



A Full-Featured Open-Source Framework for Image Processing

The Handbook

Version 2.5.2

David Tschumperlé

March 15, 2019

Contents

1	Usage	7
1.1	Overall context	7
1.2	Image definition and terminology	8
1.3	Items of a processing pipeline	8
1.4	Input data items	9
1.5	Command items and selections	10
1.6	Input/output properties	11
1.7	Substitution rules	13
1.8	Mathematical expressions	15
1.9	Image and data viewers	21
1.10	Adding custom commands	22
1.11	List of commands	23
2	List of commands	25
2.1	Global Options	25
2.2	Input / Output	25
2.3	List Manipulation	55
2.4	Mathematical Operators	60
2.5	Values Manipulation	98
2.6	Colors Manipulation	125
2.7	Geometry Manipulation	155
2.8	Filtering	185
2.9	Features Extraction	241
2.10	Image Drawing	262
2.11	Matrix Computation	286
2.12	3D Rendering	290
2.13	Control Flow	338
2.14	Arrays, Tiles and Frames	349
2.15	Artistic	365
2.16	Warpings	388
2.17	Degradations	399
2.18	Blending and Fading	406
2.19	Image Sequences and Videos	415
2.20	Convenience Functions	422
2.21	Other Interactive Commands	434
2.22	Command shortcuts	439
2.23	Examples of use	441

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:
<https://gmic.eu/reference.shtml>.

Motivations

G'MIC is a full-featured open-source framework for image processing, distributed under the CeCILL free software licenses (LGPL-like and/or GPL-compatible). It provides several user interfaces to convert/manipulate/filter/visualize generic image datasets, ranging from 1D scalar signals to 3D+t sequences of multi-spectral volumetric images, hence including 2D color images.

G'MIC user interfaces are:

- 'gmic', a command-line tool to use the **G'MIC** image processing features from a shell. In this setting, **G'MIC** may be seen as a friendly companion to the ImageMagick or GraphicsMagick software suites.
- 'libgmic', a small, portable, thread-safe and multi-threaded, C++ image processing library to be linked to third-party applications. It's simple API allows programmers to add all **G'MIC** features in their own software without much efforts (a C API is available as well).
- 'G'MIC-Qt', a plug-in to bring **G'MIC** capabilities to the image retouching and painting software GIMP and Krita. More than 500 filters are already available, sorted by category (Artistic, Black & white, Colors, Contours, Deformations, Degradations, Details, Color Grading, Frames, Layers, Light & shadows, Patterns, Rendering, Repair, Sequences, etc.).
- 'G'MIC Online', a web service to allow users applying image processing algorithms on their images, directly from a web browser.
- 'ZArt', a Qt-based interface for real-time processing of video streaming coming from webcams or video files.

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 partly under the CeCILL licenses (CeCILL-C and/or CeCILL). 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 **2.5.2**, Copyright (c) 2008-2019, David Tschumperlé
(<https://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 possibly complex image processing pipelines and operators. It can be used to convert, manipulate, filter and visualize image datasets made of one or several 1D/2D or 3D multi-spectral images.

This reference documentation describes all the technical rules governing the **G'MIC** language. As a starting point, you may want to visit our detailed tutorial pages, at:

<https://gmic.eu/tutorial/>

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 (uncompressed).
- The first image of the list has index '0' and is denoted by '[0]'. The second image of the list is denoted by '[1]', the third by '[2]' and so on.
- Negative indices are treated in a periodic way: '[-1]' refers to the last image of the list, '[-2]' to the penultimate one, etc. Thus, if the list has 4 images, '[1]' and '[-3]' both designate the second image of the list.
- A named image may be also indicated by '[name]', if 'name' uses the character set [a-zA-Z0-9_] and does not start with a number. Image names can be set or reassigned at any moment during the processing pipeline (see command 'name' for this purpose).
- **G'MIC** defines a set of various commands and substitution mechanisms to allow the design of complex pipelines and operators managing this list of images, in a very flexible way: You can insert or remove images in the list, rearrange image order, process images (individually or grouped), merge image data together, display and output image files, etc.
- Such a pipeline can be then added as a new custom **G'MIC** command (stored in a user command file), so it can be re-used afterwards in a larger 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 hard limitations on the size of the image along each dimension. For instance, the number of image slices or channels can be of arbitrary size within the limits of the available memory.
- The **width**, **height** and **depth** of an image are considered as spatial dimensions, while the **spectrum** has a multi—spectral meaning. Thus, a 4D image in **G'MIC** should be most often regarded as a 3D dataset of multi—spectral voxels. Most of the **G'MIC** commands will stick with this idea (e.g. command **'blur'** blurs images only along the spatial **'xyz'—axes**).
- **G'MIC** stores all the image data as buffers of **'float'** values (32 bits, value range **[−3.4E38,+3.4E38]**). It performs all its image processing operations with floating point numbers. Each image pixel takes then 32bits/channel (except if double—precision buffers have been enabled during the compilation of the software, in which case 64bits/channel can be the default).
- Considering **'float'**—valued pixels ensure to keep the numerical precision when executing image processing pipelines. For image input/output operations, you may want to prescribe the image datatype to be different than **'float'** (like **'bool'**, **'char'**, **'int'**, etc...). This is possible by specifying it as a file option when using I/O commands. (see section **'Input/output properties'** to learn more about file options).

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 file_out.jpg
```

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

- Each **G'MIC** item is a string that is either a **command**, a list of command **arguments**, a **filename**, or a special **input string**.

- Escape characters `'\'` and double quotes `''''` can be used to define items containing spaces or other special characters. For instance, the two strings `'single\ item'` and `''single item''` both define the same single item, with a space in it.

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 (which is equivalent to the use of `'input filename'`).
- Special filenames `'-'` and `'-.ext'` stand for the standard input/output streams, optionally forced to be in a specific `'ext'` file format (e.g. `'-.jpg'` or `'-.png'`).
- The following special input strings may be used as **G'MIC** items to create and insert new images with prescribed values, at the end of the image list:
 - . `'[selection]'` or `'[selection]xN'`: Insert 1 or N copies of already existing images. `'selection'` may represent one or several images (see section ['Command items and selections'](#) to learn more about selections).
 - . `'width[%],height[%],depth[%],spectrum[%],values'`: 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,[x,y,128]'` which creates a 256x256 RGB color image with a spatial shading on the red and green channels. (see section ['Mathematical expressions'](#) to learn more about mathematical expressions).
 - . `'(v1,v2,...)'`: 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 variable `'name'`, or assign a new value to an existing variable. Variable names must use the character set `[a-zA-Z0-9_]` and cannot start with a number.
- A variable definition is always local to the current command except when it starts by the underscore character `'_'`. In that case, it becomes also accessible by any command invoked outside the current command scope (global variable).
- If a variable name starts with two underscores `'__'`, the global variable is also shared among different threads and can be read/set by commands running in parallel (see command `'parallel'` for this purpose). Otherwise, it remains local to the thread that defined it.
- Numerical variables can be updated with the use of these special operators:
 - `'+='` (addition), `'-='` (subtraction), `'*='` (multiplication), `'/='` (division), `'%='` (modulo),

'&=' (bitwise and), '|=' (bitwise or), '^=' (power), '<<=' and '>>=' (bitwise left and right shifts). For instance, 'foo=1 foo+=3'.

- Input item 'name.=string' concatenates specified 'string' to the end of variable 'name'.
- Multiple variable assignments and updates are allowed, with expressions: 'name1,name2,...,nameN=value' or 'name1,name2,...,nameN=value1,value2,...,valueN' where assignment operator '=' can be replaced by one of the allowed operators (e.g. '+=').

1.5 Command items and selections

- A G'MIC item that is not a filename nor a special input string designates a **command**, most of the time. Generally, commands perform image processing operations on one or several available images of the list.
- Recurrent commands have two equivalent names (**regular** and **short**). For instance, command names 'resize' and 'r' refer to the same image resizing action.
- A G'MIC command may have mandatory or optional **arguments**. Command arguments must be specified in the next item on the command line. Commas ',' are used to separate multiple arguments of a single command, when required.
- The execution of a G'MIC command may be restricted only to a **subset** of the image list, by appending '[selection]' to the command name. Examples of valid syntaxes for 'selection' are:
 - . 'command[-2]': Apply command only on the penultimate image [-2] of the list.
 - . 'command[0,1,3]': Apply command only on images [0],[1] and [3].
 - . 'command[3-6]': Apply command only on images [3] to [6] (i.e. [3],[4],[5] and [6]).
 - . 'command[50%-100%]': Apply command only on the second half of the image list.
 - . 'command[0,-4--1]': Apply command only on the first image and the last four images.
 - . 'command[0-9:3]': Apply command only on images [0] to [9], with a step of 3 (i.e. on images [0], [3], [6] and [9]).
 - . 'command[0-9:25%]': Apply command only on images [0] to [9], with a step of 25% (i.e. on images [0], [3], [6] and [9]).
 - . 'command[0--1:2]': Apply command only on images of the list with even indices.
 - . 'command[0,2-4,50%--1]': Apply command on images [0],[2],[3],[4] and on the second half of the image list.
 - . 'command[^0,1]': Apply command on all images except the first two.
 - . 'command[name1,name2]': Apply command on named images 'name1' and 'name2'.
- indices in selections are always sorted in increasing order, and duplicate indices are discarded. For instance, selections '[3-1,1-3]' and '[1,1,1,3,2]' are both equivalent to '[1-3]'. If you want to repeat a single command multiple times on an image, use a 'repeat..done' loop instead. Inverting the order of images for a command is achieved by explicitly inverting the order of the images in the list, with command 'reverse[selection]'.
- Command selections '[-1]', '[-2]' and '[-3]' are so often used that they have their own shortcuts, respectively '.', '..' and '...'. For instance, command 'blur..' is equivalent to 'blur[-2]'. These shortcuts work also when specifying command arguments.

- **G'MIC** commands invoked without '[selection]' are applied on all images of the list, i.e. the default selection is '[0--1]' (except for command 'input' whose default selection is '[-1]').
- Prepending a single hyphen '-' to a **G'MIC** command is allowed. This may be useful to recognize command items more easily in a one-liner pipeline (typically invoked from a shell).
- A **G'MIC** command prepended with a plus sign '+' or a double hyphen '--' 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:
 - . **Builtin 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 builtin command is not permitted, except if all following arguments are also omitted. For instance, invoking 'plasma 10,,5' is invalid but 'plasma 10' is correct.
 - . **Custom commands**, are defined as **G'MIC** pipelines of builtin or other custom commands. They are interpreted by the **G'MIC** interpreter, and thus run a bit slower than builtin commands. Omitting arguments when invoking a custom command is permitted. For instance, expressions 'flower ,,,100,,2' or 'flower ,,' are correct.
- Most of the existing commands in **G'MIC** are actually defined as **custom commands**.
- A user can easily add its own custom commands to the **G'MIC** interpreter (see section 'Adding custom commands' for more details). New builtin commands cannot be added (unless you modify the **G'MIC** interpreter source code and recompile it).

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, .dln, .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 then processed as regular images. (see command 'split3d' for accessing each of these 3D object data separately).
- Be aware that usual file formats may be sometimes not adapted to store all the available image data, since **G'MIC** uses float-valued image buffers. For instance, saving an image that was initially loaded as a 16bits/channel image, as a .jpg file will result in a loss of information. Use the **G'MIC**-specific file extension .gmz to ensure that all data precision is preserved when saving images.

– Sometimes, file options may/must be set for file formats:

- . **Video files:** Only sub—frames of an image sequence may be loaded, using the input expression `'filename.ext,[first_frame[,last_frame[,step]]]`. Set `'last_frame==−1'` to tell it must be the last frame of the video. Set `'step'` to 0 to force an opened video file to be opened/closed. Output framerate and codec can be also set by using the output expression `'filename.avi,_fps,_codec,_keep_open={ 0 | 1 }'`. `'codec'` is a 4—char string (see <http://www.fourcc.org/codecs.php>) or `'0'` for the default codec. `'keep_open'` tells if the output video file must be kept open for appending new frames afterwards.
- . **.cimg[z] files:** Only crops and sub—images of .cimg files can be loaded, using the input expressions `'filename.cimg,N0,N1'`, `'filename.cimg,N0,N1,x0,x1'`, `'filename.cimg,N0,N1,x0,y0,x1,y1'`, `'filename.cimg,N0,N1,x0,y0,z0,x1,y1,z1'` or `'filename.cimg,N0,N1,x0,y0,z0,c0,x1,y1,z1,c1'`. Specifying `'−1'` for one coordinates stands for the maximum possible value. Output expression `'filename.cimg[z],[datatype]'` can be used to force the output pixel type. `'datatype'` can be `{ auto | uchar | char | ushort | short | uint | int | uint64 | int64 | float | double }`.
- . **.raw binary files:** Image dimensions and input pixel type may be specified when loading .raw files with input expression `'filename.raw,[datatype],[width],[height],[depth],[dim],[offset]]]`. If no dimensions are specified, the resulting image is a one—column vector with maximum possible height. Pixel type can also be specified with the output expression `'filename.raw,[datatype]'`. `'datatype'` can be the same as for .cimg[z] files.
- . **.yuv files:** Image dimensions must be specified when loading, and only sub—frames of an image sequence may be loaded, using the input expression `'filename.yuv,width,height[,chroma_subsampling[,first_frame[,last_frame[,step]]]'`. `'chroma_subsampling'` can be `{ 420 | 422 | 444 }`. When saving, chroma subsampling mode can be specified with output expression `'filename.yuv[,chroma_subsampling]'`.
- . **.tiff files:** Only sub—images of multi—pages tiff files can be loaded, using the input expression `'filename.tif,_first_frame,_last_frame,_step'`. Output expression `'filename.tiff,_datatype,_compression,_force_multipage,_use_bigtiff'` can be used to specify the output pixel type, as well as the compression method. `'datatype'` can be the same as for .cimg[z] files. `'compression'` can be `{ none (default) | lzw | jpeg }`. `'force_multipage'` can be `{ 0=no (default) | 1=yes }`. `'use_bigtiff'` can be `{ 0=no | 1=yes (default) }`.
- . **.gif files:** Animated gif files can be saved, using the input expression `'filename.gif,fps>0,nb_loops'`. Specify `'nb_loops=0'` to get an infinite number of animation loops (this is the default behavior).
- . **.jpeg files:** The output quality may be specified (in %), using the output expression `'filename.jpg,30'` (here, to get a 30% quality output). `'100'` is the default.
- . **.mnc files:** The output header can set from another file, using the output expression `'filename.mnc,header_template.mnc'`.
- . **.pan, .cpp, .hpp, .c and .h files:** The output datatype can be selected with output expression `'filename,[datatype]'`. `'datatype'` can be the same as for .cimg[z] files.
- . **.gmic files:** These filenames are assumed to be G'MIC custom commands files. Loading such a

file will add the commands it defines to the interpreter. Debug information can be enabled/disabled by the input expression `'filename.gmic[add_debug_info={ 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 `'ext'`.
- Some input/output formats and options may not be supported, depending on the configuration flags that have been set during the build of the **G'MIC** software.

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 **index** of the named image `'[name]'`. If no image has this name, the expression is substituted by the value of the **OS environment variable** with same name (it may be thus an empty string).

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

- `'$!'`: The current number of images in the list.
- `'$>'` and `'$<'`: The increasing/decreasing index of the latest (currently running) `'repeat...done'` loop. `'$>'` goes from `'0'` (first loop iteration) to `'nb_iterations - 1'` (last iteration). `'$>'` does the opposite.
- `'$/'`: The current call stack. Stack items are separated by slashes `'/'`.
- `'$|'`: The current value (expressed in seconds) of a millisecond precision timer.
- `'$~'`: The current verbosity level.
- `'$_cpus'`: The number of computation cores available on your machine.
- `'$_pid'`: The current process identifier, as an integer.
- `'$_prerelease'`: For pre-releases only, the date of the pre-release as `'yymmdd'`. For stable releases, this variable is not defined.
- `'$_version'`: A 3-digits number telling about the current version of the **G'MIC** interpreter (e.g. `'252'`).
- `'$_vt100'`: Set to `1` (default value) if colored text output is allowed on the console.
- `'$_path_rc'`: The path to the **G'MIC** folder used to store resources and configuration files (its value is OS-dependent).
- `'$_path_user'`: The path to the **G'MIC** user file `.gmic` or `user.gmic` (its value is OS-dependent).
- `'$$name'` and `'${$name}'` are both substituted by the **G'MIC** script code of the specified named **custom command**, or by an empty string if no custom command with specified name exists.
- `'${"-pipeline"}'` is substituted by the **status value** after the execution of the specified **G'MIC** pipeline (see command `'status'`). Expression `'${}'` thus stands for the **current status value**.
- `'{"string"}'` (starting with two backquotes) is substituted by a **double-quoted version** of the specified string.
- `'{/string}'` is substituted by the **escaped version** of the specified string.

- ‘{string}’ (between 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’.
- ‘{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).
- . ‘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 the characters after the last ‘.’ in the image name).
- . ‘^’: The sequence of all image values, separated by commas ‘,’.
- . ‘@subset’: The sequence of image values corresponding to the specified subset, and separated by commas ‘,’.
- . Any other ‘feature’ is considered as a **mathematical expression** associated to the image [image] and is substituted by the result of its evaluation (float value). For instance, expression ‘{0,w+h}’ is substituted by the sum of the width and height of the first image (see section ‘Mathematical expressions’ for more details). If a mathematical expression starts with an underscore ‘_’, the resulting value is truncated to a readable format. For instance, item ‘{_pi}’ is substituted by ‘3.14159’ (while ‘{pi}’ is substituted by ‘3.141592653589793’).
- . A ‘feature’ delimited by backquotes is replaced by a string whose ascii codes correspond to the list of values resulting from the evaluation of the specified mathematical expression. For instance, item ‘{‘[102,111,111]’}’ is substituted by ‘foo’ and item ‘{‘vector8(65)’}’ by ‘AAAAAAA’.

- ‘{*}’ is substituted by the **visibility state** of the instant display window [0] (can be { 0=closed | 1=visible }).
- ‘{*feature1,...,featureN}’ or ‘{*index,feature1,...,featureN}’ is substituted by a specific set of features of the instant display window #0 (or #index, if specified). Requested ‘features’ 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.
- . ‘t’: window title of the instant display.
- . ‘x’: X-coordinate of the mouse position (or -1, if outside the display area).
- . ‘y’: Y-coordinate of the mouse position (or -1, if outside the display area).
- . ‘b’: state of the mouse buttons { 1=left-but. | 2=right-but. | 4=middle-but. }.
- . ‘o’: state of the mouse wheel.
- . ‘k’: decimal code of the pressed key if any, 0 otherwise.
- . ‘c’: boolean (0 or 1) telling if the instant display has been closed recently.

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

1.8 Mathematical expressions

- **G'MIC** has an embedded **mathematical parser**. It is used to evaluate (possibly complex) expressions inside braces '{}', or formulas in commands that may take one as an argument (e.g. 'fill').
- When the context allows it, a formula is evaluated **for each pixel** of the selected images (e.g. 'fill').
- A math expression may return a **scalar** or **vector**—valued result (with a fixed number of components).
- The mathematical parser understands the following set of functions, operators and variables:
 - **Usual operators:** || (logical or), && (logical and), | (bitwise or), & (bitwise and), !=, ==, <=, >=, <, >, << (left bitwise shift), >> (right bitwise shift), -, +, *, /, % (modulo), ^ (power), ! (logical not), ~ (bitwise not), ++, --, +=, -=, *=, /=, %=, &=, |=, ^=, >>=, <<= (in-place operators).
 - **Usual math functions:** abs(), acos(), acosh(), arg(), argkth(), argmax(), argmin(), asin(), asinh(), atan(), atan2(), atanh(), avg(), bool(), cbrt(), ceil(), cos(), cosh(), cut(), exp(), fact(), fibo(), floor(), gauss(), int(), isval(), isnan(), isinf(), isint(), isbool(), isfile(), isdir(), isin(), kth(), log(), log2(), log10(), max(), med(), min(), narg(), prod(), rol() (left bit rotation), ror() (right bit rotation), round(), sign(), sin(), sinc(), sinh(), sqrt(), std(), srand(_seed), sum(), tan(), tanh(), var(), xor().
- . 'atan2(y,x)' is the version of 'atan()' with two arguments 'y' and 'x' (as in C/C++).
- . 'permut(k,n,with_order)' computes the number of permutations of **k** objects from a set of **n** objects.
- . 'gauss(x,_sigma,_is_normalized)' returns `'exp(-x^2/(2*s^2))/(is_normalized?sqrt(2*pi*s^2):1)'`.
- . 'cut(value,min,max)' returns value if it is in range [min,max], or min or max otherwise.
- . 'narg(a_1,...,a_N)' returns the number of specified arguments (here, **N**).
- . 'arg(i,a_1,...,a_N)' returns the **i**th argument **a.i**.
- . 'isval()', 'isnan()', 'isinf()', 'isint()', 'isbool()' test the type of the given number or expression, and return 0 (false) or 1 (true).
- . 'isfile('path')' (resp. 'isdir('path')') returns 0 (false) or 1 (true) whether its string argument is a path to an existing file (resp. to a directory) or not.

- . `'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'.
- . `'argmin()'`, `'argmax()'`, `'avg()'`, `'kth()'`, `'max()'`, `'med()'`, `'min()'`, `'std()'`, `'sum()'` and `'var()'` can be called with an arbitrary number of scalar/vector arguments.
- . `'round(value,rounding_value,direction)'` returns a rounded value. 'direction' can be `{ -1=to-lowest | 0=to-nearest | 1=to-highest }`.

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

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

value is 0.

- `'boundary'`: value of the default boundary conditions used when reading pixel values with the pixel access operators (i.e. when the boundary condition argument is not explicitly specified, see below for more details on pixel access operators). Its initial default value is 0.

– **Vector calculus**: Most operators are also able to work with vector-valued elements.

- `'[a0,a1,...,aN-1]'` defines a N -dimensional vector with scalar coefficients ak .
- `'vectorN(a0,a1,...,aN-1)'` does the same, with the ak being repeated periodically if only a few are specified.
- `'vector(#N,a0,a1,...,aN-1)'` does the same, and can be used for any constant expression N .
- In previous expressions, the ak can be vectors themselves, to be concatenated into a single vector.
- The scalar element ak of a vector X is retrieved by `'X[k]'`.
- The sub-vector `[X[p],X[p+s]...X[p+s*(q-1)]]` (of size q) of a vector X is retrieved by `'X[p,q,s]'`.
- Equality/inequality comparisons between two vectors is done with operators `'=='` and `'!='`.
- Some vector-specific functions can be used on vector values:
 - `'cross(X,Y)'` (cross product), `'dot(X,Y)'` (dot product), `'size(X)'` (vector dimension),
 - `'sort(X,_is_increasing,_chunk_size)'` (sorting values), `'reverse(A)'` (reverse order of components), `'shift(A,_length,_boundary_conditions)'` and
 - `'same(A,B,_nb_vals,_is_case_sensitive)'` (vector equality test).
- Function `'normP(u1,...,un)'` computes the LP-norm of the specified vector (P being an unsigned integer constant or `'inf'`). If P is omitted, the L2 norm is used.
- Function `'resize(A,size,_interpolation,_boundary_conditions)'` returns a resized version of a vector A with specified interpolation mode. `'interpolation'` can be `{ -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }`, and `'boundary_conditions'` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.
- Function `'find(A,B,_is_forward,_starting_index)'` returns the index where sub-vector B appears in vector A , (or `-1` if B is not found in A). Argument A can be also replaced by an image index `#ind`.
- A 2-dimensional vector may be seen as a complex number and used in those particular functions/operators:
 - `'**'` (complex multiplication), `'/'` (complex division), `'^^'` (complex exponentiation),
 - `'**='` (complex self-multiplication), `'/='` (complex self-division), `'^^='` (complex self-exponentiation), `'cabs()'` (complex modulus), `'carg()'` (complex argument), `'cconj()'` (complex conjugate), `'cexp()'` (complex exponential) and `'clog()'` (complex logarithm).
- A MN -dimensional vector may be seen as a $M \times N$ matrix and used in those particular functions/operators:
 - `'*'` (matrix-vector multiplication), `'det(A)'` (determinant), `'diag(V)'` (diagonal matrix from a vector), `'eig(A)'` (eigenvalues/eigenvectors), `'eye(n)'` ($n \times n$ identity matrix),
 - `'inv(A)'` (matrix inverse), `'mul(A,B,_nb_colsB)'` (matrix-matrix multiplication),
 - `'pseudoinv(A,_nb_colsA)'`, `'rot(u,v,w,angle)'` (3D rotation matrix), `'rot(angle)'` (2D rotation matrix), `'solve(A,B,_nb_colsB)'` (least-square solver of linear system $A.X = B$),
 - `'svd(A,_nb_colsA)'` (singular value decomposition), `'trace(A)'` (matrix trace) and
 - `'transp(A,_nb_colsA)'` (matrix transpose). Argument `'nb_colsB'` may be omitted if it is equal to 1.
- Specifying a vector-valued math expression as an argument of a command that operates on image values (e.g. `'fill'`) modifies the whole spectrum range of the processed image(s), for each spatial coordinates (x,y,z) . The command does not loop over the C -axis in this case.

– **String manipulation**: Character strings are defined and managed as vectors objects.

Dedicated functions and initializers to manage strings are

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

_ **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 (vectors or scalars). Variable names are case-sensitive. Use this operator in conjunction with `';` to define more complex evaluable expressions, such as `'t=cos(x);3*t^2+2*t+1'`.
These variables remain **local** to the mathematical parser and cannot be accessed outside the evaluated expression.
- . Variables defined in math parser may have a **constant** property, by specifying keyword `'const'` before the variable name (e.g. `'const foo = pi/4;'`). The value set to such a variable must be indeed a **constant scalar**. Constant variables allows certain types of optimizations in the math JIT compiler.

_ The following **specific functions** are also defined:

- . `'u(max)'` or `'u(min,max)'`: return a random value between `[0,max]` or `[min,max]`, following a uniform distribution.
- . `'i(_a,_b,_c,_d,_interpolation_type,_boundary_conditions)'`: return the value of the pixel located at position `(a,b,c,d)` in the associated image, if any (`0` otherwise).
`'interpolation_type'` can be `{ 0=nearest neighbor | other=linear }`.
`'boundary_conditions'` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.
Omitted coordinates are replaced by their default values which are respectively `x`, `y`, `z`, `c`, `interpolation` and `boundary`.
For instance command `'fill 0.5*(i(x+1)-i(x-1))'` will estimate the X-derivative of an image with a classical finite difference scheme.
- . `'j(_dx,_dy,_dz,_dc,_interpolation_type,_boundary_conditions)'` does the same for the pixel located at position `(x+dx,y+dy,z+dz,c+dc)` (pixel access relative to the current coordinates).
- . `'i[offset,_boundary_conditions]'` returns the value of the pixel located at specified `'offset'` in the associated image buffer (or `0` if offset is out-of-bounds).
- . `'j[offset,_boundary_conditions]'` does the same for an offset relative to the current pixel coordinates `(x,y,z,c)`.
- . `'i(#ind,_x,_y,_z,_c,_interpolation,_boundary_conditions)'`,

- 'j(#ind,_dx,_dy,_dz,_dc,_interpolation,_boundary_conditions)',
 - 'i[#ind,offset,_boundary_conditions]' and 'i[offset,_boundary_conditions]' are similar expressions used to access pixel values for any numbered image [ind] of the list.
- 'I/J[offset,_boundary_conditions]' and 'I/J(#ind,_x,_y,_z,_interpolation,_boundary_conditions)' do the same as 'i/j[offset,_boundary_conditions]' and
 - 'i/j(#ind,_x,_y,_z,_c,_interpolation,_boundary_conditions)' but return a vector instead of a scalar (e.g. a vector [R,G,B] for a pixel at (a,b,c) in a color image).
- 'sort(#ind,_is_increasing,_axis)' sorts the values in the specified image [ind].
- 'crop(_#ind,_x,_y,_z,_c,_dx,_dy,_dz,_dc,_boundary_conditions)' returns a vector whose values come from the cropped region of image [ind] (or from default image selected if 'ind' is not specified). Cropped region starts from point (x,y,z,c) and has a size of dx x dy x dz x dc. Arguments for coordinates and sizes can be omitted if they are not ambiguous (e.g. 'crop(#ind,x,y,dx,dy)' is a valid invocation of this function).
- 'draw(_#ind,S,x,y,z,c,dx,_dy,_dz,_dc,_opacity,_M,_max_M)' draws a sprite S in image [ind] (or in default image selected if 'ind' is not specified) at coordinates (x,y,z,c). The size of the sprite dx x dy x dz x dc must be specified. You can also specify a corresponding opacity mask M if its size matches S.
- 'polygon(_#ind,nb_vertices,coords,_opacity,_color)' draws a filled polygon in image [ind] (or in default image selected if 'ind' is not specified) at specified coordinates. It draws a single line if 'nb_vertices' is set to 2.
- 'polygon(_#ind,-nb_vertices,coords,_opacity,_pattern,_color)' draws a outlined polygon in image [ind] (or in default image selected if 'ind' is not specified) at specified coordinates and with specified line pattern. It draws a single line if 'nb_vertices' is set to 2.
- 'ellipse(_#ind,xc,yc,radius1,_radius2,_angle,_opacity,_color)' draws a filled ellipse in image [ind] (or in default image selected if 'ind' is not specified) with specified coordinates.
- 'ellipse(_#ind,xc,yc,-radius1,-_radius2,_angle,_opacity,_pattern,_color)' draws an outlined ellipse in image [ind] (or in default image selected if 'ind' is not specified).
- 'resize(#ind,w,_h,_d,_s,_interp,_boundary_conditions,_cx,_cy,_cz,_cc)' resizes an image of the associated list with specified dimension and interpolation method. When using this, function, you should consider retrieving the (non-constant) image dimensions using the dynamic functions 'w(_#ind)', 'h(_#ind)', 'd(_#ind)', 's(_#ind)', 'wh(_#ind)', 'whd(_#ind)' and 'whds(_#ind)' instead of the corresponding constant variables.
- 'if(condition,expr_then,expr_else)': return value of 'expr_then' or 'expr_else', depending on the value of 'condition' (0=false, other=true). 'expr_else' can be omitted in which case 0 is returned if the condition does not hold. Using the ternary operator 'condition?expr_then:expr_else' gives an equivalent expression.
 - For instance, G*MIC commands 'fill if(x%10==0,255,i)' and 'fill x%10?i:255' both draw blank vertical lines on every 10th column of an image.
- 'do(expression,_condition)' repeats the evaluation of 'expression' until 'condition' vanishes (or until 'expression' vanishes if no 'condition' is specified). For instance, the expression: 'if(N<2,N,n=N-1;F0=0;F1=1;do(F2=F0+F1;F0=F1;F1=F2,n=n-1))' returns the Nth value of the Fibonacci sequence, for N>=0 (e.g., 46368 for N=24).
 - 'do(expression,condition)' always evaluates the specified expression at least once, then check for the loop condition. When done, it returns the last value of 'expression'.
- 'for(init,condition,_procedure,body)' first evaluates the expression 'init', then iteratively evaluates 'body' (followed by 'procedure' if specified) while 'condition' is verified (i.e. not zero). It may happen that no iteration is done, in which case the function returns nan. Otherwise, it returns the last value of 'body'.
 - For instance, the expression: 'if(N<2,N,for(n=N;F0=0;F1=1,n=n-1,F2=F0+F1;F0=F1;F1=F2))' returns the Nth value of the Fibonacci sequence, for N>=0 (e.g., 46368 for N=24).
- 'while(condition,expression)' is exactly the same as 'for(init,condition,expression)' without the specification of an initializing expression.
- 'break()' and 'continue()' respectively breaks and continues the current running bloc (loop, init or main environment).

- . `'fsize('filename')` returns the size of the specified 'filename' (or `-1` if file does not exist).
- . `'date(attr,'path')` returns the date attribute for the given 'path' (file or directory), with 'attr' being { `0=year` | `1=month` | `2=day` | `3=day of week` | `4=hour` | `5=minute` | `6=second` }, or a vector of those values.
- . `'date(.attr)` returns the specified attribute for the current (locale) date.
- . `'print(expr1,expr2,...)` or `'print(#ind)` prints the value of the specified expressions (or image information) on the console, and returns the value of the last expression (or `nan` in case of an image). Function `'prints(expr)` also prints the string composed of the ascii characters defined by the vector-valued expression (e.g. `'prints('Hello')`).
- . `'debug(expression)` prints detailed debug information about the sequence of operations done by the math parser to evaluate the expression (and returns its value).
- . `'display(_X,_w,_h,_d,_s)` or `'display(#ind)` display the contents of the vector 'X' (or specified image) and wait for user events. if no arguments are provided, a memory snapshot of the math parser environment is displayed instead.
- . `'begin(expression)` and `'end(expression)` evaluates the specified expressions only once, respectively at the beginning and end of the evaluation procedure, and this, even when multiple evaluations are required (e.g. in `'fill ">begin(foo = 0); ++foo"`).
- . `'copy(dest,src,_nb_elts,_inc_d,_inc_s,_opacity)` copies an entire memory block of '`nb_elts`' elements starting from a source value '`src`' to a specified destination '`dest`', with increments defined by '`inc_d`' and '`inc_s`' respectively for the destination and source pointers.
- . `'stats(#ind)` returns the statistics vector of the running image '`[ind]`', i.e the vector [`im,iM,ia,iv,xm,ym,zm,cm,xM,yM,zM,cM,is,ip`] (14 values).
- . `'unref(a,b,...)` destroys references to the named variable given as arguments.
- . `'breakpoint()` inserts a possible computation breakpoint (not supported by the cli interface).
- . `'_(expr)` just ignores its arguments (mainly useful for debugging).
- . `'ext('pipeline')` executes the specified G'MIC pipeline as if it was called outside the currently evaluated expression.

– User-defined macros:

- . Custom macro functions can be defined in a math expression, using the assignment operator '=', e.g. `'foo(x,y) = cos(x + y); result = foo(1,2) + foo(2,3)`.
- . Trying to override a built-in function (e.g. `'abs()`) has no effect.
- . Overloading macros with different number of arguments is possible. Re-defining a previously defined macro with the same number of arguments discards its previous definition.
- . Macro functions are indeed processed as `macros` by the mathematical evaluator. You should avoid invoking them with arguments that are themselves results of assignments or self-operations. For instance, `'foo(x) = x + x; z = 0; foo(++z)` returns `'4` rather than expected value `'2`.
- . When substituted, macro arguments are placed inside parentheses, except if a number sign '#' is located just before or after the argument name. For instance, expression `'foo(x,y) = x*y; foo(1+2,3)` returns `'9` (being substituted as `'(1+2)*(3)`'), while expression `'foo(x,y) = x#*y#; foo(1+2,3)` returns `'7` (being substituted as `'1+2*3`).
- . Number signs appearing between macro arguments function actually count for '`empty`' separators. They may be used to force the substitution of macro arguments in unusual places, e.g. as in `'str(N) = ['I like N#'];`.

– Multi-threaded and in-place evaluation:

- . If your image data are large enough and you have several CPUs available, it is likely that the math expression passed to a '`fill`' or '`input`' command is evaluated in parallel, using multiple computation threads.

- . Starting an expression with ':' or '*' forces the evaluations required for an image to be run in parallel, even if the amount of data to process is small (beware, it may be slower to evaluate in this case!). Specify ':' (instead of '*') to avoid possible image copy done before evaluating the expression (this saves memory, but do this only if you are sure this step is not required!)
- . If the specified expression starts with '>' or '<', the pixel access operators 'i(), i[], j()' and 'j[]' return values of the image being currently modified, in forward ('>') or backward ('<') order. The multi-threading evaluation of the expression is also disabled in this case.
- . Function 'critical(operands)' forces the execution of the given operands in a single thread at a time.
- Expressions 'i(_#ind,x,-y,-z,-c)=value', 'j(_#ind,x,-y,-z,-c)=value', 'i[_#ind,offset]=value' and 'j[_#ind,offset]=value' set a pixel value at a different location than the running one in the image [ind] (or in the associated image if argument '#ind' is omitted), either with global coordinates/offsets (with 'i(...)' and 'i[...]'), or relatively to the current position (x,y,z,c) (with 'j(...)' and 'j[...]'). These expressions always return 'value'.
- The last image of the list is always associated to the evaluations of '{expressions}', e.g. G'MIC sequence '256,128 fill {w}' will create a 256x128 image filled with value 256.

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 manages interactive views of the selected image data.
- The following keyboard shortcuts are available in the interactive viewers:
 - . (mousewheel): Zoom in/out.
 - . CTRL+D: Increase window size.
 - . CTRL+C: Decrease window size.
 - . CTRL+R: Reset window size.
 - . CTRL+W: Close window.
 - . 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): Go up/down.
 - . SHIFT+(mousewheel): Go left/right.
 - . 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.
- . 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

- New custom commands can be added by the 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%/user.gmic` (on Windows). The CLI tool 'gmic' automatically runs the command 'cli_start' if defined.
- Custom command names must use character set `[a-zA-Z0-9_]` and cannot start with a number.
- Any '# comment' expression found in a custom commands file is discarded by the **G'MIC** parser, wherever it is located in a line.
- In a custom command, the following **\$—expressions** are recognized and substituted:
 - . '\$*' is substituted by a copy of the specified string of arguments.
 - . '\$""' is substituted by a copy of the specified string of arguments, each being double—quoted.
 - . '\$#' is substituted by the maximum index of known arguments (either specified by the user or set to a default value in the custom command).
 - . '\$[]' is substituted by the list of selected image indices that have been specified during the command invocation.
 - . '\$?' is substituted by a printable version of '\$[]' to be used in command descriptions.
 - . '\$i' and '\${i}' are both substituted by the *i*th specified argument. Negative indices such as '\${-j}' are allowed and refer to the *j*th latest argument. '\$0' is substituted by the custom command name.
 - . '\${i=default}' is substituted by the value of \$i (if defined) or by its new value set to 'default' otherwise ('default' may be a \$—expression as well).
 - . '\${subset}' is substituted by the argument values (separated by commas ',') of a specified argument subset. For instance expression '\${2--2}' is substituted by all specified command arguments except the first and the last one. Expression '\${^0}' is then substituted by all arguments of the invoked command (eq. to '\$*' if all 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 character set `[a-zA-Z0-9_]` and cannot start with a number.
- These particular **\$—expressions** for custom commands are **always substituted**, even in double—quoted items or when the dollar sign '\$' is escaped with a backslash '\'. To avoid

substitution, place an empty double quoted string just after the '\$' (as in '\$""1').

- Specifying arguments may be skipped when invoking a custom command, by replacing them by commas ',' as in expression 'flower „3'. Omitted arguments are set to their default values, which must be thus explicitly defined in the code of the corresponding custom command (using default argument expressions as '\${1=default}').
- If one numbered argument required by a custom command misses a value, an error is thrown by the G'MIC interpreter.

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 index number or an image name. In this case, the '[]' characters are mandatory when writing the item. A command marked with '(+)' is one of the builtin commands. 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 explicitly (command 'normalize 0,255') if you want to save and view images with 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 is useful for developers or to report possible bugs of the interpreter.

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 on stdout.

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.

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 clut**Arguments:**

- "clut_name", _resolution>0

Insert one of the 552 pre-defined CLUTs at the end of the image list.\n

'clut_name' can be { 2-strip-process | 60s | 60s_faded | 60s_faded_alt | agfa_apx_100 | agfa_apx_25 | agfa_precisa_100 | agfa_ultra_color_100 | agfa_vista_200 | alien_green | analogfx_anno_1870_color | analogfx_old_style_i | analogfx_old_style_ii | analogfx_old_style_iii | analogfx_sepia_color | analogfx_soft_sepia_i | analogfx_soft_sepia_ii | anime | apocalypse_this_very_moment | aqua | aqua_and_orange_dark | arabica_12 | ava_614 | azrael_93 | bboyz_2 | berlin_sky | black_and_white | bleach_bypass | bleachbypass_1 | bleachbypass_2 | bleachbypass_3 | bleachbypass_4 | blue_mono | blues | bob_ford | bourbon_64 | bw_1 | bw_10 | bw_2 | bw_3 | bw_4 | bw_5 | bw_6 | bw_7 | bw_8 | bw_9 | byers_11 | candlelight | chemical_168 | chrome_01 | cinematic-1 | cinematic-10 | cinematic-2 | cinematic-3 | cinematic-4 | cinematic-5 | cinematic-6 | cinematic-7 | cinematic-8 | cinematic-9 | classic_tea_and_orange | clayton_33 | clouseau_54 | cob_3 | color_rich | colornegative | contrail_35 | crispwarm | crispwinter | cubicle_99 | django_25 | domingo_145 | dropblues | earth_tone_boost | edgyember | expired_fade | expired_polaroid | extreme | fade | fade_to_green | faded | faded_47 | faded_alt | faded_analog | faded_extreme | faded_vivid | fallcolors | faux_infrared | fgcinebasic | fgcinebright | fgcinecold | fgcinedrama | fgcinetealorange_1 | fgcinetealorange_2 | fgcinevibrant | fgcinewarm | film_print_01 | film_print_02 | foggy_night | folger_50 | french_comedy | fuji_160c | fuji_160c_+ | fuji_160c_++ | fuji_160c_- | fuji_3510_constlclip | fuji_3510_constlmap | fuji_3510_cuspclip | fuji_3513_constlclip | fuji_3513_constlmap | fuji_3513_cuspclip | fuji_400h | fuji_400h_+ | fuji_400h_++ | fuji_400h_- | fuji_800z | fuji_800z_+ | fuji_800z_++ | fuji_800z_- | fuji_astia_100_generic | fuji_astia_100f | fuji_fp_100c | fuji_fp_100c_+ | fuji_fp_100c_++ | fuji_fp_100c_+++ | fuji_fp_100c_++_alt | fuji_fp_100c_- | fuji_fp_100c_- | fuji_fp_100c_alt | fuji_fp_100c_cool | fuji_fp_100c_cool_+ | fuji_fp_100c_cool_++ | fuji_fp_100c_cool_- | fuji_fp_100c_cool_- | fuji_fp_100c_negative | fuji_fp_100c_negative_+ | fuji_fp_100c_negative_++ | fuji_fp_100c_negative_+++ | fuji_fp_100c_negative_++_alt | fuji_fp_100c_negative_- | fuji_fp_100c_negative_- | fuji_fp_3000b | fuji_fp_3000b_+ | fuji_fp_3000b_++ | fuji_fp_3000b_+++ | fuji_fp_3000b_- | fuji_fp_3000b_- | fuji_fp_3000b_hc | fuji_fp_3000b_negative | fuji_fp_3000b_negative_+ | fuji_fp_3000b_negative_++ | fuji_fp_3000b_negative_+++ | fuji_fp_3000b_negative_- | fuji_fp_3000b_negative_- | fuji_fp_3000b_negative_early | fuji_fp_100c | fuji_neopan_1600 | fuji_neopan_1600_+ | fuji_neopan_1600_++ | fuji_neopan_1600_- | fuji_neopan_acros_100 | fuji_provia_100_generic | fuji_provia_100f | fuji_provia_400f | fuji_provia_400x | fuji_sensia_100 | fuji_superia_100 | fuji_superia_100_+ | fuji_superia_100_++ | fuji_superia_100_- | fuji_superia_1600 | fuji_superia_1600_+ | fuji_superia_1600_++ | fuji_superia_1600_- | fuji_superia_200 | fuji_superia_200_xpro | fuji_superia_400 | fuji_superia_400_+ | fuji_superia_400_++ | fuji_superia_400_- | fuji_superia_800 | fuji_superia_800_+ | fuji_superia_800_++ | fuji_superia_800_- | fuji_superia_hg_1600 | fuji_superia_reala_100 | fuji_superia_x-tra_800 | fuji_velvia_100_generic | fuji_velvia_50 | fuji_xtrans_iii_acros | fuji_xtrans_iii_acros+g | fuji_xtrans_iii_acros+r | fuji_xtrans_iii_acros+ye | fuji_xtrans_iii_astia | fuji_xtrans_iii_classic_chrome | fuji_xtrans_iii_mono | fuji_xtrans_iii_mono+g | fuji_xtrans_iii_mono+r | fuji_xtrans_iii_mono+ye | fuji_xtrans_iii_pro_neg_hi | fuji_xtrans_iii_pro_neg_std | fuji_xtrans_iii_provia | fuji_xtrans_iii_sepia | fuji_xtrans_iii_velvia | fusion_88 | futuristicbleak_1 | futuristicbleak_2 | futuristicbleak_3 | futuristicbleak_4 | golden | golden_bright | golden_fade | golden_mono

| golden_vibrant | goldfx_bright_spring_breeze | goldfx_bright_summer_heat | goldfx_hot_summer_heat
 | goldfx_perfect_sunset_01min | goldfx_perfect_sunset_05min | goldfx_perfect_sunset_10min
 | goldfx_spring_breeze | goldfx_summer_heat | green_blues | green_mono | green_yellow | hack-
 manite | herderite | heulandite | hiddenite | hilutite | hong_kong | horrorblue | howlite | hyla_68
 | hypersthene | ilford_delta_100 | ilford_delta_3200 | ilford_delta_3200_+ | ilford_delta_3200_++ | il-
 ford_delta_3200_- | ilford_delta_400 | ilford_fp_4_plus_125 | ilford_hp_5 | ilford_hp_5_+ | ilford_hp_5_++
 | ilford_hp_5_- | ilford_hp_5_plus_400 | ilford_hps_800 | ilford_pan_f_plus_50 | ilford_xp_2 | ko-
 dak_2383_constlclip | kodak_2383_constlmap | kodak_2383_cuspclick | kodak_2393_constlclip | ko-
 dak_2393_constlmap | kodak_2393_cuspclick | kodak_bw_400_cn | kodak_e-100_gx_ektachrome_100 | ko-
 dak_ektachrome_100_vs | kodak_ektachrome_100_vs_generic | kodak_ektar_100 | kodak_elite_100_xpro
 | kodak_elite_chrome_200 | kodak_elite_chrome_400 | kodak_elite_color_200 | kodak_elite_color_400
 | kodak_elite_extracolor_100 | kodak_hie_hs_infra | kodak_kodachrome_200 | kodak_kodachrome_25
 | kodak_kodachrome_64 | kodak_kodachrome_64_generic | kodak_portra_160 | kodak_portra_160_+
 | kodak_portra_160_++ | kodak_portra_160_- | kodak_portra_160_nc | kodak_portra_160_nc_+ | ko-
 dak_portra_160_nc_++ | kodak_portra_160_nc_- | kodak_portra_160_vc | kodak_portra_160_vc_+ | ko-
 dak_portra_160_vc_++ | kodak_portra_160_vc_- | kodak_portra_400 | kodak_portra_400_+ | ko-
 dak_portra_400_++ | kodak_portra_400_- | kodak_portra_400_nc | kodak_portra_400_nc_+ | ko-
 dak_portra_400_nc_++ | kodak_portra_400_nc_- | kodak_portra_400_uc | kodak_portra_400_uc_+ | ko-
 dak_portra_400_uc_++ | kodak_portra_400_uc_- | kodak_portra_400_vc | kodak_portra_400_vc_+ | ko-
 dak_portra_400_vc_++ | kodak_portra_400_vc_- | kodak_portra_800 | kodak_portra_800_+ | ko-
 dak_portra_800_++ | kodak_portra_800_- | kodak_portra_800_hc | kodak_t-max_100 | kodak_t-
 max_3200 | kodak_t-max_400 | kodak_tmax_3200 | kodak_tmax_3200_+ | kodak_tmax_3200_++ | ko-
 dak_tmax_3200_- | kodak_tmax_3200_alt | kodak_tri-x_400 | kodak_tri-x_400_+ | kodak_tri-x_400_++
 | kodak_tri-x_400_- | kodak_tri-x_400_alt | korben_214 | landscape_1 | landscape_10 | landscape_2
 | landscape_3 | landscape_4 | landscape_5 | landscape_6 | landscape_7 | landscape_8 | landscape_9 | late-
 sunset | lc_1 | lc_10 | lc_2 | lc_3 | lc_4 | lc_5 | lc_6 | lc_7 | lc_8 | lc_9 | lenox_340 | life_giving_tree
 | light_blow | lomo | lomography_redscale_100 | lomography_x-pro_slide_200 | lucky_64 | mckin-
 non_75 | milo_5 | mono_tinted | moody_1 | moody_10 | moody_2 | moody_3 | moody_4 | moody_5
 | moody_6 | moody_7 | moody_8 | moody_9 | moonlight | moonrise | mute_shift | muted_fade | natu-
 ral_vivid | neon_770 | nightfromday | nostalgic | nw-1 | nw-10 | nw-2 | nw-3 | nw-4 | nw-5 | nw-6
 | nw-7 | nw-8 | nw-9 | orange_tone | oranges | paladin_1875 | pasadena_21 | pink_fade | pitaya_15
 | polaroid_664 | polaroid_665 | polaroid_665_+ | polaroid_665_++ | polaroid_665_- | polaroid_665_-
 | polaroid_665_negative | polaroid_665_negative_+ | polaroid_665_negative_- | polaroid_665_negative_hc
 | polaroid_667 | polaroid_669 | polaroid_669_+ | polaroid_669_++ | polaroid_669_+++ | po-
 laroid_669_- | polaroid_669_- | polaroid_669_cold | polaroid_669_cold_+ | polaroid_669_cold_-
 | polaroid_669_cold_- | polaroid_672 | polaroid_690 | polaroid_690_+ | polaroid_690_++ | po-
 laroid_690_- | polaroid_690_- | polaroid_690_cold | polaroid_690_cold_+ | polaroid_690_cold_++
 | polaroid_690_cold_- | polaroid_690_cold_- | polaroid_690_warm | polaroid_690_warm_+ | po-
 laroid_690_warm_++ | polaroid_690_warm_- | polaroid_690_warm_- | polaroid_polachrome | po-
 laroid_px-100uv+_cold | polaroid_px-100uv+_cold_+ | polaroid_px-100uv+_cold_++ | polaroid_px-
 100uv+_cold_+++ | polaroid_px-100uv+_cold_- | polaroid_px-100uv+_cold_- | polaroid_px-100uv+_warm
 | polaroid_px-100uv+_warm_+ | polaroid_px-100uv+_warm_++ | polaroid_px-100uv+_warm_+++ | po-
 laroid_px-100uv+_warm_- | polaroid_px-100uv+_warm_- | polaroid_px-680 | polaroid_px-680_+ | po-
 laroid_px-680_++ | polaroid_px-680_- | polaroid_px-680_- | polaroid_px-680_cold | polaroid_px-
 680_cold_+ | polaroid_px-680_cold_++ | polaroid_px-680_cold_++_alt | polaroid_px-680_cold_- | po-
 laroid_px-680_cold_- | polaroid_px-680_warm | polaroid_px-680_warm_+ | polaroid_px-680_warm_++
 | polaroid_px-680_warm_- | polaroid_px-680_warm_- | polaroid_px-70 | polaroid_px-70_+ | polaroid_px-
 70_++ | polaroid_px-70_+++ | polaroid_px-70_- | polaroid_px-70_- | polaroid_px-70_cold | polaroid_px-
 70_cold_+ | polaroid_px-70_cold_++ | polaroid_px-70_cold_- | polaroid_px-70_cold_- | polaroid_px-
 70_warm | polaroid_px-70_warm_+ | polaroid_px-70_warm_++ | polaroid_px-70_warm_- | polaroid_px-

70_warm_- | polaroid_time_zero_expired | polaroid_time_zero_expired_+ | polaroid_time_zero_expired_++
 | polaroid_time_zero_expired_- | polaroid_time_zero_expired_- | polaroid_time_zero_expired_- | po-
 laroid_time_zero_expired_cold | polaroid_time_zero_expired_cold_- | polaroid_time_zero_expired_cold_-
 | polaroid_time_zero_expired_cold_- | portrait_1 | portrait_10 | portrait_2 | portrait_3 | portrait_4
 | portrait_5 | portrait_6 | portrait_7 | portrait_8 | portrait_9 | purple | purple_2 | redblueyel-
 low | reds | reds_oranges_yellows | reeve_38 | remy_24 | retro | rollei_ir_400 | rollei_ortho_25
 | rollei_retro_100_tonal | rollei_retro_80s | rotate_muted | rotate_vibrant | rotated | rotated_crush
 | saving_private_damon | smokey | smooth_cromeish | smooth_fade | soft_fade | softwarming | solar-
 ized_color | solarized_color_2 | sprocket_231 | studio_skin_tone_shaper | summer | summer_alt | sunny
 | sunny_alt | sunny_rich | sunny_warm | super_warm | super_warm_rich | sutro_fx | tealmagenta gold
 | tealorange | tealorange_1 | tealorange_2 | tealorange_3 | technicalfx_backlight_filter | teigen_28 | ten-
 siongreen_1 | tensiongreen_2 | tensiongreen_3 | tensiongreen_4 | the_matrices | trent_18 | tweed_71
 | vibrant | vibrant_alien | vibrant_contrast | vibrant_cromeish | vintage | vintage_alt | vintage_brighter
 | vintage_chrome | vireo_37 | warm | warm_highlight | warm_yellow | zed_32 | zeke_39 | zil-
 verfx_bw_solarization | zilverfx_infrared | zilverfx_vintage_bw }



Example 1 : `clut summer`

2.2.3 *command (+)*

Arguments:

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

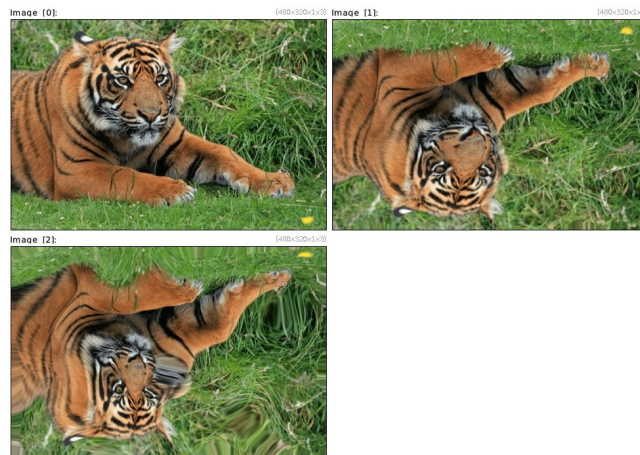
Import G'MIC custom commands from specified file, URL or string.

(*eq. to 'm'*) . \n).

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

Default value:

- `'_add_debug_info=1'.`



Example 2: `image.jpg` command `"foo : mirror y deform $"1" +foo[0] 5 +foo[0] 15`

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[%]>=0, _Y[%]>=0, _Z[%]>=0, _exit_on_anykey={ 0 | 1 }`

Display selected images in an interactive viewer (use the instant display window [0] if opened).
(eq. to `'d'`) .\n).

Arguments `'X'`, `'Y'`, `'Z'` determine the initial selection view, for 3D volumetric images.

Default value:

- `'X=Y=Z=0'` and `'exit_on_anykey=0'`.

Tutorial page:

https://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], _exit_on_anykey={ 0 | 1 }`
- `_exit_on_anykey={ 0 | 1 }`

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

Default values:

- `'_[background_image]=(default)'` and `'exit_on_anykey=0'`.

2.2.8 *display_array*

Arguments:

- `_width>0, _height>0`

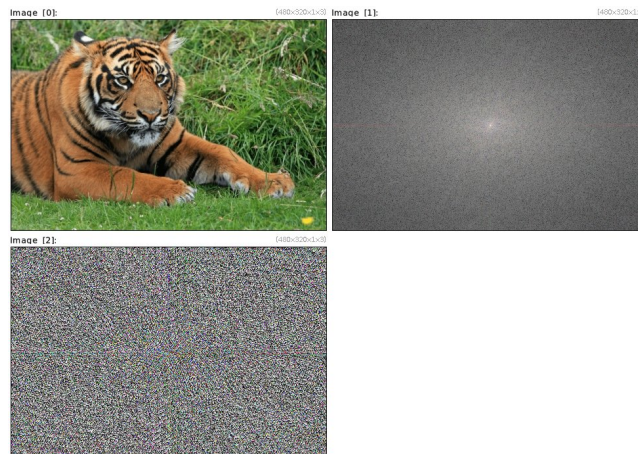
Display images in interactive windows where pixel neighborhoods can be explored.

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>=0, _height>=0, _plot_type, _vertex_type, _xmin, _xmax, _ymin, _ymax, _xlabel, _ylabel`

Render graph plot from selected image data.

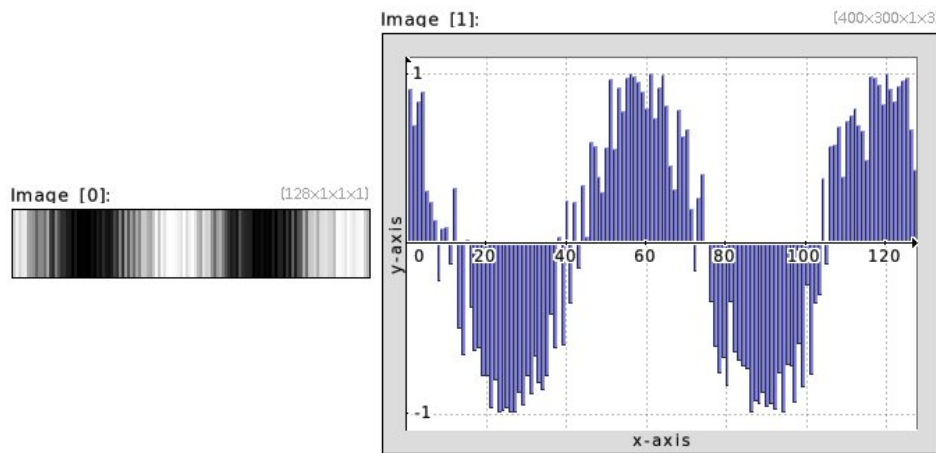
'plot_type' can be { 0=none | 1=lines | 2=splines | 3=bar }.

'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. if specified 'width' or 'height' is '0', then image size is set to half the screen size.

Default values:

- 'width=0', 'height=0', 'plot_type=1', 'vertex_type=1',
'xmin=xmax=ymin=ymax=0 (auto)', 'xlabel="x-axis"' and 'ylabel="y-axis"'.



Example 4 : 128,1,1,1, 'cos (x/10+u)' +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.

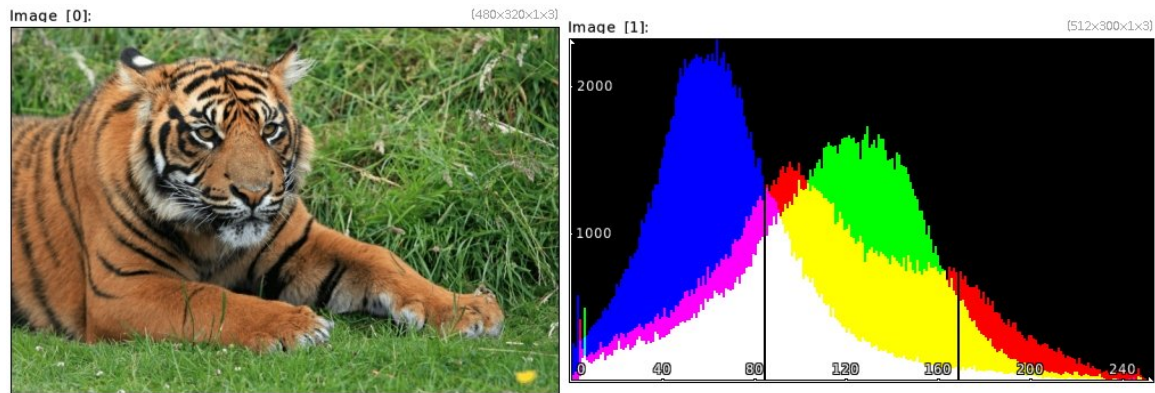
Render a channel-by-channel histogram.

If selected images have several slices, the rendering is performed for all input slices.

'expression' is a mathematical expression used to transform the histogram data for visualization purpose. (eq. to 'dh'). if specified 'width' or 'height' is '0', then image size is set to half the screen size.

Default values:

- 'width=0', 'height=0', 'clusters=256', 'min_value=0%', 'max_value=100%',
'show_axes=1' and 'expression=i'.



Example 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.

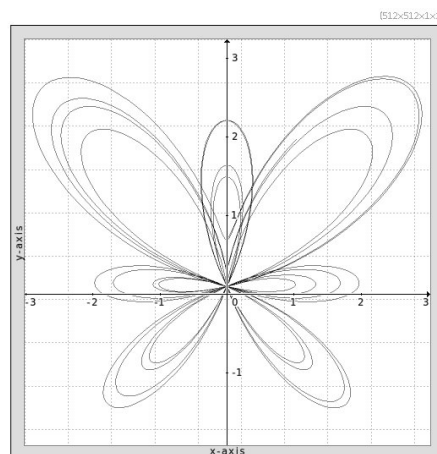
Curve points are defined as pixels of a 2 or 3-channel image.

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

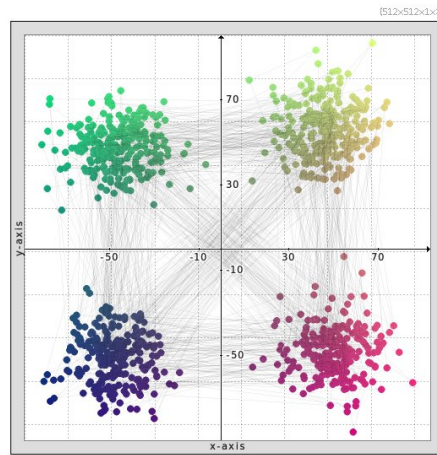
If `'outline_opacity>1'`, the outline is colored according to the specified vertex colors and `'outline_opacity-1'` is used as the actual drawing opacity.

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,u(-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_parallel*

Display each selected image in a separate interactive display window.
(eq. to 'dp').

2.2.14 *display_parallel0*

Display each selected image in a separate interactive display window, without value normalization.
(eq. to 'dp0').

2.2.15 *display_polar*

Arguments:

- `_width>32, _height>32, _outline_type, _fill_R, _fill_G, _fill_B, _theta_start, _theta_end, _xlabel, _ylabel`

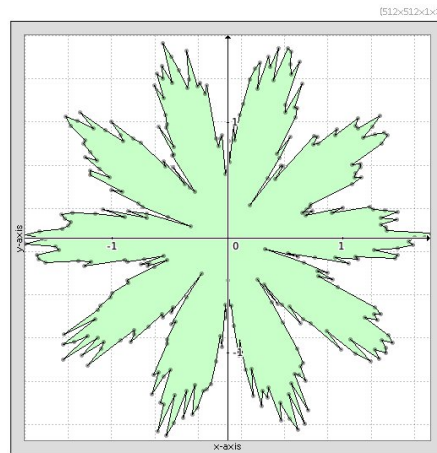
Render polar curve from selected image data.

'outline_type' can be { `r<0`=dots with radius -r | `0`=no outline | `r>0`=lines+dots with radius r }.

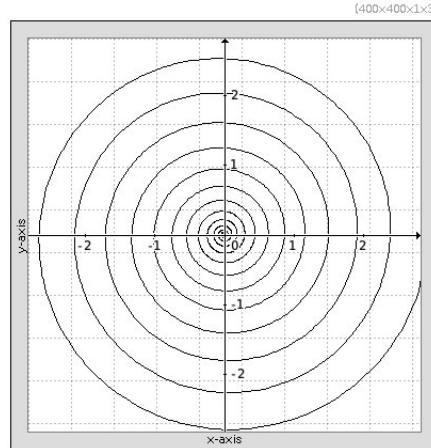
'fill_color' can be { `-1`=no fill | `R,G,B`=fill with specified color }.

Default values:

- `'width=500', 'height=width', 'outline_type=1', 'fill_R=fill_G=fill_B=200', 'theta_start=0', 'theta_end=360', 'xlabel="x-axis"' and 'ylabel="y-axis"'`.



Example 8 : `300,1,1,1,'0.3+abs(cos(10*pi*x/w))+u(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}`

2.2.16 *display_quiver*

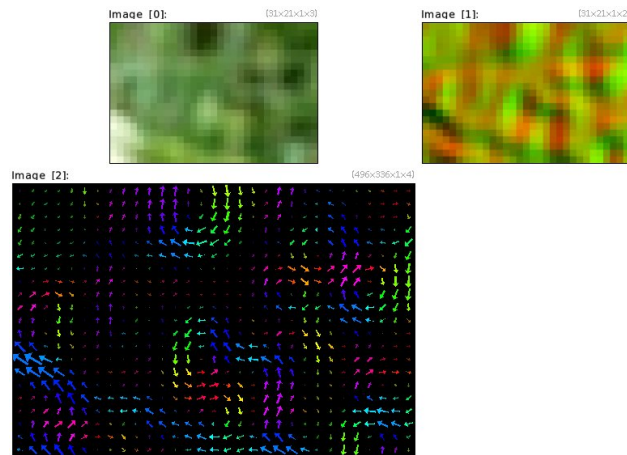
Arguments:

- `_size_factor>0, _arrow_size>0, _color_mode={ 0=monochrome | 1=grayscale | 2=color }`

Render selected images of 2D vectors as a field of 2D arrows.
(eq. to 'dq').

Default values:

- `'size_factor=16', 'arrow_size=1.5' and 'color_mode=1'.`



Example 10: `image.jpg +luminance gradient[-1] xy rv[-2,-1] *[-2] -1 a[-2,-1] c crop 60,10,90,30 +display-quiver[1] ,`

2.2.17 *display_rgba*

Arguments:

- `_background_RGB_color`

Render selected RGBA images over a checkerboard or colored background. (eq. to 'drgba').

Default values:

- `'background_RGB_color=undefined'` (checkerboard).



Example 11: `image.jpg +norm threshold[-1] 40% blur[-1] 3 normalize[-1] 0,255 append c display_rgba`

2.2.18 *display_tensors*

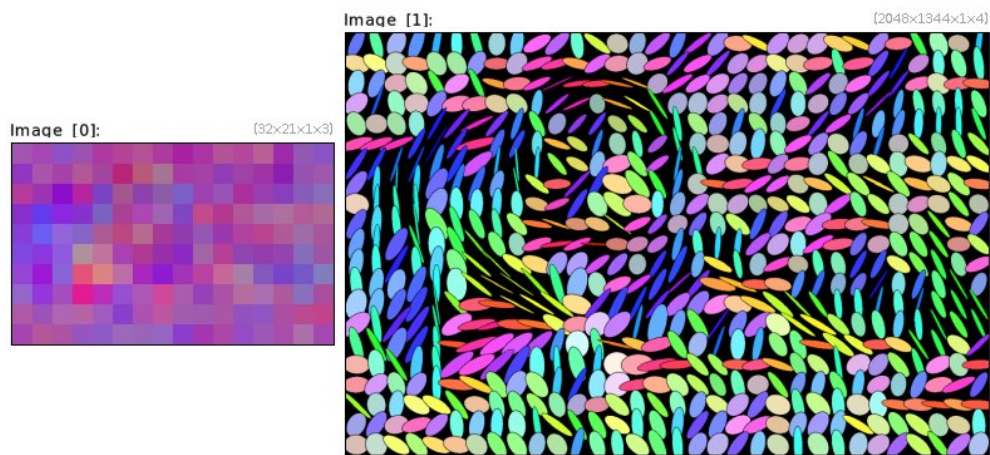
Arguments:

```
• _size_factor>0, _ellipse_size>=0, _color_mode={ 0=monochrome | 1=grayscale
| 2=color }, _outline>=0
```

Render selected images of tensors as a field of 2D ellipses.
(eq. to 'dt').

Default values:

- 'size_factor=16', 'ellipse_size=1.5', 'color_mode=2' and 'outline=2'.



Example 12 : image.jpg diffusiontensors 0.1,0.9 resize2dx 32 +display_tensors 64,2

Tutorial page:

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

2.2.19 *display_warp*

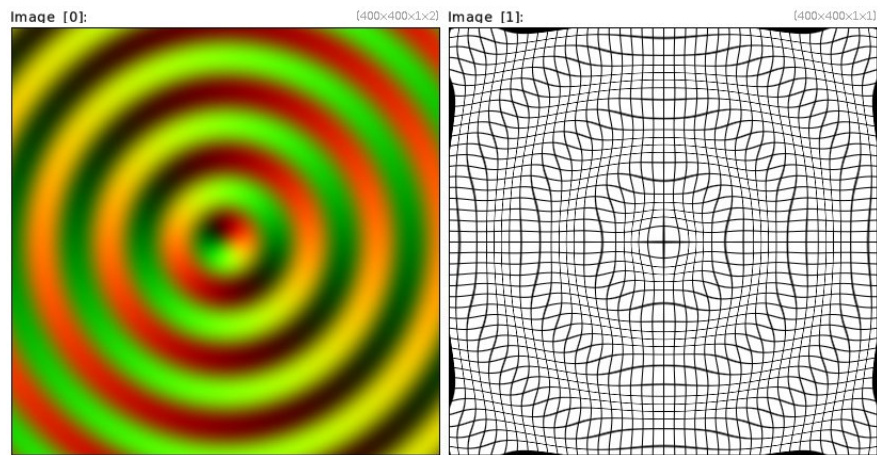
Arguments:

- _cell_size>0

Render selected 2D warping fields.
(eq. to 'dw').

Default value:

- 'cell_size=15'.



Example 13 :

