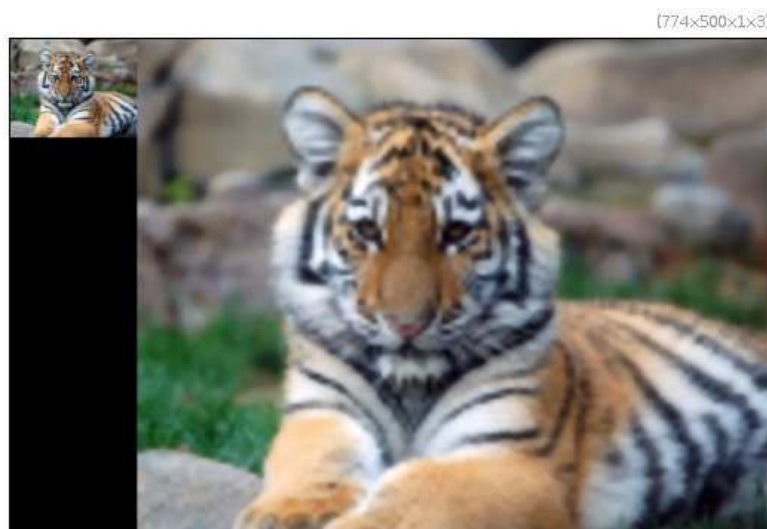
Example 254 : `(1,2,3;4,5,6;7,8,9) --unroll y`

2.7.44 *-upscale smart*

Arguments: `width[%], height[%], _depth, _smoothness>=0, _anisotropy=[0,1], -sharpening>=0`

Upscale selected images with an edge-preserving algorithm.

Default values: `'height=100%', 'depth=100%', 'smoothness=2', 'anisotropy=0.4' and 'sharpening=10'.`



Example 255 : `image.jpg -resize2dy 100 --upscale-smart 500%,500% -append x`

2.7.45 *-warp (+)*

Arguments: `[warping-field], _mode, _interpolation, _boundary, _nb_frames>0`

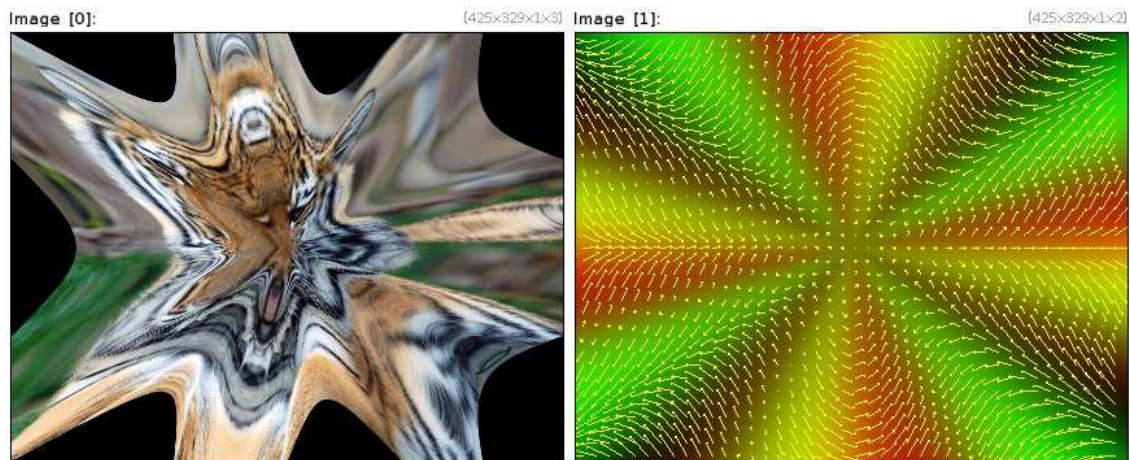
Warp selected image with specified displacement field.

'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' can be { 0=dirichlet | 1=neumann | 2=periodic }.

Default values: `'mode=0', 'interpolation=1', 'boundary=1' and 'nb_frames=1'.`



Example 256 : 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],10,0.2,1,1,100
```

Tutorial page:

http://gmic.eu/tutorial/_warp.shtml

2.8 Filtering

2.8.1 *-bandpass*

Arguments: `_min_freq[%],_max_freq[%]`

Apply bandpass filter to selected images.

Default values: `'min_freq=0'` and `'max_freq=20%'`.



Example 257 : `image.jpg -bandpass 1%,3%`

Tutorial page:

http://gmic.eu/tutorial/_bandpass.shtml

2.8.2 *-bilateral (+)*

Arguments: `[guide],std_variation_s>0[%],std_variation_r[%]>0,_sampling_s>=0,_sampling_r>=0` | `std_variation_s>0[%],std_variation_r[%]>0,_sampling_s>=0,_sampling_r>=0`

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 258 : `image.jpg [0] -repeat 5 -bilateral[-1] 10,10 -done`

2.8.3 *-blur* (+)

Arguments: `std_deviation>=0[%],_boundary,_kernel` | `axes,std_deviation>=0[%],_boundary,_kernel`

Blur selected images by a quasi-gaussian or gaussian filter (recursive implementation).
(eq. to `'-b'`).

`'boundary'` can be { 0=dirichlet | 1=neumann } and `'kernel'` can be { 0=quasi-gaussian (faster) | 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=1'` and `'kernel=0'`.



Example 259 : `image.jpg --blur 5,0 --blur[0] 5,1`



Example 260 : `image.jpg --blur y,10%`

Tutorial page:

http://gmic.eu/tutorial/_blur.shtml

2.8.4 *-blur_angular*

Arguments: `amplitude[%],_center_x[%],_center_y[%]`

Apply angular blur on selected images.

Default values: `'center_x=center_y=50%'`.



Example 261 : `image.jpg --blur_angular 2%`

Tutorial page:

http://gmic.eu/tutorial/_blur_angular.shtml

2.8.5 *-blur_linear*

Arguments: `amplitude1[%],_amplitude2[%],_angle,_boundary={`
 `0=dirichlet | 1=neumann }`

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

Default values: `'amplitude2=0', 'angle=0' and 'boundary=1'.`



Example 262 : `image.jpg --blur_linear 10,0,45`

Tutorial page:

http://gmics.eu/tutorial/_blur_linear.shtml

2.8.6 *-blur_radial*

Arguments: `amplitude[%],_center_x[%],_center_y[%]`

Apply radial blur on selected images.

Default values: `'center_x=center_y=50%'.`



Example 263 : `image.jpg --blur_radial 2%`

Tutorial page:

http://gmic.eu/tutorial/_blur_radial.shtml

2.8.7 *-blur_selective*

Arguments: `sigma>=0, _edges>0, _nb_scales>0`

Blur selected images using selective gaussian scales.

Default values: `'sigma=5', 'edges=0.5' and 'nb_scales=5'.`



Example 264 : `image.jpg -noise 20 -cut 0,255 --local[-1] -repeat 4
-blur_selective , -done -endlocal`

Tutorial page:

http://gmic.eu/tutorial/_blur_selective.shtml

2.8.8 *-blur_x*

Arguments: `amplitude[%]>=0, _boundary={ 0=dirichlet | 1=neumann }`

Blur selected images along the x-axis.

Default value: `'boundary=1'`.



Example 265 : `image.jpg --blur_x 6`

Tutorial page:

http://gmics.eu/tutorial/_blur_x.shtml

2.8.9 *-blur_xy*

Arguments: `amplitude_x[%], amplitude_y[%], _boundary={ 0=dirichlet | 1=neumann }`

Blur selected images along the X and Y axes.

Default value: `'boundary=1'`.



Example 266 : `image.jpg --blur-xy 6`

Tutorial page:

http://gmic.eu/tutorial/_blur-xy.shtml

2.8.10 *-blur-xyz*

Arguments: `amplitude-x[%],amplitude-y[%],amplitude-z, _boundary={
0=dirichlet | 1=neumann }`

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

Default value: `'boundary=1'`.

Tutorial page:

http://gmic.eu/tutorial/_blur-xyz.shtml

2.8.11 *-blur-y*

Arguments: `amplitude[%]>=0, _boundary={ 0=dirichlet | 1=neumann }`

Blur selected images along the y-axis.

Default value: `'boundary=1'`.



Example 267 : `image.jpg --blur_y 6`

Tutorial page:

http://gmic.eu/tutorial/_blur_y.shtml

2.8.12 *-blur_z*

Arguments: `amplitude[%]>=0, boundary={ 0=dirichlet | 1=neumann }`

Blur selected images along the z-axis.

Default value: `'boundary=1'`.

Tutorial page:

http://gmic.eu/tutorial/_blur_z.shtml

2.8.13 *-bokeh*

Arguments: `_amplitude>=0, _smoothness>=0, 0<=_density<=100, _bokeh_size>0, -0<=_bokeh_outline_size<=100, _bokeh_outline_amplitude>=0, _bokeh_smoothness>=0`

Create a Bokeh effect from selected images.

Default values: `'amplitude=200', 'smoothness=2', 'density=0.2', 'bokeh_size=24', 'bokeh_outline_size=10', 'bokeh_outline_amplitude=1'` and `'bokeh_smoothness=0.1'`.



Example 268 : `image.jpg --bokeh ,`

2.8.14 *-boxfilter* (+)

Arguments: `size>=0[%],_order,_boundary` |
 `axes,size>=0[%],_order,_boundary`

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

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

'boundary' can be { 0=dirichlet | 1=neumann } and 'kernel' can be { 0=quasi-gaussian (faster) | 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: 'order=0' and 'boundary=1'.



Example 269 : `image.jpg --boxfilter 5%`



Example 270 : `image.jpg --boxfilter y,3,1`

2.8.15 *-compose freq*

Compose selected low and high frequency parts into new images.



Example 271 : `image.jpg -split-freq 2% -mirror[-1] x -compose-freq`

2.8.16 *-convolve* (+)

Arguments: `[mask],_boundary,_is.normalized={ 0 | 1 }`

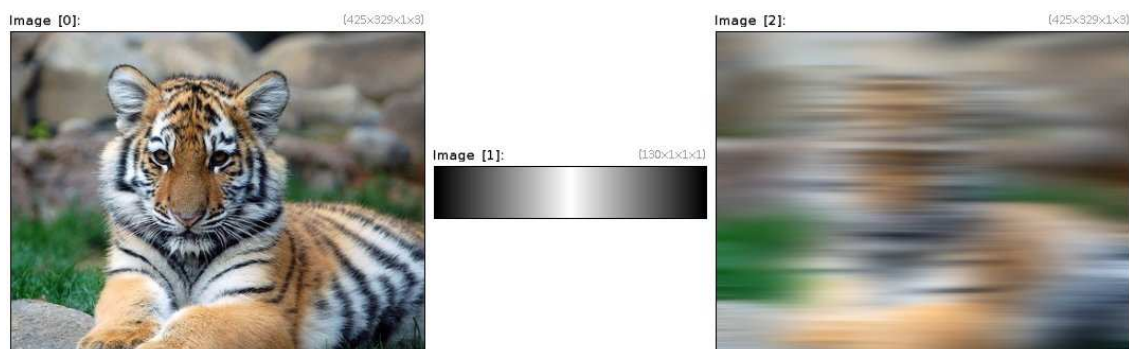
Convolve selected images by specified mask.

'boundary' can be { 0=dirichlet | 1=neumann }.

Default values: 'boundary=1' and 'is.normalized=0'.



Example 272 : `image.jpg (0,1,0;1,-4,1;0,1,0) -convolve[-2] [-1] -keep[-2]`



Example 273 : `image.jpg (0,1,0) -resize[-1] 130,1,1,1,3 --convolve[0] [1]`

Tutorial page:

http://gmics.eu/tutorial/_convolve.shtml

2.8.17 *-convolve fft*

Arguments: `[mask]`

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



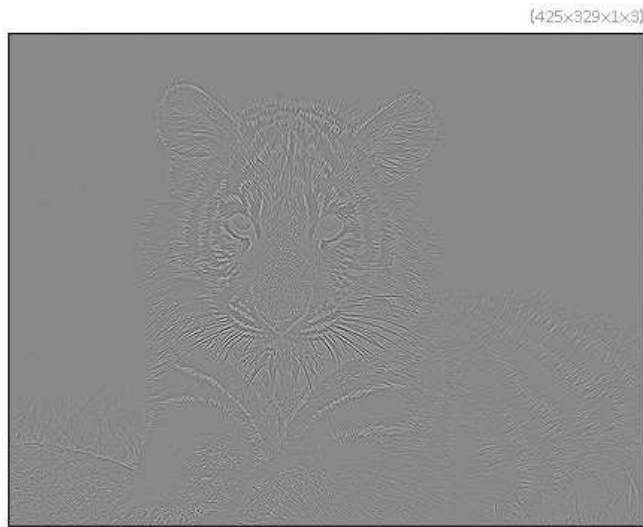
Example 274 : `image.jpg 100%,100% -gaussian[-1] 20,1,45 --convolve.fft[0] [1]`

2.8.18 *-correlate (+)*

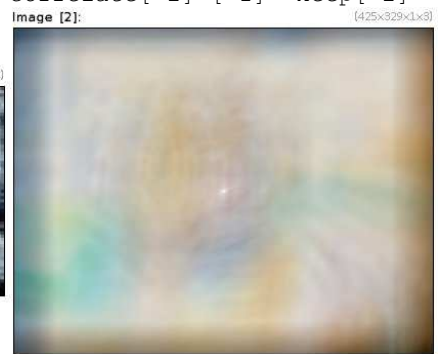
Arguments: `[mask], _boundary, _is.normalized={ 0 | 1 }`

Correlate selected images by specified mask.
'boundary' can be { 0=dirichlet | 1=neumann }.

Default values: 'boundary=1' and 'is.normalized=0'.



Example 275 : `image.jpg (0,1,0;1,-4,1;0,1,0) -correlate[-2] [-1] -keep[-2]`



Example 276 : `image.jpg --crop 40%,40%,60%,60% --correlate[0] [-1],0,1`

2.8.19 *-cross_correlation*

Arguments: `[mask]`

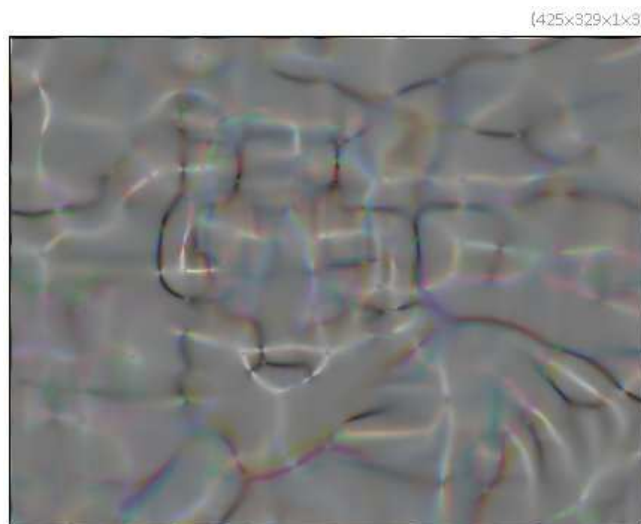
Compute cross-correlation of selected images with specified mask.



Example 277 : `image.jpg --shift -30,-20 --cross_correlation[0] [1]`

2.8.20 -curvature

Compute isophote curvatures on selected images.



Example 278 : `image.jpg -blur 10 -curvature`

2.8.21 -dct

Arguments: `-{ x | y | z }..{ x | y | z }` |
(no arg)

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

Default values: (no arg)



Example 279 : `image.jpg --dct --idct[-1] -abs[-2] -+[-2] 1 -log[-2]`

Tutorial page:

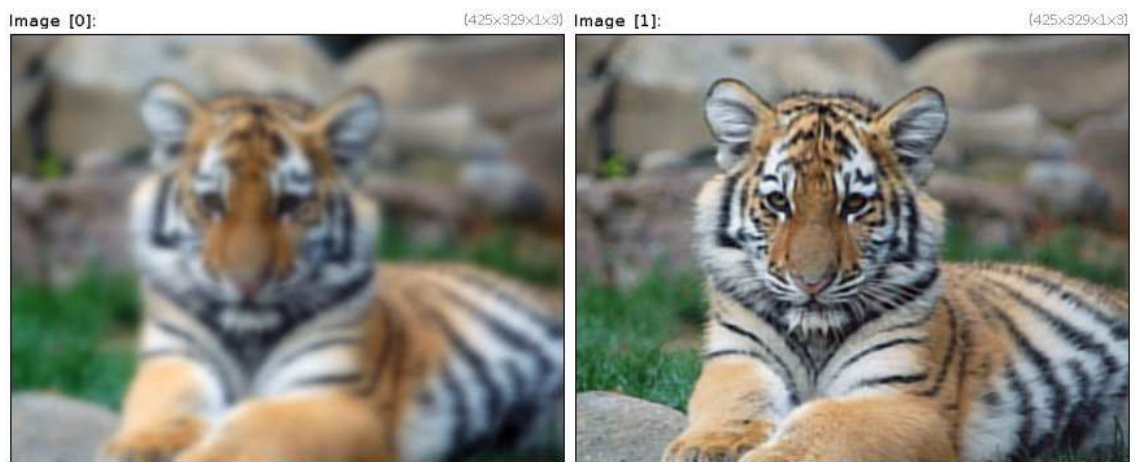
http://gmic.eu/tutorial/_dct-and-idct.shtml

2.8.22 *-deblur*

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

Deblur image using a regularized Jansson-Van Cittert algorithm.

Default values: `'nb_iter=10', 'dt=20', 'regul=0.7' and 'regul_type=1'.`



Example 280 : `image.jpg -blur 3 --deblur 3,40,20,0.01`

2.8.23 *-deblur_goldmeinel*

Arguments: `sigma>=0, _nb_iter>=0, _acceleration>=0, _kernel_type={0=quasi-gaussian (faster) | 1=gaussian }.`

Deblur selected images using Gold-Meinel algorithm

Default values: `'nb_iter=8', 'acceleration=1' and 'kernel_type=1'.`



Example 281 : `image.jpg --blur 1 --deblur_goldmeinel[-1] 1`

2.8.24 *-deblur richardsonlucy*

Arguments: `sigma>=0, nb_iter>=0, _kernel_type={ 0=quasi-gaussian (faster) | 1=gaussian }`.

Deblur selected images using Richardson-Lucy algorithm.

Default values: `'nb_iter=50'` and `'kernel_type=1'`.



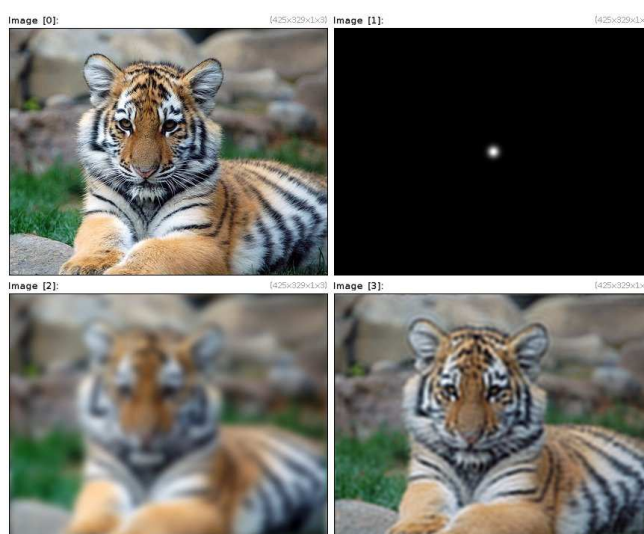
Example 282 : `image.jpg --blur 1 --deblur_richardsonlucy[-1] 1`

2.8.25 *-deconvolve fft*

Arguments: `[kernel], _regularization>=0`

Deconvolve selected images by specified mask in the fourier space.

Default value: `'regularization>=0'`.



Example 283 : `image.jpg --gaussian 5 --convolve-fft[0] [1]
--deconvolve-fft[-1] [1]`

2.8.26 *-deinterlace*

Arguments: `_method={ 0 | 1 }`

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

Default value: `'method=0'`.



Example 284 : `image.jpg --rotate 3,1,1,50%,50% -resize 100%,50% -resize
100%,200%,1,3,4 -shift[-1] 0,1 -add --deinterlace 1`

2.8.27 *-denoise (+)*

Arguments: `std_variation_s>=0, _std_variation_p>=0, _patch_size>=0, _looku-
p_size>=0, _smoothness, _fast_approx={ 0 | 1 }`

Denoise selected images by non-local patch averaging.

Default values: `'std_variation_p=10', 'patch_size=5', 'lookup_size=6'`
and `'smoothness=1'`.



Example 285 : `image.jpg --denoise 5,5,8`

2.8.28 *-denoise haar*

Arguments: `_threshold>=0, _nb_scales>=0, _cycle_spinning>0`

Denoise selected image 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 286 : `image.jpg -noise 20 -c 0,255 --denoise.haar[-1] 0.8`

2.8.29 -deriche (+)

Arguments: `std_variation>=0[%],order={ 0 | 1 | 2 },axis={ x
| y | z | c },_boundary`

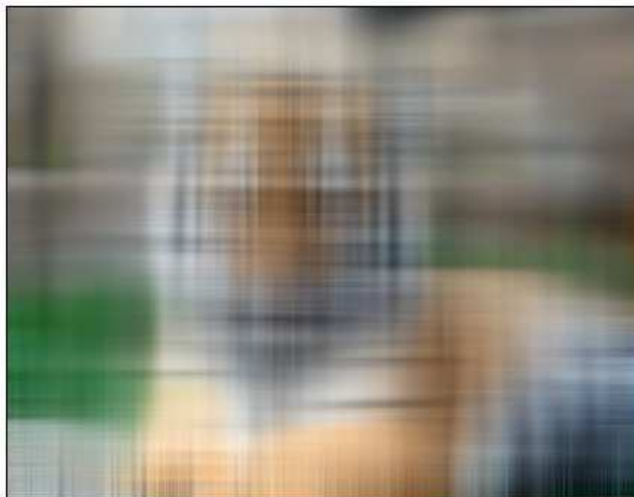
Apply Deriche recursive filter on selected images, along specified axis and with specified standard deviation, order and boundary conditions.

'boundary' can be { 0=dirichlet | 1=neumann }.

Default value: 'boundary=1'.



Example 287 : `image.jpg --deriche 3,1,x`
(425x329x1x3)



Example 288 : `image.jpg --deriche 30,0,x -deriche[-2] 30,0,y -add`

Tutorial page:

http://gmic.eu/tutorial/_deriche.shtml

2.8.30 *-dilate (+)*

Arguments: `size>=0` |
 `size_x>=0,size_y>=0,size_z>=0` |
 `[mask],_boundary,_is.normalized={ 0 | 1 }`

Dilate selected images by a rectangular or the specified structuring element.

'boundary' can be { 0=dirichlet | 1=neumann }.

Default values: 'size_z=1', 'boundary=1' and 'is.normalized=0'.



Example 289 : `image.jpg --dilate 10`

2.8.31 *-dilate_circ*

Arguments: `_size>=0,_boundary,_is.normalized={ 0 | 1 }`

Apply circular dilation of selected image by specified size.

Default values: 'boundary=1' and 'is.normalized=0'.



Example 290 : `image.jpg --dilate_circ 7`

2.8.32 *-dilate_oct*

Arguments: `_size>=0, _boundary, _is_normalized={ 0 | 1 }`

Apply octagonal dilation of selected image by specified size.

Default values: `'boundary=1'` and `'is_normalized=0'`.



Example 291 : `image.jpg --dilate_oct 7`

2.8.33 *-dilate_threshold*

Arguments: `size_x>=1, size_y>=1, size_z>=1, _threshold>=0, _boundary`

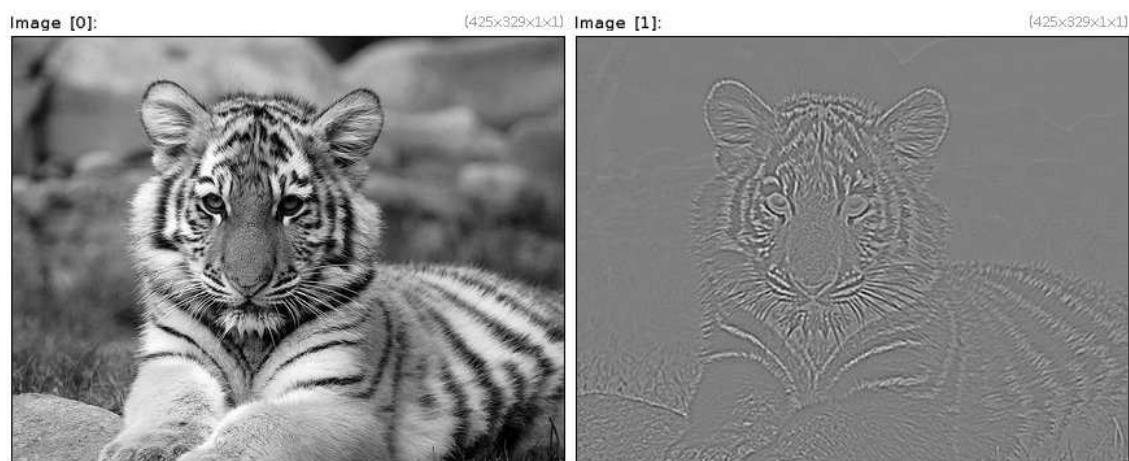
Dilate selected images in the (X,Y,Z,I) space.

'boundary' can be { 0=dirichlet | 1=neumann }.

Default values: 'size_y=size_x', 'size_z=1', 'threshold=255' and 'boundary=1'.

2.8.34 -divergence

Compute divergence of selected vector fields.



Example 292 : `image.jpg -luminance --gradient -append[-2,-1] c
-divergence[-1]`

2.8.35 -dog

Arguments: `_sigma1>=0 [%], _sigma2>=0 [%]`

Compute difference of gaussian on selected images.

Default values: 'sigma1=2%' and 'sigma2=3%'.



Example 293 : `image.jpg --dog 2,3`

2.8.36 *-diffusientensors*

Arguments: `_sharpness>=0,0<=_anisotropy<=1,_alpha[%],_sigma[%],is_sqrt=-{ 0 | 1 }`

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'.`



Example 294 : `image.jpg -diffusientensors 0.8 -abs -pow 0.2`

Tutorial page:

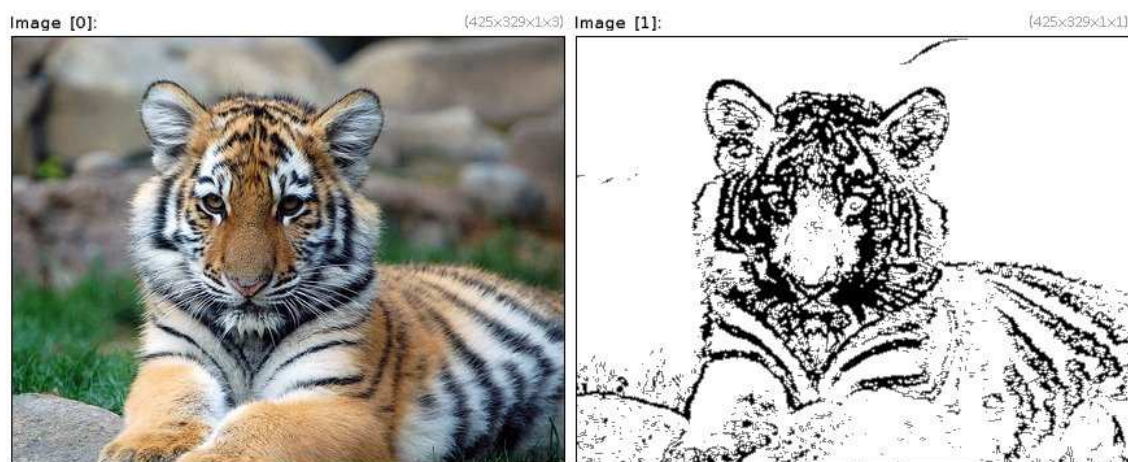
http://gmics.eu/tutorial/_diffusientensors.shtml

2.8.37 *-edges*

Arguments: `_threshold[%]>=0`

Estimate contours of selected images.

Default value: `'edges=15%'`



Example 295 : `image.jpg --edges 15%`

2.8.38 *-erode (+)*

Arguments: `size>=0` |
 `size_x>=0,size_y>=0,_size_z>=0` |
 `[mask],_boundary,_is_normalized={ 0 | 1 }`

Erode selected images by a rectangular or the specified structuring element.

'boundary' can be { 0=dirichlet | 1=neumann }.

Default values: 'size_z=1', 'boundary=1' and 'is_normalized=0'.



Example 296 : `image.jpg --erode 10`

2.8.39 *-erode_circ*

Arguments: `_size>=0, _boundary, _is.normalized={ 0 | 1 }`

Apply circular erosion of selected images by specified size.

Default values: `'boundary=1'` and `'is.normalized=0'`.



Example 297 : `image.jpg --erode_circ 7`

2.8.40 *-erode_oct*

Arguments: `_size>=0, _boundary, _is.normalized={ 0 | 1 }`

Apply octagonal erosion of selected images by specified size.

Default values: `'boundary=1'` and `'is.normalized=0'`.



Example 298 : `image.jpg --erode-oct 7`

2.8.41 *-erode threshold*

Arguments: `size_x>=1,size_y>=1,size_z>=1,threshold>=0,boundary`

Erode selected images in the (X,Y,Z,I) space.

'boundary' can be { 0=dirichlet | 1=neumann }.

Default values: 'size_y=size_x', 'size_z=1', 'threshold=255' and 'boundary=1'.

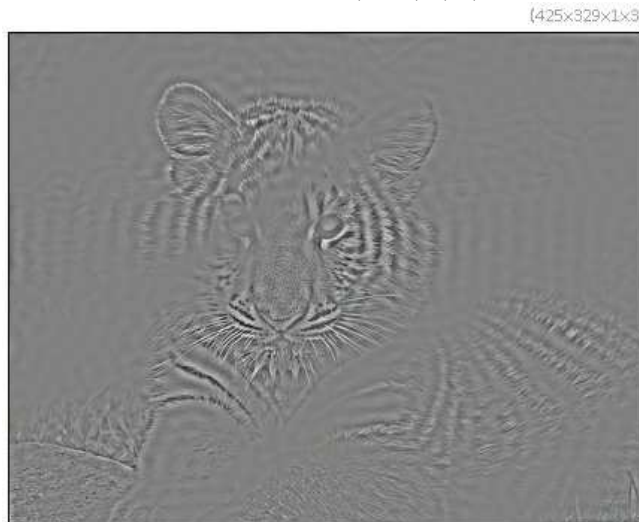
2.8.42 *-fft (+)*

Arguments: `-{ x | y | z }..{ x | y | z }`

Compute the direct fourier transform (real and imaginary parts) of selected images, optionally along the specified axes only.



Example 299 : `image.jpg -luminance --fft -append[-2,-1] c -norm[-1] -log[-1] -shift[-1] 50%,50%,0,0,2`



Example 300 : `image.jpg -fft -shift 50%,50%,0,0,2 -ellipse 50%,50%,30,30,0,1,0 -shift -50%,-50%,0,0,2 -ifft -remove[-1]`

Tutorial page:

http://gmic.eu/tutorial/_fft.shtml

2.8.43 *-gradient (+)*

Arguments: `{ x | y | z }..{ x | y | z },_scheme | (no arg)`

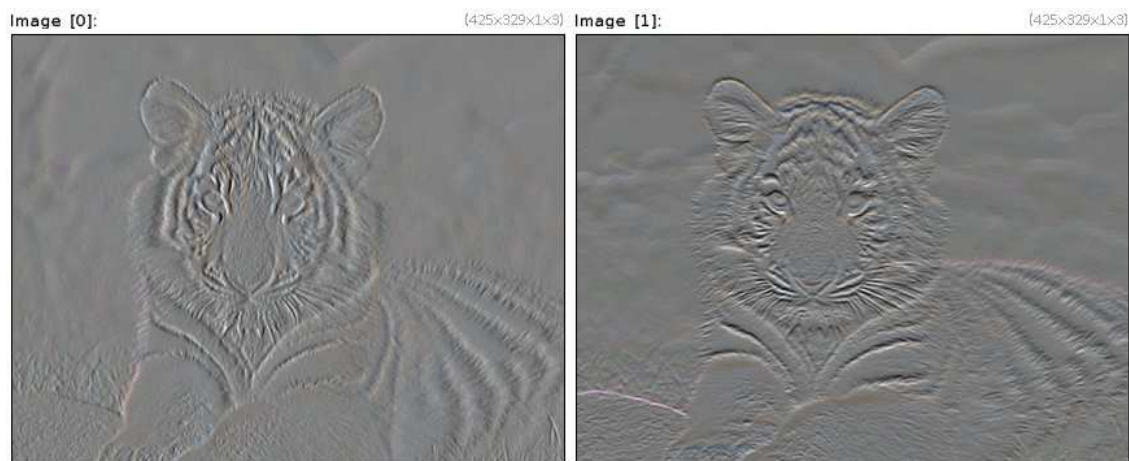
Compute the gradient components (first derivatives) of selected images.

(eq. to `'-g'`).

`'scheme'` can be { `-1=backward` | `0=centered` | `1=forward`
 | `2=sobel` | `3=rotation-invariant` (default) | `4=deriche`
 | `5=vanvliet` }.

(no arg) compute all significant 2d/3d components.

Default value: `'scheme=3'`.



Example 301 : `image.jpg -gradient`

Tutorial page:

http://gmics.eu/tutorial/_gradient.shtml

2.8.44 *-gradient norm*

Compute gradient norm of selected images.



Example 302 : `image.jpg --gradient.norm -equalize[-1]`

Tutorial page:

http://gmic.eu/tutorial/_gradient_norm.shtml

2.8.45 -gradient_orientation

Arguments: `_dimension={1,2,3}`

Compute N-d gradient orientation of selected images.

Default value: `'dimension=3'`.



Example 303 : `image.jpg --gradient_orientation 2`

2.8.46 -guided (+)

Arguments: `[guide],radius[%]>0,regularization>0` |
 `radius[%]>0,regularization>0`

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,"
 Pattern Analysis and Machine Intelligence,
 IEEE Transactions on , vol.35, no.6, pp.1397,1409, June 2013



Example 304 : `image.jpg [0] --guided[-1] 5,400`

2.8.47 *-haar*

Arguments: `scale>0`

Compute the direct haar multiscale wavelet transform of selected images.

Tutorial page:

http://gmics.eu/tutorial/_haar.shtml

2.8.48 *-heat flow*

Arguments: `_nb_iter>=0, _dt, _keep_sequence={ 0 | 1 }`

Apply iterations of the heat flow on selected images.

Default values: `'nb_iter=10', 'dt=30' and 'keep_sequence=0'.`



Example 305 : `image.jpg --heat_flow 20`

2.8.49 -hessian (+)

Arguments: { xx | xy | xz | yy | yz | zz }..{ xx | xy
 | xz | yy | yz | zz } |
 (no arg)

Compute the hessian components (second derivatives) of selected images.

(no arg) compute all significant components.



Example 306 : image.jpg -hessian

2.8.50 -idct

Arguments: -{ x | y | z }..{ x | y | z } |
 (no arg)

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

Default values: (no arg)

Tutorial page:

http://gmics.eu/tutorial/_dct-and-idct.shtml

2.8.51 -iee

Compute gradient-orthogonal-directed 2nd derivative of image(s).



Example 307 : `image.jpg -iee`

2.8.52 *-ifft (+)*

Arguments: `-{ x | y | z }..{ x | y | z }`

Compute the inverse fourier transform (real and imaginary parts) of selected images. optionally along the specified axes only.

Tutorial page:

http://gmics.eu/tutorial/_fft.shtml

2.8.53 *-ihaar*

Arguments: `scale>0`

Compute the inverse haar multiscale wavelet transform of selected images.

2.8.54 *-inn*

Compute gradient-directed 2nd derivative of image(s).



Example 308 : `image.jpg -inn`

2.8.55 *-inpaint (+)*

Arguments: `[mask] |`
`[mask],0,_fast_method |`
`[mask],_patch_size>=1,_lookup_size>=1,_lookup_factor>=0,_loo-`
`kup_increment!=0,_blend_size>=0,0<=_blend_threshold<=1,_blen-`
`d_decay>=0,_blend_scales>=1,_is_blend_outer={ 0 | 1 }`

Inpaint selected images by specified mask.

If no patch size (or 0) is specified, inpainting is done using a fast average or median algorithm.

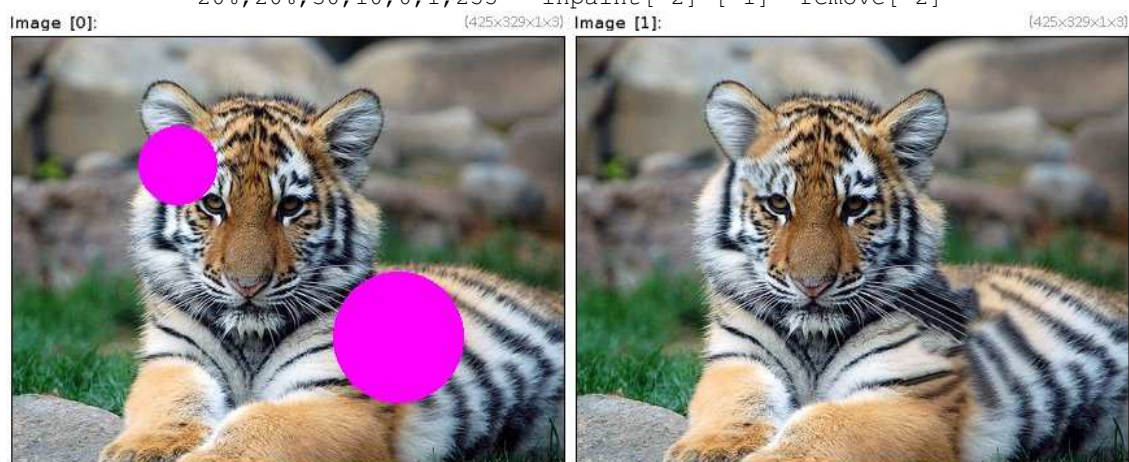
Otherwise, it used a patch-based reconstruction method, that can be very time consuming.

'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'.



Example 309 : `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]`



Example 310 : `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]`

2.8.56 *-inpaint_flow*

Arguments: `_nb_iter1>=0, _nb_iter2>=0, _dt>=0, _alpha, _sigma`

Apply iteration of the inpainting flow on selected images.

Default values: `'_nb_iter1=4', '_nb_iter2=15', '_dt=15', '_alpha=1' and '_sigma=3'.`



Example 311 : `image.jpg 100%,100% -ellipse[-1] 30%,30%,40,30,0,1,255 -reverse -inpaint_flow ,`

2.8.57 -inpaint_holes

Arguments: `maximal_area[%]>=0, tolerance>=0, is.high.connectivity={0 | 1 }`

Inpaint all connected regions having an area less than specified value.

Default values: `'maximal_area=4', 'tolerance=0' and 'is.high.connectivity=0'.`



Example 312 : `image.jpg -noise 5%,2 --inpaint_holes 8,40`

2.8.58 *-kuwahara*

Arguments: size>0

Apply Kuwahara filter of specified size on selected images.



Example 313 : image.jpg --kuwahara 5

2.8.59 *-laplacian*

Compute Laplacian of selected images.



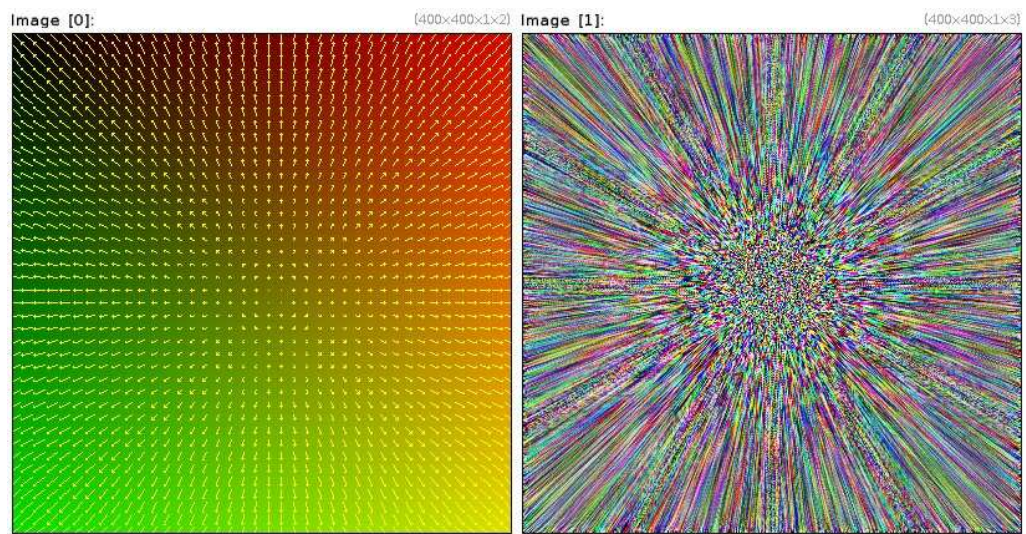
Example 314 : image.jpg -laplacian

2.8.60 *-lic*

Arguments: _amplitude>0, _channels>0

Render LIC representation of selected vector fields.

Default values: 'amplitude=30' and 'channels=1'.



Example 315 : `400,400,1,2,'if(c==0,x-w/2,y-h/2)' --lic 200,3 -quiver[-2]
[-2],10,-13,1,1,255`

2.8.61 *-map_tones*

Arguments: `_threshold>=0, _gamma>=0, _smoothness>=0, nb_iter>=0`

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 316 : `image.jpg --map-tones ,`

2.8.62 *-map-tones-fast*

Arguments: `_radius[%]>=0, _power>=0`

Apply fast tone mapping operator on selected images.

Default values: `'radius=3%'` and `'power=0.3'`.



Example 317 : `image.jpg --map-tones-fast ,`

2.8.63 *-meancurvature-flow*

Arguments: `_nb_iter>=0, _dt, _sequence_flag={ 0 | 1 }`

Apply iterations of the mean curvature flow on selected images.

Default values: 'nb_iter=10', 'dt=30' and 'keep_sequence=0'.



Example 318 : `image.jpg --meancurvature_flow 20`

2.8.64 *-median (+)*

Arguments: `size>=0, _threshold>0`

Apply (opt. thresholded) median filter on selected images with structuring element size x size.



Example 319 : `image.jpg --median 5`

2.8.65 *-nlmeans*

Arguments: `_patch_radius>0, _spatial_bandwidth>0, _tonal_bandwidth>0, _patch_measure_command`

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 ($\exp(-d^2/_\text{tonal_bandwidth}^2)$) with d the euclidian distance between image patches.

Default values: 'patch_radius=4', 'spatial_bandwidth=4', 'tonal_bandwidth=10' and 'patch_measure_command=-norm'.



Example 320 : `image.jpg --noise 10 -nlmeans[-1] 4,4,{0.6*${-std_noise}}`

2.8.66 -nlmeans_core

Arguments: `_reference_image, _scaling_map, patch_radius>0, _spatial_bandwidth>0`

Apply non local means denoising using a image for weight and a map for scaling

2.8.67 -normalize_local

Arguments: `_amplitude>=0, _radius>0, _n_smooth>=0[%], _a_smooth>=0[%], _is_cut={ 0 | 1 }, _min=0, _max=255`

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 321 : `image.jpg --normalize-local 8,10`

2.8.68 *-normalized_cross_correlation*

Arguments: `[mask]`

Compute normalized cross-correlation of selected images with specified mask.



Example 322 : `image.jpg --shift -30,-20 --normalizedcross.correlation[0] [1]`

2.8.69 *-peronamalik_flow*

Arguments: `K_factor>0, nb_iter>=0, dt, keep_sequence={ 0 | 1 }`

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 323 : `image.jpg --heat_flow 20`

2.8.70 *-phase_correlation*

Arguments: `[destination]`

Estimate translation vector between selected source images and specified destination.



Example 324 : `image.jpg --shift -30,-20 --phase_correlation[0] [1] -unroll[-1]`
y

2.8.71 *-pde_flow*

Arguments: `_nb_iter>=0, _dt, _velocity_command, _keep_sequence={ 0 | 1 }`

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 325 : `image.jpg --pde_flow 20`

2.8.72 *-periodize_poisson*

Periodize selected images using a Poisson solver in Fourier space.



Example 326 : `image.jpg --periodize_poisson -array 2,2,2`

2.8.73 *-red_eye*

Arguments: `0<=_threshold<=100, _smoothness>=0, 0<=attenuation<=1`

Attenuate red-eye effect in selected images.

Default values: 'threshold=75', 'smoothness=3.5' and 'attenuation=0.1'.



Example 327 : `image.jpg --red-eye ,`

2.8.74 *-remove_hotpixels*

Arguments: `_mask_size>0, _threshold[%]>0`

Remove hot pixels in selected images.

Default values: `'mask_size=3'` and `'threshold=10%'`.



Example 328 : `image.jpg -noise 10,2 --remove_hotpixels ,`

2.8.75 *-remove_pixels*

Arguments: `number_of_pixels[%]>=0`

Remove specified number of pixels (i.e. set them to 0) from the set of non-zero pixels in selected images.



Example 329 : `image.jpg --remove_pixels 50%`

2.8.76 *-sharpen (+)*

Arguments: `amplitude>=0` |
`amplitude>=0,edge>=0,_alpha,_sigma`

Sharpen selected images by inverse diffusion or shock filters methods.

'edge' must be specified to enable shock-filter method.

Default values: 'alpha=0' and 'sigma=0'.



Example 330 : `image.jpg --sharpen 300`



Example 331 : `image.jpg -blur 5 --sharpen[-1] 300,1`

2.8.77 *-smooth (+)*

Arguments: `amplitude>=0, _sharpness>=0, _anisotropy, _alpha, _sigma, _dl>0, --da>0, _precision>0, interpolation, _fast_approx={ 0 | 1 } | nb_iterations>=0, _sharpness>=0, _anisotropy, _alpha, _sigma, _dt>0, 0 | [tensor_field], _amplitude>=0, _dl>0, _da>0, _precision>0, interpolation, _fast_approx={ 0 | 1 } | [tensor_field], _nb_iters>=0, _dt>0, 0`

Smooth selected images anisotropically using diffusion PDE's, with specified field of diffusion tensors.

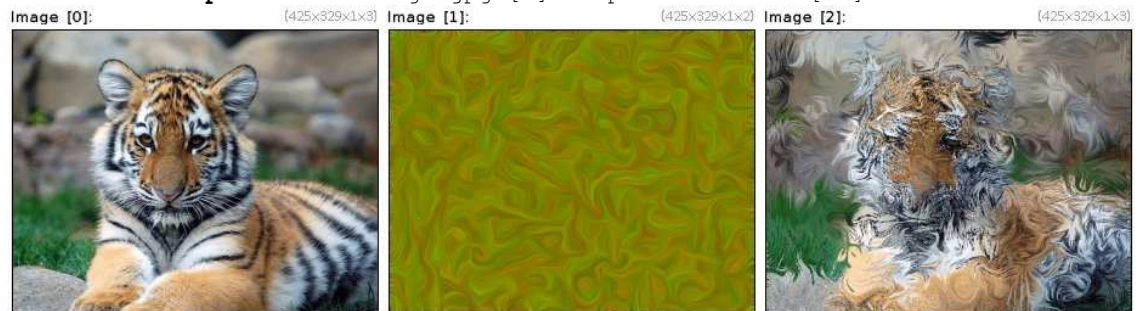
'anisotropy' must be in [0,1].

'interpolation' can be { 0=nearest | 1=linear | 2=runge-kutta }.

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'.`



Example 332 : `image.jpg [0] -repeat 3 -smooth[-1] 20 -done`



Example 333 : `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`

Tutorial page:

http://gmic.eu/tutorial/_smooth.shtml

2.8.78 *-split_freq*

Arguments: `smoothness>0[%]`

Split selected images into low and high frequency parts.



Example 334 : `image.jpg -split.freq 2%`

2.8.79 *-solidify*

Replace transparent regions of a RGBA image by morphologically interpolated color.



Example 335 : `image.jpg --luminance -ge[-1] 120 -*[-1] 255 -append c
--solidify -display.rgba`

2.8.80 *-solidify linear*

Arguments: `_sigma>=1, _dsigma>=1, 0<=_precision<=1`

Replace transparent regions of a RGBA image by linearly interpolated color.

Default values: `'sigma=1.5', 'dsigma=1' and 'precision=0.5'.`



Example 336 : `image.jpg --luminance -ge[-1] 120 -*[-1] 255 -append c
--solidify_linear , -display_rgba`

2.8.81 *-solidify_watershed*

Replace transparent regions of RGBA image by color propagation.



Example 337 : `image.jpg --luminance -ge[-1] 120 -*[-1] 255 -append c
--solidify_watershed -display_rgba`

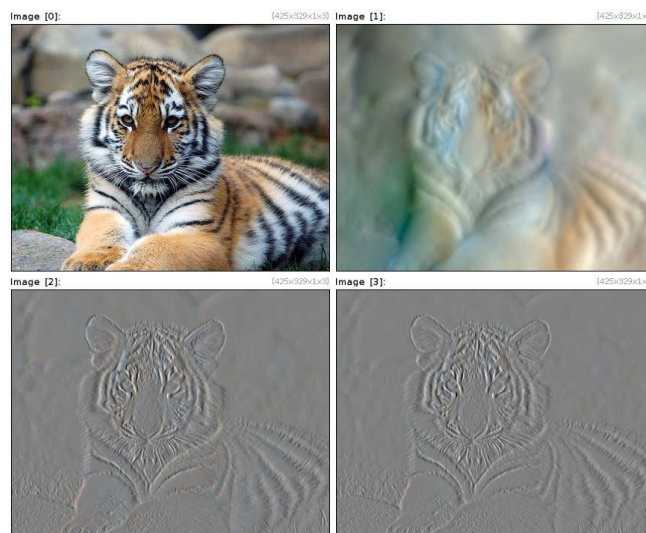
2.8.82 *-solve_poisson*

Arguments: `"laplacian_command", _nb.iterations>=0, _time_step>0, _nb.scale-
s>=0`

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'`.



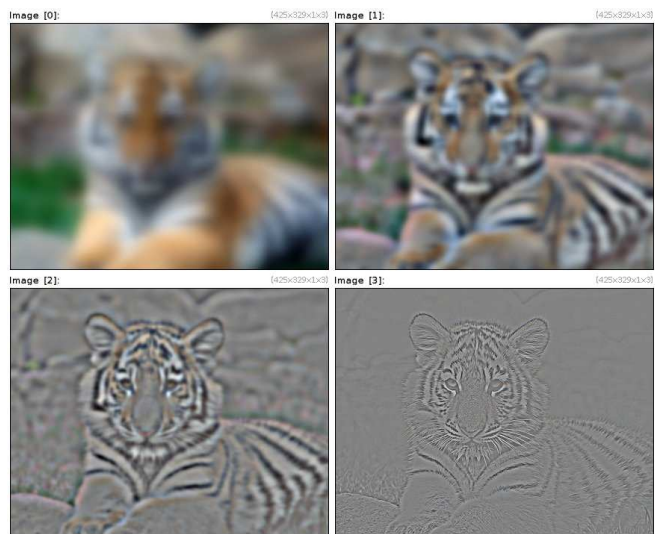
Example 338 : `image.jpg -m "foo : -gradient x" --solve_poisson foo --foo[0] --laplacian[1]`

2.8.83 *-split_details*

Arguments: `_nb_scales>0, _base_scale[%]>=0, _detail_scale[%]>=0`

Split selected images into `'nb_scales'` spatial scales (gaussian pyramids).

Default values: `'nb_scales=4'`, `'base_scale=2%'` and `'detail_scale=0.5%'`.



Example 339 : `image.jpg -split_details ,`

2.8.84 `-structuretensors (+)`

Arguments: `_scheme`

Compute the structure tensor field of selected images.
 'scheme' can be { 0=centered | 1=forward-backward1
 | 2=forward-backward2 }.

Default value: `'scheme=2'`.



Example 340 : `image.jpg -structuretensors -abs -pow 0.2`

Tutorial page:

http://gmic.eu/tutorial/_structuretensors.shtml

2.8.85 -syntexturize

Arguments: `_width[%]>0,_height[%]>0`

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/

Default values: `'width=height=100%'`.



Example 341 : `image.jpg -crop 2,282,50,328 --syntexturize 320,320`

2.8.86 -tv_flow

Arguments: `_nb_iter>=0,_dt,_sequence_flag={ 0 | 1 }`

Apply iterations of the total variation flow on selected images.

Default values: `'nb_iter=10', 'dt=30' and 'keep_sequence=0'`.



Example 342 : `image.jpg --tv_flow 40`

2.8.87 *-unsharp*

Arguments: `radius[%]>=0, _amount>=0, _threshold[%]>=0`

Apply unsharp mask on selected images.

Default values: `'amount=2'` and `'threshold=0'`.



Example 343 : `image.jpg -blur 3 --unsharp 1.5,15 -cut 0,255`

2.8.88 *-unsharp_octave*

Arguments: `_nb_scales>0, _radius[%]>=0, _amount>=0, threshold[%]>=0`

Apply octave sharpening on selected images.

Default values: 'nb_scales=4', 'radius=1', 'amount=2' and 'threshold=0'.



Example 344 : `image.jpg -blur 3 --unsharp-octave 4,5,15 -cut 0,255`

2.8.89 -vanvliet (+)

Arguments: `std_variation>=0[%],order={ 0 | 1 | 2 | 3 },axis={ x | y | z | c },_boundary`

Apply Vanvliet recursive filter on selected images, along specified axis and with specified standard deviation, order and boundary conditions.

'boundary' can be { 0=dirichlet | 1=neumann }.

Default value: 'boundary=1'.



Example 345 : `image.jpg --vanvliet 3,1,x`



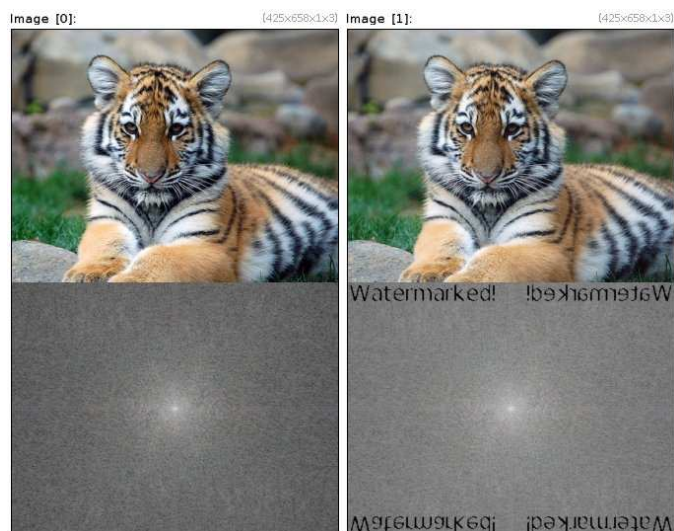
Example 346 : `image.jpg --vanvliet 30,0,x -vanvliet[-2] 30,0,y -add`

2.8.90 *-watermark_fourier*

Arguments: `text,_size>0`

Add a textual watermark in the frequency domain of selected images.

Default value: `'size=33'`.



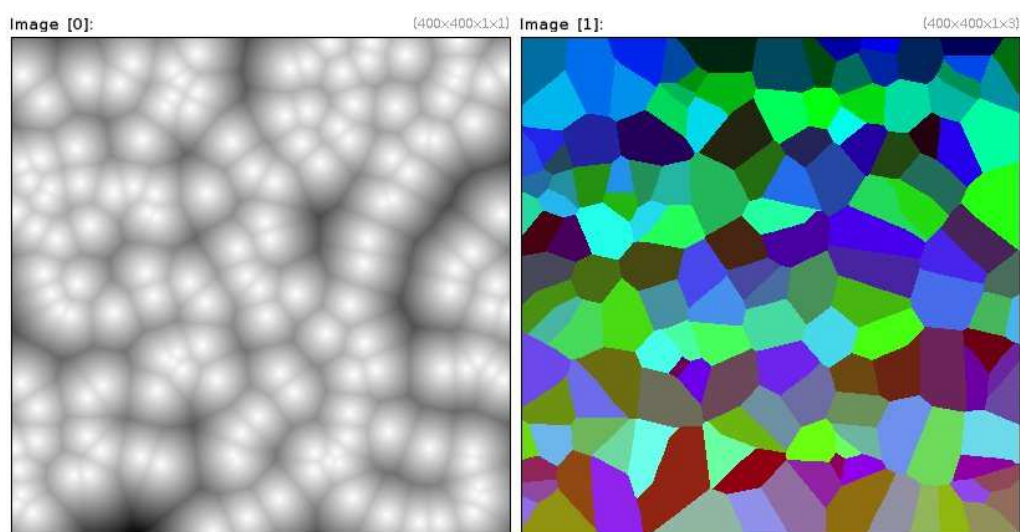
Example 347 : `image.jpg --watermark_fourier "Watermarked!" --display_fft -remove[-3,-1] -normalize 0,255 -append[-4,-2] y -append[-2,-1] y`

2.8.91 -watershed (+)

Arguments: `[priority_image],_fill_lines={ 0 | 1 }`

Compute the watershed transform of selected images.

Default value: `'fill_lines=1'`.



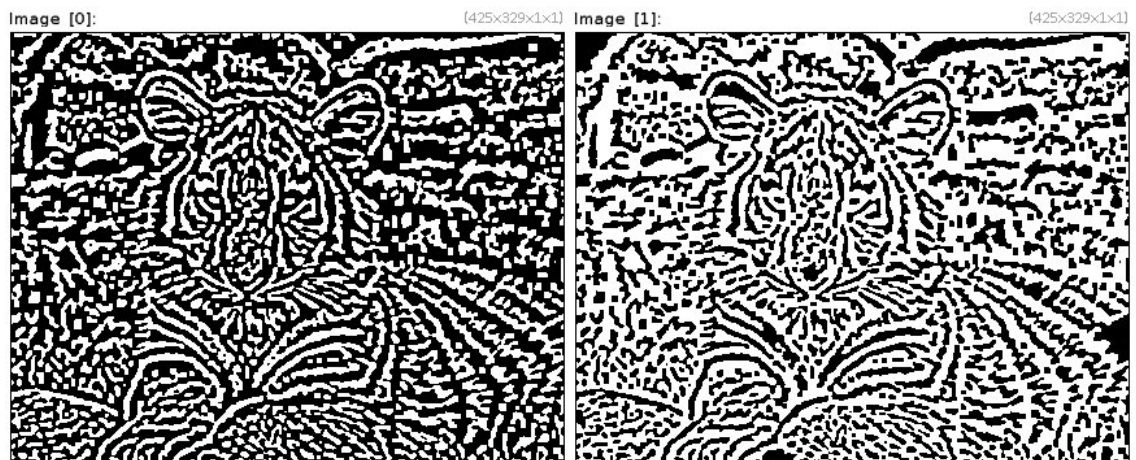
Example 348 : `400,400 -noise 0.2,2 --distance 1 -mul[-1] -1 -label[-2]
-watershed[-2] [-1] -mod[-2] 256 -map[-2] 0 -reverse`

2.9 Features extraction**2.9.1 -area**

Arguments: `tolerance>=0,is_high_connectivity={ 0 | 1 }`

Compute area of connected components in selected images.

Default values: `'is_high_connectivity=0'`.



Example 349 : `image.jpg -luminance -stencil[-1] 1 --area 0`

Tutorial page:

http://gmic.eu/tutorial/_area.shtml

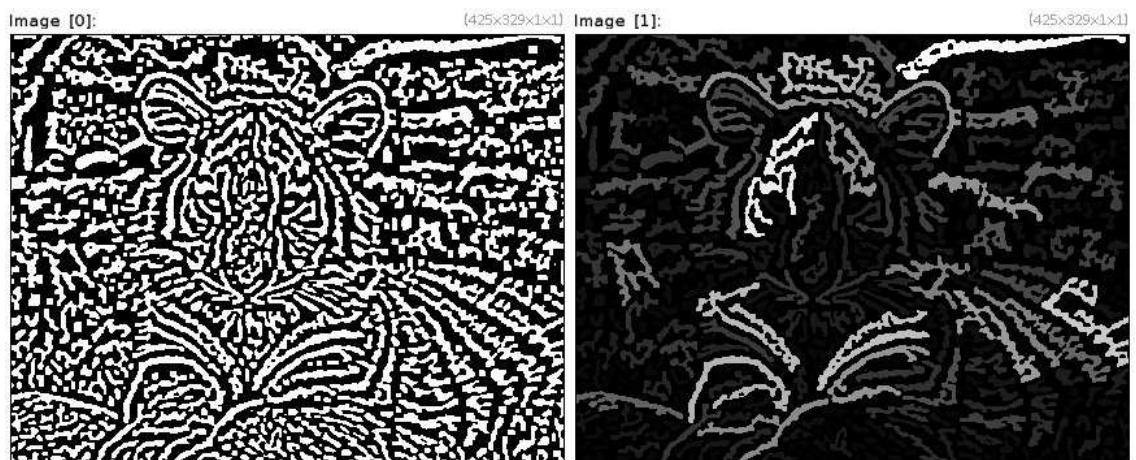
2.9.2 *-area_fg*

Arguments: `tolerance>=0, is_high_connectivity={ 0 | 1 }`

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 350 : `image.jpg -luminance -stencil[-1] 1 --area_fg 0`

2.9.3 *-at_line*

Arguments: `x0[%],y0[%],z0[%],x1[%],y1[%],z1[%]`

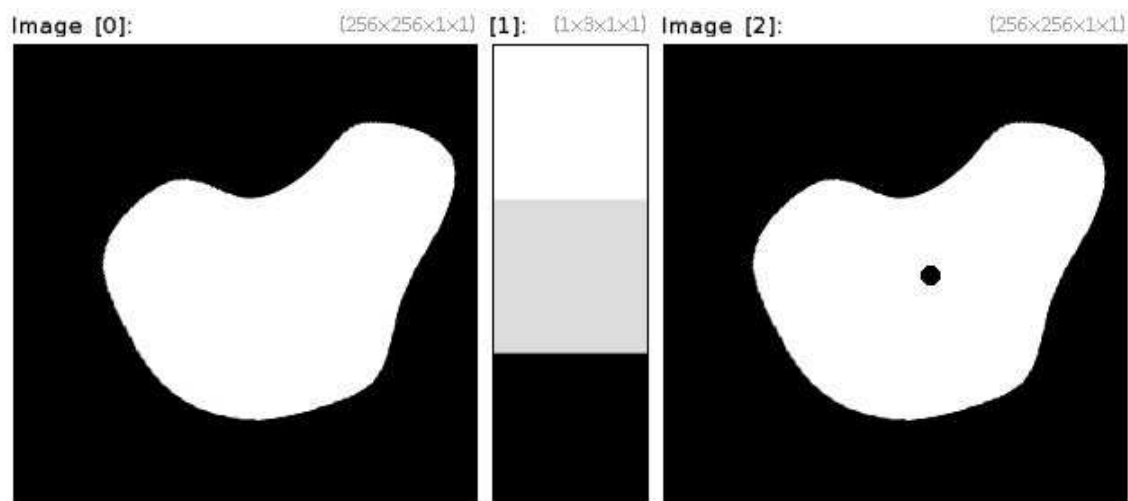
Retrieve pixels of the selected images belonging to the specified line $(x_0, y_0, z_0) - (x_1, y_1, z_1)$.



Example 351 : `image.jpg --at_line 0,0,0,100%,100%,0`

2.9.4 *-barycenter*

Compute the barycenter vector of pixel values.



Example 352 : `256,256 -ellipse 50%,50%,20%,20%,0,1,1 -deform 20 --barycenter
--ellipse[-2] {[0,1]},5,5,0,10`

2.9.5 *-detect_skin*

Arguments: `0<=tolerance<=1, _skin_x, _skin_y, _skin_radius>=0`

Detect skin in selected color images and output an appartenance 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 value: '`tolerance=0.5`' and '`skin_x=skiny=radius=-1`'.

2.9.6 *-displacement (+)*

Arguments: `[source_image], _smoothness, _precision>=0, _nb_scales>=0, iteration_max>=0, is_backward={ 0 | 1 }, -[constraints]`

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 needed scales is estimated from the image size.

When specified, an image of constraints is a Nx4 or Nx6 image that contains N ground-truth values for the estimated displacement field, each column encoding a position and its value as (x,y,ux,uy) (for a 2d field), or (x,y,z,ux,uy,uz) (for a 3d field).

Default values: '`smoothness=0.1`', '`precision=7`', '`nb_scales=0`', '`iteration_max=10000`', '`is_backward=1`' and '`[constraints]=(unused)`'.



Example 353 : `image.jpg --rotate 3,1,0,50%,50%,0.9 --displacement[-1] [-2]
-quiver[-1] [-1],15,-20,1,1,{1.5*iM}`

2.9.7 *-distance (+)*

Arguments: `isovalue[%],_metric |`
`isovalue[%],[metric],_method`

Compute the unsigned distance function to specified isovalue, opt. according to a custom metric.

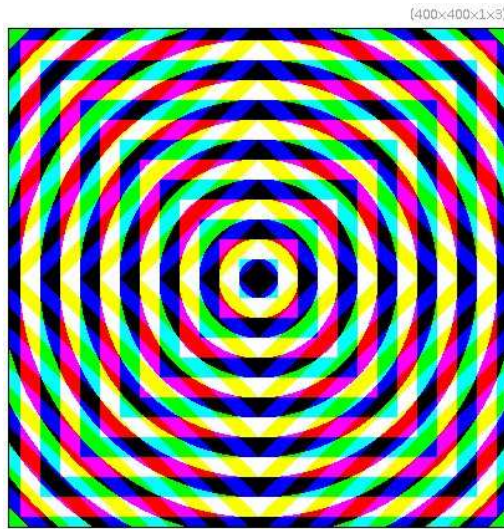
'metric' can be { 0=chebyshev | 1=manhattan | 2=euclidean
| 3=squared-euclidean }.

'method' can be { 0=fast-marching | 1=low-connectivity dijkstra
| 2=high-connectivity dijkstra | 3=1+return path | 4=2+return path }.

Default value: 'metric=2' and 'method=0'.



Example 354 : `image.jpg -threshold 20% -distance 0 -pow 0.3`



Example 355 : `400,400 -set 1,50%,50% --distance[0] 1,2 --distance[0] 1,1
-distance[0] 1,0 -mod 32 -threshold 16 -append c`

Tutorial page:

http://gmic.eu/tutorial/_distance.shtml

2.9.8 *-float2fft8*

Convert selected float-valued images to 8bits fourier representations.

2.9.9 *-fft82float*

Convert selected 8bits fourier representations to float-valued images.

2.9.10 *-fftpolar*

Compute fourier transform of selected images, as centered magnitude/phase images.



Example 356 : `image.jpg -fftpolar -ellipse 50%,50%,10,10,0,1,0 -ifftpolar`

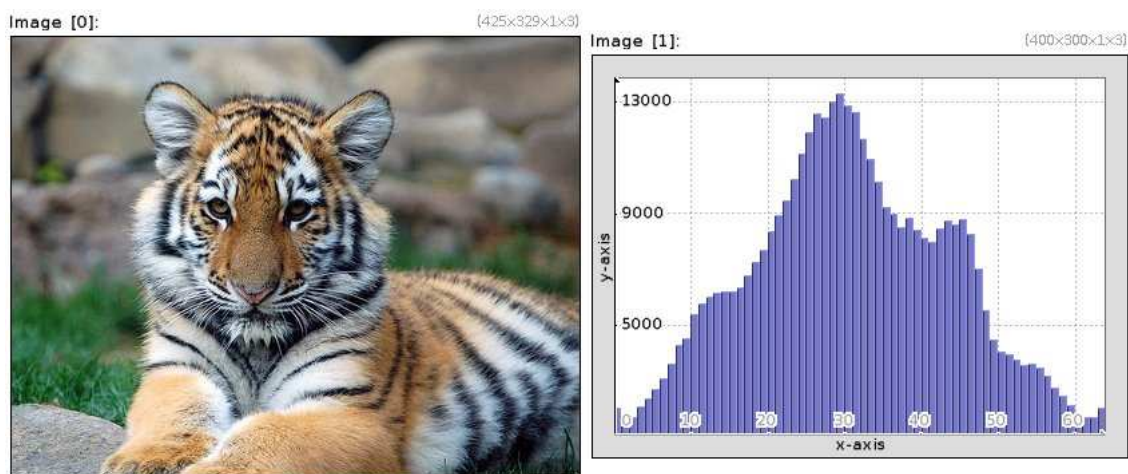
2.9.11 *-histogram (+)*

Arguments: `_nb_levels>0[%],_value0[%],_value1[%]`

Compute the histogram of selected images.

If value range is set, the histogram is estimated only for pixels in the specified value range. Argument 'value1' must be specified if 'value0' is set.

Default values: 'nb_levels=256', 'value0=0%' and 'value1=100%'.



Example 357 : `image.jpg --histogram 64 -display_graph[-1] 400,300,3`

2.9.12 *-histogram_nd*

Arguments: `nb_levels>0[%],_value0[%],_value1[%]`

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%'`.



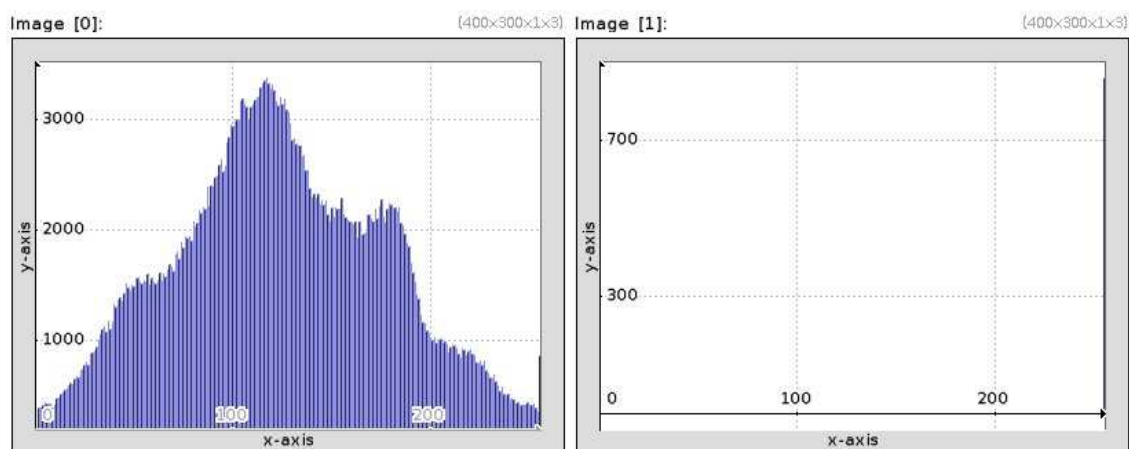
Example 358 : `image.jpg -channels 0,1 --histogram_nd 256`

2.9.13 *-histogram_cumul*

Arguments: `_nb_levels>0,_is_normalized={ 0 | 1 },_val0[%],_val1[%]`

Compute cumulative histogram of selected images.

Default values: `'nb_levels=256'`, `'is_normalized=0'` and `'val0=val1=0'`.



Example 359 : `image.jpg --histogram_cumul 256 -histogram[0] 256 -display_graph 400,300,3`

2.9.14 *-histogram_pointwise*

Arguments: `nb_levels>0[%],_value0[%],_value1[%]`

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.

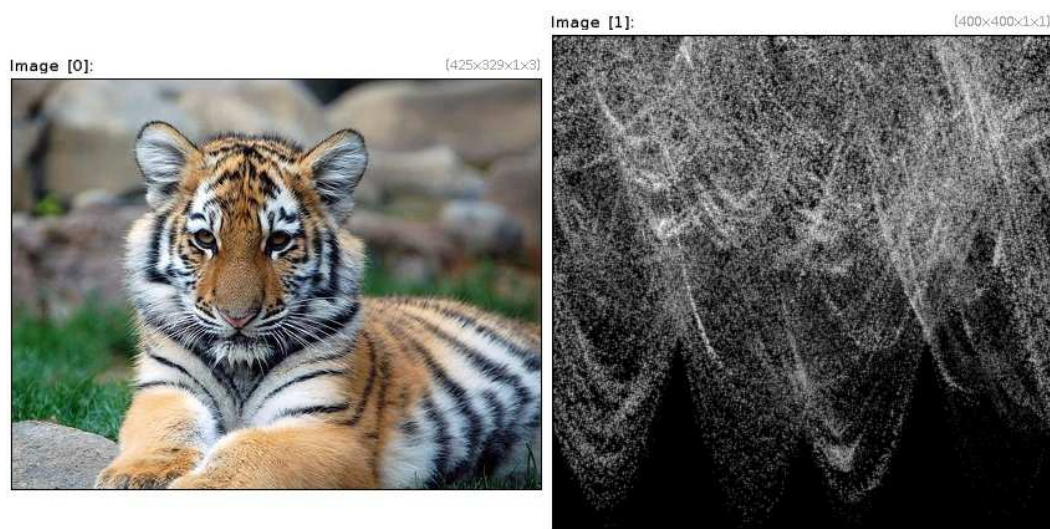
Default values: `'value0=0%'` and `'value1=100%'`.

2.9.15 *-hough*

Arguments: `_width>0,_height>0,gradient_norm_voting={ 0 | 1 }`

Compute hough transform (theta,rho) of selected images.

Default values: `'width=512'`, `'height=width'` and `'gradient_norm_voting=1'`.



Example 360 : `image.jpg --blur[-1] 1.5 -hough[-1] 400,400 -blur[-1] 0.5
--[-1] 1 -log[-1]`

2.9.16 *-ifftpolar*

Compute inverse fourier transform of selected images, from centered magnitude/phase images.

2.9.17 *-isophotes*

Arguments: `_nb_levels>0`

Render isophotes of selected images on a transparent background.

Default value: `'nb_levels=64'`



Example 361 : `image.jpg -blur 2 -isophotes 6 -dilate_circ 5 -display_rgba`

2.9.18 *-label (+)*

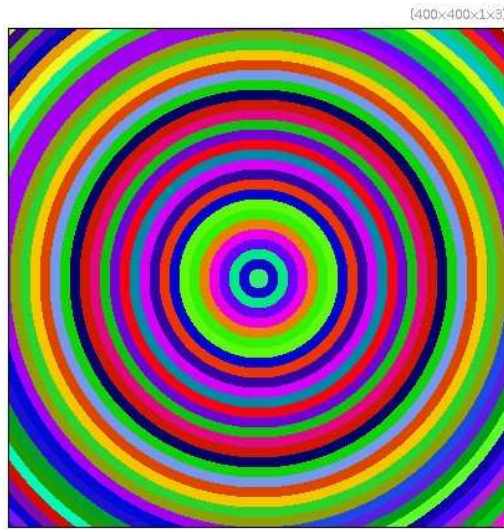
Arguments: `_tolerance>=0, is_high_connectivity={ 0 | 1 }`

Label connected components in selected images.

Default values: `'tolerance=0'` and `'is_high_connectivity=0'`.



Example 362 : `image.jpg -luminance -threshold 60% -label -normalize 0,255
-map 0`



Example 363 : `400,400 -set 1,50%,50% -distance 1 -mod 16 -threshold 8 -label
-mod 255 -map 2`

Tutorial page:

http://gmic.eu/tutorial/_label.shtml

2.9.19 *-label fg*

Arguments: `tolerance>=0,is_high_connectivity={ 0 | 1 }`

Label connected components for non-zero values (foreground) in selected images.

Similar to `'-label'` except that 0-valued pixels are not labeled.

Default value: `'is_high_connectivity=0'`.

2.9.20 *-max_patch*

Arguments: `_patch_size>=1`

Return locations of maximal values in local patch-based neighborhood of given size for selected images.

Default value: `'patch_size=16'`.



Example 364 : `image.jpg -norm --max_patch 16`

2.9.21 *-min_patch*

Arguments: `_patch_size>=1`

Return locations of minimal values in local patch-based neighborhood of given size for selected images.

Default value: `'patch_size=16'`.



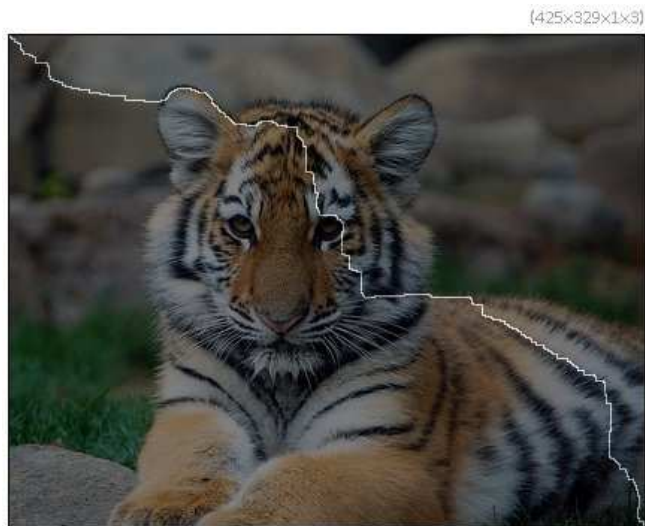
Example 365 : `image.jpg -norm --min_patch 16`

2.9.22 *-minimal_path*

Arguments: `x0[%]>=0,y0[%]>=0,z0[%]>=0,x1[%]>=0,y1[%]>=0,z1[%]>=0,_is_high_connectivity={ 0 | 1 }`

Compute minimal path between two points on selected potential maps.

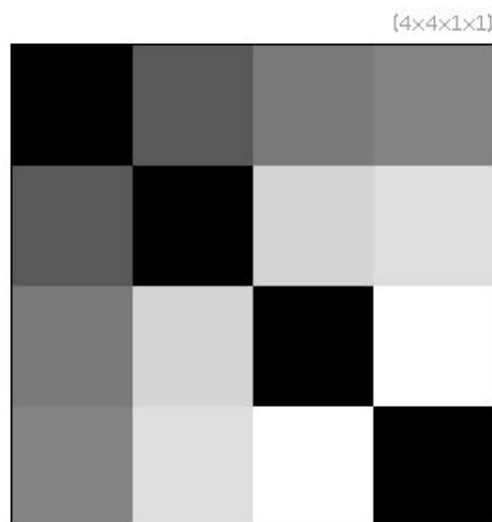
Default value: `'is_high_connectivity=0'`.



Example 366 : `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] -resize[-1] [-2],0 -or`

2.9.23 *-mse (+)*

Compute MSE (Mean-Squared Error) matrix between selected images.

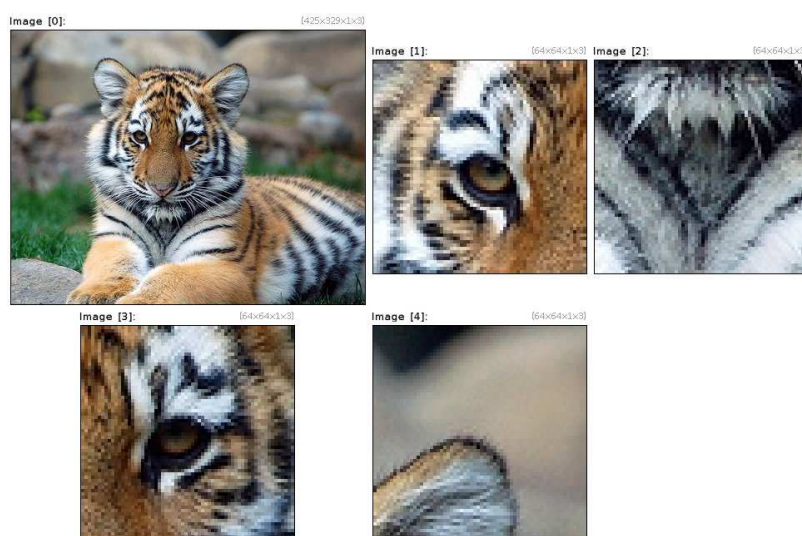


Example 367 : `image.jpg --noise 30 --noise[0] 35 --noise[0] 38 -cut[-1] 0,255 -mse`

2.9.24 *-patches*

Arguments: `patch_width>0,patch_height>0,patch_depth>0,x0,y0,z0,_x1,_y1,-_z1,...,_xN,_yN,-_zN`

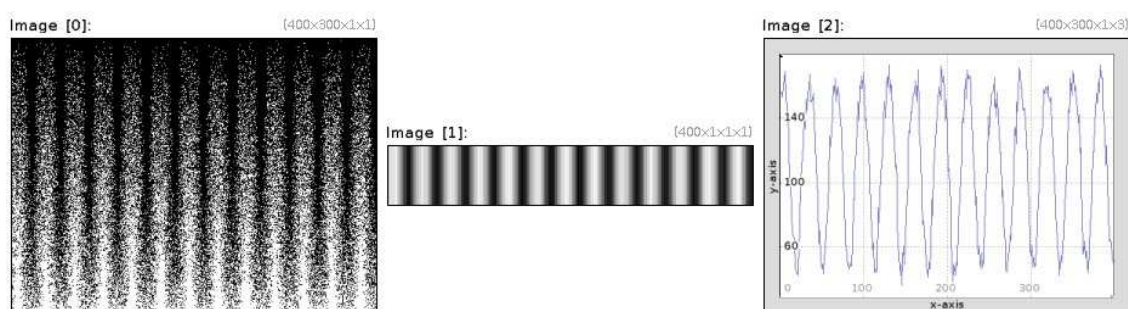
Extract N+1 patches from selected images, centered at specified locations.



Example 368 : `image.jpg --patches 64,64,1,153,124,0,184,240,0,217,126,0,275,38,0`

2.9.25 *-plot2value*

Retrieve values from selected 2d graph plots.



Example 369 : `400,300,1,1,'if(y>300*abs(cos(x/10+2*?)),1,0)' --plot2value --display_graph[-1] 400,300`

2.9.26 *-pointcloud*

Arguments: `_type = { -X=-X-opacity | 0=binary | 1=cumulative`

```
| 2=label },_width,_height>0,_depth>0
```

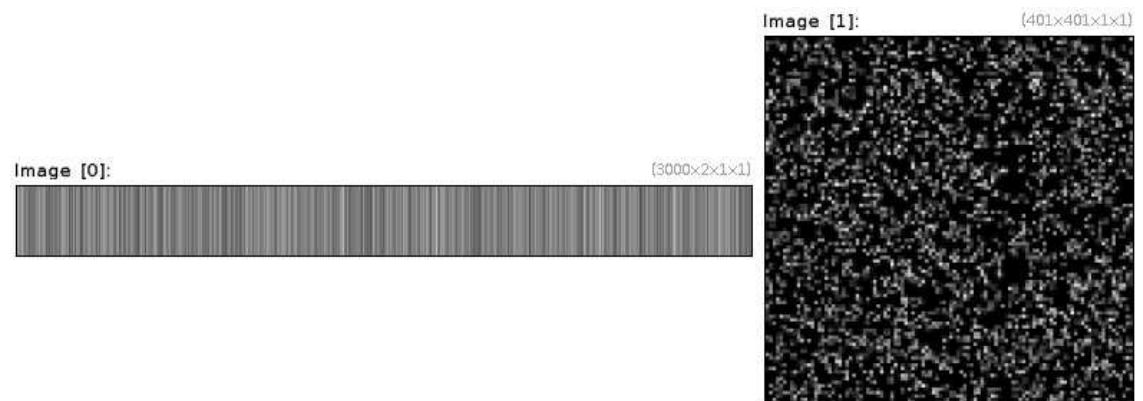
Convert a Nx1, Nx2, Nx3 or NxM image as a point cloud in a 1d/2d or 3d binary image.

If 'M'>3, the 3-to-M lines sets the (M-3)-dimensional color at each point.

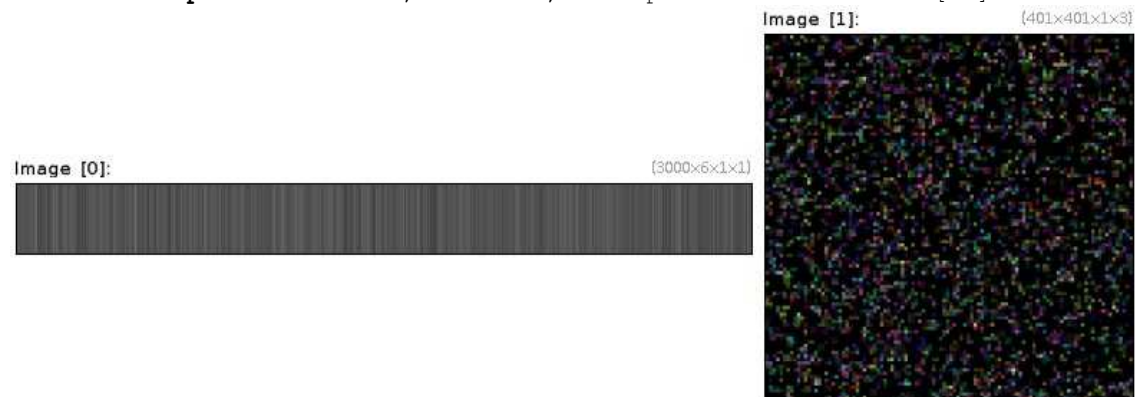
Parameters 'width','height' and 'depth' are related to the size of the final image : - If set to 0, the size is automatically set along the specified axis. - If set to N>0, the size along the specified axis is N. - If set to N<0, the size along the specified axis is at most N.

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'.



Example 370 : 3000,2 -rand 0,400 --pointcloud 0 -dilate[-1] 3



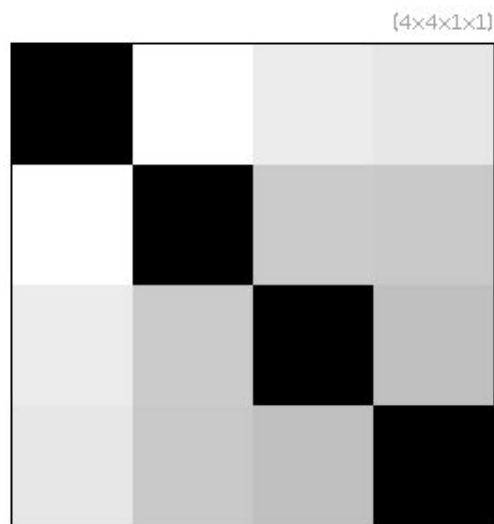
Example 371 : 3000,2 -rand 0,400 {w} {w},3 -rand[-1] 0,255 -append y
--pointcloud 0 -dilate[-1] 3

2.9.27 *-psnr*

Arguments: _max_value

Compute PSNR (Peak Signal-to-Noise Ratio) matrix between selected images.

Default value: 'max_value=255'.



Example 372 : image.jpg --noise 30 --noise[0] 35 --noise[0] 38 -cut[-1] 0,255
 -psnr 255 -replace_inf 0

2.9.28 *-segment_watershed*

Arguments: _threshold>=0, _fill_lines={ 0 | 1 }

Apply watershed segmentation on selected images.

Default values: 'threshold=2' and 'fill_lines=1'.



Example 373 : `image.jpg --segment.watershed 2,0`

2.9.29 *-skeleton*

Arguments: `_smoothness[%]>=0`

Compute skeleton of binary shapes using distance transform.

Default value: `'smoothness=0'`.



Example 374 : `image.jpg -threshold 50% --skeleton 0`

2.9.30 *-ssd_patch*

Arguments: `[patch],_use_fourier={ 0 | 1 },_boundary_conditions={ 0=dirichlet | 1=neumann }`

Compute fields of SSD between selected images and specified

patch.

Argument 'boundary_conditions' is valid only when 'use_fourier=0'.

Default value: 'use_fourier=0' and 'boundary_conditions=0'.



Example 375 : `image.jpg --crop 20%,20%,35%,35% --ssd_patch[0] [1],0,0`

2.9.31 -thinning

Compute skeleton of binary shapes using morphological thinning (This is a quite slow iterative proces)

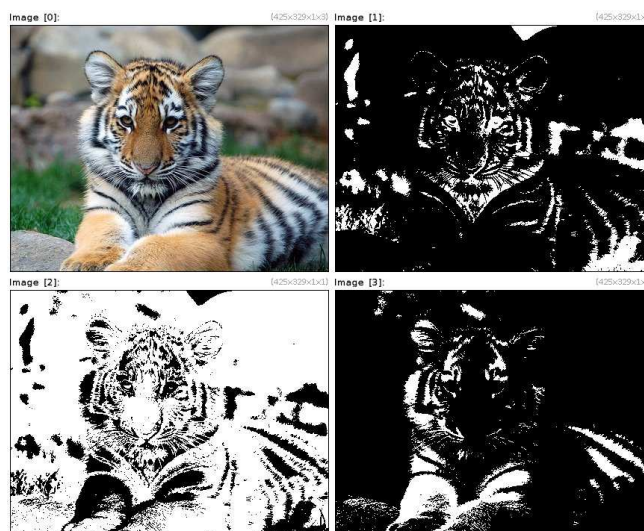


Example 376 : `image.jpg -threshold 50% --thinning`

2.9.32 -tones

Arguments: `N>0`

Get N tones masks from selected images.



Example 377 : `image.jpg --tones 3`

2.9.33 *-topographic_map*

Arguments: `_nb_levels>0, _smoothness`

Render selected images as topographic maps.

Default values: `'nb_levels=16'` and `'smoothness=2'`.



Example 378 : `image.jpg --topographic_map 10`

2.9.34 *-variance_patch*

Arguments: `_patch_size>=1`

Compute variance of each images patch centered at (x,y) , in selected images.

Default value: `'patch.size=16'`



Example 379 : `image.jpg --variance_patch`

2.10 Image drawing

2.10.1 *-axes*

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

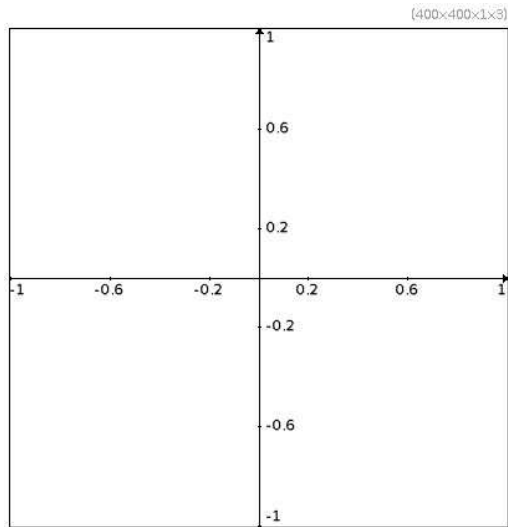
Draw xy-axes on selected images.

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

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 380 : `400,400,1,3,255 -axes -1,1,1,-1`

2.10.2 *-ball*

Arguments: `_size>0, _R,_G,_B,0<=_specular_light<=8,0<=_specular_size<=8-
_shadow>=0`

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 381 : `-repeat 9 -ball {1.5^($>+2)},${-RGB} -done -append x`

2.10.3 *-chessboard*

Arguments: `size1>0, _size2>0, _offset1, _offset2, _angle, _opacity, _color1, .- , _color2, ..`

Draw chessboard on selected images.

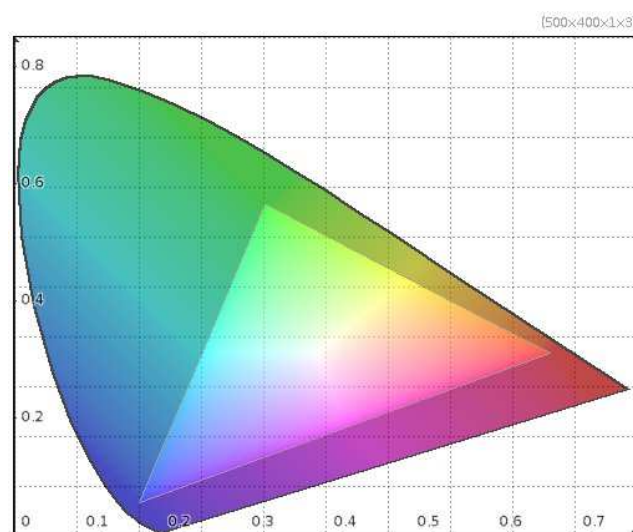
Default values: `'size2=size1', 'offset1=offset2=0', 'angle=0', 'opacity=1', 'color1=0' and 'color2=255'`.



Example 382 : `image.jpg -chessboard 32,32,0,0,25,0.3,255,128,0,0,128,255`

2.10.4 *-cie1931*

Draw CIE-1931 chromaticity diagram on selected images.



Example 383 : 500,400,1,3 -cie1931

2.10.5 -circle

Arguments: x[%],y[%],R[%],_opacity,_pattern,_color1,..

Draw specified colored circle on selected images.
A radius of '100%' stands for ' $\sqrt{\text{width}^2 + \text{height}^2}$ '.
'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 384 : image.jpg -repeat 300 -circle
{?(100)}%,{?(100)}%,{?(30)},0.3,\${-RGB} -done -circle 50%,50%,100,0.7,255

2.10.6 -ellipse (+)

Arguments: x[%],y[%],R[%],r[%],_angle,_opacity,_pattern,_color1,..

Draw specified colored ellipse on selected images.
A radius of '100%' stands for ' $\sqrt{\text{width}^2 + \text{height}^2}$ '.
'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 385 : `image.jpg -repeat 300 -ellipse
 {?(100)}%,{?(100)}%,{?(30)},{?(30)},{?(180)},0.3,{-RGB} -done -ellipse
 50%,50%,100,100,0,0.7,255`

2.10.7 *-flood* (+)

Arguments: `x[%],_y[%],_z[%],_tolerance>=0,_is_high_connectivity={ 0
 | 1 },_opacity,_color1,..`

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'.`



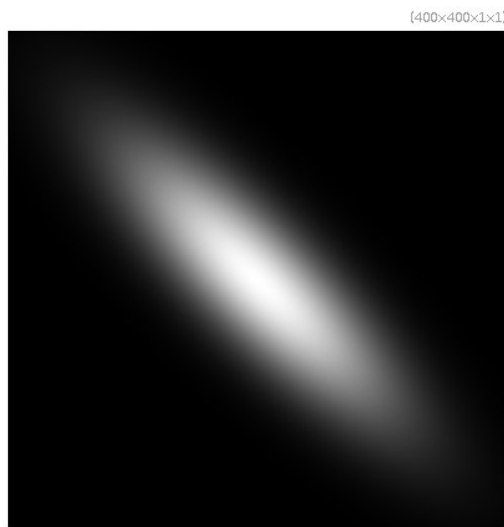
Example 386 : `image.jpg -repeat 1000 -flood
{?(100)}%,{?(100)}%,0,20,0,1,${-RGB} -done`

2.10.8 *-gaussian*

Arguments: `_sigma1[%],_sigma2[%],_angle`

Draw a centered gaussian on selected images, with specified standard deviations and orientation.

Default values: `'sigma1=3', 'sigma2=sigma1' and 'angle=0'`.



Example 387 : `400,400 -gaussian 100,30,45`

Tutorial page:

http://gmic.eu/tutorial/_gaussian.shtml

2.10.9 -graph (+)

Arguments: `[function_image],_plot_type,_vertex_type,_ymin,_ymax,_opacity,_y,_pattern,_color1,..` | `'formula',_resolution>=0,_plot_type,_vertex_type,_xmin,_xmax,_ymin,_ymax,_opacity,_pattern,_color1,..`

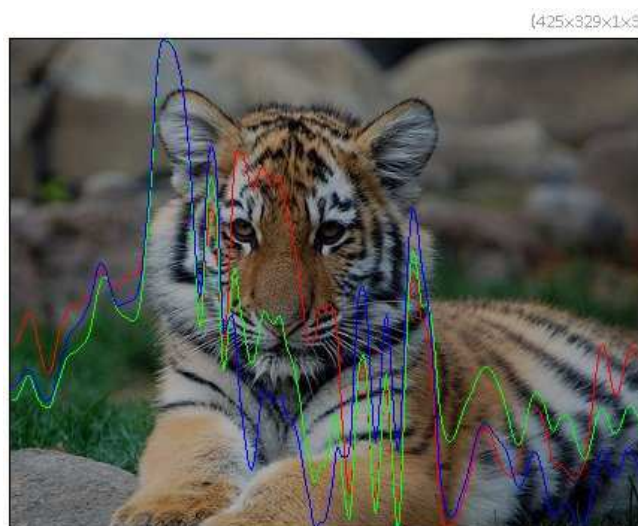
Draw specified function graph on selected images.

'plot_type' can be { 0=none | 1=lines | 2=splines | 3=bar }.

'vertex_type' can be { 0=none | 1=points | 2,3=crosses | 4,5=circles | 6,7=squares }.

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

Default values: 'plot_type=1', 'vertex_type=1', 'ymin=ymax=0 (auto)', 'opacity=1', 'pattern=(undefined)' and 'color1=0'.



Example 388 : `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]`

2.10.10 -grid

Arguments: `size_x[%]>=0,size_y[%]>=0,_offset_x[%],_offset_y[%],_opacity-`