```
400,400,1,2,'x=x-w/2;y=y-h/2;r=sqrt(x*x+y*y);a=atan2(y,x);5*sin(r/10)*[cos(a),sin(a)]'
+display-warp 10
```

2.2.20 *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

2.2.21 *echo (+)*

Arguments:

- message

Output specified message on the error output.

(eq. to 'e') .\n).

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

2.2.22 *echo_file*

Arguments:

- filename,message

Output specified message, appending it to specified output file.

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

2.2.23 *echo_stdout*

Arguments:

- `message`

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

2.2.24 *function1d*

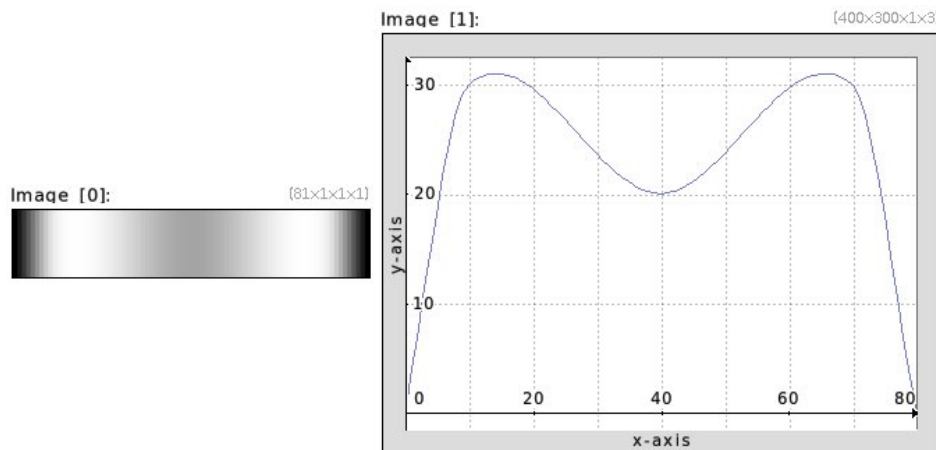
Arguments:

- `0<=smoothness<=1,x0>=0,y0,x1>=0,y1,...,xn>=0,yn`

Insert continuous 1D function from specified list of keypoints (xk,yk) in range [0,max(xk)] (xk are positive integers).

Default values:

- `'smoothness=1'` and `'x0=y0=0'`.



Example 14 : `function1d 1,0,0,10,30,40,20,70,30,80,0 +display_graph 400,300`

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] },-{ value1,_value2,... | 'formula' }`
- `(value1{, | ; | / | ^}value2{, | ; | / | ^}...)`
- `0`

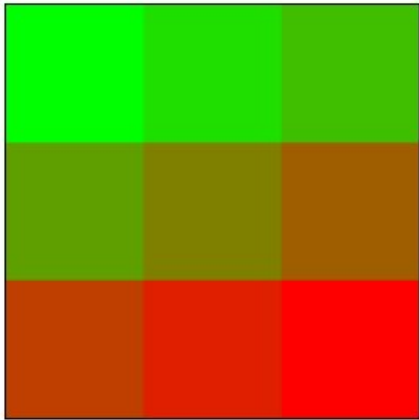
Insert a new image taken from a filename or from a copy of an existing image [index], or insert new image with specified dimensions and values. Single quotes may be omitted in 'formula'. Specifying argument '0' inserts an 'empty' image.
(eq. to 'i' | (no arg)).

Default values:

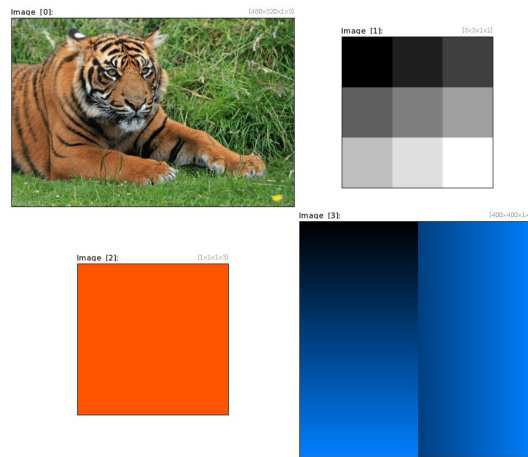
- 'nb_copies=1', 'height=depth=spectrum=1' and 'value1=0'.



Example 15 : input image.jpg
[3x3x1x2]



Example 16 : input (1,2,3;4,5,6;7,8,9^9,8,7;6,5,4;3,2,1)



Example 17: `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:

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

2.2.26 *input_cube*

Arguments:

- `"filename",_convert_1d_cluts_to_3d={ 0 | 1 }`.

Insert CLUT data from a .cube filename (Adobe CLUT file format).

Default value:

- `'convert_1d_cluts_to_3d=1'`.

2.2.27 *input_glob*

Arguments:

- `pattern`

Insert new images from several filenames that match the specified glob pattern.

2.2.28 *input_gpl*

Arguments:

- `filename`

Input specified filename as a .gpl palette data file.

2.2.29 *output (+)*

Arguments:

- `[type:]filename, _format_options`

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

Default value:

- `'format_options'=(undefined) .`

2.2.30 *output_cube*

Arguments:

- `filename`

Output selected CLUTs as a .cube file (Adobe CLUT format).

2.2.31 *output_ggr*

Arguments:

- `filename, _gradient_name`

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

2.2.32 *outputn*

Arguments:

- `filename, _index`

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

2.2.33 *outputp*

Arguments:

- `prefix`

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

Default value:

- `'prefix=-' .`

2.2.34 *outputw*

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

2.2.35 *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'`.

2.2.36 *pass (+)*

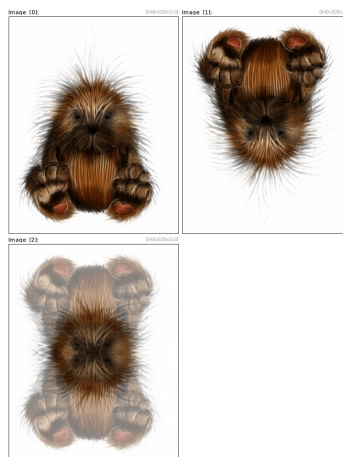
Arguments:

- `_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 18 : `command "average : pass$""1 add[^-1] [-1] remove[-1] div 2" sample ? +mirror y +average[0] [1]`

2.2.37 *plot* (+)

Arguments:

- `_plot_type, _vertex_type, _xmin, _xmax, _ymin, _ymax, _exit_on_anykey={ 0 | 1 }`
- `'formula', _resolution>=0, _plot_type, _vertex_type, _xmin, _xmax, _ymin, _ymax, _exit_on_anykey={ 0 | 1 }`

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

'plot_type' can be { 0=none | 1=lines | 2=splines | 3=bar }.

'vertex_type' can be { 0=none | 1=points | 2,3=crosses | 4,5=circles | 6,7=squares }.

'xmin', 'xmax', 'ymin', 'ymax' set the coordinates of the displayed xy-axes.

Default values:

- `'plot_type=1', 'vertex_type=1', 'xmin=xmax=ymin=ymax=0 (auto)' and 'exit_on_anykey=0'.`

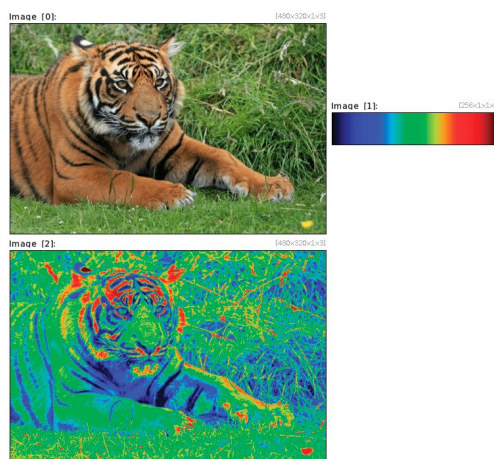
2.2.38 *print* (+)

Output information on selected images, on the standard error (stderr).

(eq. to 'p').

2.2.39 *rainbow_lut*

Input a 256-entries RGB colormap of rainbow colors.



Example 19 : `image.jpg rainbow_lut +luminance[-2] map[-1] [-2]`

2.2.40 *screen* (+)

Arguments:

- `_x0[%], _y0[%], _x1[%], _y1[%]`

Take screenshot, optionally grabbed with specified coordinates, and insert it at the end of the image list.

2.2.41 *select* (+)

Arguments:

- `feature_type, _X[%]>=0, _Y[%]>=0, _Z[%]>=0, _exit_on_anykey={ 0 | 1 }, _is_deep_selection={ 0 | 1 }`

Interactively select a feature from selected images (use the instant display window [0] if opened).

'feature_type' can be { 0=point | 1=segment | 2=rectangle | 3=ellipse }.

Arguments 'X','Y','Z' determine the initial selection view, for 3D volumetric images.

The retrieved feature is returned as a 3D vector (if 'feature_type==0') or as a 6d vector (if 'feature_type!=0') containing the feature coordinates.

Default values:

- `'X=Y=Z=(undefined)', 'exit_on_anykey=0' and 'is_deep_selection=0'.`

2.2.42 *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 { auto | uchar | char | ushort | short | uint | int | uint64 | int64 | float | double }.

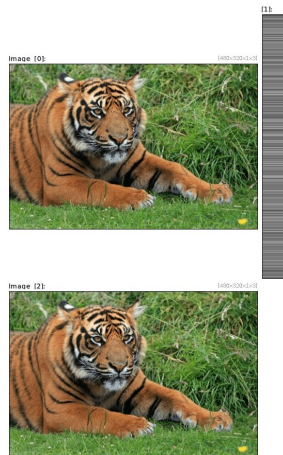
Specify 'datatype' if all selected images have a range of values constrained to a particular datatype, in order to minimize the memory footprint.

The resulting image has only integers values in [0,255] and can then be saved as a raw image of unsigned chars (doing so will output a valid .cimg[z] or .gmz file).

If 'store_names' is set to '1', serialization uses the .gmz format to store data in memory (otherwise the .cimg[z] format).

Default values:

- `'datatype=auto', 'is_compressed=1' and 'store_names=1'.`



Example 20 : `image.jpg +serialize uchar +unserialize[-1]`

2.2.43 *shape_circle*

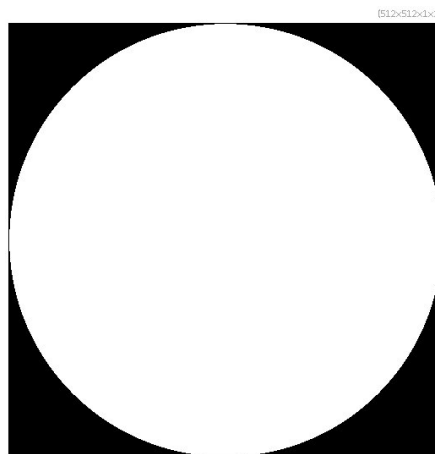
Arguments:

- `_size>=0`

Input a 2D circle binary shape with specified size.

Default value:

- `'size=512'`.



Example 21 : `shape_circle` ,

2.2.44 *shape_cupid*

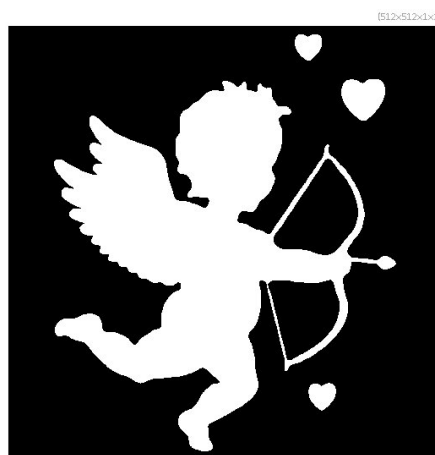
Arguments:

- `_size>=0`

Input a 2D cupid binary shape with specified size.

Default value:

- `'size=512'`.



Example 22 : `shape_cupid` ,

2.2.45 *shape_diamond*

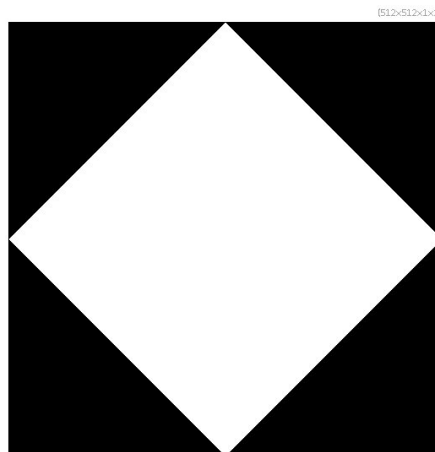
Arguments:

- `_size>=0`

Input a 2D diamond binary shape with specified size.

Default value:

- `'size=512'` .



Example 23 : `shape_diamond` ,

2.2.46 *shape_fern*

Arguments:

- `_size>=0, _density[%]>=0, _angle, 0<=_opacity<=1, _type={ 0=Asplenium
adiantum-nigrum | 1=Thelypteridaceae }`

Input a 2D Barnsley fern with specified size.

Default value:

- `'size=512', 'density=50%', 'angle=30', 'opacity=0.3' and 'type=0'` .



Example 24 : `shape_fern` ,

2.2.47 *shape_gear*

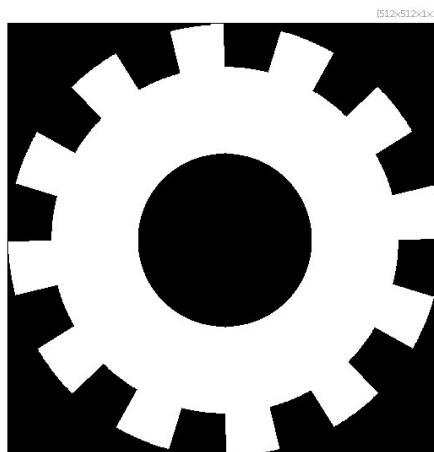
Arguments:

- `_size>=0, _nb_teeth>0, 0<=_height_teeth<=100, 0<=_offset_teeth<=100, 0<=_inner_radius<=100`

Input a 2D gear binary shape with specified size.

Default value:

- `'size=512', 'nb_teeth=12', 'height_teeth=20', 'offset_teeth=0' and 'inner_radius=40'`.



Example 25 : `shape_gear` ,

2.2.48 *shape_heart*

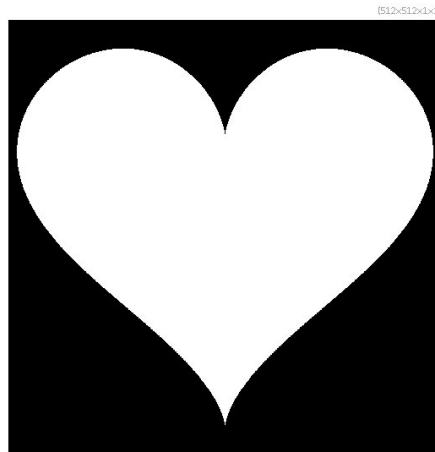
Arguments:

- `_size>=0`

Input a 2D heart binary shape with specified size.

Default value:

- `'size=512'`.



Example 26 : `shape_heart` ,

2.2.49 *shape_polygon*

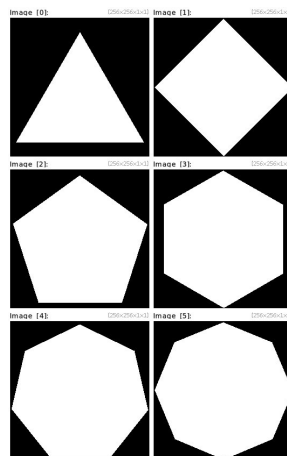
Arguments:

- `_size>=0, _nb_vertices>=3, _angle`

Input a 2D polygonal binary shape with specified geometry.

Default value:

- `'size=512', 'nb_vertices=5' and 'angle=0'`.



Example 27 : `repeat 6 shape_polygon 256, {3+$>} done`

2.2.50 *shape_snowflake*

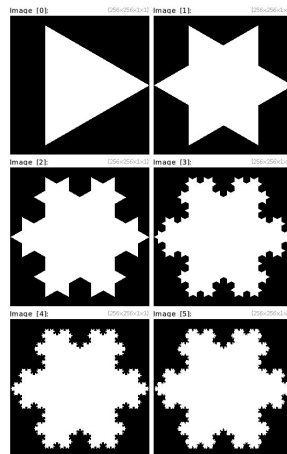
Arguments:

- `size>=0, 0<=_nb_recursions<=6`

Input a 2D snowflake binary shape with specified size.

Default values:

- `'size=512'` and `'nb_recursions=5'`.



Example 28 : `repeat 6 shape_snowflake 256,$> done`

2.2.51 *shape_star*

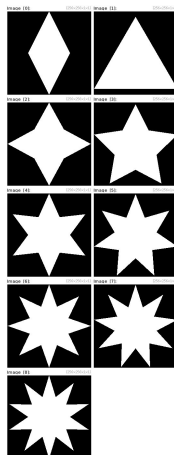
Arguments:

- `_size>=0, _nb_branches>0, 0<=_thickness<=1`

Input a 2D star binary shape with specified size.

Default values:

- `'size=512', 'nb_branches=5' and 'thickness=0.38'`.



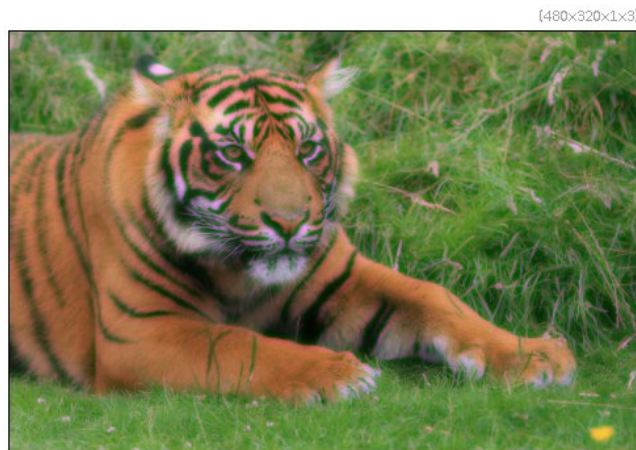
Example 29 : `repeat 9 shape_star 256,{>+2} done`

2.2.52 *shared* (+)

Arguments:

- `x0[%],x1[%],y[%],z[%],v[%]`
- `y0[%],y1[%],z[%],v[%]`
- `z0[%],z1[%],v[%]`
- `v0[%],v1[%]`
- `v0[%]`
- `(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 30 : `image.jpg shared 1 blur[-1] 3 remove[-1]`



Example 31 : `image.jpg repeat {s} shared 25%,75%,0,$> mirror[-1] x remove[-1] done`

Tutorial page:

https://gmics.eu/tutorial/_shared.shtml

2.2.53 *sample*

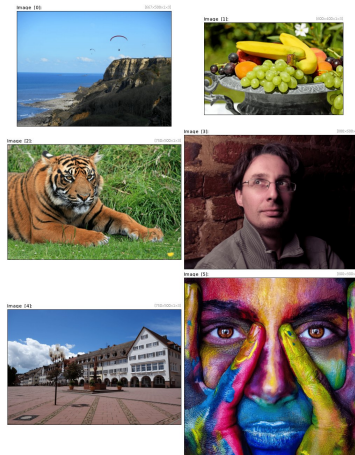
Arguments:

- `_name1={ ? | apples | barbara | boats | bottles | butterfly | cameraman | car | cat | cliff | colorful | david | dog | duck | eagle | elephant | earth | flower | fruits | gmicky | gmicky_mahvin | gmicky_wilber | greece | gummy | house | inside | landscape | leaf | lena | leno | lion | mandrill | monalisa | monkey | parrots | pencils | peppers | roddy | rooster | rose | square | teddy | tiger | tulips | wall | waterfall | zelda }, _name2, ..., _nameN, _width={ >=0 | 0 (auto) }, _height = { >=0 | 0 (auto) }`
- (no arg)

Input a new sample RGB image (opt. with specified size).

(eq. to 'sp').\n).

Argument 'name' can be replaced by an integer which serves as a sample index.



Example 32 : repeat 6 sample done

2.2.54 *srnd* (+)

Arguments:

- value
- (no arg)

Set random generator seed.

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

2.2.55 *string*

Arguments:

- "string"

Insert new image containing the ascii codes of specified string.



Example 33 : `string "foo bar"`

2.2.56 *testimage2d*

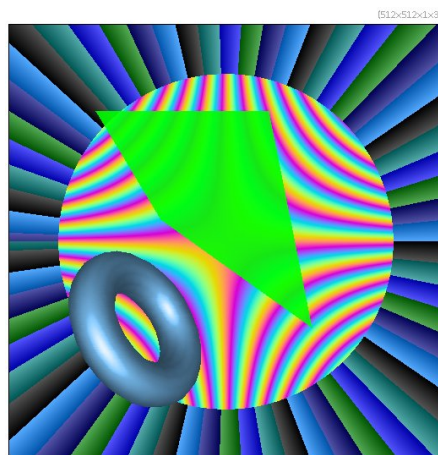
Arguments:

- `_width>0, _height>0, _spectrum>0`

Input a 2D synthetic image.

Default values:

- `'width=512', 'height=width' and 'spectrum=3'.`



Example 34 : `testimage2d 512`

2.2.57 *uncommand (+)*

Arguments:

- `command_name[, _command.name2, ...]`
- `*`

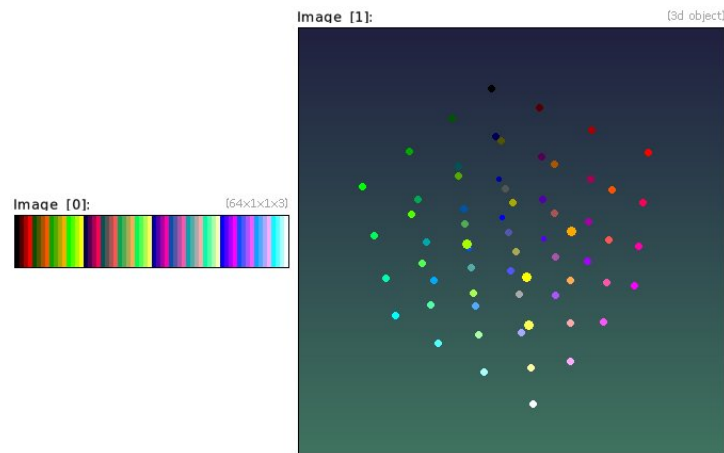
Discard definition of specified custom commands.
 Set argument to '*' for discarding all existing custom commands.

2.2.58 *uniform_distribution*

Arguments:

- `nb_levels>=1, spectrum>=1`

Input set of uniformly distributed spectrum-d points in $[0,1]^{\text{spectrum}}$.



Example 35 : `uniform_distribution 64,3 * 255 +distribution3d circles3d[-1] 10`

2.2.59 *unserialize* (+)

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

2.2.60 *update*

Update commands from the latest definition file on the G'MIC server.
 (eq. to 'up').

2.2.61 *verbose* (+)

Arguments:

- `level`
- `{ + | - }`

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

(eq. to 'v') . \n).

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

Default value:

- `'level=0' .`

2.2.62 *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'`.

2.2.63 *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.

2.2.64 *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') . \n).

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'.

Invoke 'window' with no selection to make the window visible, if it has been closed by the user.

Default values:

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

2.3 List Manipulation

2.3.1 *keep* (+)

Keep only selected images.
(*eq. to 'k'*).



Example 36 : `image.jpg split x keep[0-50%:2] append x`



Example 37 : `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 38 : `image.jpg split x,3 move[1] 0`
 (480x320x1x3)



Example 39 : `image.jpg split x move[50%--1:2] 0 append x`

2.3.3 *name* (+)

Arguments:

- `"name1", "name2", ...`

Set names of selected images. - If the selection contains a single image, then it is assumed the command has a single name argument (possibly containing multiple comas). - If the selection contains more than one image, each command argument defines a single image name for each image of the selection. (*eq. to 'nm'*).



Example 40 : `image.jpg name image blur[image] 2`

Tutorial page:

https://gmic.eu/tutorial/_name.shtml

2.3.4 *remove* (+)

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



Example 41 : `image.jpg split x remove[30%-70%] append x`



Example 42 : `image.jpg split x remove[0-50%:2] append x`

2.3.5 *remove_duplicates*

Remove duplicates images in the selected images list.



Example 43 : `(1,2,3,4,2,4,3,1,3,4,2,1) split x remove_duplicates append x`

2.3.6 *remove_empty*

Remove empty images in the selected image list.

2.3.7 *reverse (+)*

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



Example 44 : `image.jpg split x,3 reverse[-2,-1]`
(480x320x1x3)



Example 45 : `image.jpg split x,-16 reverse[50%-100%] append x`

2.3.8 *sort_list*

Arguments:

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

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

Default values:

- `'ordering='+', 'criterion=i'.`



Example 46 : (1;4;7;3;9;2;4;7;6;3;9;1;0;3;3;2) split y sort_list +,i append y

2.3.9 *sort_str*

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

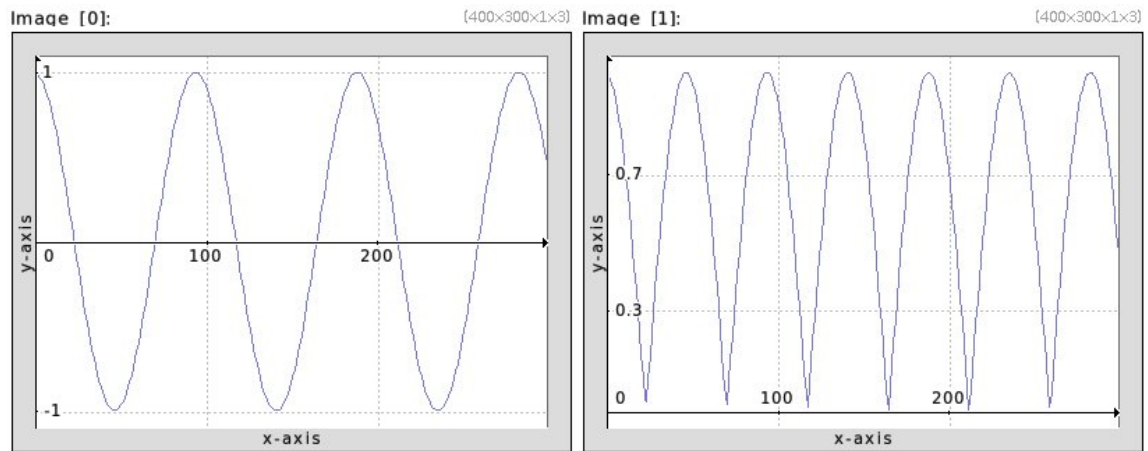
2.4 Mathematical Operators

2.4.1 *abs (+)*

Compute the pointwise absolute values of selected images.



Example 47 : image.jpg +sub {ia} abs[-1]



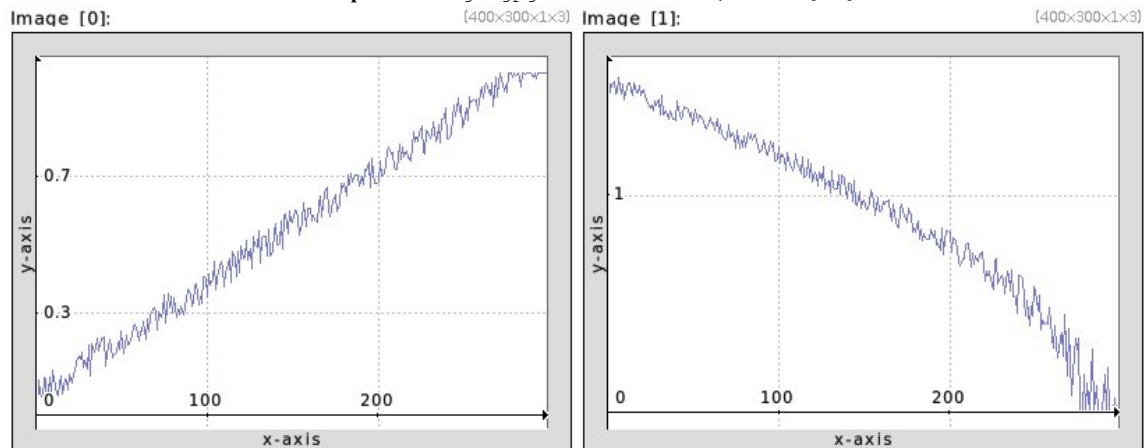
Example 48 : `300,1,1,1,'cos(20*x/w)'+abs display_graph 400,300`

2.4.2 *acos* (+)

Compute the pointwise arccosine of selected images.



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



Example 50 : `300,1,1,1,'cut(x/w+0.1*u,0,1)'+acos display_graph 400,300`

Tutorial page:

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

2.4.3 *acosh* (+)

Compute the pointwise hyperbolic arccosine of selected images.

2.4.4 *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 51 : `image.jpg +add 30% cut 0,255`



Example 52 : `image.jpg +blur 5 normalize 0,255 add[1] [0]`



Example 53 : `image.jpg add '80*cos(80*(x/w-0.5)*(y/w-0.5)+c)' cut 0,255`

[480x320x1x3]



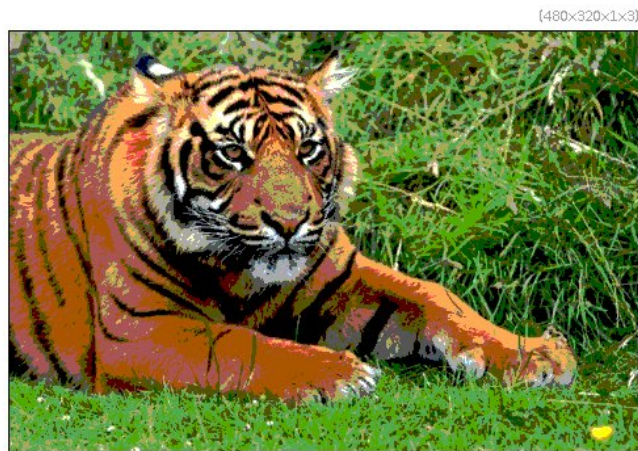
Example 54 : `image.jpg repeat 9 +rotate[0] {$>*36},1,0,50%,50% done add div 10`

2.4.5 *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 55 : `image.jpg` and `{128+64}`



Example 56 : `image.jpg +mirror x` and

2.4.6 *argmax*

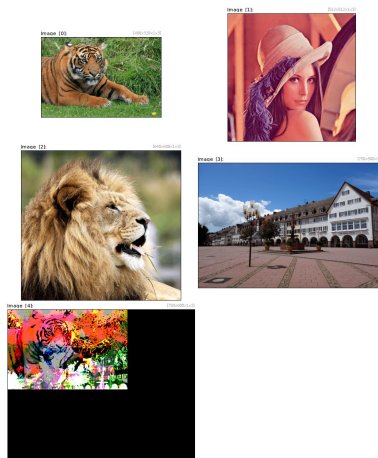
Compute the argmax of selected images. Returns a single image with each pixel value being the index of the input image with maximal value.



Example 57 : `image.jpg sample lena,lion,square +argmax`

2.4.7 *argmin*

Compute the argmin of selected images. Returns a single image with each pixel value being the index of the input image with minimal value.



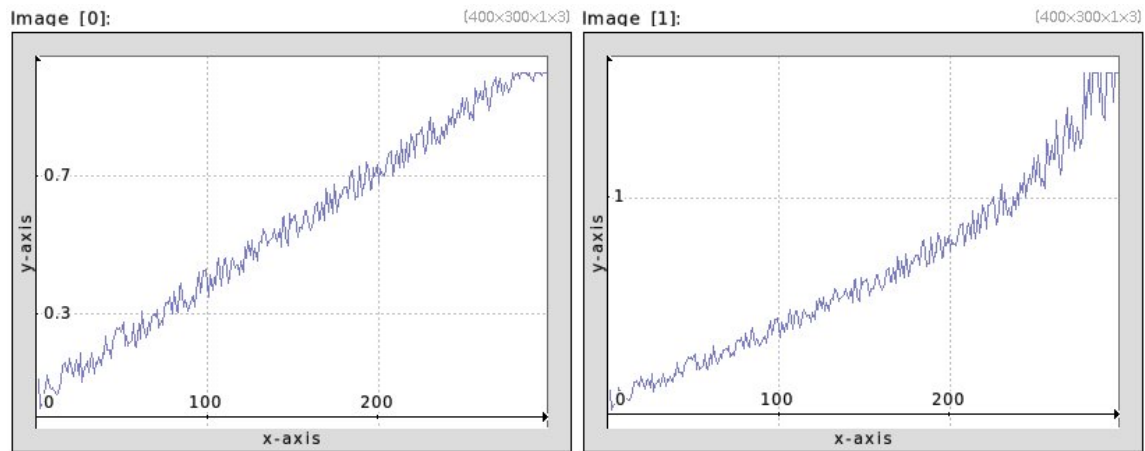
Example 58 : `image.jpg sample lena,lion,square +argmin`

2.4.8 *asin (+)*

Compute the pointwise arcsine of selected images.



Example 59 : `image.jpg +normalize -1,1 asin[-1]`



Example 60 : `300,1,1,1,'cut (x/w+0.1*u,0,1)' +asin display_graph 400,300`

Tutorial page:

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

2.4.9 *asinh* (+)

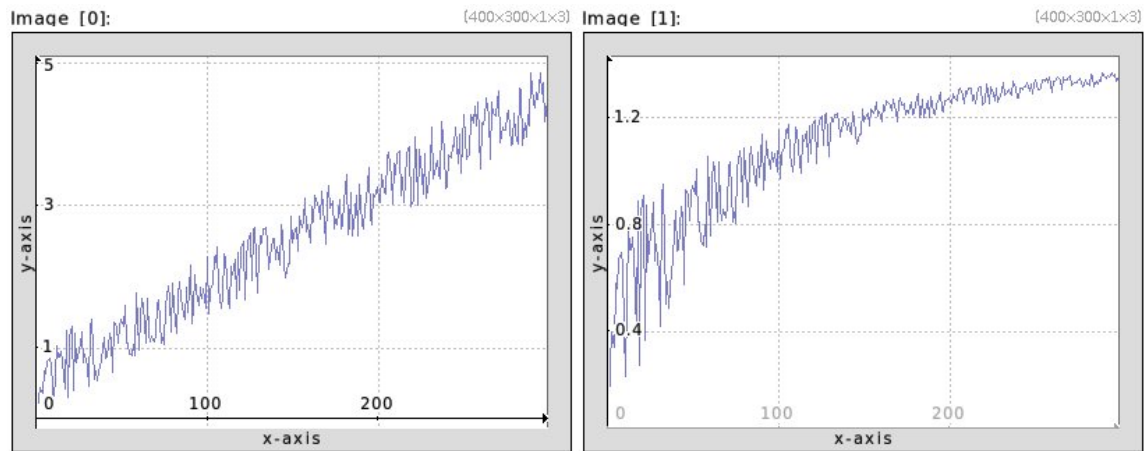
Compute the pointwise hyperbolic arcsine of selected images.

2.4.10 *atan* (+)

Compute the pointwise arctangent of selected images.



Example 61 : `image.jpg +normalize 0,8 atan[-1]`



Example 62 : `300,1,1,1,'4*x/w+u' +atan display_graph 400,300`

Tutorial page:

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

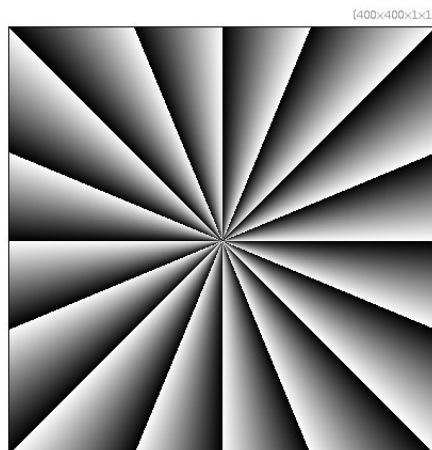
2.4.11 *atan2* (+)

Arguments:

- `[x_argument]`

Compute the pointwise oriented arctangent of selected images.

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



Example 63 : `(-1,1) (-1;1) resize 400,400,1,1,3 atan2[1] [0] keep[1] mod {pi/8}`

Tutorial page:

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

2.4.12 *atanh* (+)

Compute the pointwise hyperbolic arctangent of selected images.

2.4.13 *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 `'<<'`).



Example 64 : `image.jpg bsl 'round(3*x/w,0)' cut 0,255`

2.4.14 *bsr* (+)

Arguments:

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

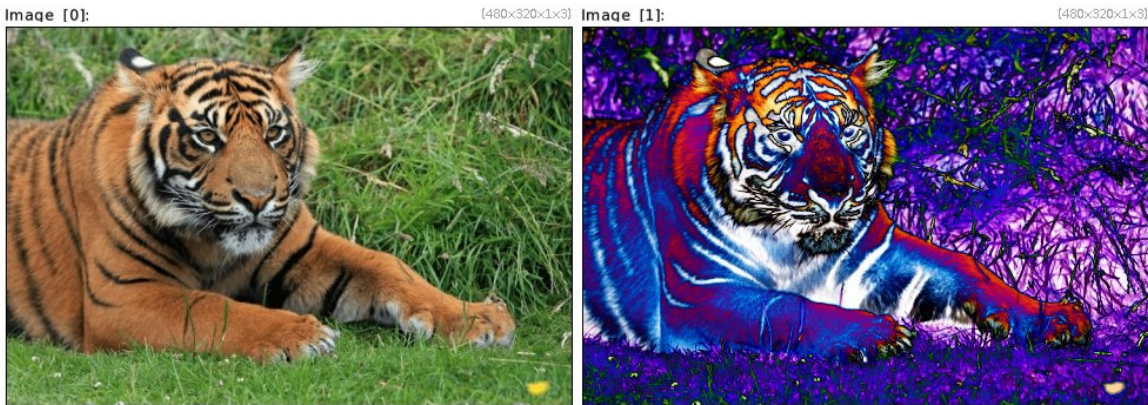
Compute the bitwise right shift of selected images with specified value, image or” mathematical expression, or compute the pointwise sequential bitwise right shift of selected images. (eq. to `'>>'`).



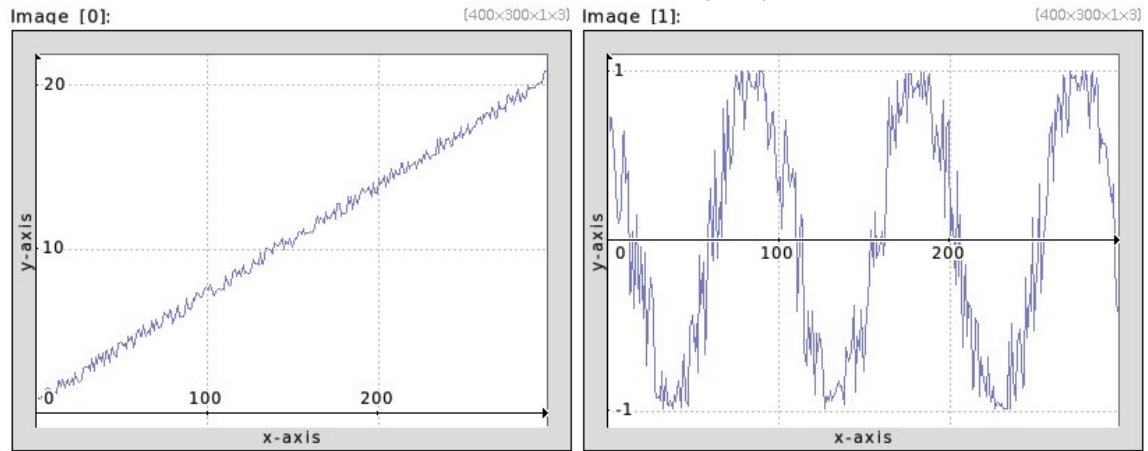
Example 65 : `image.jpg bsr 'round(3*x/w,0)' cut 0,255`

2.4.15 *cos* (+)

Compute the pointwise cosine of selected images.



Example 66 : `image.jpg +normalize 0,{2*pi} cos[-1]`



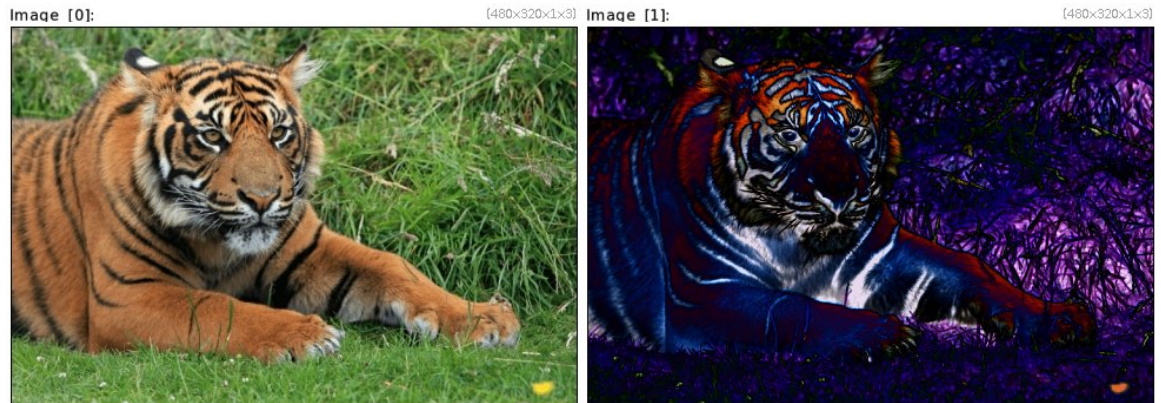
Example 67 : `300,1,1,1,'20*x/w+u' +cos display_graph 400,300`

Tutorial page:

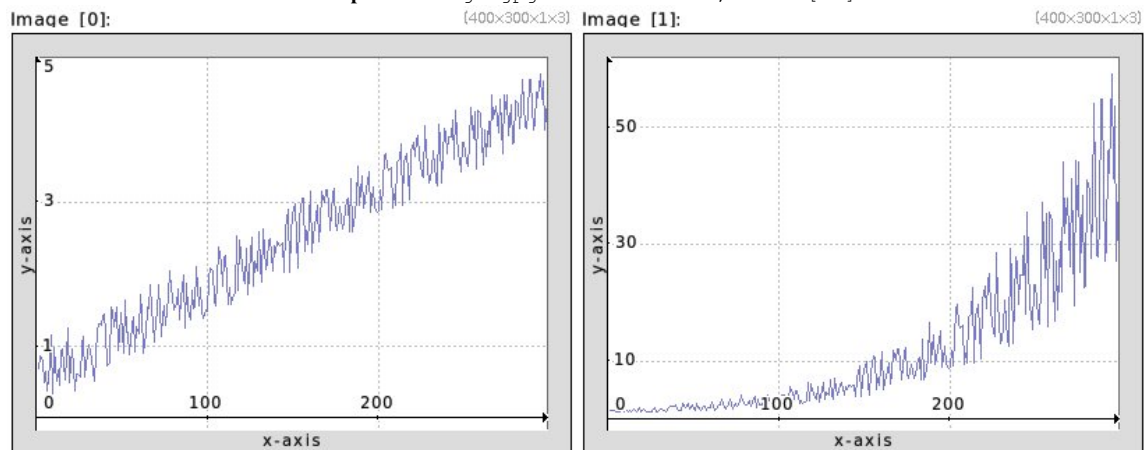
<https://gmic.eu/tutorial/trigometric-and-inverse-trigometric-commands.shtml>

2.4.16 *cosh* (+)

Compute the pointwise hyperbolic cosine of selected images.



Example 68 : `image.jpg +normalize -3,3 cosh[-1]`



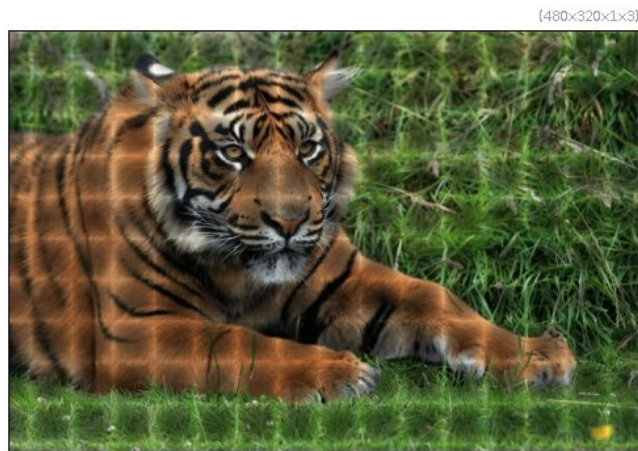
Example 69 : `300,1,1,1,'4*x/w+u' +cosh display_graph 400,300`

2.4.17 *div* (+)**Arguments:**

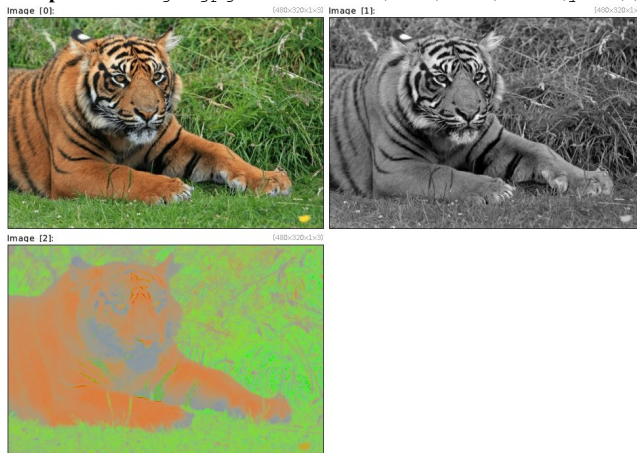
- `value[%]`
- `[image]`
- `'formula'`
- `(no arg)`

Divide selected images by specified value, image or mathematical expression, or compute the pointwise quotient of selected images.

(eq. to `'/'`).



Example 70 : `image.jpg div '1+abs(cos(x/10)*sin(y/10))'`



Example 71 : `image.jpg +norm add[-1] 1 +div`

2.4.18 *div_complex*

Arguments:

- `[divider_real,divider_imag],_epsilon>=0`

Perform division of the selected complex pairs (real1,imag1,...,realN,imagN) of images by specified complex pair of images (divider_real,divider_imag).

In complex pairs, the real image must be always located before the imaginary image in the image list.

Default value:

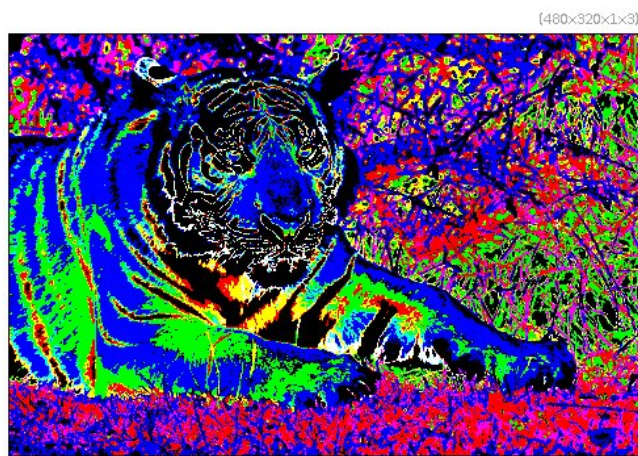
- `'epsilon=1e-8'`.

2.4.19 *eq (+)*

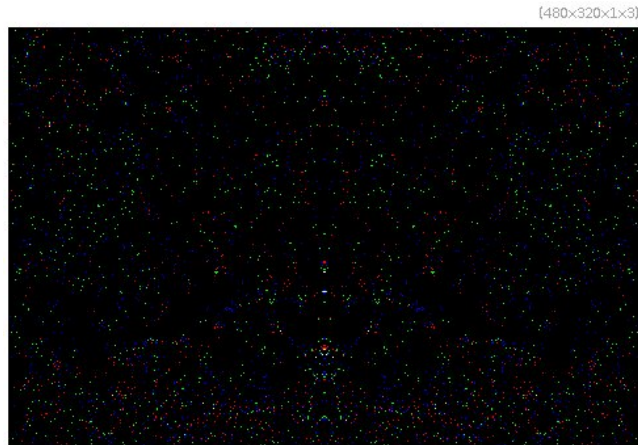
Arguments:

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



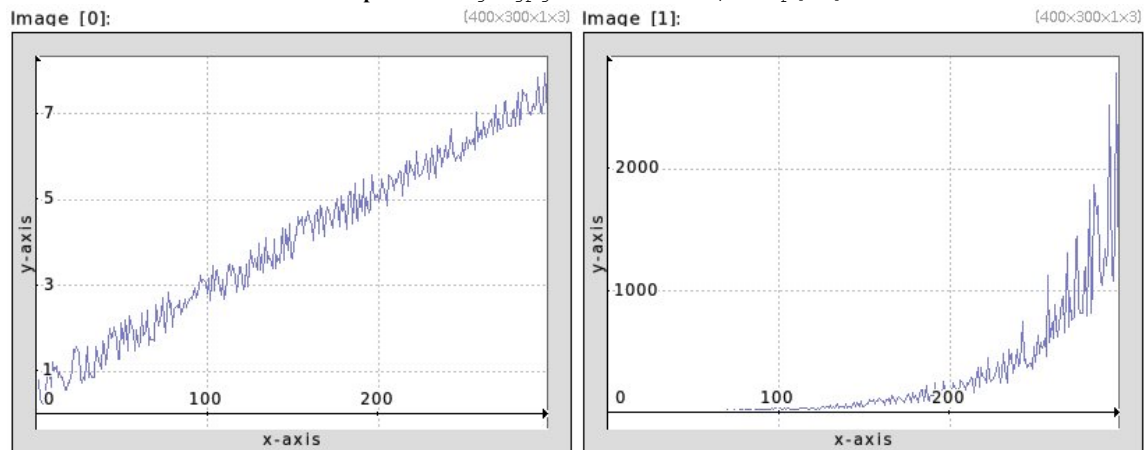
Example 73: `image.jpg +mirror x eq`

2.4.20 *exp* (+)

Compute the pointwise exponential of selected images.



Example 74: `image.jpg +normalize 0,2 exp[-1]`



Example 75: `300,1,1,1,'7*x/w+u' +exp display-graph 400,300`

2.4.21 *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 '`>=`').



Example 76 : `image.jpg ge {ia}`



Example 77 : `image.jpg +mirror x ge`

2.4.22 *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 78 : `image.jpg gt {ia}`



Example 79 : `image.jpg +mirror x gt`

2.4.23 *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 80 : `image.jpg le {ia}`



Example 81 : `image.jpg +mirror x le`

2.4.24 *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 82 : `image.jpg lt {ia}`



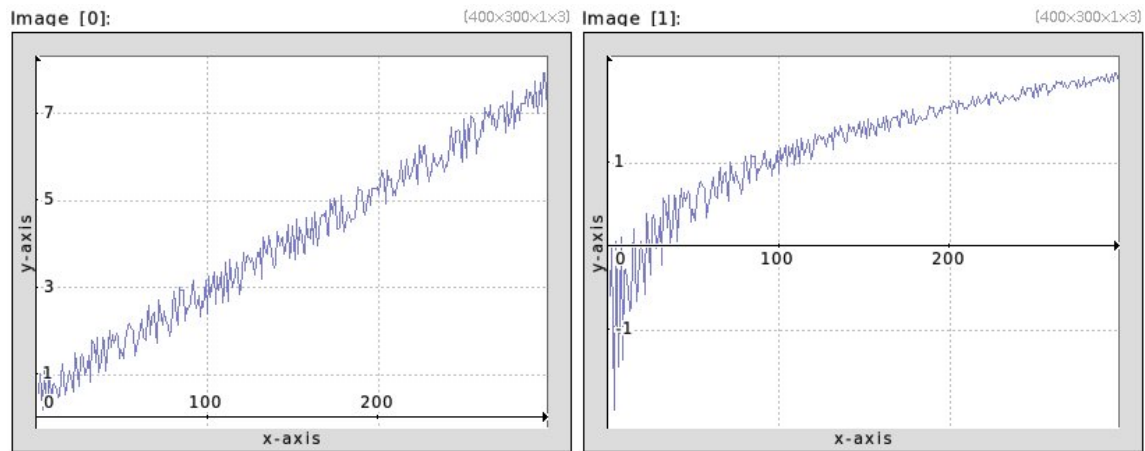
Example 83 : `image.jpg +mirror x lt`

2.4.25 $\log (+)$

Compute the pointwise base-e logarithm of selected images.



Example 84 : `image.jpg +add 1 log[-1]`



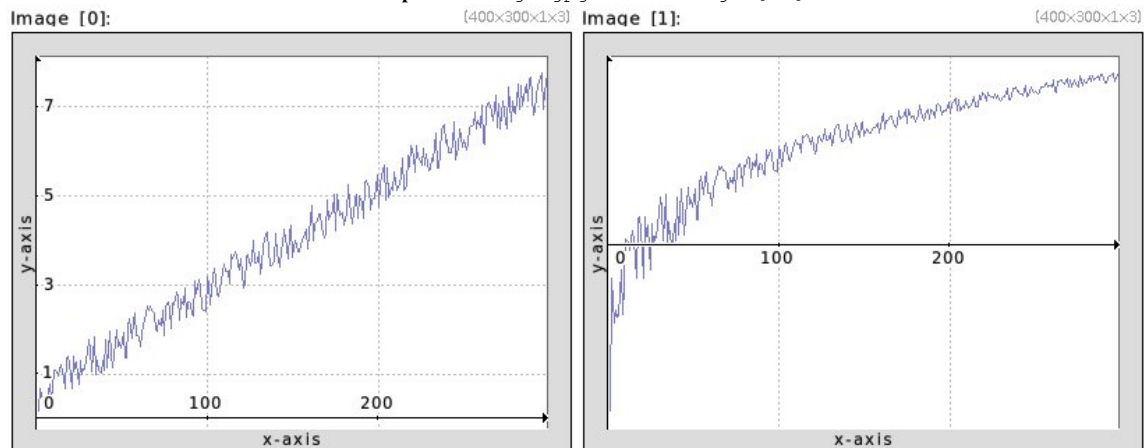
Example 85 : `300,1,1,1,'7*x/w+u' +log display-graph 400,300`

2.4.26 *log10* (+)

Compute the pointwise base-10 logarithm of selected images.



Example 86 : `image.jpg +add 1 log10[-1]`



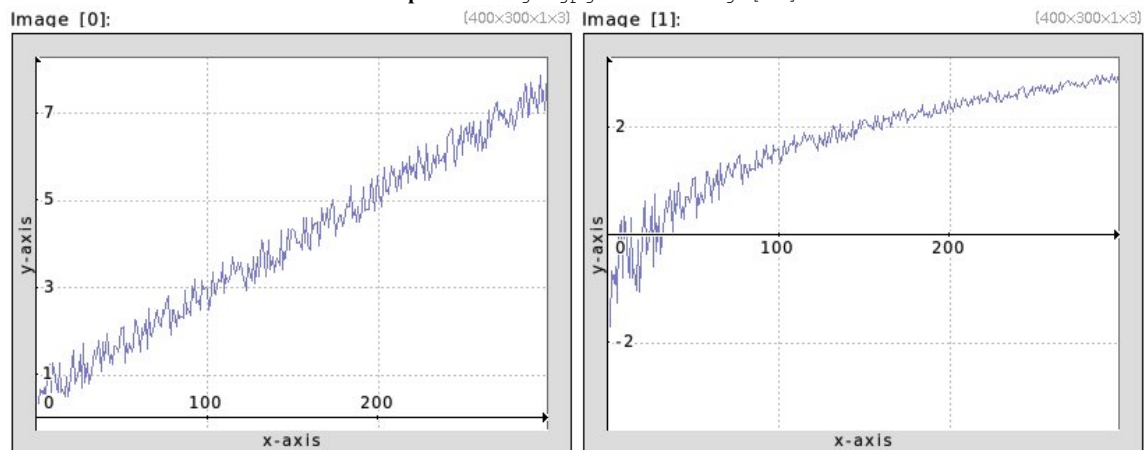
Example 87 : `300,1,1,1,'7*x/w+u' +log10 display-graph 400,300`

2.4.27 \log_2 (+)

Compute the pointwise base-2 logarithm of selected images



Example 88 : `image.jpg +add 1 log2[-1]`



Example 89 : `300,1,1,1,'7*x/w+u' +log2 display_graph 400,300`

2.4.28 \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 90 : `image.jpg +mirror x max`



Example 91 : `image.jpg max 'R=((x/w-0.5)^2+(y/h-0.5)^2)^0.5;255*R'`

2.4.29 *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 'm/').

2.4.30 *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 92 : `image.jpg +mirror x min`



Example 93 : `image.jpg min 'R=((x/w-0.5)^2+(y/h-0.5)^2)^0.5;255*R'`

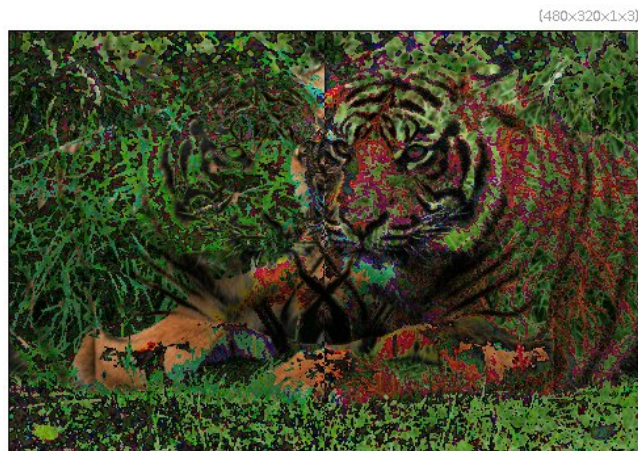
2.4.31 *mod* (+)

Arguments:

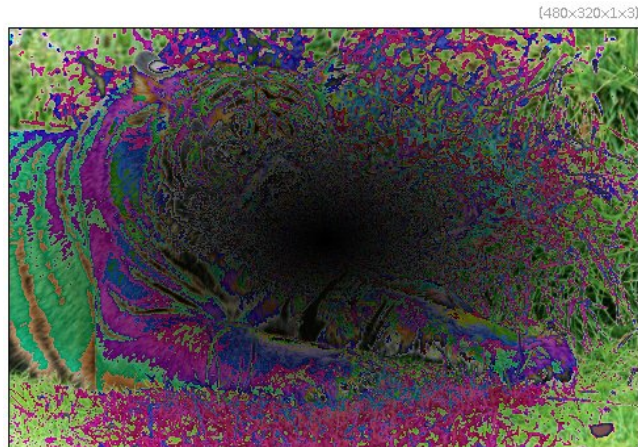
- value[%]
- [image]
- 'formula'
- (no arg)

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 94 : `image.jpg +mirror x mod`



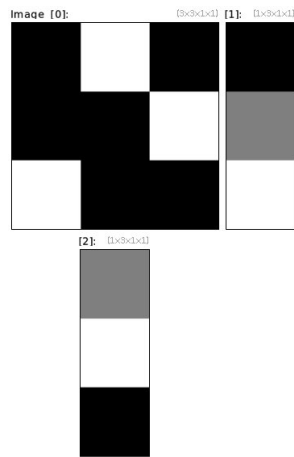
Example 95 : `image.jpg mod 'R=((x/w-0.5)^2+(y/h-0.5)^2)^0.5;255*R'`

2.4.32 *mmul* (+)

Arguments:

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

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* `'m*'`).



Example 96 : `(0,1,0;0,0,1;1,0,0) (1;2;3) +mmul`

2.4.33 *mul* (+)

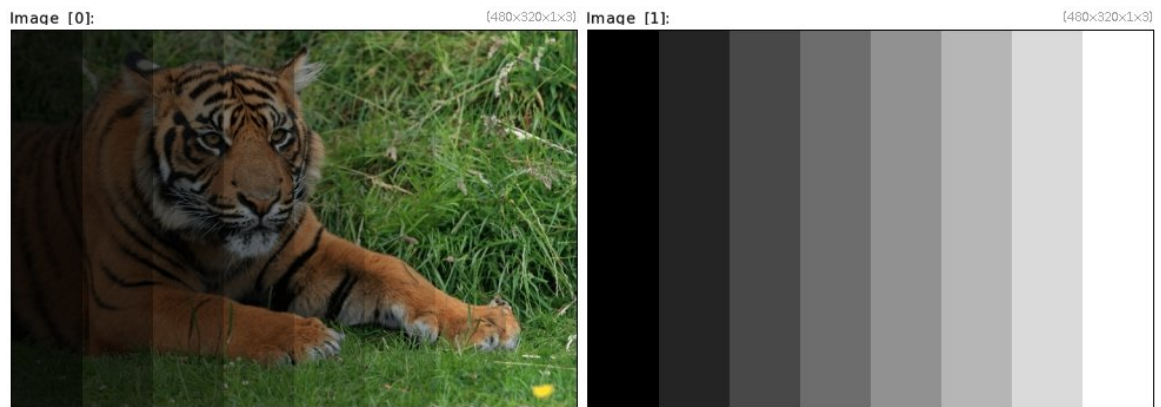
Arguments:

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

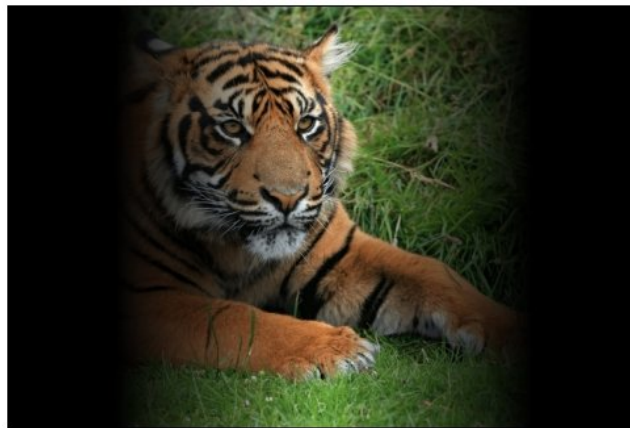
Multiply selected images by specified value, image or mathematical expression, or compute the pointwise product of selected images.
(*eq. to ' * '*).



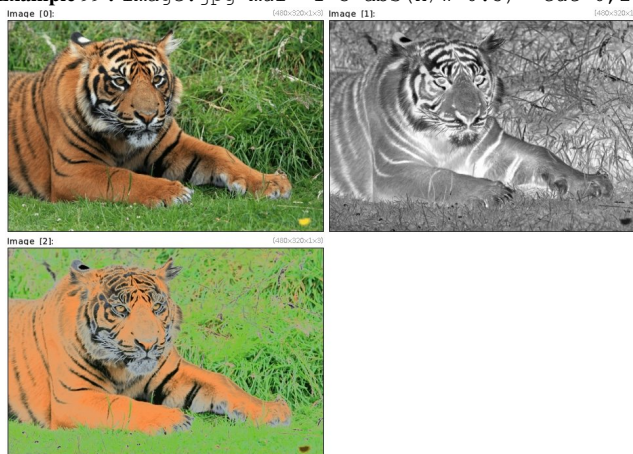
Example 97 : `image.jpg +mul 2 cut 0,255`



Example 98 : `image.jpg (1,2,3,4,5,6,7,8) resize[-1] [0] mul[0] [-1]`
 (480x320x1x3)



Example 99 : `image.jpg mul '1-3*abs(x/w-0.5)' cut 0,255`



Example 100 : `image.jpg +luminance negate[-1] +mul`

2.4.34 *mul_channels*

Arguments:

- `value1, value2, ..., valueN`

Multiply channels of selected images by specified sequence of values.



Example 101 : `image.jpg +mul_channels 1,0.5,0.8`

2.4.35 *mul_complex*

Arguments:

- `[multiplier_real,multiplier_imag]`

Perform multiplication of the selected complex pairs (`real1,imag1,...,realN,imagN`) of images by specified complex pair of images (`multiplier_real,multiplier_imag`).

In complex pairs, the real image must be always located before the imaginary image in the image list.

2.4.36 *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 102 : `image.jpg round 40 neq {round(ia, 40)}`

2.4.37 *or* (+)

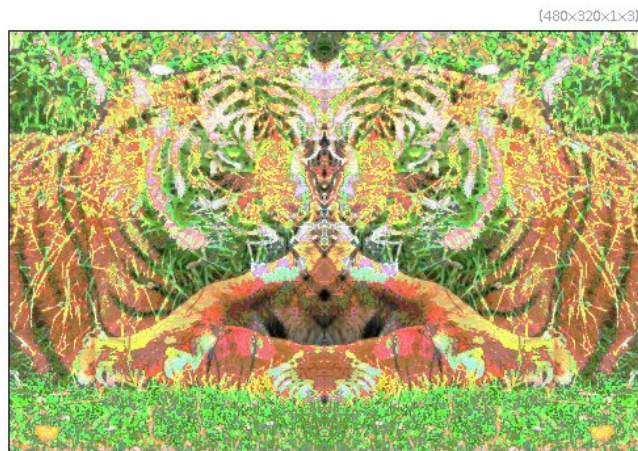
Arguments:

- `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 103 : `image.jpg or 128`



Example 104 : `image.jpg +mirror x or`

2.4.38 *pow* (+)

Arguments:

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

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



Example 105 : `image.jpg div 255 +pow 0.5 mul 255`



Example 106 : `image.jpg gradient pow 2 add pow 0.2`

2.4.39 *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 107 : `image.jpg rol 'round(3*x/w,0)' cut 0,255`

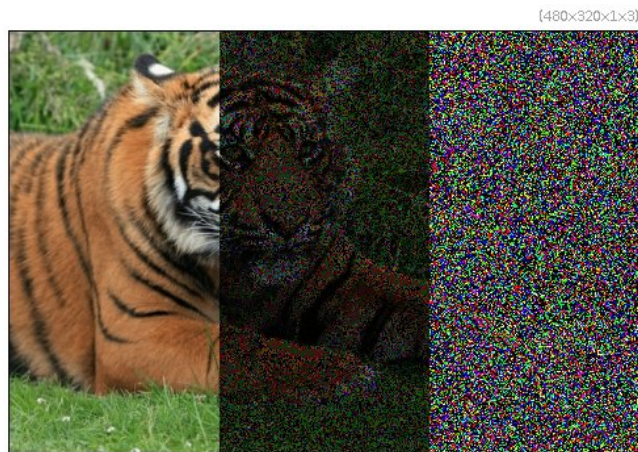
2.4.40 *ror* (+)

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.



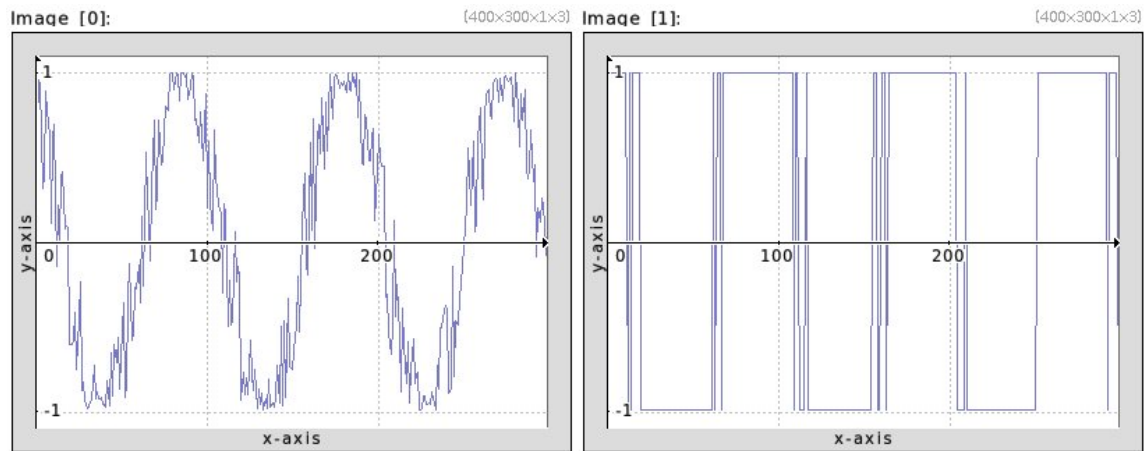
Example 108 : `image.jpg ror 'round(3*x/w,0)' cut 0,255`

2.4.41 *sign (+)*

Compute the pointwise sign of selected images.



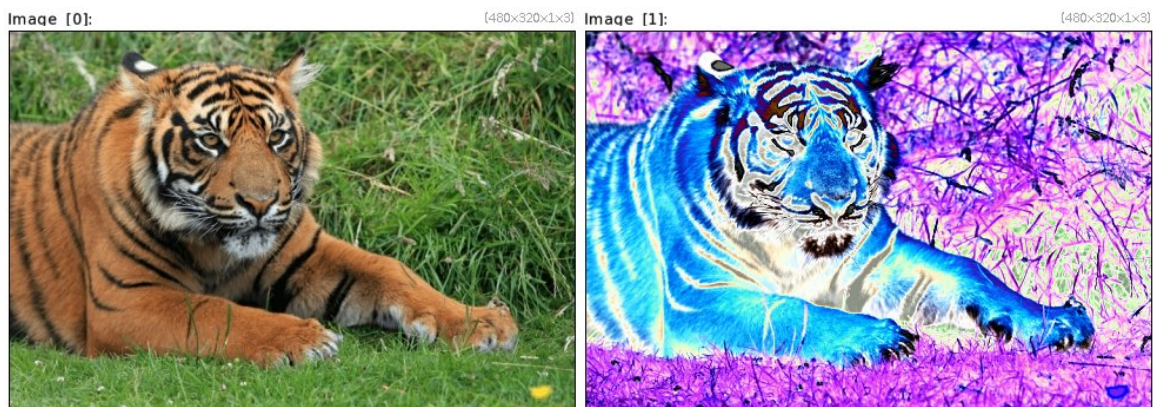
Example 109 : `image.jpg +sub {ia} sign[-1]`



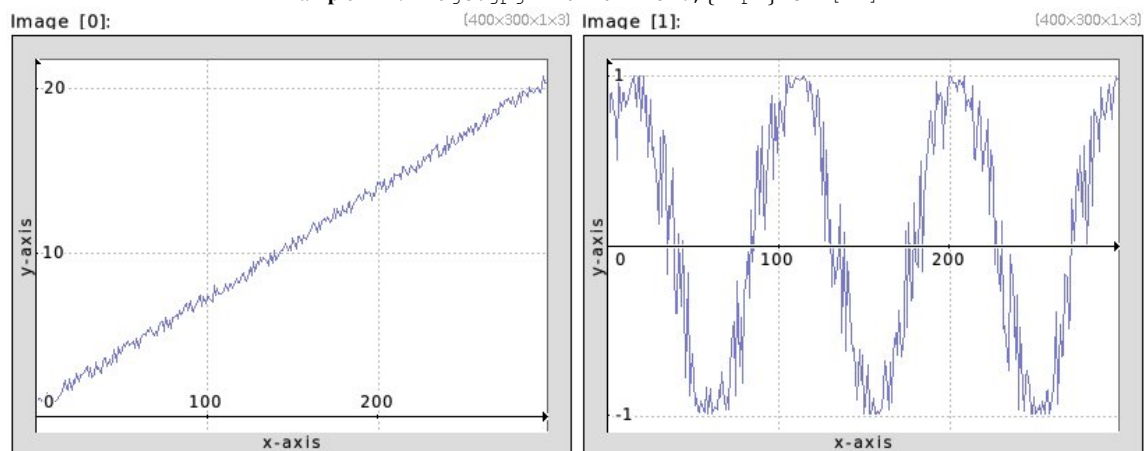
Example 110 : `300,1,1,1,'cos(20*x/w+u)' +sign display-graph 400,300`

2.4.42 *sin* (+)

Compute the pointwise sine of selected images.



Example 111 : `image.jpg +normalize 0,{2*pi} sin[-1]`



Example 112 : `300,1,1,1,'20*x/w+u' +sin display-graph 400,300`

Tutorial page:

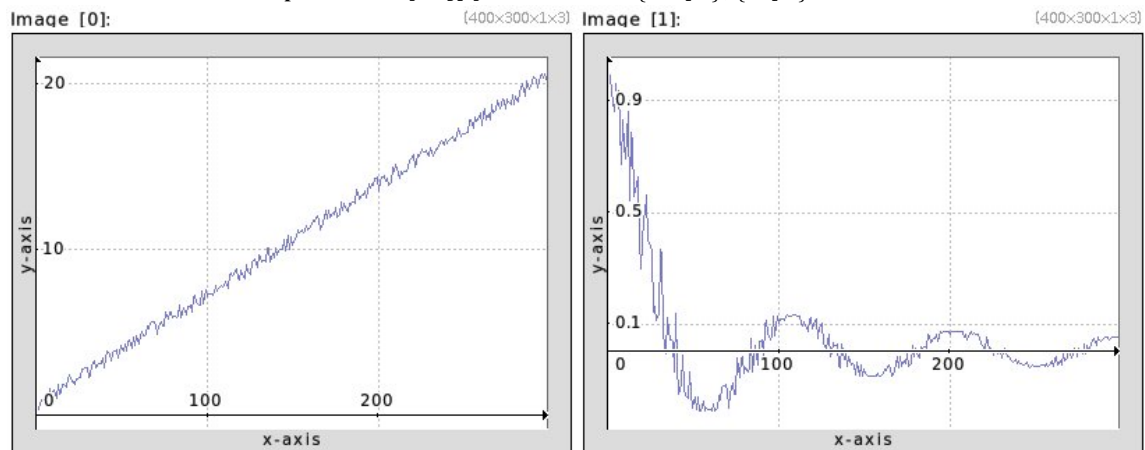
<https://gmic.eu/tutorial/trigometric-and-inverse-trigometric-commands.shtml>

2.4.43 *sinc* (+)

Compute the pointwise sinc function of selected images.



Example 113: `image.jpg +normalize {-2*pi},{2*pi} sinc[-1]`



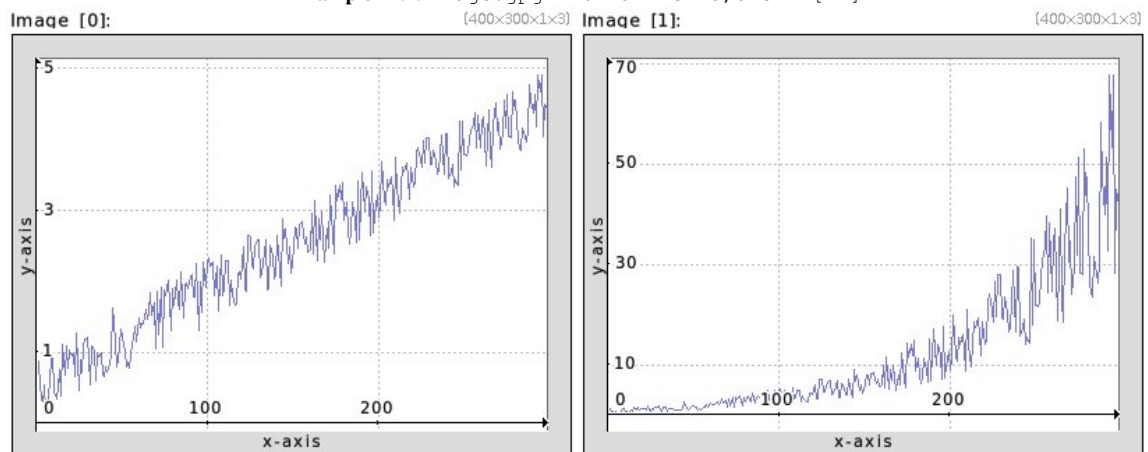
Example 114: `300,1,1,1,'20*x/w+u' +sinc display-graph 400,300`

2.4.44 *sinh* (+)

Compute the pointwise hyperbolic sine of selected images.



Example 115: `image.jpg +normalize -3,3 sinh[-1]`



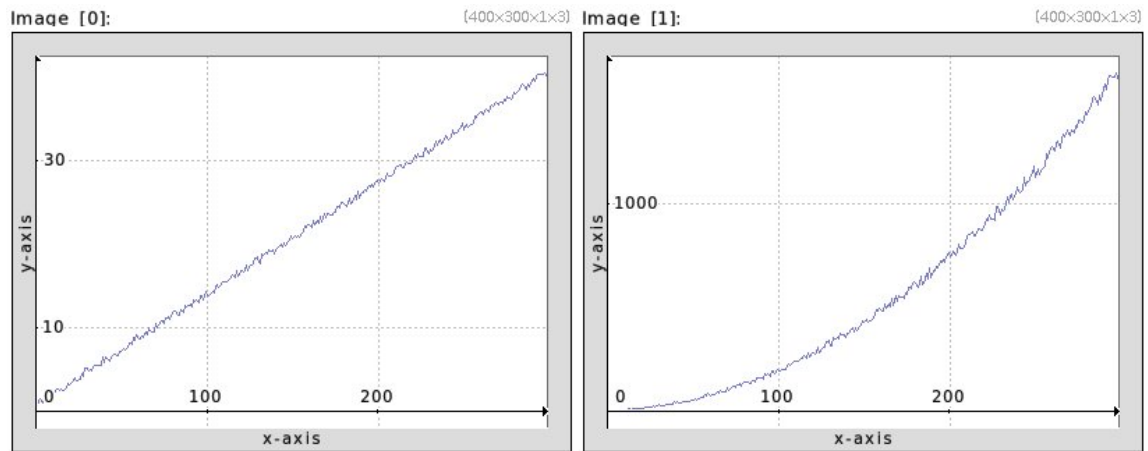
Example 116: `300,1,1,1,'4*x/w+u' +sinh display_graph 400,300`

2.4.45 *sqr* (+)

Compute the pointwise square function of selected images.



Example 117: `image.jpg +sqr`

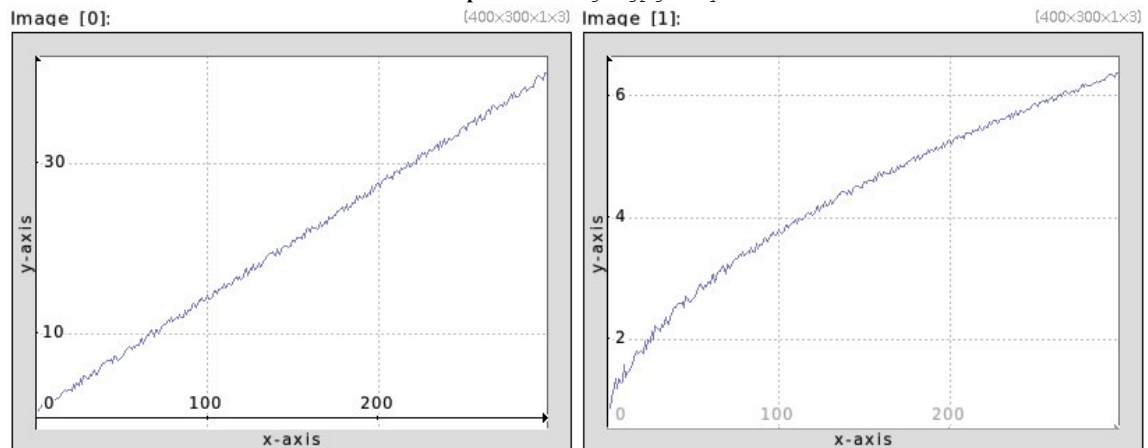


2.4.46 *sqr* (+)

Compute the pointwise square root of selected images.



Example 119 : `image.jpg +sqr`



2.4.47 *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.

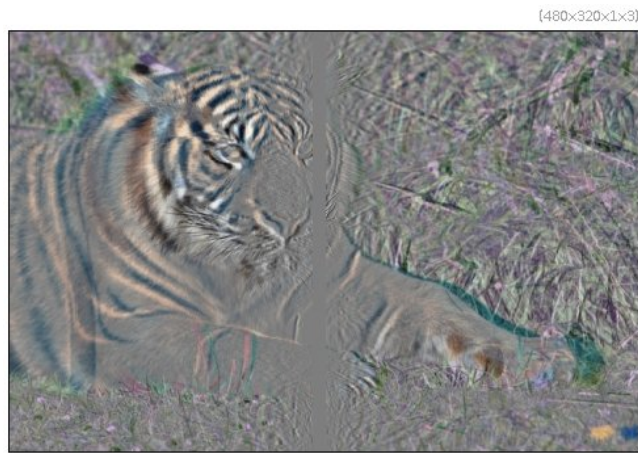
(*eq. to ' - '*).



Example 121 : `image.jpg +sub 30% cut 0,255`



Example 122 : `image.jpg +mirror x sub[-1] [0]`



Example 123 : `image.jpg sub 'i (w/2+0.9*(x-w/2),y)'`



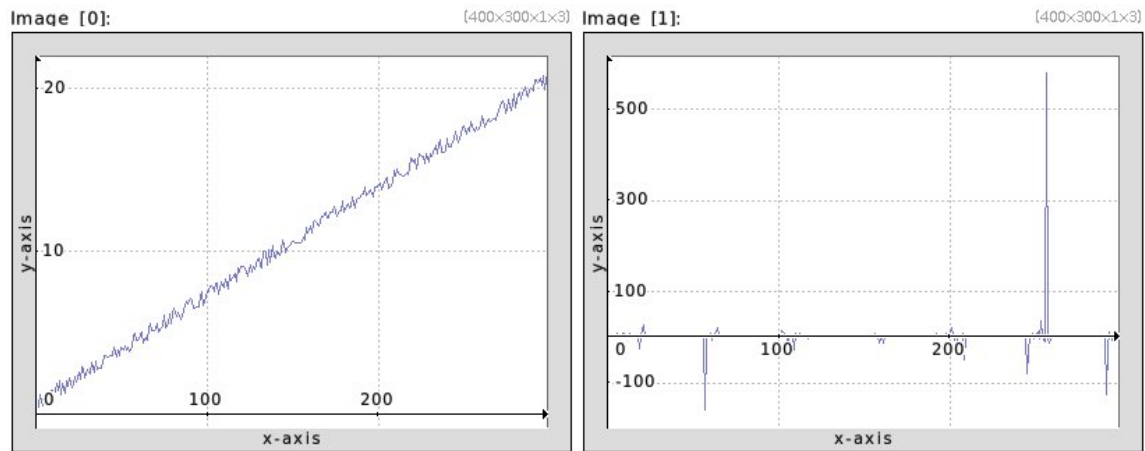
Example 124 : `image.jpg +mirror x sub`

2.4.48 *tan* (+)

Compute the pointwise tangent of selected images.



Example 125 : `image.jpg +normalize {-0.47*pi},{0.47*pi} tan[-1]`



Example 126 : `300,1,1,1,'20*x/w+u' +tan display_graph 400,300`

Tutorial page:

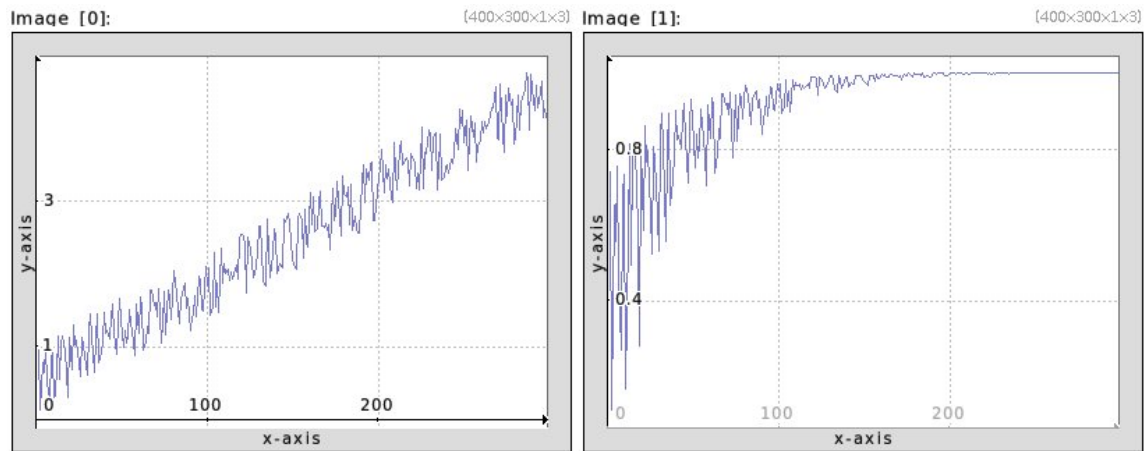
<https://gmic.eu/tutorial/trigometric-and-inverse-trigometric-commands.shtml>

2.4.49 *tanh* (+)

Compute the pointwise hyperbolic tangent of selected images.



Example 127 : `image.jpg +normalize -3,3 tanh[-1]`



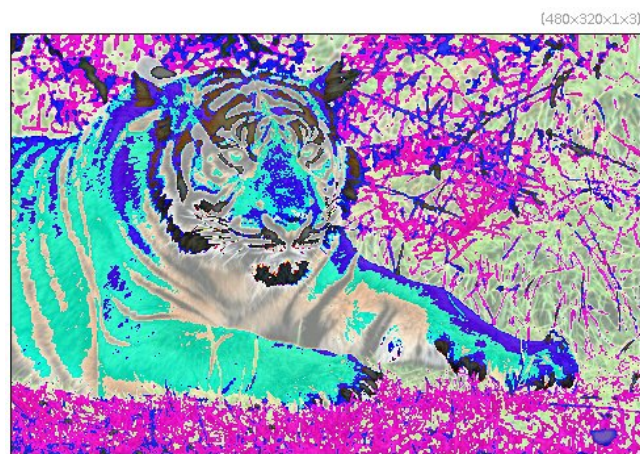
Example 128 : `300,1,1,1,'4*x/w+u' +tanh display_graph 400,300`

2.4.50 *xor* (+)

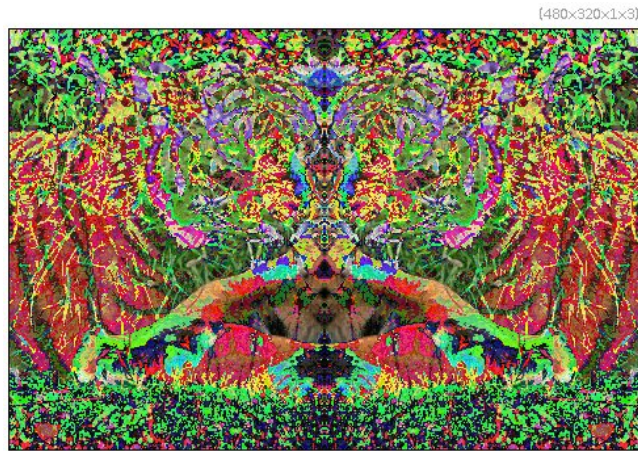
Arguments:

- `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 129 : `image.jpg xor 128`



Example 130 : `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 131 : `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 132 : `image.jpg +apply-gamma 2`

2.5.3 *balance_gamma*

Arguments:

- `_ref_color1, ...`

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

Default value:

- `'ref_color1=128'`



Example 133 : `image.jpg +balance_gamma 128,64,64`

2.5.4 *cast*

Arguments:

- `datatype.source,datatype.target`

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

'datatype.source' and 'datatype.target' can be { uchar | char | ushort | short | uint | int | uint64 | int64 | float | double }.

2.5.5 *complex2polar*

Compute complex to polar transforms of selected images.



Example 134 : `image.jpg +fft complex2polar[-2,-1] log[-2] shift[-2] 50%,50%,0,0,2 remove[-1]`

2.5.6 *compress_clut*

Arguments:

- `_max_error>0, _avg_error>0, _max_nbpoints>=8`

Compress selected color LUTs as sequences of colored keypoints.

Default values:

- `'max_error=8', 'avg_error=2' and 'max_nbpoints=2048'.`

2.5.7 *compress_rle*

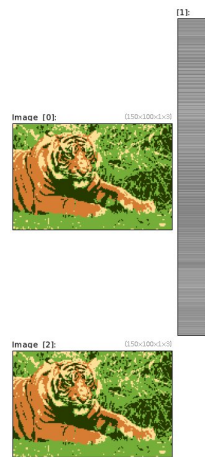
Arguments:

- `_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'.`



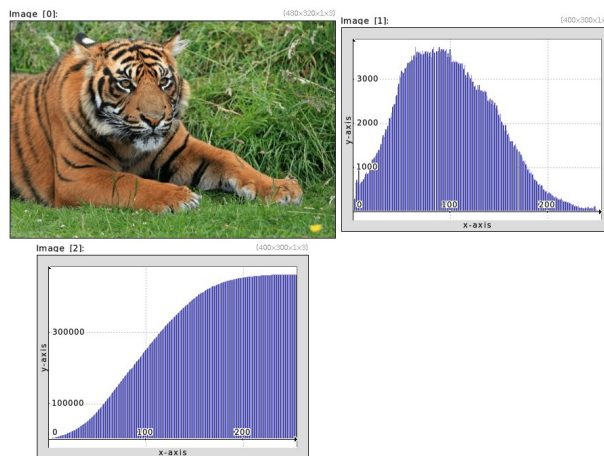
Example 135 : `image.jpg resize2dy 100 quantize 4 round +compress_rle , +decompress_rle[-1]`

2.5.8 *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 136 : `image.jpg +histogram +cumulate[-1] display_graph[-2,-1] 400,300,3`

2.5.9 *cut (+)*

Arguments:

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

Cut values of selected images in specified range.

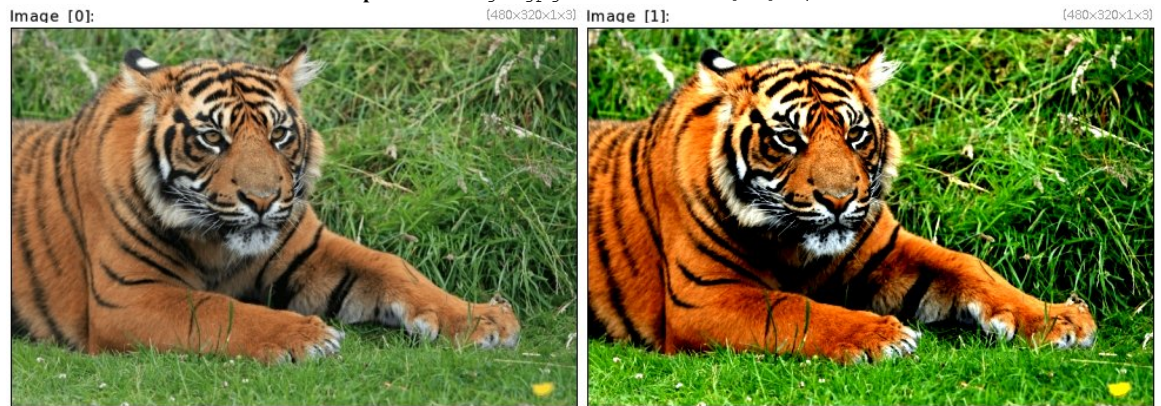
(eq. to 'c') .\n).

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

In interactive mode, the chosen cut values are returned in the status.



Example 137 : image.jpg +add 30% cut [-1] 0,255



Example 138 : image.jpg +cut 25%,75%

2.5.10 *decompress_clut_rbf*

Arguments:

- `_width>0, _height>0, _depth>0`

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

Default value:

- `'width=height=depth=48'.`

2.5.11 *decompress_clut*

Arguments:

- `_width>0, _height>0, _depth>0`

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

Default value:

- `'width=height=depth=64'`.

2.5.12 *decompress_rle*

Decompress selected data vectors, using RLE algorithm.

2.5.13 *discard* (+)

Arguments:

- `_value1, _value2, ...`
- `{ x | y | z | c } ... { x | y | z | c }, _value1, _value2, ...`
- (no arg)

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 139 : `(1;2;3;4;3;2;1) +discard 2`



Example 140 : `(1,2,2,3,3,3,4,4,4,4) +discard x`

2.5.14 *eigen2tensor*

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

Tutorial page:

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

2.5.15 *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 | uint64 | int64 | float | double }.

2.5.16 *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 141 : `image.jpg +equalize`



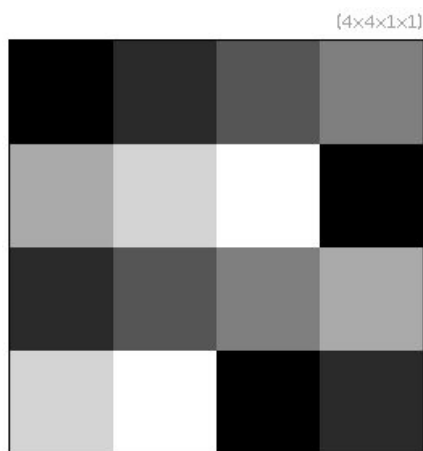
Example 142 : `image.jpg +equalize 4,0,128`

2.5.17 *fill* (+)

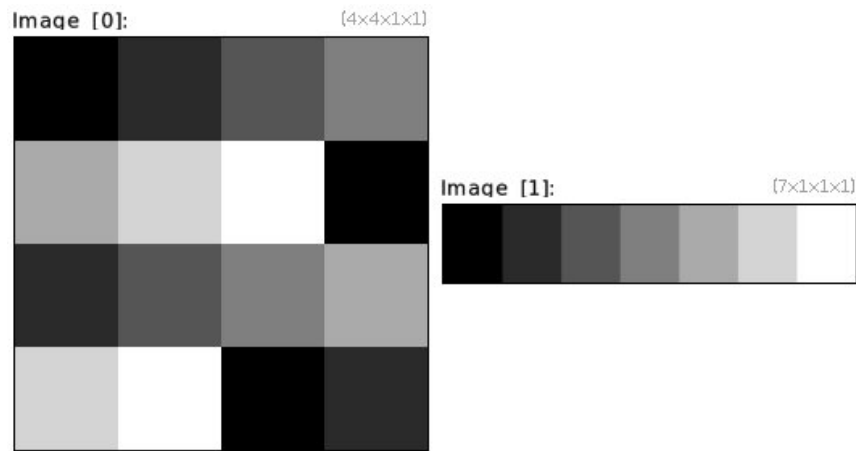
Arguments:

- `value1, _value2, ...`
- `[image]`
- `'formula'`

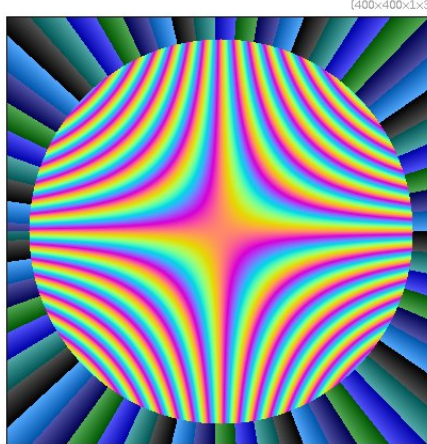
Fill selected images with values read from the specified value list, existing image or mathematical expression. Single quotes may be omitted in 'formula'. (eq. to 'f').



Example 143 : `4,4 fill 1,2,3,4,5,6,7`



Example 144 : `4,4 (1,2,3,4,5,6,7) fill [-2] [-1]`



Example 145 : `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))))"`

Tutorial page:

https://gmic.eu/tutorial/_fill.shtml

2.5.18 *index (+)*

Arguments:

- `{ [palette] | predefined_palette }, 0<=_dithering<=1, _map_palette={ 0 | 1 }`

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

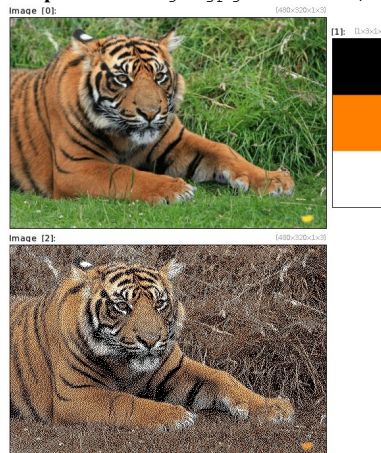
'predefined_palette' can be { 0=default | 1=HSV | 2=lines | 3=hot | 4=cool | 5=jet | 6=flag | 7=cube }.

Default values:

- `'dithering=0'` and `'map_palette=0'`.



Example 146: `image.jpg +index 1,1,1`



Example 147: `image.jpg (0;255;255^0;128;255^0;0;255) +index[-2] [-1],1,1`

Tutorial page:

https://gmic.eu/tutorial/_index.shtml

2.5.19 *inrange*

Arguments:

- `min[%],max[%]`

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



Example 148 : `image.jpg +inrange 25%, 75%`

2.5.20 *map* (+)

Arguments:

- `[palette], _boundary_conditions`
- `predefined_palette, _boundary_conditions`

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_conditions' can be { 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }.

Default value:

- `'boundary_conditions=0'`.



Example 149 : `image.jpg +luminance map[-1] 3`



Example 150 : `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]`

Tutorial page:

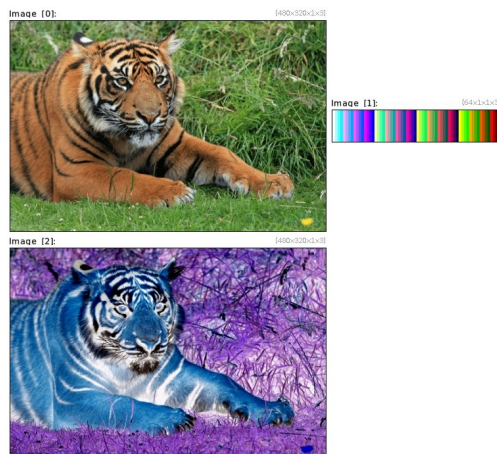
https://gmic.eu/tutorial/_map.shtml

2.5.21 *map_clut*

Arguments:

- `[clut]` | `"clut_name"`

Map specified RGB color LUT to selected images.



Example 151 : `image.jpg uniform_distribution {2^6},3 mirror[-1] x +map_clut[0] [1]`

2.5.22 *mix_channels*

Arguments:

- `(a00, ..., aMN)`
- `[matrix]`

Apply specified matrix to channels of selected images.



Example 152 : `image.jpg +mix_channels (0,1,0;1,0,0;0,0,1)`

2.5.23 *negate*

Arguments:

- `base_value`
- `(no arg)`

Negate image values.

Default value:

- `'base_value=(undefined)'`.



Example 153 : `image.jpg +negate`

2.5.24 *noise (+)*

Arguments:

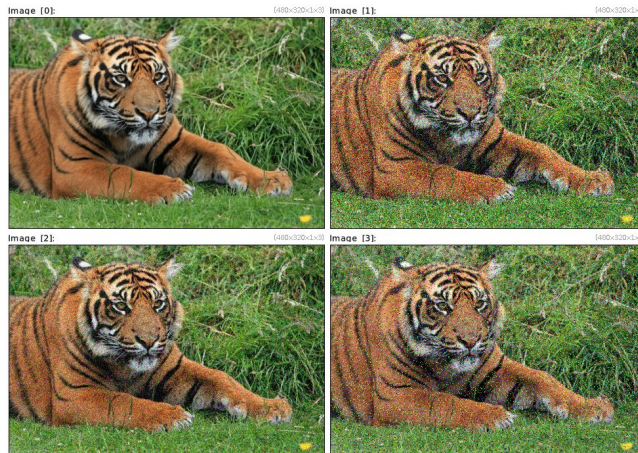
- `std_deviation>=0[%],noise_type`

Add random noise to selected images.

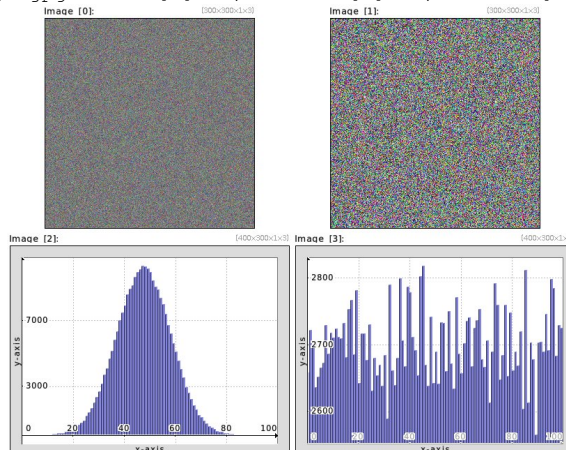
'noise_type' can be { 0=gaussian | 1=uniform | 2=salt&pepper | 3=poisson | 4=rice }.

Default value:

- 'noise_type=0'.



Example 154: `image.jpg +noise[0] 50,0 +noise[0] 50,1 +noise[0] 10,2 cut 0,255`



Example 155: `300,300,1,3 [0] noise[0] 20,0 noise[1] 20,1 +histogram 100 display-graph[-2,-1] 400,300,3`

2.5.25 *noise_perlin*

Arguments:

- `_scale_x[%]>0, _scale_y[%]>0, _scale_z[%]>0, _seed_x, _seed_y, _seed_z`

Render 2D or 3D Perlin noise on selected images, from specified coordinates.

The Perlin noise is a specific type of smooth noise, described here :
'https://en.wikipedia.org/wiki/Perlin_noise'.

Default values:

- 'scale_x=scale_y=scale_z=16' and 'seed_x=seed_y=seed_z=0'.

Tutorial page:

https://gmic.eu/tutorial/500,500,1,3noise_perlin,.shtml

2.5.26 *noise_poissondisk*

Arguments:

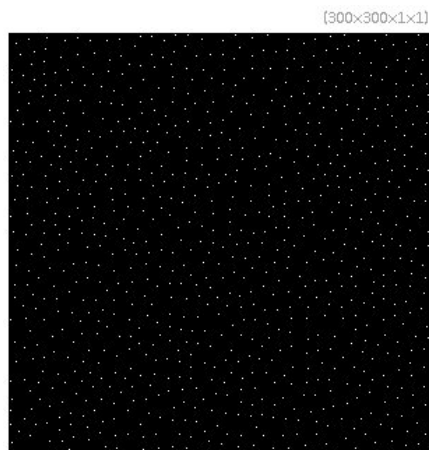
- `_radius[%]>0, _max_sample_attempts>0`

Add poisson disk sampling noise to selected images.

Implements the algorithm from the article "Fast Poisson Disk Sampling in Arbitrary Dimensions", by Robert Bridson (SIGGRAPH'2007).

Default values:

- 'radius=8' and 'max_sample_attempts=30'.



Example 156 : 300,300 noise_poissondisk 8

2.5.27 *normp*

Arguments:

- `p>=0`

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

Default value:

- 'p=2'.



Example 157 : `image.jpg +normp[0] 0 +normp[0] 1 +normp[0] 2 +normp[0] inf`

2.5.28 *norm*

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



Example 158 : `image.jpg +norm`

Tutorial page:

https://gmic.eu/tutorial/_norm.shtml

2.5.29 *normalize (+)*

Arguments:

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

Linearly normalize values of selected images in specified range.
(*eq. to 'n'*).



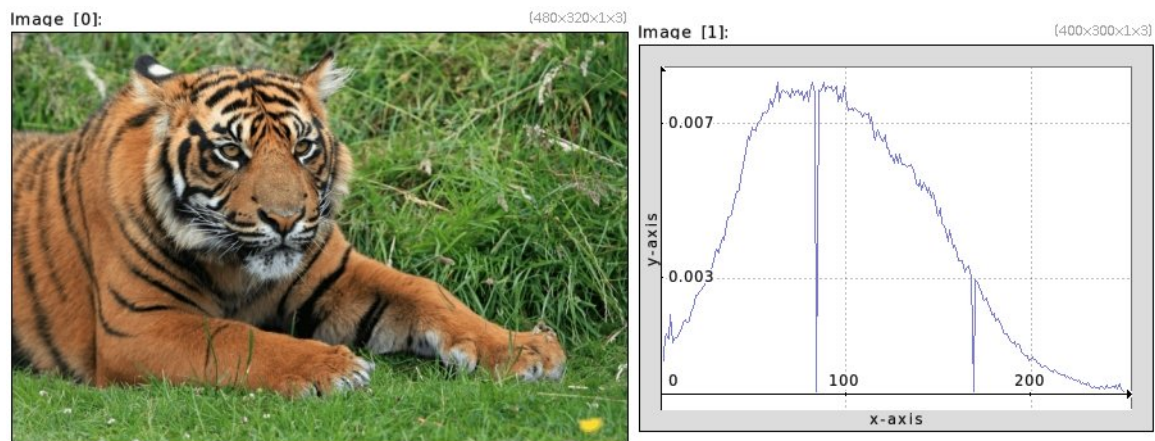
Example 159 : `image.jpg split x,2 normalize[-1] 64,196 append x`

Tutorial page:

https://gmics.eu/tutorial/_normalize.shtml

2.5.30 *normalize_sum*

Normalize selected images with a unitary sum.



Example 160 : `image.jpg +histogram normalize_sum[-1] display_graph[-1] 400,300`

2.5.31 *not*

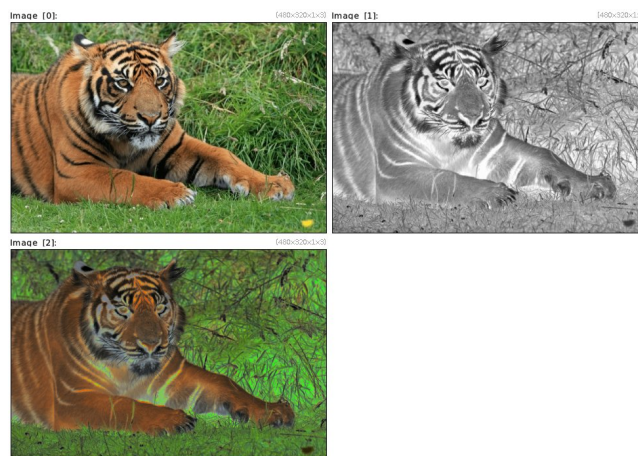
Apply boolean not operation on selected images.



Example 161 : `image.jpg +ge 50% +not [-1]`

2.5.32 *orientation*

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



Example 162 : `image.jpg +orientation +norm [-2] negate [-1] mul [-2] [-1] reverse [-2, -1]`

Tutorial page:

https://gmics.eu/tutorial/_orientation.shtml

2.5.33 *oneminus*

For each selected image, compute one minus image.



Example 163 : `image.jpg normalize 0,1 +oneminus`

2.5.34 *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 164 : `image.jpg luminance +otsu ,`

2.5.35 *polar2complex*

Compute polar to complex transforms of selected images.

2.5.36 *quantize*

Arguments:

- `nb_levels>=1, _keep_values={ 0 | 1 }, _is_uniform={ 0 | 1 }`

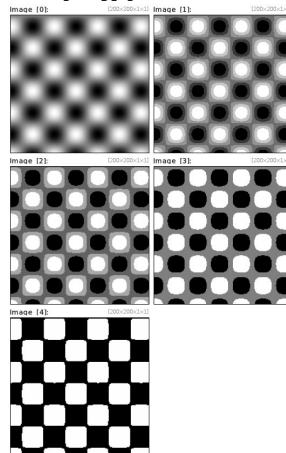
Quantize selected images.

Default value:

- `'keep_values=1'` and `'is_uniform=0'`.



Example 165 : `image.jpg luminance +quantize 3`