```
,_pattern,_color1,..
```

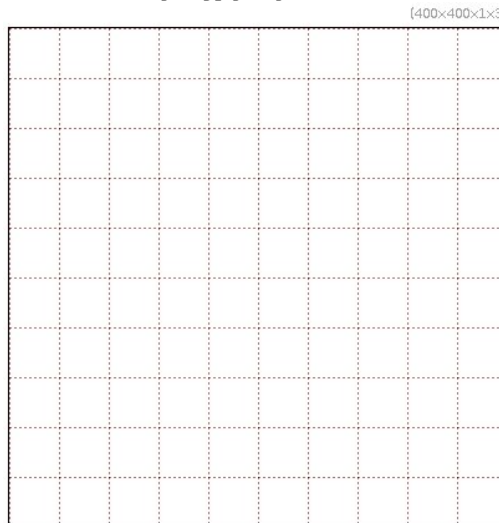
Draw xy-grid on selected images.

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

Default values: 'offset_x=offset_y=0', 'opacity=1', 'pattern=(undefined)' and 'color1=0'.



Example 389 : `image.jpg -grid 10%,10%,0,0,0.5,255`



Example 390 : `400,400,1,3,255 -grid 10%,10%,0,0,0.3,0xCCCCCCCC,128,32,16`

2.10.11 -image (+)

Arguments: `[sprite], -x[%], -y[%], -z[%], -c[%], -opacity, -[sprite_mask], -max_opacity_mask`

Draw specified sprite image on selected images.
(eq. to '-j').

Default values: `'x=y=z=c=0', 'opacity=1', 'sprite_mask=(undefined)'`
and `'max_opacity_mask=1'`.



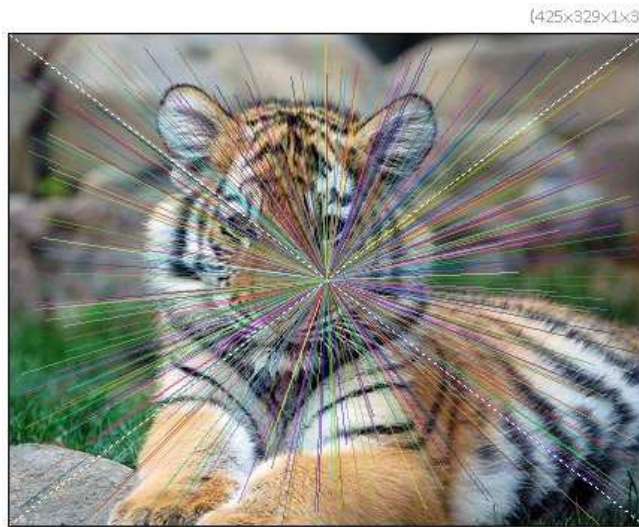
Example 391 : `image.jpg --crop 40%,40%,60%,60% -resize[-1] 200%,200%,1,3,5
-frame[-1] 2,2,0 -image[0] [-1],30%,30% -keep[0]`

2.10.12 -line (+)

Arguments: `x0[%], y0[%], x1[%], y1[%], -opacity, -pattern, -color1, ..`

Draw specified colored line on selected images.
'pattern' is a hexadecimal number starting with '0x' which can be omitted even if a color is specified.

Default values: `'opacity=1', 'pattern=(undefined)'` and `'color1=0'`.



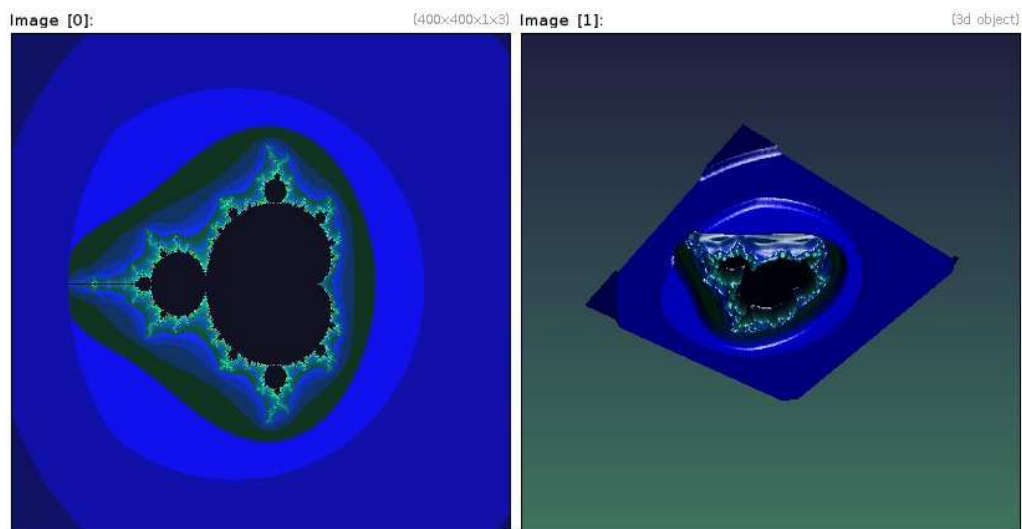
Example 392 : `image.jpg -repeat 500 -line 50%,50%,{?(w)},{?(h)},0.5,{-RGB}
-done -line 0,0,100%,100%,1,0xCCCCCCCC,255 -line
100%,0,0,100%,1,0xCCCCCCCC,255`

2.10.13 *-mandelbrot (+)*

Arguments: `z0r,z0i,z1r,z1i,iteration_max>=0,is_julia={ 0 | 1
,_c0r,_c0i,_opacity`

Draw mandelbrot/julia fractal on selected images.

Default values: `'iteration_max=100', 'is_julia=0', 'c0r=c0i=0' and
'opacity=1'.`



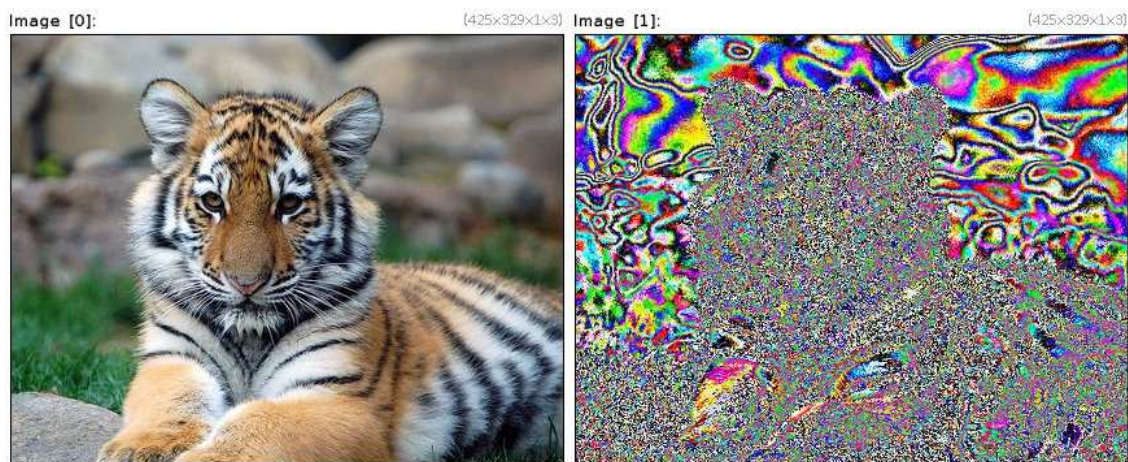
Example 393 : 400,400 -mandelbrot -2.5,-2,2,2,1024 -map 0 --blur 2
-elevation3d[-1] -0.2

2.10.14 -marble

Arguments: _image_weight, _pattern_weight, _angle, _amplitude, _sharpness>=0, _anisotropy>=0, _alpha, _sigma, _cut_low>=0, _cut_high>=0

Render marble like pattern on selected images.

Default values: 'image_weight=0.2', 'pattern_weight=0.1', 'angle=45', 'amplitude=0', 'sharpness=0.4', 'anisotropy=0.8', 'alpha=0.6', 'sigma=1.1' and 'cut_low=cut_high=0'.

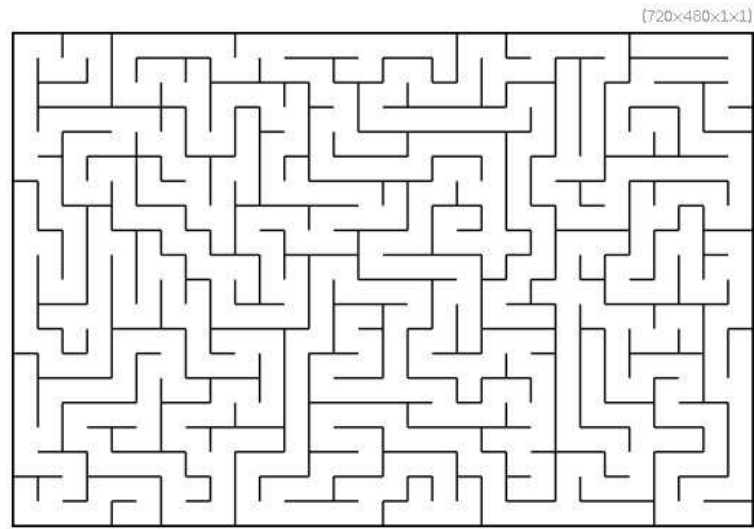


Example 394 : image.jpg --marble ,

2.10.15 -maze

Arguments: _width>0, _height>0, _cell_size>0

Input maze with specified size.



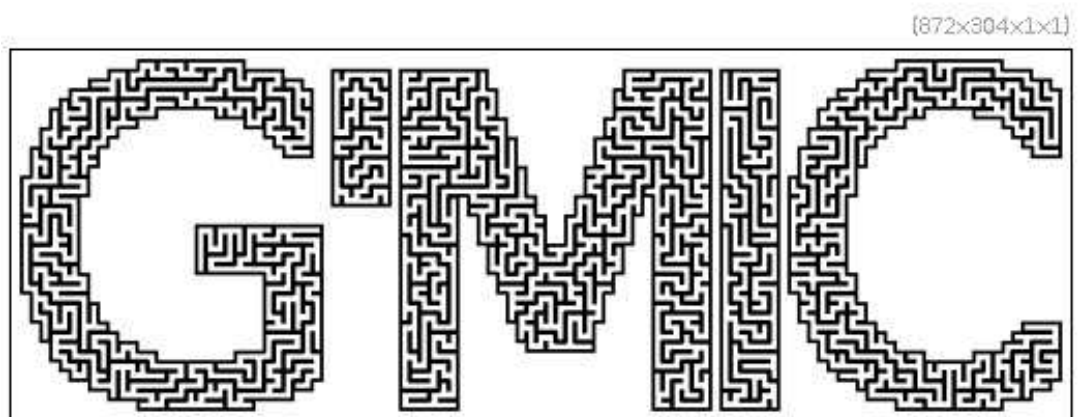
Example 395 : `-maze 30,20 -negative -normalize 0,255`

2.10.16 *-maze_mask*

Arguments: `_cellsize>0`

Input maze according to size and shape of selected mask images.

Mask may contain disconnected shapes.



Example 396 : `0 -text "G'MIC",0,0,53,1,1 -dilate 3 -autocrop 0 -frame 1,1,0
-maze_mask 8 -dilate 3 -negative -* 255`

2.10.17 -object3d (+)

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`

Draw specified 3d object on selected images.

(eq. to '-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 397 : `image.jpg -torus3d 100,10 -cone3d 30,-120 -add3d[-2,-1]
-rotate3d[-1] 1,1,0,60 -object3d[0] [-1],50%,50% -keep[0]`

2.10.18 -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`

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.

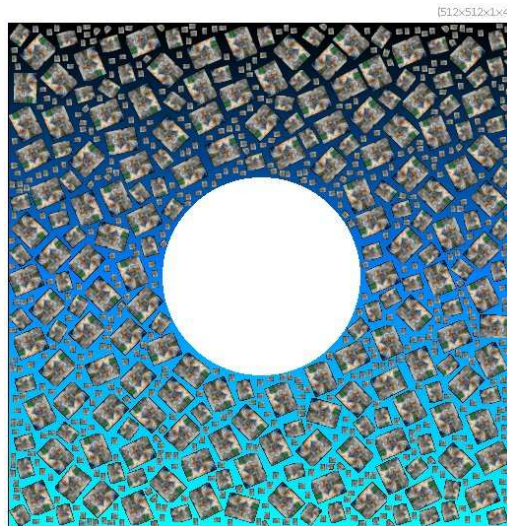
'nb_scales' sets the number of decreasing scales considered for all specified sprites to be packed.

'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'.



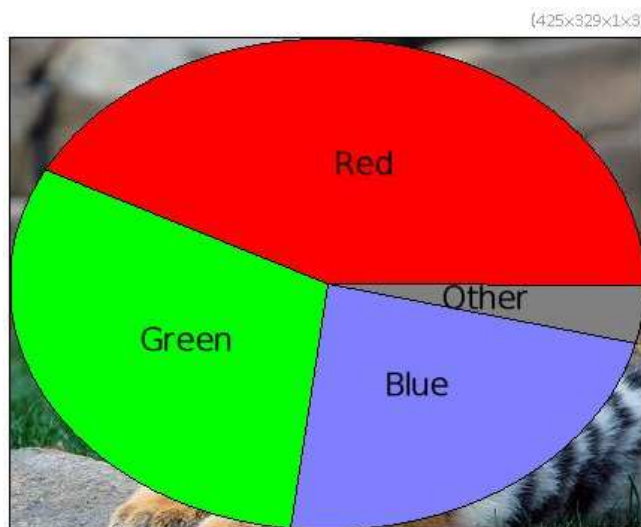
Example 398 : `512,512,1,3,"min(255,y*c/2)" 100%,100% -circle
50%,50%,100,1,255 -append c image.jpg -resize2dy[-1] 24 -to_rgba -pack_sprites
3,25`

2.10.19 *-piechart*

Arguments: `label_height>=0,label_R,label_G,label_B,"label1",value1,R1,G-`