Example 166 : `200,200,1,1,'cos(x/10)*sin(y/10)'+quantize[0] 6 +quantize[0] 4 +quantize[0] 3 +quantize[0] 2`

2.5.37 *quantize_area*

Arguments:

- `_min_area>0`

Quantize selected images such that each flat region has an area greater or equal to `'min_area'`.

Default value:

- `'min_area=10'`.



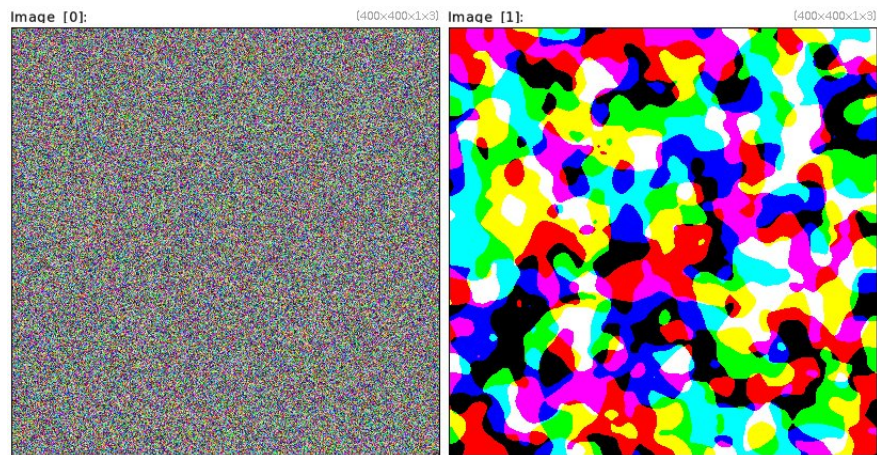
Example 167: `image.jpg quantize 3 +blur 1 round[-1] +quantize_area[-1] 2`

2.5.38 *rand* (+)

Arguments:

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

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



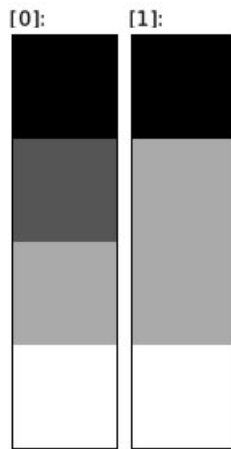
Example 168: `400,400,1,3 rand -10,10 +blur 10 sign[-1]`

2.5.39 *replace*

Arguments:

- `source,target`

Replace pixel values in selected images.



Example 169 : (1;2;3;4) +replace 2,3

2.5.40 *replace_inf*

Arguments:

- `_expression`

Replace all infinite values in selected images by specified expression.



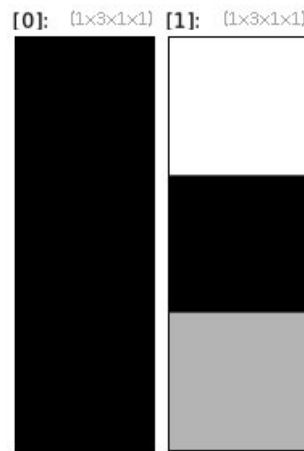
Example 170 : (0;1;2) log +replace_inf 2

2.5.41 *replace_nan*

Arguments:

- `_expression`

Replace all NaN values in selected images by specified expression.



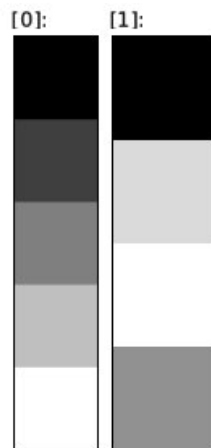
Example 171 : `(-1;0;2) sqrt +replace.nan 2`

2.5.42 *replace_seq*

Arguments:

- `"search_seq", "replace_seq"`

Search and replace a sequence of values in selected images.



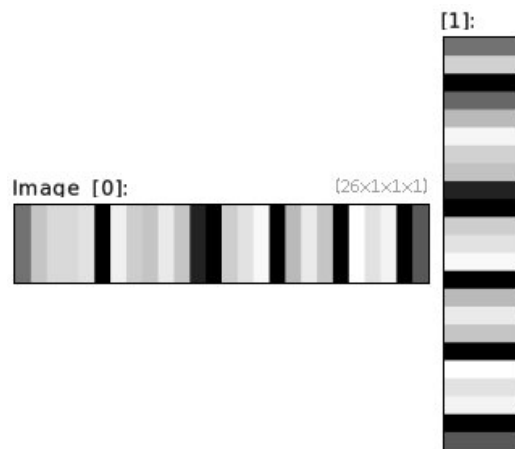
Example 172 : `(1;2;3;4;5) +replace_seq "2,3,4", "7,8"`

2.5.43 *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 173: (`{'Hello there, how are you ?'}`) `+replace_str "Hello there", "Hi David"`

2.5.44 *round* (+)

Arguments:

- `rounding_value >= 0, _rounding_type`
- (no arg)

Round values of selected images.

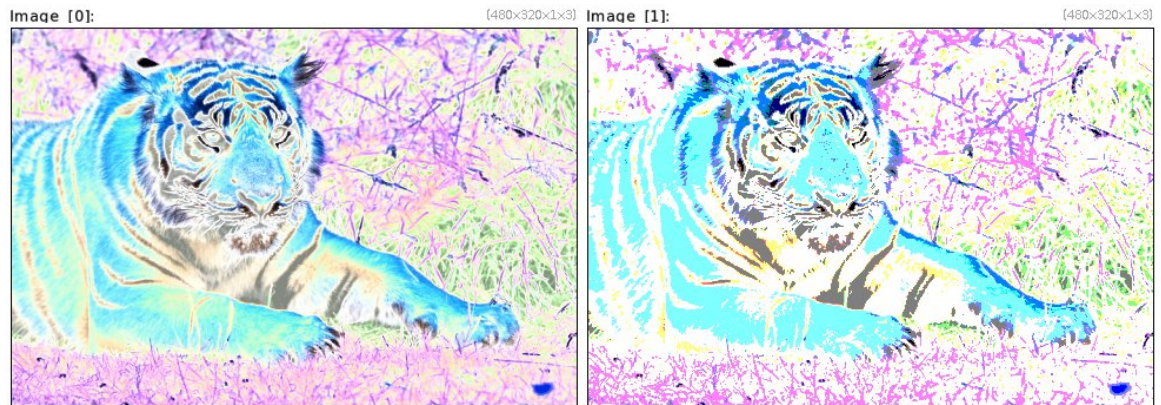
'rounding_type' can be { -1=backward | 0=nearest | 1=forward }.

Default value:

- `'rounding_type=0'`.



Example 174: `image.jpg +round 100`



Example 175 : `image.jpg mul {pi/180} sin +round`

2.5.45 *roundify*

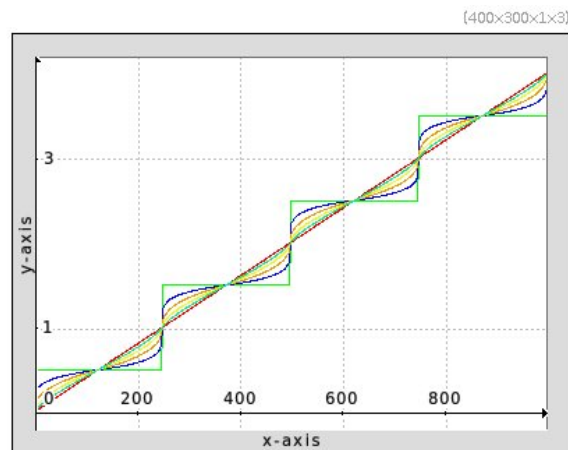
Arguments:

- `gamma>=0`

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

Default value:

- `'gamma=0'` .



Example 176 : `1000 fill '4*x/w' repeat 5 +roundify[0] {$>*0.2} done append c display_graph 400,300`

2.5.46 *set (+)*

Arguments:

- `value, -x[%], -y[%], -z[%], -c[%]`

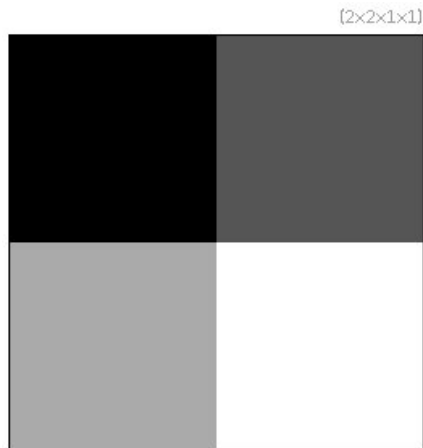
Set pixel value in selected images, at specified coordinates.

(eq. to ' = ') . \n).

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

Default values:

- ' x=y=z=c=0 ' .



Example 177 : 2,2 set 1,0,0 set 2,1,0 set 3,0,1 set 4,1,1



Example 178 : image.jpg repeat 10000 set 255,{u(100)}%,{u(100)}%,0,{u(100)}% done

2.5.47 *set_vector_covariance*

Arguments:

- coef1,coef2,...,coefN

Apply linear transformation on selected images so that the covariance matrix of their vector-valued pixels is prescribed to given matrix. Matrix size 'N' must be equal to 'spectrum^2'.

2.5.48 *threshold (+)*

Arguments:

- `value[%],_is-soft={ 0 | 1 }`
- `(no arg)`

Threshold values of selected images.

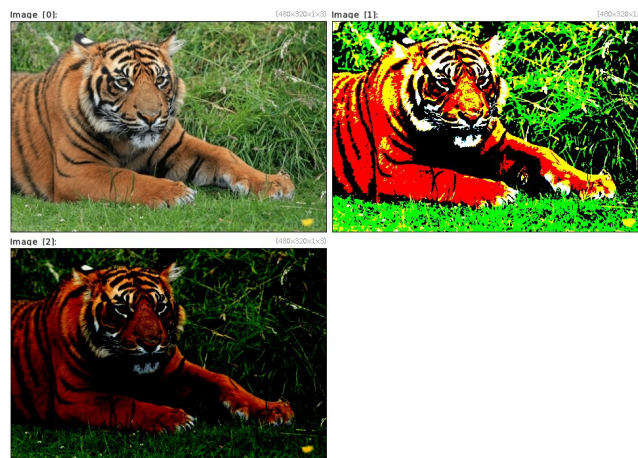
'soft' can be { 0=hard-thresholding | 1=soft-thresholding }.

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

In interactive mode, the chosen threshold value is returned in the status.

Default value:

- `'is-soft=0'`.



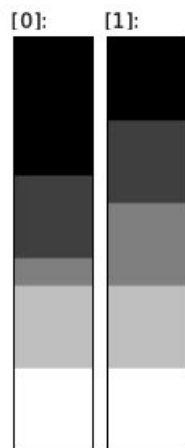
Example 179 : `image.jpg +threshold[0] 50% +threshold[0] 50%,1`

Tutorial page:

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

2.5.49 *unrepeat*

Remove repetition of adjacent values in selected images.



Example 180 : `(1;1;1;1;1;2;2;2;3;4;4;4;5;5;5) +unrepeat`

2.5.50 *vector2tensor*

Convert selected vector fields to corresponding tensor fields.

2.6 Colors Manipulation**2.6.1 *adjust_colors*****Arguments:**

- `-100<=_brightness<=100, -100<=_contrast<=100, -100<=_gamma<=100, -100<=_hue_shift<=100, -100<=_saturation<=100, _value_min, _value_max`

Perform a global adjustment of colors on selected images.

Range of correct image values are considered to be in `[value_min, value_max]` (e.g. `[0,255]`).

If `'value_min==value_max==0'`, value range is estimated from min/max values of selected images.

Processed images have pixel values constrained in `[value_min, value_max]`.

Default values:

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



Example 181 : `image.jpg +adjust_colors 0,30,0,0,30`

2.6.2 *apply_channels***Arguments:**

- `"command", color_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'`) `.\n`).

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

You can also make the processing focus on a few particular channels of this colorspace, by setting `'color_channels'` as `'colorspace_channel'` (e.g. `'hsv.h'` for the hue).

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

Default value:

- `'value_action=0'`.



Example 182 : `image.jpg +apply_channels "equalize blur 2",ycbcr-cbcr`

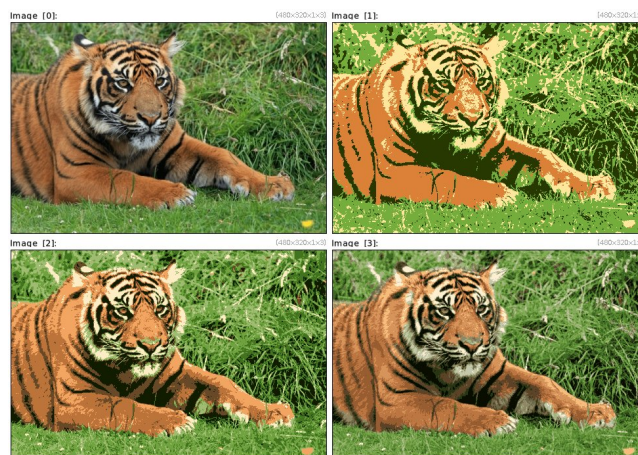
2.6.3 *autoindex***Arguments:**

- `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=1'`.



Example 183 : `image.jpg +autoindex[0] 4 +autoindex[0] 8 +autoindex[0] 16`

2.6.4 *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 184 : `image.jpg rgb2bayer 0 +bayer2rgb 1,1,0.5`

2.6.5 *cmy2rgb*

Convert color representation of selected images from CMY to RGB.

2.6.6 *cmyk2rgb*

Convert color representation of selected images from CMYK to RGB.

2.6.7 *colorblind*

Arguments:

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

Simulate color blindness vision.



Example 185 : `image.jpg +colorblind 0`

2.6.8 *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=1' and 'sort_vectors=1'.



Example 186 : `image.jpg +colormap[0] 4 +colormap[0] 8 +colormap[0] 16`

Tutorial page:

https://gmics.eu/tutorial/_colormap.shtml

2.6.9 *compose_channels*

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

Default value:

- '1=+'.



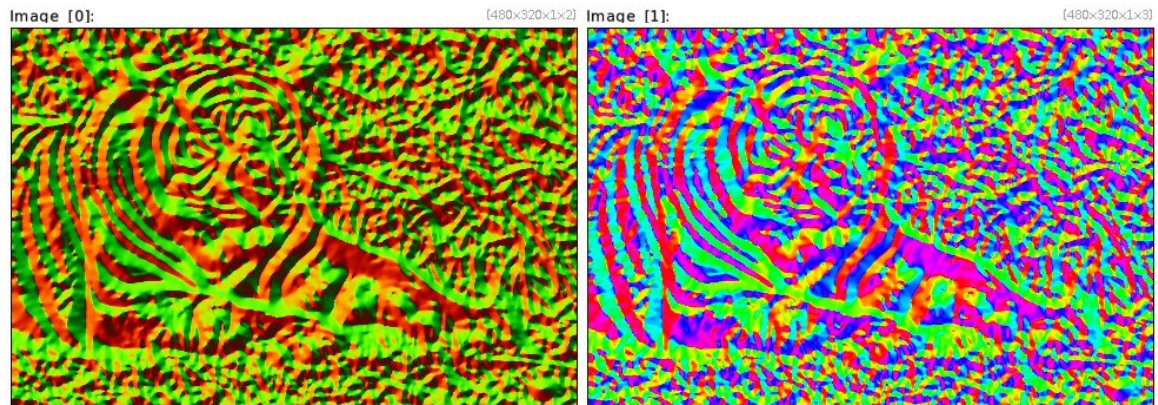
Example 187 : `image.jpg +compose-channels` and

Tutorial page:

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

2.6.10 *direction2rgb*

Compute RGB representation of selected 2D direction fields.



Example 188 : `image.jpg luminance gradient append c blur 2 orientation +direction2rgb`

2.6.11 *ditheredbw*

Create dithered B&W version of selected images.



Example 189 : `image.jpg +equalize ditheredbw[-1]`

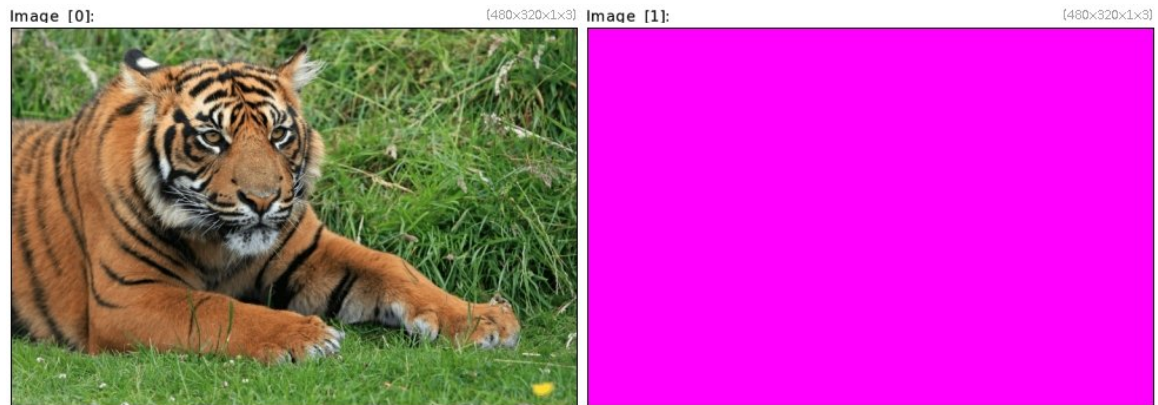
2.6.12 *fill_color*

Arguments:

- `col1, ..., colN`

Fill selected images with specified color.

(eq. to 'fc').



Example 190 : `image.jpg +fill_color 255,0,255`

Tutorial page:

https://gmic.eu/tutorial/_fill_color.shtml

2.6.13 *gradient2rgb*

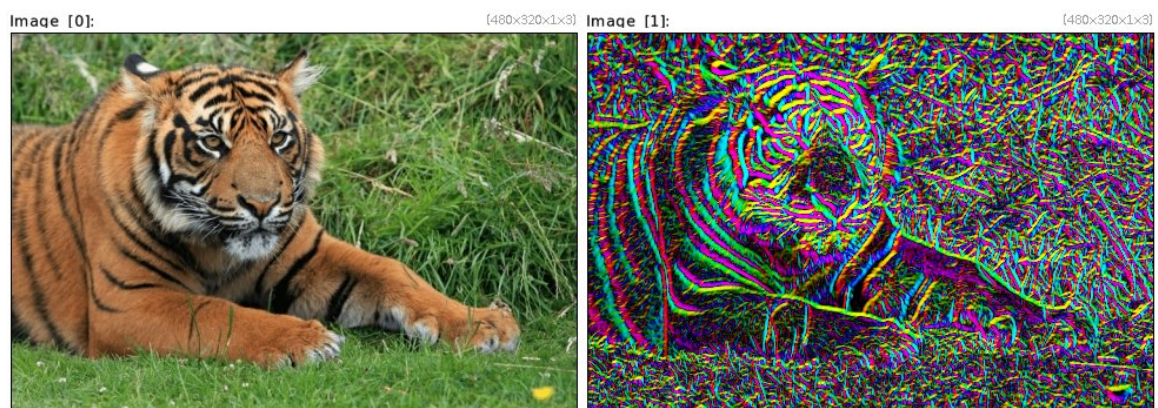
Arguments:

- `_is_orientation={ 0 | 1 }`

Compute RGB representation of 2D gradient of selected images.

Default value:

- `'is_orientation=0'`.



Example 191 : `image.jpg +gradient2rgb 0 equalize[-1]`

2.6.14 *hcy2rgb* :

Convert color representation of selected images from HCY to RGB.

2.6.15 *hsi2rgb*

Convert color representation of selected images from HSI to RGB.

2.6.16 *hsi2rgb*

Convert color representation of selected images from HSI8 to RGB.

2.6.17 *hsl2rgb*

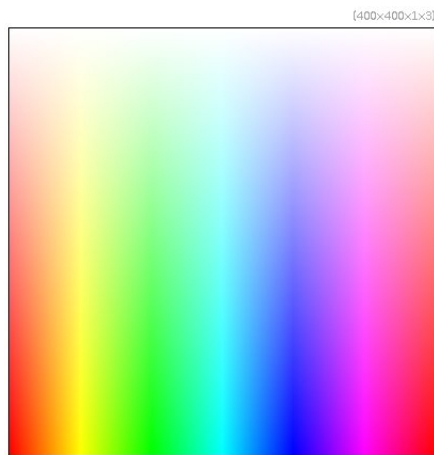
Convert color representation of selected images from HSL to RGB.

2.6.18 *hsl82rgb*

Convert color representation of selected images from HSL8 to RGB.

2.6.19 *hsv2rgb*

Convert color representation of selected images from HSV to RGB.



Example 192 : `(0,360;0,360^0,0;1,1^1,1;1,1) resize 400,400,1,3,3 hsv2rgb`

2.6.20 *hsv82rgb*

Convert color representation of selected images from HSV8 to RGB.

2.6.21 *int2rgb*

Convert color representation of selected images from INT24 to RGB.

2.6.22 *lab2lch*

Convert color representation of selected images from Lab to Lch.

2.6.23 *lab2rgb***Arguments:**

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from Lab to RGB.

Default value:

- `'illuminant=1'`.



Example 193 : `(50,50;50,50^-3,3;-3,3^-3,-3;3,3) resize 400,400,1,3,3 lab2rgb`

2.6.24 *lab2srgb*

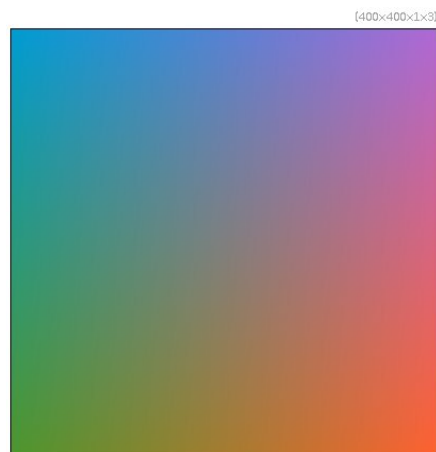
Arguments:

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from Lab to sRGB.

Default value:

- `'illuminant=1'`.



Example 194 : `(50,50;50,50^-3,3;-3,3^-3,-3;3,3) resize 400,400,1,3,3 lab2rgb`

2.6.25 *lab82srgb*

Arguments:

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from Lab8 to sRGB.

Default value:

- `'illuminant=1'`.



Example 195 : `(50,50;50,50^-3,3;-3,3^-3,-3;3,3) resize 400,400,1,3,3 lab2rgb`

2.6.26 *lab2xyz*

Arguments:

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from Lab to XYZ.

Default value:

- `'illuminant=1'`.

2.6.27 *lab82rgb*

Arguments:

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from Lab8 to RGB.

Default value:

- `'illuminant=1'`.

2.6.28 *lch2lab*

Convert color representation of selected images from Lch to Lab.

2.6.29 *lch2rgb*

Arguments:

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from Lch to RGB.

Default value:

- `'illuminant=1'`.

2.6.30 *lch82rgb*

Arguments:

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from Lch8 to RGB.

Default value:

- `'illuminant=1'`.

2.6.31 *luminance*

Compute luminance of selected sRGB images.



Example 196 : `image.jpg +luminance`

2.6.32 *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 197 : `image.jpg +mix_rgb 0,1,0,1,0,0,0,0,1`

Tutorial page:

https://gmic.eu/tutorial/_mix_rgb.shtml

2.6.33 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 198 : `pseudogray 5`

2.6.34 *replace_color*

Arguments:

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

Replace pixels from/to specified colors in selected images.



Example 199 : `image.jpg +replace_color 40,3,204,153,110,255,0,0`

2.6.35 *retinex*

Arguments:

- `_value_offset>0, _colorspace={ hsi | hsv | lab | lrgb | rgb | ycbcr }, 0<=_min_cut<=100, 0<=_max_cut<=100, _sigma_low>0, _sigma_mid>0, _sigma_high>0`

Apply multi-scale retinex algorithm on selected images to improve color consistency. (as described in the page <http://www.ipol.im/pub/art/2014/107/>).

Default values:

- `'offset=1', 'colorspace=hsv', 'min_cut=1', 'max_cut=1', 'sigma_low=15', 'sigma_mid=80' and 'sigma_high=250'.`

2.6.36 *rgb2bayer*

Arguments:

- `_start_pattern=0, _color_grid=0`

Transform selected color images to RGB-Bayer sampled images.

Default values:

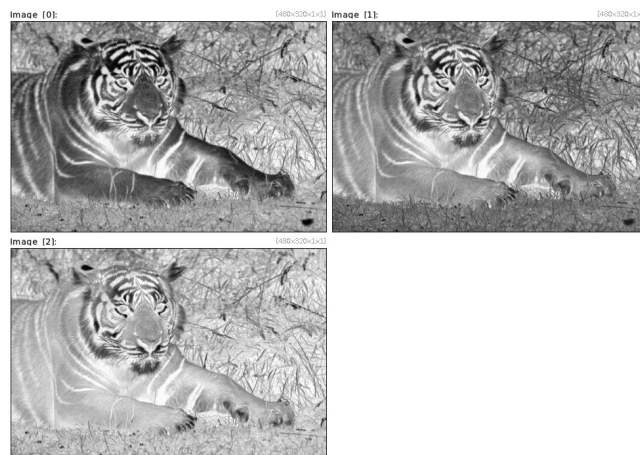
- `'start_pattern=0' and 'color_grid=0'.`



Example 200 : `image.jpg +rgb2bayer 0`

2.6.37 *rgb2cmy*

Convert color representation of selected images from RGB to CMY.



Example 201 : `image.jpg rgb2cmy split c`

2.6.38 *rgb2cmyk*

Convert color representation of selected images from RGB to CMYK.



Example 202 : `image.jpg rgb2cmyk split c`

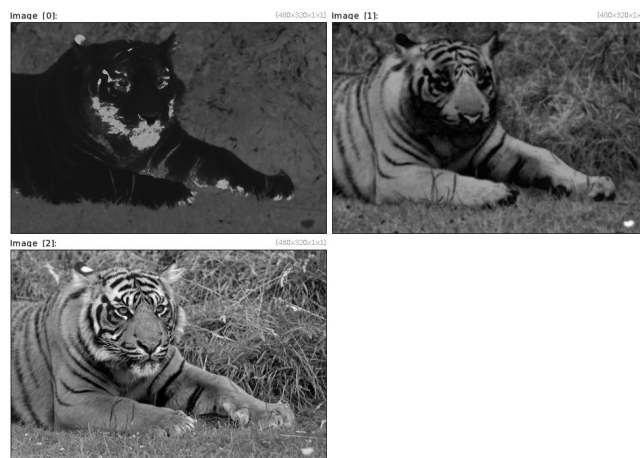
(480x320x1x3)



Example 203 : `image.jpg rgb2cmyk split c fill[3] 0 append c cmyk2rgb`

2.6.39 *rgb2hcy*

Convert color representation of selected images from RGB to HCY.



Example 204 : `image.jpg rgb2hcy split c`

2.6.40 *rgb2hsi*

Convert color representation of selected images from RGB to HSI.



Example 205 : `image.jpg rgb2hsi split c`

2.6.41 *rgb2hsi8*

Convert color representation of selected images from RGB to HSI8.



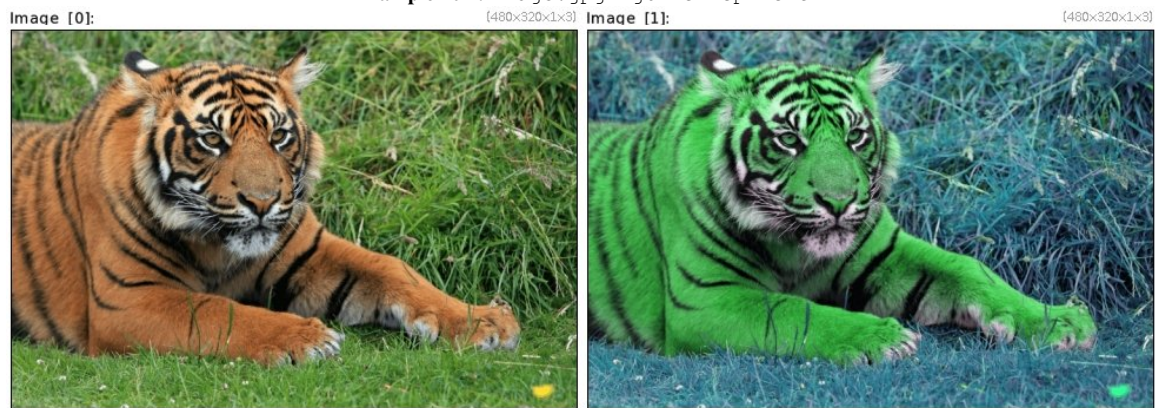
Example 206 : `image.jpg rgb2hsi8 split c`

2.6.42 *rgb2hsl*

Convert color representation of selected images from RGB to HSL.



Example 207 : `image.jpg rgb2hsl split c`



Example 208 : `image.jpg rgb2hsl +split c add[-3] 100 mod[-3] 360 append[-3--1] c hsl2rgb`

2.6.43 *rgb2hsl8*

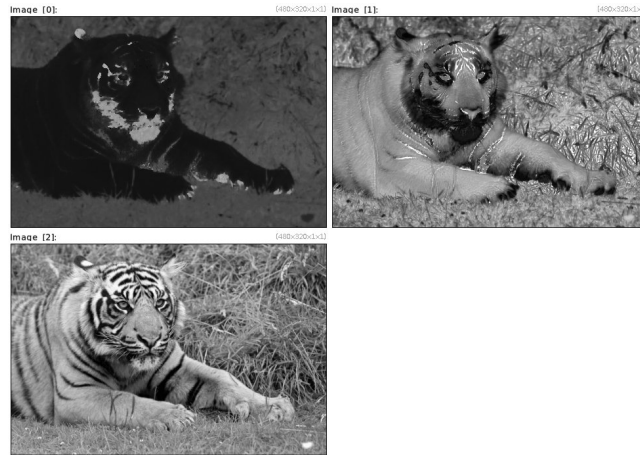
Convert color representation of selected images from RGB to HSL8.



Example 209 : `image.jpg rgb2hsl8 split c`

2.6.44 *rgb2hsv*

Convert color representation of selected images from RGB to HSV.



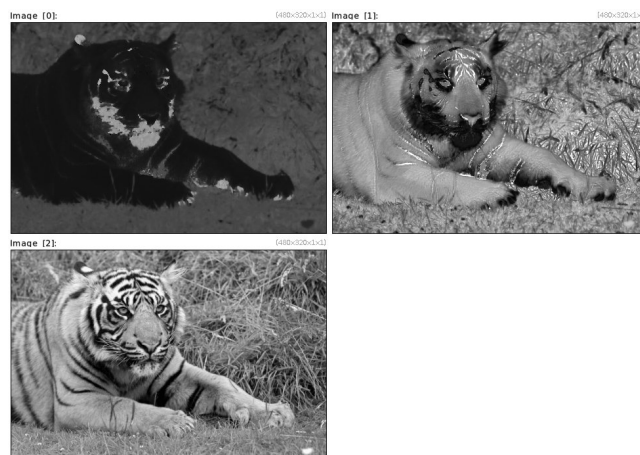
Example 210 : `image.jpg rgb2hsv split c`



Example 211 : `image.jpg rgb2hsv +split c add[-2] 0.3 cut[-2] 0,1 append[-3--1] c hsv2rgb`

2.6.45 *rgb2hsv8*

Convert color representation of selected images from RGB to HSV8.



Example 212 : `image.jpg rgb2hsv8 split c`

2.6.46 *rgb2lab*

Arguments:

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from RGB to Lab.

Default value:

- `'illuminant=1'`.

2.6.47 *rgb2lab8*

Arguments:

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from RGB to Lab8.

Default value:

- `'illuminant=1'`.



Example 213 : `image.jpg rgb2lab8 split c`

2.6.48 *rgb2lch*

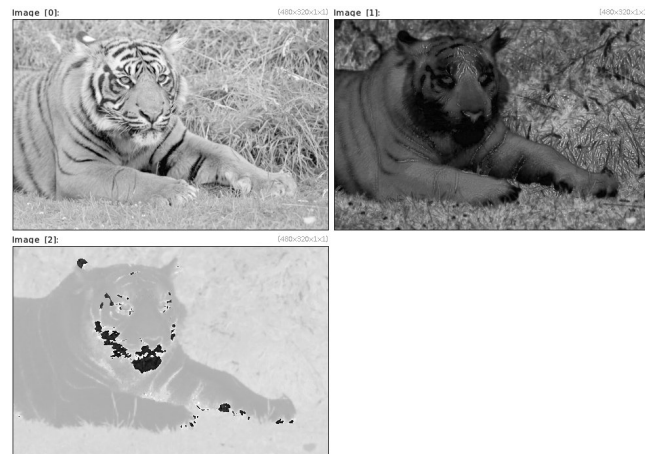
Arguments:

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from RGB to Lch.

Default value:

- `'illuminant=1'`.



Example 214 : `image.jpg rgb2lch split c`

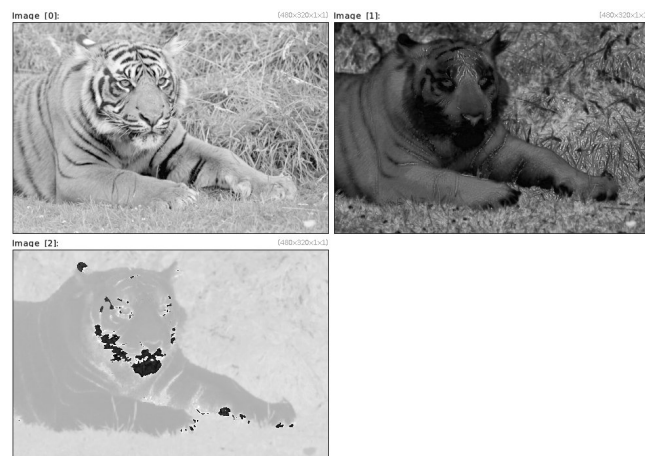
2.6.49 *rgb2lch8***Arguments:**

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from RGB to Lch8.

Default value:

- `'illuminant=1'`.



Example 215 : `image.jpg rgb2lch8 split c`

2.6.50 *rgb2luv*

Convert color representation of selected images from RGB to LUV.



Example 216 : `image.jpg rgb2luv split c`

2.6.51 *rgb2int*

Convert color representation of selected images from RGB to INT24 scalars.



Example 217 : `image.jpg rgb2int`

2.6.52 *rgb2srgb*

Convert color representation of selected images from linear RGB to sRGB.

2.6.53 *rgb2xyz*

Arguments:

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from RGB to XYZ.

Default value:

- `'illuminant=1'`.



Example 218 : `image.jpg rgb2xyz split c`

2.6.54 *rgb2xyz8***Arguments:**

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from RGB to XYZ8.

Default value:

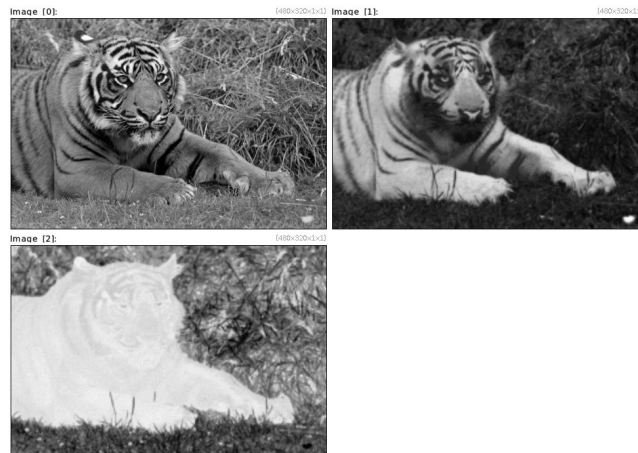
- `'illuminant=1'`.



Example 219 : `image.jpg rgb2xyz8 split c`

2.6.55 *rgb2yiq*

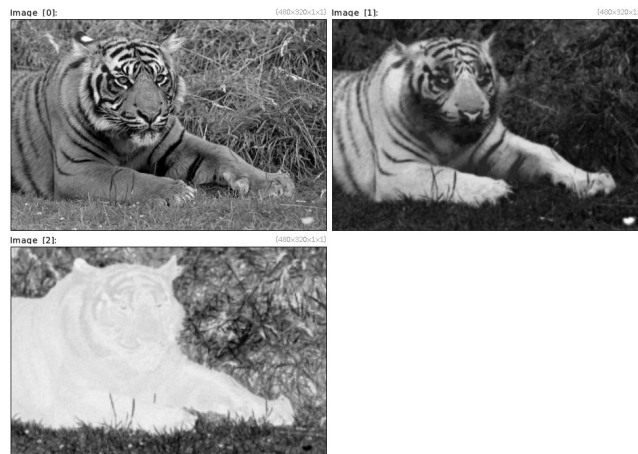
Convert color representation of selected images from RGB to YIQ.



Example 220 : `image.jpg rgb2yiq split c`

2.6.56 *rgb2yiq8*

Convert color representation of selected images from RGB to YIQ8.



Example 221 : `image.jpg rgb2yiq8 split c`

2.6.57 *rgb2ycbcr*

Convert color representation of selected images from RGB to YCbCr.



Example 222 : `image.jpg rgb2ycbcr split c`

2.6.58 *rgb2yuv*

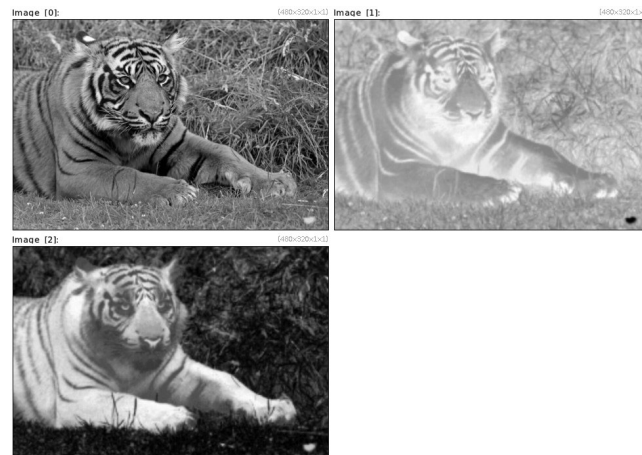
Convert color representation of selected images from RGB to YUV.



Example 223 : `image.jpg rgb2yuv split c`

2.6.59 *rgb2yuv8*

Convert color representation of selected images from RGB to YUV8.



Example 224 : `image.jpg rgb2yuv8 split c`

2.6.60 *remove_opacity*

Remove opacity channel of selected images.

2.6.61 *select_color*

Arguments:

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

Select pixels with specified color in selected images.



Example 225 : `image.jpg +select_color 40,204,153,110`

Tutorial page:

https://gmics.eu/tutorial/_select_color.shtml

2.6.62 *sepia*

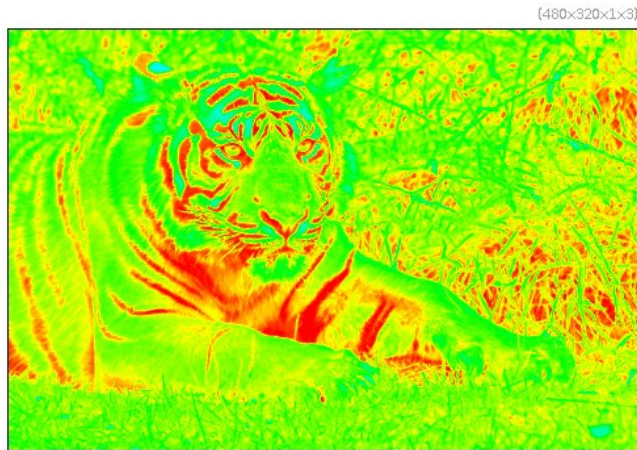
Apply sepia tones effect on selected images.



Example 226 : `image.jpg sepia`

2.6.63 *solarize*

Solarize selected images.



Example 227 : `image.jpg solarize`

2.6.64 *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 228 : `image.jpg quantize 5 +split.colors , display.rgb`

2.6.65 *split_opacity*

Split color and opacity parts of selected images.

2.6.66 *srgb2lab*

Arguments:

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from sRGB to Lab.

Default value:

- `'illuminant=1'`.



Example 229 : `image.jpg srgb2lab split c`



Example 230 : `image.jpg srgb2lab +split c mul[-2,-1] 2.5 append[-3--1] c lab2srgb`

2.6.67 *srgb2lab8*

Arguments:

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from sRGB to Lab8.

Default value:

- `'illuminant=1'.`

2.6.68 *srgb2rgb*

Convert color representation of selected images from sRGB to linear RGB.

2.6.69 *to_a*

Force selected images to have an alpha channel.

2.6.70 *to_color*

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

2.6.71 *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'.`

2.6.72 *to_gray*

Force selected images to be in GRAY mode.



Example 231 : `image.jpg +to_gray`

2.6.73 *to_graya*

Force selected images to be in GRAYA mode.

2.6.74 *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 values:

- `'max_step=5', 'is_perceptual_constraint=1' and 'bits_depth=8'.`

The original pseudo-gray technique has been introduced by Rich Franzen [<http://r0k.us/graphics/pseudoGrey.html>].

Extension of this technique to arbitrary increments for more tones, has been done by David Tschumperle.

2.6.75 *to_rgb*

Force selected images to be in RGB mode.

2.6.76 *to_rgba*

Force selected images to be in RGBA mode.

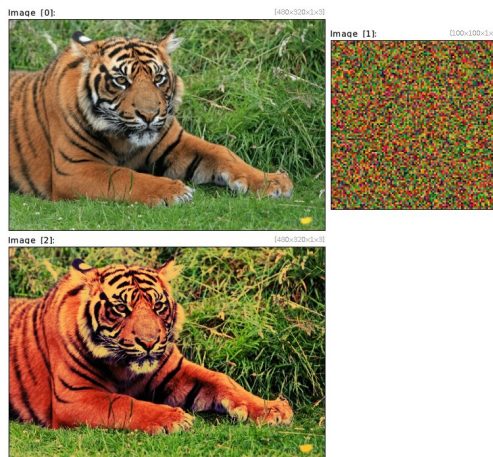
2.6.77 *transfer_histogram***Arguments:**

- `[reference_image], _nb_levels>0, _color_channels`

Transfer histogram of the specified reference image to selected images.
Argument 'color channels' is the same as with command 'apply_channels'.

Default value:

- 'nb_levels=256' and 'color_channels=all'.



Example 232 : `image.jpg 100,100,1,3,"u([256,200,100])" +transfer_histogram[0] [1]`

2.6.78 *transfer_rgb*

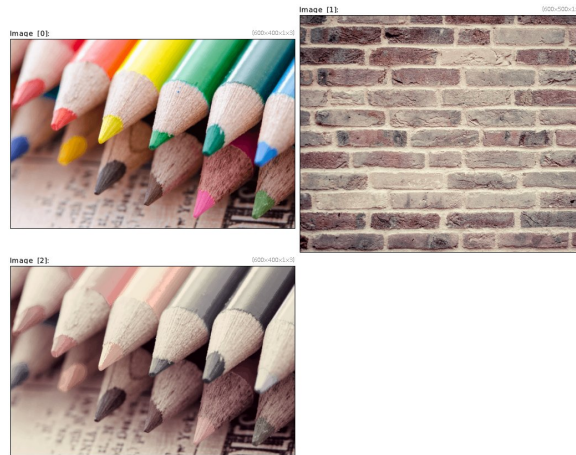
Arguments:

- `[target],_gamma>=0,_regularization>=0,_luminosity_constraints>=0,_rgb_resolution>=0,_is_constraints={ 0 | 1 }`

Transfer colors from selected source images to selected reference image (given as argument).
'gamma' determines the importance of color occurrences in the matching process (0=none to 1=huge).
'regularization' determines the number of guided filter iterations to remove quantization effects.
'luminosity_constraints' tells if luminosity constraints must be applied on non-confident matched colors.
'is_constraints' tells if additional hard color constraints must be set (opens an interactive window).

Default values:

- 'gamma=0.3', 'regularization=8', 'luminosity_constraints=0.1', 'rgb_resolution=64' and 'is_constraints=0'.



Example 233: `sample pencils,wall +transfer.rgb[0] [1],0,0.01`

2.6.79 *xyz2lab*

Arguments:

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from XYZ to Lab.

Default value:

- `'illuminant=1'.`

2.6.80 *xyz2rgb*

Arguments:

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from XYZ to RGB.

Default value:

- `'illuminant=1'.`

2.6.81 *xyz82rgb*

Arguments:

- `illuminant={ 0=D50 | 1=D65 }`
- `(no arg)`

Convert color representation of selected images from XYZ8 to RGB.

Default value:

- `'illuminant=1'.`

2.6.82 *ycbcr2rgb*

Convert color representation of selected images from YCbCr to RGB.

2.6.83 *yiqr2rgb*

Convert color representation of selected images from YIQ to RGB.

2.6.84 *yiqr82rgb*

Convert color representation of selected images from YIQ8 to RGB.

2.6.85 *yuv2rgb*

Convert color representation of selected images from YUV to RGB.

2.6.86 *yuv82rgb*

Convert selected images from YUV8 to RGB color bases.

2.7 Geometry Manipulation**2.7.1 *append* (+)****Arguments:**

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

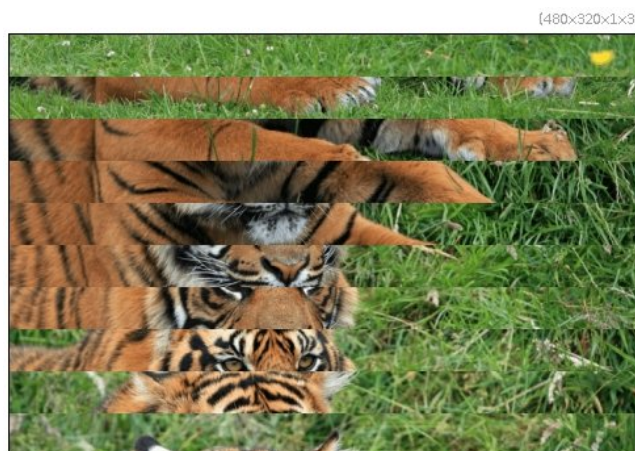
Append specified image to selected images, or all selected images together, along specified axis.
(*eq. to 'a') .\n*).

'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 234 : `image.jpg split y,10 reverse append y`



Example 235 : `image.jpg repeat 5 +rows[0] 0,{10+18*$>}% done remove[0] append x,0.5`



Example 236 : `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 237 : `image.jpg split xy,4 append_tiles ,`

2.7.3 *apply_scales*

Arguments:

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

Apply specified command on different scales of selected images.

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

Default value:

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



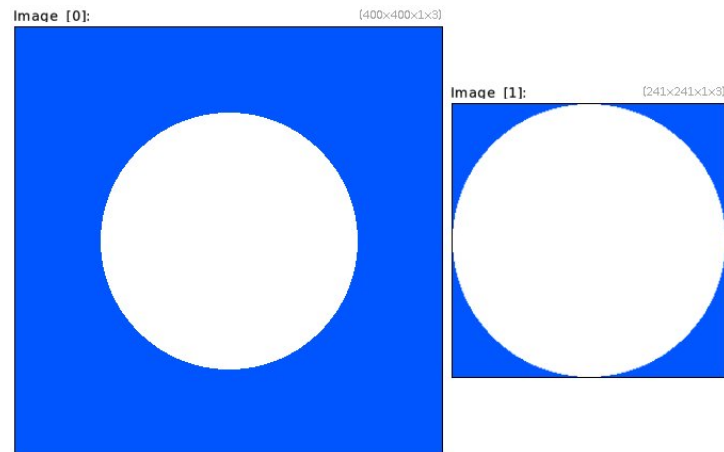
Example 238 : `image.jpg apply_scales "blur 5 sharpen 1000",4`

2.7.4 *autocrop (+)*

Arguments:

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

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



Example 239 : `400,400,1,3 fill-color 64,128,255 ellipse 50%,50%,120,120,0,1,255 +autocrop`

2.7.5 *autocrop_components*

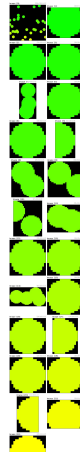
Arguments:

- `_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 240 : `256,256 noise 0.1,2 eq 1 dilate_circ 20 label_fg 0,1 normalize 0,255 +neq 0
*[-1] 255 append c +autocrop_components ,`

2.7.6 *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 241 : `image.jpg +fill[-1] 0 ellipse[-1] 50%,50%,30%,20%,0,1,1 autocrop_seq 0`

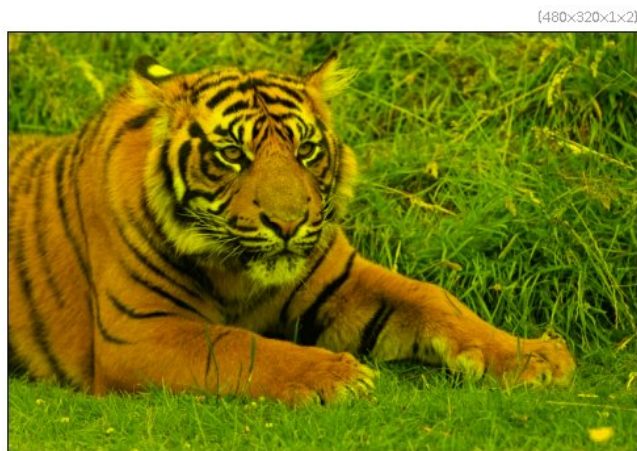
2.7.7 *channels (+)*

Arguments:

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

Keep only specified channels of selected images.

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



Example 242 : `image.jpg channels 0,1`

[480x320x1x3]

**Example 243 :** `image.jpg luminance channels 0,2`

2.7.8 *columns* (+)

Arguments:

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

Keep only specified columns of selected images.

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

[361x320x1x3]

**Example 244 :** `image.jpg columns -25%,50%`

2.7.9 *crop* (+)

Arguments:

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

- (no arg)

Crop selected images with specified region coordinates.

(eq. to 'z') .\n).

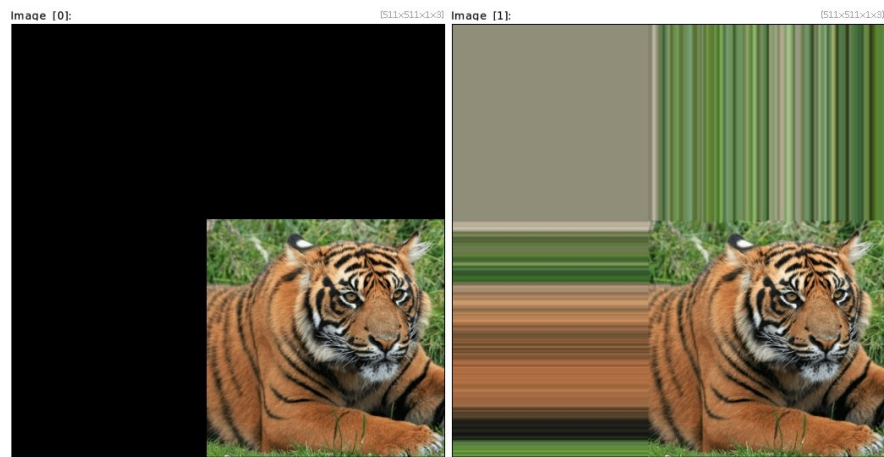
'boundary_conditions' can be { 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }.

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

In interactive mode, the chosen crop coordinates are returned in the status.

Default value:

- 'boundary_conditions=0'.



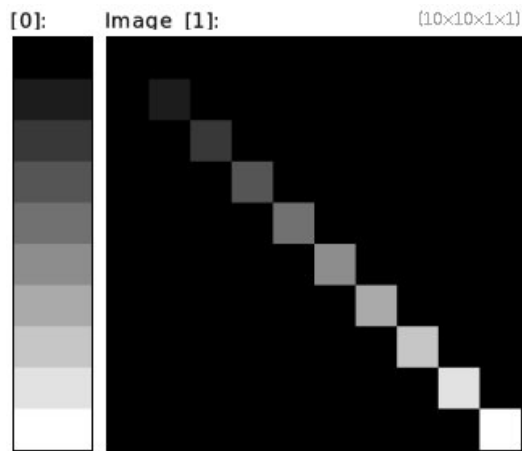
Example 245 : image.jpg +crop -230,-230,280,280,1 crop[0] -230,-230,280,280,0



Example 246 : image.jpg crop 25%,25%,75%,75%

2.7.10 diagonal

Transform selected vectors as diagonal matrices.



Example 247 : `1,10,1,1,'y' +diagonal`

2.7.11 *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'.`

2.7.12 *expand_x*

Arguments:

- `size_x>=0, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic
| 3=mirror }`

Expand selected images along the x-axis.

Default value:

- `'boundary_conditions=1'.`



Example 248 : `image.jpg expand_x 30,0`

2.7.13 *expand_xy*

Arguments:

- `size>=0, boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

Expand selected images along the xy-axes.

Default value:

- `'boundary_conditions=1'`.



Example 249 : `image.jpg expand_xy 30,0`

2.7.14 *expand_xyz*

Arguments:

- `size>=0, boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

Expand selected images along the xyz-axes.

Default value:

- `'boundary_conditions=1'`.

2.7.15 *expand_y*

Arguments:

- `size_y>=0, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

Expand selected images along the y-axis.

Default value:

- `'boundary_conditions=1'`.



Example 250 : `image.jpg expand_y 30,0`

2.7.16 *expand_z*

Arguments:

- `size_z>=0, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

Expand selected images along the z-axis.

Default value:

- `'boundary_conditions=1'`.

2.7.17 *extract_region*

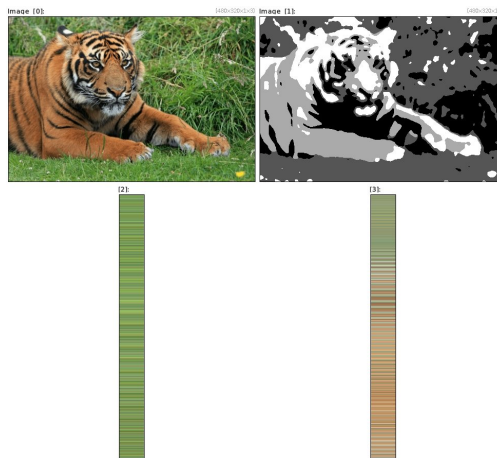
Arguments:

- `[label_image],_extract_xyz_coordinates={ 0 | 1 },_label_1,...,_label_M`

Extract all pixels of selected images whose corresponding label in '[label_image]' is equal to 'label_m', and output them as M column images.

Default value:

- `'extract_xyz_coordinates=0'`.



Example 251 : `image.jpg +blur 3 quantize. 4,0 +extract_region[0] [1],0,1,3`

2.7.18 *montage*

Arguments:

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

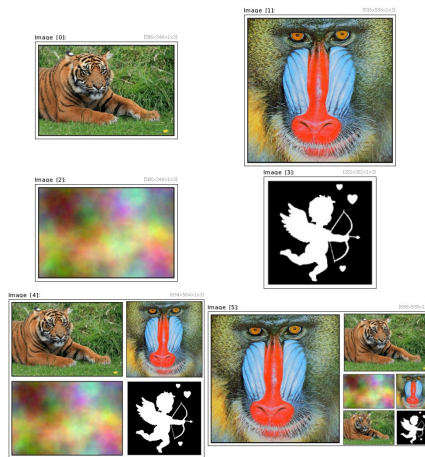
Create a single image montage from selected images, according to specified layout code : - 'X' to assemble all images using an automatically estimated layout. - 'H' to assemble all images horizontally. - 'V' to assemble all images vertically. - 'A' to assemble all images as an horizontal array. - 'B' 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 index (treated periodically) or a nested layout expression 'Hb:c','Vb:c','Rb' or 'Mb' itself.

For example, layout code 'H0:V1:2' creates an image where image [0] is on the left, and images [1] and [2] vertically packed on the right.

Default values:

- `'layout_code=X', 'montage_mode=2', output_mode='0' and 'processing_command=""'`.



Example 252: `image.jpg sample ? +plasma[0] shape_cupid 256 normalize 0,255 frame 3,3,0 frame 10,10,255 to_rgb +montage A +montage[^-1] H1:V0:VH2:1H0:3`

2.7.19 *mirror* (+)

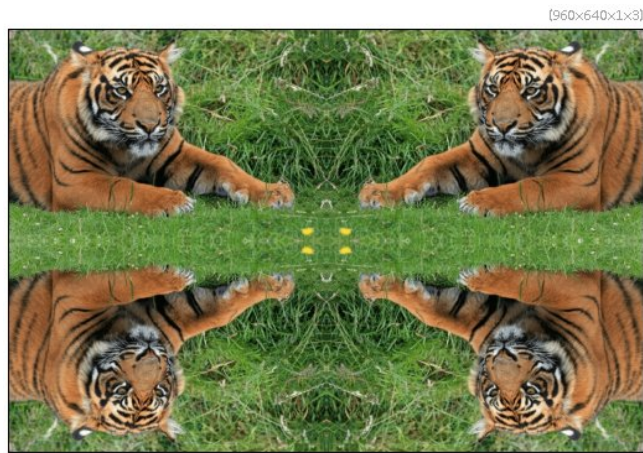
Arguments:

- { x | y | z }...{ x | y | z }

Mirror selected images along specified axes.



Example 253: `image.jpg +mirror y +mirror[0] c`



Example 254 : `image.jpg +mirror x +mirror y append_tiles 2,2`

2.7.20 *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. 'xyz', 'cxyz', ...



Example 255 : `image.jpg permute yxzc`

2.7.21 *resize* (+)

Arguments:

- `[image], _interpolation, _boundary_conditions, _ax, _ay, _az, _ac`
- `{[image.w] | width>0[%]}, -{[image.h] | height>0[%]}, -{[image.d]`
`| depth>0[%]}, -{[image.s]`
`| spectrum>0[%]}, _interpolation, _boundary_conditions, _ax, _ay, _az, _ac`
- (no arg)

Resize selected images with specified geometry.

(eq. to 'r') . \n).

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

'boundary_conditions' has different meanings, according to the chosen 'interpolation' mode : . When 'interpolation=={ -1 | 1 | 2 | 4 }', 'boundary_conditions' is meaningless. . When 'interpolation==0', 'boundary_conditions' can be { 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }. . When 'interpolation=={ 3 | 5 | 6 }', 'boundary_conditions' can be { 0=none | 1=neumann }.

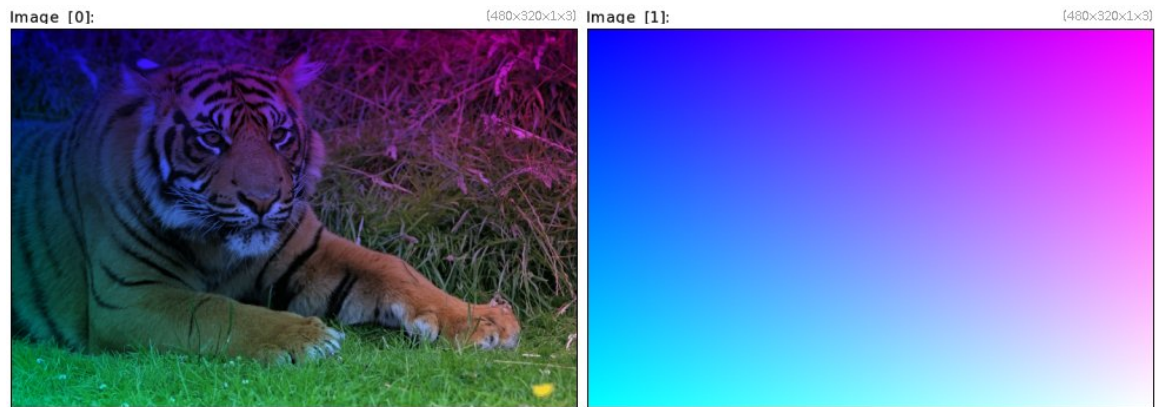
'ax,ay,az,ac' set the centering along each axis when 'interpolation=0 or 4'

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

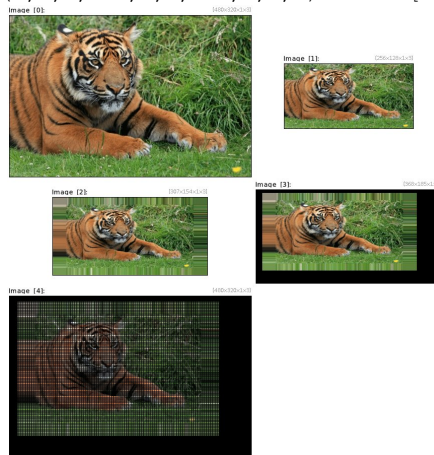
(no arg) runs interactive mode (uses the instant display window [0] if opened).

Default values:

- 'interpolation=1', 'boundary_conditions=0' and 'ax=ay=az=ac=0'.



Example 256 : `image.jpg (0,1;0,1^0,0;1,1^1,1;1,1) resize[-1] [-2],3 mul[-2] [-1]`



Example 257 : `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`

2.7.22 *resize_mn*

Arguments:

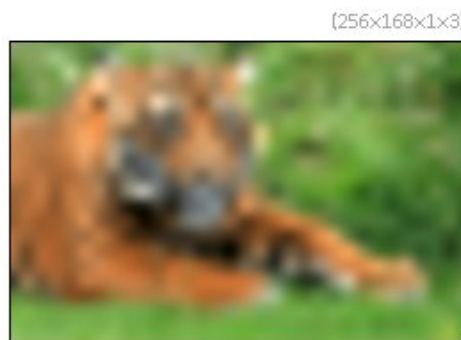
- `width[%]>=0, _height[%]>=0, _depth[%]>=0, _B.value, _C.value`

Resize selected images with Mitchell-Netravali filter (cubic).

For details about the method, see: <https://de.wikipedia.org/wiki/Mitchell-Netravali-Filter>

Default values:

- `'height=100%', 'depth=100%', 'B=0.3333' and 'C=0.3333'.`



Example 258: `image.jpg resize2dx 32 resize_mn 800%,800%`

2.7.23 *resize_pow2*

Arguments:

- `_interpolation, _boundary_conditions, _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_conditions' has different meanings, according to the chosen 'interpolation' mode : . When 'interpolation=={ -1 | 1 | 2 | 4 }', 'boundary_conditions' is meaningless. . When 'interpolation==0', 'boundary_conditions' can be { 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }. . When 'interpolation=={ 3 | 5 | 6 }', 'boundary_conditions' can be { 0=none | 1=neumann }.

'ax,ay,az,ac' set the centering along each axis when 'interpolation=0' (set to '0' by default, must be defined in range [0,1]).

Default values:

- `'interpolation=0', 'boundary_conditions=0' and 'ax=ay=az=ac=0'.`



Example 259 : `image.jpg +resize.pow2[-1] 0`

2.7.24 *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'.

2.7.25 *resize2dx*

Arguments:

- `width[%]>0,interpolation,boundary_conditions,_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_conditions' has different meanings, according to the chosen 'interpolation' mode : . When 'interpolation'=={ -1 | 1 | 2 | 4 }, 'boundary_conditions' is meaningless. . When 'interpolation'==0, 'boundary_conditions' can be { 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }. . When 'interpolation'=={ 3 | 5 | 6 }, 'boundary_conditions' can be { 0=none | 1=neumann }.

'ax,ay,az,ac' set the centering along each axis when 'interpolation=0'
(set to '0' by default, must be defined in range [0,1]).

Default values:

- 'interpolation=3', 'boundary_conditions=0' and 'ax=ay=az=ac=0'.



Example 260 : `image.jpg +resize2dx 100,2 append x`

2.7.26 *resize2dy*

Arguments:

- `height[%]>=0, _interpolation, _boundary_conditions, _ax, _ay, _az, _ac`

Resize selected images along the y-axis, preserving 2D ratio.

(eq. to 'r2dy') . \n).

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

'boundary_conditions' has different meanings, according to the chosen 'interpolation' mode : . When 'interpolation=={ -1 | 1 | 2 | 4 }', 'boundary_conditions' is meaningless. . When 'interpolation==0', 'boundary_conditions' can be { 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }. . When 'interpolation=={ 3 | 5 | 6 }', 'boundary_conditions' can be { 0=none | 1=neumann }.

'ax,ay,az,ac' set the centering along each axis when 'interpolation=0'

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

Default values:

- 'interpolation=3', 'boundary_conditions=0' and 'ax=ay=az=ac=0'.



Example 261 : `image.jpg +resize2dy 100,2 append x`

2.7.27 *resize3dx*

Arguments:

- `width[%]>0, _interpolation, _boundary_conditions, _ax, _ay, _az, _ac`

Resize selected images along the x-axis, preserving 3D ratio.

(eq. to 'r3dx') .\n).

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

'boundary_conditions' has different meanings, according to the chosen 'interpolation' mode : . When 'interpolation=={ -1 | 1 | 2 | 4 }', 'boundary_conditions' is meaningless. . When 'interpolation==0', 'boundary_conditions' can be { 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }. . When 'interpolation=={ 3 | 5 | 6 }', 'boundary_conditions' can be { 0=none | 1=neumann }.

'ax,ay,az,ac' set the centering along each axis when 'interpolation=0' (set to '0' by default, must be defined in range [0,1]).

Default values:

- 'interpolation=3', 'boundary_conditions=0' and 'ax=ay=az=ac=0'.

2.7.28 *resize3dy*

Arguments:

- `height[%]>0, _interpolation, _boundary_conditions, _ax, _ay, _az, _ac`

Resize selected images along the y-axis, preserving 3D ratio.

(eq. to 'r3dy') .\n).

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

'boundary_conditions' has different meanings, according to the chosen 'interpolation' mode : . When 'interpolation=={ -1 | 1 | 2 | 4 }', 'boundary_conditions' is meaningless. . When 'interpolation==0', 'boundary_conditions' can be { 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }. . When 'interpolation=={ 3 | 5 | 6 }', 'boundary_conditions' can be { 0=none | 1=neumann }.

'ax,ay,az,ac' set the centering along each axis when 'interpolation=0' (set to '0' by default, must be defined in range [0,1]).

Default values:

- 'interpolation=3', 'boundary_conditions=0' and 'ax=ay=az=ac=0'.

2.7.29 *resize3dz*

Arguments:

- `depth[%]>0, _interpolation, _boundary_conditions, _ax, _ay, _az, _ac`

Resize selected images along the z-axis, preserving 3D ratio.

(eq. to `r3dz'`) . \n).

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

'boundary_conditions' has different meanings, according to the chosen 'interpolation' mode : . When 'interpolation=={ -1 | 1 | 2 | 4 }', 'boundary_conditions' is meaningless. . When 'interpolation==0', 'boundary_conditions' can be { 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }. . When 'interpolation=={ 3 | 5 | 6 }', 'boundary_conditions' can be { 0=none | 1=neumann }.

'ax,ay,az,ac' set the centering along each axis when 'interpolation=0' (set to '0' by default, must be defined in range [0,1]).

Default values:

- 'interpolation=3', 'boundary_conditions=0' and 'ax=ay=az=ac=0'.

2.7.30 *rotate* (+)

Arguments:

- `angle, _interpolation, _boundary_conditions, _center_x[_], _center_y[_]`
- `u, v, w, angle, interpolation, boundary_conditions, _center_x[_], _center_y[_], _center_z[_]`

Rotate selected images with specified angle (in deg.), and optionally 3D axis (u,v,w).

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

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

When a rotation center (cx,cy,cz) is specified, the size of the image is preserved.

Default values:

- 'interpolation=1', 'boundary_conditions=0' and 'center_x=center_y=(undefined)'.



Example 262 : `image.jpg +rotate -25,1,2,50%,50% rotate[0] 25`

2.7.31 *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'`.

2.7.32 *rows (+)*

Arguments:

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

Keep only specified rows of selected images.

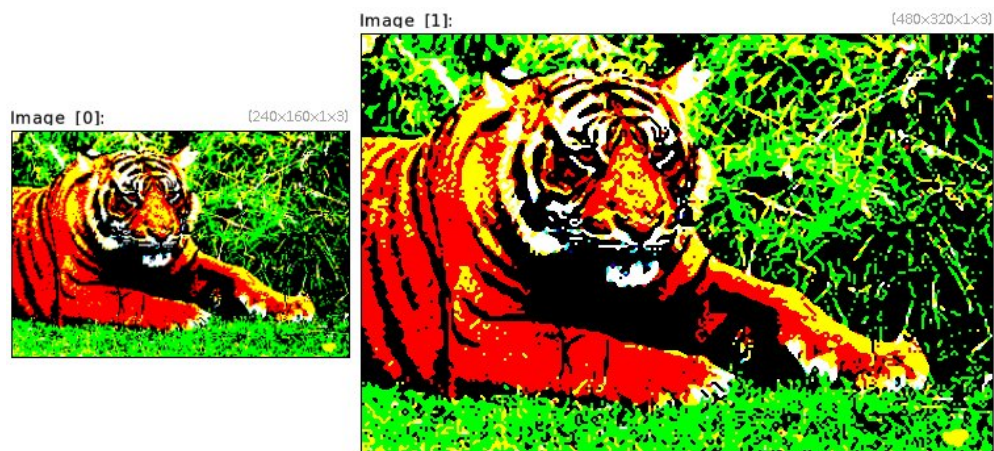
Dirichlet boundary conditions are used when specified rows are out of range.



Example 263 : `image.jpg rows -25%,50%`

2.7.33 *scale2x*

Resize selected images using the Scale2x algorithm.



Example 264 : `image.jpg threshold 50% resize 50%,50% +scale2x`

2.7.34 *scale3x*

Resize selected images using the Scale3x algorithm.



Example 265 : `image.jpg threshold 50% resize 33%,33% +scale3x`

2.7.35 *scale_dcci2x***Arguments:**

- `_edge_threshold>=0, _exponent>0, _extend_lpx={ 0=false | 1=true }`

Double image size using directional cubic convolution interpolation, as described in https://en.wikipedia.org/wiki/Directional_Cubic_Convolution_Interpolation.

Default values:

- `'edge_threshold=1.15', 'exponent=5' and 'extend_lpx=0'.`



Example 266 : `image.jpg +scale_dcci2x ,`

2.7.36 *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 267 : `image.jpg seamcarve 60%`

2.7.37 *shift (+)*

Arguments:

- `vx[%], _vy[%], _vz[%], _vc[%], _boundary_conditions, _interpolation={ 0=nearest_neighbor | 1=linear }`

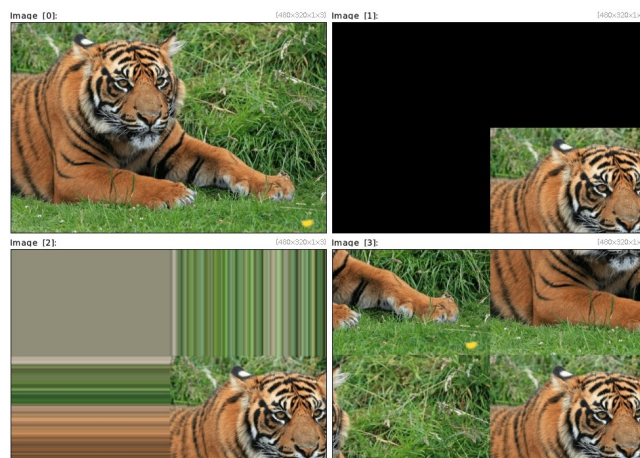