```
1,B1,...,"labelN",valueN,RN,GN,BN
```

Draw pie chart on selected (RGB) images.



Example 399 : `image.jpg -piechart`

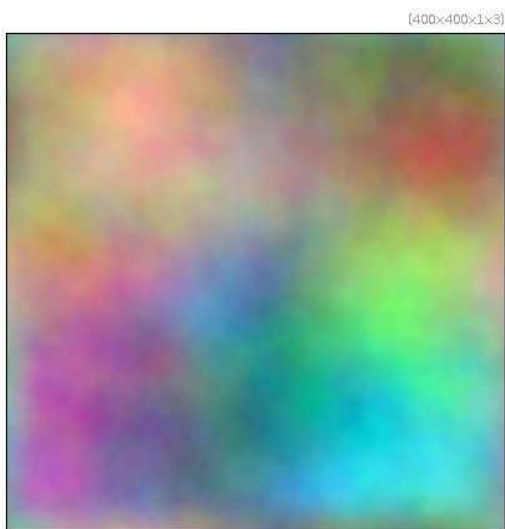
```
25,0,0,0,"Red",55,255,0,0,"Green",40,0,255,0,"Blue",30,128,128,255,"Other",5,128,128,128
```

2.10.20 *-plasma* (+)

Arguments: `_alpha,_beta,_scale>=0`

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'.`



Example 400 : 400,400,1,3 -plasma

Tutorial page:

http://gmics.eu/tutorial/_plasma.shtml

2.10.21 -point (+)

Arguments: x[%],y[%],z[%],_opacity,_color1,..

Set specified colored pixel on selected images.

Default values: 'z=0', 'opacity=1' and 'color1=0'.



Example 401 : `image.jpg -repeat 10000 -point {?(100)}%,{?(100)}%,0,1,${-RGB}
-done`

2.10.22 *-polka_dots*

Arguments: `diameter>=0, _density, _offset1, _offset2, _angle, _aliasing, _shading, _opacity, _color, ...`

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 402 : `image.jpg -polka_dots 10,15,0,0,20,10,1,0.5,0,128,255`

2.10.23 *-polygon (+)*

Arguments: `N>=1,x1[_%],y1[_%],...,xN[_%],yN[_%], _opacity, _pattern, _color1, ..`

Draw specified colored N-vertices polygon on selected images. 'pattern' is a 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'.`



Example 403 : `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%,1,0xCCCCCCCC,255`
(425x329x1x3)



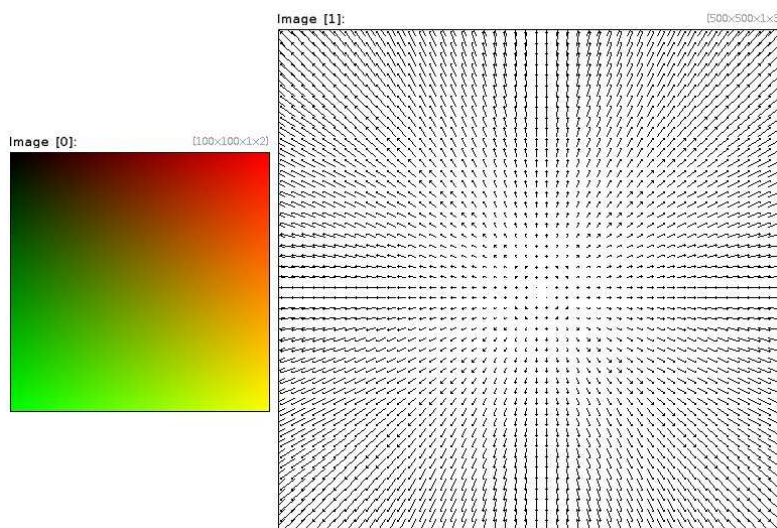
Example 404 : `image.jpg 2,16,1,1,'?(if(x,{h},{w}))' -polygon[-2]`
`{h},{^},0.6,255,0,255 -remove[-1]`

2.10.24 *-quiver* (+)

Arguments: `[function_image],_sampling>0,_factor,_is_arrow={ 0 | 1`
`},_opacity,_pattern,_color1,..`

Draw specified 2d vector/orientation field on selected images.
 'pattern' is an hexadecimal number starting with '0x' which can
 be omitted even if a color is specified.

Default values: 'sampling=25', 'factor=-20', 'is.arrow=1', 'opacity=1', 'pattern=(undefined)' and 'color1=0'.



Example 405 : `100,100,1,2,'if (c==0,x-w/2,y-h/2)' 500,500,1,3,255 -quiver[-1] [-2],10`



Example 406 : `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],10,10,1,0.8,255`

2.10.25 -rectangle

Arguments: `x0[%],y0[%],x1[%],y1[%],_opacity,_pattern,_color1,..`

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'.



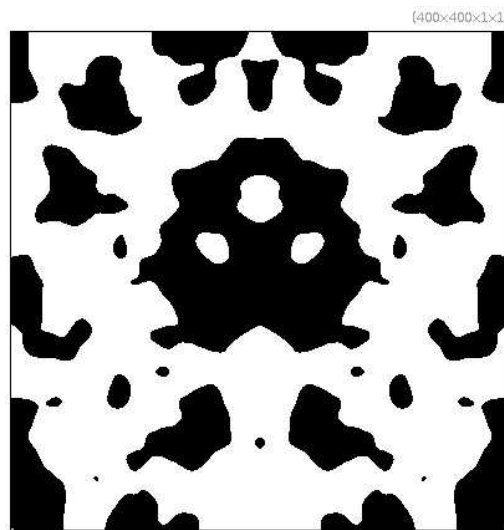
Example 407 : `image.jpg -repeat 30 -rectangle {?(100)}%,{?(100)}%,{?(100)}%,{?(100)}%,0.3,${-RGB} -done`

2.10.26 -rorschach

Arguments: 'smoothness[%]>=0', 'mirroring={ 0=none | 1=x | 2=y | 3=xy }

Render rorschach-like inkblots on selected images.

Default values: 'smoothness=5%' and 'mirroring=1'.



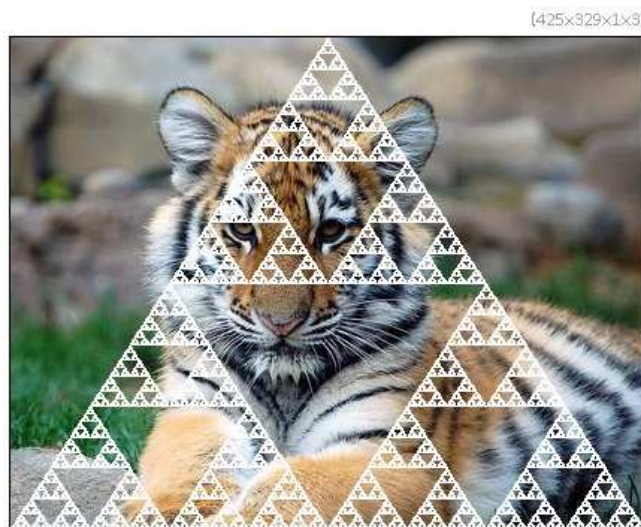
Example 408 : 400,400 -rorschach 3%

2.10.27 -sierpinski

Arguments: recursion_level>=0

Draw Sierpinski triangle on selected images.

Default value: 'recursion_level=7'.



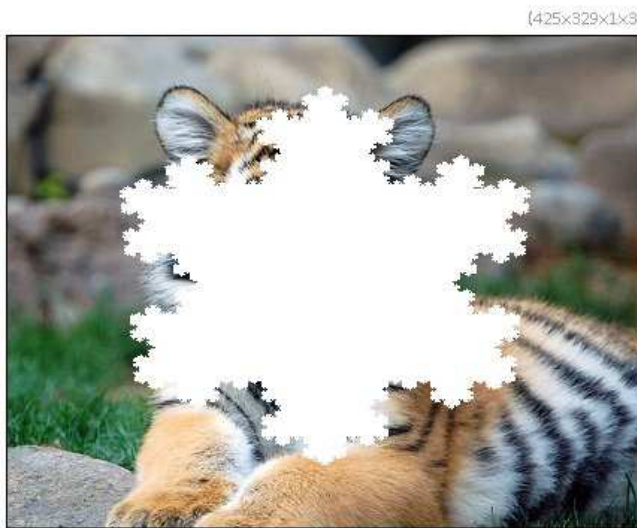
Example 409 : image.jpg -sierpinski 7

2.10.28 *-snowflake*

Arguments: `_recursion>=0, _x0, _y0, _x1, _y1, _x2, _y2, _opacity, _coll, ..._coll-N`

Draw a Koch snowflake on selected images.

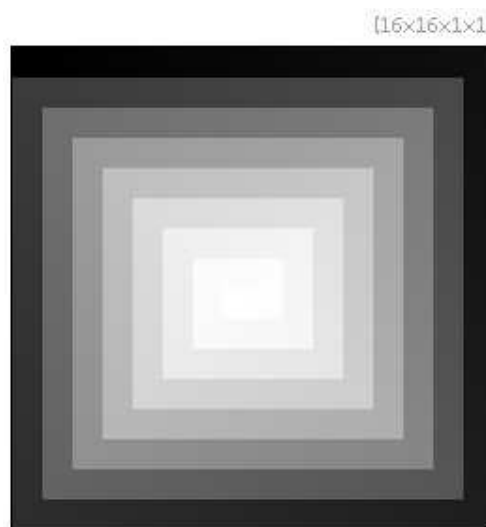
Default values: `'recursion=4', 'x0=20', 'y0=70', 'x1=80', 'y1=70', 'x2=50', 'y2=10', 'opacity=1' and 'coll=255'.`



Example 410 : `image.jpg -snowflake 4`

2.10.29 *-spiralbw*

Draw (squared) spiral on selected images.



Example 411 : 16,16 -spiralbw

2.10.30 -spline

Arguments: x0[%],y0[%],u0[%],v0[%],x1[%],y1[%],u1[%],v1[%],_nb-vertices->=2,_opacity,_color1,..

Draw specified colored spline curve on selected images (cubic hermite spline).

Default values: 'nb_vertices=256', 'opacity=1' and 'color1=0'.



Example 412 : image.jpg -repeat 30 -spline
{?(100)}%,{?(100)}%,{?(-600,600)},{?(-600,600)},{?(100)}%,{?(100)}%,{?(-600,600)},{?(-600,600)},256,
-done

2.10.31 -text (+)

Arguments: text,_x[%],_y[%],_font_height[%]>=0,_opacity,_color1,..

Draw specified colored text string on selected images.
(eq. to '-t').

Exact pre-defined sizes are '13','23','53' and '103'.

Using these sizes ensures you draw binary letters without anti-aliasing.

Any other font size is interpolated from an exact size (the upper when possible).

Specifying an empty target image resizes it to new dimensions such that the image contains the entire text string.

Default values: 'opacity=1' and 'color1=0'.



Example 413 : `image.jpg -resize2dy 600 y=0 -repeat 30 -text {2*$>}" : This is a nice text, isn't it ?",10,$y,{2*$>},0.9,255 y={$y+2*$>} -done`



Example 414 : `0 -text "G'MIC",0,0,23,1,255`

2.10.32 *-text_outline*

Arguments: `text, _x[%], _y[%], _font_height>0, _outline>=0, _opacity, _color1-...`

Draw specified colored and outlined text string on selected images.

Default values: 'x=y=2', 'font_height=13', 'outline=2', 'opacity=1' and 'color1=255'.

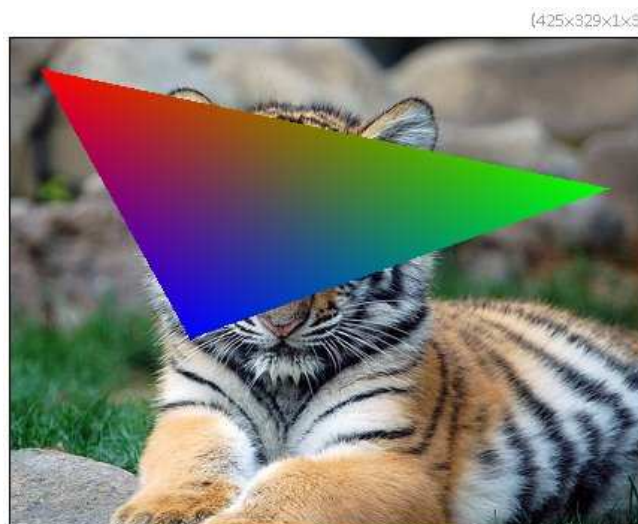


Example 415 : `image.jpg -text-outline "Hi there!",10,10,63,3`

2.10.33 *-triangle_shade*

Arguments: `x0,y0,x1,y0,x2,y2,R0,G0,B0,...,R1,G1,B1,...,R2,G2,B2,....`

Draw triangle with interpolated colors on selected images.



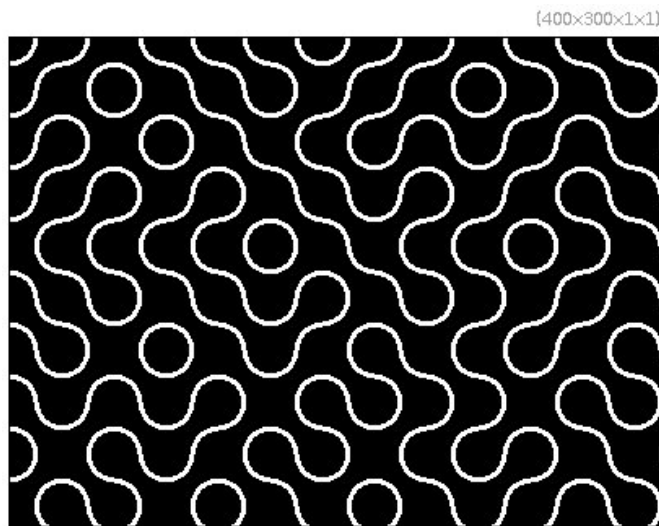
Example 416 : `image.jpg -triangle-shade
20,20,400,100,120,200,255,0,0,0,255,0,0,0,255`

2.10.34 *-truchet*

Arguments: `_scale>0, _radius>=0, _pattern_type={ 0=straight
| 1=curved }`

Fill selected images with random truchet patterns.

Default values: `'scale=32', 'radius=5' and 'pattern_type=1'.`



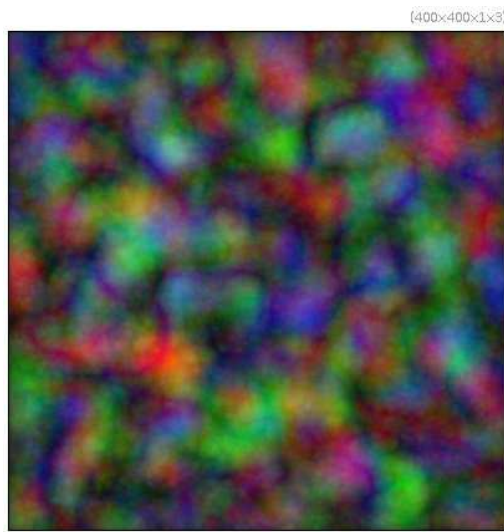
Example 417 : `400,300 -truchet ,`

2.10.35 *-turbulence*

Arguments: `_radius>0, _octaves={1,2,3...,12}, _alpha>0, _difference={-10,1-
0}, _mode={0,1,2,3}`

Render fractal noise or turbulence on selected images.

Default values: `'radius=32', 'octaves=6', 'alpha=3', 'difference=0'
and 'mode=0'.`



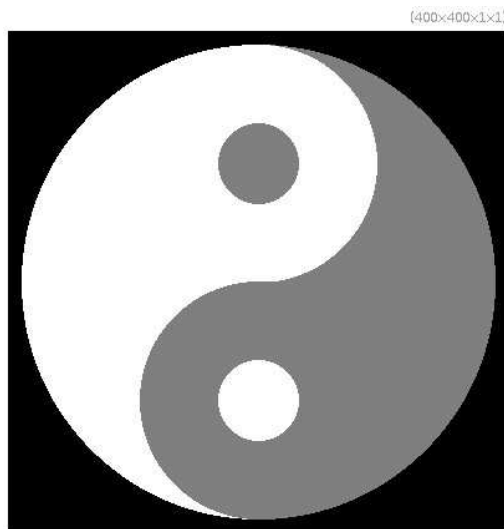
Example 418 : 400,400,1,3 -turbulence 16

Tutorial page:

http://gmic.eu/tutorial/_turbulence.shtml

2.10.36 -yinyang

Draw a yin-yang symbol on selected images.



Example 419 : 400,400 -yinyang

2.11 Matrix computation

2.11.1 *-dijkstra* (+)

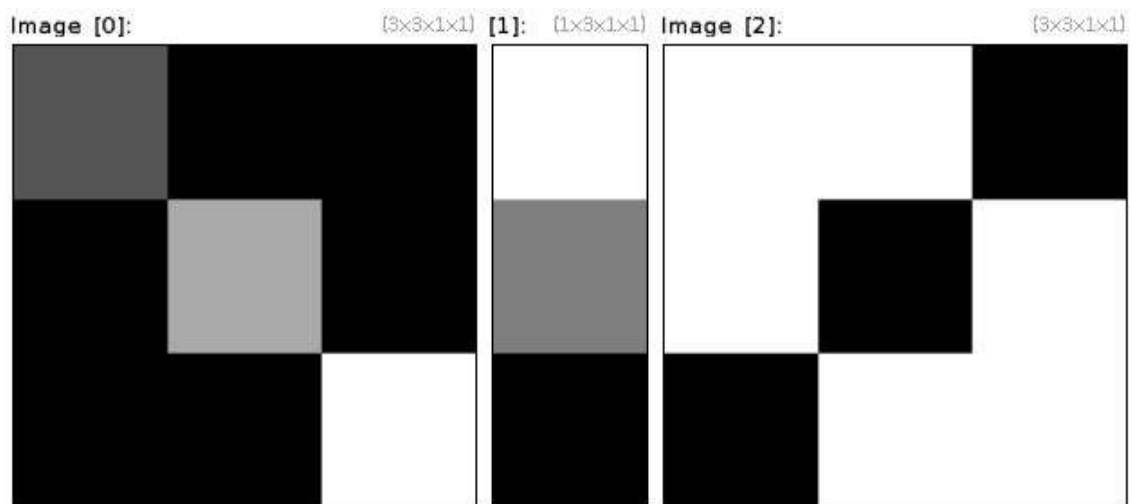
Arguments: `starting_node>=0,ending_node>=0`

Compute minimal distances and pathes from specified adjacency matrices by the Dijkstra algorithm.

2.11.2 *-eigen* (+)

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.



Example 420 : `(1,0,0;0,2,0;0,0,3) --eigen`



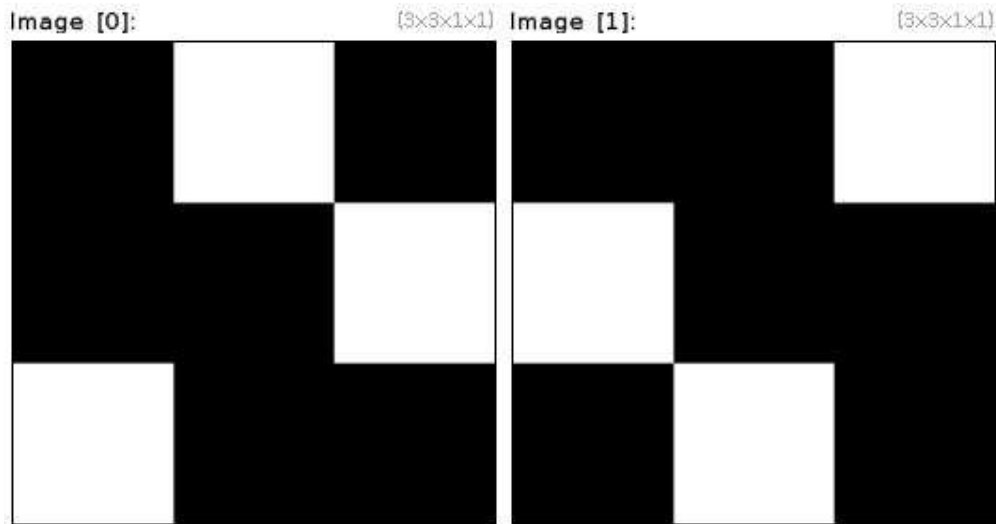
Example 421 : `image.jpg -structuretensors -blur 2 -eigen -split[0] c`

Tutorial page:

http://gmics.eu/tutorial/_eigen.shtml

2.11.3 -invert (+)

Compute the inverse of the selected matrices.



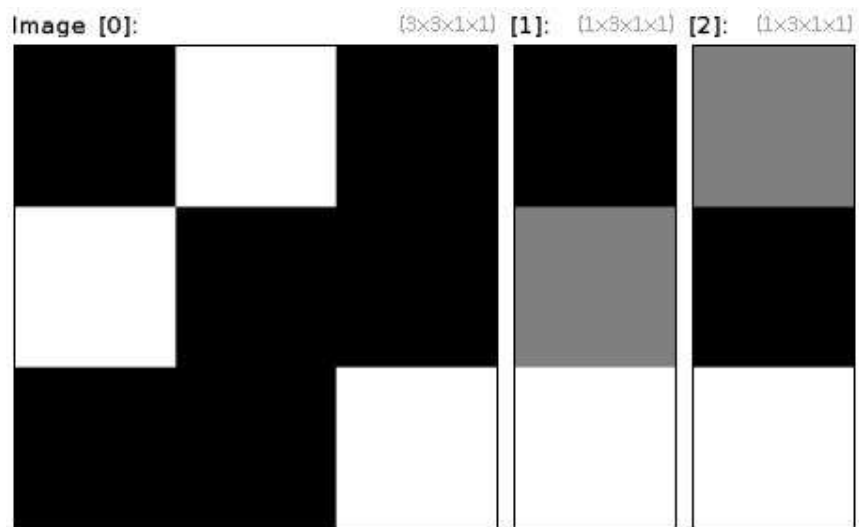
Example 422 : (0,1,0;0,0,1;1,0,0) --invert

2.11.4 -solve (+)

Arguments: [image]

Solve linear system $AX = B$ for selected B-matrices and specified A-matrix.

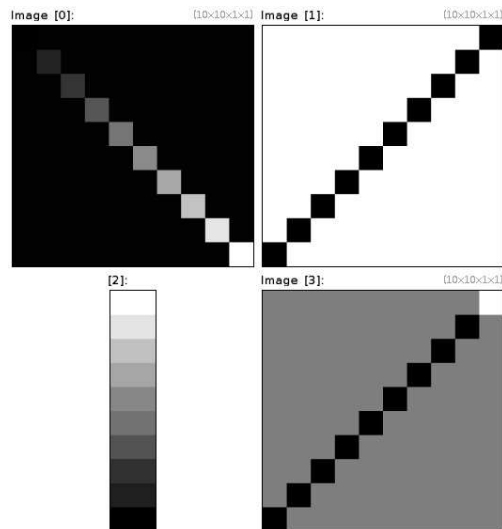
If the system is under- or over-determined, the least square solution is returned.



Example 423 : (0,1,0;1,0,0;0,0,1) (1;2;3) --solve[-1] [-2]

2.11.5 *-svd (+)*

Compute SVD decomposition of selected matrices.



Example 424 : `10,10,1,1,'if (x==y,x+?(-0.2,0.2),0)'` `--svd`

2.11.6 *-transpose*

Transpose selected matrices.

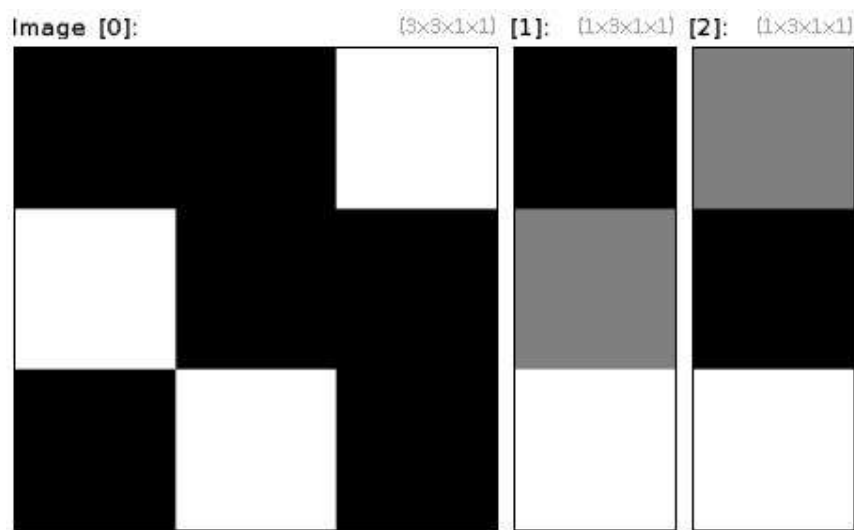


Example 425 : `image.jpg --transpose`

2.11.7 *-trisolve (+)*

Arguments: `[image]`

Solve tridiagonal system $AX = B$ for selected B-vectors and specified tridiagonal A-matrix.
 Tridiagonal matrix must be stored as a 3 column vector, where 2nd column contains the diagonal coefficients, while 1st and 3rd columns contain the left and right coefficients.



Example 426 : (0,0,1;1,0,0;0,1,0) (1;2;3) --trisolve[-1] [-2]

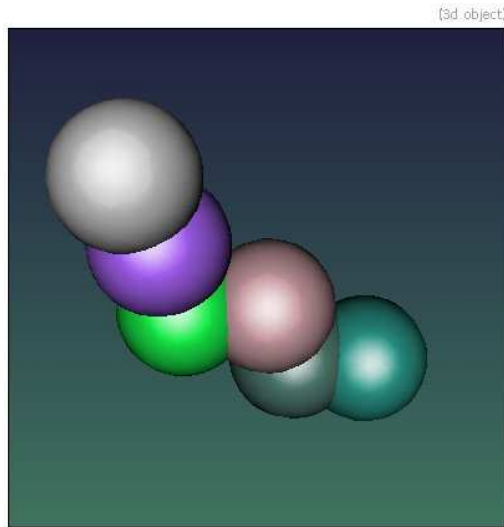
2.12 3d rendering

2.12.1 -add3d (+)

Arguments: tx,ty,tz |
 [object3d] |
 (no arg)

Shift selected 3d objects with specified displacement vector,
 or merge them with specified 3d object, or merge all selected 3d
 objects together.
 (eq. to '-+3d').

Default values: 'ty=tz=0'.



Example 427 : `-sphere3d 10 -repeat 5 --add3d[-1] 10,{?(-10,10)},0
-color3d[-1] ${-RGB} -done -add3d`



Example 428 : `-repeat 20 -torus3d 15,2 -color3d[-1] ${-RGB} -mul3d[-1] 0.5,1
-if {$>%2} -rotate3d[-1] 0,1,0,90 -endif -add3d[-1] 70 -add3d -rotate3d[-1]
0,0,1,18 -done -double3d 0`

2.12.2 *-animate3d*

Arguments: `_width>0, _height>0, _angle_dx, _angle_dy, _angle_dz, _zoom_factor>=0, _filename`

Animate selected 3d objects in a window.

If argument 'filename' is provided, each frame of the animation is saved as a numbered filename.

Default values: 'width=640', 'height=480', 'angle_dx=0', 'angle_dy=1', 'angle_dz=0', 'zoom_factor=1' and 'filename=(undefined)'.

2.12.3 *-apply_camera3d*

Arguments: `pos_x, pos_y, pos_z, target_x, target_y, target_z, up_x, up_y, up_z`

Apply 3d camera matrix to selected 3d objects.

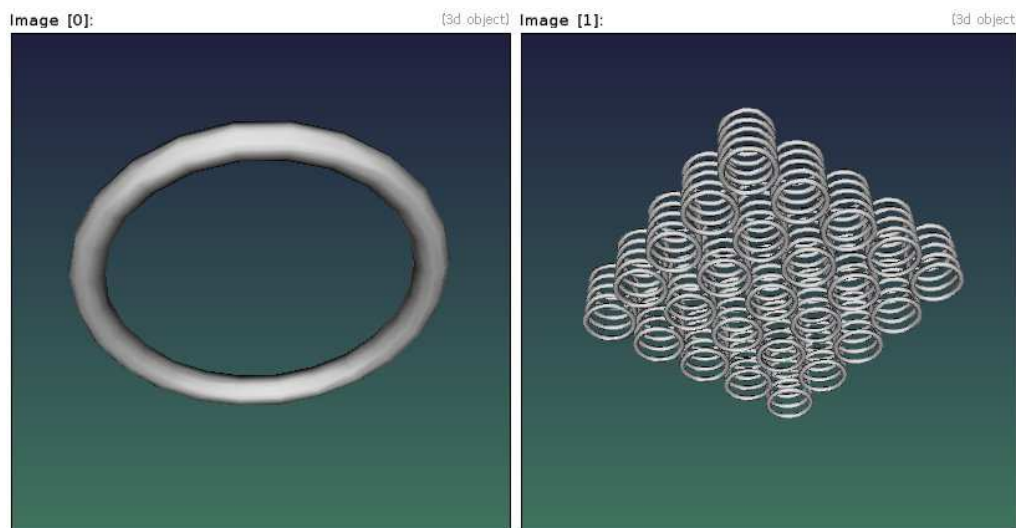
Default values: 'target_x=0', 'target_y=0', 'target_z=0', 'up_x=0', 'up_y=-1' and 'up_z=0'.

2.12.4 *-array3d*

Arguments: `size_x>=1, _size_y>=1, _size_z>=1, _offset_x[%], _offset_y[%], _offset_z[%]`

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 429 : `-torus3d 10,1 --array3d 5,5,5,110%,110%,300%`

2.12.5 *-arrow3d*

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

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

Default values: 'radius=5%', 'head.length=25%' and 'head.radius=15%'.



Example 430 : `-repeat 10 a={>2*pi/10} -arrow3d
0,0,0,{cos($a)},{sin($a)},-0.5 -done -+3d`

2.12.6 *-axes3d*

Arguments: `_size_x, _size_y, _size_z, _font.size>0, _label_x, _label_y, _label_z`

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' and 'label_z=Z'.



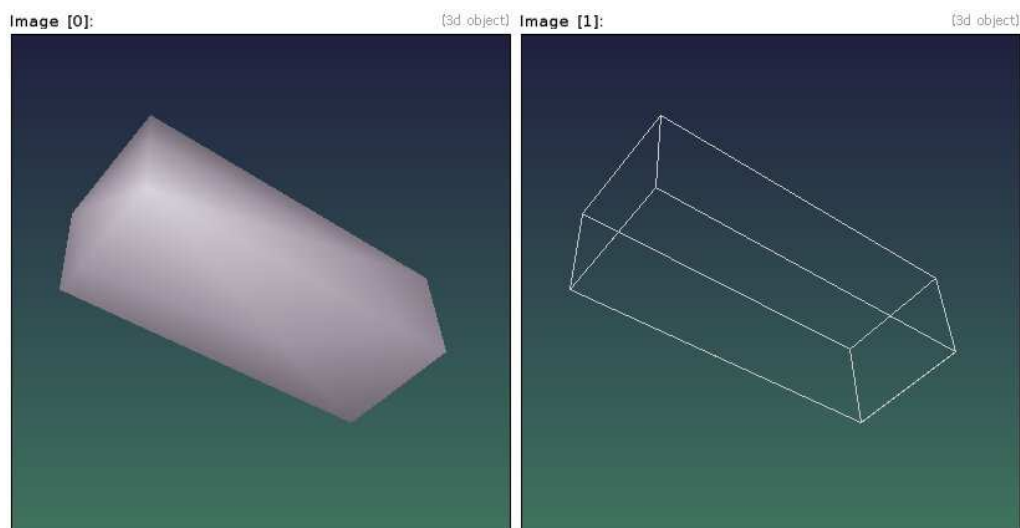
Example 431 : `-axes3d ,`

2.12.7 `-box3d`

Arguments: `_size_x, _size_y, _size_z`

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

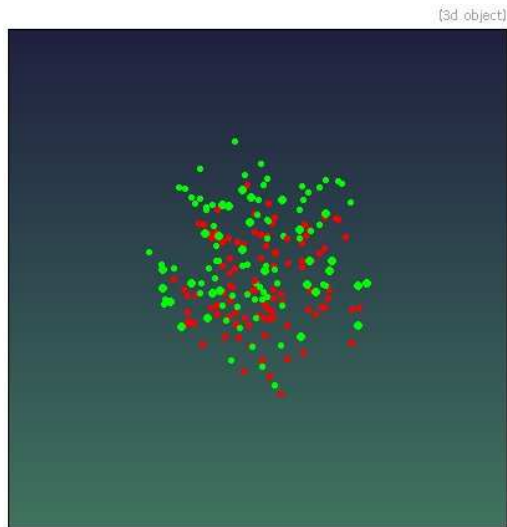
Default values: `'size_x=1'` and `'size_z=size_y=size_x'`.



Example 432 : `-box3d 100,40,30 --primitives3d 1 -color3d[-2] ${-RGB}`

2.12.8 *-center3d*

Center selected 3d objects at $(0,0,0)$.
(eq. to *'-c3d'*).



Example 433 : `-repeat 100 -circle3d {?(100)},{?(100)},{?(100)},2 -done -add3d
-color3d[-1] 255,0,0 --center3d -color3d[-1] 0,255,0 -add3d`

2.12.9 *-circle3d*

Arguments: `_x0,_y0,_z0,_radius>=0`

Input 3d circle at specified coordinates.

Default values: `'x0=y0=z0=0'` and `'radius=1'`.



Example 434 : `-repeat 500 a={>*pi/250} -circle3d
{cos(3*$a)},{sin(2*$a)},0,{a/50} -color3d[-1] ${-RGB},0.4 -done -add3d`

2.12.10 *-circles3d*

Arguments: `_radius>=0, _is_filled={ 0 | 1 }`

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

Default values: `'radius=1'` and `'is_filled=1'`.



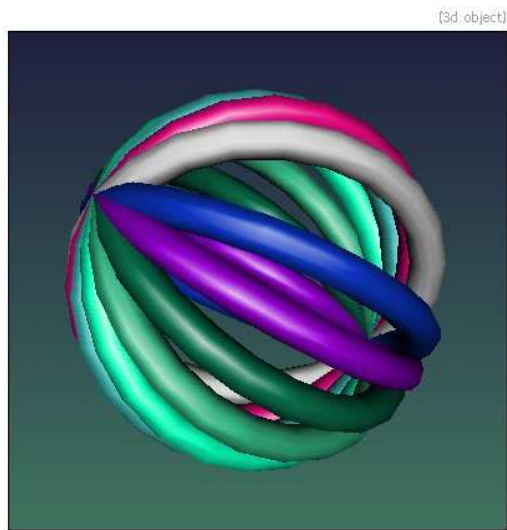
Example 435 : `image.jpg -luminance -resize2dy 40 -threshold 50% -* 255
-pointcloud3d -color3d[-1] 255,255,255 -circles3d 0.7`

2.12.11 *-color3d (+)*

Arguments: `R, _G, _B, _opacity`

Set color and opacity of selected 3d objects.
(eq. to `'-col3d'`).

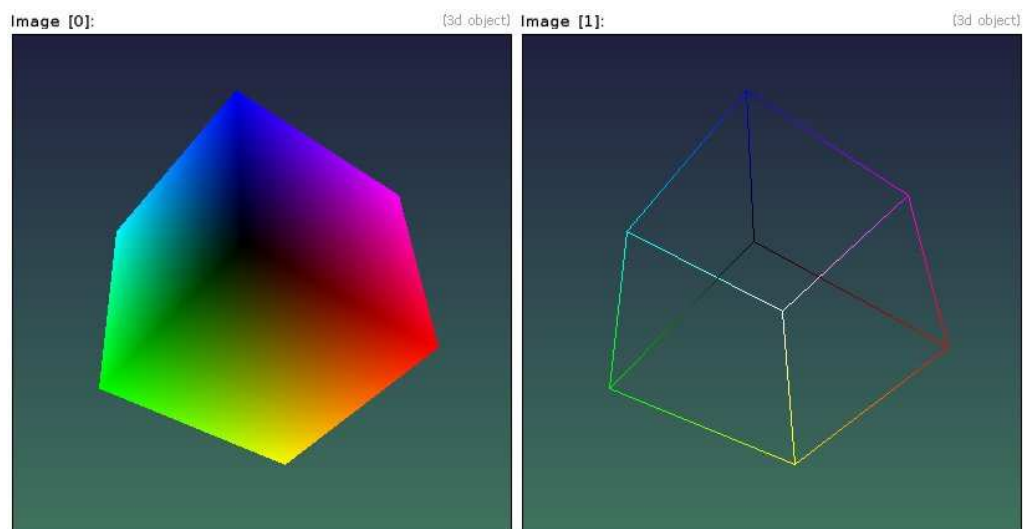
Default value: `'B=G=R'` and `'opacity=(undefined)'`.



Example 436 : `-torus3d 100,10 -double3d 0 -repeat 7 --rotate3d[-1] 1,0,0,20
-color3d[-1] ${-RGB} -done -add3d`

2.12.12 *-colorcube3d*

Input 3d color cube.



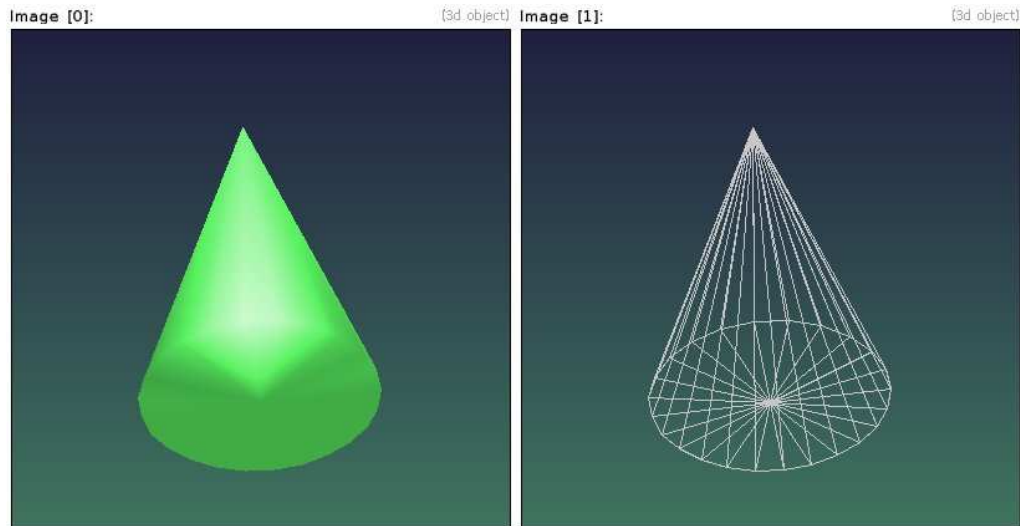
Example 437 : `-colorcube3d -mode3d 2 --primitives3d 1`

2.12.13 *-cone3d*

Arguments: `_radius, _height, _nb-subdivisions>0`

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

Default value: `'radius=1','height=1'` and `'nb_subdivisions=24'`.



Example 438 : `-cone3d 10,40 --primitives3d 1 -color3d[-2] ${-RGB}`

2.12.14 *-cubes3d*

Arguments: `_size>=0`

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

Default value: `'size=1'`.

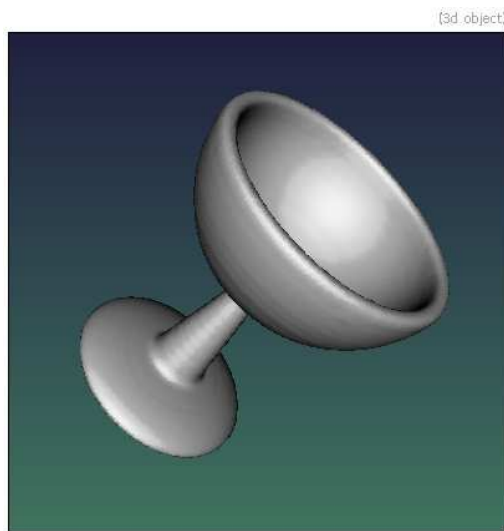


Example 439 : `image.jpg -luminance -resize2dy 40 -threshold 50% -* 255
-pointcloud3d -color3d[-1] 255,255,255 -cubes3d 1`

2.12.15 *-cup3d*

Arguments: `_resolution>0`

Input 3d cup object.



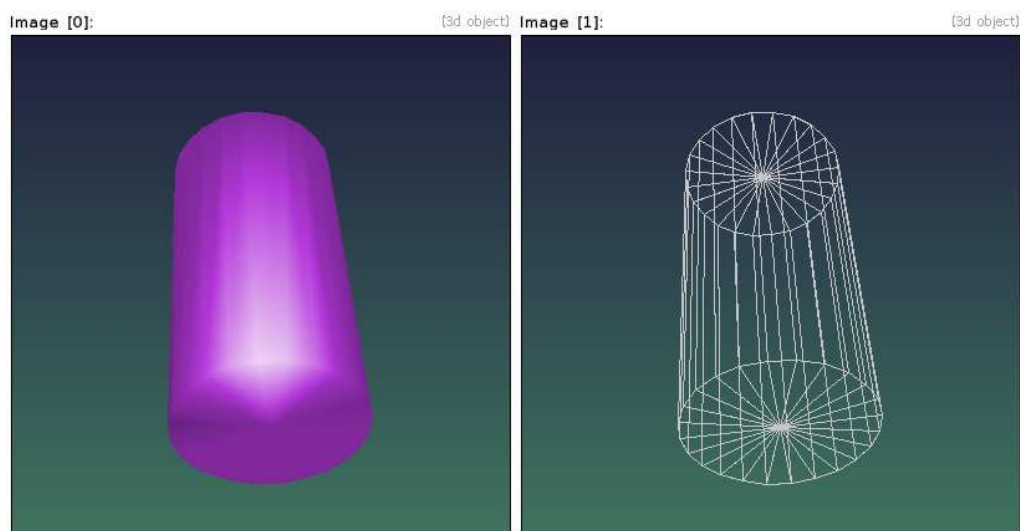
Example 440 : `-cup3d ,`

2.12.16 *-cylinder3d*

Arguments: `_radius,height,nb_subdivisions>0`

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

Default value: `'radius=1','height=1' and 'nb.subdivisions=24'.`



Example 441 : `-cylinder3d 10,40 --primitives3d 1 -color3d[-2] ${-RGB}`

2.12.17 *-delaunay3d*

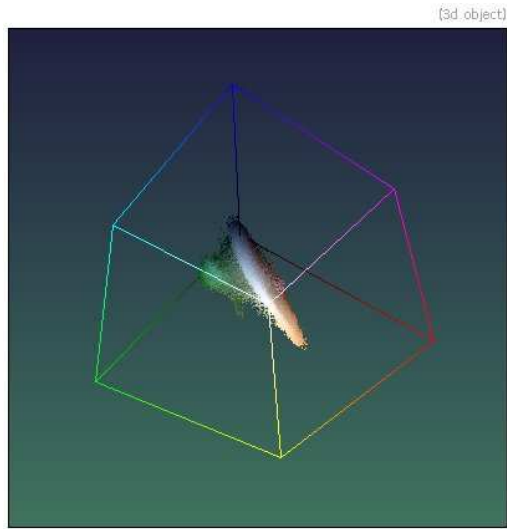
Generate 3d delaunay triangulations from selected images. One assumes that the selected input images are binary images containing the set of points to mesh. The output 3d object is a mesh composed of non-oriented triangles.



Example 442 : `500,500 -noise 0.05,2 -* 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]`

2.12.18 *-distribution3d*

Get 3d color distribution of selected images.



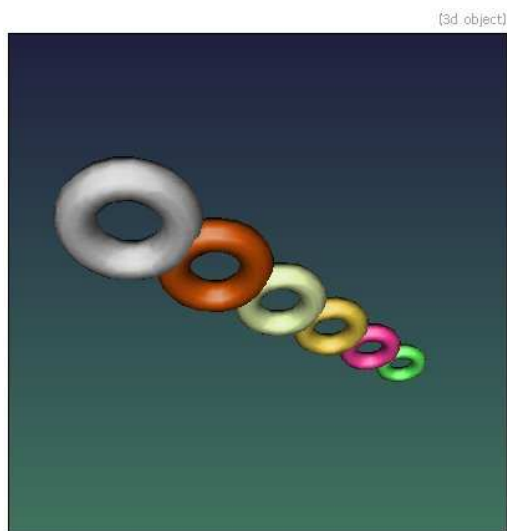
Example 443 : `image.jpg -distribution3d -colorcube3d -primitives3d[-1] 1
-add3d`

2.12.19 *-div3d (+)*

Arguments: `factor` |
`factor_x, factor_y, _factor_z`

Scale selected 3d objects isotropically or anisotropically, with the inverse of specified factors.
(eq. to `'-/3d'`).

Default value: `'factor_z=0'`.



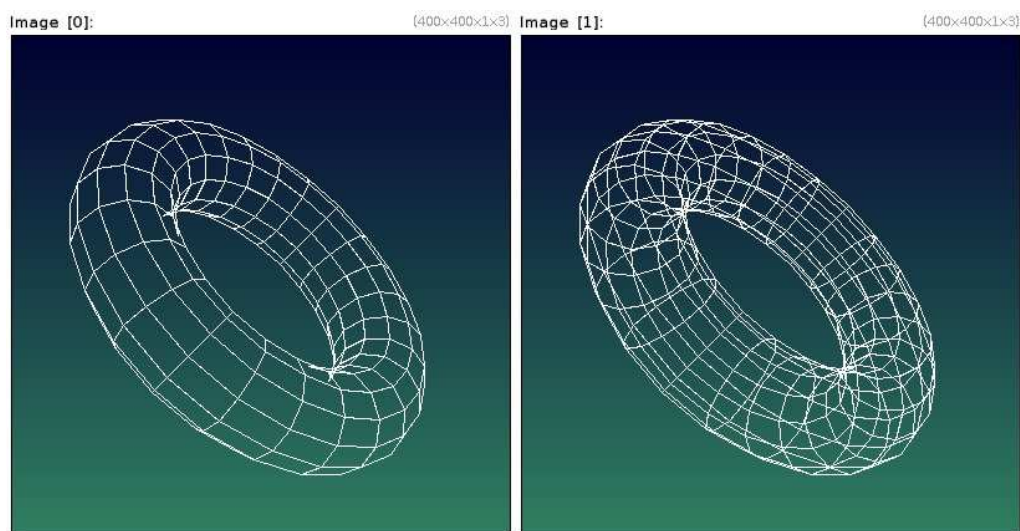
Example 444 : `-torus3d 5,2 -repeat 5 --add3d[-1] 12,0,0 -div3d[-1] 1.2
-color3d[-1] ${-RGB} -done -add3d`

2.12.20 *-double3d (+)*

Arguments: `_is_double_sided={ 0 | 1 }`

Enable/disable double-sided mode for 3d rendering.
(eq. to `'-db3d'`).

Default value: `'is_double_sided=1'`.



Example 445 : `-mode3d 1 -repeat 2 -torus3d 100,30 -rotate3d[-1] 1,1,0,60
-double3d $> -snapshot3d[-1] 400 -done`

2.12.21 *-elevation3d (+)*

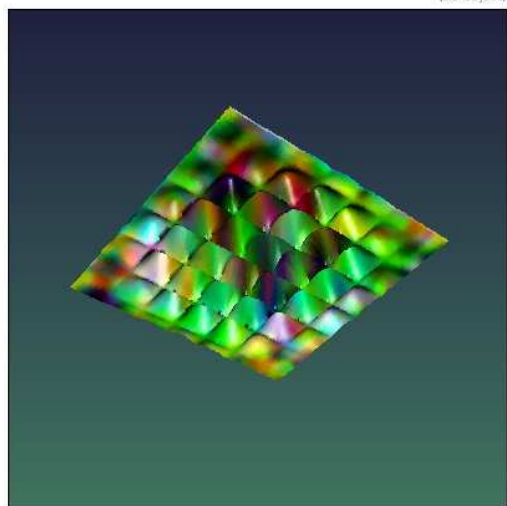
Arguments: `z-factor |
[elevation_map] |
'formula' |
(no arg)`

Build 3d elevation of selected images, 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.



Example 446 : `image.jpg -blur 5 -elevation3d 0.5`



Example 447 : `128,128,1,3,?(255) -plasma 10,3 -blur 4 -sharpen 10000
-elevation3d[-1]
'X=(x-64)/6;Y=(y-64)/6;100*exp(-(X^2+Y^2)/30)*abs(cos(X)*sin(Y))'`

2.12.22 *-empty3d*

Input empty 3d object.



Example 448 : `-empty3d`

2.12.23 *-extrude3d*

Arguments: `_depth>0, _resolution>0, _smoothness[%]>=0`

Generate extruded 3d object from selected binary XY-profiles.

Default values: `'depth=16', 'resolution=1024' and 'smoothness=0.5%'`.



Example 449 : `image.jpg -threshold 50% -extrude3d 16`

2.12.24 *-focale3d* (+)

Arguments: *focale*

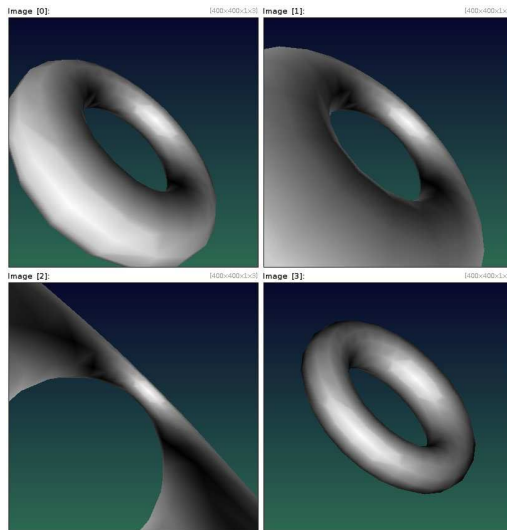
Set 3d *focale*.

(eq. to *'-f3d'*).

Set *'focale'* to 0 to enable parallel projection (instead of perspective).

Set negative *'focale'* will disable 3d sprite zooming.

Default value: *'focale=700'*.

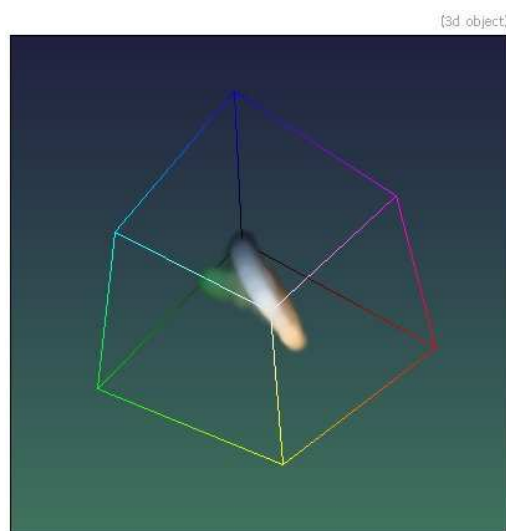


Example 450 : `-repeat 5 -torus3d 100,30 -rotate3d[-1] 1,1,0,60 -focale3d
{${<*90}} -snapshot3d[-1] 400 -done -remove[0]`

2.12.25 *-gaussians3d*

Arguments: *_size>0, _opacity*

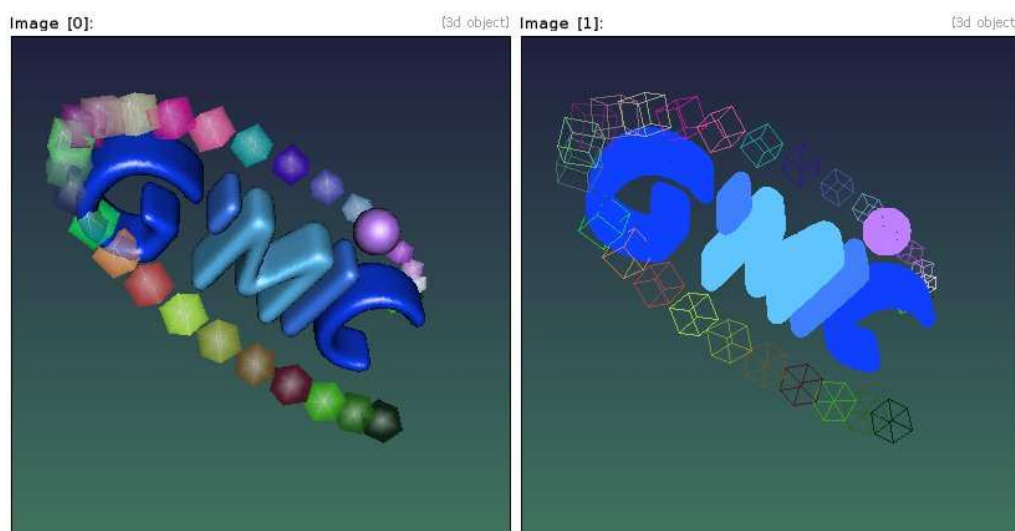
Convert selected 3d objects into set of 3d gaussian-shaped sprites.



Example 451 : `image.jpg -r2dy 32 -distribution3d -gaussians3d 20 -colorcube3d -primitives3d[-1] 1 -+3d`

2.12.26 -gmic3d

Input a 3d G'MIC logo.



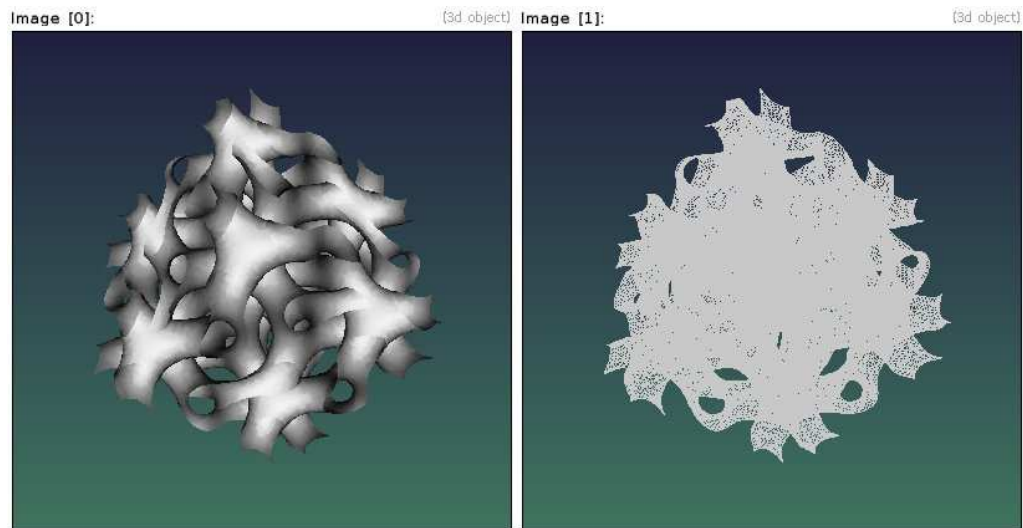
Example 452 : `-gmic3d --primitives3d 1`

2.12.27 -gyroid3d

Arguments: `_resolution>0, _zoom`

Input 3d gyroid at $(0,0,0)$, with specified resolution.

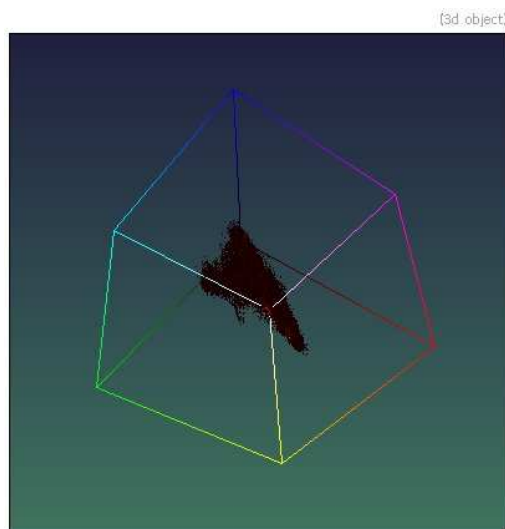
Default values: 'resolution=32' and 'zoom=5'.



Example 453 : `-gyroid3d 48 --primitives3d 1`

2.12.28 *-histogram3d*

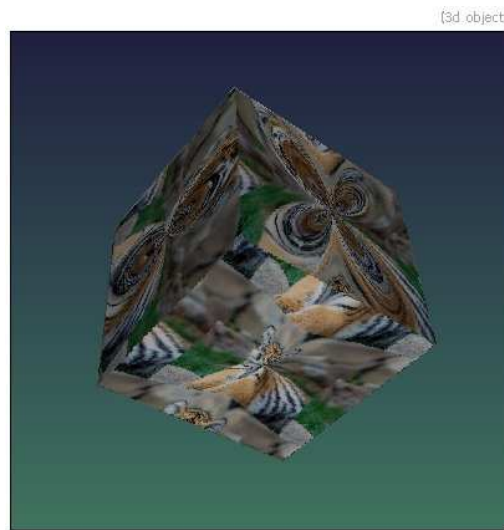
Get 3d color histogram of selected images.



Example 454 : `image.jpg -histogram3d -colorcube3d -primitives3d[-1] 1 -add3d`

2.12.29 *-image6cube3d*

Generate 3d mapped cubes from 6-sets of selected images.



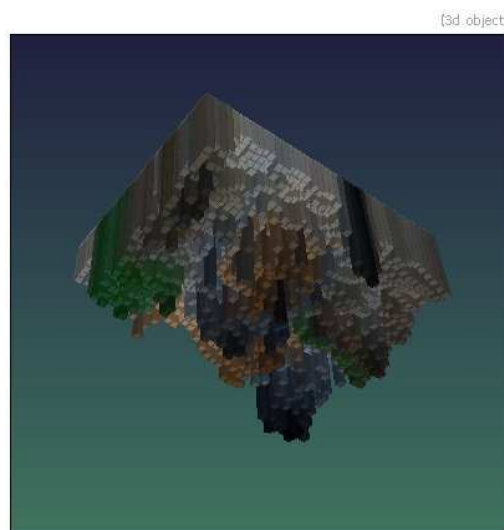
Example 455 : `image.jpg -animate flower,"30,0","30,5",6 -image6cube3d`

2.12.30 *-imageblocks3d*

Arguments: `_maximum_elevation, _smoothness[%]>=0`

Generate 3d blocks from selected images.
Transparency of selected images is taken into account.

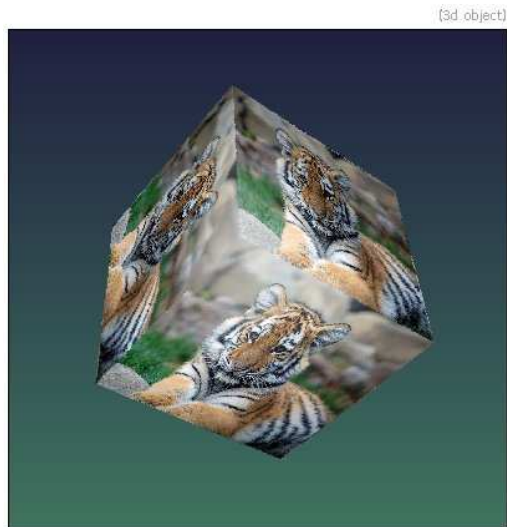
Default values: `'maximum_elevation=10'` and `'smoothness=0'`.



Example 456 : `image.jpg -resize2dy 32 -imageblocks3d -20 -m3d 3`

2.12.31 *-imagecube3d*

Generate 3d mapped cubes from selected images.



Example 457 : `image.jpg -imagecube3d`

2.12.32 *-imageplane3d*

Generate 3d mapped planes from selected images.



Example 458 : `image.jpg -imageplane3d`

2.12.33 *-imagepyramid3d*

Generate 3d mapped pyramids from selected images.



Example 459 : `image.jpg -imagepyramid3d`

2.12.34 *-imagerubik3d*

Arguments: `_xy_tiles>=1, 0<=xy_shift<=100, 0<=z_shift<=100`

Generate 3d mapped rubik's cubes from selected images.

Default values: `'xy_tiles=3', 'xy_shift=5' and 'z_shift=5'.`



Example 460 : `image.jpg -imagerubik3d ,`

2.12.35 *-imagesphere3d*

Arguments: `_resolution1>=3,_resolution2>=3`

Generate 3d mapped sphere from selected images.

Default values: `'resolution1=32'` and `'resolutions2=16'`.



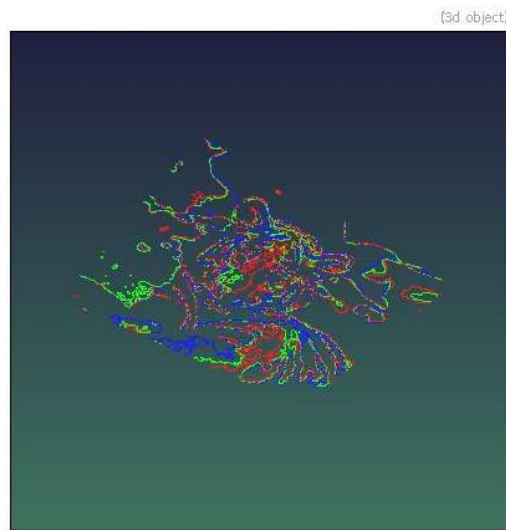
Example 461 : `image.jpg -imagesphere3d 32,16`

2.12.36 *-isoline3d (+)*

Arguments: `isovalue[%] |`
`'formula',value,_x0,_y0,_x1,_y1,_size_x>0[%],_size_y>0[%]`

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'`.



Example 462 : `image.jpg -blur 1 -isoline3d 50%`



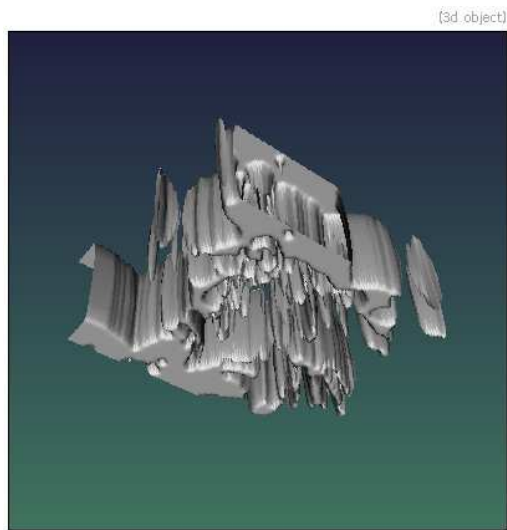
Example 463 : `-isoline3d 'X=x-w/2;Y=y-h/2;(X^2+Y^2)%20',10,-10,-10,10,10`

2.12.37 *-isosurface3d* (+)

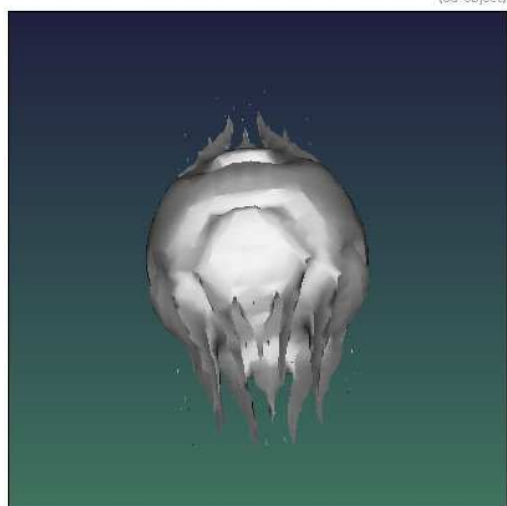
Arguments: `isovalue[%]` | `'formula',value,_x0,_y0,_z0,_x1,_y1,_z1,_size_x>0[%],_size_y>0[%],_size_z>0[%]`

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'`.



Example 464 : `image.jpg -resize2dy 128 -luminance -threshold 50% -expand_z 2,0
-blur 1 -isosurface3d 50% -mul3d 1,1,30`



Example 465 : `-isosurface3d 'x^2+y^2+abs(z)^abs(4*cos(x*y*z*3))',3`

2.12.38 *-label3d*

Arguments: `"text", font_height>=0, _opacity, _color1, ...`

Generate 3d text label.

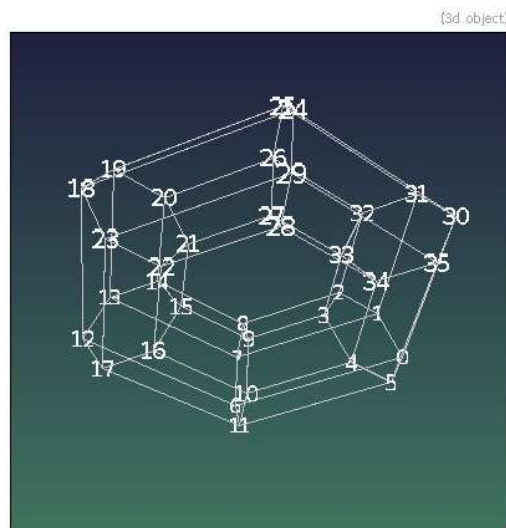
Default values: `'font_height=13', 'opacity=1' and
'color=255,255,255'.`

2.12.39 -label points3d

Arguments: `_label_size>0, _opacity`

Add a numbered label to all vertices of selected 3d objects.

Default values: `'label_size=13'` and `'opacity=0.8'`.



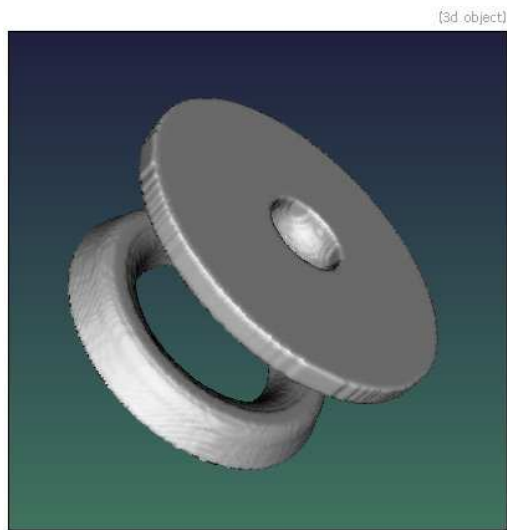
Example 466 : `-torus3d 100,40,6,6 -label.points3d 23,1 -mode3d 1`

2.12.40 -lathe3d

Arguments: `_resolution>0, _smoothness[%]>=0, _max_angle>=0`

Generate 3d object from selected binary XY-profiles.

Default values: `'resolution=128'`, `'smoothness=0.5%'` and `'max_angle=361'`.



Example 467 : `300,300 -rand -1,1 -blur 40 -sign -normalize 0,255 -lathe3d ,`

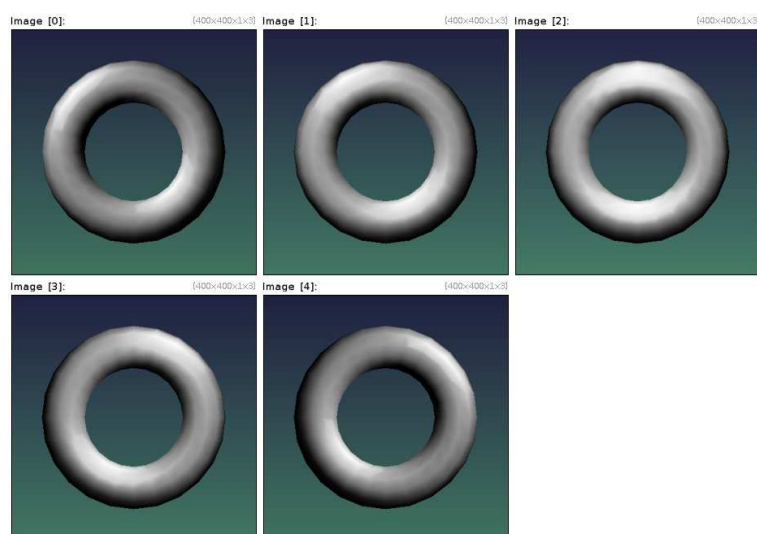
2.12.41 *-light3d (+)*

Arguments: `position_x,position_y,position_z` |
 `[texture]` |
 `(no arg)`

Set the light coordinates or the light texture for 3d rendering.

(eq. to `'-l3d'`).

(no arg) resets the 3d light to default.

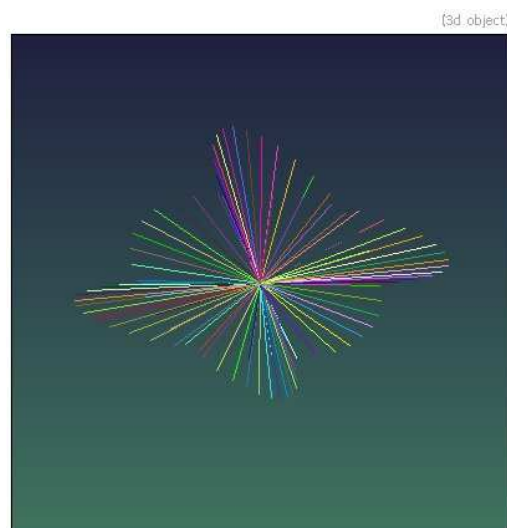


Example 468 : `-torus3d 100,30 -double3d 0 -specs3d 1.2 -repeat 5 -light3d
{>*100},0,-300 --snapshot3d[0] 400 -done -remove[0]`

2.12.42 *-line3d*

Arguments: `x0,y0,z0,x1,y1,z1`

Input 3d line at specified coordinates.



Example 469 : `-repeat 100 a={>*pi/50} -line3d 0,0,0,{cos(3*$a)},{sin(2*$a)},0
-color3d[-1] ${-RGB} -done -add3d`

2.12.43 *-lissajous3d*

Arguments: `resolution>1,a,A,b,B,c,C`

Input 3d lissajous curves $(x(t)=\sin(a*t+A*2*\pi), y(t)=\sin(b*t+B*2*\pi), z(t)=\sin(c*$

Default values: `'resolution=1024', 'a=2', 'A=0', 'b=1', 'B=0', 'c=0'`
and `'C=0'`.



Example 470 : `-lissajous3d ,`

2.12.44 *-mode3d (+)*

Arguments: `_mode`

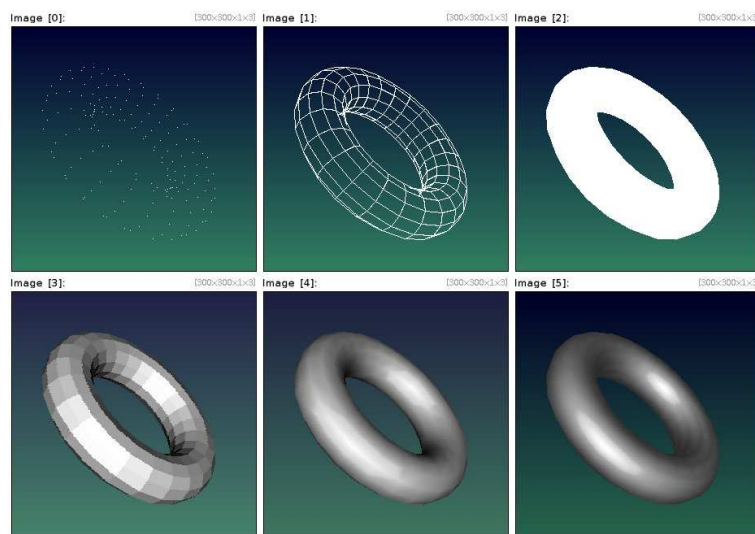
Set static 3d rendering mode.

(eq. to `'-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 value: `'mode=4'`.



Example 471 : `(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]`

2.12.45 *-moded3d* (+)

Arguments: `_mode`

Set dynamic 3d rendering mode for interactive 3d viewer.
(eq. to `'-md3d'`).

'mode' can be { `-1=bounding-box` | `0=dots` | `1=wireframe`
| `2=flat` | `3=flat-shaded` | `4=gouraud-shaded`
| `5=phong-shaded` }.

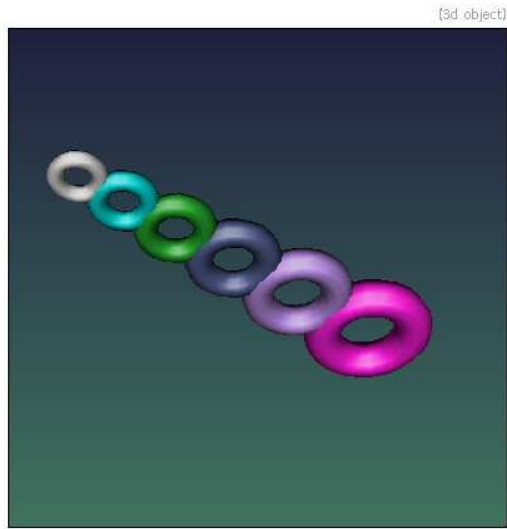
Default value: `'mode=-1'`.

2.12.46 *-mul3d* (+)

Arguments: `factor` |
`factor_x, factor_y, _factor_z`

Scale selected 3d objects isotropically or anisotropically, with
specified factors.
(eq. to `'-*3d'`).

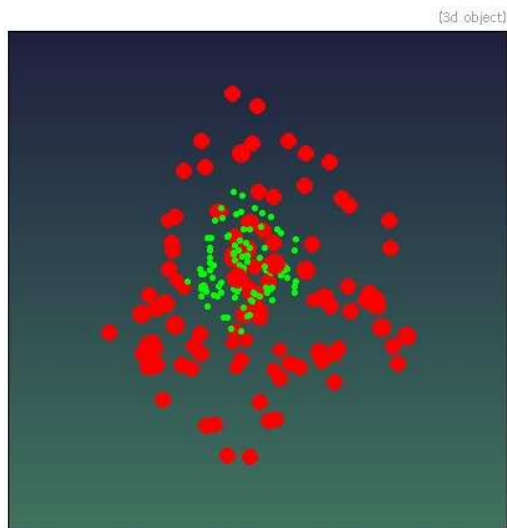
Default value: `'factor_z=0'`.



Example 472 : `-torus3d 5,2 -repeat 5 --add3d[-1] 10,0,0 -mul3d[-1] 1.2
-color3d[-1] ${-RGB} -done -add3d`

2.12.47 *-normalize3d*

Normalize selected 3d objects to unit size.
(eq. to `'-n3d'`).



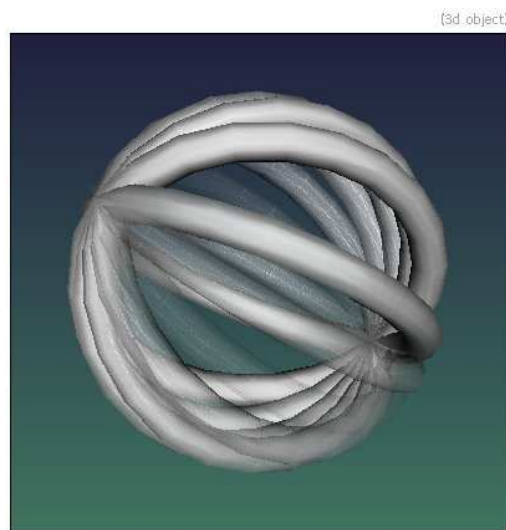
Example 473 : `-repeat 100 -circle3d {?(3)},{?(3)},{?(3)},0.1 -done -add3d
-color3d[-1] 255,0,0 --normalize3d[-1] -color3d[-1] 0,255,0 -add3d`

2.12.48 *-opacity3d (+)*

Arguments: `_opacity`

Set opacity of selected 3d objects.
(eq. to '-o3d').

Default value: 'opacity=1'.



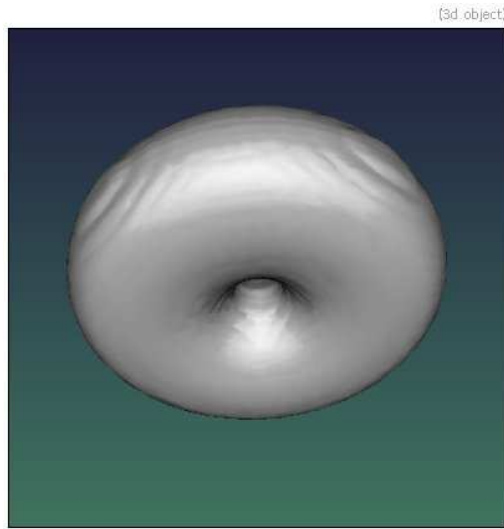
Example 474 : `-torus3d 100,10 -double3d 0 -repeat 7 --rotate3d[-1] 1,0,0,20
-opacity3d[-1] {?} -done -add3d`

2.12.49 *-parametric3d*

Arguments: `_x(a,b), _y(a,b), _z(a,b), _amin, _amax, _bmin, _bmax, _res_a>0, _res_b>0, _res_x>0, _res_y>0, _res_z>0, _smoothness>=0, _isovalue>=0`

Input 3d object from specified parametric surface
(`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%'.`



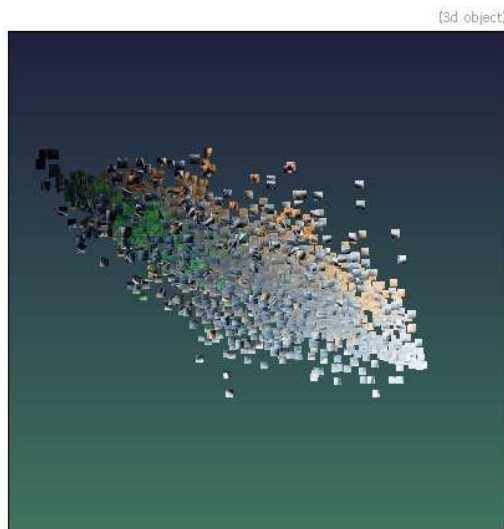
Example 475 : `-parametric3d` ,

2.12.50 `-pca_patch3d`

Arguments: `_patch_size>0, _M>0, _N>0, _normalize_input={ 0 | 1 }, _normalize_output={ 0 | 1 }, _lambda_xy`

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'`.



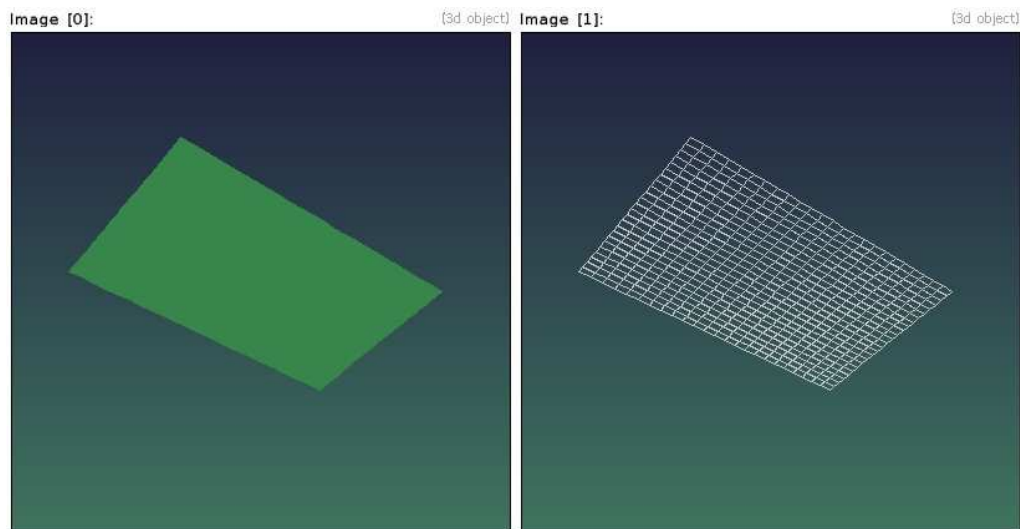
Example 476 : `image.jpg -pca_patch3d 7`

2.12.51 *-plane3d*

Arguments: `_size_x, _size_y, _nb_subdivisions_x>0, _nb_subdivisions_y>0`

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 477 : `-plane3d 50,30 --primitives3d 1 -color3d[-2] ${-RGB}`

2.12.52 *-point3d*

Arguments: `x0,y0,z0`

Input 3d point at specified coordinates.



Example 478 : `-repeat 1000 a={>pi/500} -point3d {cos(3*$a)},{sin(2*$a)},0
-color3d[-1] ${-RGB} -done -add3d`

2.12.53 *-pointcloud3d*

Convert selected planar or volumetric images to 3d point clouds.

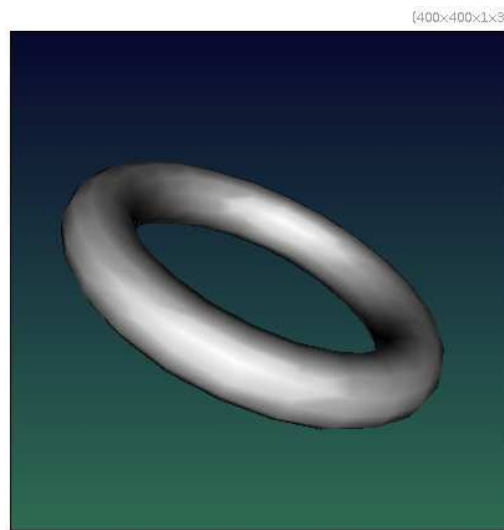


Example 479 : `image.jpg -luminance -resize2dy 100 -threshold 50% -* 255
-pointcloud3d -color3d[-1] 255,255,255`

2.12.54 *-pose3d*

Arguments: `p1,...,p12`

Apply 3d pose matrix to selected 3d objects.



Example 480 : `-torus3d 100,20 -pose3d
0.152437,1.20666,-0.546366,0,-0.535962,0.559129,1.08531,0,1.21132,0.0955431,0.548966,0,0,0,-206,1
-snapshot3d 400`

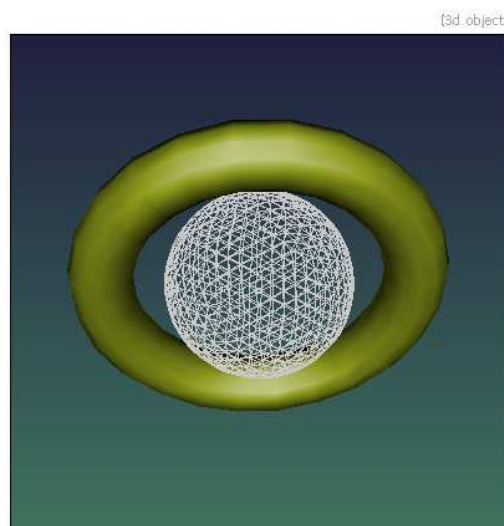
2.12.55 *-primitives3d (+)*

Arguments: `mode`

Convert primitives of selected 3d objects.

(eq. to `'-p3d'`).

'mode' can be { 0=points | 1=segments | 2=non-textured }.



Example 481 : `-sphere3d 30 -primitives3d 1 -torus3d 50,10 -color3d[-1]
${-RGB} -add3d`

2.12.56 -projections3d

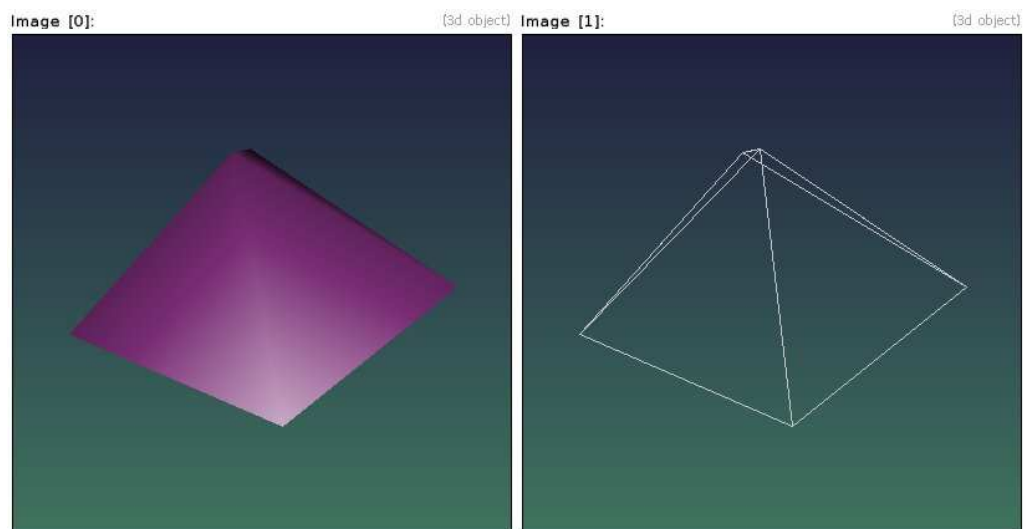
Arguments: `_x[%],_y[%],_z[%],_is_bounding_box={ 0 | 1 }`

Generate 3d xy,xz,yz projection planes from specified volumetric images.

2.12.57 -pyramid3d

Arguments: `width,height`

Input 3d pyramid at (0,0,0), with specified geometry.

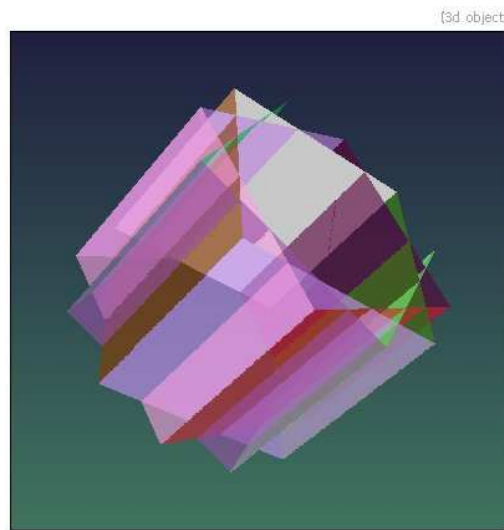


Example 482 : `-pyramid3d 100,100 --primitives3d 1 -color3d[-2] ${-RGB}`

2.12.58 -quadrangle3d

Arguments: `x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3`

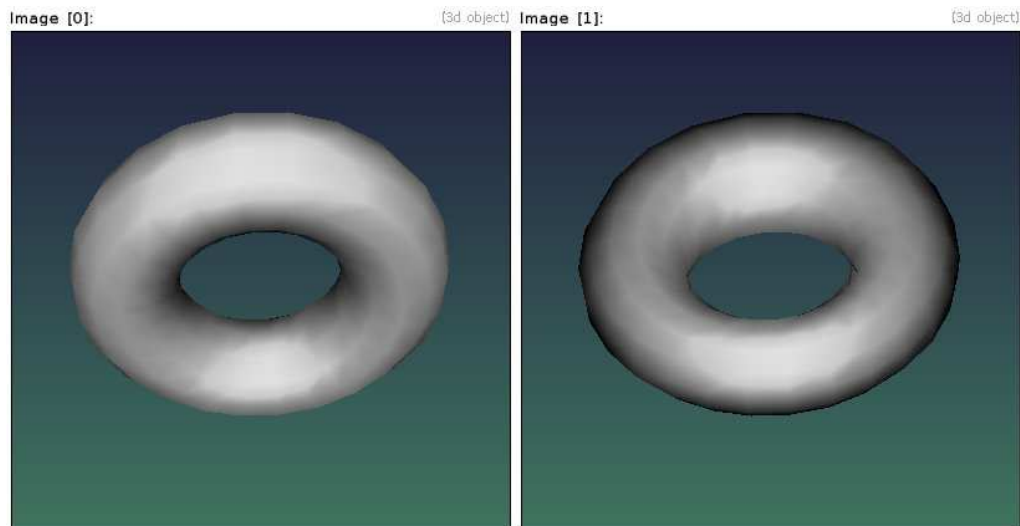
Input 3d quadrangle at specified coordinates.



Example 483 : `-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`

2.12.59 *-reverse3d (+)*

Reverse primitive orientations of selected 3d objects.
(eq. to `'-rv3d'`).



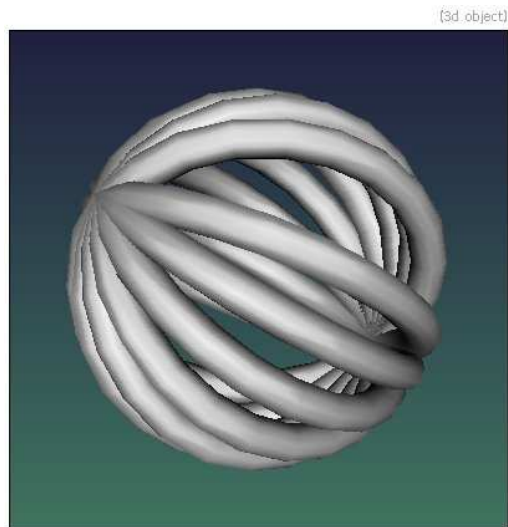
Example 484 : `-torus3d 100,40 -double3d 0 --reverse3d`

2.12.60 *-rotate3d (+)*

Arguments: `u,v,w,angle`

Rotate selected 3d objects around specified axis with specified

angle (in deg.).
(eq. to '-r3d').

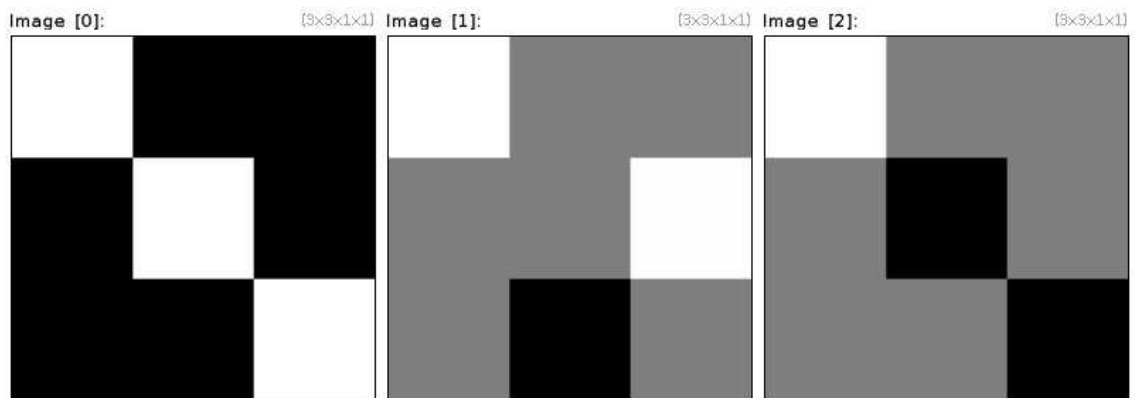


Example 485 : `-torus3d 100,10 -double3d 0 -repeat 7 --rotate3d[-1] 1,0,0,20
-done -add3d`

2.12.61 *-rotation3d*

Arguments: `u,v,w,angle`

Input 3x3 rotation matrix with specified axis and angle (in deg).

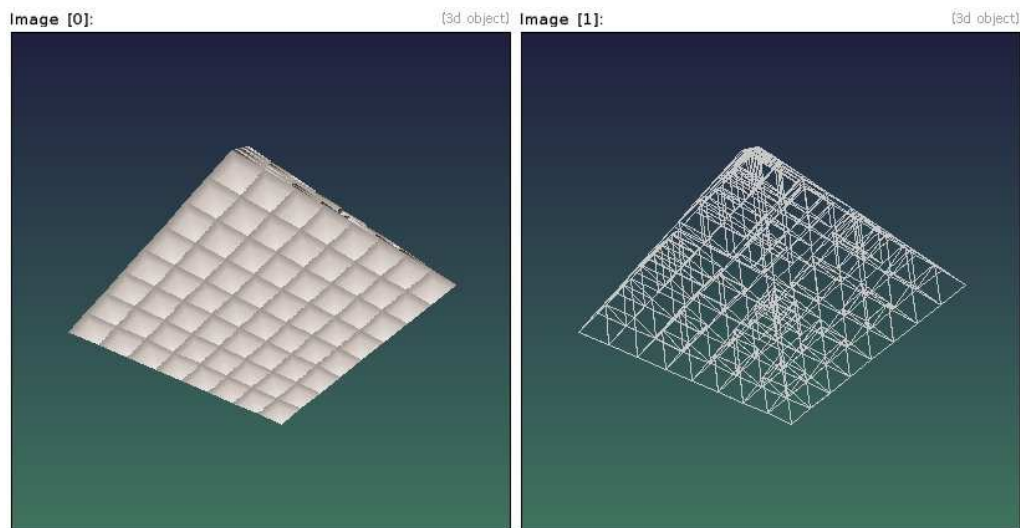


Example 486 : `-rotation3d 1,0,0,0 -rotation3d 1,0,0,90 -rotation3d 1,0,0,180`

2.12.62 *-sierpinski3d*

Arguments: `_recursion_level>=0, _width, _height`

Input 3d Sierpinski pyramid.



Example 487 : `-sierpinski3d 3 --primitives3d 1 -color3d[-2] ${-RGB}`

2.12.63 *-size3d*

Return bounding box size of the last selected 3d object.

2.12.64 *-skeleton3d*

Arguments: `_metric, _frame_type={ 0=squares | 1=diamonds
| 2=circles | 3=auto }, _skeleton_opacity, _frame_opacity, _is_frame_wireframe=
={ 0 | 1 }`

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 488 : `-cupid 480 --skeleton3d ,`

2.12.65 *-snapshot3d*

Arguments: `_size>0, _zoom>=0, _backgroundR, _backgroundG, _backgroundB, _backgroundA` | `[background_image], zoom>=0`

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)'`.



Example 489 : `-torus3d 100,20 -rotate3d 1,1,0,60 -snapshot3d
400,1.2,128,64,32`



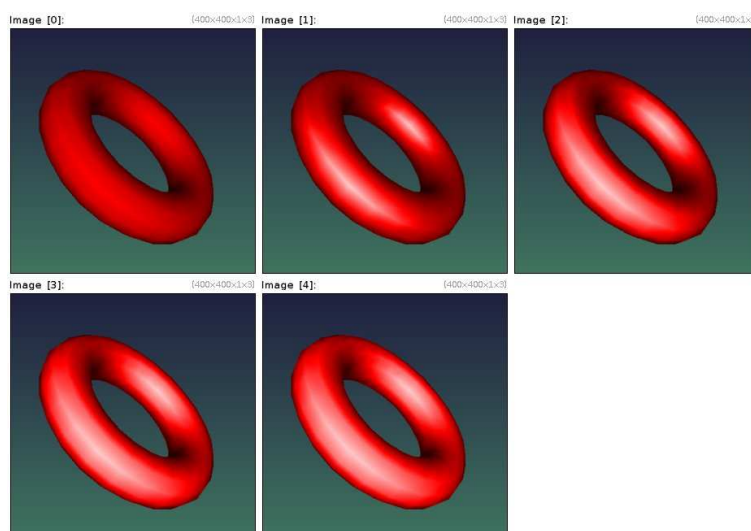
Example 490 : `-torus3d 100,20 -rotate3d 1,1,0,60 -testimage2d 400
--snapshot3d[0] [1],1.2`

2.12.66 `-spec13d (+)`

Arguments: `value` ≥ 0

Set lightness of 3d specular light.
(eq. to `'-sl3d'`).

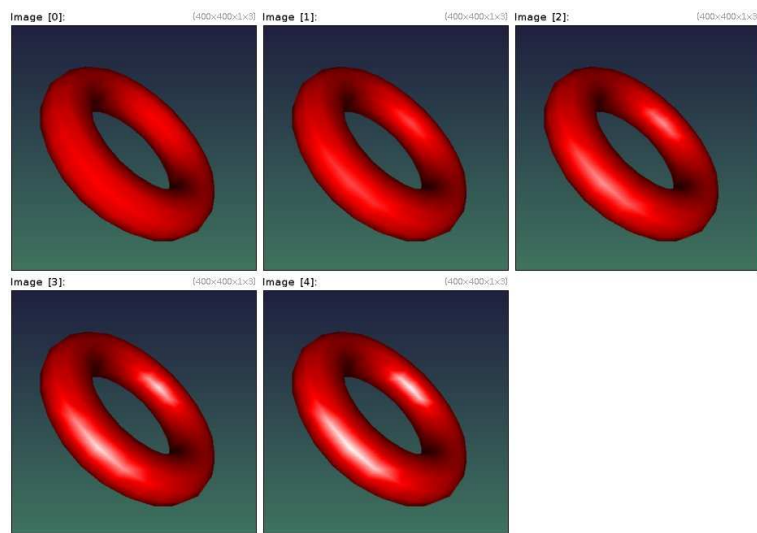
Default value: `'value=0.15'`.



Example 491 : `(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 -spec13d {0,[$>]} -snapshot3d[-1] 400 -done
-remove[0]`

2.12.67 -specs3d (+)**Arguments:** `value` ≥ 0

Set shininess of 3d specular light.
(eq. to `'-ss3d'`).

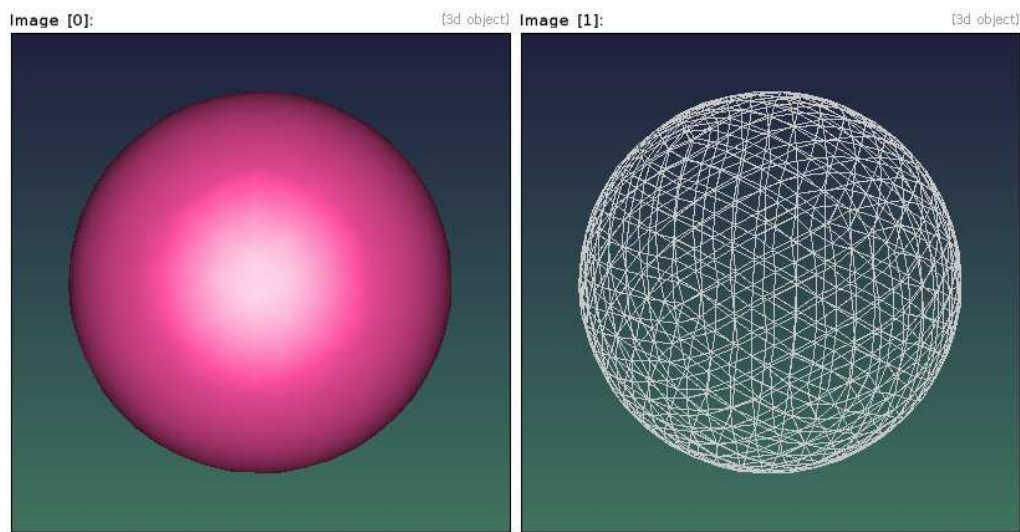
Default value: `'value=0.8'`.

Example 492 : `(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]`

2.12.68 -sphere3d (+)**Arguments:** `radius, nb recursions` ≥ 0

Input 3d sphere at `(0,0,0)`, with specified geometry.

Default value: `'nb recursions=3'`.



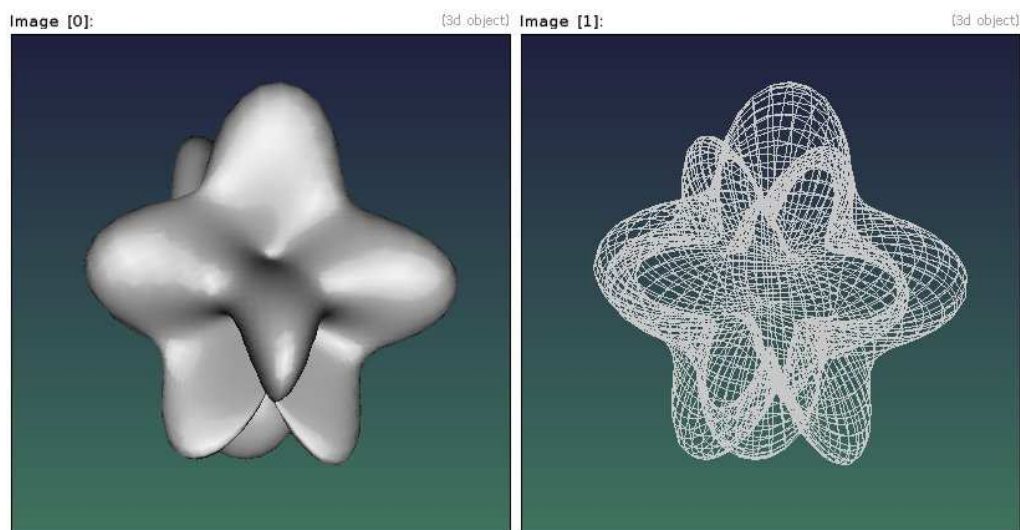
Example 493 : `-sphere3d 100 --primitives3d 1 -color3d[-2] ${-RGB}`

2.12.69 *-spherical3d*

Arguments: `_nb_azimuth>=3, _nb_zenith>=3, _radius_function(phi,theta)`

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(3*theta)"'`



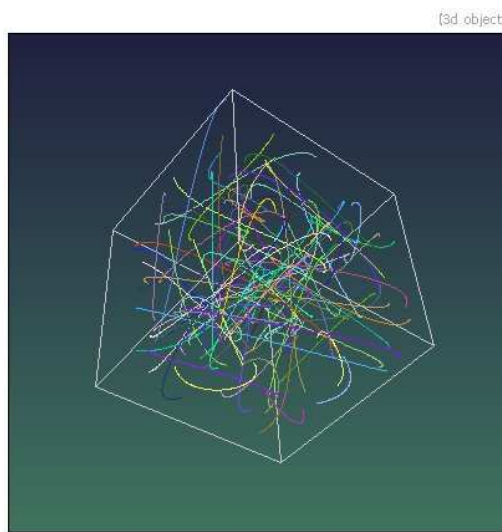
Example 494 : `-spherical3d 64 --primitives3d 1`

2.12.70 -spline3d

Arguments: `x0[%],y0[%],z0[%],u0[%],v0[%],w0[%],x1[%],y1[%],z1[%],u1[%],-v1[%],w1[%],_nb.vertices>=2`

Input 3d spline with specified geometry.

Default values: `'nb.vertices=128'`.



Example 495 : `-repeat 100 -spline3d
{?},{?},{?},{?},{?},{?},{?},{?},{?},{?},{?},{?},{?},128 -color3d[-1] ${-RGB} -done
-box3d 1 -primitives3d[-1] 1 -+3d`

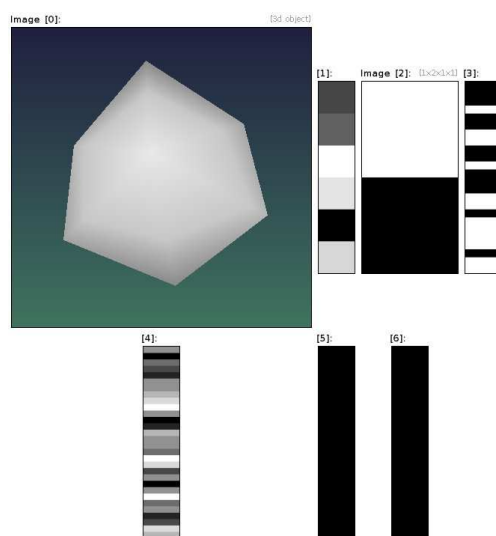
2.12.71 -split3d (+)

Arguments: `_keep_shared.data={ 0 | 1 }`

Split selected 3d objects into 6 feature vectors : { header, sizes, vertices, primitives, colors, opacities }.
(eq. to '-s3d').

To recreate the 3d object, append these 6 images along the y-axis.

Default value: `'keep_shared.data=1'`.



Example 496 : `-box3d 100 --split3d`

2.12.72 *-sprite3d*

Convert selected images as 3d sprites.
Selected image with alpha channels are managed.



Example 497 : `image.jpg -sprite3d`

2.12.73 *-sprites3d*

Convert selected 3d objects as sprites clouds, where the specified 2d sprite is the last selected image.
If the selected sprite has a 4th channel, it stands for the sprite alpha-channel (in [0,255]).



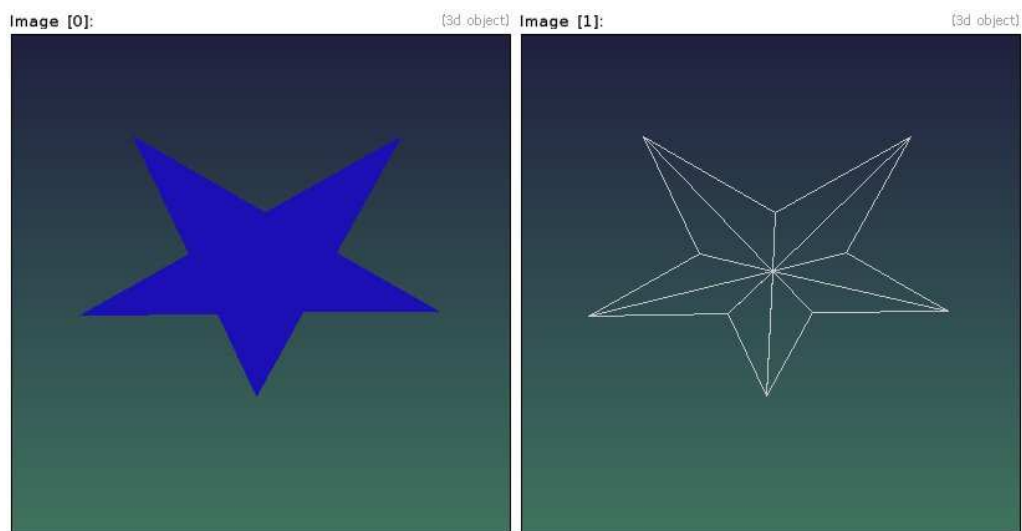
Example 498 : `-torus3d 100,20 image.jpg -resize2dy[-1] 64 100%,100%
-gaussian[-1] 30%,30% -*[-1] 255 -append[-2,-1] c --sprites3d -display_rgba[-2]`

2.12.74 *-star3d*

Arguments: `_nb_branches>0, 0<=_thickness<=1`

Input 3d star at $(0,0,0)$, with specified geometry.

Default values: `'nb_branches=5'` and `'thickness=0.38'`.



Example 499 : `-star3d , --primitives3d 1 -color3d[-2] ${-RGB}`

2.12.75 *-streamline3d (+)*

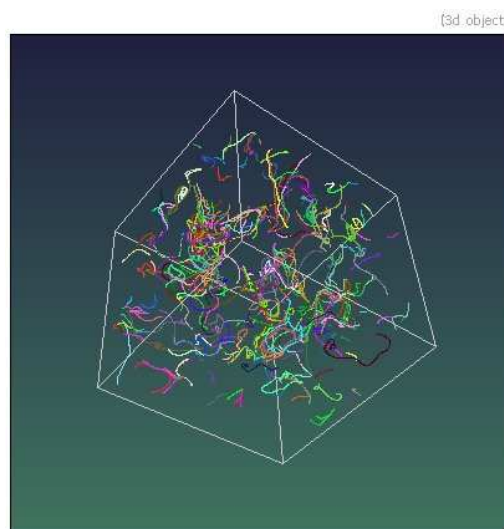
Arguments: `x[%], y[%], z[%], _L>=0, _dl>0, _interpolation, _is_backward={`

```
0 | 1 },_is_oriented={ 0 | 1 } |
    'formula',x,y,z,_L>=0,_dl>0,_interpolation,_is_backward={ 0
| 1 },_is_oriented={ 0 | 1 }
```

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'.



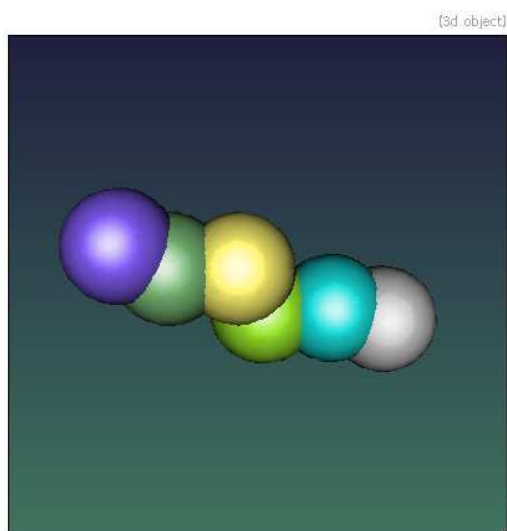
```
Example 500 : 100,100,100,3 -rand -10,10 -blur 3 -repeat 300
--streamline3d[0] {?(100)},{?(100)},{?(100)},1000,1,1 -color3d[-1] ${-RGB}
-done -remove[0] -box3d 100 -primitives3d[-1] 1 -add3d
```

2.12.76 -sub3d (+)

Arguments: tx,ty,tz

Shift selected 3d objects with the opposite of specified displacement vector.
(eq. to '--3d').

Default values: 'ty=tz=0'.



Example 501 : `-sphere3d 10 -repeat 5 --sub3d[-1] 10,{?(-10,10)},0
-color3d[-1] ${-RGB} -done -add3d`

2.12.77 *-superformula3d*

Arguments: `resolution>1,m>=1,n1,n2,n3`

Input 2d superformula curve as a 3d object.

Default values: `'resolution=1024', 'm=8', 'n1=1', 'n2=5' and 'n3=8'.`



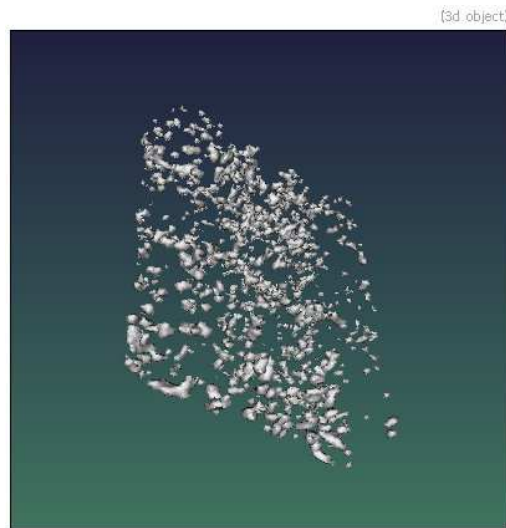
Example 502 : `-superformula3d ,`

2.12.78 *-text pointcloud3d*

Arguments: `_"text1",_"text2",_smoothness`

Input 3d text pointcloud from the two specified strings.

Default values: `'text1="text1"', 'text2="text2"'` and `'smoothness=1'`.



Example 503 : `-text.pointcloud3d "G'MIC","Rocks!"`

2.12.79 *-text3d*

Arguments: `text,_font_height>0,_depth>0,_smoothness`

Input a 3d text object from specified text.

Default values: `'font_height=53', 'depth=10'` and `'smoothness=1.5'`.



Example 504 : `-text3d "G'MIC as a\n3D logo!"`

2.12.80 `-texturize3d` (+)

Arguments: `[ind_texture],-[ind_coords]`

Texturize selected 3d objects with specified texture and coordinates.

(eq. to `'-t3d'`).

When `'[ind_coords]'` is omitted, default XY texture projection is performed.

Default value: `'ind_coords=(undefined)'`.



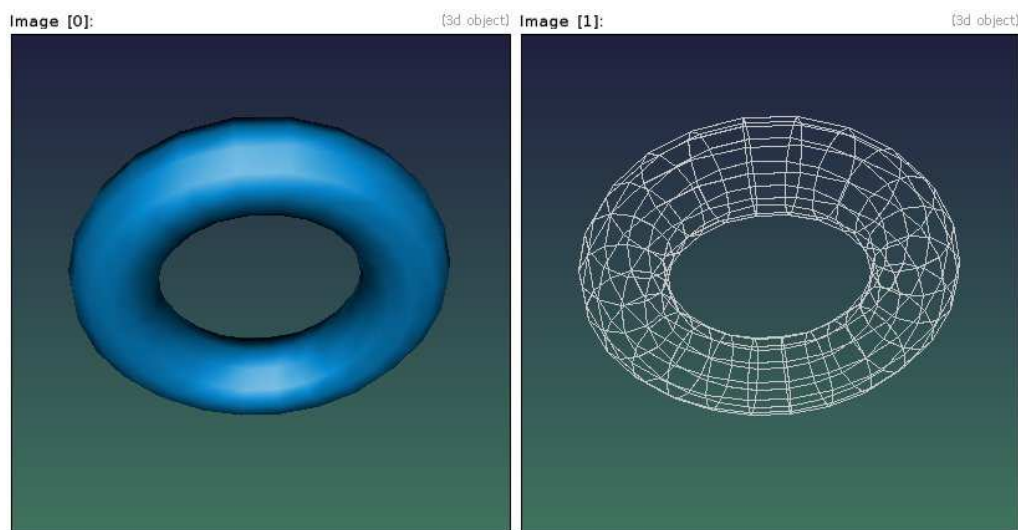
Example 505 : `image.jpg -torus3d 100,30 -texturize3d[-1] [-2] -keep[-1]`

2.12.81 *-torus3d*

Arguments: `_radius1, _radius2, _nb_subdivisions1>2, _nb_subdivisions2>2`

Input 3d torus at $(0,0,0)$, with specified geometry.

Default values: `'radius1=1', 'radius2=0.3', 'nb_subdivisions1=24'`
and `'nb_subdivisions2=12'`.

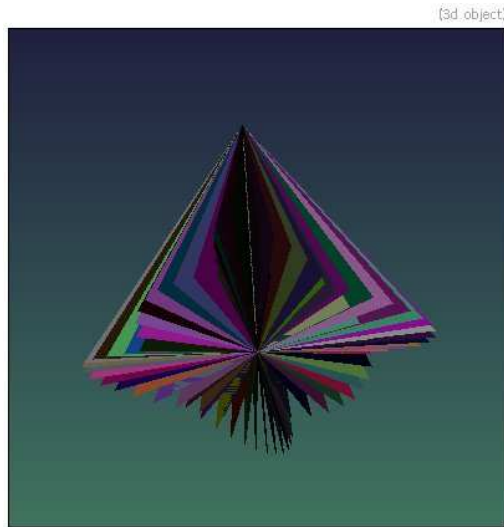


Example 506 : `-torus3d 10,3 --primitives3d 1 -color3d[-2] ${-RGB}`

2.12.82 *-triangle3d*

Arguments: `x0,y0,z0,x1,y1,z1,x2,y2,z2`

Input 3d triangle at specified coordinates.



Example 507 : `-repeat 100 a={>*pi/50} -triangle3d
0,0,0,0,0,3,{cos(3*$a)},{sin(2*$a)},0 -color3d[-1] ${-RGB} -done -add3d`

2.12.83 *-volume3d*

Transform selected 3d volumetric images as 3d parallelepipedic objects.



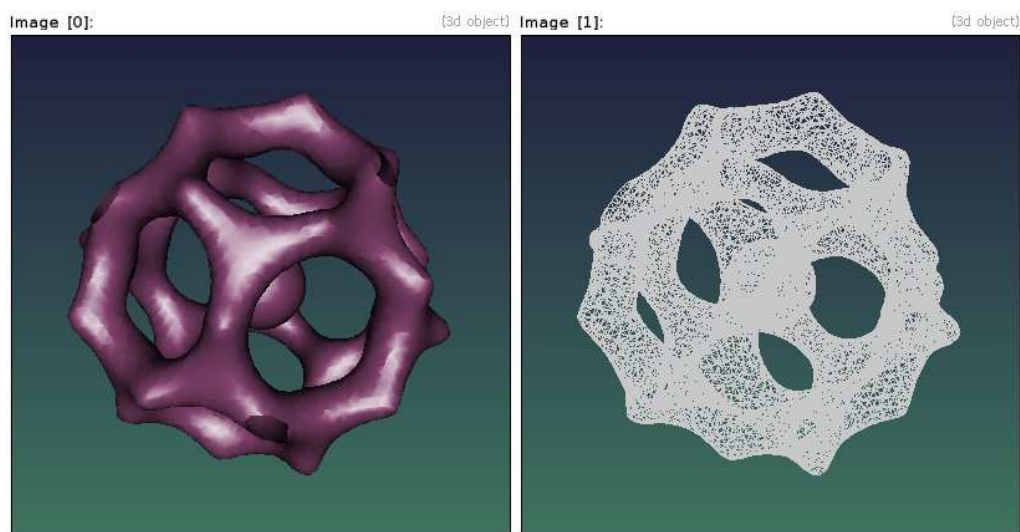
Example 508 : `image.jpg -animate blur,0,5,30 -a z -volume3d`

2.12.84 *-weird3d*

Arguments: `_resolution>0`

Input 3d weird object at (0,0,0), with specified resolution.

Default value: `'resolution=32'`.



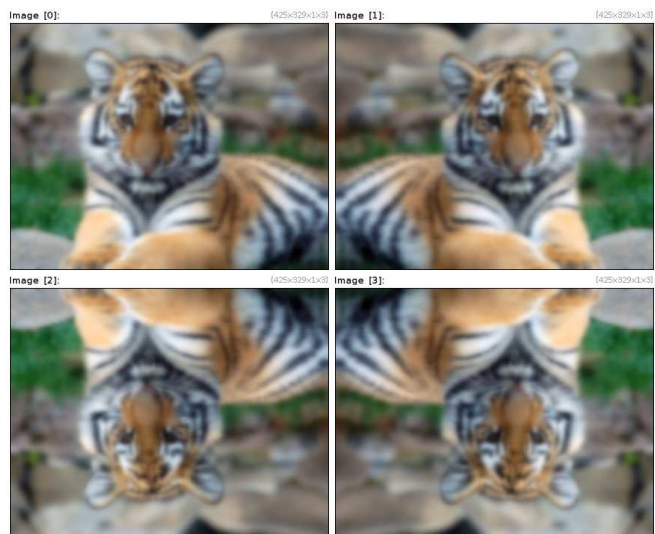
Example 509 : `-weird3d 48 --primitives3d 1 -color3d[-2] ${-RGB}`

2.13 Program control

2.13.1 *-apply_parallel*

Arguments: `"command"`

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

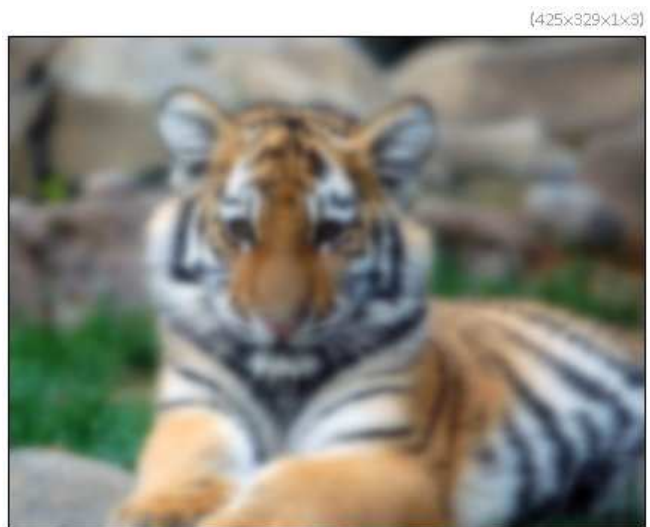


Example 510 : `image.jpg --mirror x --mirror y -apply-parallel "-blur 3"`

2.13.2 *-apply_parallel_channels*

Arguments: `"command"`

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



Example 511 : `image.jpg -apply-parallel-channels "-blur 3"`

2.13.3 *-apply_parallel_overlap*

Arguments: `"command",overlap[%],nb_threads={ 0=auto | 1 | 2
| 4 | 8 | 16 }`

Apply specified command on each of the selected images, by parallelizing it on 'nb_threads' overlapped sub-images.

(eq. to '-apo').

'nb_threads' must be a power of 2.

Default values: `'overlap=0','nb_threads=0'`.



Example 512 : `image.jpg --apply_parallel_overlap "-smooth 500,0,1",1`

2.13.4 *-apply_timeout*

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

Apply a command with a timeout.

2.13.5 *-check (+)*

Arguments: `expression`

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

If 'expression' is not evaluable, it is regarded as a filename and checked if it exists.

2.13.6 *-check3d* (+)

Arguments: `_is_full_check={ 0 | 1 }`

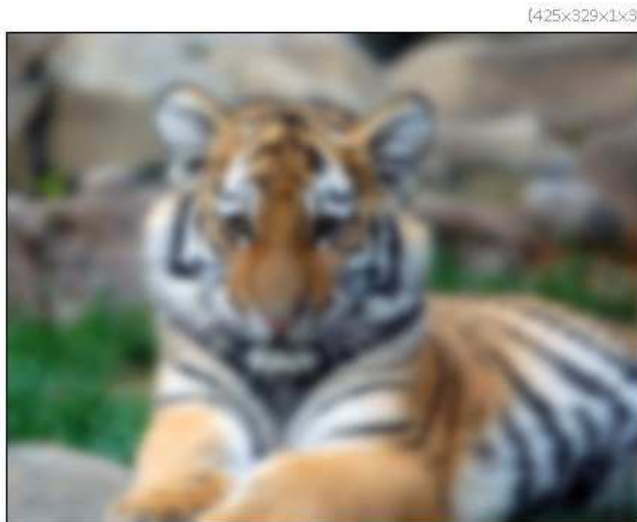
Check validity of selected 3d vector objects, and display an error message if one of the selected image is not a valid 3d vector object.

Full 3d object check is slower but more precise.

Default value: `'is_full_check=1'`.

2.13.7 *-continue* (+)

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



Example 513 : `image.jpg -repeat 10 -blur 1 -if {1==1} -continue -endif
-deform 10 -done`

2.13.8 *-break* (+)

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



Example 514 : `image.jpg -repeat 10 -blur 1 -if {l==1} -break -endif -deform
10 -done`

2.13.9 *-do (+)*

Start a 'do..while' block.



Example 515 : `image.jpg -luminance i={ia+2} -do -set 255,{?(100)}%,{?(100)}%
-while {ia<$i}`

2.13.10 *-done (+)*

End a 'repeat..done' block, and go to associated '-repeat' position, if iterations remain.

2.13.11 -elif(+)

Arguments: boolean |
 filename

Start a 'elif...[else]...endif' block if previous '-if' was not verified and test if specified boolean is true, or if specified filename exists.

'boolean' can be a float number standing for { 0=false
| other=true }.

2.13.12 -else(+)

Execute following commands if previous '-if' or '-elif' conditions failed.

2.13.13 -endif(+)

End a 'if...[elif]...[else]...endif' block.

2.13.14 -endlocal(+)

End a 'local...endlocal' block.
(eq. to '-endl').

2.13.15 -error(+)

Arguments: message

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.

2.13.16 -exec(+)

Arguments: command

Execute external command using a system call.

The status value is then set to the error code returned by the system call.

(eq. to '-x').

2.13.17 *-if* (+)

Arguments: boolean |
 filename

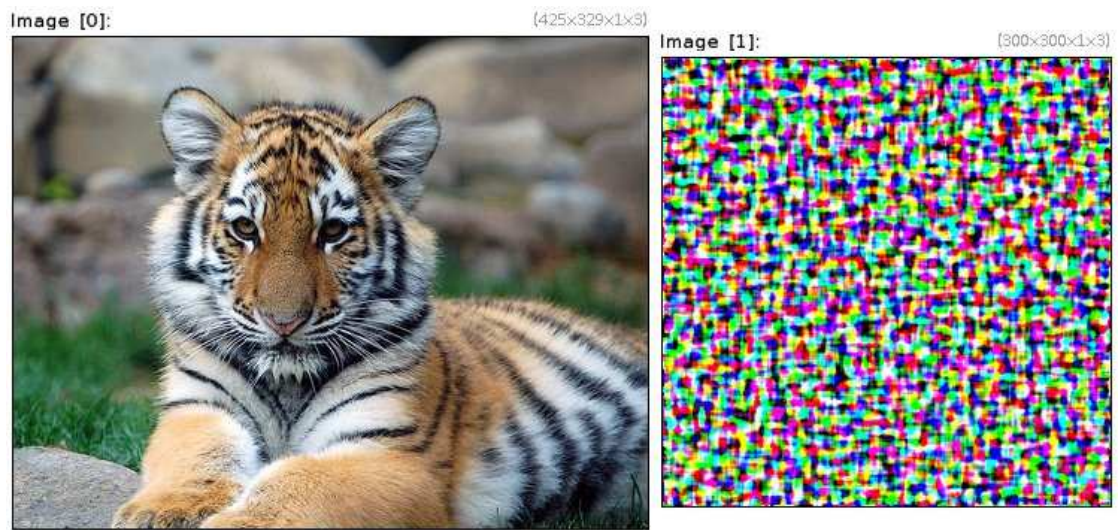
Start a 'if...[elif]...[else]..endif' block and test if specified boolean is true, or if specified filename exists.
'boolean' can be a float number standing for { 0=false
| other=true }.



Example 516 : `image.jpg -if {ia<64} -add 50% -elif {ia<128} -add 25% -elif {ia<192} -sub 25% -else -sub 50% -endif -cut 0,255`

2.13.18 *-local* (+)

Start a 'local...[onfail]..endlocal' block, with selected images.
(eq. to '-l').



Example 517 : `image.jpg -local[] 300,300,1,3 -rand[0] 0,255 -blur 4 -sharpen 1000 -endlocal`



Example 518 : `image.jpg --local -repeat 3 -deform 20 -done -endlocal`

Tutorial page:

http://gmic.eu/tutorial/_local.shtml

2.13.19 -mutex (+)

Arguments: `indice,_action={ 0=unlock | 1=lock }`

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.

'indice' designates the mutex indice, in [0,255].

Default value: `'action=1'`.

2.13.20 *-noarg* (+)

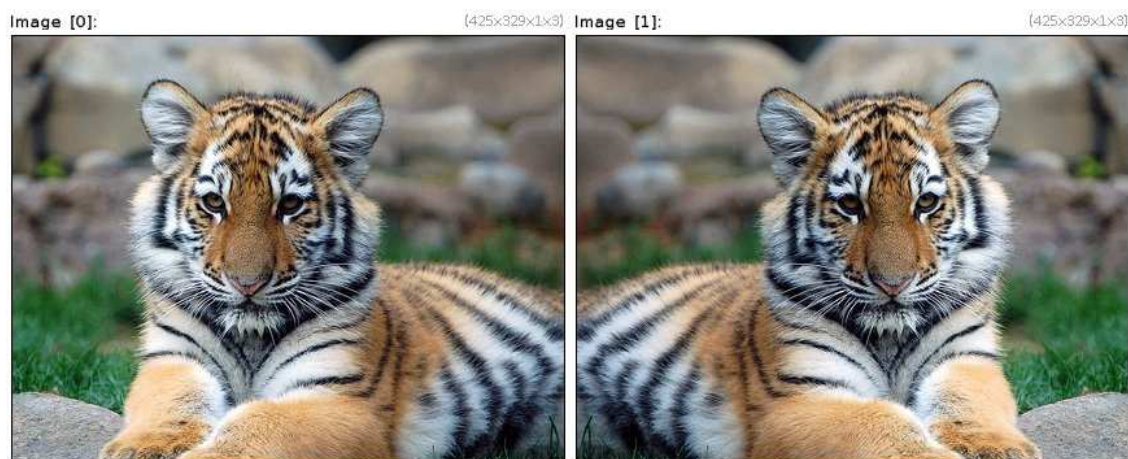
Used in a custom command, `'-noarg'` tells the command that its argument list have not been used finally, and so they must be evaluated next in the G'MIC pipeline, just as if the custom command takes no arguments at all.

Use this command to write a custom command which can decide if it takes arguments or not.

2.13.21 *-onfail* (+)

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 519 : `image.jpg --local -blur -3 -onfail -mirror x -endlocal`

2.13.22 *-parallel* (+)

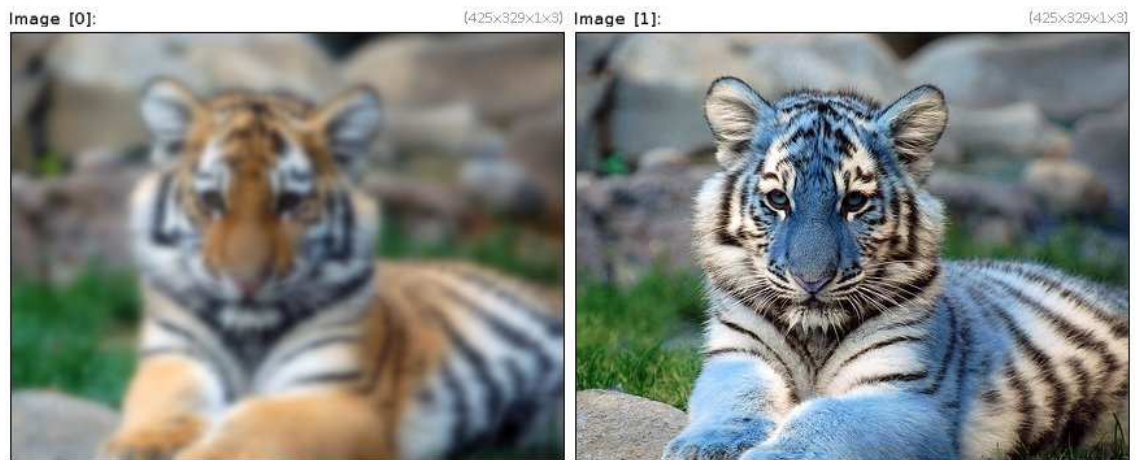
Arguments: `_wait_threads, "command1", "command2", ...`

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 value: `'wait_threads=1'`.



Example 520 : `image.jpg [0] -parallel "-blur[0] 3","-mirror[1] c"`

2.13.23 *-progress (+)*

Arguments: `0<=value<=100` |
 `-1`

Set the progress indice of the current processing pipeline.
This command is useful only when G'MIC is used by an embedding application.

2.13.24 *-quit (+)*

Quit interpreter.
(eq. to `'-q'`).

2.13.25 *-repeat (+)*

Arguments: `nb-iterations, _variable-name`

Start iterations of a `'repeat..done'` block.



Example 521 : `image.jpg -split y -repeat $!,n -shift[$n] $<,0,0,0,2 -done
-append y`



Example 522 : `image.jpg -mode3d 2 -repeat 4 -imagecube3d -rotate3d 1,1,0,40
-snapshot3d 400,1.4 -done`

Tutorial page:

http://gmick.eu/tutorial/_repeat.shtml

2.13.26 *-return (+)*

Return from current custom command.

2.13.27 -rprogress

Arguments: `0<=value<=100` | `-1` | `"command",0<=value_min<=100,0<=value_max<=100`

Set the progress indice of the current processing pipeline (relatively to previously defined progress bounds), or call the specified command with specified progress bounds.

2.13.28 -skip (+)

Arguments: `item`

Do nothing but skip specified item.

2.13.29 -status (+)

Arguments: `value`

Set current status value. Used to define a returning value in a function.
(eq. to `'-u'`).



Example 523 : `image.jpg -command "foo : u0=Dark u1=Bright -status ${u{ia}>=128}}" -text.outline ${-foo},2,2,23,2,1,255`

2.13.30 -while (+)

Arguments: `boolean` |

filename

End a 'do..while' block and go back to associated '-do' if specified boolean is true or if specified filename exists. 'boolean' can be a float number standing for { 0=false | other=true }.

2.14 Arrays, tiles and frames

2.14.1 -array

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

Create MxN array from selected images.

Default values: 'N=M' and 'expand_type=0'.



Example 524 : `image.jpg -array 3,2,2`

2.14.2 -array fade

Arguments: `M>0, N>0, 0<=_fade_start<=100, 0<=_fade_end<=100, _expand_type={ 0=min | 1=max | 2=all }`

Create MxN array from selected images.

Default values: 'N=M', 'fade_start=60', 'fade_end=90' and 'expand_type=1'.



Example 525 : `image.jpg -array_fade 3,2`

2.14.3 *-array_mirror*

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

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

Default values: `'dir=2'` and `'expand_type=0'`.



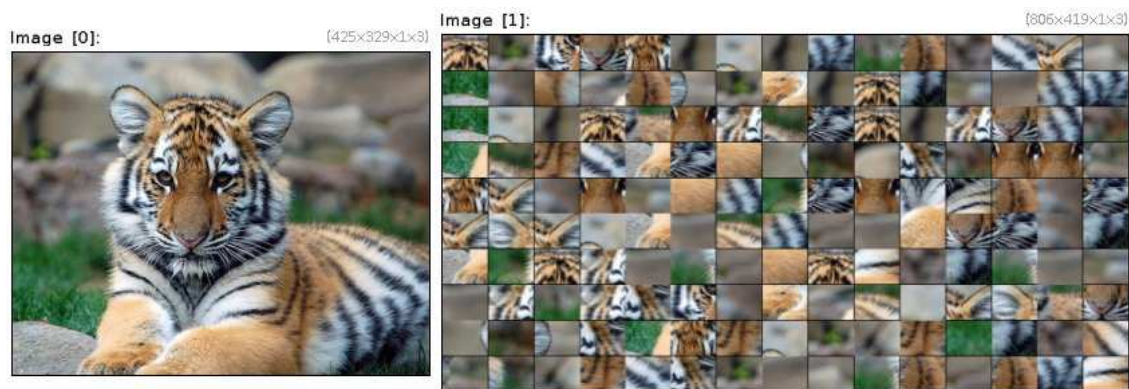
Example 526 : `image.jpg -array_mirror 2`

2.14.4 *-array_random*

Arguments: `Ms>0, _Ns>0, _Md>0, _Nd>0`

Create `Md×Nd` array of tiles from selected `Ms×Ns` source arrays.

Default values: `'Ns=Ms', 'Md=Ms'` and `'Nd=Ns'`.



Example 527 : `image.jpg --array-random 8,8,15,10`

2.14.5 *-frame_blur*

Arguments: `_sharpness>0, _size>=0, _smoothness, _shading, _blur`

Draw RGBA-colored round frame in selected images.

Default values: `'sharpness=10', 'size=30', 'smoothness=0', 'shading=1'` and `'blur=3%'`.



Example 528 : `image.jpg -frame_blur 3,30,8,10%`

2.14.6 *-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`

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 529 : `image.jpg -frame_cube ,`

2.14.7 *-frame fuzzy*

Arguments: `size_x[%]>=0, _size_y[%]>=0, _fuzzyness>=0, _smoothness[%]>=0, _R, _G, _B, _A`

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 530 : `image.jpg -frame.fuzzy 20`

2.14.8 *-frame painting*

Arguments: `_size[%]>=0, 0<=_contrast<=1, _profile_smoothness[%]>=0, _R, _G, _B, _vignette_size[%]>=0, _vignette_contrast>=0, _defects_contrast>=0, 0<=_defects_density<=100, _defects_size>=0, _defects_smoothness[%]>=0, _serial_number`

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'.`



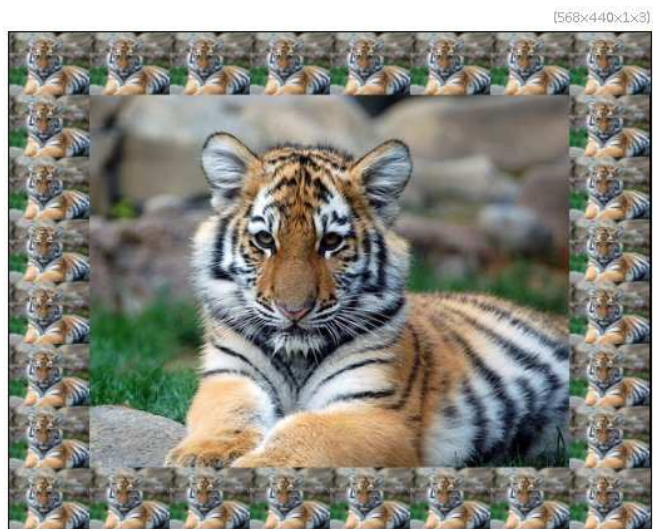
Example 531 : `image.jpg -frame.painting ,`

2.14.9 *-frame pattern*

Arguments: `M>=3, _constrain.size={ 0 | 1 } |`
`M>=3, _[frame_image], _constrain.size={ 0 | 1 }`

Insert selected pattern frame in selected images.

Default values: 'pattern=0' and 'constrain.size=0'.



Example 532 : `image.jpg -frame.pattern 8`

2.14.10 *-frame round*

Arguments: `_sharpness>0, _size>=0, _smoothness, _shading, _R, _G, _B, _A`

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 533 : `image.jpg -frame-round 10`

2.14.11 *-frame x*

Arguments: `size_x[%], _col1, ..., _colN`

Insert colored frame along the x-axis in selected images.

Default values: `'col1=col2=col3=255' and 'col4=255'.`



Example 534 : `image.jpg -frame_x 20,255,0,255`

2.14.12 *-frame_xy*

Arguments: `size_x[%],_size_y[%],_col1,...,_colN`

Insert colored frame along the x-axis in selected images.

Default values: `'size_y=size_x', 'col1=col2=col3=255' and 'col4=255'.`
(eq. to `'-frame'`).



Example 535 : `image.jpg -frame_xy 1,1,0 -frame_xy 20,10,255,0,255`

2.14.13 *-frame xyz*

Arguments: size-x[%],_size-y[%],_size-z[%]-col1,...,_colN

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'.

2.14.14 *-frame y*

Arguments: size-y[%],_col1,...,_colN

Insert colored frame along the y-axis in selected images.

Default values: 'col1=col2=col3=255' and 'col4=255'.



Example 536 : image.jpg -frame y 20,255,0,255

2.14.15 *-img2ascii*

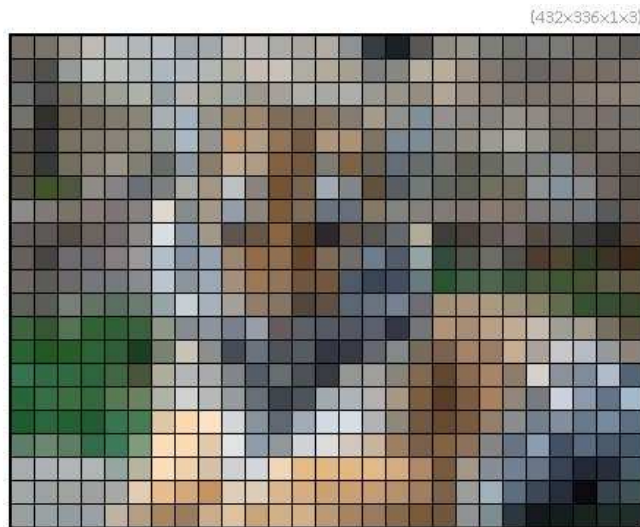
Arguments: _charset,_analysis_scale>0,_analysis_smoothness[%]>=0,_synthesis_scale>0,_output_ascii_filename

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.

2.14.16 *-imagegrid*

Default value: 'N=M'.



2.14.17 *-imagegrid_hexagonal*

```
Arguments:    _resolution.y>0,1<=_outline<=0,_is_antialiasing={ 0 | 1
}
```


Create hexagonal grids from selected images.

Default values: `'resolution.y>0'`, `'outline=0.1'` and `'is_antialiasing=1'`.

2.14.18 *-linearize_tiles*

Arguments: `M>0, N>0`

Linearize MxN tiles on selected images.

Default value: `'N=M'`.



Example 539 : `image.jpg --linearize_tiles 16`

2.14.19 *-map_sprites*

Arguments: `_nb_sprites>=1, _allow_rotation={ 0=none | 1=90 deg. | 2=180 deg. }`

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 540 : `image.jpg -r2dy 48 -repeat 16 -ball {8+2*$>}, ${-RGB} -*[-1]
{(1+$>)/16} -done -map_sprites 16`

2.14.20 *-pack*

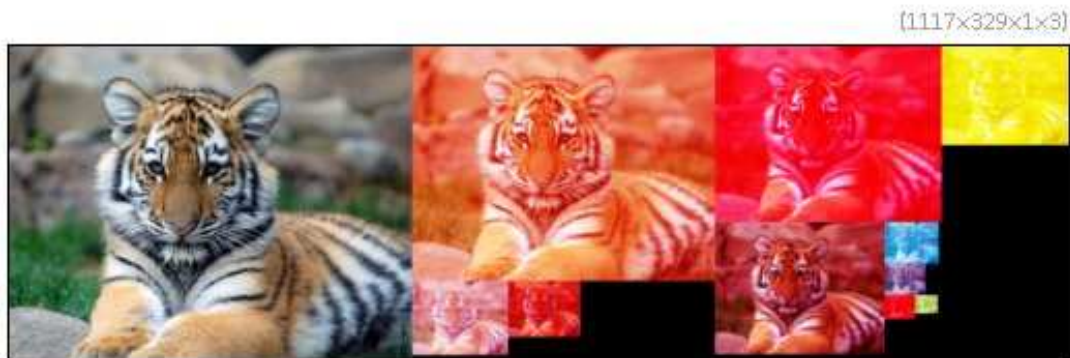
Arguments: `is_ratio_constraint={ 0 | 1 }, _sort_criterion`

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)'.



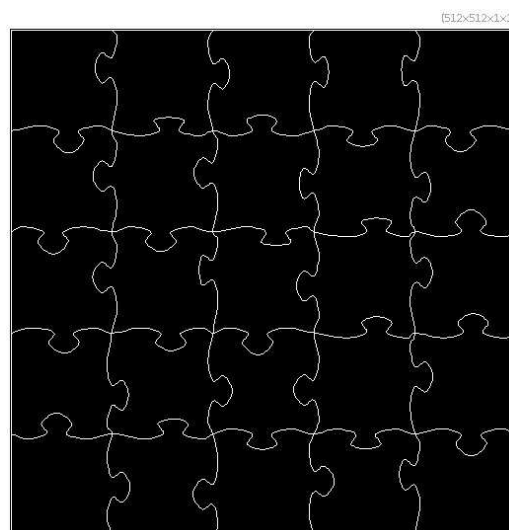
Example 541 : `image.jpg -repeat 10 --resize2dx[-1] 75% -balance_gamma[-1] ${-RGB} -done -pack 0`

2.14.21 *-puzzle*

Arguments: `_width>0, _height>0, _M>=1, _N>=1, _curvature, _centering, _connectors_variability, _resolution>=1`

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 542 : `-puzzle ,`

2.14.22 *-quadratize_tiles*

Arguments: $M > 0, N > 0$

Quadratize $M \times N$ tiles on selected images.

Default value: $'N=M'$.



Example 543 : `image.jpg --quadratize_tiles 16`

2.14.23 *-rotate_tiles*

Arguments: $angle, M > 0, N > 0$

Apply $M \times N$ tiled-rotation effect on selected images.

Default values: $'M=8'$ and $'N=M'$.



Example 544 : `image.jpg -to_rgba -rotate_tiles 10,8 -drop_shadow 10,10 -display_rgba`

2.14.24 *-shift_tiles*

Arguments: `M>0, N>0, _amplitude`

Apply MxN tiled-shift effect on selected images.

Default values: `'N=M'` and `'amplitude=20'`.



Example 545 : `image.jpg --shift_tiles 8,8,10`

2.14.25 *-taquin*

Arguments: `M>0, N>0, _remove_tile={ 0=none | 1=first | 2=last`


```
| 3=random },_relief,_border.thickness[%],_border.outline[%],_outline-  
color
```

Create MxN taquin puzzle from selected images.

Default value: 'N=M', 'relief=50', 'border.thickness=5',
'border.outline=0' and 'remove.tile=0'.



Example 546 : image.jpg --taquin 8

2.14.26 *-tunnel*

Arguments: _level>=0, _factor>0, _centering_x, _centering_y, _opacity, _angle

Apply tunnel effect on selected images.

Default values: 'level=9', 'factor=80%', 'centering_x=centering_y=0.5',
'opacity=1' and 'angle=0'



Example 547 : `image.jpg --tunnel 20`

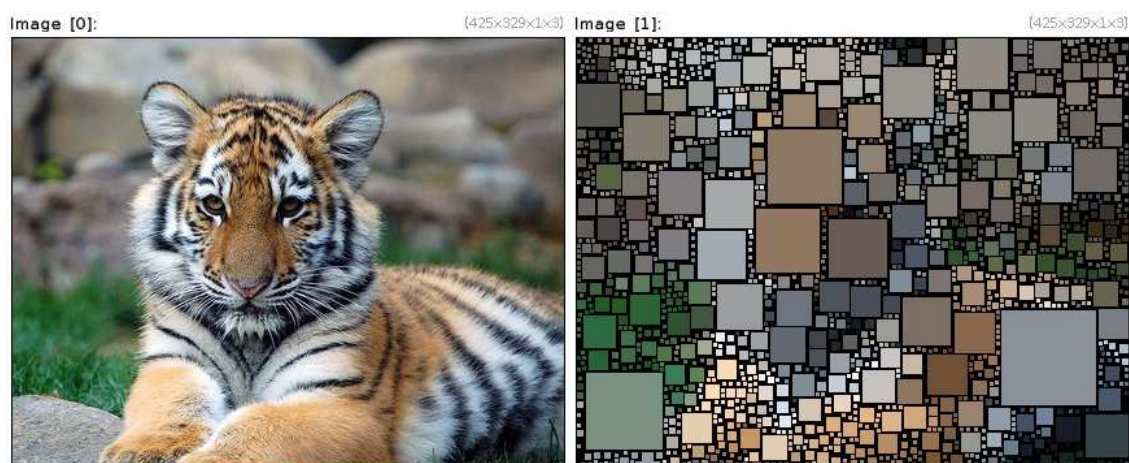
2.15 Artistic

2.15.1 *-boxfitting*

Arguments: `_min_box_size>=1, _max_box_size>=0, _initial_density>=0, _nb_attempts>=1`

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_tries=3'.`



Example 548 : `image.jpg --boxfitting ,`

2.15.2 *-cartoon*

Arguments: `_smoothness, _sharpening, _threshold>=0, _thickness>=0, _color>=0, quantization>0`

Apply cartoon effect on selected images.

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



Example 549 : `image.jpg --cartoon 3,80,15`

2.15.3 *-color_ellipses*

Arguments: `_count>0, _radius>=0, _opacity>=0`

Add random color ellipses to selected images.

Default values: `'count=400', 'radius=5' and 'opacity=0.1'.`



Example 550 : `image.jpg --color.ellipses ,,0.15`

2.15.4 *-cubism*

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

Apply cubism effect on selected images.

Default values: `'density=50', 'thickness=10', 'max_angle=75', 'opacity=0.7' and 'smoothness=0'.`



Example 551 : `image.jpg --cubism ,`

2.15.5 *-draw_whirl*

Arguments: `_amplitude>=0`

Apply whirl drawing effect on selected images.

Default value: `'amplitude=100'`.



Example 552 : `image.jpg --draw-whirl ,`

2.15.6 *-drawing*

Arguments: `_amplitude>=0`

Apply drawing effect on selected images.

Default value: `'amplitude=200'`.



Example 553 : `image.jpg --drawing ,`

2.15.7 *-drop_shadow*

Arguments: `_offset_x[%], _offset_y[%], _smoothness[%]>=0, 0<=_curvature<=1-`


```
,_expand.size={ 0 | 1 }
```

Drop shadow behind selected images.

Default values: 'offset_x=20', 'offset_y=offset_x', 'smoothness=5', 'curvature=0' and 'expand_size=1'.



Example 554 : `image.jpg -drop-shadow 10,20,5,0.5 -expand_xy 20,0 -display-rgba`

2.15.8 *-ellipsionism*

Arguments: `_R>0[%],_r>0[%],_smoothness>=0[%],_opacity,_outline>0,_density>0`

Apply ellipsionism filter to selected images.

Default values: 'R=10', 'r=3', 'smoothness=1%', 'opacity=0.7', 'outline=8' and 'density=0.6'.



Example 555 : `image.jpg --ellipsionism ,`

2.15.9 *-fire_edges*

Arguments: `_edges>=0,0<=_attenuation<=1,_smoothness>=0,_threshold>=0,_nb_frames>0,_starting_frame>=0,frame_skip>=0`

Generate fire effect from edges of selected images.

Default values: `'edges=0.7', 'attenuation=0.25', 'smoothness=0.5', 'threshold=25', 'nb_frames=1', 'starting_frame=20' and 'frame_skip=0'.`



Example 556 : `image.jpg -fire_edges ,`

2.15.10 *-fractalize*

Arguments: `0<=detail_level<=1`

Randomly fractalize selected images.

Default value: `'detail_level=0.8'`



Example 557 : `image.jpg --fractalize ,`

2.15.11 *-glow*

Arguments: `_amplitude>=0`

Add soft glow on selected images.

Default value: `'amplitude=1%'`.



Example 558 : `image.jpg --glow ,`

2.15.12 *-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`

Apply halftone dithering to selected images.

Default values: `'nb_levels=5', 'size_dark=8', 'size_bright=8',
'shape=5' and 'smoothness=0'.`



Example 559 : `image.jpg --halftone ,`

2.15.13 *-hardsketchbw*

Arguments: `_amplitude>=0, _density>=0, _opacity, 0<=_edge_threshold<=100, _-
is_fast={ 0 | 1 }`

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 560 : `image.jpg --hardsketchbw 200,70,0.1,10 -median[-1] 2 --local -reverse -blur[-1] 3 -blend[-2,-1] overlay -endlocal`

2.15.14 *-hearts*

Arguments: `_density>=0`

Apply heart effect on selected images.

Default value: `'density=10'`.



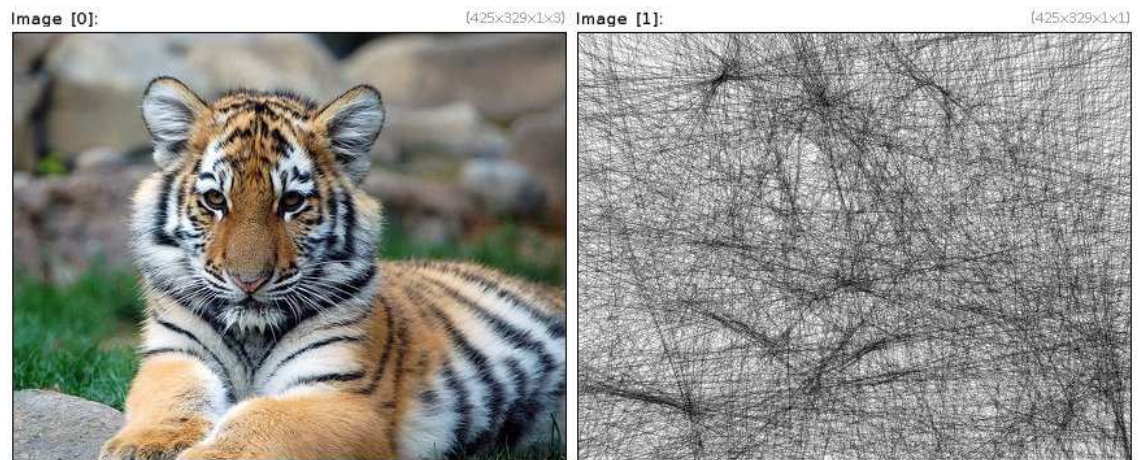
Example 561 : `image.jpg --hearts ,`

2.15.15 *-houghsketchbw*

Arguments: `_density>=0, _radius>0, 0<=_threshold<=100, 0<=_opacity<=1, _votesize[%]>0`

Apply hough B&W sketch effect on selected images.

Default values: `'density=8', 'radius=5', 'threshold=80', 'opacity=0.1' and 'votesize=100%'`.



Example 562 : `image.jpg --houghsketchbw ,`

2.15.16 *-lightrays*

Arguments: `100<=_density<=0, _center_x[%], _center_y[%], _ray_length>=0, _ray_attenuation>=0`

Generate ray lights from the edges of selected images.

Defaults values : `'density=50%', 'center_x=50%', 'center_y=50%', 'ray_length=0.9'` and `'ray_attenuation=0.5'`.



Example 563 : `image.jpg --lightrays , -+ -c 0,255`

2.15.17 *-light_relief*

Arguments: `_ambient_light, _specular_lightness, _specular_size, _light_smo-`

```
othness, _darkness, _xl, _yl, _zl, _zscale, _opacity.is.heightmap={ 0 | 1 }
```

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 564 : `image.jpg --blur 2 -light_relief[-1] 0.3,4,0.1,0`

2.15.18 -mosaic

Arguments: `_density>=0, _edges={ 0 | 1 }`

Create random mosaic from selected images.

Default values: 'density=0.8' and 'edges=1'.



Example 565 : `image.jpg --mosaic ,`

2.15.19 *-old_photo*

Apply old photo effect on selected images.



Example 566 : `image.jpg --old_photo`

2.15.20 *-pencilbw*

Arguments: `_size>=0, _amplitude>=0`

Apply B&W pencil effect on selected images.

Default values: `'size=0.3'` and `'amplitude=60'`.



Example 567 : `image.jpg --pencilbw ,`

2.15.21 *-polaroid*

Arguments: `_size1>=0, _size2>=0`

Create polaroid effect in selected images.

Default values: 'size1=10' and 'size2=20'.



Example 568 : `image.jpg -to.rgb -polaroid 5,30 -rotate 20 -drop_shadow ,
-display_rgba`

2.15.22 -polygonize

Arguments: `_warp_amplitude>=0, _smoothness[%]>=0, _min_area[%]>=0, _resolution_x[%]>0, _resolution_y[%]>0`

Apply polygon effect on selected images.

Default values: 'warp_amplitude=300', 'smoothness=2%',
'min_area=0.1%', 'resolution_x=resolution_y=10%'.



Example 569 : `image.jpg --polygonize ,`

2.15.23 *-poster_edges*

Arguments: `0<=_edge_threshold<=100,0<=_edge_shade<=100,_edge.thickness>=0,_edge_antialiasing>=0,0<=_posterization_level<=15,_posterization_antialiasing>=0`

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 570 : `image.jpg --poster_edges ,`

2.15.24 *-poster hope*

Arguments: `_smoothness>=0`

Apply Hope stencil poster effect on selected images.

Default value: `'smoothness=3'`.



Example 571 : `image.jpg --poster hope ,`

2.15.25 *-rodilius*

Arguments: `0<=_amplitude<=100, 0<=_thickness<=100, _sharpness>=0, _nb_orientations>0, _offset, _color_mode={ 0=darker | 1=brighter }`

Apply rodilius (fractalius-like) filter on selected images.

Default values: `'amplitude=10', 'thickness=10', 'sharpness=400', 'nb_orientations=7', 'offset=0' and 'color_mode=1'`.



Example 572 : `image.jpg --rodilius 12,10,300,10 -normalize-local[-1] 10,6`

2.15.26 *-stained_glass*

Arguments: `_edges[%]>=0, shading>=0, is_thin_separators={ 0 | 1 }`

Generate stained glass from selected images.

Default values: `'edges=40%', 'shading=0.2' and 'is_precise=0'.`



Example 573 : `image.jpg --stainedglass ,`

2.15.27 *-star*

Arguments: `_width>0, _height>0, _nb.branches>0, 0<=_thickness<=1`

Input star binary mask with specified size.

Default values: 'width=height=512', 'nb.branches=5' and 'thickness=0.38'.



Example 574 : `-star ,`

2.15.28 *-stars*

Arguments: `_density[%]>=0, _depth>=0, _size>0, _nb.branches>=1, 0<=_thickness<=1, _smoothness[%]>=0, _R, _G, _B, _opacity`

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'.



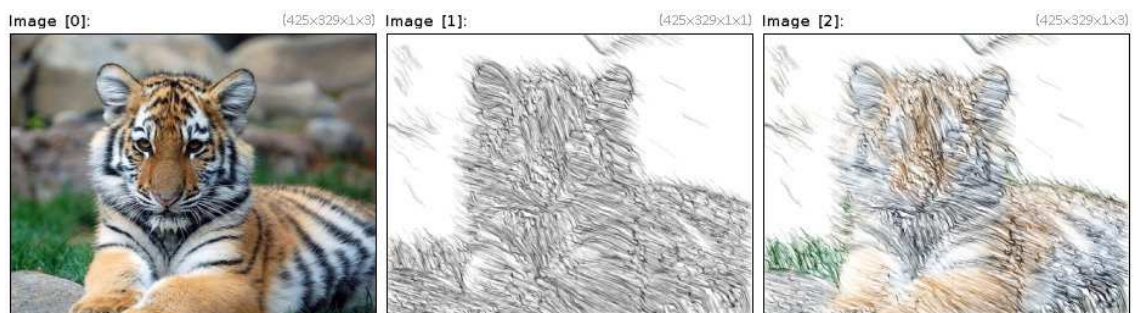
Example 575 : `image.jpg -stars ,`

2.15.29 *-sketchbw*

Arguments: `_nb_orients>0, _start_angle, _angle_range>=0, _length>=0, _threshold>=0, _opacity, _bgfactor>=0, _density>0, _sharpness>=0, _anisotropy>=0, _smoothness>=0, _coherence>=0, _is_boost={ 0 | 1 }, _is_curved={ 0 | 1 }`

Apply sketch effect to selected images.

Default values: `'nb_orients=2', 'start_angle=45', 'angle_range=180', 'length=30', 'threshold=1', '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 576 : `image.jpg --sketchbw 1 --local -reverse -blur[-1] 3
-blend[-2,-1] overlay -endlocal`

2.15.30 *-sponge*

Arguments: `_size>0`

Apply sponge effect on selected images.

Default value: `'size=13'`.



Example 577 : `image.jpg --sponge ,`

2.15.31 *-stencil*

Arguments: `_radius[%]>=0, _smoothness>=0, _iterations>=0`

Apply stencil filter on selected images.

Default values: `'radius=3', 'smoothness=1' and 'iterations=8'`.



Example 578 : `image.jpg --stencil 1,10,3`

2.15.32 *-stencilbw*

Arguments: `_edges>=0, _smoothness>=0`

Apply B&W stencil effect on selected images.

Default values: `'edges=15'` and `'smoothness=10'`.



Example 579 : `image.jpg --stencilbw 40,4`

2.15.33 *-tetris*

Arguments: `_scale>0`

Apply tetris effect on selected images.

Default value: `'scale=10'`.



Example 580 : `image.jpg --tetris 10`

2.15.34 -warhol

Arguments: `_M>0, _N>0, _smoothness>=0, _color>=0`

Create MxN Andy Warhol-like artwork from selected images.

Default values: `'M=3', 'N=M', 'smoothness=2'` and `'color=20'`.



Example 581 : `image.jpg --warhol 5,3,3,40`

2.15.35 -weave

Arguments: `_density>=0, 0<=_thickness<=100, 0<=_shadow<=100, _shading>=0, --fibers_amplitude>=0, _fibers_smoothness>=0, _angle, -1<=_x_curvature<=1, -1<=_y_curvature<=1`

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 582 : `image.jpg --weave ,`

2.15.36 *-whirls*

Arguments: `_texture>=0, _smoothness>=0, _darkness>=0, _lightness>=0`

Add random whirl texture to selected images.

Default values: `'texture=3', 'smoothness=6', 'darkness=0.5' and 'lightness=1.8'`.



Example 583 : `image.jpg --whirls ,`

2.16 Warpings

2.16.1 *-euclidean2polar*

Arguments: `_center_x[%], _center_y[%], _n>0, boundary={ 0=dirichlet`

```
| 1=neumann | 2=periodic }
```

Apply euclidean to polar transform on selected images.

Default values: 'center_x=center_y=50%', 'n=1' and 'boundary=1'.



Example 584 : `image.jpg --euclidean2polar ,`

2.16.2 *-deform*

Arguments: `_amplitude>=0`

Apply random smooth deformation on selected images.

Default value: 'amplitude=10'.



Example 585 : `image.jpg --deform[0] 10 --deform[0] 20`

2.16.3 *-fisheye*

Arguments: `_center_x, _center_y, 0<=_radius<=100, _amplitude>=0`

Apply fish-eye deformation on selected images.

Default values: 'x=y=50', 'radius=50' and 'amplitude=1.2'.



Example 586 : `image.jpg --fisheye ,`

2.16.4 *-flower*

Arguments: `_amplitude, _frequency, _offset_r[%], _angle, _center_x[%], _center_y[%], boundary={ 0=dirichlet | 1=neumann | 2=periodic }`

Apply flower deformation on selected images.

Default values: 'amplitude=30', 'frequency=6', 'offset_r=0', 'angle=0', 'center_x=center_y=50%' and 'boundary=2'.



Example 587 : `image.jpg -flower ,`

2.16.5 *-kaleidoscope*

Arguments: `_center_x[_],_center_y[_],_radius,_angle,_boundary={`
 `0=dirichlet | 1=neumann | 2=periodic }`

Create kaleidoscope effect from selected images.

Default values: `'center_x=center_y=50%', 'radius=100', 'angle=30'`
 and `'boundary=1'`.



Example 588 : `image.jpg --kaleidoscope ,`

2.16.6 *-map_sphere*

Arguments: `_width>0,_height>0,_radius,_dilation>0,_fading>=0,_fading-po-`
 `wer>=0`

Map selected images on a sphere.

Default values: `'width=height=512', 'radius=100', 'dilation=0.5',`
 `'fading=0' and 'fading_power=0.5'.`



Example 589 : `image.jpg --map_sphere ,`

2.16.7 *-polar2euclidean*

Arguments: `_center_x[_],_center_y[_],_n>0,_boundary={ 0=dirichlet
| 1=neumann | 2=periodic }`

Apply euclidean to polar transform on selected images.

Default values: `'center_x=center_y=50%', 'n=1' and 'boundary=1'.`



Example 590 : `image.jpg --euclidean2polar ,`

2.16.8 *-raindrops*

Arguments: `_amplitude,_density>=0,_wavelength>=0,_merging_steps>=0`

Apply raindrops deformation on selected images.

Default values: 'amplitude=80', 'density=0.1', 'wavelength=1' and 'merging_steps=0'.



Example 591 : `image.jpg --raindrops ,`

2.16.9 -ripple

Arguments: `_amplitude, _bandwidth, _shape={ 0=bloc | 1=triangle
| 2=sine | 3=sine+ | 4=random }, _angle, _offset`

Apply ripple deformation on selected images.

Default values: 'amplitude=10', 'bandwidth=10', 'shape=2', 'angle=0' and 'offset=0'.



Example 592 : `image.jpg --ripple ,`

2.16.10 *-rotoidoscope*

Arguments: `_center_x[%],_center_y[%],_tiles>0,_smoothness[%]>=0,_boundary={ 0=dirichlet | 1=neumann | 2=periodic }`

Create rotational kaleidoscope effect from selected images.

Default values: `'cx=cy=50%', 'tiles=10', 'smoothness=1' and 'boundary=1'.`



Example 593 : `image.jpg --rotoidoscope ,`

2.16.11 *-symmetrize*

Arguments: `_x[%],_y[%],_angle,_boundary={ 0=dirichlet | 1=neumann | 2=periodic },_is_antisymmetry={ 0 | 1 },_swap_sides={ 0 | 1 }`

Symmetrize selected image regarding specified axis.

Default values: `'x=y=50%', 'angle=90', 'boundary=1', 'is_antisymmetry=0' and 'swap_sides=0'.`



Example 594 : `image.jpg --symmetrize 50%,50%,45 --symmetrize[-1] 50%,50%,-45`

2.16.12 *-transform polar*

Arguments: `"expr_radius",_"expr_angle",_center_x[_%],_center_y[_%],_boundary={ 0=dirichlet | 1=neumann }`

Apply user-defined transform on polar representation of selected images.

Default values: `'expr_radius=R-r', 'expr_angle=a', 'center_x=center_y=50%'` and `'boundary=1'`.



Example 595 : `image.jpg --transform.polar[0] R*(r/R)^2,a --transform.polar[0] r,2*a`

2.16.13 *-twirl*

Arguments: `_amplitude,_center_x[_%],_center_y[_%],_boundary={ 0=dirichlet | 1=neumann | 2=periodic }`

Apply twirl deformation on selected images.

Default values: `'amplitude=1', 'center_x=center_y=50%'` and `'boundary=1'`.



Example 596 : `image.jpg --twirl 0.6`

2.16.14 *-warp perspective*

Arguments: `_x-angle, _y-angle, _zoom>0, _x-center, _y-center, boundary={
0=dirichlet | 1=neumann | 2=periodic }`

Warp selected images with perspective deformation.

Default values: `'x-angle=1.5', 'y-angle=0', 'zoom=1',
'x-center=y-center=50' and 'boundary=2'.`



Example 597 : `image.jpg --warp-perspective ,`

2.16.15 *-water*

Arguments: `_amplitude>=0, _smoothness>=0`

Apply water deformation on selected images.

Default values: 'amplitude=30' and 'smoothness=1.5'.



Example 598 : `image.jpg --water ,`

2.16.16 -wave

Arguments: `_amplitude>=0, _frequency>=0, _center.x, _center.y`

Apply wave deformation on selected images.

Default values: 'amplitude=4', 'frequency=0.4' and 'center.x=center.y=50'.



Example 599 : `image.jpg --wave ,`

2.16.17 *-wind*

Arguments: `_amplitude>=0, _angle, 0<=_attenuation<=1, _threshold`

Apply wind effect on selected images.

Default values: `'amplitude=20', 'angle=0', 'attenuation=0.7' and 'threshold=20'.`



Example 600 : `image.jpg --wind ,`

2.16.18 *-zoom*

Arguments: `_factor, _cx, _cy, _cz, boundary={ 0=dirichlet | 1=neumann | 2=periodic }`

Apply zoom factor to selected images.

Default values: `'factor=1', 'cx=cy=cz=0.5' and 'boundary=0'.`



Example 601 : `image.jpg --zoom[0] 0.6 --zoom[0] 1.5`

2.17 Degradations

2.17.1 *-cracks*

Arguments: `_density>=0, _amplitude, _relief={ 0 | 1 }`

Add random cracks to selected images.

Default values: `'density=0.2', 'amplitude=40'` and `'relief=0'`.



Example 602 : `image.jpg --cracks 0.2,60,1`

2.17.2 *-light patch*

Arguments: `_density>0, _darkness>=0, _lightness>=0`

Add light patches to selected images.

Default values: `'density=10', 'darkness=0.9'` and `'lightness=1.7'`.



Example 603 : `image.jpg --light_patch 20,0.9,4`

2.17.3 *-noise_hurl*

Arguments: `_amplitude>=0`

Add hurl noise to selected images.

Default value: `'amplitude=10'`.



Example 604 : `image.jpg --noise_hurl ,`

2.17.4 *-pixelize*

Arguments: `_scale_x>0, _scale_y>0, _scale_z>0`

Pixelize selected images with specified scales.

Default values: 'scale_x=20' and 'scale_y=scale_z=scale_x'.



Example 605 : `image.jpg --pixelize ,`

2.17.5 -scanlines

Arguments: `_amplitude, _bandwidth, _shape={ 0=bloc | 1=triangle
| 2=sine | 3=sine+ | 4=random }, _angle, _offset`

Apply ripple deformation on selected images.

Default values: 'amplitude=60', 'bandwidth=2', 'shape=0', 'angle=0' and 'offset=0'.



Example 606 : `image.jpg --ripple ,`

2.17.6 *-shade_stripes*

Arguments: `_frequency>=0, _direction={ 0=horizontal | 1=vertical }, _darkness>=0, _lightness>=0`

Add shade stripes to selected images.

Default values: `'frequency=5', 'direction=1', 'darkness=0.8' and 'lightness=2'`.



Example 607 : `image.jpg --shade_stripes 30`

2.17.7 *-shadow_patch*

Arguments: `_opacity>=0`

Add shadow patches to selected images.

Default value: `'opacity=0.7'`.



Example 608 : `image.jpg --shadow_patch 0.4`

2.17.8 *-spread*

Arguments: `_dx>=0, _dy>=0, _dz>=0`

Spread pixel values of selected images randomly along x,y and z.

Default values: `'dx=3', 'dy=dx' and 'dz=0'`.



Example 609 : `image.jpg --spread 3`

2.17.9 *-stripes_y*

Arguments: `_frequency>=0`

Add vertical stripes to selected images.

Default value: 'frequency=10'.



Example 610 : `image.jpg --stripes_y ,`

2.17.10 *-texturize_canvas*

Arguments: `_amplitude>=0, _fibrousness>=0, _emboss_level>=0`

Add paint canvas texture to selected images.

Default values: 'amplitude=20', 'fibrousness=3' and 'emboss_level=0.6'.



Example 611 : `image.jpg --texturize_canvas ,`

2.17.11 *-texturize_paper*

Add paper texture to selected images.



Example 612 : `image.jpg --texturize_paper`

2.17.12 *-vignette*

Arguments: `_strength>=0, 0<=_radius_min<=100, 0<=_radius_max<=100`

Add vignette effect to selected images.

Default values: `'strength=100', 'radius_min=70' and 'radius_max=90'.`



Example 613 : `image.jpg --vignette ,`

2.17.13 *-watermark_visible*

Arguments: `_text, 0<_opacity<1, _size>0, _angle, _mode={ 0=remove
| 1=add }, _smoothness>=0`

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 614 : image.jpg --watermark.visible ,0.7

2.18 Blending and fading

2.18.1 -blend

Arguments: [layer],blending_mode,0<=_opacity<=1,_selection_is={
0=base-layers | 1=top-layers } |
blending_mode,0<=_opacity<=1

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 | lighten
| lightness | linearburn | linearlight | luminance
| multiply | negation | or | overlay | pinlight | red
| reflect | saturation | seamless | seamless_mixed
| screen | shapeaverage | shapeaverage0 | softburn
| softdodge | softlight | stamp | subtract | value
| vividlight | xor }.

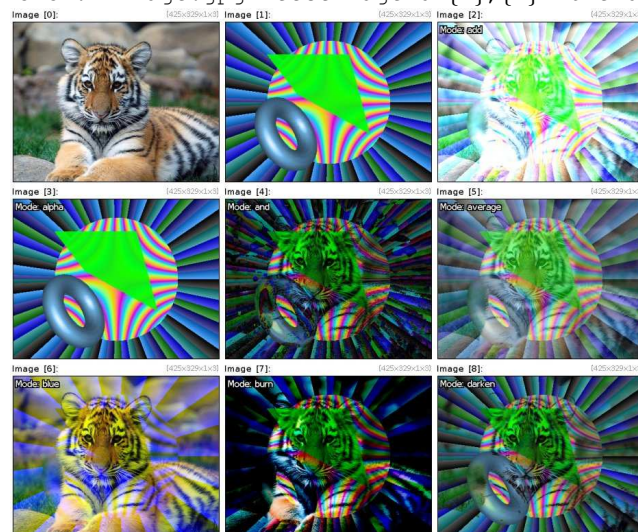
Default values: 'blending_mode=alpha', 'opacity=1' and 'selection_is=0'.



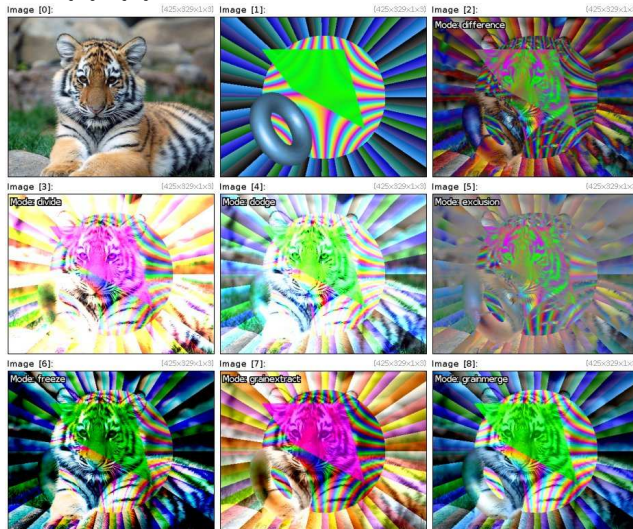
Example 615 : `image.jpg --drop_shadow , -resize2dy[-1] 200 -rotate[-1] 20 --blend alpha -display_rgba[-2]`



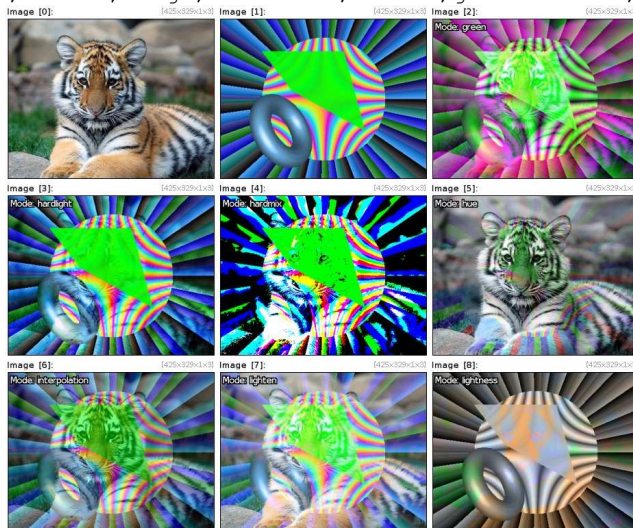
Example 616 : `image.jpg -testimage2d {w},{h} -blend overlay`



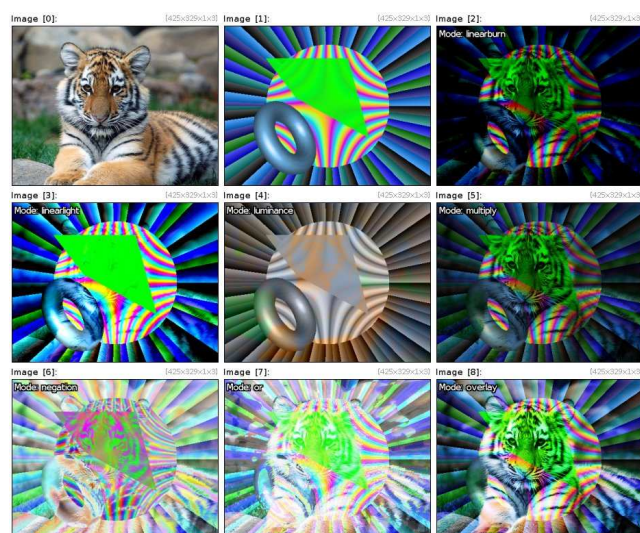
Example 617 : `-m "ex : $"=arg -repeat $"% --blend[0,1] ${arg{$>+1}}
 -text_outline[-1] Mode:\ " \("${arg{$>+1}}),2,2,23,2,1,255 -done" image.jpg
 -testimage2d {w},{h} -ex add,alpha,and,average,blue,burn,darken`



Example 618 : `-m "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`



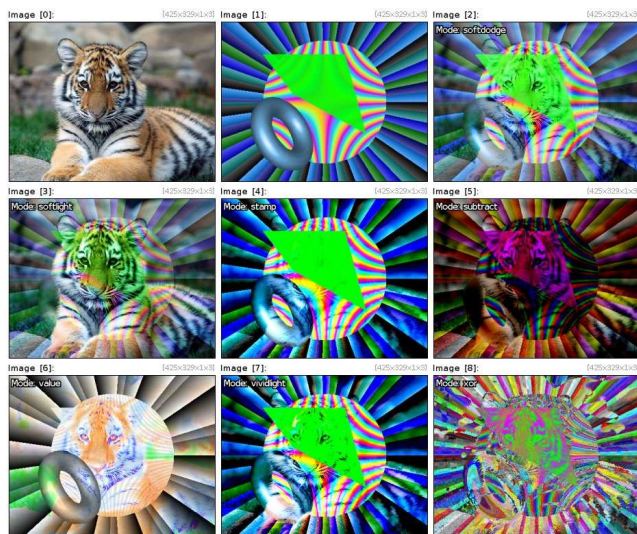
Example 619 : `-m "ex : $"=arg -repeat $"% --blend[0,1] ${arg{$>+1}}
 -text_outline[-1] Mode:\ " \("${arg{$>+1}}),2,2,23,2,1,255 -done" image.jpg
 -testimage2d {w},{h} -ex
 green,hardlight,hardmix,hue,interpolation,lighten,lightness`



Example 620 : `-m "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 621 : `-m "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`

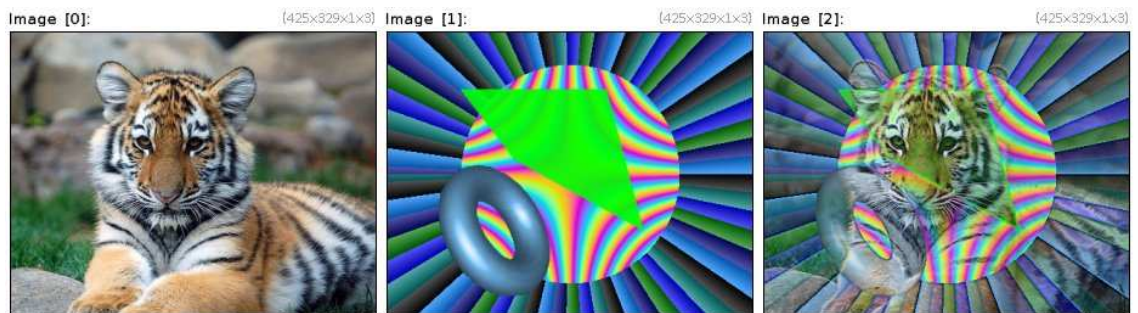


Example 622 : `-m "ex : $"=arg -repeat $"% --blend[0,1] ${arg{$>+1}}
 -text_outline[-1] Mode:\ " \("${arg{$>+1}}),2,2,23,2,1,255 -done" image.jpg
 -testimage2d {w},{h} -ex
 softdodge,softlight,stamp,subtract,value,vividlight,xor`

2.18.2 *-blend edges*

Arguments: `smoothness[%]>=0`

Blend selected images together using 'edges' mode.

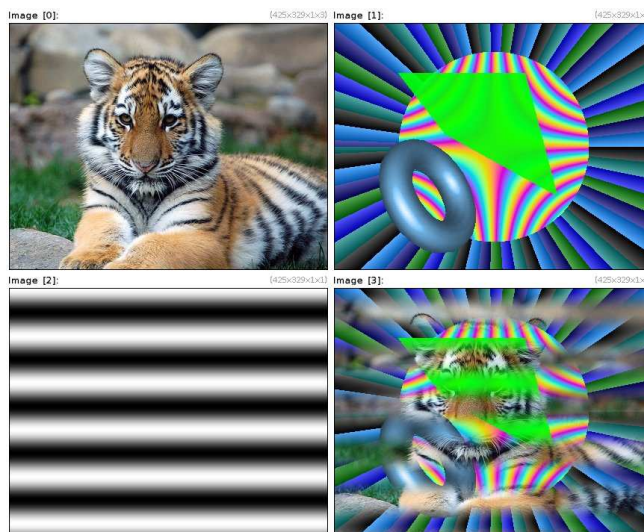


Example 623 : `image.jpg -testimage2d {w},{h} --blend-edges 0.8`

2.18.3 *-blend fade*

Arguments: `[fading_shape]`

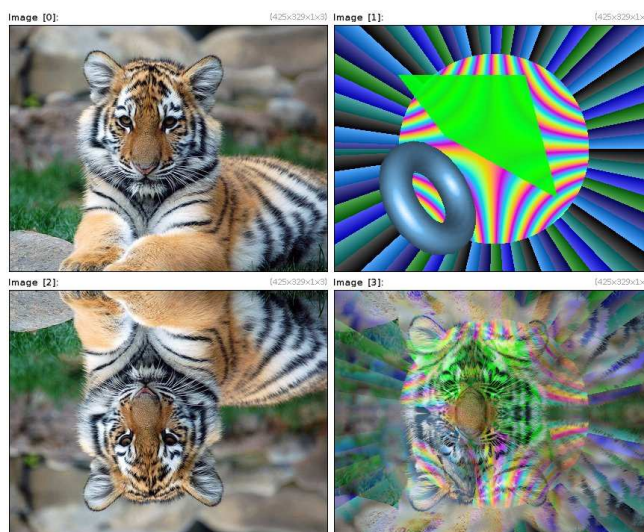
Blend selected images together using specified fading shape.



Example 624 : `image.jpg -testimage2d {w},{h} 100%,100%,1,1,'cos(y/10)'`
`-normalize[-1] 0,1 --blend.fade[0,1] [2]`

2.18.4 *-blend_median*

Blend selected images together using 'median' mode.



Example 625 : `image.jpg -testimage2d {w},{h} --mirror[0] y --blendmedian`

2.18.5 *-blend_seamless*

Arguments: `_ismixedmode={ 0 | 1 },_inner.fading[%]>=0,_outer.fading[%-]
]>=0`

Blend selected images using a seamless blending mode

(Poisson-based).

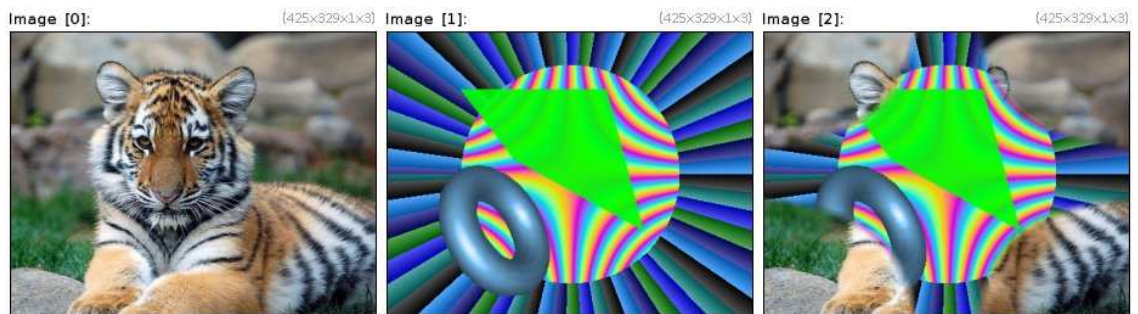
Default values: 'is_mixed=0', 'inner_fading=0' and 'outer_fading=100%'.

2.18.6 *-fade_diamond*

Arguments: $0 \leq \text{_start} \leq 100, 0 \leq \text{_end} \leq 100$

Create diamond fading from selected images.

Default values: 'start=80' and 'end=90'.



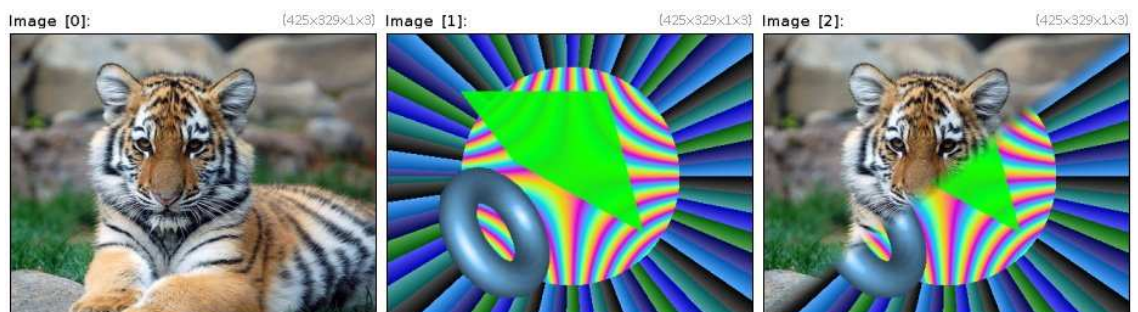
Example 626 : `image.jpg -testimage2d {w},{h} --fade_diamond 80,85`

2.18.7 *-fade_linear*

Arguments: $\text{_angle}, 0 \leq \text{_start} \leq 100, 0 \leq \text{_end} \leq 100$

Create linear fading from selected images.

Default values: 'angle=45', 'start=30' and 'end=70'.



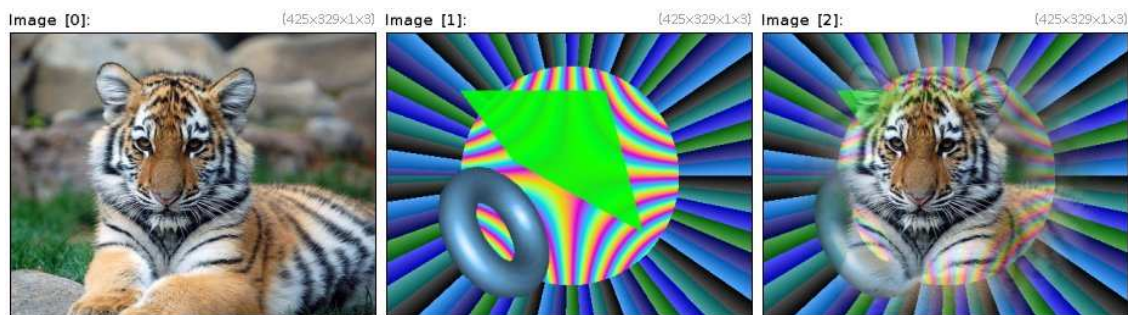
Example 627 : `image.jpg -testimage2d {w},{h} --fade_linear 45,48,52`

2.18.8 *-fade_radial*

Arguments: $0 \leq \text{_start} \leq 100, 0 \leq \text{_end} \leq 100$

Create radial fading from selected images.

Default values: 'start=30' and 'end=70'.



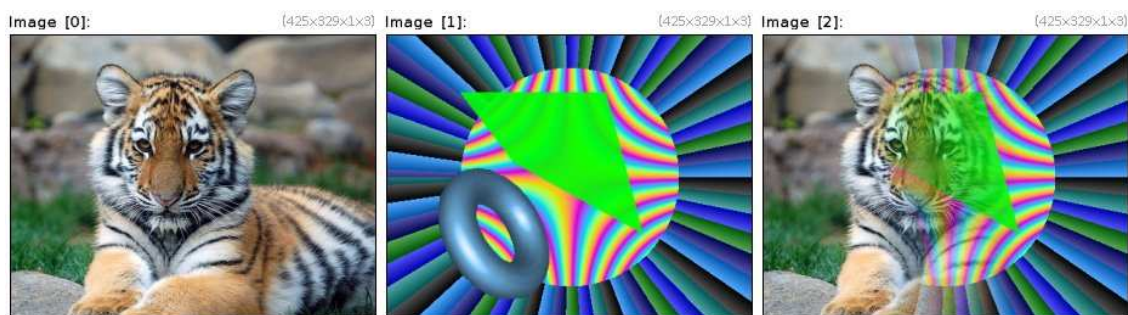
Example 628 : `image.jpg -testimage2d {w},{h} --fade_radial 30,70`

2.18.9 *-fade_x*

Arguments: $0 \leq \text{_start} \leq 100, 0 \leq \text{_end} \leq 100$

Create horizontal fading from selected images.

Default values: 'start=30' and 'end=70'.



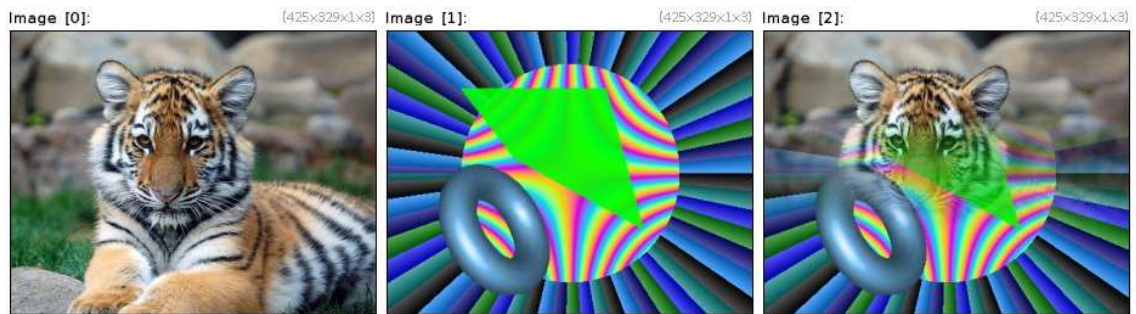
Example 629 : `image.jpg -testimage2d {w},{h} --fade_x 30,70`

2.18.10 *-fade_y*

Arguments: $0 \leq \text{_start} \leq 100, 0 \leq \text{_end} \leq 100$

Create vertical fading from selected images.

Default values: 'start=30' and 'end=70'.



Example 630 : `image.jpg -testimage2d {w},{h} --fade-y 30,70`

2.18.11 *-fade_z*

Arguments: `0<=_start<=100,0<=_end<=100`

Create transversal fading from selected images.

Default values: `'start=30'` and `'end=70'`.

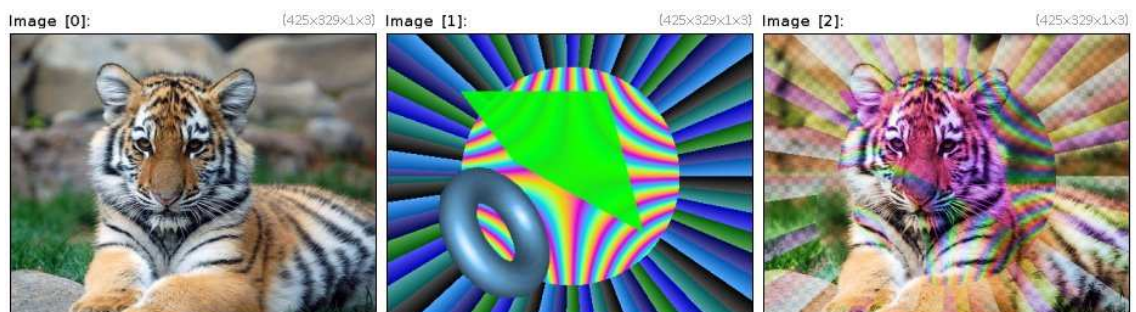
2.18.12 *-sub_alpha*

Arguments: `[base_image],_opacity_gain>=1`

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 `alpha_blend(B,A-B) = A`.

Default value: `'opacity_gain=1'`.



Example 631 : `image.jpg -testimage2d {w},{h} --sub_alpha[0] [1] -display_rgba`

2.19 Image sequences and videos

2.19.1 *-animate*

Arguments: `filter_name, "param1_start,...,paramN_start", "param1_end,...,paramN_end", nb_frames>=0, _output_frames={ 0 | 1 }, _output_filename | delay>0`

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

Default value: `'delay=30'`.



Example 632 : `image.jpg -animate flower,"0,3","20,8",9`

2.19.2 *-apply_camera*

Arguments: `_command, _camera_index>=0, _skip_frames>=0, _output_filename`

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

Default values: `'command=""`, `'camera_index=0'` (default camera), `'skip_frames=0'` and `'output_filename=""`.

2.19.3 *-apply_files*

Arguments: `"command", "filename-pattern", _output_prefix, _output_extension, _view_window={ 0 | 1 }`

Apply specified command on all specified image files, by reading them one by one, and save result by appending 'output_prefix' to each original filename.

If 'output_extension' is set, the output files are written using the specified extension instead of keeping the original one.

Default value: 'output_prefix=gmic-', 'output_extension=""' and 'view_window=0'.

2.19.4 *-apply_video*

Arguments: _command, _input_filename, _output_filename, _first_frame>=0, _last_frame={ >=0 | -1=last }, _frame_step>=1

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

Default values: 'command=""', 'input_filename=""', 'output_filename=""', 'first_frame=0', 'last_frame=-1' and 'frame_step=1'.

2.19.5 *-average_video*

Arguments: input_filename, _first_frame>=0, _last_frame={ >=0 | -1=last }, _frame_step>=1, _output_filename

Return the average of all frames of a video file.

If a display window is opened, the frames are displayed in it during processing.

2.19.6 *-files2video*

Arguments: "filename_pattern", _output_filename, _fps>0, _codec

Convert several files into a single video file.

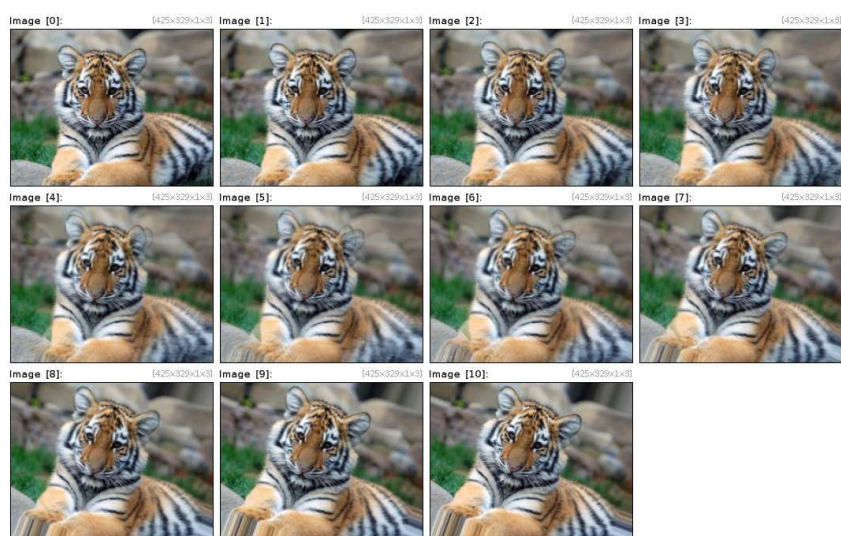
Default values: 'output_filename=output.avi', 'fps=25' and 'codec=mp4v'.

2.19.7 *-morph*

Arguments: nb_frames>0, _smoothness>=0, _precision>=0

Create morphing sequence between selected images.

Default values: 'smoothness=0.1' and 'precision=5'.



Example 633 : `image.jpg --rotate 20,1,1,50%,50% -morph 9`

2.19.8 *-register_nonrigid*

Arguments: `[destination],_smoothness>=0,_precision>0,_nb_scale>=0`

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 634 : `image.jpg --rotate 20,1,1,50%,50% --register_nonrigid[0] [1]`

2.19.9 *-register rigid*

Arguments: [destination],_smoothness>=0,_boundary={ 0=dirichlet
| 1=neumann | 2=periodic }

Register selected source images with specified destination image, using rigid warp (shift).

Default values: 'smoothness=1' and 'boundary=0'.



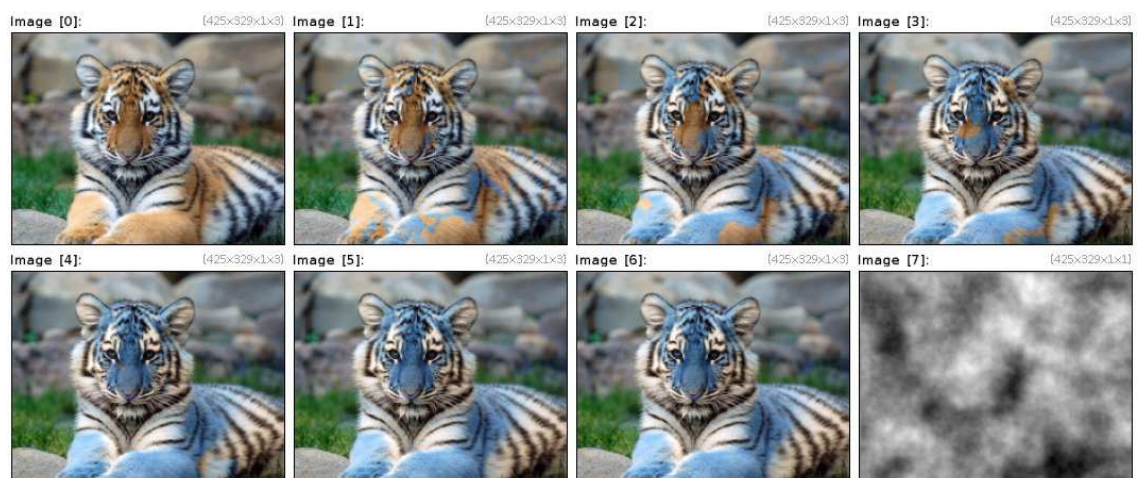
Example 635 : image.jpg --shift 30,20 --register_rigid[0] [1]

2.19.10 *-transition*

Arguments: [transition_shape],nb_added_frames>=0,100>=shading>=0,_single-frame-only={ -1=disabled | >=0 }

Generate a transition sequence between selected images.

Default values: 'shading=0' and 'single_frame_only=-1'.



Example 636 : `image.jpg --mirror c 100%,100% -plasma[-1] 1,1,6
-transition[0,1] [2],5`

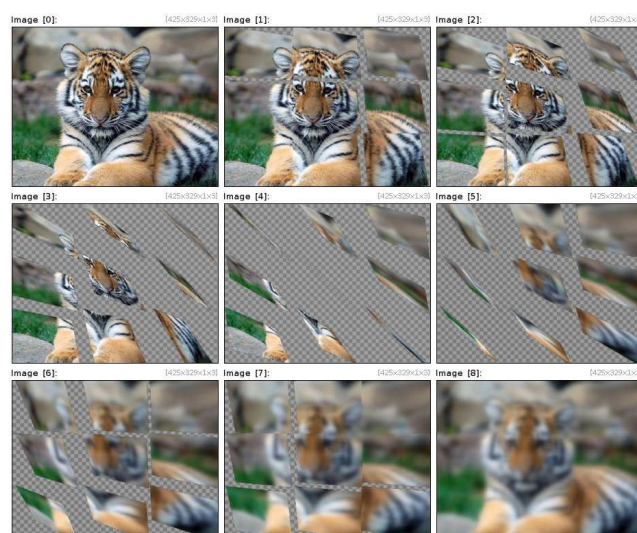
2.19.11 *-transition3d*

Arguments: `_nb_frames>=2, _nb_xtiles>0, _nb_ytiles>0, _axis_x, _axis_y, _axis_z, _is_antialias={ 0 | 1 }`

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 637 : `image.jpg --blur 5 -transition3d 9 -display.rgb`

2.19.12 *-video2files*

Arguments: `input_filename, output_filename, _first_frame>=0, _last_frame={ >=0 | -1=last }, _frame_step>=1`

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'.`

2.20 PINK-library operators

2.20.1 *-output_pink3d*

Arguments: `filename`

Save selected images as P5-coded PPM files (PINK extension for 3d volumetric images).

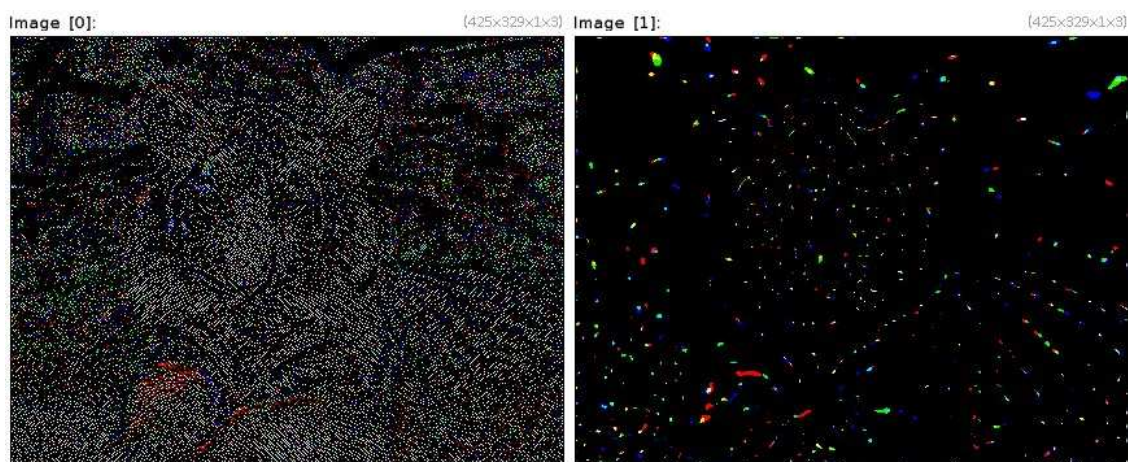
2.20.2 *-pink*

Pink wrapper name, p1, .. , pn (requires the PINK library to be installed).

(<http://pinkhq.com/>) prepares input, calls external "name input p1 ... pn output" and reads output (/tmp)



Example 638 : `image.jpg --pink asfr,5 -pink[0] asf,5`



Example 639 : `image.jpg --blur 2 -pink maxima,4`

2.20.3 *-pink_grayscale*

Arguments: `_connectivity={ 4 | 8 | 6 | 26 }, _lambda=0`

(http://pinkhq.com/doxygen/grayscale_8c.html)

Grayscale homotopic skeleton (requires the PINK library to be installed).

Default values: `'connectivity=4'` and `'lambda=0'`.



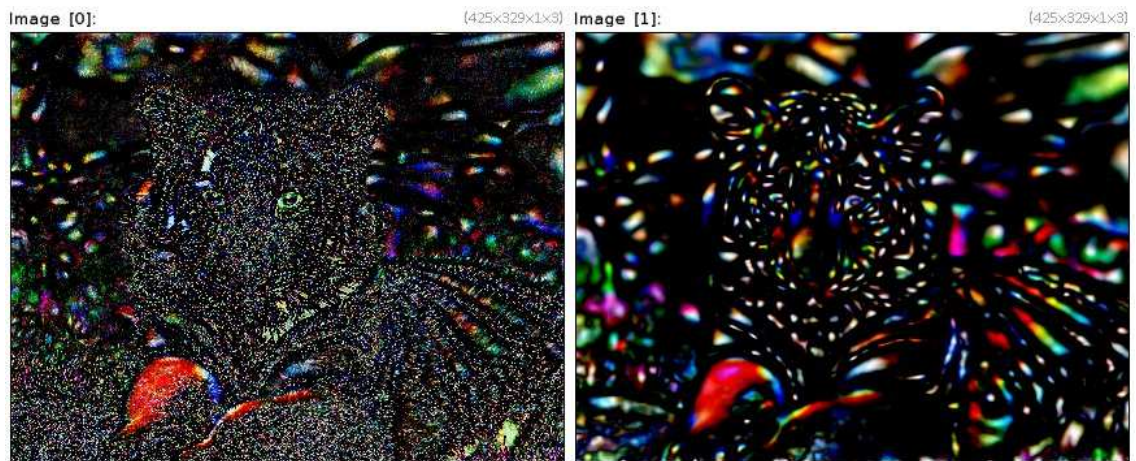
Example 640 : `image.jpg --pink_grayscale , --pink_grayscale[0] ,10
--pink_grayscale[0] ,100 -appendtiles 2`

2.20.4 *-pink_heightmaxima*

Arguments: `_connectivity={ 4 | 8 | 6 | 26 },_height=1`

(http://pinkhq.com/doxygen/heightmaxima_8c.html)
Heightmaxima filtering (requires the PINK library to be installed).

Default values: `'connectivity=4'` and `'height=1'`.



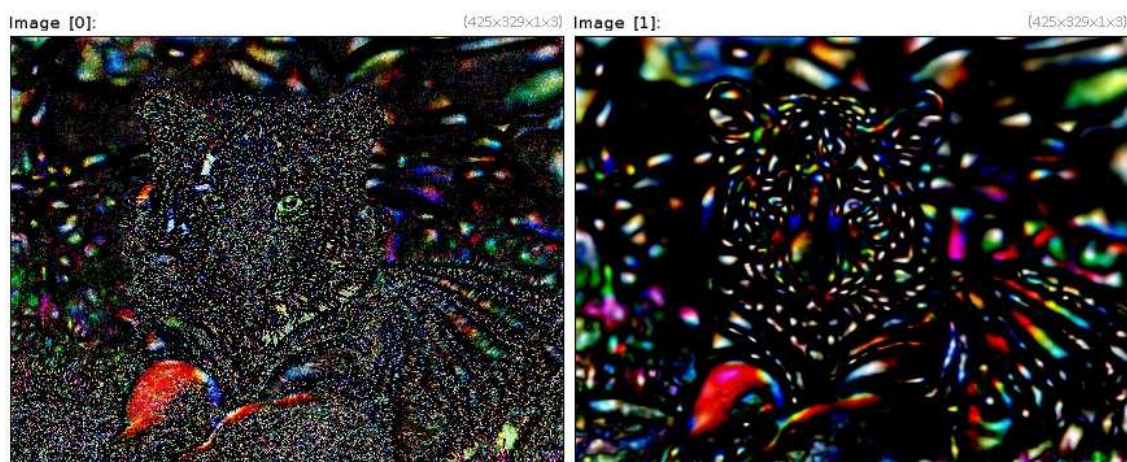
Example 641 : `image.jpg --blur 2 --pink_heightminima ,15
--pink_heightmaxima[0,1] ,15 --[-3,-1] --[-3,-1] -keep[-1,-2]`

2.20.5 *-pink_heightminima*

Arguments: `_connectivity={ 4 | 8 | 6 | 26 },_height=1`

(http://pinkhq.com/doxygen/heightminima_8c.html)
Heightminima filtering (requires the PINK library to be installed).

Default values: `'connectivity=4'` and `'height=1'`.



Example 642 : `image.jpg --blur 2 --pink.heightminima ,15
--pink.heightmaxima[0,1] ,15 --[-3,-1] --[-3,-1] -keep[-1,-2]`

2.20.6 *-pink_htkern*

Arguments: `_connectivity={ 4 | 8 | 6 | 26 }, _type={" " | u}`

(http://pinkhq.com/doxygen/htkern_8c.html)

(http://pinkhq.com/doxygen/htkernu_8c.html)

Grayscale ultimate homotopic thinning/thickening without condition (requires the PINK library to be installed).

Default values: 'connectivity=4' and 'type=" " '.



Example 643 : `image.jpg --pink.htkern ,u --pink.htkern[0] , ---[-1,-2]
-remove[0]`

2.20.7 *-pink_lvkernel*

Arguments: `_connectivity={ 4 | 8 | 6 | 26 }, _type={" " | u}`

(http://pinkhq.com/doxygen/lvkern_8c.html)

(http://pinkhq.com/doxygen/lvkernu_8c.html)

Grayscale ultimate leveling thinning/thickening without condition (requires the PINK library to be installed).

Default values: 'connectivity=4' and 'type=""'.



Example 644 : `image.jpg -pink_lvkernel ,u`

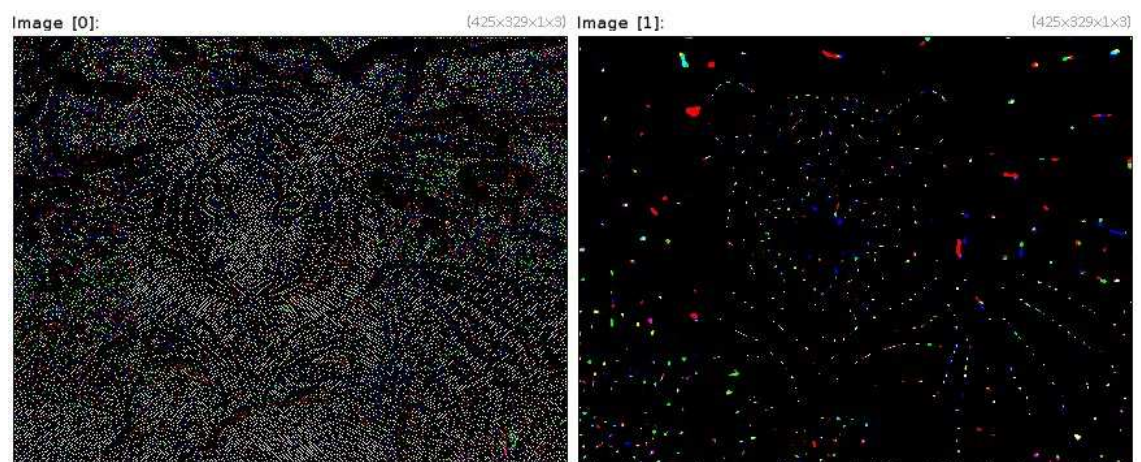
2.20.8 *-pink_reg_minima*

Arguments: `_connectivity={ 4 | 8 | 6 | 26 }`

(http://pinkhq.com/doxygen/minima_8c.html)

Regional minima (requires the PINK library to be installed).

Default values: 'connectivity=4'.



Example 645 : `image.jpg --blur 2 -pink.reg.minima ,`

2.20.9 *-pink_skelcurv*

Arguments: `_prio={0 | 1 | 2 | 3 | 4 | 8 | 6 | 26}, _connectivity={ 4 | 8 | 6 | 26 }, _inhibit={""}`

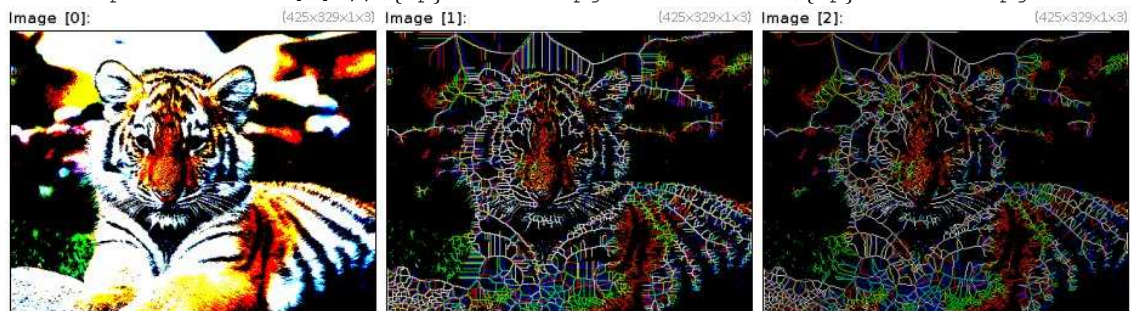
(http://pinkhq.com/doxygen/skelcurv_8c.html)

Curvilinear binary skeleton guided by a priority function or image (requires the PINK library to be installed).

Default values: `'prio=0', 'connectivity=4' and 'inhibit=""`.



Example 646 : `image.jpg -threshold 50% {w},{h} -fill[-1] 'if(x>w/2,255,0)'
tp=${-path.tmp} -output[-1] ${tp}/inhibit.pgm -remove[-1] --pink_skelcurv[0] ,
--pink_skelcurv[0] ,, ${tp}/inhibit.pgm -exec "rm "${tp}"/inhibit.pgm"`



Example 647 : `image.jpg -threshold 50% --pink_skelcurv , --pink_skelcurv[-2]
,8`

2.20.10 *-pink_skelend*

Arguments: `_connectivity={ 4 | 8 | 6 | 26 }, _n=0`

(http://pinkhq.com/doxygen/skelend_8c.html)

Homotopic skeleton of a 2d or 3d binary image with dynamic detection of end points (requires the PINK library to be

installed).

Default values: 'connectivity=4' and 'n=0'.



Example 648 : `image.jpg -threshold 50% --pink_skelend , --pink_skelend[-2] ,-1`

2.20.11 *-pink_skeleton*

Arguments: `_prio={0 | 1 | 2 | 3 | 4 | 8 | 6 | 26},_connectivity={ 4 | 8 | 6 | 26 },_inhibit=""`

(http://pinkhq.com/doxygen/skeleton_8c.html)

Ultimate binary skeleton guided by a priority image (requires the PINK library to be installed).

Default values: 'prio=0', 'connectivity=4' and 'inhibit=""'.



Example 649 : `image.jpg -threshold 50% --pink_skeleton[-1] ,`

2.20.12 *-pink_skelpar*

Arguments: `_algorithm={0..29},_nsteps=1,_inhibit=""`

(http://pinkhq.com/doxygen/skelpar_8c.html)
 Parallel binary skeleton (requires the PINK library to be installed).

Default values: 'algorithm=4', 'nsteps=-1' and 'inhibit=""'.



Example 650 : `image.jpg -threshold 50% --pink_skelpar[-1] 0 --pink_skelpar[-1] 2`

2.20.13 *-pink_wshed*

Arguments: `_connectivity={ 4 | 8 | 6 | 26 },_inverse={ 0 | 1 },_height=0`

(http://pinkhq.com/doxygen/wshedtopo_8c.html)
 Watershed (requires the PINK library to be installed).

Default values: 'connectivity=4', 'inverse=0' and 'height=0'.



Example 651 : `image.jpg --pink_wshed ,1,5 -pink_wshed[0] ,,5`

2.21 Convenience functions

2.21.1 *-alert*

Arguments: `_title, _message, _label_button1, _label_button2, ...`

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.'`

2.21.2 *-arg*

Arguments: `n, _arg1, ..., _argN`

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

2.21.3 *-arg2var*

Arguments: `variable_name, argument_1, ..., argument_N`

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).

2.21.4 *-at*

Arguments: `-x, -y, -z`

Return a specified vector-valued point `(x,y,z)` from the latest of the selected images.

2.21.5 *-autocrop_coords*

Arguments: `value1, value2, ... | auto`

Return coordinates `(x0,y0,z0,x1,y1,z1)` of the autocrop that could be performed on the latest of the selected images.

Default value: `'auto'`

2.21.6 -average_color

Return the average color of the latest of the selected images.

2.21.7 -basename

Arguments: `file_path, _variable_name_for_folder`

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).

2.21.8 -bin

Arguments: `binary_int1, ...`

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

2.21.9 -bin2dec

Arguments: `binary_int1, ...`

Convert specified binary integers into their decimal representations.

2.21.10 -dec

Arguments: `decimal_int1, ...`

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

2.21.11 -dec2str

Arguments: `decimal_int1, ...`

Convert specifical decimal integers into its string representation.

2.21.12 -dec2bin

Arguments: decimal_int1,...

Convert specified decimal integers into their binary representations.

2.21.13 -dec2hex

Arguments: decimal_int1,...

Convert specified decimal integers into their hexadecimal representations.

2.21.14 -dec2oct

Arguments: decimal_int1,...

Convert specified decimal integers into their octal representations.

2.21.15 -fact

Arguments: value

Return the factorial of the specified value.

2.21.16 -file_mv

Arguments: filename_src,filename_dest

Rename or move a file from a location \$1 to another location \$2.

2.21.17 -file_rand

Return a random filename for storing temporary data.

2.21.18 -file_rm

Arguments: filename

Delete a file.

2.21.19 *-filename*

Arguments: `filename, number1, number2, ..., numberN`

Return a filename numbered with specified indices.

2.21.20 *-files (+)*

Arguments: `_mode, path`

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 relative paths.

Default value: `'mode=5'`.

2.21.21 *-fitratio_wh*

Arguments: `min_width, min_height, ratio_wh`

Return a 2d size 'width,height' which is bigger than 'min_width,min_height' and has the specified w/h ratio.

2.21.22 *-fitscreen*

Arguments: `width, height, _depth, _minimal_size[%], _maximal_size[%]`

Return the 'ideal' size WxH for a window intended to display an image of specified size on screen.

2.21.23 *-fps*

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.

2.21.24 *-gcd*

Arguments: `a, b`

Return the GCD (greatest common divisor) between a and b.

2.21.25 -head

Arguments: length>=0

Return the first 'length' values of the last image, or all its values if image size is less than 'length'.

2.21.26 -hex

Arguments: hexadecimal_int1,...

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

2.21.27 -hex2dec

Arguments: hexadecimal_int1,...

Convert specified hexadecimal integers into their decimal representations.

2.21.28 -hex2str

Arguments: hexadecimal_string

Convert specified hexadecimal string into a string.

2.21.29 -img2str

Return the content of the latest of the selected image as a special G'MIC input string.

2.21.30 -img2text

Arguments: _line_separator

Return text contained in a multi-line image.

Default value: 'line_separator='.

2.21.31 -img82hex

Convert selected 8bits-valued vectors into their hexadecimal representations (ascii-encoded).

2.21.32 -hex2img8

Convert selected hexadecimal representations (ascii-encoded) into 8bits-valued vectors.

2.21.33 -is_3d

Return 1 if all of the selected image are 3d objects, 0 otherwise.

2.21.34 -is_image_arg

Arguments: string

Return 1 if specified string looks like '[ind]'.

2.21.35 -is_percent

Arguments: string

Return 1 if specified string ends with a '%', 0 otherwise.

2.21.36 -is_windows

Return 1 if current computer OS is Windows, 0 otherwise.

2.21.37 -mad

Return the MAD (Maximum Absolute Deviation) of the last selected image.

The MAD is defined as $MAD = \text{med}_i | x_i - \text{med}_j(x_j) |$

2.21.38 -max_w

Return the maximal width between selected images.

2.21.39 -max_h

Return the maximal height between selected images.

2.21.40 -max_d

Return the maximal depth between selected images.

2.21.41 -max_s

Return the maximal spectrum between selected images.

2.21.42 -max_wh

Return the maximal wxh size of selected images.

2.21.43 -max_whd

Return the maximal wxhxd size of selected images.

2.21.44 -max_whds

Return the maximal wxhxdxs size of selected images.

2.21.45 -med

Return the median value of the last selected image.

2.21.46 -color_med

Return the median color value of the last selected image.

2.21.47 -min_w

Return the minimal width between selected images.

2.21.48 -min_h

Return the minimal height between selected images.

2.21.49 -min_d

Return the minimal depth between selected images.

2.21.50 -min_s

Return the minimal s size of selected images.

2.21.51 -min_wh

Return the minimal wxh size of selected images.

2.21.52 -min_whd

Return the minimal wxhxd size of selected images.

2.21.53 -min_whds

Return the minimal wxhxdxs size of selected images.

2.21.54 *-normalize filename*

Arguments: filename

Return a "normalized" version of the specified filename, without spaces and capital letters.

2.21.55 *-oct*

Arguments: octal_int1,...

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

2.21.56 *-oct2dec*

Arguments: octal_int1,...

Convert specified octal integers into their decimal representations.

2.21.57 *-padint*

Arguments: number, _size>0

Return a integer with 'size' digits (eventually left-padded with '0').

2.21.58 *-path_gimp*

Return a path to store GIMP configuration files for one user (whose value is OS-dependent).

2.21.59 *-path_tmp*

Return a path to store temporary files (whose value is OS-dependent).

2.21.60 *-path_prc*

Return a path to the parent of resources directory.
On Unix, add a point at the end of the path.

2.21.61 -path_rc

Return a path to store persistent resources for one user (whose value is OS-dependent).

2.21.62 -quote

Arguments: string

Return a "quotified" version of the string.

2.21.63 -region_feature

Arguments: region_label, feature, _default_value

Return feature for a specified region.

This function requires two images [img, region_label] in the selection.

Argument 'feature' is a string that corresponds to the way the feature would be asked for the entire image.

Default value: 'default_value=0'.



Example 652 : `image.jpg --luminance -quantize[-1] 2 -label[-1] 0,1
mean=${"-region_feature[0,1] 10,\"{ia}\""} sum=${"-region_feature[0,1]
10,\"{is}\""}`

2.21.64 -reset

Reset global parameters of the interpreter environment.

2.21.65 -RGB

Return a random int-valued RGB color.

2.21.66 -RGBA

Return a random int-valued RGBA color.

2.21.67 -str

Arguments: string

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

2.21.68 -str2hex

Arguments: string

Convert specified string into a sequence of hexadecimal values.

2.21.69 -stresc

Arguments: val1,...,valN

Return escaped string from specified ascii codes.

2.21.70 -strcat

Arguments: string1,string2,...

Return the concatenation of all strings passed as arguments.

2.21.71 -strcmp

Arguments: string1,string2

Return 1 if the two strings are equal, 0 otherwise.

2.21.72 -strlen

Arguments: string1

Return the length of specified string argument.

2.21.73 -strreplace

Arguments: string, search, replace

Search and replace substrings in an input string.

2.21.74 -struncase

Arguments: string

Return a lower-case version of the specified string.

2.21.75 -strver

Return the current version number of the G'MIC interpreter, as a string.

2.21.76 -tic

Initialize tic-toc timer.

Use it in conjunction with '-toc'.

2.21.77 -toc

Display elapsed time of the tic-toc timer since the last call to '-tic'.

Use it in conjunction with '-tic'.

2.21.78 -std_noise

Return the estimated noise standard deviation of the last selected image.

2.22 Other interactive commands**2.22.1 -demo**

Arguments: _run_in_parallel={ 0=no | 1=yes | 2=auto }

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

2.22.2 -x_2048

Launch the 2048 game.

2.22.3 -x_blobs

Launch the blobs editor.

2.22.4 *-x bouncing*

Launch the bouncing balls demo.

2.22.5 *-x color_curves*

Arguments: `_colorspace={ rgb | cmy | cmyk | hsi | hsl | hsv
| lab | lch | ycbcr | last }`

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 value: `'colorspace=rgb'`.

2.22.6 *-x colorize*

Arguments: `_is_lineart={ 0 | 1 },_max_resolution={ 0 | >=128
,_multichannels_output={ 0 | 1 },-[palette1],[-palette2]`

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'`.

2.22.7 *-x fire*

Launch the fire effect demo.

2.22.8 *-x fireworks*

Launch the fireworks demo.

2.22.9 *-x fisheye*

Launch the fish-eye effect demo.

2.22.10 *-x fourier*

Launch the fourier filtering demo.

2.22.11 *-x histogram*

Launch the histogram demo.

2.22.12 -x *hough*

Launch the hough transform demo.

2.22.13 -x *jawbreaker*

Arguments: `0<_width<20,0<_height<20,0<_balls<=8`

Launch the Jawbreaker game.

2.22.14 -x *landscape*

Launch the virtual landscape demo.

2.22.15 -x *life*

Launch the game of life.

2.22.16 -x *light*

Launch the light effect demo.

2.22.17 -x *mandelbrot*

Arguments: `_julia={ 0 | 1 },_c0r,_c0i`

Launch Mandelbrot/Julia explorer.

2.22.18 -x *metaballs3d*

Launch the 3d metaballs demo.

2.22.19 -x *minesweeper*

Arguments: `8<=_width=<20,8<=_height<=20`

Launch the Minesweeper game.

2.22.20 -x *minimal_path*

Launch the minimal path demo.

2.22.21 -x *pacman*

Launch pacman game.

2.22.22 -x *paint*

Launch the interactive painter.

2.22.23 -x *plasma*

Launch the plasma effect demo.

2.22.24 -x *quantize_rgb*

Arguments: _nbcolors>=2

Launch the RGB color quantization demo.

2.22.25 -x *reflection3d*

Launch the 3d reflection demo.

2.22.26 -x *rubber3d*

Launch the 3d rubber object demo.

2.22.27 -x *segment*

Arguments: _max.resolution={ 0 | >=128 }

Segment foreground from background in selected opaque RGB images, interactively.
Return RGBA images with binary alpha-channels.

Default value: 'max.resolution=1024'.

2.22.28 -x *select_color*

Arguments: _variable.name

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 value: 'variable.name=xsc.variable'.

2.22.29 -x *select_function1d*

Arguments: _variable.name, _background.curve.R, _background.curve.G, _background.curve.B

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'.

2.22.30 -x_select_palette

Arguments: _variable_name, _number_of_columns= { 0=auto | >0 }

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 (ad 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'.

2.22.31 -x_shadebobs

Launch the shade bobs demo.

2.22.32 -x_spline

Launch spline curve editor.

2.22.33 -x_tetris

Launch tetris game.

2.22.34 -x_tictactoe

Launch tic-tac-toe game.

2.22.35 -x_waves

Launch the image waves demo.

2.22.36 -x *whirl***Arguments:** `_opacity>=0`

Launch the fractal whirls demo.

Default values: `'opacity=0.2'`.**2.23 Commands shortcuts**

- `'-h'` is equivalent to `'-help'`.
- `'-m'` (+) is equivalent to `'-command'`.
- `'-d'` (+) is equivalent to `'-display'`.
- `'-d0'` is equivalent to `'-display0'`.
- `'-d3d'` (+) is equivalent to `'-display3d'`.
- `'-da'` is equivalent to `'-display_array'`.
- `'-dfft'` is equivalent to `'-display_fft'`.
- `'-dg'` is equivalent to `'-display_graph'`.
- `'-dh'` is equivalent to `'-display_histogram'`.
- `'-dp'` is equivalent to `'-display_polar'`.
- `'-drgba'` is equivalent to `'-display_rgba'`.
- `'-dt'` is equivalent to `'-display_tensors'`.
- `'-dw'` is equivalent to `'-display_warp'`.
- `'-e'` (+) is equivalent to `'-echo'`.
- `'-i'` (+) is equivalent to `'-input'`.
- `'-o'` (+) is equivalent to `'-output'`.
- `'-on'` is equivalent to `'-outputn'`.
- `'-op'` is equivalent to `'-outputp'`.
- `'-ow'` is equivalent to `'-outputw'`.
- `'-ox'` is equivalent to `'-outputx'`.
- `'-p'` (+) is equivalent to `'-print'`.
- `'-sh'` (+) is equivalent to `'-shared'`.
- `'-up'` is equivalent to `'-update'`.
- `'-v'` (+) is equivalent to `'-verbose'`.
- `'-w'` (+) is equivalent to `'-window'`.
- `'-k'` (+) is equivalent to `'-keep'`.
- `'-mv'` (+) is equivalent to `'-move'`.
- `'-nm'` (+) is equivalent to `'-name'`.
- `'-nms'` is equivalent to `'-names'`.
- `'-rm'` (+) is equivalent to `'-remove'`.
- `'-rv'` (+) is equivalent to `'-reverse'`.
- `'-+'` (+) is equivalent to `'-add'`.
- `'-&'` (+) is equivalent to `'-and'`.

- `'-<<<'` (+) is equivalent to `'-bsl'`.
- `'->>>'` (+) is equivalent to `'-bsr'`.
- `'-/'` (+) is equivalent to `'-div'`.
- `'-=='` (+) is equivalent to `'-eq'`.
- `'->='` (+) is equivalent to `'-ge'`.
- `'->'` (+) is equivalent to `'-gt'`.
- `'-<='` (+) is equivalent to `'-le'`.
- `'-<'` (+) is equivalent to `'-lt'`.
- `'-//'` (+) is equivalent to `'-mdiv'`.
- `'-%'` (+) is equivalent to `'-mod'`.
- `'-**'` (+) is equivalent to `'-mmul'`.
- `'-*'` (+) is equivalent to `'-mul'`.
- `'-!='` (+) is equivalent to `'-neq'`.
- `'-|'` (+) is equivalent to `'-or'`.
- `'-^'` (+) is equivalent to `'-pow'`.
- `'--'` (+) is equivalent to `'-sub'`.
- `'-c'` (+) is equivalent to `'-cut'`.
- `'-f'` (+) is equivalent to `'-fill'`.
- `'-ir'` is equivalent to `'-inrange'`.
- `'-n'` (+) is equivalent to `'-normalize'`.
- `'-='` (+) is equivalent to `'-set'`.
- `'-ac'` is equivalent to `'-apply-channels'`.
- `'-fc'` is equivalent to `'-fill-color'`.
- `'-a'` (+) is equivalent to `'-append'`.
- `'-z'` (+) is equivalent to `'-crop'`.
- `'-r'` (+) is equivalent to `'-resize'`.
- `'-rr2d'` is equivalent to `'-resize-ratio2d'`.
- `'-r2dx'` is equivalent to `'-resize2dx'`.
- `'-r2dy'` is equivalent to `'-resize2dy'`.
- `'-r3dx'` is equivalent to `'-resize3dx'`.
- `'-r3dy'` is equivalent to `'-resize3dy'`.
- `'-r3dz'` is equivalent to `'-resize3dz'`.
- `'-s'` (+) is equivalent to `'-split'`.
- `'-y'` (+) is equivalent to `'-unroll'`.
- `'-b'` (+) is equivalent to `'-blur'`.
- `'-g'` (+) is equivalent to `'-gradient'`.
- `'-j'` (+) is equivalent to `'-image'`.
- `'-j3d'` (+) is equivalent to `'-object3d'`.
- `'-t'` (+) is equivalent to `'-text'`.
- `'-+3d'` (+) is equivalent to `'-add3d'`.
- `'-c3d'` is equivalent to `'-center3d'`.
- `'-col3d'` (+) is equivalent to `'-color3d'`.
- `'-/3d'` (+) is equivalent to `'-div3d'`.
- `'-db3d'` (+) is equivalent to `'-double3d'`.

- `'-f3d'` (+) is equivalent to `'-focale3d'`.
- `'-l3d'` (+) is equivalent to `'-light3d'`.
- `'-m3d'` (+) is equivalent to `'-mode3d'`.
- `'-md3d'` (+) is equivalent to `'-moded3d'`.
- `'-*3d'` (+) is equivalent to `'-mul3d'`.
- `'-n3d'` is equivalent to `'-normalize3d'`.
- `'-o3d'` (+) is equivalent to `'-opacity3d'`.
- `'-p3d'` (+) is equivalent to `'-primitives3d'`.
- `'-rv3d'` (+) is equivalent to `'-reverse3d'`.
- `'-r3d'` (+) is equivalent to `'-rotate3d'`.
- `'-sl3d'` (+) is equivalent to `'-spec13d'`.
- `'-ss3d'` (+) is equivalent to `'-specs3d'`.
- `'-s3d'` (+) is equivalent to `'-split3d'`.
- `'--3d'` (+) is equivalent to `'-sub3d'`.
- `'-t3d'` (+) is equivalent to `'-texturize3d'`.
- `'-ap'` is equivalent to `'-apply-parallel'`.
- `'-apc'` is equivalent to `'-apply-parallel.channels'`.
- `'-apo'` is equivalent to `'-apply-parallel.overlap'`.
- `'-endl'` (+) is equivalent to `'-endlocal'`.
- `'-x'` (+) is equivalent to `'-exec'`.
- `'-l'` (+) is equivalent to `'-local'`.
- `'-q'` (+) is equivalent to `'-quit'`.
- `'-u'` (+) is equivalent to `'-status'`.
- `'-frame'` is equivalent to `'-frame.xy'`.

2.24 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 -o output.jpg
- Create a volumetric image from a movie sequence:
gmic input.mpg -a z -o output.hdr
- Compute image gradient norm:
gmic input.bmp -gradient_norm
- Denoise a color image:
gmic image.jpg -denoise 30,10 -o denoised.jpg

- Compose two images using overlay layer blending:
`gmic image1.jpg image2.jpg --blend overlay -o blended.jpg`
- Evaluate a mathematical expression:
`gmic -e "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 -f "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,"?(0,255)" --plasma 10,3 --blur 4 --sharpen 10000 \`
`--elevation3d[-1] ""X=(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 -m3d 5 --md3d 5 --db3d 0 --isosurface3d ""x^2+y^2+abs(z)^abs(4*cos(x*y*z*3))"" ,3`
- Render a G'MIC 3d logo:
`gmic 0 --text G\MIC,0,0,53,1,1,1,1 --expand_xy 10,0 --blur 1 -n 0,100 --plasma 0.4 --+ \`
`--blur 1 --elevation3d -0.1 --md3d 4`
- Generate a 3d ring of torii:
`gmic --repeat 20 --torus3d 15,2 --color3d[-1] ""?(60,255)},{?(60,255)},{?(60,255)}" \`
`--*3d[-1] 0.5,1 --if ""{>%2}" --r3d[-1] 0,1,0,90 --endif --+3d[-1] 70 --+3d \`
`--r3d 0,0,1,18 --done --md3d 3 --m3d 5 --db3d 0`
- Create a vase from a 3d isosurface:
`gmic --md3d 4 --isosurface3d ""x^2+2*abs(y/2)*sin(2*y)^2+z^2-3',0" --sphere3d 1.5 \`
`--3d[-1] 0,5 --plane3d 15,15 --r3d[-1] 1,0,0,90 --center3d[-1] --+3d[-1] 0,3.2 \`
`--color3d[-1] 180,150,255 --color3d[-2] 128,255,0 --color3d[-3] 255,128,0 --+3d`
- Display filtered webcam stream:
`gmic --apply_camera \""--mirror x --mirror y --+ -/ 4\""`
- Launch a set of G'MIC interactive demos:
`gmic --demo`

Index of commands

- RGB, 426
- RGBA, 427
- abs, 50
- acos, 50
- add, 51
- add3d, 277
- alert, 418
- and, 53
- animate, 405
- animate3d, 278
- append, 141
- append_tiles, 142
- apply_camera, 405
- apply_camera3d, 279
- apply_channels, 118
- apply_curve, 92
- apply_files, 405
- apply_gamma, 93
- apply_parallel, 329
- apply_parallel_channels, 330
- apply_parallel_overlap, 331
- apply_timeout, 331
- apply_video, 406
- area, 227
- area_fg, 228
- arg, 418
- arg2var, 418
- array, 341
- array3d, 279
- array_fade, 341
- array_mirror, 342
- array_random, 343
- arrow3d, 279
- asin, 54
- at, 418
- at_line, 229
- atan, 55
- atan2, 56
- autocrop, 143
- autocrop_components, 144
- autocrop_coords, 418
- autocrop_seq, 145
- autoindex, 118
- average_color, 419
- average_video, 406
- axes, 247
- axes3d, 280
- balance_gamma, 93
- ball, 248
- bandpass, 171
- barycenter, 229
- basename, 419
- bayer2rgb, 119
- bilateral, 172
- bin, 419
- bin2dec, 419
- blend, 396
- blend_edges, 400
- blend_fade, 400
- blend_median, 401
- blend_seamless, 401
- blur, 173
- blur_angular, 174
- blur_linear, 175
- blur_radial, 175
- blur_selective, 176
- blur_x, 177
- blur_xy, 177
- blur_xyz, 178
- blur_y, 178
- blur_z, 179
- bokeh, 179

- box3d, 281
- boxfilter, 180
- boxfitting, 357
- break, 332
- bsl, 57
- bsr, 58
- camera, 19
- cartoon, 358
- center3d, 282
- channels, 146
- check, 331
- check3d, 332
- chessboard, 249
- ciel931, 249
- circle, 250
- circle3d, 282
- circles3d, 283
- cmy2rgb, 119
- cmyk2rgb, 120
- color3d, 283
- color_ellipses, 358
- color_med, 424
- colorcube3d, 284
- colormap, 120
- columns, 146
- command, 20
- complex2polar, 94
- compose_channels, 120
- compose_freq, 181
- compress_rle, 94
- cone3d, 284
- continue, 332
- convolve, 182
- convolve_fft, 183
- correlate, 183
- cos, 59
- cosh, 60
- cracks, 389
- crop, 147
- cross_correlation, 184
- cubes3d, 285
- cubism, 359
- cumulate, 95
- cup3d, 286
- cupid, 20
- cursor, 21
- curvature, 185
- cut, 95
- cylinder3d, 286
- dct, 185
- deblur, 186
- deblur_goldmeinel, 186
- deblur_richardsonlucy, 187
- debug, 19
- dec, 419
- dec2bin, 420
- dec2hex, 420
- dec2oct, 420
- dec2str, 419
- deconvolve_fft, 187
- deform, 379
- deinterlace, 188
- delaunay3d, 287
- demo, 428
- denoise, 188
- denoise_haar, 189
- deriche, 190
- detect_skin, 230
- diagonal, 148
- diffusientensors, 194
- dijkstra, 274
- dilate, 191
- dilate_circ, 191
- dilate_oct, 192
- dilate_threshold, 192
- direction2rgb, 121
- discard, 96
- displacement, 230
- display, 21
- display0, 21
- display3d, 22
- display_array, 22
- display_fft, 22
- display_graph, 22
- display_histogram, 23
- display_parametric, 24
- display_polar, 25
- display_rgba, 27

-display_tensors, 27
-display_warp, 28
-distance, 231
-distribution3d, 287
-ditheredbw, 121
-div, 61
-div3d, 288
-divergence, 193
-do, 333
-document_gmic, 29
-dog, 193
-done, 333
-double3d, 289
-draw_whirl, 359
-drawing, 360
-drop_shadow, 360
-echo, 29
-echo_file, 29
-echo_stdout, 30
-edges, 194
-eigen, 274
-eigen2tensor, 97
-elevate, 149
-elevation3d, 289
-elif, 334
-ellipse, 250
-ellipsionism, 361
-else, 334
-empty3d, 290
-endian, 97
-endif, 334
-endlocal, 334
-eq, 62
-equalize, 98
-erode, 195
-erode_circ, 196
-erode_oct, 196
-erode_threshold, 197
-error, 334
-euclidean2polar, 378
-exec, 334
-exp, 63
-expand_x, 149
-expand_xy, 150
-expand_xyz, 151
-expand_y, 151
-expand_z, 151
-extrude3d, 291
-fact, 420
-fade_diamond, 402
-fade_linear, 402
-fade_radial, 403
-fade_x, 403
-fade_y, 403
-fade_z, 404
-fft, 197
-fft82float, 232
-fftpolar, 232
-file_mv, 420
-file_rand, 420
-file_rm, 420
-filename, 421
-files, 421
-files2video, 406
-fill, 99
-fill_color, 122
-fire_edges, 362
-fisheye, 379
-fitratio_wh, 421
-fitscreen, 421
-float2fft8, 232
-float2int8, 100
-flood, 251
-flower, 380
-focale3d, 292
-fps, 421
-fractalize, 363
-frame_blur, 343
-frame_cube, 344
-frame_fuzzy, 345
-frame_painting, 345
-frame_pattern, 346
-frame_round, 347
-frame_x, 347
-frame_xy, 348
-frame_xyz, 349
-frame_y, 349
-function1d, 30

- gaussian, 252
- gaussians3d, 292
- gcd, 421
- ge, 64
- glow, 363
- gm3d, 293
- gmicky, 30
- gmicky.wilber, 31
- gradient, 198
- gradient2rgb, 122
- gradient_norm, 199
- gradient_orientation, 200
- graph, 253
- grid, 253
- gt, 65
- guided, 200
- gyroid3d, 293
- haar, 201
- halftone, 364
- hardsketchbw, 364
- head, 422
- heart, 31
- hearts, 365
- heat_flow, 201
- help, 19
- hessian, 202
- hex, 422
- hex2dec, 422
- hex2img8, 423
- hex2str, 422
- histogram, 233
- histogram3d, 294
- histogram_cumul, 234
- histogram_nd, 234
- histogram_pointwise, 235
- hough, 235
- houghsketchbw, 365
- hsi2rgb, 123
- hsi82rgb, 123
- hsl2rgb, 123
- hsl82rgb, 123
- hsv2rgb, 123
- hsv82rgb, 124
- idct, 202
- iee, 202
- if, 335
- ifft, 203
- ifftpolar, 236
- ihaar, 203
- image, 255
- image6cube3d, 294
- imageblocks3d, 295
- imagecube3d, 296
- imagegrid, 350
- imagegrid_hexagonal, 350
- imageplane3d, 296
- imagepyramid3d, 296
- imagerubik3d, 297
- imagesphere3d, 298
- img2ascii, 349
- img2str, 422
- img2text, 422
- img82hex, 423
- index, 100
- inn, 203
- inpaint, 204
- inpaint_flow, 205
- inpaint_holes, 206
- input, 32
- input_gpl, 34
- inrange, 101
- int2rgb, 124
- int82float, 100
- invert, 275
- is_3d, 423
- is_image_arg, 423
- is_percent, 423
- is_windows, 423
- isoline3d, 298
- isophotes, 236
- isosurface3d, 299
- kaleidoscope, 381
- keep, 44
- kuwahara, 207
- lab2lch, 124
- lab2rgb, 124
- lab82rgb, 125
- label, 237

-label3d, 300
-label_fg, 238
-labelpoints3d, 301
-laplacian, 207
-lathe3d, 301
-lch2lab, 125
-lch2rgb, 125
-lch82rgb, 125
-le, 66
-lic, 207
-light3d, 302
-light_patch, 389
-light_relief, 366
-lightrays, 366
-line, 255
-line3d, 303
-linearizetiles, 351
-lissajous3d, 303
-local, 335
-log, 68
-log10, 69
-log2, 70
-lt, 67
-luminance, 125
-mad, 423
-mandelbrot, 256
-map, 102
-map_clut, 103
-map_sphere, 381
-map_sprites, 351
-map_tones, 208
-map_tones_fast, 209
-marble, 257
-max, 71
-max_d, 423
-max_h, 423
-max_patch, 238
-max_s, 424
-max_w, 423
-max_wh, 424
-max_whd, 424
-max_whds, 424
-maze, 257
-maze_mask, 258
-mdiv, 72
-meancurvature_flow, 209
-med, 424
-median, 210
-min, 72
-min_d, 424
-min_h, 424
-min_patch, 239
-min_s, 424
-min_w, 424
-min_wh, 424
-min_whd, 424
-min_whds, 424
-minimal_path, 239
-mirror, 153
-mix_channels, 103
-mix_rgb, 125
-mmul, 74
-mod, 73
-mode3d, 304
-moded3d, 305
-montage, 152
-morph, 406
-mosaic, 367
-move, 45
-mse, 240
-mul, 75
-mul3d, 305
-mul_channels, 76
-mutex, 336
-name, 46
-names, 46
-negative, 104
-neq, 77
-nlmeans, 210
-nlmeans_core, 211
-noarg, 337
-noise, 104
-noise_hurl, 390
-norm, 105
-normalize, 106
-normalize3d, 306
-normalize_filename, 425
-normalize_local, 211

- normalize_sum, 107
- normalized_cross_correlation, 212
- not, 107
- object3d, 259
- oct, 425
- oct2dec, 425
- old_photo, 368
- oneminus, 108
- onfail, 337
- opacity3d, 306
- or, 78
- orientation, 107
- otsu, 108
- output, 34
- output_ggr, 34
- output_pink3d, 410
- outputn, 35
- outputp, 35
- outputw, 35
- outputx, 35
- pack, 352
- pack_sprites, 259
- padint, 425
- parallel, 337
- parametric3d, 307
- pass, 35
- patches, 241
- path_gimp, 425
- path_prc, 425
- path_rc, 426
- path_tmp, 425
- pca_patch3d, 308
- pde_flow, 213
- pencilbw, 368
- periodize_poisson, 214
- permute, 153
- peronamalik_flow, 212
- phase_correlation, 213
- piechart, 260
- pink, 410
- pink_grayskel, 411
- pink_heightmaxima, 412
- pink_heightminima, 412
- pink_htkern, 413
- pink_lvkern, 413
- pink_regminima, 414
- pink_skelcurv, 415
- pink_skelend, 415
- pink_skeleton, 416
- pink_skelpar, 416
- pink_wshed, 417
- pixelize, 390
- plane3d, 309
- plasma, 261
- plot, 36
- plot2value, 241
- point, 262
- point3d, 309
- pointcloud, 241
- pointcloud3d, 310
- polar2complex, 109
- polar2euclidean, 382
- polaroid, 368
- polka_dots, 263
- polygon, 263
- polygonize, 369
- pose3d, 310
- poster_edges, 370
- poster_hope, 371
- pow, 79
- primitives3d, 311
- print, 37
- progress, 338
- projections3d, 312
- pseudogray, 126
- psnr, 243
- puzzle, 353
- pyramid3d, 312
- quadrangle3d, 312
- quadratize_tiles, 354
- quantize, 109
- quit, 338
- quiver, 264
- quote, 426
- rainbow_lut, 37
- raindrops, 382
- rand, 110

- rectangle, 265
- redeye, 214
- region.feature, 426
- register_nonrigid, 407
- register_rigid, 408
- remove, 46
- remove_duplicates, 47
- remove_empty, 48
- remove_hotpixels, 215
- remove_opacity, 136
- remove_pixels, 215
- repeat, 338
- replace, 111
- replace_color, 127
- replace_inf, 111
- replace_nan, 112
- replace_seq, 112
- replace_str, 113
- reset, 426
- resize, 154
- resize2dx, 156
- resize2dy, 157
- resize3dx, 158
- resize3dy, 159
- resize3dz, 159
- resize_pow2, 155
- resize_ratio2d, 156
- return, 339
- reverse, 48
- reverse3d, 313
- rgb2bayer, 127
- rgb2cmy, 128
- rgb2cmyk, 128
- rgb2hsi, 129
- rgb2hsi8, 130
- rgb2hsl, 130
- rgb2hsl8, 131
- rgb2hsv, 131
- rgb2hsv8, 131
- rgb2int, 134
- rgb2lab, 132
- rgb2lab8, 132
- rgb2lch, 133
- rgb2lch8, 133
- rgb2luv, 133
- rgb2srgb, 134
- rgb2xyz, 134
- rgb2xyz8, 135
- rgb2ycbcr, 135
- rgb2yuv, 135
- rgb2yuv8, 136
- ripple, 383
- rodgy, 37
- rodilius, 371
- rol, 80
- ror, 81
- rorschach, 266
- rotate, 160
- rotate3d, 313
- rotate_tileable, 160
- rotate_tiles, 354
- rotation3d, 314
- rotoidoscope, 384
- round, 114
- roundify, 114
- rows, 161
- rprogress, 340
- scale2x, 161
- scale3x, 162
- scanlines, 391
- seamcarve, 162
- segment_watershed, 243
- select, 37
- select_color, 136
- sepia, 137
- serialize, 38
- set, 115
- shade_stripes, 392
- shadow_patch, 392
- shared, 39
- sharpen, 216
- shift, 163
- shift_tiles, 355
- shrink_x, 164
- shrink_xy, 164
- shrink_xyz, 165
- shrink_y, 165
- shrink_z, 165

- sierpinski, 267
- sierpinski3d, 314
- sign, 81
- sin, 82
- sinc, 83
- sinh, 84
- size3d, 315
- skeleton, 244
- skeleton3d, 315
- sketchbw, 374
- skip, 340
- slices, 165
- smooth, 217
- snapshot3d, 316
- snowflake, 268
- solarize, 137
- solidify, 219
- solidify_linear, 219
- solidify_watershed, 220
- solve, 275
- solve_poisson, 220
- sort, 166
- sort_list, 49
- sort_str, 50
- specl3d, 317
- specs3d, 318
- sphere3d, 318
- spherical3d, 319
- spiralbw, 268
- spline, 269
- spline3d, 320
- split, 166
- split3d, 320
- split_colors, 138
- split_details, 221
- split_freq, 218
- split_opacity, 139
- split_tiles, 168
- sponge, 375
- spread, 393
- sprite3d, 321
- sprites3d, 321
- sqr, 85
- sqrt, 86
- srand, 40
- srgb2rgb, 139
- ssd_patch, 244
- stained_glass, 372
- star, 372
- star3d, 322
- stars, 373
- status, 340
- std_noise, 428
- stencil, 375
- stencilbw, 376
- str, 427
- str2hex, 427
- strcat, 427
- strcmp, 427
- streamline3d, 322
- stresc, 427
- stripes_y, 393
- strlen, 427
- strreplace, 428
- structuretensors, 222
- struncase, 428
- strver, 428
- sub, 87
- sub3d, 323
- sub_alpha, 404
- superformula3d, 324
- svd, 276
- symmetrize, 384
- syntexturize, 223
- tan, 89
- tanh, 90
- taquin, 355
- testimage2d, 41
- tetris, 376
- text, 269
- text3d, 325
- text_outline, 270
- text_pointcloud3d, 325
- texturize3d, 326
- texturize_canvas, 394
- texturize_paper, 394
- thinning, 245
- threshold, 116

-tic, 428	-wait, 43
-to_a, 139	-warhol, 377
-to_color, 139	-warn, 43
-to_colormode, 139	-warp, 170
-to_gray, 139	-warp_perspective, 386
-to_graya, 139	-water, 386
-to_pseudogray, 140	-watermark_fourier, 226
-to_rgb, 140	-watermark_visible, 395
-to_rgba, 140	-watershed, 227
-toc, 428	-wave, 387
-tones, 245	-weave, 377
-topographic_map, 246	-weird3d, 328
-torus3d, 327	-while, 340
-transfer_colors, 140	-whirls, 378
-transform_polar, 385	-wind, 388
-transition, 408	-window, 43
-transition3d, 409	-x_2048, 428
-transpose, 276	-x_blobs, 428
-triangle3d, 327	-x_bouncing, 429
-triangle_shade, 271	-x_color_curves, 429
-trisolve, 276	-x_colorize, 429
-truchet, 272	-x_fire, 429
-tunnel, 356	-x_fireworks, 429
-turbulence, 272	-x_fisheye, 429
-tv_flow, 223	-x_fourier, 429
-twirl, 385	-x_histogram, 429
-uncommand, 41	-x_hough, 430
-uncompress_rle, 117	-x_jawbreaker, 430
-uniform_distribution, 41	-x_landscape, 430
-unrepeat, 117	-x_life, 430
-unroll, 169	-x_light, 430
-unserialize, 42	-x_mandelbrot, 430
-unsharp, 224	-x_metaballs3d, 430
-unsharp_octave, 224	-x_minesweeper, 430
-update, 42	-x_minimal_path, 430
-upscale_smart, 170	-x_pacman, 430
-vanvliet, 225	-x_paint, 430
-variance_patch, 246	-x_plasma, 431
-vector2tensor, 117	-x_quantize_rgb, 431
-verbose, 42	-x_reflection3d, 431
-version, 19	-x_rubber3d, 431
-video2files, 409	-x_segment, 431
-vignette, 395	-x_select_color, 431
-volume3d, 328	-x_select_function1d, 431

- x_select_palette, 432
- x_shadebobs, 432
- x_spline, 432
- x_tetris, 432
- x_tictactoe, 432
- x_waves, 432
- x_whirl, 433
- xor, 91
- xyz2rgb, 140
- xyz82rgb, 141
- ycbcr2rgb, 141
- yinyang, 273
- yuv2rgb, 141
- yuv82rgb, 141
- zoom, 388

□ End of document.