Shift selected images by specified displacement vector.

Displacement vector can be non-integer in which case linear interpolation of the shift is computed.

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

Default value:

- `'boundary_conditions=0' and 'interpolation=0'.`



Example 268 : `image.jpg +shift[0] 50%,50%,0,0,0 +shift[0] 50%,50%,0,0,1 +shift[0] 50%,50%,0,0,2`

2.7.38 *shrink_x*

Arguments:

- `size_x>=0`

Shrink selected images along the x-axis.



Example 269 : `image.jpg shrink_x 30`

2.7.39 *shrink_xy*

Arguments:

- `size>=0`

Shrink selected images along the xy-axes.



Example 270 : `image.jpg shrink_xy 30`

2.7.40 *shrink_xyz*

Arguments:

- `size>=0`

Shrink selected images along the xyz-axes.

2.7.41 *shrink_y*

Arguments:

- `size_y>=0`

Shrink selected images along the y-axis.



Example 271 : `image.jpg shrink_y 30`

2.7.42 *shrink_z*

Arguments:

- `size_z >= 0`

Shrink selected images along the z-axis.

2.7.43 *slices (+)*

Arguments:

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

Keep only specified slices of selected images.

Dirichlet boundary conditions are used when specified slices are out of range.

2.7.44 *sort (+)*

Arguments:

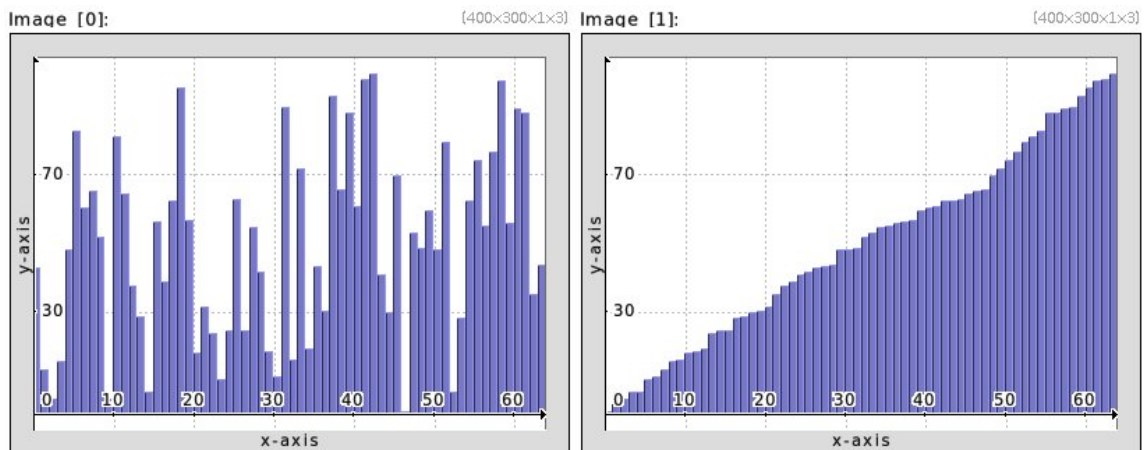
- `_ordering={ + | - }, _axis={ x | y | z | c }`

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 272 : `64 rand 0,100 +sort display_graph 400,300,3`

2.7.45 *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 arg)

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

(eq. to 's') .\n).

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

Default value:

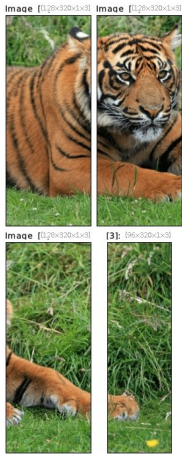
- 'split_mode=-1'.



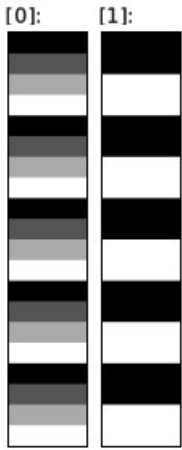
Example 273 : `image.jpg split c`



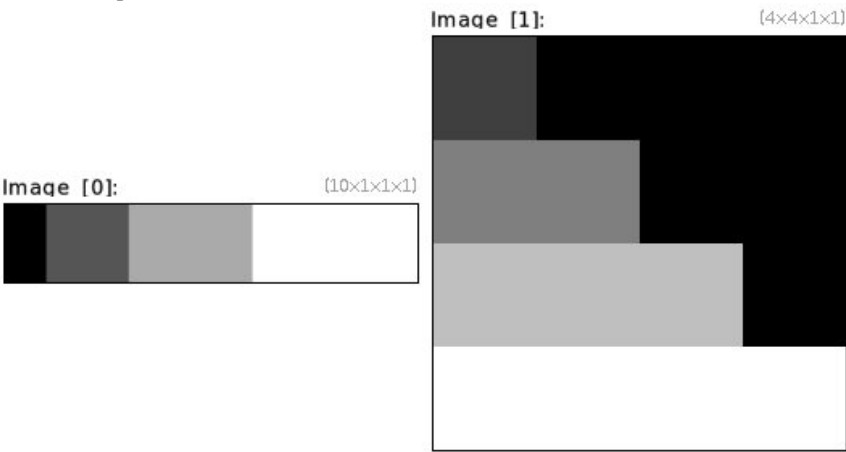
Example 274 : `image.jpg split y,3`



Example 275 : image.jpg split x,-128



Example 276 : 1,20,1,1, "1,2,3,4" +split -,2,3 append[1--1] y



Example 277 : (1,2,2,3,3,3,4,4,4,4) +split x,0 append[1--1] y

2.7.46 *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 278: `image.jpg +local split_tiles 5,4 blur 3,0 sharpen 700 append_tiles 4,5 endl`

2.7.47 *undistort*

Arguments:

- `-1<=_amplitude<=1, _aspect_ratio, _zoom, _center_x[%], _center_y[%],
_boundary_conditions`

Correct barrel/pincushion distortions occurring with wide-angle lens.

References: [1] Zhang Z. (1999). Flexible camera calibration by viewing a plane from unknown orientation.
[2] Andrew W. Fitzgibbon (2001). Simultaneous linear estimation of multiple view geometry and lens distortion.

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

Default values:

- `'amplitude=0.25', 'aspect_ratio=0', 'zoom=0', 'center_x=center_y=50%' and
'boundary_conditions=0'`.

2.7.48 *unroll (+)*

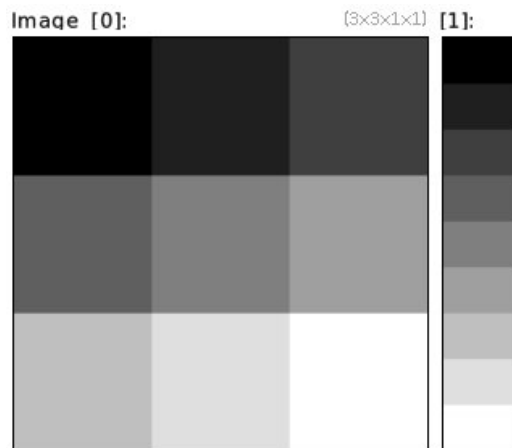
Arguments:

- `_axis={ x | y | z | c }`

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

Default value:

- 'axis=y'.



Example 279 : (1,2,3;4,5,6;7,8,9) +unroll y

2.7.49 *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 280 : image.jpg resize2dy 100 +upscale_smart 500%,500% append x

2.7.50 *warp* (+)

Arguments:

- `[warping_field],_mode,_interpolation,_boundary_conditions,_nb_frames>0`

Warp selected images with specified displacement field.

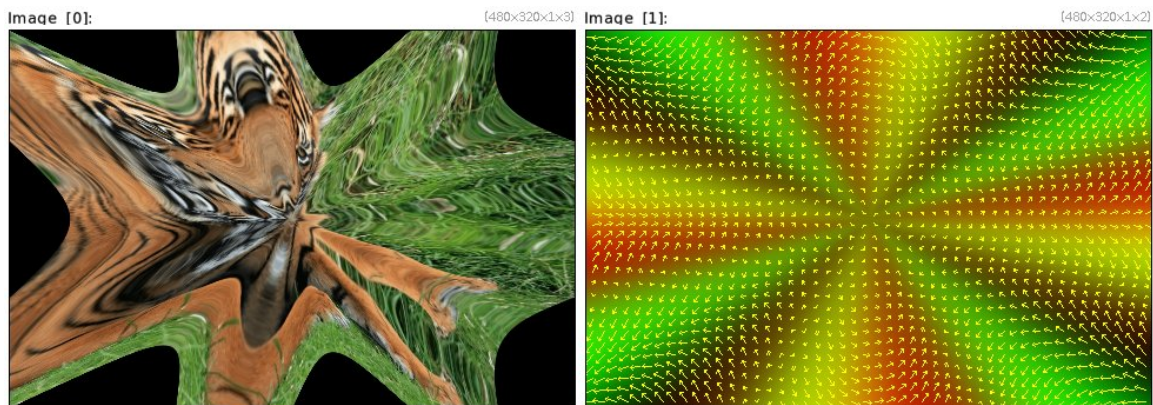
'mode' can be { 0=backward-absolute | 1=backward-relative | 2=forward-absolute | 3=forward-relative }.

'interpolation' can be { 0=nearest-neighbor | 1=linear | 2=cubic }.

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

Default values:

- 'mode=0', 'interpolation=1', 'boundary_conditions=1' and 'nb_frames=1'.



Example 281 : `image.jpg 100%,100%,1,2,'X=x/w-0.5;Y=y/h-0.5;R=(X*X+Y*Y)^0.5;A=atan2(Y,X);130*R*if(c==0,cos(4*A),sin(8*A))' warp[-2] [-1],1,1,0 quiver[-1] [-1],10,1,1,1,100`

Tutorial page:

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

2.7.51 *warp_patch*

Arguments:

- `[warping_field],patch_width>=1,_patch_height>=1,_patch_depth>=1,_std_factor>0,_boundary_conditions.`

Patch-warp selected images, with specified 2D or 3D displacement field (in backward-absolute mode).

Argument 'std_factor' sets the std of the gaussian weights for the patch overlap, equal to 'std = std_factor*patch_size'.

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

Default values:

- 'std_factor=0.3' and 'boundary_conditions=3'.

2.8 Filtering

2.8.1 *bandpass*

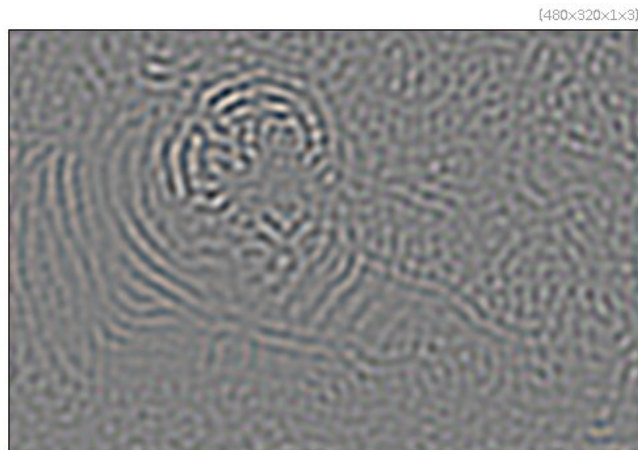
Arguments:

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

Apply bandpass filter to selected images.

Default values:

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



Example 282 : `image.jpg bandpass 1%,3%`

Tutorial page:

https://gmics.eu/tutorial/_bandpass.shtml

2.8.2 *bilateral (+)*

Arguments:

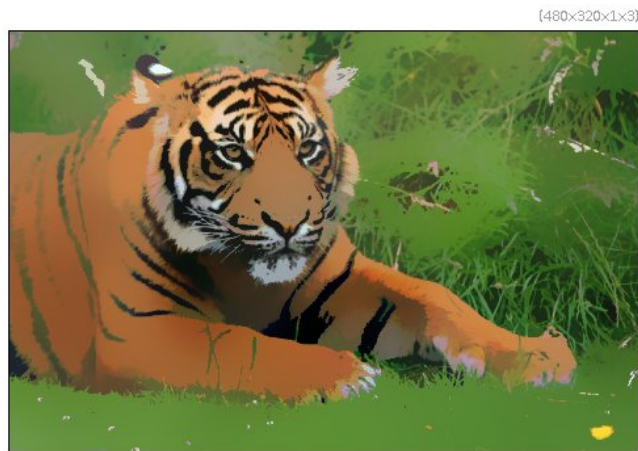
- `[guide], std_deviation_s[%]>=0, std_deviation_r[%]>=0, _sampling_s>=0, _sampling_r>=0`
- `std_deviation_s[%]>=0, std_deviation_r[%]>=0, _sampling_s>=0, _sampling_r>=0`

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

If a guide image is provided, it is used to 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 283 : `image.jpg repeat 5 bilateral 10,10 done`

2.8.3 *blur* (+)

Arguments:

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

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

(*eq. to 'b'*) . \n).

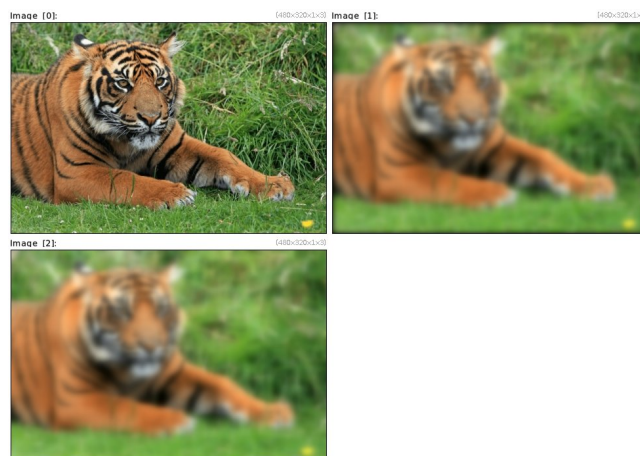
'boundary_conditions' can be { 0=dirichlet | 1=neumann } and 'kernel' can be { 0=quasi-gaussian (faster) | 1=gaussian }.

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

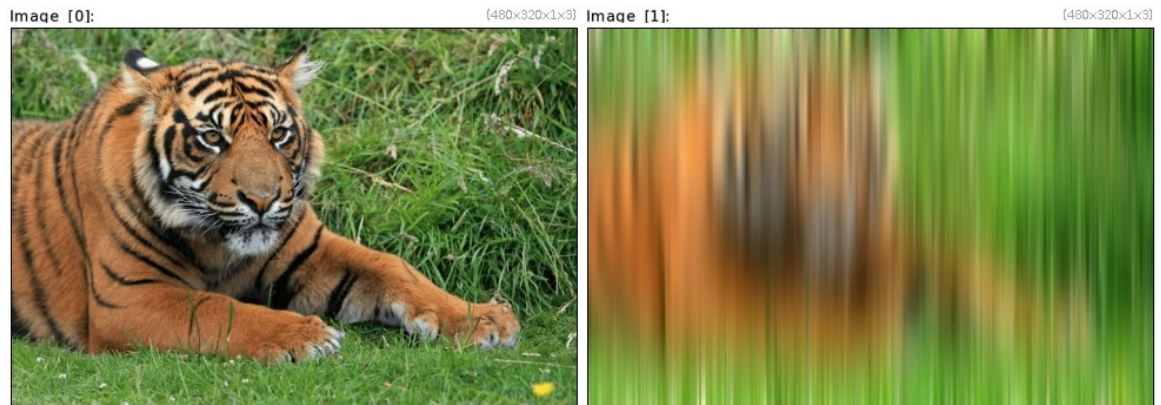
Specifying one axis multiple times apply also the blur multiple times.

Default values:

- 'boundary_conditions=1' and 'kernel=0'.



Example 284 : `image.jpg +blur 5,0 +blur[0] 5,1`



Example 285 : `image.jpg +blur y,10%`

Tutorial page:

https://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 286 : `image.jpg blur.angular 2%`

Tutorial page:

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

2.8.5 *blur_bloom*

Arguments:

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

Apply a bloom filter that blend multiple blur filters of different radii, resulting in a larger but sharper glare than a simple blur.

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

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

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

Default values:

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



Example 287: `image.jpg blur.bloom ,`

2.8.6 *blur_linear*

Arguments:

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

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

Default values:

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



Example 288 : `image.jpg blur-linear 10,0,45`

Tutorial page:

https://gmic.eu/tutorial/_blur_linear.shtml

2.8.7 *blur_radial*

Arguments:

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

Apply radial blur on selected images.

Default values:

- `'center_x=center_y=50%'`.



Example 289 : `image.jpg blur-radial 2%`

Tutorial page:

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

2.8.8 *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 290 : `image.jpg noise 20 cut 0,255 +local[-1] repeat 4 blur_selective , done endlcal`

Tutorial page:

https://gmics.eu/tutorial/_blur_selective.shtml

2.8.9 *blur_x*

Arguments:

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

Blur selected images along the x-axis.

Default value:

- `'boundary_conditions=1'.`



Example 291 : image.jpg +blur_x 6

Tutorial page:

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

2.8.10 *blur_xy*

Arguments:

- `amplitude_x[%], amplitude_y[%], boundary_conditions={ 0=dirichlet | 1=neumann }`

Blur selected images along the X and Y axes.

Default value:

- `'boundary_conditions=1'`.



Example 292 : image.jpg +blur_xy 6

Tutorial page:

https://gmics.eu/tutorial/_blur_xy.shtml

2.8.11 *blur_xyz*

Arguments:

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

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

Default value:

- `'boundary_conditions=1'`.

Tutorial page:

https://gmic.eu/tutorial/_blur_xyz.shtml

2.8.12 *blur_y*

Arguments:

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

Blur selected images along the y-axis.

Default value:

- `'boundary_conditions=1'`.



Example 293 : `image.jpg +blur.y 6`

Tutorial page:

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

2.8.13 *blur_z*

Arguments:

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

Blur selected images along the z-axis.

Default value:

- `'boundary_conditions=1'`.

Tutorial page:

https://gmics.eu/tutorial/_blur_z.shtml

2.8.14 *boxfilter* (+)

Arguments:

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

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

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

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

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

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

Default values:

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



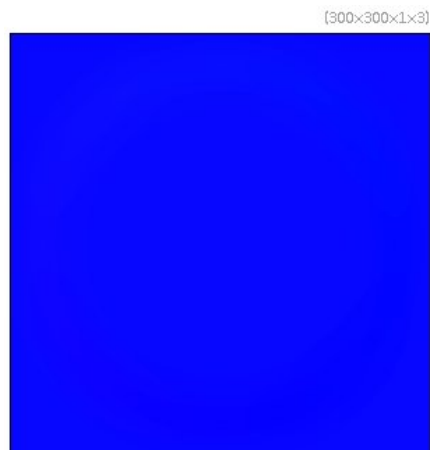
Example 294 : `image.jpg +boxfilter 5%`



Example 295 : `image.jpg +boxfilter y,3,1`

2.8.15 *bump2normal*

Convert selected bumpmaps to normalmaps.



Example 296 : `300,300 circle 50%,50%,128,1,1 blur 5% bump2normal`

2.8.16 *compose freq*

Compose selected low and high frequency parts into new images.



Example 297 : `image.jpg split.freq 2% mirror[-1] x compose.freq`

2.8.17 *convolve* (+)

Arguments:

- `[mask],_boundary_conditions,_is_normalized={ 0 | 1 }`

Convolve selected images by specified mask.

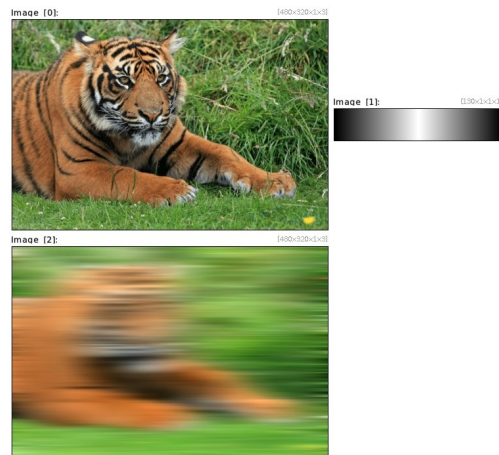
'boundary_conditions' can be { 0=dirichlet | 1=neumann }.

Default values:

- 'boundary_conditions=1' and 'is_normalized=0'.



Example 298 : `image.jpg (0,1,0;1,-4,1;0,1,0) convolve[-2] [-1] keep[-2]`



Example 299 : `image.jpg (0,1,0) resize[-1] 130,1,1,1,3 +convolve[0] [1]`

Tutorial page:

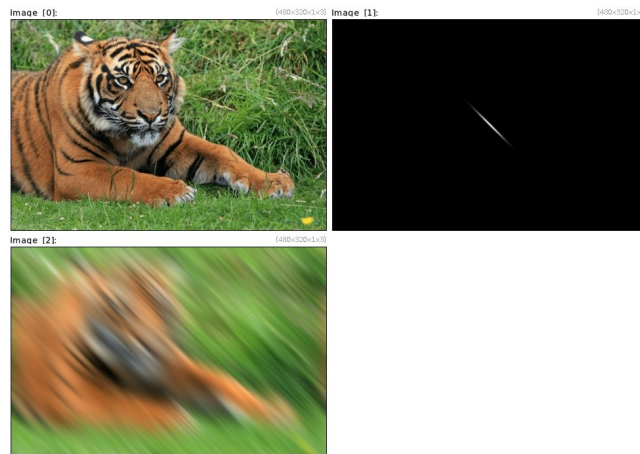
https://gmic.eu/tutorial/_convolve.shtml

2.8.18 *convolve_fft*

Arguments:

- `[mask]`

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



Example 300 : `image.jpg 100%,100% gaussian[-1] 20,1,45 +convolve_fft[0] [1]`

2.8.19 *correlate (+)*

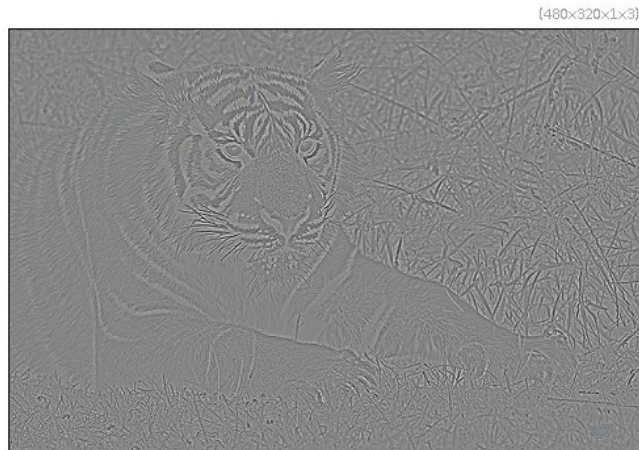
Arguments:

- `[mask],_boundary_conditions,_is.normalized={ 0 | 1 }`

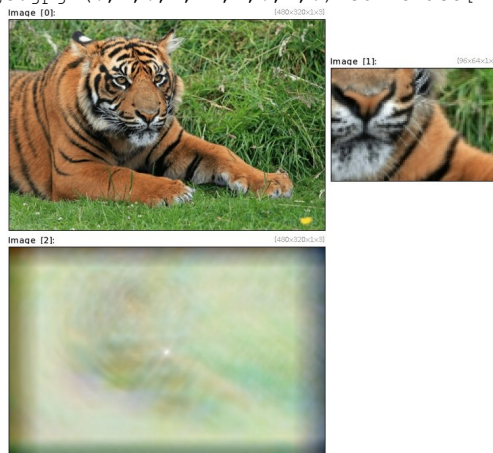
Correlate selected images by specified mask.
 'boundary_conditions' can be { 0=dirichlet | 1=neumann }.

Default values:

- 'boundary_conditions=1' and 'is_normalized=0'.



Example 301 : `image.jpg (0,1,0;1,-4,1;0,1,0) correlate[-2] [-1] keep[-2]`



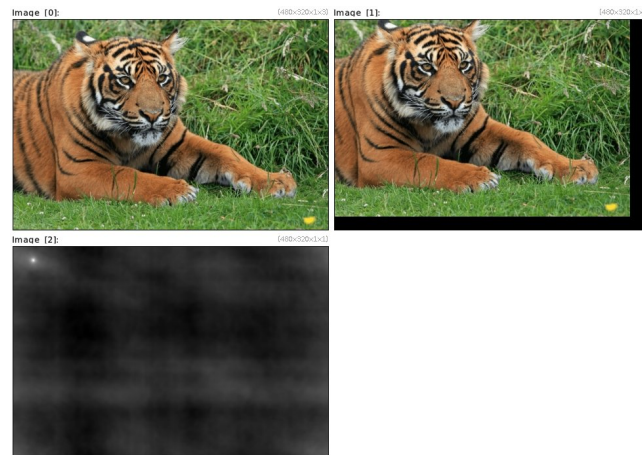
Example 302 : `image.jpg +crop 40%,40%,60%,60% +correlate[0] [-1],0,1`

2.8.20 *cross_correlation*

Arguments:

- [mask]

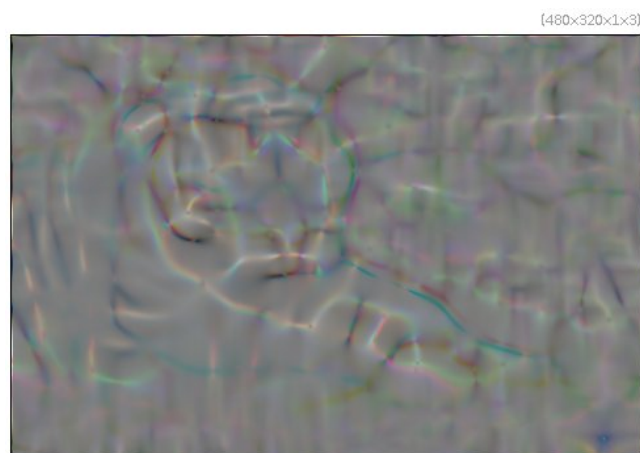
Compute cross-correlation of selected images with specified mask.



Example 303 : `image.jpg +shift -30,-20 +cross_correlation[0] [1]`

2.8.21 *curvature*

Compute isophote curvatures on selected images.



Example 304 : `image.jpg blur 10 curvature`

2.8.22 *dct*

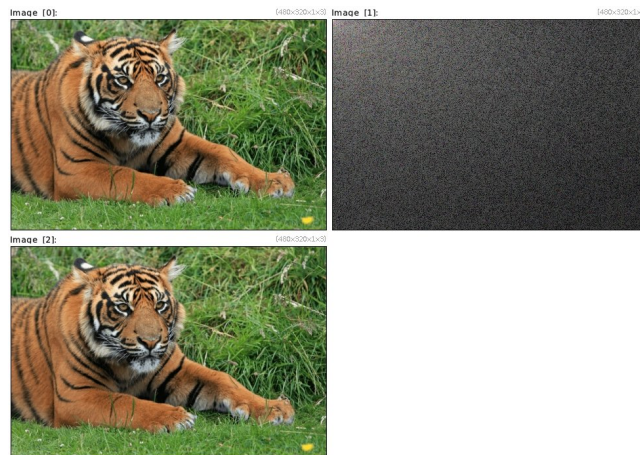
Arguments:

- `-{ 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 305 : `image.jpg +dct +idct [-1] abs [-2] + [-2] 1 log [-2]`

Tutorial page:

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

2.8.23 *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 306 : `image.jpg blur 3 +deblur 3,40,20,0.01`

2.8.24 *deblur_goldmeinel*

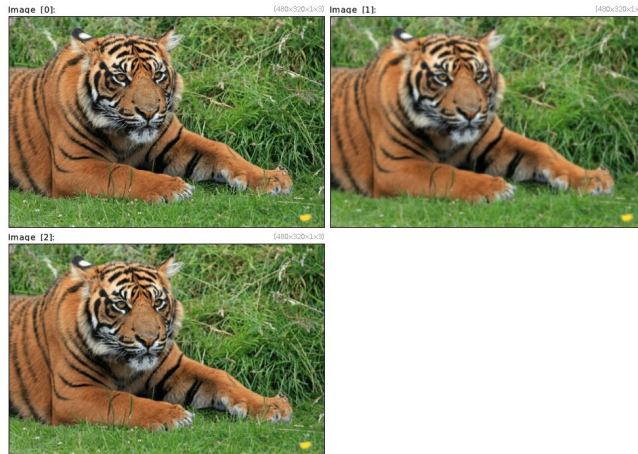
Arguments:

- `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 307 : `image.jpg +blur 1 +deblur.goldmein1[-1] 1`

2.8.25 *deblur_richardsonlucy*

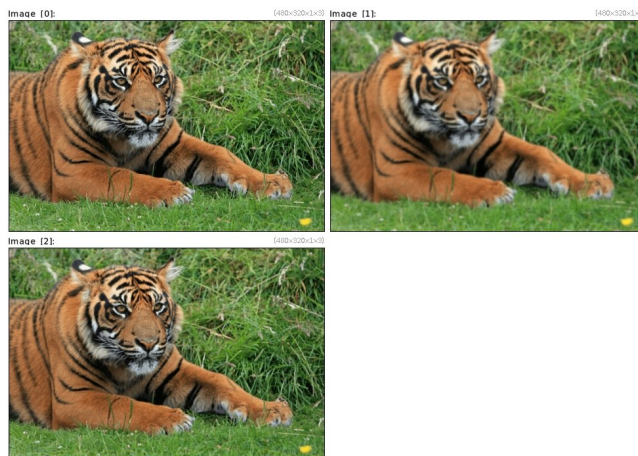
Arguments:

- `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 308 : `image.jpg +blur 1 +deblur_richardsonlucy[-1] 1`

2.8.26 *deconvolve_fft*

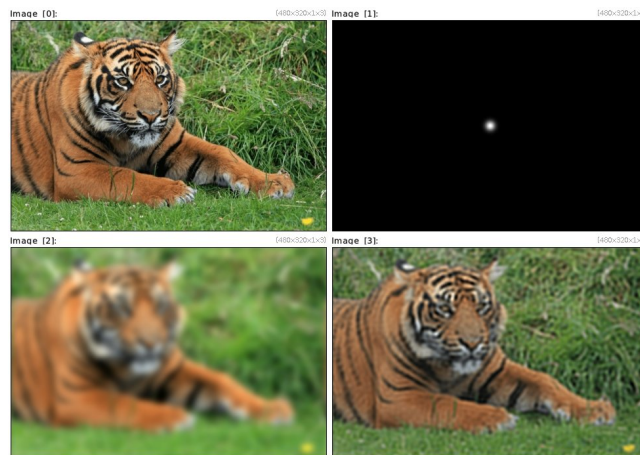
Arguments:

- `[kernel],_regularization>=0`

Deconvolve selected images by specified mask in the fourier space.

Default value:

- `'regularization>=0'`.



Example 309 : `image.jpg +gaussian 5 +convolve_fft[0] [1] +deconvolve_fft[-1] [1]`

2.8.27 *deinterlace*

Arguments:

- `_method={ 0 | 1 }`

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

Default value:

- `'method=0'`.



Example 310 : `image.jpg +rotate 3,1,1,50%,50% resize 100%,50% resize 100%,200%,1,3,4
shift[-1] 0,1 add +deinterlace 1`

2.8.28 *denoise (+)*

Arguments:

- `std_deviation_s>=0, _std_deviation_p>=0, _patch_size>0, _lookup_size>0, _smoothness, _fast_approx={ 0 | 1 }`

Denoise selected images by non-local patch averaging.

Default values:

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



Example 311 : `image.jpg +denoise 5,5,8`

2.8.29 *denoise_haar*

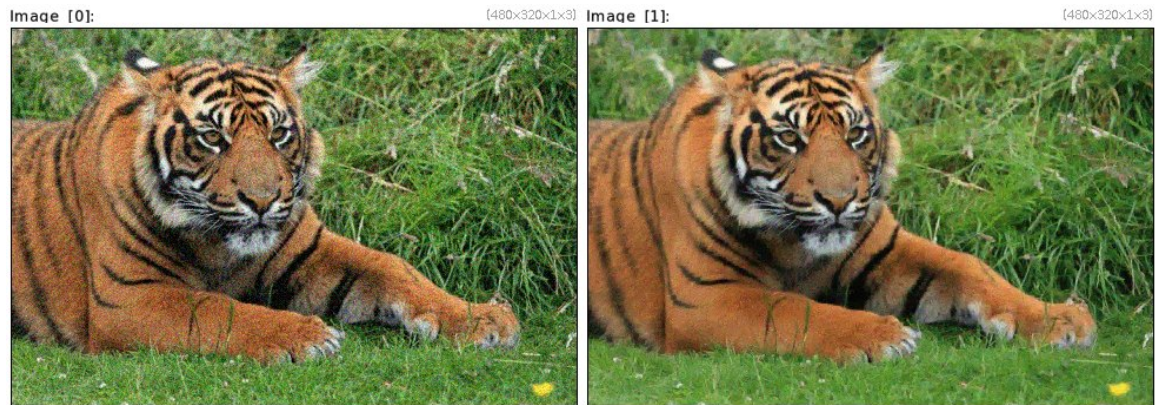
Arguments:

- `_threshold>=0, _nb_scales>=0, _cycle_spinning>0`

Denoise selected images using haar-wavelet thresholding with cycle spinning. Set `'nb_scales==0'` to automatically determine the optimal number of scales.

Default values:

- `'threshold=1.4', 'nb_scale=0' and 'cycle_spinning=10'.`



Example 312 : `image.jpg noise 20 cut 0,255 +denoise_haar[-1] 0.8`

2.8.30 *denoise_patchpca*

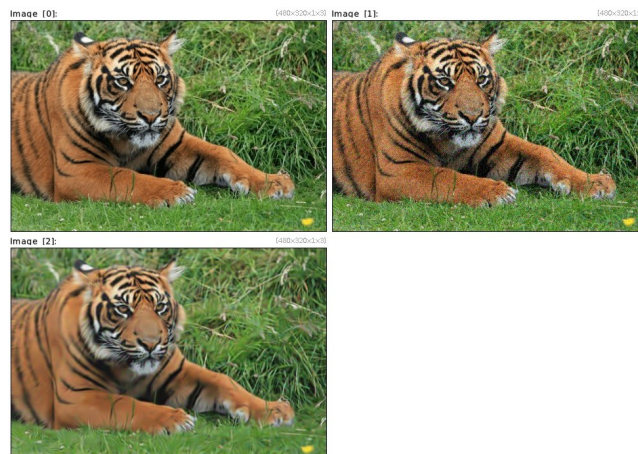
Arguments:

- `_strength>=0, _patch_size>0, _lookup_size>0, _spatial_sampling>0`

Denoise selected images using the patch-pca algorithm.

Default values:

- `'patch_size=7', 'lookup_size=11', 'details=1.8' and 'spatial_sampling=5'.`



Example 313 : `image.jpg +noise 20 cut [-1] 0,255 +denoise_patchpca[-1] ,`

2.8.31 *deriche (+)*

Arguments:

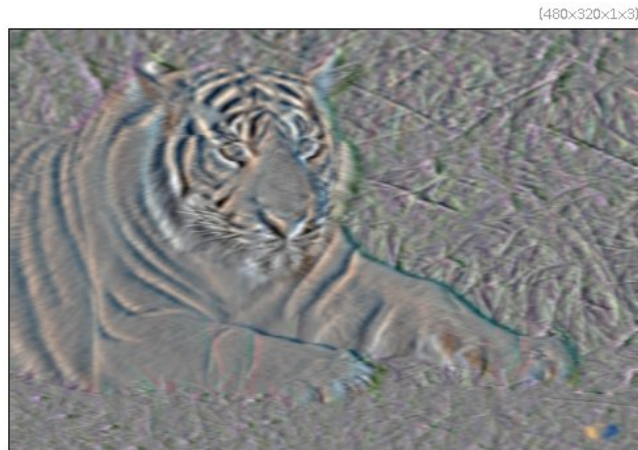
- `std_deviation>=0[%], order={ 0 | 1 | 2 }, axis={ x | y | z | c }, _boundary_conditions`

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

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

Default value:

- 'boundary_conditions=1'.



Example 314 : image.jpg deriche 3,1,x



Example 315 : image.jpg +deriche 30,0,x deriche[-2] 30,0,y add

Tutorial page:

https://gmics.eu/tutorial/_deriche.shtml

2.8.32 dilate (+)

Arguments:

- size>=0
- size_x>=0,size_y>=0,size_z>=0
- [kernel],_boundary_conditions,_is_real={ 0=binary-mode | 1=real-mode }

Dilate selected images by a rectangular or the specified structuring element.
 'boundary_conditions' can be { 0=dirichlet | 1=neumann }.

Default values:

- 'size_z=1', 'boundary_conditions=1' and 'is_real=0'.



Example 316 : image.jpg +dilate 10

2.8.33 dilate_circ

Arguments:

- _size>=0, _boundary_conditions, _is_normalized={ 0 | 1 }

Apply circular dilation of selected images by specified size.

Default values:

- 'boundary_conditions=1' and 'is_normalized=0'.



Example 317 : image.jpg +dilate_circ 7

2.8.34 *dilate_oct*

Arguments:

- `_size>=0, _boundary_conditions, _is_normalized={ 0 | 1 }`

Apply octagonal dilation of selected images by specified size.

Default values:

- `'boundary_conditions=1'` and `'is_normalized=0'`.



Example 318 : `image.jpg +dilate_oct 7`

2.8.35 *dilate_threshold*

Arguments:

- `size_x>=1, size_y>=1, size_z>=1, threshold>=0, boundary_conditions`

Dilate selected images in the (X,Y,Z,I) space.
'boundary_conditions' can be { 0=dirichlet | 1=neumann }.

Default values:

- `'size_y=size_x', 'size_z=1', 'threshold=255' and 'boundary_conditions=1'`.

2.8.36 *divergence*

Compute divergence of selected vector fields.



Example 319 : `image.jpg luminance +gradient append[-2,-1] c divergence[-1]`

2.8.37 *dog*

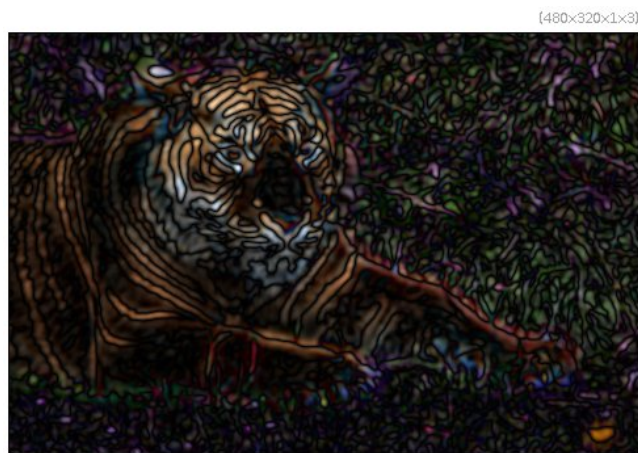
Arguments:

- `_sigma1>=0[%], _sigma2>=0[%]`

Compute difference of gaussian on selected images.

Default values:

- `'sigma1=2%' and 'sigma2=3%'.`



Example 320 : `image.jpg dog 2,3`

2.8.38 *diffusortensors*

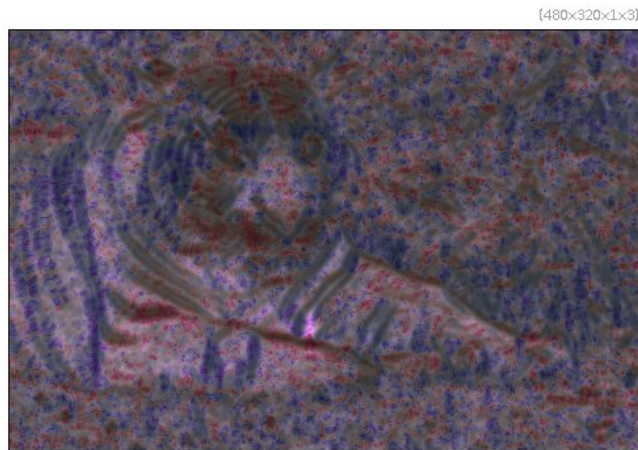
Arguments:

- `_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 321 : `image.jpg diffusiontensors 0.8 abs pow 0.2`

Tutorial page:

https://gmic.eu/tutorial/_diffusiontensors.shtml

2.8.39 edges**Arguments:**

- `_threshold[%]>=0`

Estimate contours of selected images.

Default value:

- 'edges=15%'



Example 322 : `image.jpg +edges 15%`

2.8.40 *erode* (+)

Arguments:

- `size>=0`
- `size_x>=0,size_y>=0,_size_z>=0`
- `[kernel],_boundary_conditions,_is_real={ 0=binary-mode | 1=real-mode }`

Erode selected images by a rectangular or the specified structuring element.
'boundary_conditions' can be { 0=dirichlet | 1=neumann }.

Default values:

- `'size_z=1', 'boundary_conditions=1' and 'is_real=0'.`



Example 323 : image.jpg +erode 10

2.8.41 *erode_circ*

Arguments:

- `_size>=0,_boundary_conditions,_is_normalized={ 0 | 1 }`

Apply circular erosion of selected images by specified size.

Default values:

- `'boundary_conditions=1' and 'is_normalized=0'.`



Example 324 : `image.jpg +erode_circ 7`

2.8.42 *erode_oct*

Arguments:

- `_size>=0, _boundary_conditions, _is_normalized={ 0 | 1 }`

Apply octagonal erosion of selected images by specified size.

Default values:

- `'boundary_conditions=1'` and `'is_normalized=0'`.



Example 325 : `image.jpg +erode_oct 7`

2.8.43 *erode_threshold*

Arguments:

- `size_x>=1, size_y>=1, size_z>=1, threshold>=0, boundary_conditions`

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

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

Default values:

- `'size_y=size_x', 'size_z=1', 'threshold=255'` and `'boundary_conditions=1'`.

2.8.44 *fft* (+)

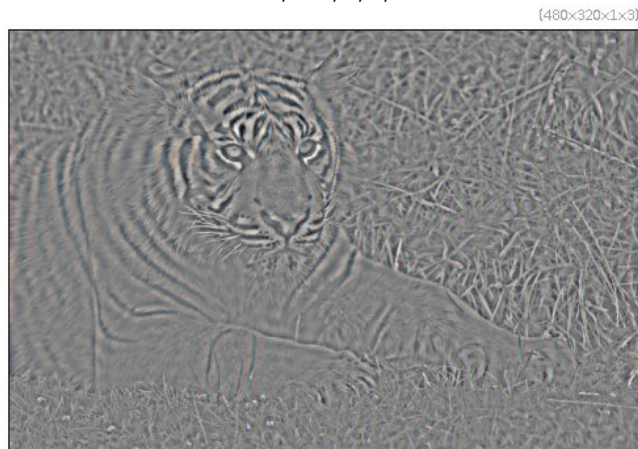
Arguments:

- $\{-x \mid y \mid z\} \dots \{x \mid y \mid z\}$

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



Example 326 : `image.jpg luminance +fft append[-2,-1] c norm[-1] log[-1] shift[-1] 50%,50%,0,0,2`



Example 327 : `image.jpg w2={int(w/2)} h2={int(h/2)} fft shift $w2,$h2,0,0,2 ellipse $w2,$h2,30,30,0,1,0 shift -$w2,-$h2,0,0,2 ifft remove[-1]`

Tutorial page:

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

2.8.45 *gradient* (+)

Arguments:

- $\{x \mid y \mid z\} \dots \{x \mid y \mid z\}, \text{-scheme}$
- (no arg)

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

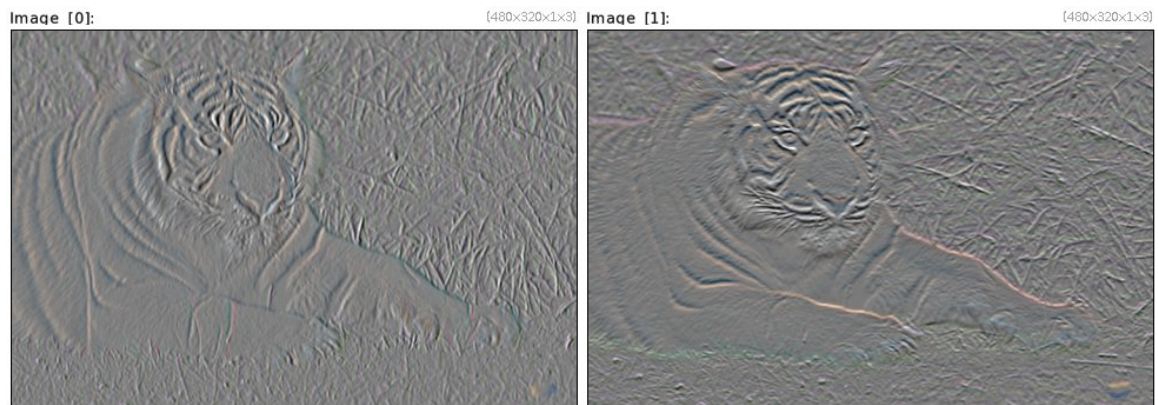
(eq. to 'g') .\n).

'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'.



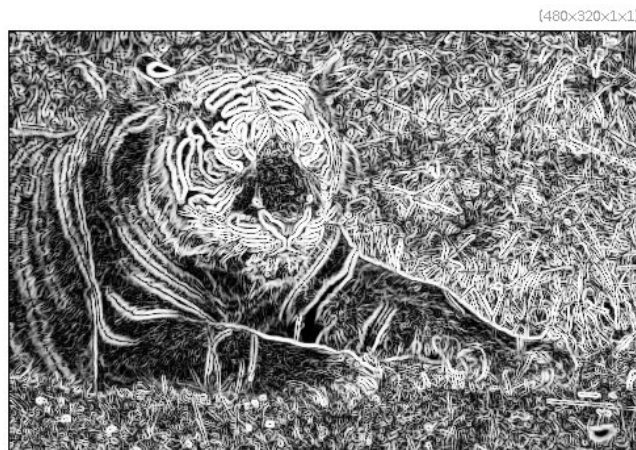
Example 328 : image.jpg gradient

Tutorial page:

https://gmic.eu/tutorial/_gradient.shtml

2.8.46 *gradient_norm*

Compute gradient norm of selected images.



Example 329 : image.jpg gradient_norm equalize

Tutorial page:

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

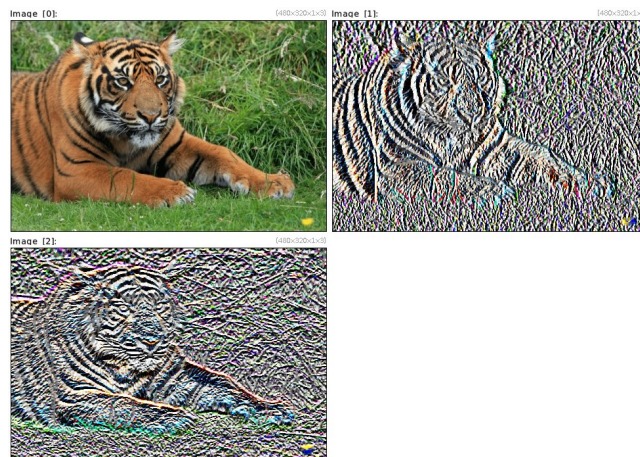
2.8.47 *gradient_orientation***Arguments:**

- `_dimension={1,2,3}`

Compute N-d gradient orientation of selected images.

Default value:

- `'dimension=3'`.



Example 330 : `image.jpg +gradient_orientation 2`

2.8.48 *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 331 : `image.jpg +guided 5,400`

2.8.49 *haar*

Arguments:

- `scale>0`

Compute the direct haar multiscale wavelet transform of selected images.

Tutorial page:

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

2.8.50 *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'.`



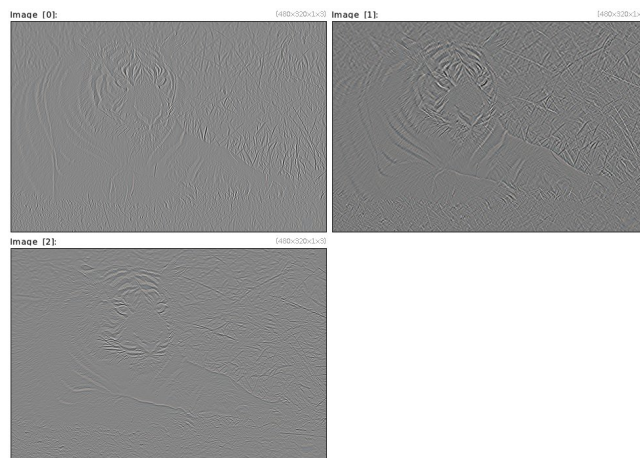
Example 332 : `image.jpg +heat_flow 20`

2.8.51 *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 333 : image.jpg hessian

2.8.52 *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:

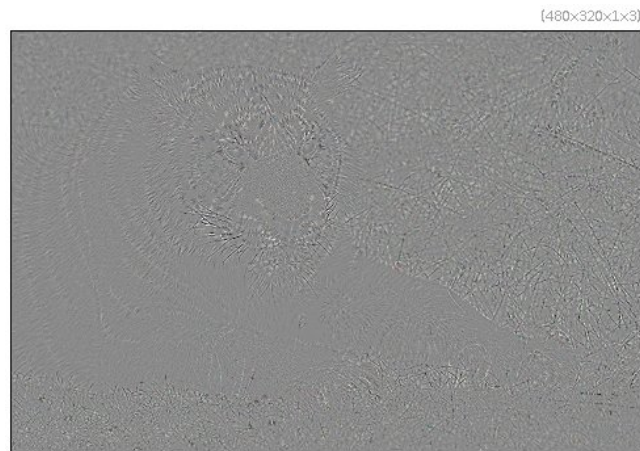
- (no arg)

Tutorial page:

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

2.8.53 *iee*

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



Example 334 : `image.jpg iee`

2.8.54 *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:

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

2.8.55 *ihaar*

Arguments:

- `scale>0`

Compute the inverse haar multiscale wavelet transform of selected images.

2.8.56 *ilaplacian*

Arguments:

- `{ nb_iterations>0 | 0 },_time_step>0,-[initial_estimate]`

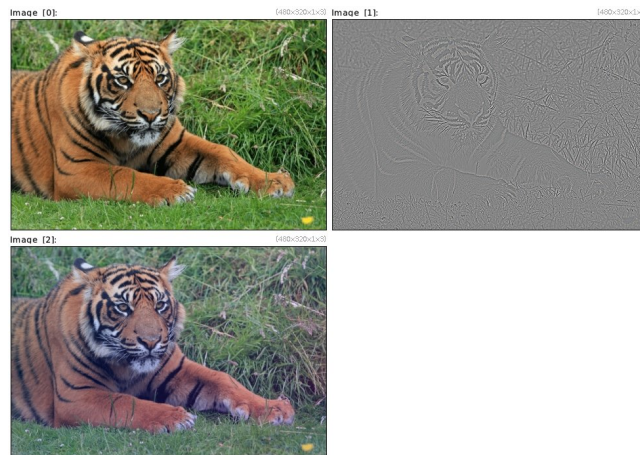
Invert selected Laplacian images.

If given 'nb_iterations' is '0', inversion is done in Fourier space (single iteration), otherwise, by applying 'nb_iterations' of a Laplacian-inversion PDE flow (with specified 'time_step').

Note that the resulting inversions are just estimation of possible/approximated solutions.

Default values:

- `'nb_iterations=0','time_step=10' and '[initial_estimated]=(undefined)'`.



Example 335 : `image.jpg +laplacian +ilaplacian[-1] 0`

2.8.57 *inn*

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



Example 336 : `image.jpg inn`

2.8.58 *inpaint (+)*

Arguments:

- `[mask]`
- `[mask], 0, _fast_method`
- `[mask], _patch_size>=1, _lookup_size>=1, _lookup_factor>=0, _lookup_increment!=0, _blend_size>=0, 0<=_blend_threshold<=1, _blend_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 337: `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 338: `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]`

2.8.59 *inpaint_diffusion*

Arguments:

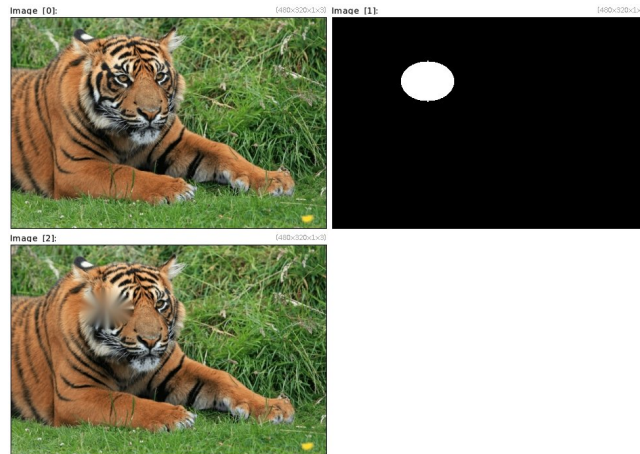
- `[mask],_nb_scales[%]>=0,_diffusion_type={ 0=isotropic | 1=delaunay-guided | 2=edge-guided | 3=mask-guided },_diffusion_iter>=0`

Inpaint selected images by specified mask using a multiscale transport-diffusion algorithm.

If 'diffusion type==3', non-zero values of the mask (e.g. a distance function) are used to guide the diffusion process.

Default values:

- 'nb_scales=75%', 'diffusion_type=1' and 'diffusion_iter=20'.



Example 339 : `image.jpg 100%,100% ellipse[-1] 30%,30%,40,30,0,1,255 +inpaint_diffusion[0] [1]`

2.8.60 *inpaint_flow***Arguments:**

- `[mask],_nb_global_iter>=0,_nb_local_iter>=0,_dt>0,_alpha>=0,_sigma>=0`

Apply iteration of the inpainting flow on selected images.

Default values:

- 'nb_global_iter=4', 'nb_global_iter=15', 'dt=10', 'alpha=1' and 'sigma=3'.



Example 340 : `image.jpg 100%,100% ellipse[-1] 30%,30%,40,30,0,1,255 inpaint_flow[0] [1]`

2.8.61 *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'.`



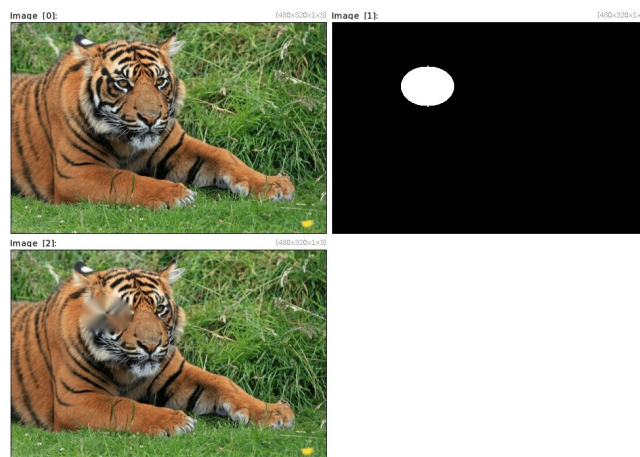
Example 341 : `image.jpg noise 5%,2 +inpaint.holes 8,40`

2.8.62 *inpaint_morpho*

Arguments:

- `[mask]`

Inpaint selected images by specified mask using morphological operators.



Example 342 : `image.jpg 100%,100% ellipse[-1] 30%,30%,40,30,0,1,255 +inpaint_morpho[0] [1]`

2.8.63 *inpaint_matchpatch*

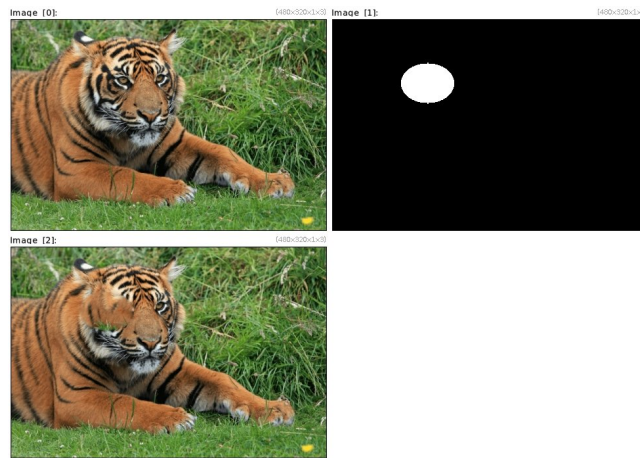
Arguments:

- `[mask],_nb_scales={ 0=auto | >0 }`
`},_patch_size>0,_nb_iterations_per_scale>0,_blend_size>=0,_allow_outer_blending={ 0 | 1 },_is_already_initialized={ 0 | 1 }`

Inpaint selected images by specified binary mask, using a multi-scale matchpatch algorithm.

Default values:

- `'nb_scales=0', 'patch_size=9', 'nb_iterations_per_scale=10', 'blend_size=5', 'allow_outer_blending=1' and 'is_already_initialized=0'.`



Example 343 : `image.jpg 100%,100% ellipse[-1] 30%,30%,40,30,0,1,255 +inpaint_matchpatch[0] [1]`

2.8.64 *kuwahara*

Arguments:

- `size>0`

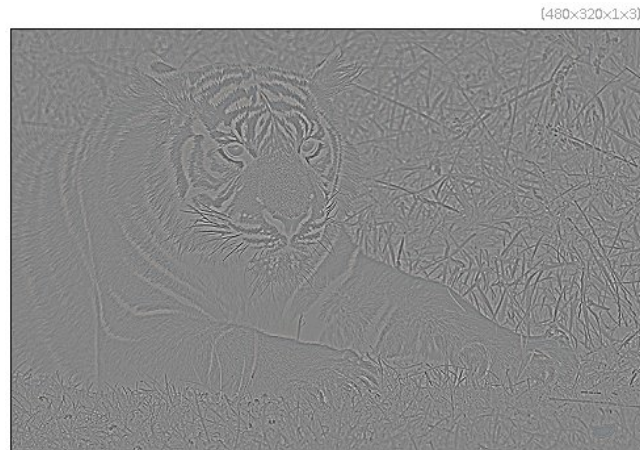
Apply Kuwahara filter of specified size on selected images.



Example 344 : `image.jpg +kuwahara 5`

2.8.65 *laplacian*

Compute Laplacian of selected images.



Example 345 : `image.jpg laplacian`

2.8.66 *lic*

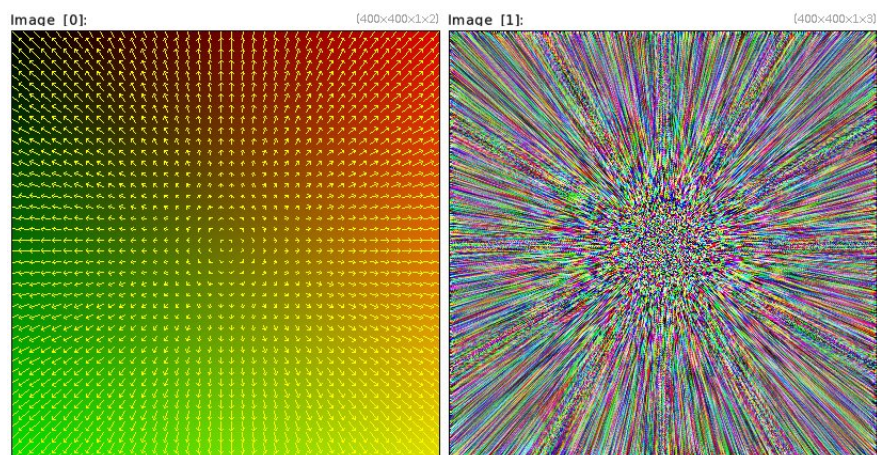
Arguments:

- `_amplitude>0, _channels>0`

Render LIC representation of selected vector fields.

Default values:

- `'amplitude=30' and 'channels=1'.`



Example 346 : `400,400,1,2,'if(c==0,x-w/2,y-h/2)'+lic 200,3 quiver[-2] [-2],10,1,1,1,255`

2.8.67 *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 347 : `image.jpg +map_tones ,`

2.8.68 *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 348 : `image.jpg +map_tones_fast ,`

2.8.69 *meancurvature_flow*

Arguments:

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

Apply iterations of the mean curvature flow on selected images.

Default values:

- `'nb_iter=10', 'dt=30' and 'keep_sequence=0'.`



Example 349 : `image.jpg +meancurvature_flow 20`

2.8.70 *median (+)*

Arguments:

- `size>=0, _threshold>0`

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



Example 350 : `image.jpg +median 5`

2.8.71 *nlmeans*

Arguments:

- `[guide], _patch_radius>0, _spatial_bandwidth>0, _tonal_bandwidth>0, _patch_measure_command`
- `_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 ($\exp(-d^2/_tonal_bandwidth^2)$) with d the euclidean distance between image patches.

Default values:

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



Example 351: `image.jpg +noise 10 nlmeans[-1] 4,4,{0.6*${-std.noise}}`

2.8.72 *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

2.8.73 *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'.

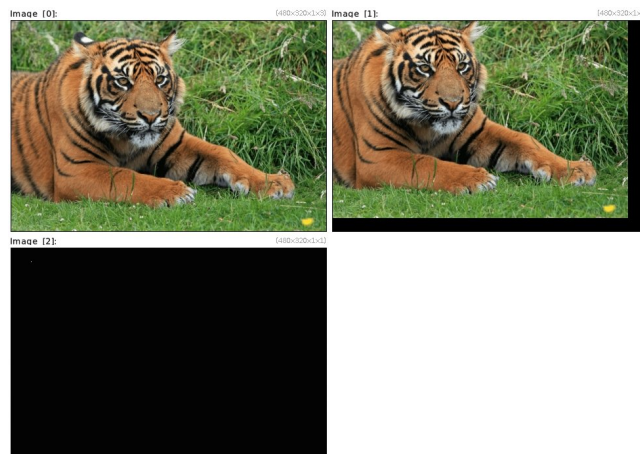


Example 352 : `image.jpg normalize_local 8,10`

2.8.74 *normalized_cross_correlation***Arguments:**

- [mask]

Compute normalized cross-correlation of selected images with specified mask.



Example 353 : `image.jpg +shift -30,-20 +normalized_cross_correlation[0] [1]`

2.8.75 *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 354 : `image.jpg +heat_flow 20`

2.8.76 *phase_correlation*

Arguments:

- `[destination]`

Estimate translation vector between selected source images and specified destination.



Example 355 : `image.jpg +shift -30,-20 +phase_correlation[0] [1] unroll[-1] y`

2.8.77 *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 356 : `image.jpg +pde_flow 20`

2.8.78 *periodize_poisson*

Periodize selected images using a Poisson solver in Fourier space.



Example 357 : `image.jpg +periodize_poisson array 2,2,2`

2.8.79 *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 358 : `image.jpg +red_eye ,`

2.8.80 *remove_hotpixels*

Arguments:

- `_mask_size>0, _threshold[%]>0`

Remove hot pixels in selected images.

Default values:

- `'mask_size=3' and 'threshold=10%'`.



Example 359 : `image.jpg noise 10,2 +remove_hotpixels ,`

2.8.81 *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 360 : `image.jpg +remove.pixels 50%`

2.8.82 *rolling_guidance*

Arguments:

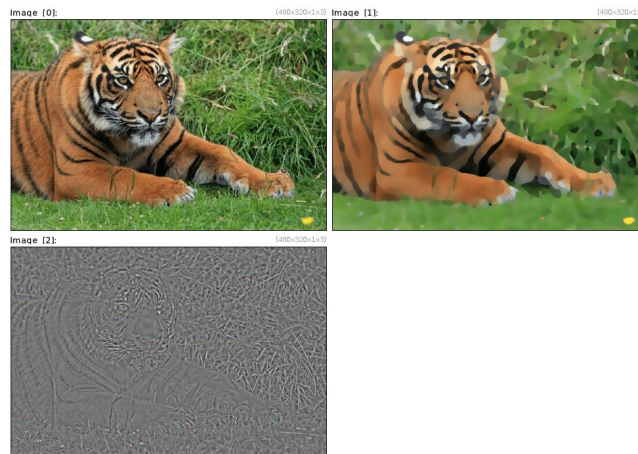
- `std_deviation_s[%]>=0, std_deviation_r[%]>=0, _precision>=0`

Apply the rolling guidance filter on selected image.

Rolling guidance filter is a fast image abstraction filter, described in: "Rolling Guidance Filter", Qi Zhang Xiaoyong, Shen Li, Xu Jiaya Jia, ECCV'2014.

Default values:

- `'std_deviation_s=4', 'std_deviation_r=10' and 'precision=0.5'.`



Example 361 : `image.jpg +rolling_guidance , +-`

2.8.83 *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 362 : image.jpg sharpen 300



Example 363 : image.jpg blur 5 sharpen 300,1

2.8.84 *smooth* (+)

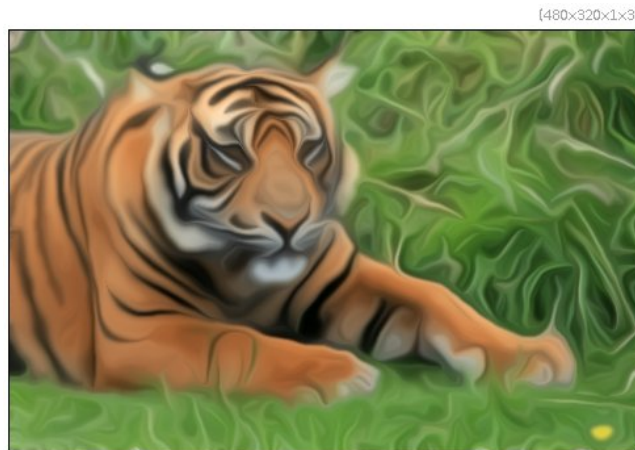
Arguments:

- amplitude[%]>=0, _sharpness>=0, 0<=_anisotropy<=1, _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

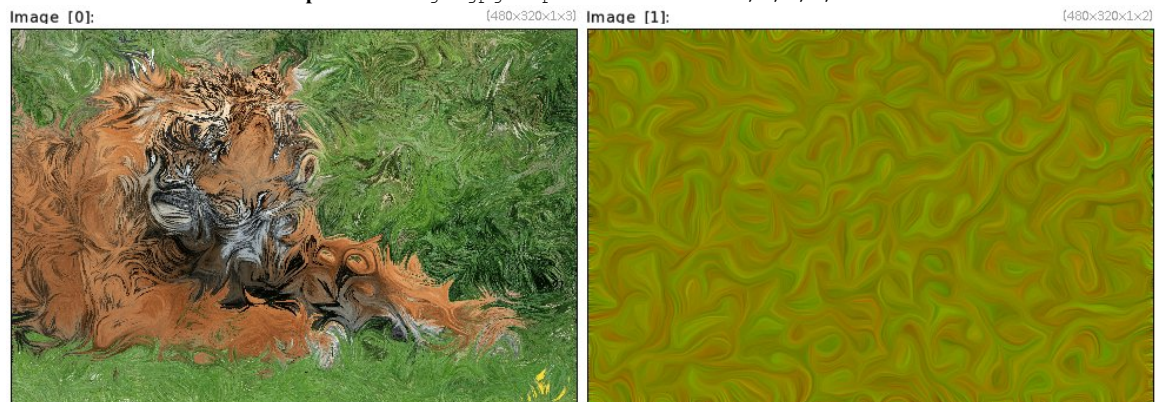
Smooth selected images anisotropically using diffusion PDE's, with specified field of diffusion tensors. 'interpolation' can be { 0=nearest | 1=linear | 2=runge-kutta }.

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 364 : image.jpg repeat 3 smooth 40,0,1,1,2 done



Example 365 : 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:

https://gmics.eu/tutorial/_smooth.shtml

2.8.85 *split_freq*

Arguments:

- smoothness>0 [%]

Split selected images into low and high frequency parts.



Example 366 : `image.jpg split_freq 2%`

2.8.86 *solve_poisson*

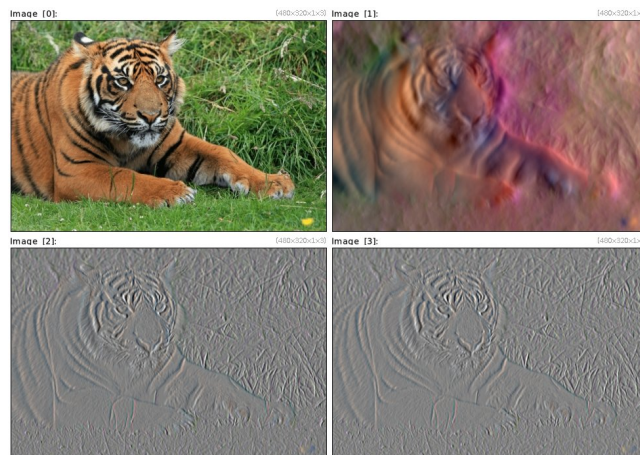
Arguments:

- `"laplacian_command", _nb_iterations>=0, _time_step>0, _nb_scales>=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 367 : `image.jpg command "foo : gradient x" +solve_poisson foo +foo[0] +laplacian[1]`

2.8.87 *split_details*

Arguments:

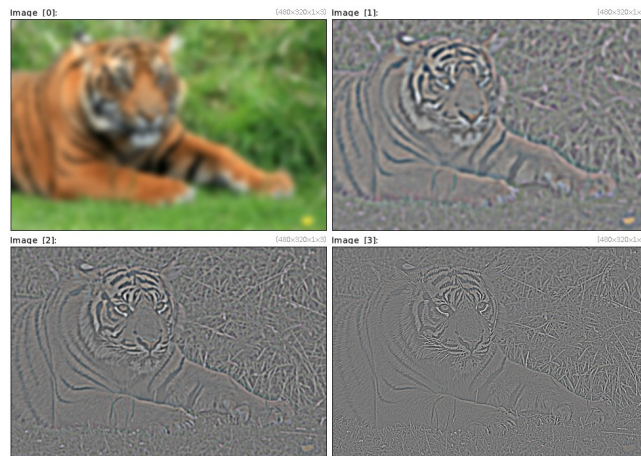
- `_nb_scales>0, _base_scale[%]>=0, _detail_scale[%]>=0`

Split selected images into 'nb_scales' detail scales.

If 'base_scale'=='detail_scale'==0, the image decomposition is done with 'a trous' wavelets. Otherwise, it uses laplacian pyramids with linear standard deviations.

Default values:

- 'nb_scales=4', 'base_scale=0' and 'detail_scale=0'.



Example 368 : `image.jpg split_details ,`

2.8.88 *structuretensors* (+)

Arguments:

- `_scheme={ 0=centered | 1=forward/backward }`

Compute the structure tensor field of selected images.

Default value:

- 'scheme=1'.



Example 369 : `image.jpg structuretensors abs pow 0.2`

Tutorial page:

https://gmic.eu/tutorial/_structuretensors.shtml

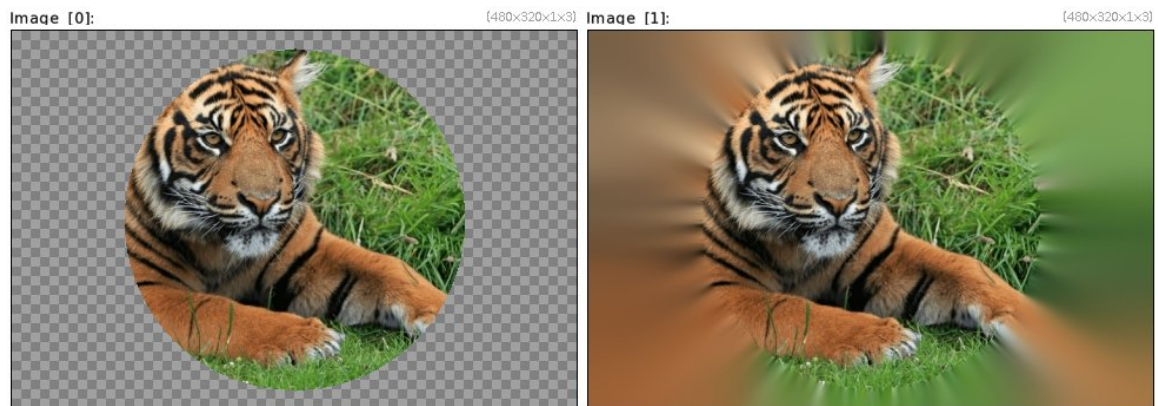
2.8.89 *solidify***Arguments:**

- `_smoothness[%]>=0, _diffusion-type={ 0=isotropic | 1=delaunay-oriented | 2=edge-oriented }, _diffusion_iter>=0`

Solidify selected transparent images.

Default values:

- `'smoothness=75%', 'diffusion_type=1' and 'diffusion_iter=20'.`



Example 370 : `image.jpg 100%,100% circle[-1] 50%,50%,25%,1,255 append c +solidify , display_rgba`

2.8.90 *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 371 : `image.jpg crop 2,282,50,328 +syntexturize 320,320`

2.8.91 *syntexturize matchpatch*

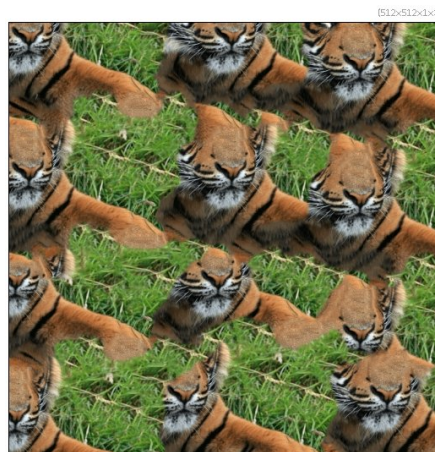
Arguments:

- `_width[%]>0, _height[%]>0, _nb_scales>=0, _patch_size>0, _blending_size>=0, _precision>=0`

Resynthesize 'width'x'height' versions of selected micro-textures using a patch-matching algorithm. If 'nbscales==0', the number of scales used is estimated from the image size.

Default values:

- `'width=height=100%', 'nb_scales=0', 'patch_size=7', 'blending_size=5' and 'precision=1'.`



Example 372 : `image.jpg crop 25%,25%,75%,75% syntexturize.matchpatch 512,512`

2.8.92 *tv_flow*

Arguments:

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

Apply iterations of the total variation flow on selected images.

Default values:

- `'nb_iter=10', 'dt=30' and 'keep_sequence=0'.`



Example 373 : `image.jpg +tv_flow 40`

2.8.93 *unsharp*

Arguments:

- `radius[%]>=0, _amount>=0, _threshold[%]>=0`

Apply unsharp mask on selected images.

Default values:

- `'amount=2' and 'threshold=0'.`



Example 374 : `image.jpg blur 3 +unsharp 1.5,15 cut 0,255`

2.8.94 *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 375 : `image.jpg blur 3 +unsharp_octave 4,5,15 cut 0,255`

2.8.95 *vanvliet (+)*

Arguments:

- `std_deviation>=0[%], order={ 0 | 1 | 2 | 3 }, axis={ x | y | z | c }, _boundary_conditions`

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

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

Default value:

- `'boundary_conditions=1'.`



Example 376 : `image.jpg +vanvliet 3,1,x`

[480x320x1x3]

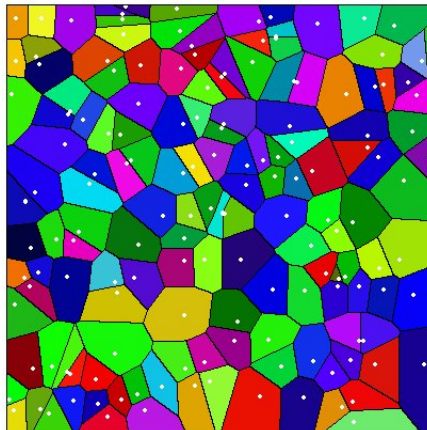


Example 377 : `image.jpg +vanvliet 30,0,x vanvliet[-2] 30,0,y add`

2.8.96 *voronoi*

Compute the discrete Voronoi diagram of non-zero pixels in selected images.

[400x400x1x3]



Example 378 : `400,400 noise 0.2,2 eq 1 +label_fg 0 voronoi[-1] +gradient[-1] xy,1
append[-2,-1] c norm[-1] ==[-1] 0 map[-2] 2,2 mul[-2,-1] normalize[-2] 0,255
dilate_circ[-2] 4 reverse max`

2.8.97 *watermark_fourier*

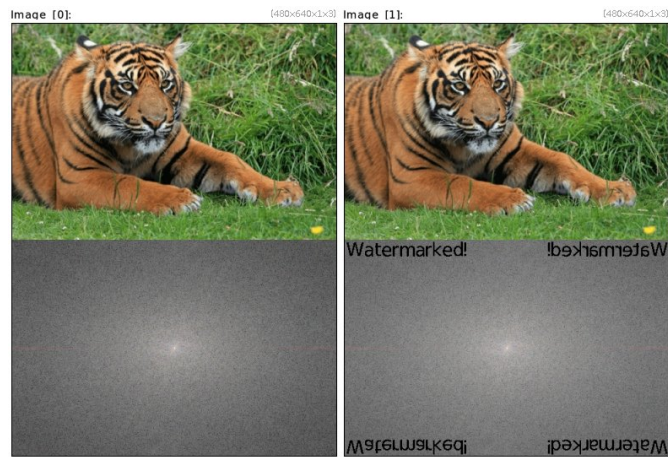
Arguments:

- `text,_size>0`

Add a textual watermark in the frequency domain of selected images.

Default value:

- `'size=33'`.



Example 379 : `image.jpg +watermark-fourier "Watermarked!" +display-fft remove[-3,-1] normalize 0,255 append[-4,-2] y append[-2,-1] y`

2.8.98 *watershed* (+)

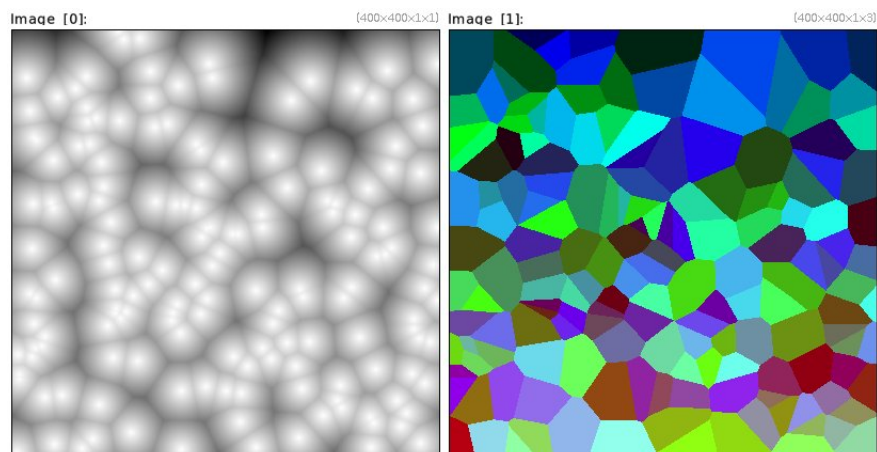
Arguments:

- `[priority_image],_is_high_connectivity={ 0 | 1 }`

Compute the watershed transform of selected images.

Default value:

- `'is_high_connectivity=1'`.



Example 380 : `400,400 noise 0.2,2 eq 1 +distance 1 mul[-1] -1 label[-2] watershed[-2] [-1] mod[-2] 256 map[-2] 0 reverse`

2.9 Features Extraction

2.9.1 *area*

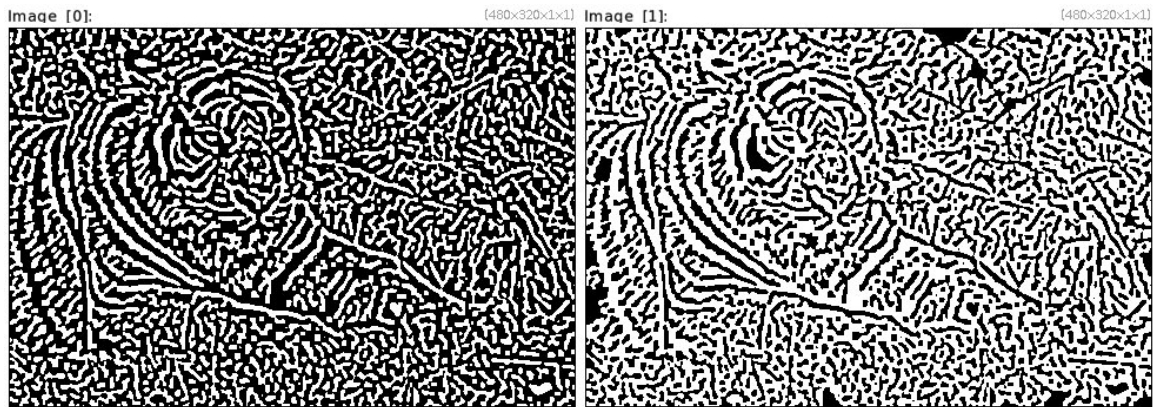
Arguments:

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

Compute area of connected components in selected images.

Default values:

- `'is_high_connectivity=0'`.



Example 381 : `image.jpg luminance stencil[-1] 1 +area 0`

Tutorial page:

https://gmics.eu/tutorial/_area.shtml

2.9.2 *area_fg*

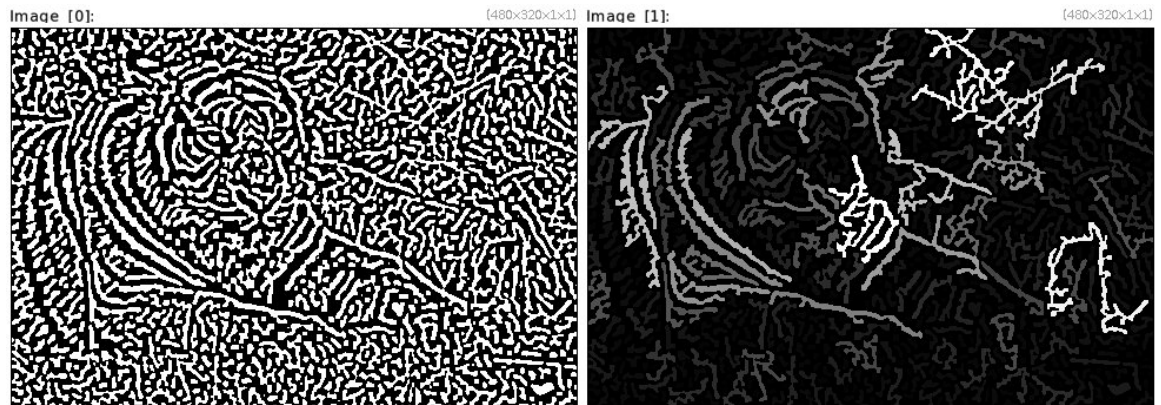
Arguments:

- `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 382 : `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 (x0,y0,z0)-(x1,y1,z1).



Example 383 : `image.jpg +at_line 0,0,0,100%,100%,0`

2.9.4 *at_quadrangle*

Arguments:

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

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

'interpolation' can be { 0=nearest-neighbor | 1=linear | 2=cubic }.

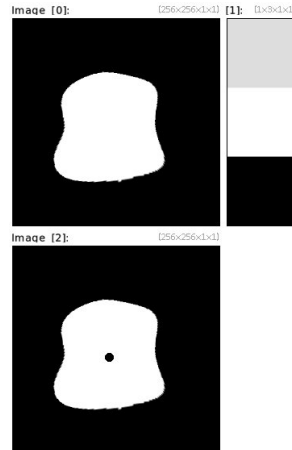
'boundary_conditions' can be { 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }.



Example 384 : `image.jpg params=5%,5%,95%,5%,60%,95%,40%,95% +at_quadrangle $params polygon..4,$params,0.5,255`

2.9.5 *barycenter*

Compute the barycenter vector of pixel values.



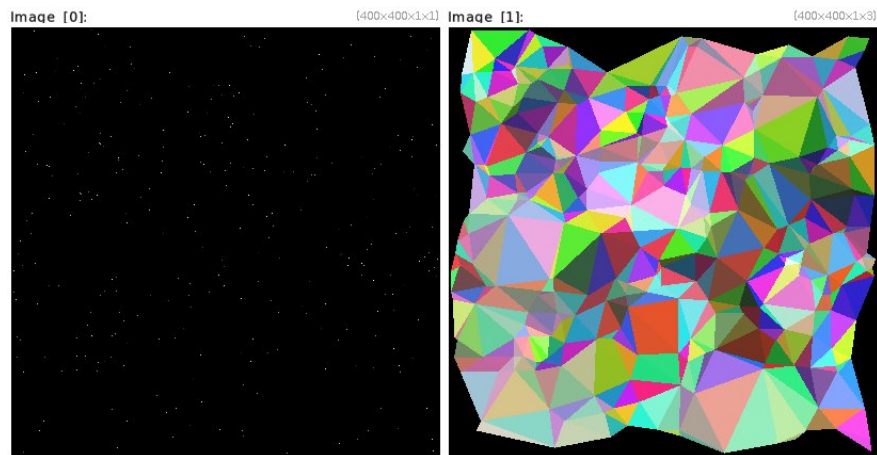
Example 385 : `256,256 ellipse 50%,50%,20%,20%,0,1,1 deform 20 +barycenter +ellipse[-2]{@0,1},5,5,0,10`

2.9.6 *delaunay*

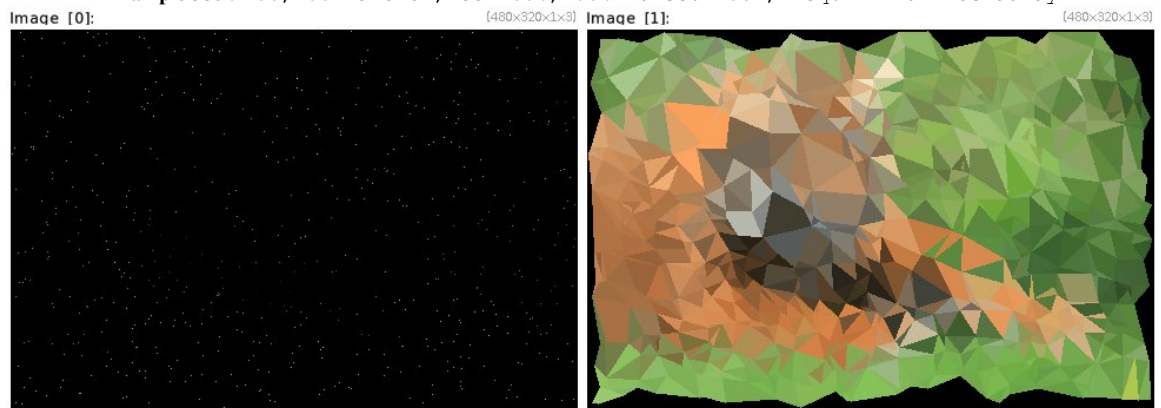
Generate discrete 2D Delaunay triangulation of non-zero pixels in selected images.

Input images must be scalar.

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



Example 386 : 400,400 rand 32,255 100%,100% noise. 0.4,2 eq. 1 mul +delaunay



Example 387 : image.jpg b 1% 100%,100% noise. 0.8,2 eq. 1 mul +delaunay channels 0,2

2.9.7 *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'.`

2.9.8 *displacement (+)*

Arguments:

```

• [source_image],_smoothness,_precision>=0,_nb_scales>=0,_iteration_max>=0,
is_backward={ 0 | 1
},_[guide]

```

Estimate displacement field between specified source and selected target images.

If 'smoothness>=0', regularization type is set to isotropic, else to anisotropic.

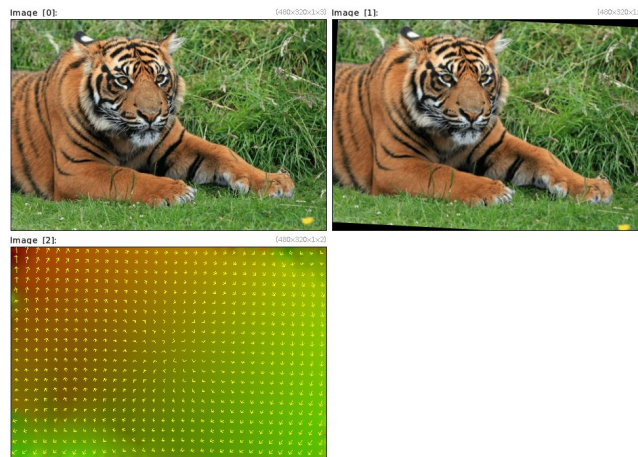
If 'nbscales==0', the number of scales used is estimated from the image size.

Default values:

```

• 'smoothness=0.1', 'precision=5', 'nb_scales=0', 'iteration_max=10000',
'is_backward=1' and '[guide]=(unused)'.

```



Example 388 : `image.jpg +rotate 3,1,0,50%,50% +displacement[-1] [-2] quiver[-1] [-1],15,1,1,1,{1.5*iM}`

2.9.9 distance (+)

Arguments:

- `isovalue[%],_metric`
- `isovalue[%],[metric],_method`

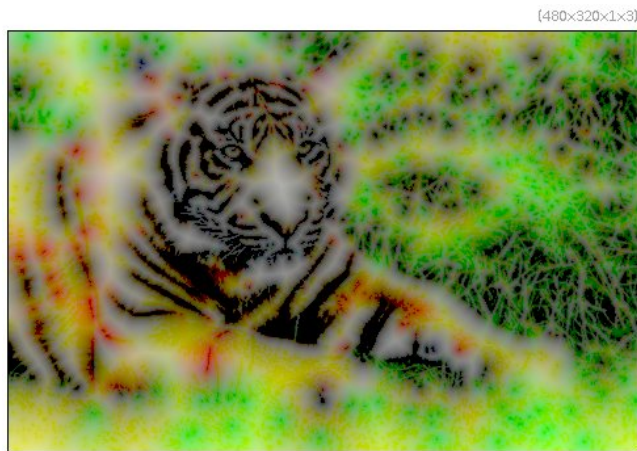
Compute the unsigned distance function to specified isovalue, opt. according to a custom metric.

'metric' can be { 0=chebyshev | 1=manhattan | 2=euclidean | 3=squared-euclidean }.

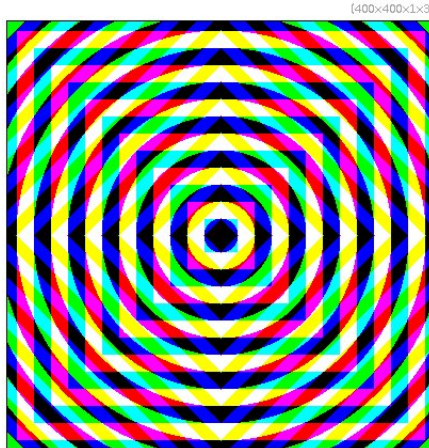
'method' can be { 0=fast-marching | 1=low-connectivity dijkstra | 2=high-connectivity dijkstra | 3=1+return path | 4=2+return path }.

Default value:

- 'metric=2' and 'method=0'.



Example 389 : `image.jpg threshold 20% distance 0 pow 0.3`



Example 390 : `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:

https://gmic.eu/tutorial/_distance.shtml

2.9.10 *fftpolar*

Compute fourier transform of selected images, as centered magnitude/phase images.



Example 391 : `image.jpg fftpolar ellipse 50%,50%,10,10,0,1,0 ifftpolar`

2.9.11 *histogram (+)*

Arguments:

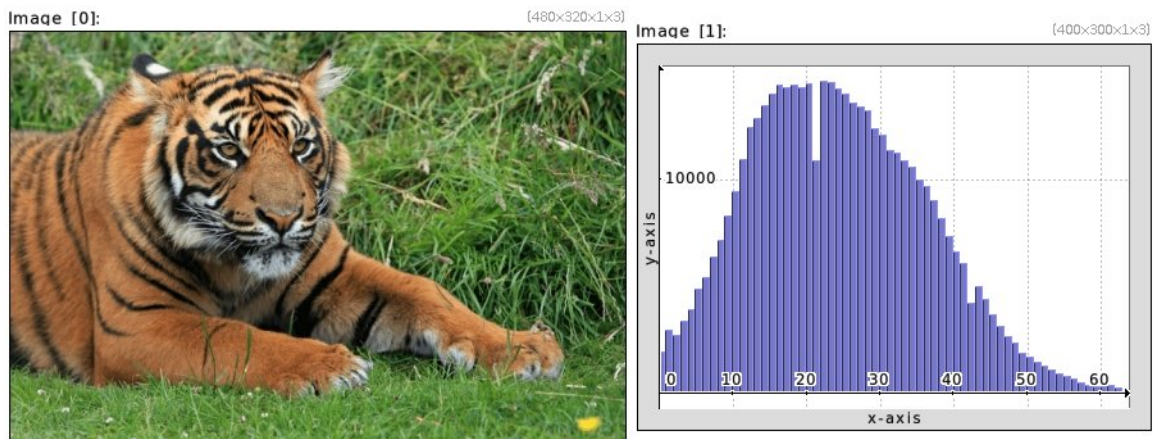
- `_nb_levels>0[%],_value0[%],_value1[%]`

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 392 : `image.jpg +histogram 64 display_graph[-1] 400,300,3`

2.9.12 *histogram_nd*

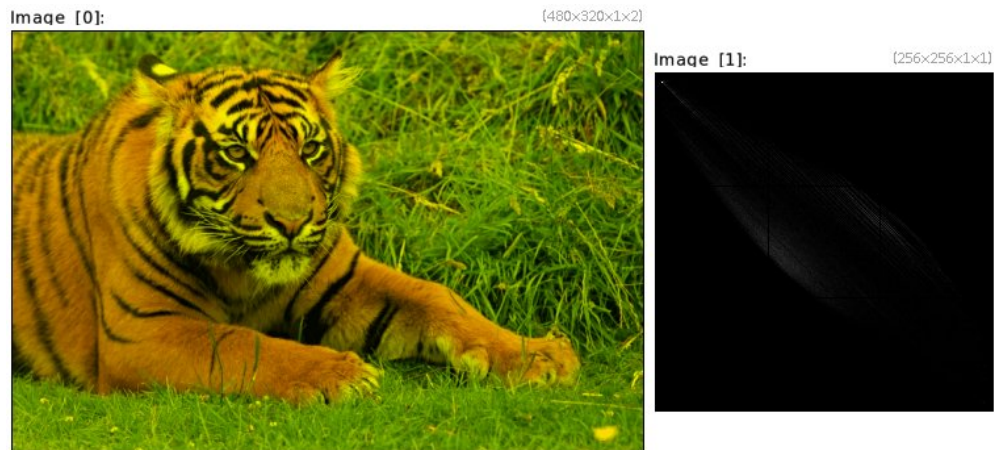
Arguments:

- `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 393 : `image.jpg channels 0,1 +histogram.nd 256`

2.9.13 *histogram_cumul*

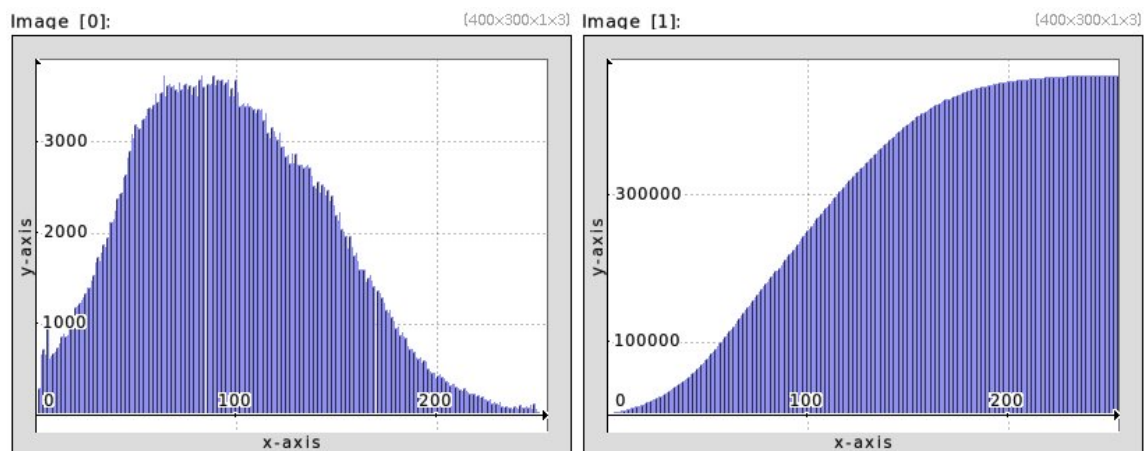
Arguments:

- `_nb_levels>0,_is_normalized={ 0 | 1 },_val0[%],_val1[%]`

Compute cumulative histogram of selected images.

Default values:

- `'nb_levels=256', 'is_normalized=0', 'val0=0%' and 'val1=100%'`.



Example 394 : `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*

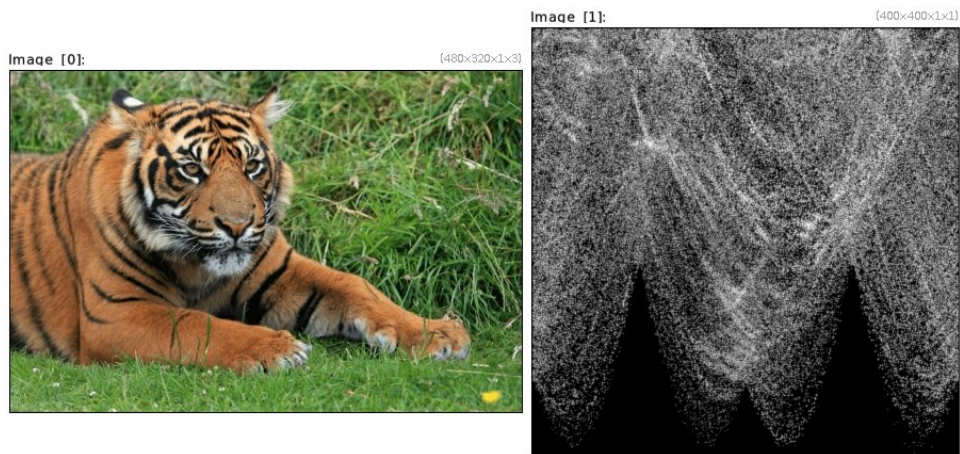
Arguments:

- `_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 395 : `image.jpg +blur 1.5 hough[-1] 400,400 blur[-1] 0.5 add[-1] 1 log[-1]`

2.9.16 *iftpolar*

Compute inverse fourier transform of selected images, from centered magnitude/phase images.

2.9.17 *isophotes*

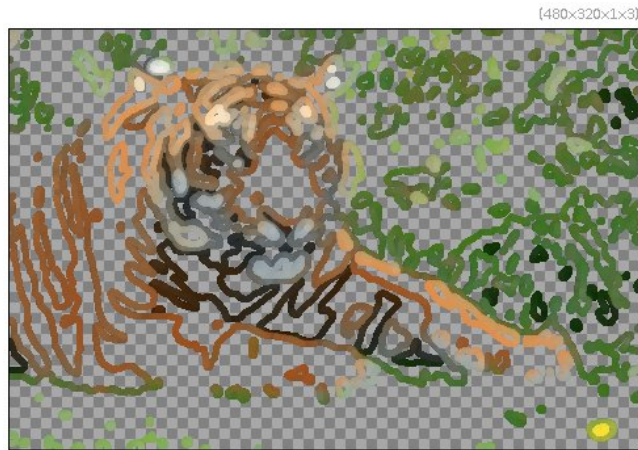
Arguments:

- `_nb_levels>0`

Render isophotes of selected images on a transparent background.

Default value:

- `'nb_levels=64'`



Example 396 : `image.jpg blur 2 isophotes 6 dilate_circ 5 display_rgba`

2.9.18 *label* (+)

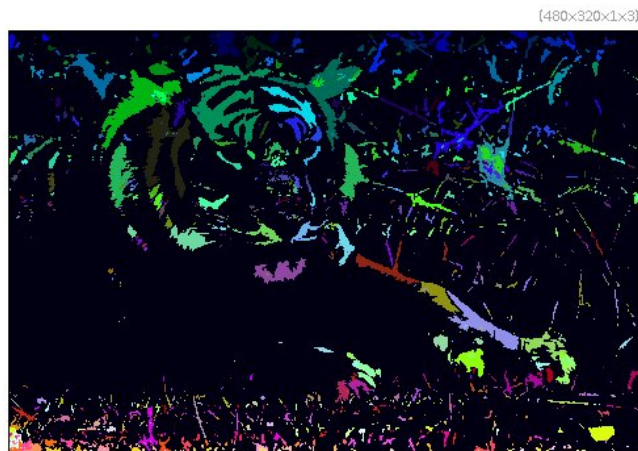
Arguments:

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

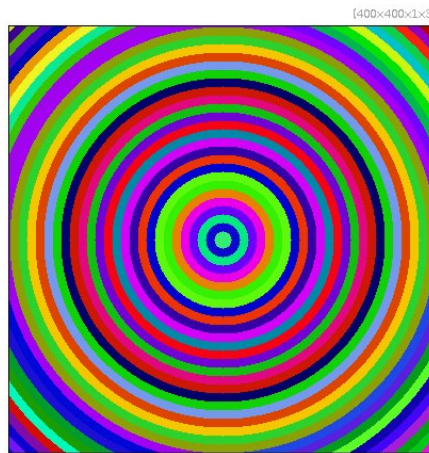
Label connected components in selected images.

Default values:

- `'tolerance=0' and 'is_high_connectivity=0'.`



Example 397 : `image.jpg luminance threshold 60% label normalize 0,255 map 0`



Example 398 : 400,400 set 1,50%,50% distance 1 mod 16 threshold 8 label mod 255 map 2

Tutorial page:

https://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 399 : `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 400 : `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:

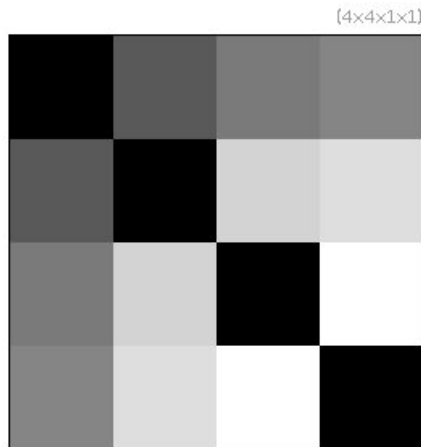
- `'is_high_connectivity=0'`.



Example 401 : `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.

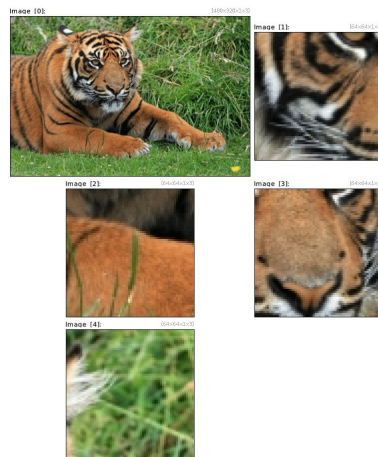


Example 402 : `image.jpg +noise 30 +noise[0] 35 +noise[0] 38 cut. 0,255 mse`

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 403 : `image.jpg +patches 64,64,1,153,124,0,184,240,0,217,126,0,275,38,0`

2.9.25 *matchpatch* (+)

Arguments:

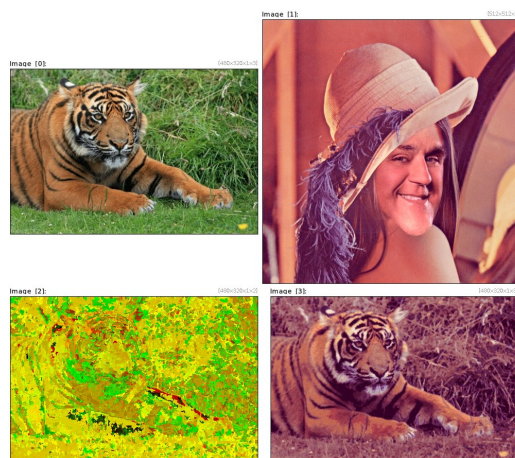
- `[patch_image], patch_width>=1, patch_height>=1, patch_depth>=1, nb_iterations>=0, nb_randoms>=0, occ_penalization, output_score={ 0 | 1 },-[guide]`

Estimate correspondence map between selected images and specified patch image, using a patch-matching algorithm.

Each pixel of the returned correspondence map gives the location (p,q) of the closest patch in the specified patch image. If 'output_score=1', the third channel also gives the corresponding matching score for each patch as well.

Default values:

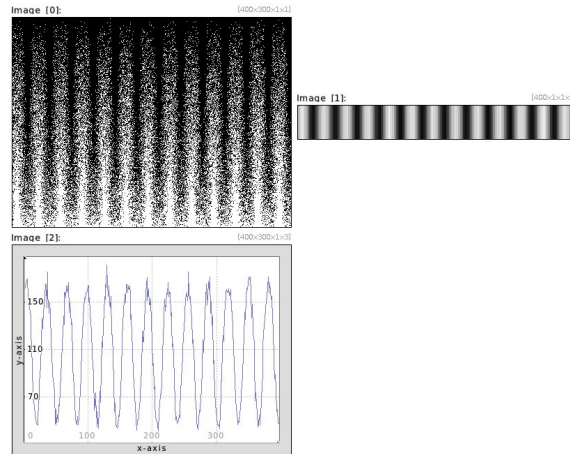
- `'patch_height=patch_width', 'patch_depth=1', 'nb_iterations=5', 'nb_randoms=5', 'occ_penalization=0', 'output_score=0' and 'guide=(undefined)'`.



Example 404 : `image.jpg sample ? to-rgb +matchpatch[0] [1],3 +warp[-2] [-1],0`

2.9.26 *plot2value*

Retrieve values from selected 2D graph plots.



Example 405 : `400,300,1,1,'if(y>300*abs(cos(x/10+2*u)),1,0)'+plot2value+display-graph[-1]`
`400,300`

2.9.27 *pointcloud*

Arguments:

```
• .type = { -X=-X-opacity | 0=binary | 1=cumulative | 2=label
| 3=retrieve coordinates },_width,_height>0,_depth>0
```

Render a set of point coordinates, as a point cloud in a 1D/2D or 3D binary image (or do the reverse, i.e. retrieve coordinates of non-zero points from a rendered point cloud).

Input point coordinates can be a $N \times M \times 1 \times 1$, $N \times 1 \times 1 \times M$ or $1 \times N \times 1 \times M$ image, where 'N' is the number of points, and M the point coordinates.

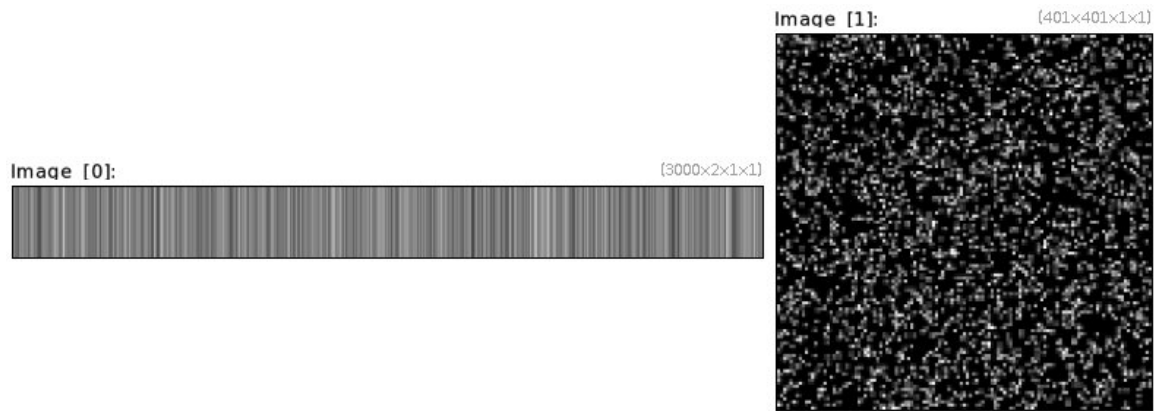
If 'M' > 3, the 3-to-M components sets the (M-3)-dimensional color at each point.

Parameters 'width', 'height' and 'depth' are related to the size of the final image : - If set to 0, the size is automatically set along the specified axis. - If set to $N > 0$, the size along the specified axis is N. - If set to $N < 0$, the size along the specified axis is at most N.

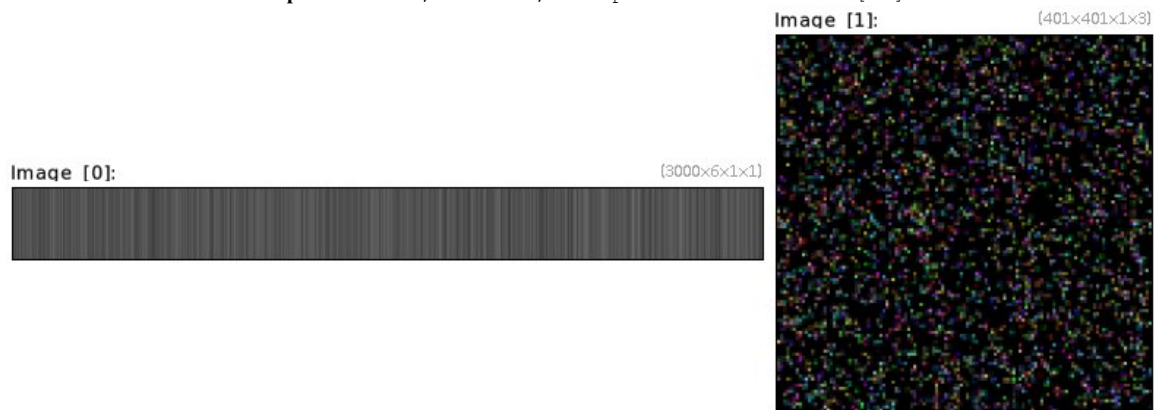
Points with coordinates that are negative or higher than specified ('width', 'height', 'depth') are not plotted.

Default values:

- 'type=0' and 'max_width=max_height=max_depth=0'.



Example 406 : `3000,2 rand 0,400 +pointcloud 0 dilate[-1] 3`



Example 407 : `3000,2 rand 0,400 {w} {w},3 rand[-1] 0,255 append y +pointcloud 0 dilate[-1] 3`

2.9.28 *psnr*

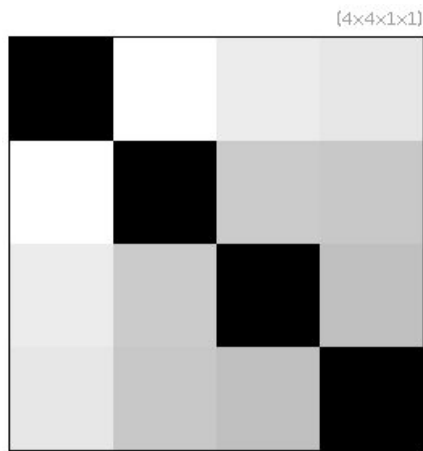
Arguments:

- `_max_value`

Compute PSNR (Peak Signal-to-Noise Ratio) matrix between selected images.

Default value:

- `'max_value=255'`.



Example 408 : `image.jpg +noise 30 +noise[0] 35 +noise[0] 38 cut[-1] 0,255 psnr 255 replace.inf 0`

2.9.29 *segment_watershed*

Arguments:

- `_threshold>=0`

Apply watershed segmentation on selected images.

Default values:

- `'threshold=2'.`



Example 409 : `image.jpg segment-watershed 2`

2.9.30 *shape2bump*

Arguments:

- `_resolution>=0, 0<=_weight_avg_max_avg<=1, _dilation, _smoothness>=0`

Estimate bumpmap from binary shape in selected images.

Default value:

- `'resolution=256', 'weight_avg_max=0.75', 'dilation=0' and 'smoothness=100'.`

2.9.31 *skeleton*

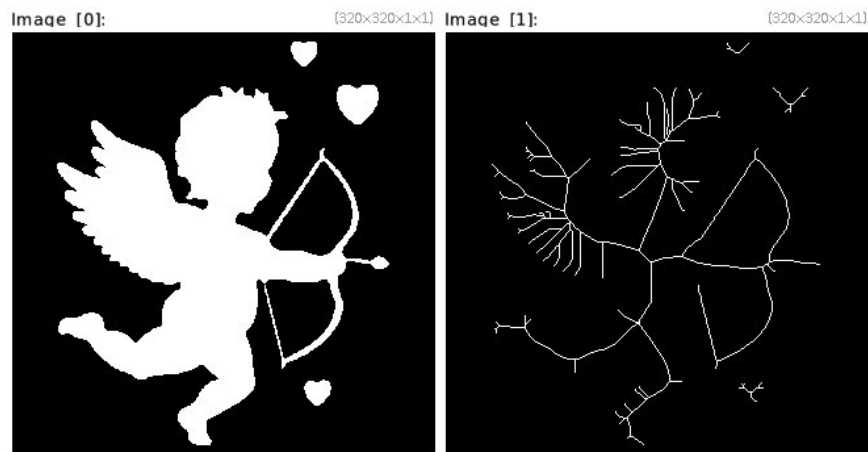
Arguments:

- `_boundary_conditions={ 0=dirichlet | 1=neumann }`

Compute skeleton of binary shapes using distance transform and constrained thinning.

Default value:

- `'boundary_conditions=1'.`



Example 410 : `shape.cupid 320 +skeleton 0`

2.9.32 *slic*

Arguments:

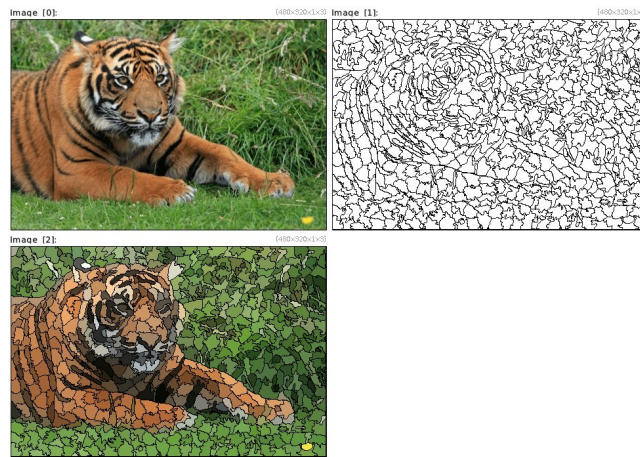
- `size>0, _regularity>=0, _nb_iterations>0`

Segment selected 2D images with superpixels, using the SLIC algorithm (Simple Linear Iterative Clustering). Scalar images of increasingly labeled pixels are returned.

Reference paper: Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Ssstrunk, S. (2010). Slic superpixels (No. EPFL-REPORT-149300).

Default values:

- `'size=16', 'regularity=10' and 'nb_iterations=10'.`



Example 411 : `image.jpg +srgb2lab slic[-1] 16 +blend shapeaverage f[-2] "j(1,0)==i && j(0,1)==i" *[-1] [-2]`

2.9.33 *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 412 : `image.jpg +crop 20%,20%,35%,35% +ssd_patch[0] [1],0,0`

2.9.34 *thinning*

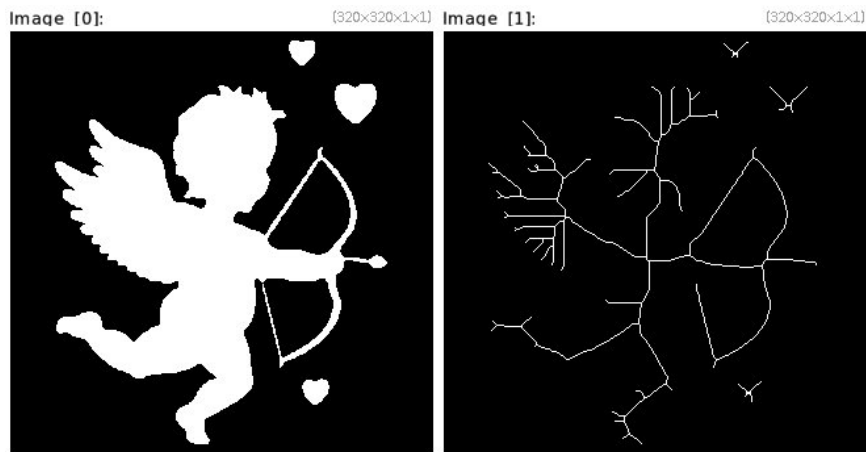
Arguments:

- `_boundary_conditions={ 0=dirichlet | 1=neumann }`

Compute skeleton of binary shapes using morphological thinning
(beware, this is a quite slow iterative process)

Default value:

- `'boundary_conditions=1'`.



Example 413 : `shape_cupid 320 +thinning`

2.9.35 *tones*

Arguments:

- `N>0`

Get N tones masks from selected images.



Example 414 : `image.jpg +tones 3`

2.9.36 *topographic_map*

Arguments:

- `_nb_levels>0, _smoothness`

Render selected images as topographic maps.

Default values:

- `'_nb_levels=16' and '_smoothness=2'.`



Example 415 : `image.jpg topographic_map 10`

2.9.37 *tsp*

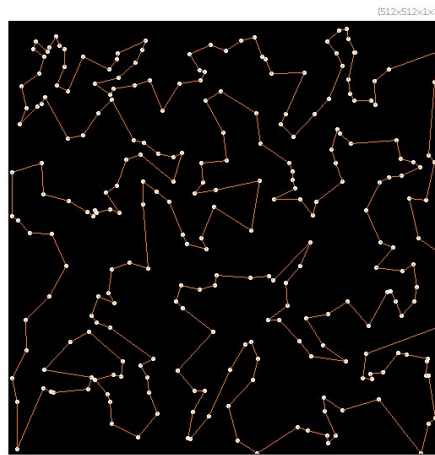
Arguments:

- `_precision>=0`

Try to solve the 'travelling salesman' problem, using a combination of greedy search and 2-opt algorithms. Selected images must have dimensions $N \times 1 \times 1 \times C$ to represent N cities each with C -dimensional coordinates. This command re-order the selected data along the x-axis so that the point sequence becomes a shortest path.

Default values:

- `'_precision=256'.`



Example 416 : `256,1,1,2 rand 0,512 tsp , 512,512,1,3 repeat {0,w} circle[-1]
 {0,I[$>]},2,1,255,255,255 line[-1] {0,boundary=2;[I[$>],I[$>+1]]},1,255,128,0 done
 keep[-1]`

2.9.38 *variance_patch*

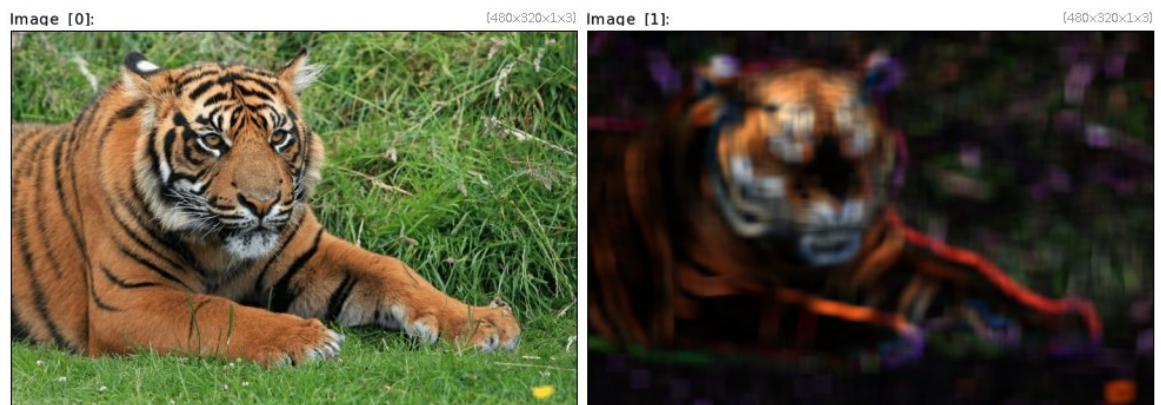
Arguments:

- `_patch.size>=1`

Compute variance of each images patch centered at (x,y), in selected images.

Default value:

- `'patch.size=16'`



Example 417 : `image.jpg +variance_patch`

2.10 Image Drawing

2.10.1 *arrow*

Arguments:

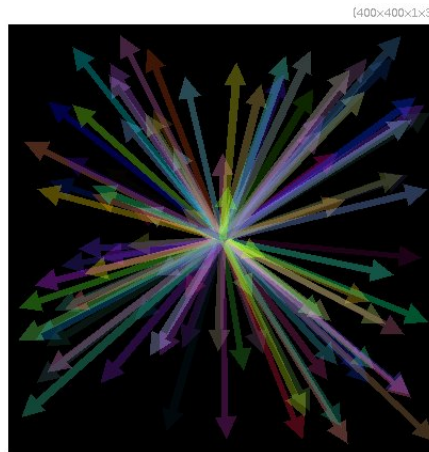
- `x0[%],y0[%],x1[%],y1[%],_thickness[%]>=0,_head.length[%]>=0,-_head.thickness[%]>=0,_opacity,_pattern,_color1,...`

Draw specified arrow on selected images.

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

Default values:

- `'thickness=1%', 'head.length=10%', 'head.thickness=3%', 'opacity=1', 'pattern=(undefined)' and 'color1=0'.`



Example 418 : `400,400,1,3 repeat 100 arrow 50%,50%,{u(100)}%,{u(100)}%,3,20,10,0.3,${-RGB} done`

2.10.2 axes

Arguments:

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

Draw xy-axes on selected images.

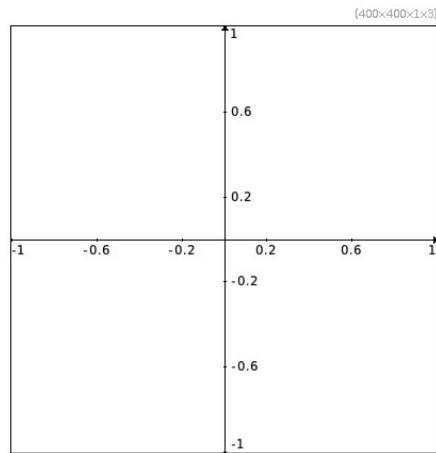
'pattern' is an hexadecimal number starting with '0x' which can be omitted even if a color is specified.

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 419 : `400,400,1,3,255 axes -1,1,1,-1`

2.10.3 *ball*

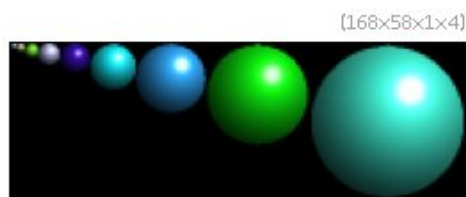
Arguments:

- `_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 420 : `repeat 9 ball {1.5^($>+2)}, ${-RGB} done append x`

2.10.4 *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 421 : `image.jpg chessboard 32,32,0,0,25,0.3,255,128,0,0,128,255`

2.10.5 *cie1931*

Draw CIE-1931 chromaticity diagram on selected images.



Example 422 : `500,400,1,3 cie1931`

2.10.6 *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 a 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 423: `image.jpg repeat 300 circle {u(100)}%,{u(100)}%,{u(30)},0.3,${-RGB} done circle 50%,50%,100,0.7,255`

2.10.7 *close binary*

Arguments:

- `0<=_endpoint_rate<=100,_endpoint_connectivity>=0,_spline_distmax>=0,_segment_distmax>=0,0<=_spline_anglemax<=180,_spline_roundness>=0,_area_min>=0,_allow_self_intersection={ 0 | 1 }`

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

Default values:

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

2.10.8 *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 a 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 424 : image.jpg repeat 300 ellipse
 {u(100)}%, {u(100)}%, {u(30)}, {u(30)}, {u(180)}, 0.3, \${-RGB} done ellipse
 50%, 50%, 100, 100, 0, 0.7, 255

2.10.9 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 425 : image.jpg repeat 1000 flood {u(100)}%, {u(100)}%, 0, 20, 0, 1, \${-RGB} done

2.10.10 *gaussian*

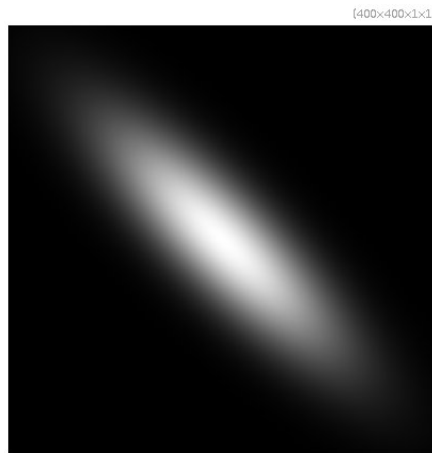
Arguments:

- `_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 426 : `400,400 gaussian 100,30,45`

Tutorial page:

https://gmic.eu/tutorial/_gaussian.shtml

2.10.11 *graph (+)*

Arguments:

- `[function.image], _plot.type, _vertex.type, _ymin, _ymax, _opacity, _pattern, _color1, ...`
- `'formula', _resolution>=0, _plot.type, _vertex.type, _xmin, _xmax, _ymin, _ymax, _opacity, _pattern, _color1, ...`

Draw specified function graph on selected images.

'plot.type' can be { 0=none | 1=lines | 2=splines | 3=bar }.

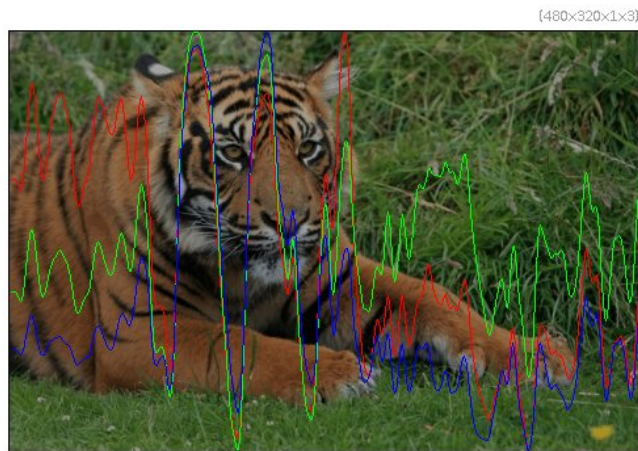
'vertex.type' can be { 0=none | 1=points | 2,3=crosses | 4,5=circles | 6,7=squares }.

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

Default values:

- `'plot.type=1', 'vertex.type=1', 'ymin=ymax=0 (auto)', 'opacity=1', 'pattern=(undefined)'`

and `'color1=0'.`



Example 427: `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]`

2.10.12 *grid*

Arguments:

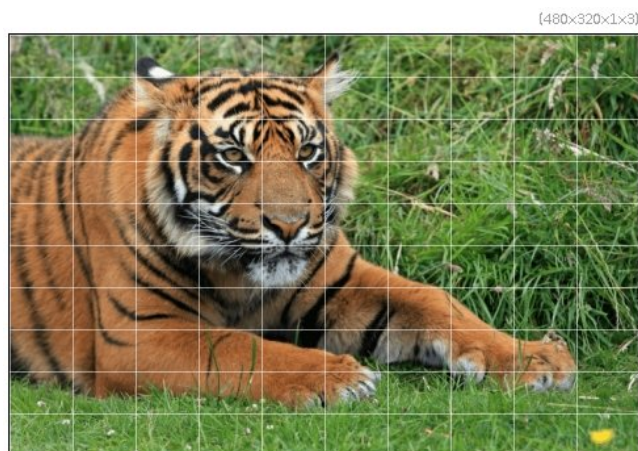
- `size_x[%]>=0,size_y[%]>=0,_offset_x[_%],_offset_y[_%],_opacity,_pattern,_color1,
 ...`

Draw xy-grid on selected images.

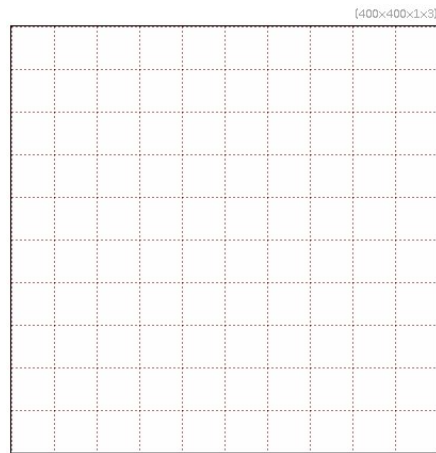
'pattern' is an hexadecimal number starting with '0x' which can be omitted even if a color is specified.

Default values:

- `'offset_x=offset_y=0', 'opacity=1', 'pattern=(undefined)' and 'color1=0'.`



Example 428: `image.jpg grid 10%,10%,0,0,0.5,255`



Example 429 : `400,400,1,3,255 grid 10%,10%,0,0,0.3,0xCCCCCCCC,128,32,16`

2.10.13 *image* (+)

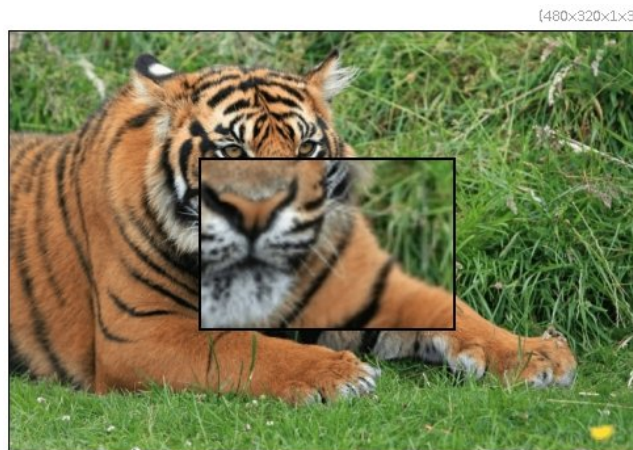
Arguments:

- `[sprite],_x[%],_y[%],_z[%],_c[%],_opacity,-[opacity_mask],_max_opacity_mask`

Draw specified sprite image on selected images.
(eq. to 'j').

Default values:

- `'x=y=z=c=0', 'opacity=1', 'opacity_mask=(undefined)' and 'max_opacity_mask=1'.`



Example 430 : `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]`

2.10.14 *line* (+)

Arguments:

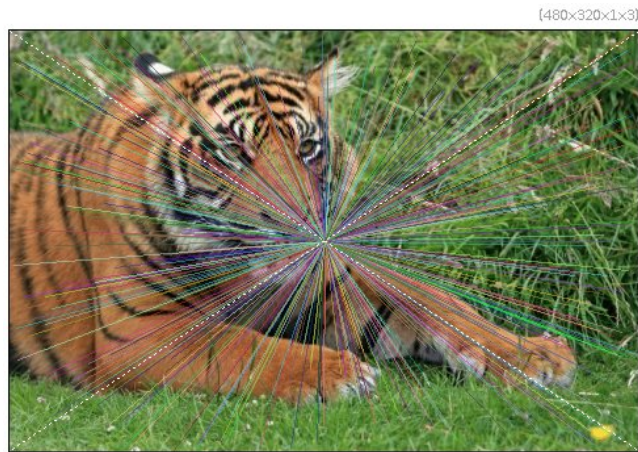
- `x0[%],y0[%],x1[%],y1[%],_opacity,_pattern,_color1,...`

Draw specified colored line on selected images.

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

Default values:

- `'opacity=1', 'pattern=(undefined)' and 'color1=0'.`



Example 431 : `image.jpg repeat 500 line 50%,50%,{u(w)},{u(h)},0.5,{-RGB} done line 0,0,100%,100%,1,0xCCCCCCCC,255 line 100%,0,0,100%,1,0xCCCCCCCC,255`

2.10.15 *linethick*

Arguments:

- `x0[%],y0[%],x1[%],y1[%],_thickness,_opacity,_color1`

Draw specified colored thick line on selected images.

Default values:

- `'thickness=2', 'opacity=1' and 'color1=0'.`



Example 432 : `400,400,1,3 repeat 100 linethick {u([w,h,w,h,5])},0.5,{-RGB} done`

2.10.16 *mandelbrot* (+)

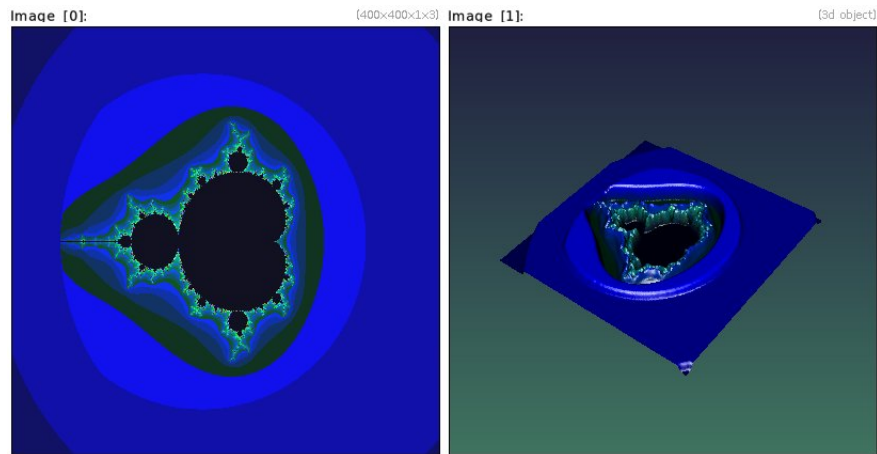
Arguments:

- `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 433 : `400,400 mandelbrot -2.5,-2,2,2,1024 map 0 +blur 2 elevation3d[-1] -0.2`

2.10.17 *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'.`



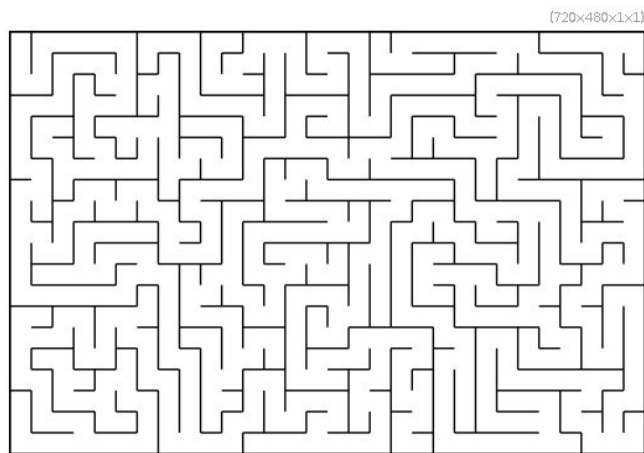
Example 434 : `image.jpg +marble ,`

2.10.18 *maze*

Arguments:

- `_width>0, _height>0, _cell.size>0`

Input maze with specified size.



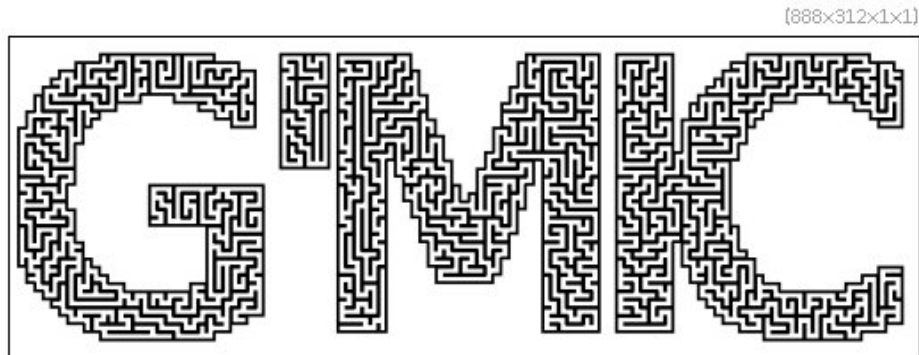
Example 435 : `maze 30,20 negate normalize 0,255`

2.10.19 *maze_mask*

Arguments:

- `_cellsize>0`

Input maze according to size and shape of selected mask images.
Mask may contain disconnected shapes.



Example 436 : `0 text "G'MIC",0,0,53,1,1 dilate 3 autocrop 0 frame 1,1,0 maze_mask 8 dilate 3
negate mul 255`

2.10.20 *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'*) .\n).

'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 437 : `image.jpg torus3d 100,10 cone3d 30,-120 add3d[-2,-1] rotate3d. 1,1,0,60
object3d[0] [-1],50%,50% keep[0]`

2.10.21 *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.

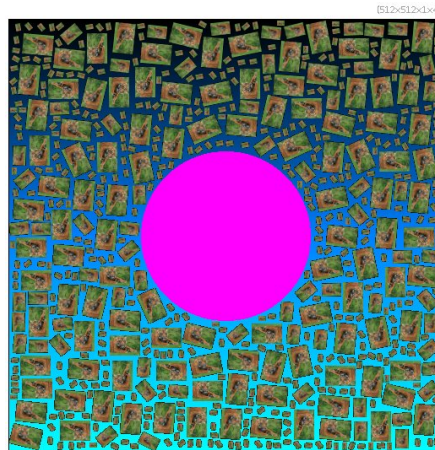
'min_scale' (in %) sets the minimal size considered for packing (specified as a percentage of the original sprite size).

'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'`.



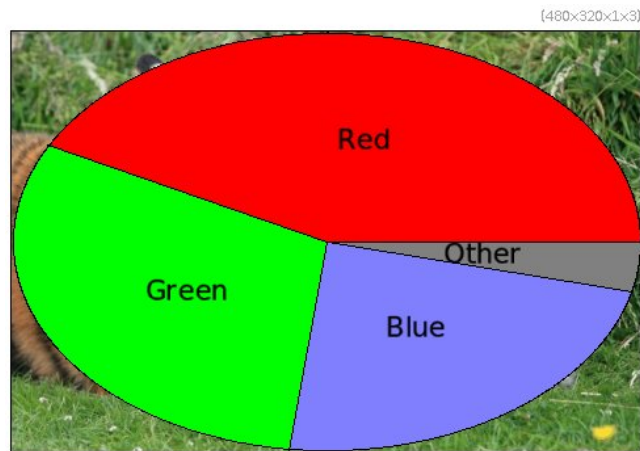
Example 438 : `512,512,1,3,"min(255,y*c/2)" 100%,100% circle 50%,50%,100,1,255 append c
image.jpg resize2dy[-1] 24 to_rgba pack_sprites 3,25`

2.10.22 *piechart*

Arguments:

- `label.height>=0, label_R, label_G, label_B, "label1", value1, R1, G1, B1, ..., "labelN", valueN, RN, GN, BN`

Draw pie chart on selected (RGB) images.



Example 439 : `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`

2.10.23 *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 440 : `400,400,1,3 plasma`

Tutorial page:

https://gmic.eu/tutorial/_plasma.shtml

2.10.24 *point* (+)

Arguments:

- `x[%],y[%],z[%],_opacity,_color1,...`

Set specified colored pixel on selected images.

Default values:

- `'z=0', 'opacity=1' and 'color1=0'.`



Example 441 : `image.jpg repeat 10000 point {u(100)}%,{u(100)}%,0,1,${-RGB} done`

2.10.25 *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 442 : `image.jpg polka_dots 10,15,0,0,20,10,1,0.5,0,128,255`

2.10.26 *polygon* (+)

Arguments:

- `N>=1,x1[%],y1[%],...,xN[%],yN[%],_opacity,_pattern,_color1,...`

Draw specified colored N-vertices polygon on selected images.

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

Default values:

- `'opacity=1', 'pattern=(undefined)' and 'color1=0'.`



Example 443 : `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`



Example 444 : `image.jpg 2,16,1,1,'u(if(x,{h},{w}))' polygon[-2] {h},{^},0.6,255,0,255
remove[-1]`

2.10.27 *quiver*

Arguments:

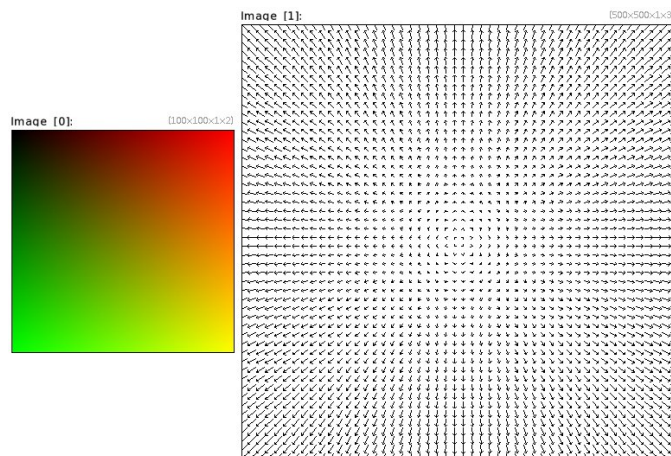
- `[function_image],_sampling[%]>0,_factor>=0,_is_arrow={ 0 | 1 },_opacity,_color1,...`

Draw specified 2D vector/orientation field on selected images.

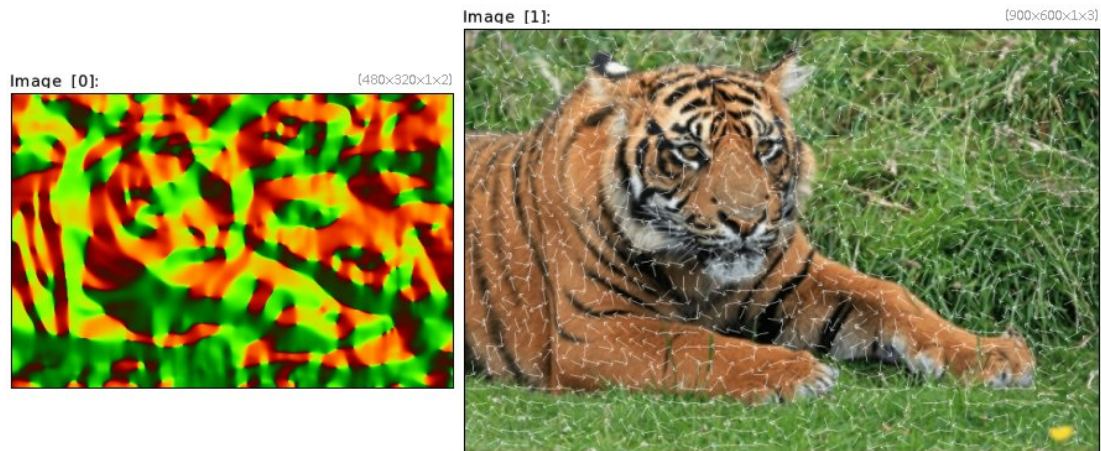
Default values:

- `'sampling=5%','factor=1','is_arrow=1','opacity=1','pattern=(undefined)'`

and `'color1=0'`.



Example 445 : `100,100,1,2,'if(c==0,x-w/2,y-h/2)' 500,500,1,3,255 quiver[-1] [-2],10`



Example 446 : `image.jpg +resize2dy 600 luminance[0] gradient[0] mul[1] -1 reverse[0,1]
append[0,1] c blur[0] 8 orientation[0] quiver[1] [0],20,1,1,0.8,255`

2.10.28 *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 447 : `image.jpg repeat 30 rectangle
{u(100)}%,{u(100)}%,{u(100)}%,{u(100)}%,0.3,${-RGB} done`

2.10.29 *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 448 : 400,400 rorschach 3%

2.10.30 *sierpinski*

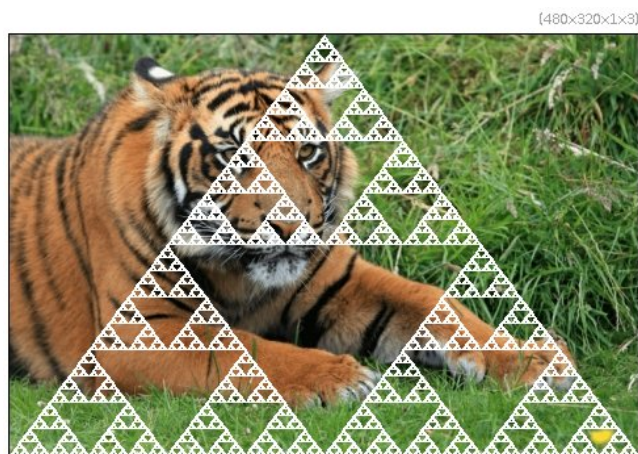
Arguments:

- `recursion.level>=0`

Draw Sierpinski triangle on selected images.

Default value:

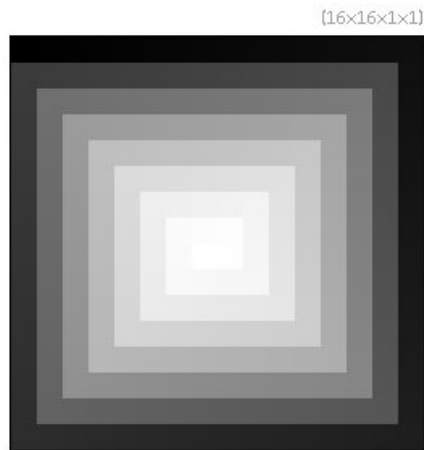
- `'recursion.level=7'.`



Example 449 : image.jpg sierpinski 7

2.10.31 *spiralbw*

Draw (squared) spiral on selected images.



Example 450 : 16,16 spiralbw

2.10.32 *spline*

Arguments:

- `x0[%],y0[%],u0[%],v0[%],x1[%],y1[%],u1[%],v1[%],_opacity,_color1,...`

Draw specified colored spline curve on selected images (cubic hermite spline).

Default values:

- `'opacity=1'` and `'color1=0'` .



Example 451 : `image.jpg repeat 30 spline {u(100)}%,{u(100)}%,{u(-600,600)},{u(-600,600)},
{u(100)}%,{u(100)}%,{u(-600,600)},{u(-600,600)},0.6,255
done`

2.10.33 *tetraedron_shade*

Arguments:

- `x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3,R0,G0,B0,...,R1,G1,B1,...,R2,G2,B2,...,R3,G3,B3,...`

Draw tetraedron with interpolated colors on selected (volumetric) images.

2.10.34 *text (+)*

Arguments:

- `text, _x[%], _y[%], _font_height[%]>=0, _opacity, _color1, ...`

Draw specified colored text string on selected images.

(*eq. to 't'*) .\n).

Sizes '13' and '128' are special and correspond to binary fonts (no-antialiasing).

Any other font size is rendered with anti-aliasing.

Specifying an empty target image resizes it to new dimensions such that the image contains the entire text string.

Default values:

- `'opacity=1' and 'color1=0'.`



Example 452: `image.jpg resize2dy 600 y=0 repeat 30 text {2*$>} " : This is a nice text, isn't it ?", 10, $y, {2*$>}, 0.9, 255 y+={2*$>} done`



Example 453 : `0 text "G'MIC",0,0,23,1,255`

2.10.35 *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=1%', 'font_height=7.5%', 'outline=2', 'opacity=1' and 'color1=255'.`



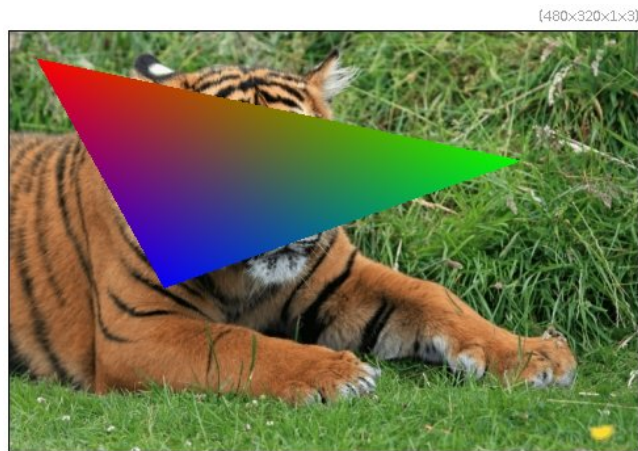
Example 454 : `image.jpg text_outline "Hi there!",10,10,63,3`

2.10.36 *triangle_shade*

Arguments:

- `x0, y0, x1, y1, x2, y2, R0, G0, B0, ..., R1, G1, B1, ..., R2, G2, B2, ...`

Draw triangle with interpolated colors on selected images.



Example 455 : `image.jpg triangle.shade 20,20,400,100,120,200,255,0,0,0,255,0,0,0,255`

2.10.37 *truchet*

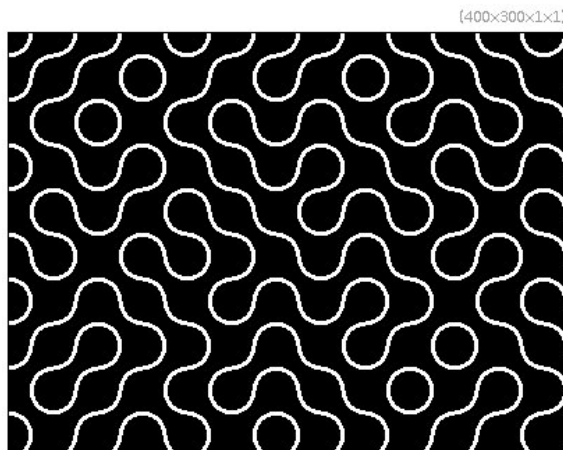
Arguments:

- `_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 456 : `400,300 truchet ,`

2.10.38 *turbulence*

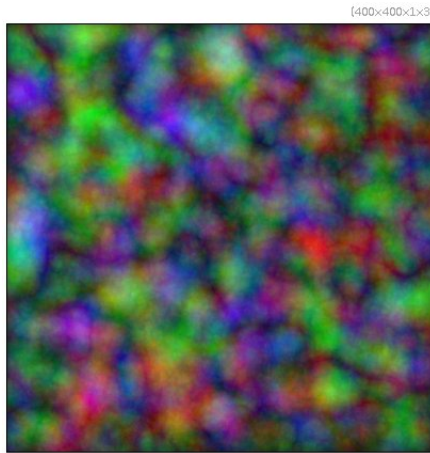
Arguments:

- `_radius>0, _octaves={1,2,3...,12}, _alpha>0, _difference={-10,10}, _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'.



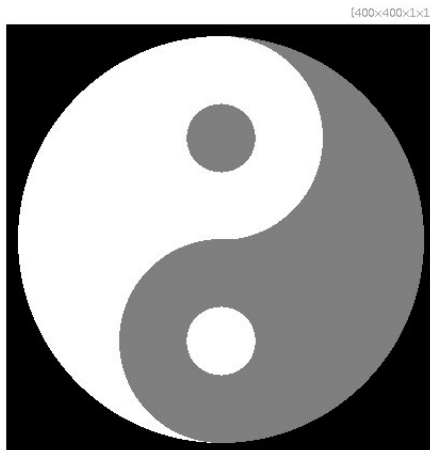
Example 457 : 400,400,1,3 turbulence 16

Tutorial page:

https://gmic.eu/tutorial/_turbulence.shtml

2.10.39 yinyang

Draw a yin-yang symbol on selected images.



Example 458 : 400,400 yinyang

2.11 Matrix Computation**2.11.1 dijkstra (+)****Arguments:**

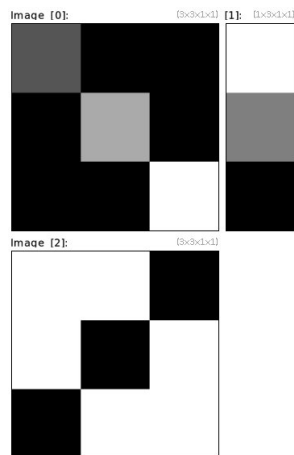
- `starting_node>=0,ending_node>=0`

Compute minimal distances and paths from specified adjacency matrices by the Dijkstra algorithm.

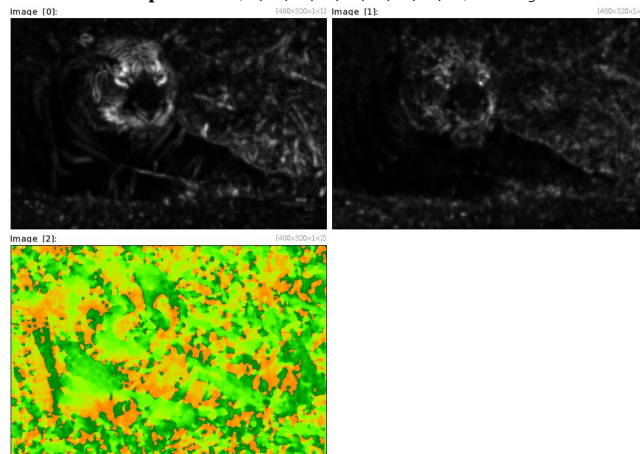
2.11.2 *eigen* (+)

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 459 : `(1,0,0;0,2,0;0,0,3) +eigen`



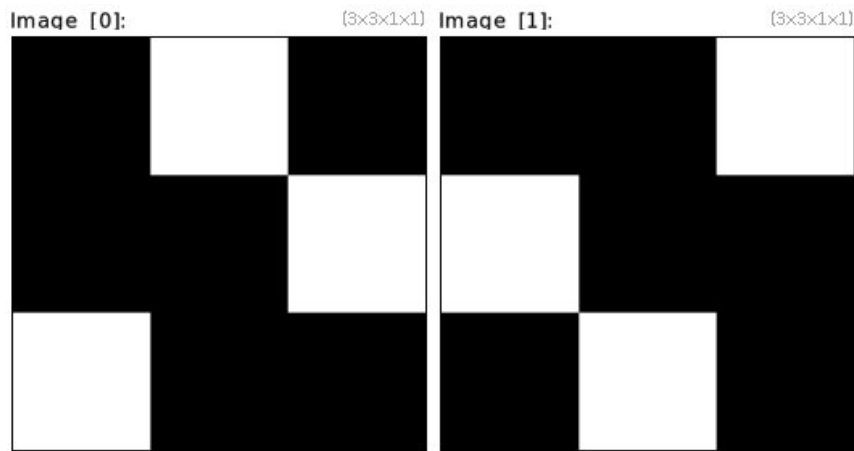
Example 460 : `image.jpg structuretensors blur 2 eigen split[0] c`

Tutorial page:

https://gmics.eu/tutorial/_eigen.shtml

2.11.3 *invert* (+)

Compute the inverse of the selected matrices.



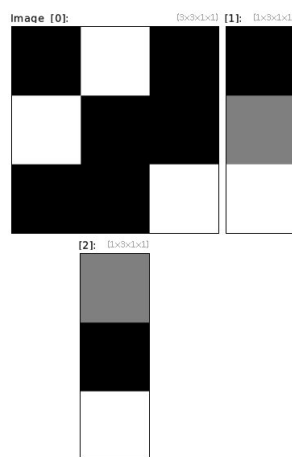
Example 461 : (0,1,0;0,0,1;1,0,0) +invert

2.11.4 *solve* (+)

Arguments:

- [image]

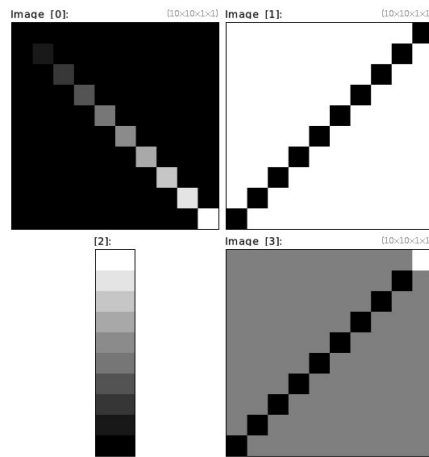
Solve linear system $AX = B$ for selected B-matrices and specified A-matrix.
If the system is under- or over-determined, the least square solution is returned.



Example 462 : (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 463 : `10,10,1,1,'if (x==y,x+u(-0.2,0.2),0)'+svd`

2.11.6 *transpose*

Transpose selected matrices.



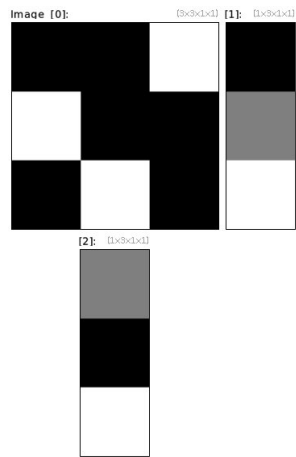
Example 464 : `image.jpg +transpose`

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 465 : `(0,0,1;1,0,0;0,1,0) (1;2;3) +trisolve[-1] [-2]`

2.12 3D Rendering

2.12.1 *add3d* (+)

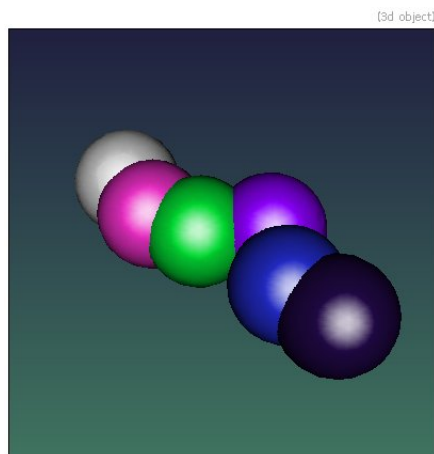
Arguments:

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



Example 467 : `repeat 20 torus3d 15,2 color3d[-1] ${-RGB} mul3d[-1] 0.5,1 if {$>%2}
rotate3d[-1] 0,1,0,90 fi add3d[-1] 70 add3d rotate3d[-1] 0,0,1,18 done double3d 0`

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:

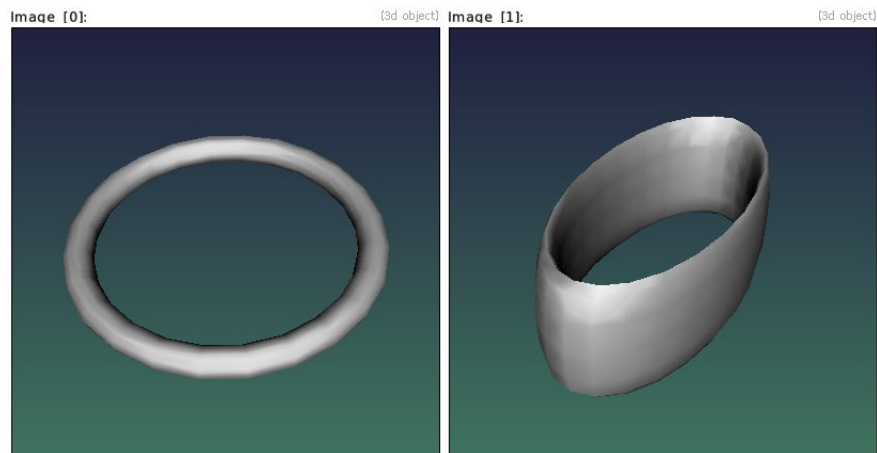
- `'target_x=0', 'target_y=0', 'target_z=0', 'up_x=0', 'up_y=-1' and 'up_z=0'`.

2.12.4 *apply_matrix3d*

Arguments:

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

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



Example 468: `torus3d 10,1 +apply_matrix3d {mul(rot(1,0,1,-15),[1,0,0,0,2,0,0,0,8],3)}`
`double3d 0`

2.12.5 *array3d*

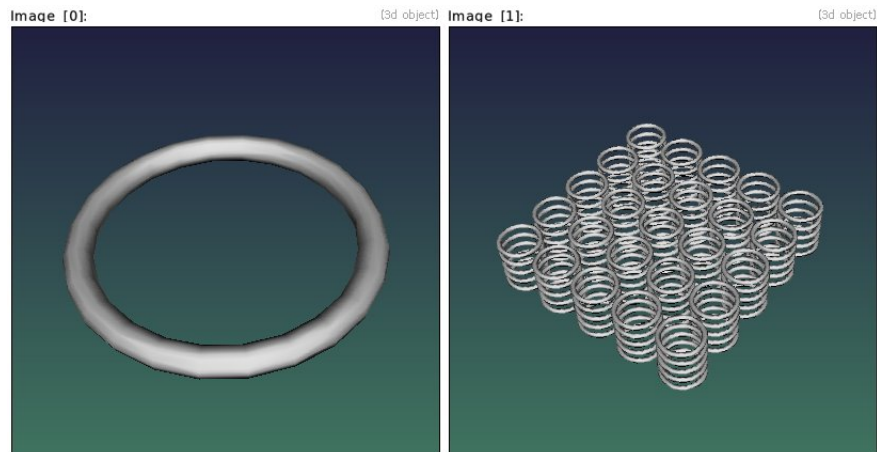
Arguments:

- `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 469: `torus3d 10,1 +array3d 5,5,5,110%,110%,300%`

2.12.6 *arrow3d*

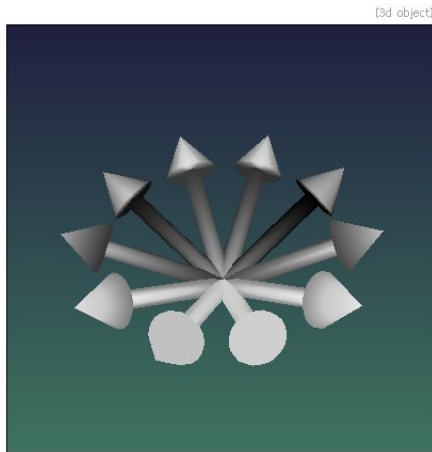
Arguments:

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

2.12.7 axes3d

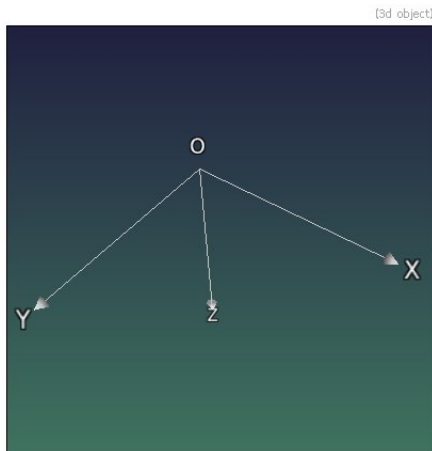
Arguments:

- `_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 471: `axes3d ,`

2.12.8 *box3d*

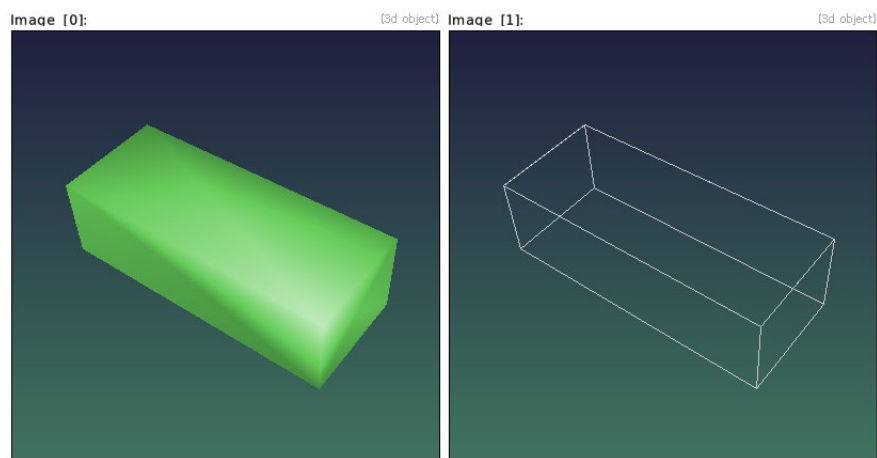
Arguments:

- `_size_x, _size_y, _size_z`

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

Default values:

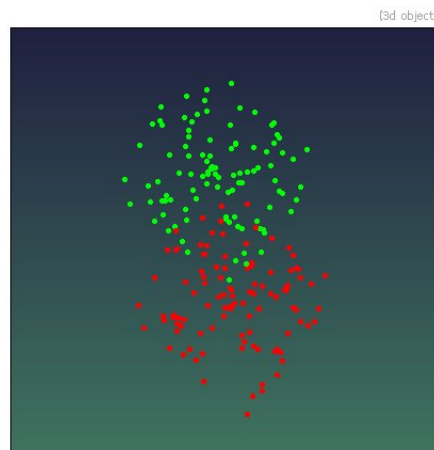
- `'size_x=1' and 'size_z=size_y=size_x'.`



Example 472 : `box3d 100,40,30 +primitives3d 1 color3d[-2] ${-RGB}`

2.12.9 *center3d*

Center selected 3D objects at (0,0,0).
(eq. to `'c3d'`).



Example 473 : `repeat 100 circle3d {u(100)}, {u(100)}, {u(100)}, 2 done add3d color3d[-1] 255,0,0
+center3d color3d[-1] 0,255,0 add3d`

2.12.10 *circle3d*

Arguments:

- `_x0, _y0, _z0, _radius >= 0`

Input 3D circle at specified coordinates.

Default values:

- `'x0=y0=z0=0'` and `'radius=1'`.



Example 474: `repeat 500 a={ $>*pi/250 } circle3d { cos(3*$a) }, { sin(2*$a) }, 0, { $a/50 } color3d[-1] $ {-RGB}, 0.4 done add3d`

2.12.11 *circles3d*

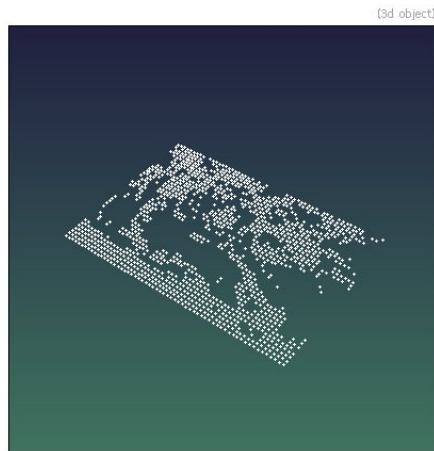
Arguments:

- `_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 475: `image.jpg luminance resize2dy 40 threshold 50% * 255 pointcloud3d color3d[-1] 255,255,255 circles3d 0.7`

2.12.12 *color3d* (+)

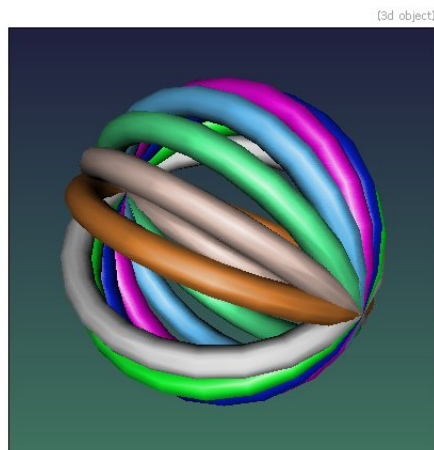
Arguments:

- `R, _G, _B, _opacity`

Set color and opacity of selected 3D objects.
(*eq. to 'col3d'*).

Default value:

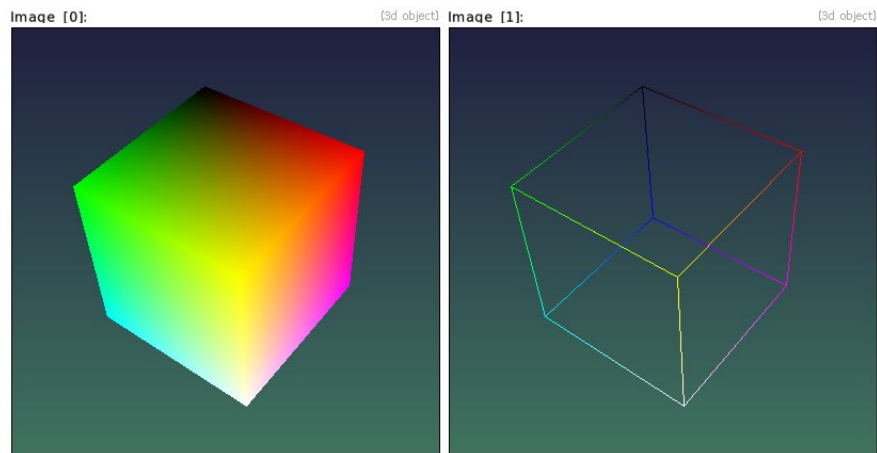
- `'B=G=R' and 'opacity=(undefined)'`.



Example 476: `torus3d 100,10 double3d 0 repeat 7 +rotate3d[-1] 1,0,0,20 color3d[-1] ${-RGB} done add3d`

2.12.13 *colorcube3d*

Input 3D color cube.



Example 477 : `colorcube3d mode3d 2 +primitives3d 1`

2.12.14 *cone3d*

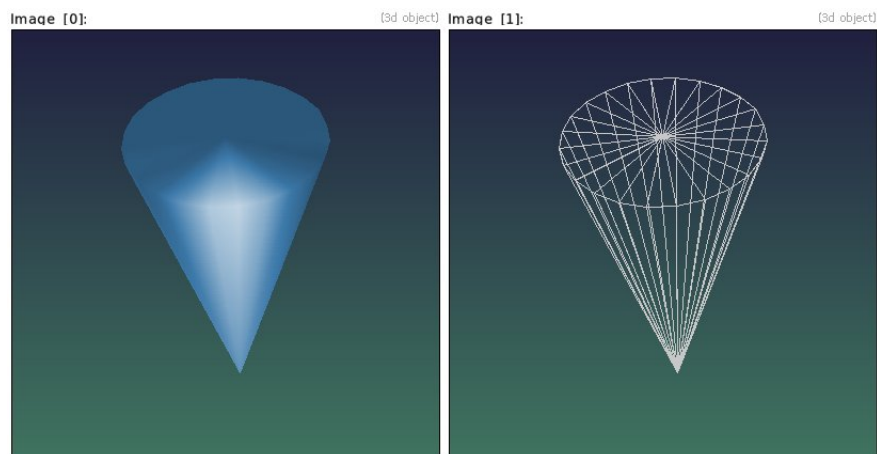
Arguments:

- `_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 478 : `cone3d 10,40 +primitives3d 1 color3d[-2] ${-RGB}`

2.12.15 *cubes3d*

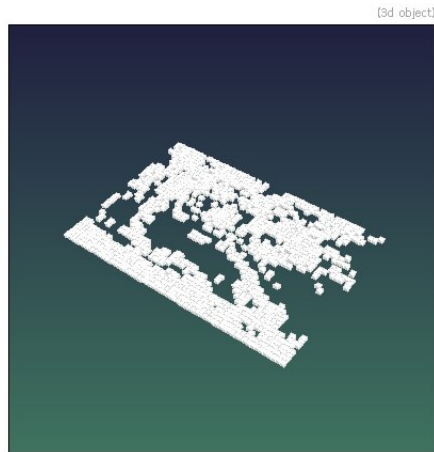
Arguments:

- `_size >= 0`

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

Default value:

- `'size=1'`.

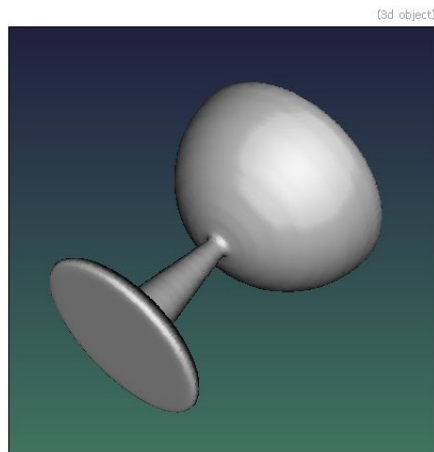


Example 479 : `image.jpg luminance resize2dy 40 threshold 50% * 255 pointcloud3d color3d[-1] 255,255,255 cubes3d 1`

2.12.16 *cup3d***Arguments:**

- `_resolution>0`

Input 3D cup object.



Example 480 : `cup3d ,`

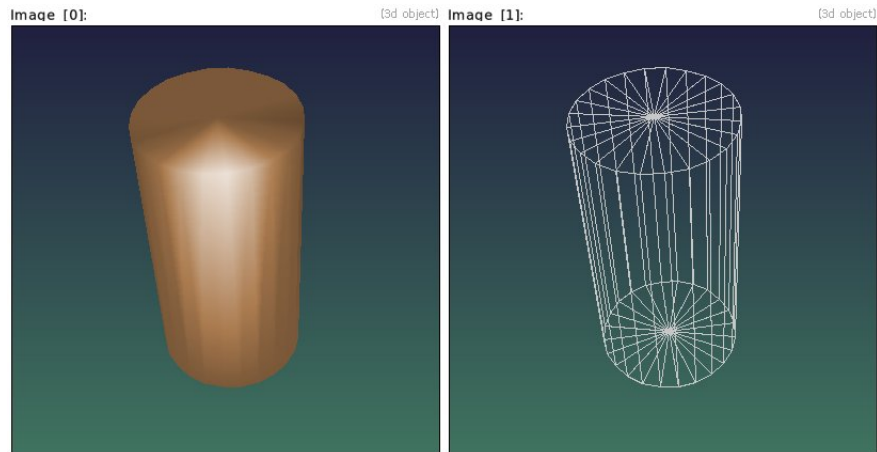
2.12.17 *cylinder3d***Arguments:**

- `_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'.



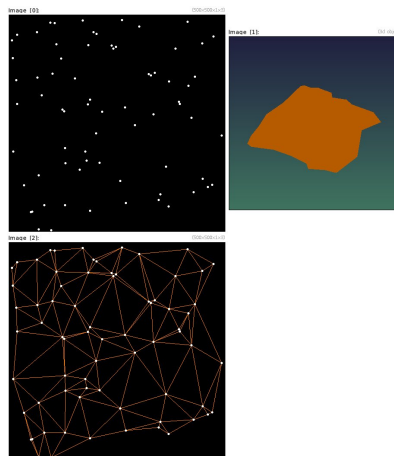
Example 481 : `cylinder3d 10,40 +primitives3d 1 color3d[-2] ${-RGB}`

2.12.18 *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.

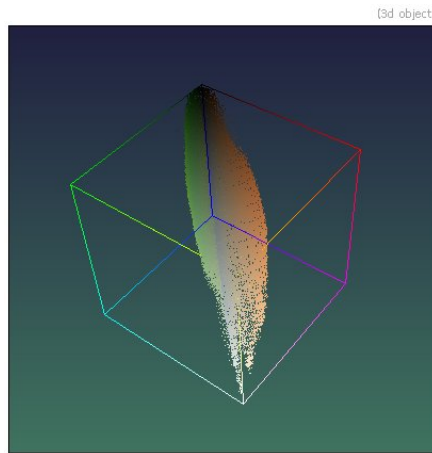
The output 3D object is a mesh composed of non-oriented triangles.



Example 482 : `500,500 noise 0.05,2 eq 1 * 255 +delaunay3d color3d[1] 255,128,0 dilate_circ[0] 5 to_rgb[0] +object3d[0] [1],0,0,0,1,1 max[-1] [0]`

2.12.19 *distribution3d*

Get 3D color distribution of selected images.



Example 483: `image.jpg distribution3d colorcube3d primitives3d[-1] 1 add3d`

2.12.20 *div3d* (+)

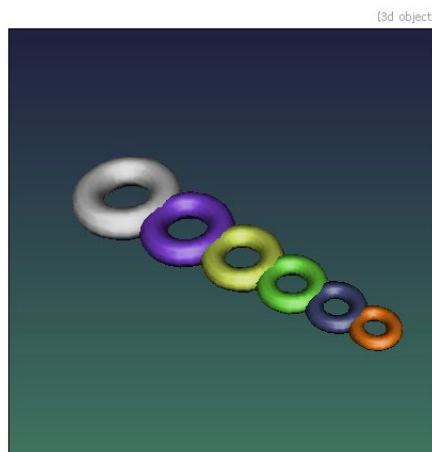
Arguments:

- `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 484: `torus3d 5,2 repeat 5 +add3d[-1] 12,0,0 div3d[-1] 1.2 color3d[-1] ${-RGB} done
add3d`

2.12.21 *double3d* (+)

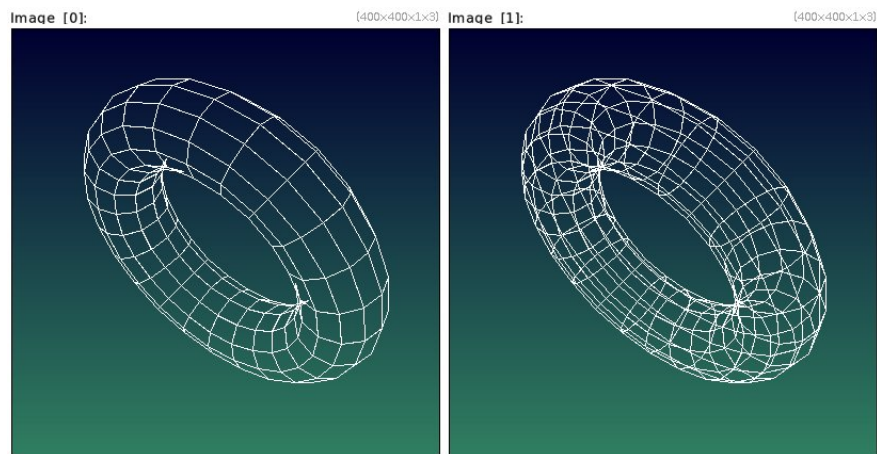
Arguments:

- `_is_double_sided={ 0 | 1 }`

Enable/disable double-sided mode for 3D rendering.
(eq. to 'db3d').

Default value:

- `'_is_double_sided=1'`.



Example 485: `mode3d 1 repeat 2 torus3d 100,30 rotate3d[-1] 1,1,0,60 double3d $>
snapshot3d[-1] 400 done`

2.12.22 *elevation3d* (+)

Arguments:

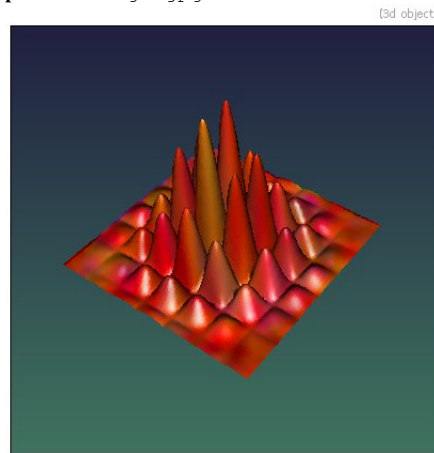
- `z-factor`
- `[elevation_map]`
- `'formula'`
- `(no arg)`

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.



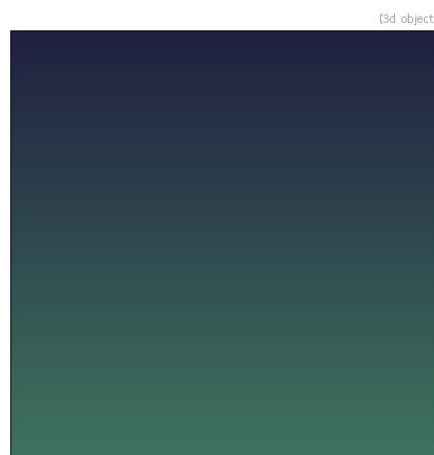
Example 486 : `image.jpg blur 5 elevation3d 0.5`



Example 487 : `128,128,1,3,u(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))'`

2.12.23 *empty3d*

Input empty 3D object.



Example 488 : `empty3d`

2.12.24 *extrude3d*

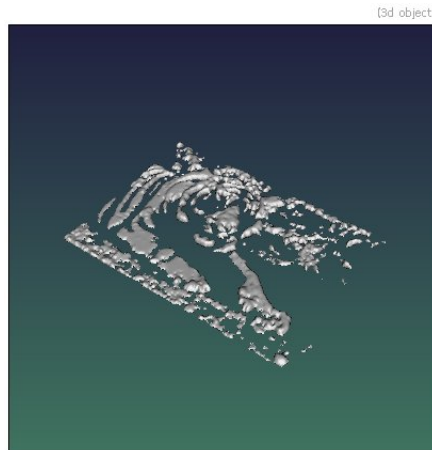
Arguments:

- `_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 489 : `image.jpg threshold 50% extrude3d 16`

2.12.25 *focale3d (+)*

Arguments:

- `focale`

Set 3D focale.

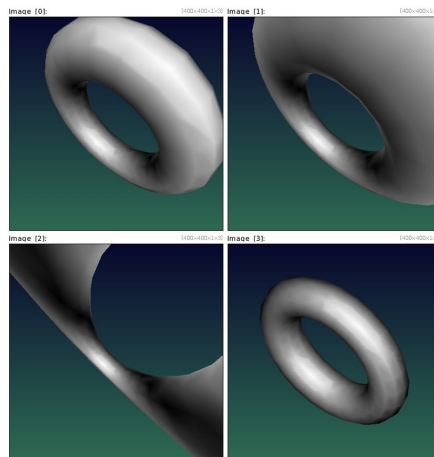
(eq. to `'f3d'`) .\n).

Set 'focale' to 0 to enable parallel projection (instead of perspective).

Set negative 'focale' will disable 3D sprite zooming.

Default value:

- `'focale=700'`.



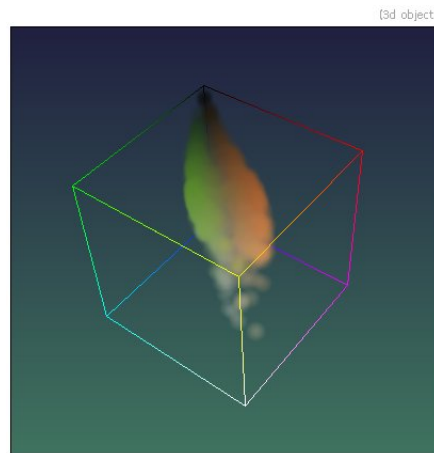
Example 490 : `repeat 5 torus3d 100,30 rotate3d[-1] 1,1,0,60 focale3d {$<*90} snapshot3d[-1]
400 done remove[0]`

2.12.26 *gaussians3d*

Arguments:

- `_size>0, _opacity`

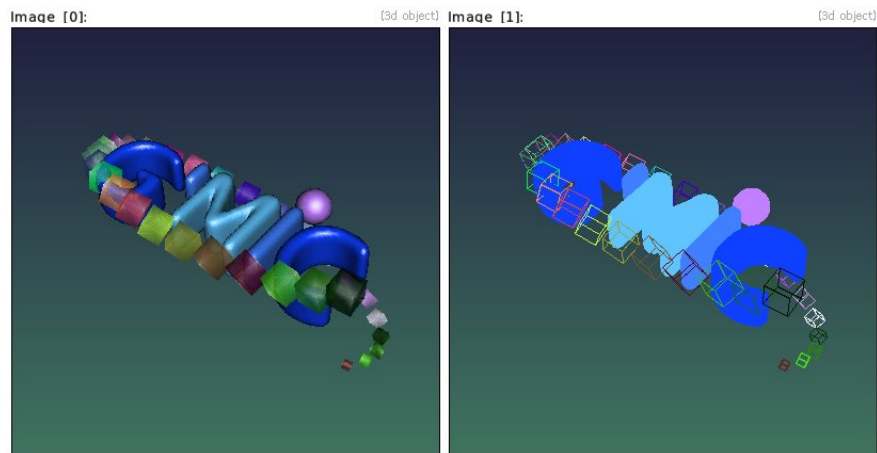
Convert selected 3D objects into set of 3D gaussian-shaped sprites.



Example 491 : `image.jpg r2dy 32 distribution3d gaussians3d 20 colorcube3d primitives3d[-1] 1
+3d`

2.12.27 *gm3d*

Input a 3D G'MIC logo.



Example 492 : `gmic3d +primitives3d 1`

2.12.28 *gyroid3d*

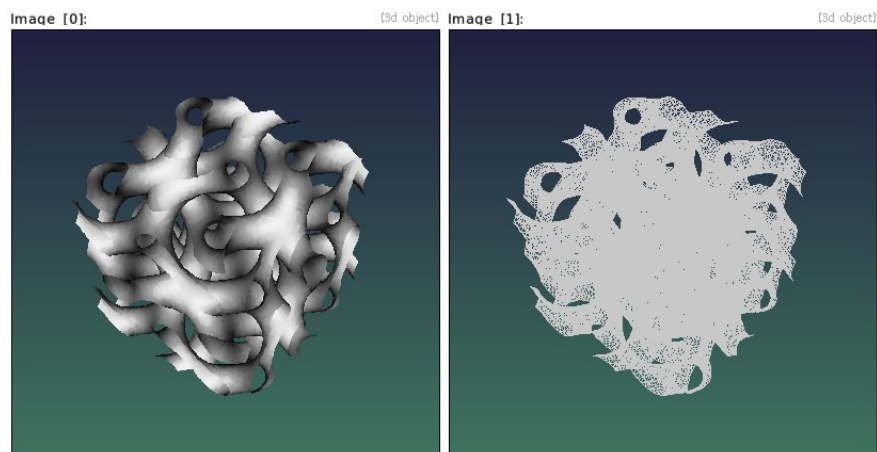
Arguments:

- `_resolution>0, _zoom`

Input 3D gyroid at (0,0,0), with specified resolution.

Default values:

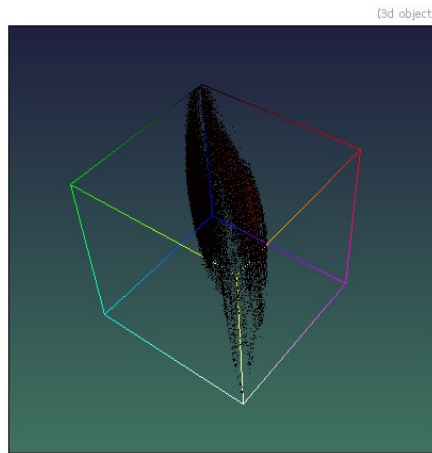
- `'resolution=32'` and `'zoom=5'`.



Example 493 : `gyroid3d 48 +primitives3d 1`

2.12.29 *histogram3d*

Get 3D color histogram of selected images.



Example 494 : `image.jpg histogram3d colorcube3d primitives3d[-1] 1 add3d`

2.12.30 *image6cube3d*

Generate 3D mapped cubes from 6-sets of selected images.



Example 495 : `image.jpg animate flower,"30,0","30,5",6 image6cube3d`

2.12.31 *imageblocks3d*

Arguments:

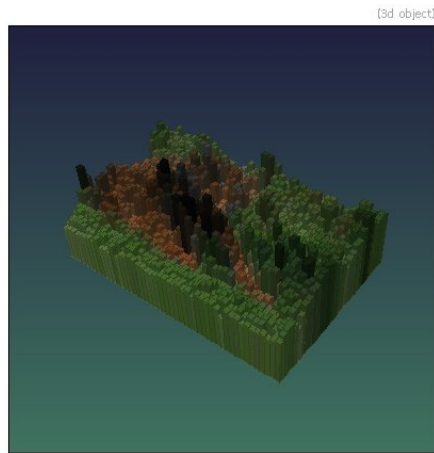
- `_maximum_elevation, _smoothness[%]>=0`

Generate 3D blocks from selected images.

Transparency of selected images is taken into account.

Default values:

- `'maximum_elevation=10' and 'smoothness=0'.`



Example 496 : `image.jpg resize2dy 32 imageblocks3d -20 mode3d 3`

2.12.32 *imagecube3d*

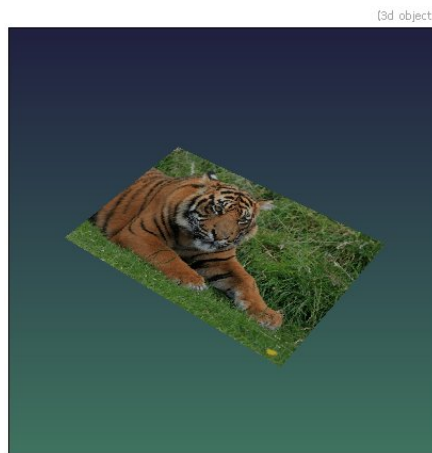
Generate 3D mapped cubes from selected images.



Example 497 : `image.jpg imagecube3d`

2.12.33 *imageplane3d*

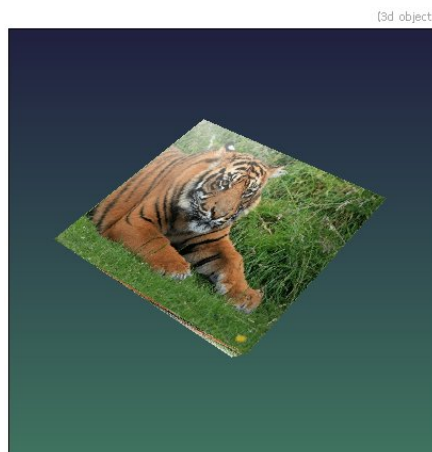
Generate 3D mapped planes from selected images.



Example 498 : `image.jpg imageplane3d`

2.12.34 *imagepyramid3d*

Generate 3D mapped pyramids from selected images.



Example 499 : `image.jpg imagepyramid3d`

2.12.35 *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 500 : `image.jpg imagerubik3d ,`

2.12.36 *imagesphere3d*

Arguments:

- `_resolution1>=3, _resolution2>=3`

Generate 3D mapped sphere from selected images.

Default values:

- `'resolution1=32' and 'resolutions2=16'.`



Example 501 : `image.jpg imagesphere3d 32,16`

2.12.37 *isoline3d (+)*

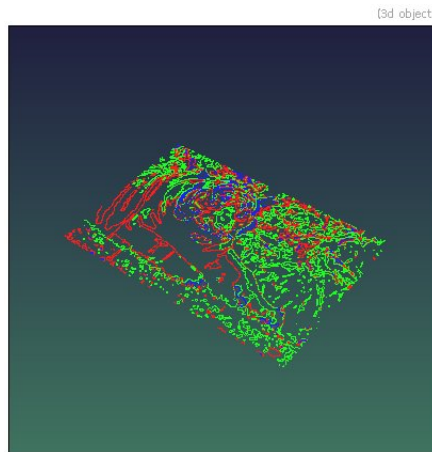
Arguments:

- `isovalue[%]`
- `'formula', value, _x0, _y0, _x1, _y1, _size-x>0[%], _size-y>0[%]`

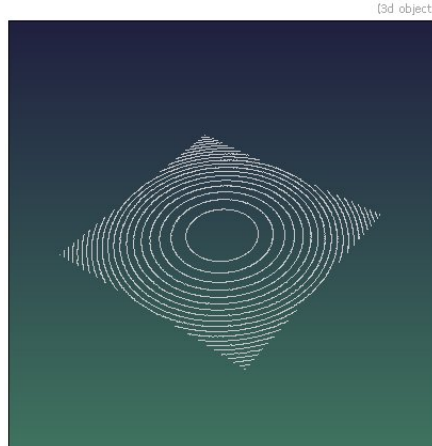
Extract 3D isolines with specified value from selected images or from specified formula.

Default values:

- 'x0=y0=-3', 'x1=y1=3' and 'size_x=size_y=256'.



Example 502 : `image.jpg blur 1 isoline3d 50%`



Example 503 : `isoline3d 'X=x-w/2;Y=y-h/2;(X^2+Y^2)%20',10,-10,-10,10,10`

2.12.38 *isosurface3d* (+)

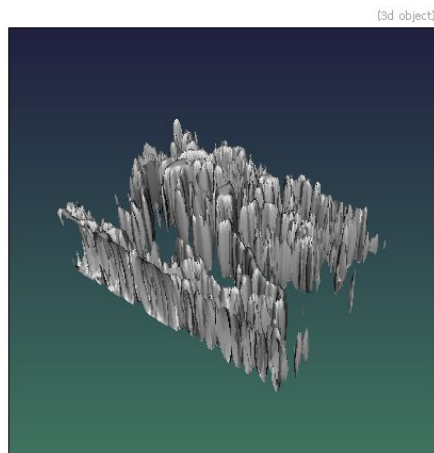
Arguments:

- `isovalue[%]`
- `'formula',value,_x0,_y0,_z0,_x1,_y1,_z1,_size_x>0[%],_size_y>0[%],_size_z>0[%]`

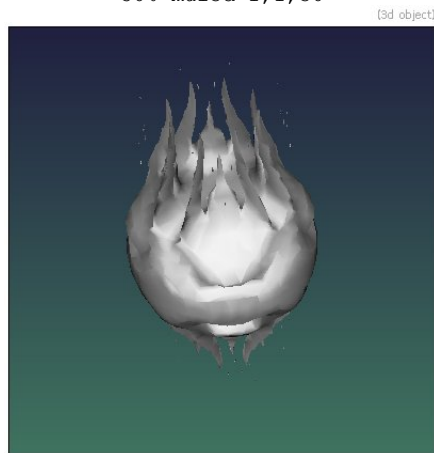
Extract 3D isosurfaces with specified value from selected images or from specified formula.

Default values:

- 'x0=y0=z0=-3', 'x1=y1=z1=3' and 'size_x=size_y=size_z=32'.



Example 504 : `image.jpg resize2dy 128 luminance threshold 50% expand.z 2,0 blur 1 isosurface3d 50% mul3d 1,1,30`



Example 505 : `isosurface3d 'x^2+y^2+abs(z)^abs(4*cos(x*y*z*3))',3`

2.12.39 *label3d*

Arguments:

- `"text", font_height>=0, _opacity, _color1, ...`

Generate 3D text label.

Default values:

- `'font_height=13', 'opacity=1' and 'color=255,255,255'.`

2.12.40 *label_points3d*

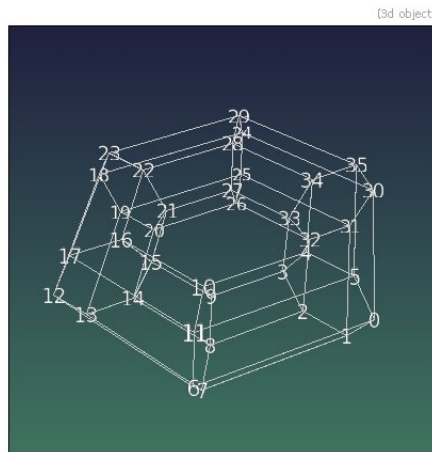
Arguments:

- `_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 506 : `torus3d 100,40,6,6 label.points3d 23,1 mode3d 1`

2.12.41 *lathe3d*

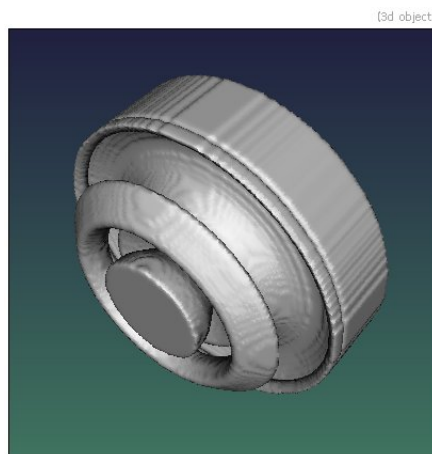
Arguments:

- `_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 507 : `300,300 rand -1,1 blur 40 sign normalize 0,255 lathe3d ,`

2.12.42 *light3d* (+)

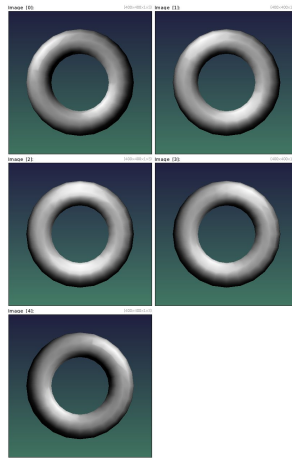
Arguments:

- `position_x, position_y, position_z`
- `[texture]`
- `(no arg)`

Set the light coordinates or the light texture for 3D rendering.

(*eq. to 'l3d'*) . \n).

(no arg) resets the 3D light to default.



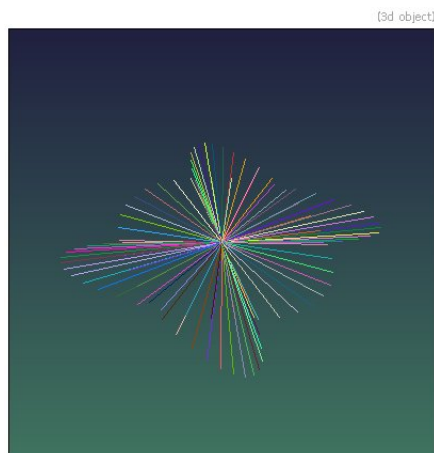
Example 508 : `torus3d 100,30 double3d 0 specs3d 1.2 repeat 5 light3d {${>*100}},0,-300
+snapshot3d[0] 400 done remove[0]`

2.12.43 *line3d*

Arguments:

- `x0,y0,z0,x1,y1,z1`

Input 3D line at specified coordinates.



Example 509 : `repeat 100 a={ $>*pi/50 } line3d 0,0,0,{cos(3*$a)},{sin(2*$a)},0 color3d. ${-RGB}
done add3d`

2.12.44 *lissajous3d*

Arguments:

- `resolution>1,a,A,b,B,c,C`

Input 3D lissajous curves ($x(t)=\sin(a*t+A*2*\pi)$, $y(t)=\sin(b*t+B*2*\pi)$, $z(t)=\sin(c*t+C*2*\pi)$).

Default values:

- `'resolution=1024', 'a=2', 'A=0', 'b=1', 'B=0', 'c=0' and 'C=0'.`



Example 510 : `lissajous3d ,`

2.12.45 *mode3d (+)*

Arguments:

- `_mode`

Set static 3D rendering mode.

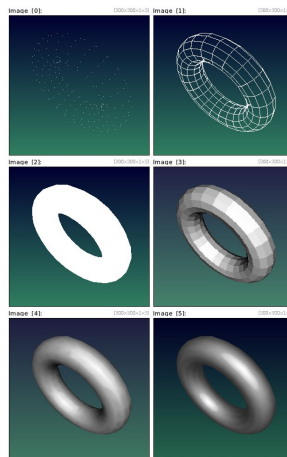
(*eq. to 'm3d'*) . \n).

'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 511: `(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]`

2.12.46 *moded3d* (+)

Arguments:

- `_mode`

Set dynamic 3D rendering mode for interactive 3D viewer.

(*eq. to 'md3d'*) . \n).

'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'` .

2.12.47 *mul3d* (+)

Arguments:

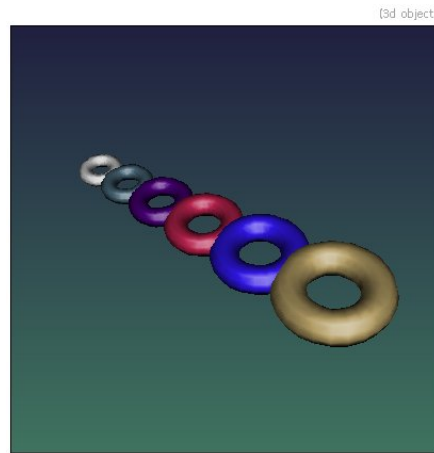
- `factor`
- `factor_x, factor_y, _factor_z`

Scale selected 3D objects isotropically or anisotropically, with specified factors.

(*eq. to '*3d'*).

Default value:

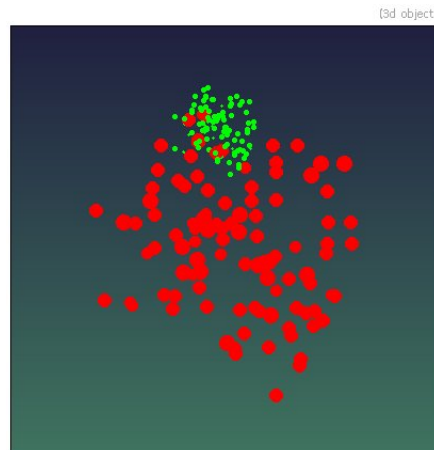
- `'factor_z=0'` .



Example 512: `torus3d 5,2 repeat 5 +add3d[-1] 10,0,0 mul3d[-1] 1.2 color3d[-1] ${-RGB} done
add3d`

2.12.48 *normalize3d*

Normalize selected 3D objects to unit size.
(eq. to 'n3d').



Example 513: `repeat 100 circle3d {u(3)},{u(3)},{u(3)},0.1 done add3d color3d[-1] 255,0,0
+normalize3d[-1] color3d[-1] 0,255,0 add3d`

2.12.49 *opacity3d (+)*

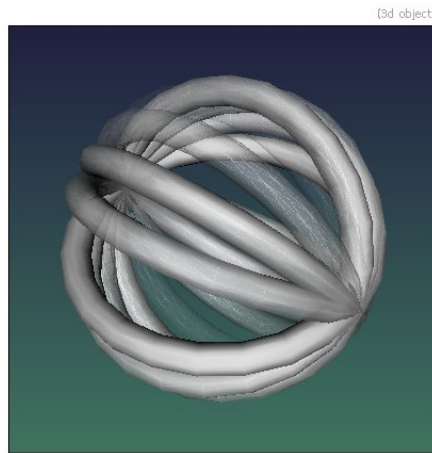
Arguments:

- `_opacity`

Set opacity of selected 3D objects.
(eq. to 'o3d').

Default value:

- `'opacity=1'`.



Example 514 : `torus3d 100,10 double3d 0 repeat 7 +rotate3d[-1] 1,0,0,20 opacity3d[-1] {u}
done add3d`

2.12.50 *parametric3d*

Arguments:

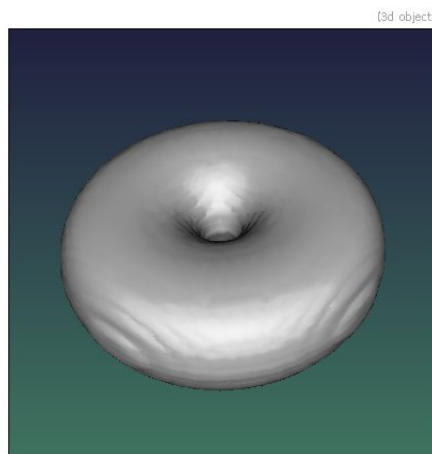
- `_x(a,b), _y(a,b), _z(a,b), _amin, _amax, _bmin, _bmax, _res_a>0, _res_b>0, _res_x>0, _res_y>0, _res_z>0, _smoothness>=0, _isovalue>=0`

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 515 : `parametric3d ,`

2.12.51 *pca_patch3d*

Arguments:

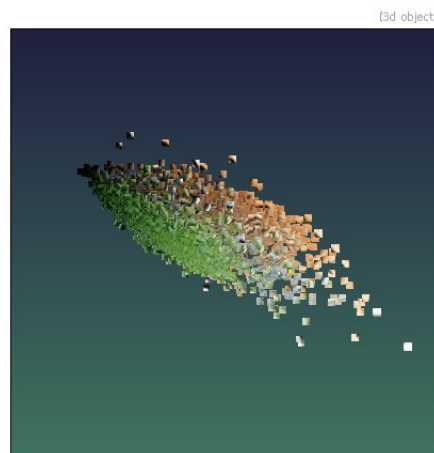
- `_patch_size>0, _M>0, _N>0, _normalize_input={ 0 | 1 }, _normalize_output={ 0 | 1 }, _lambda_xy`

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 516 : `image.jpg pca_patch3d 7`

2.12.52 *plane3d*

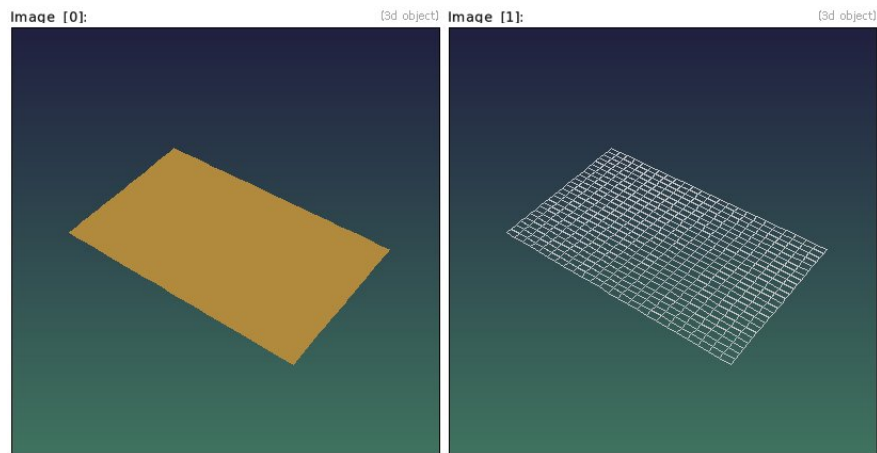
Arguments:

- `_size_x, _size_y, _nb_subdivisions_x>0, _nb_subdisivions_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 517: `plane3d 50,30 +primitives3d 1 color3d[-2] ${-RGB}`

2.12.53 *point3d*

Arguments:

- `x0,y0,z0`

Input 3D point at specified coordinates.



Example 518: `repeat 1000 a={>*pi/500} point3d {cos(3*$a)},{sin(2*$a)},0 color3d[-1] ${-RGB}
done add3d`

2.12.54 *pointcloud3d*

Convert selected planar or volumetric images to 3D point clouds.



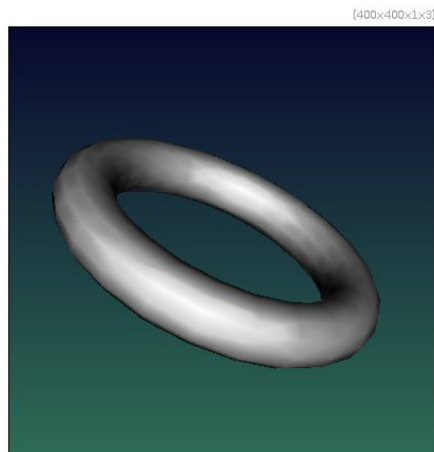
Example 519 : `image.jpg luminance resize2dy 100 threshold 50% mul 255 pointcloud3d
color3d[-1] 255,255,255`

2.12.55 *pose3d*

Arguments:

- `p1, ..., p12`

Apply 3D pose matrix to selected 3D objects.



Example 520 : `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`

2.12.56 *primitives3d*

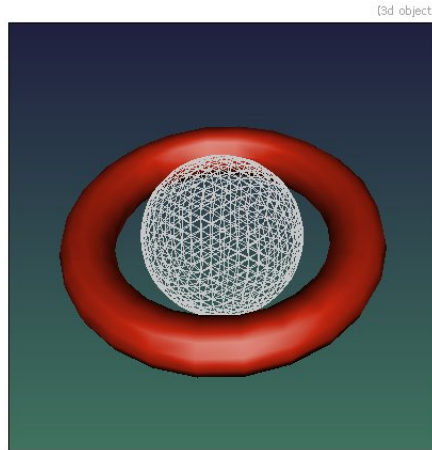
Arguments:

- `mode`

Convert primitives of selected 3D objects.

(eq. to 'p3d') .\n).

'mode' can be { 0=points | 1=outlines | 2=non-textured }.



Example 521 : `sphere3d 30 primitives3d 1 torus3d 50,10 color3d[-1] ${-RGB} add3d`

2.12.57 *projections3d*

Arguments:

- `_x[%],_y[%],_z[%],_is_bounding_box={ 0 | 1 }`

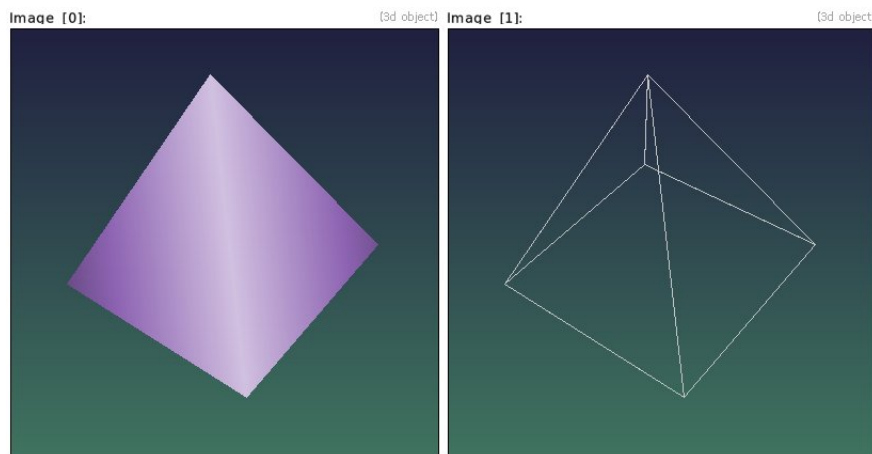
Generate 3D xy,xz,yz projection planes from specified volumetric images.

2.12.58 *pyramid3d*

Arguments:

- `width,height`

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



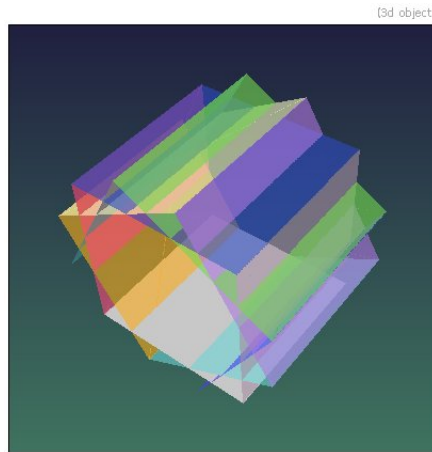
Example 522 : `pyramid3d 100,-100 +primitives3d 1 color3d[-2] ${-RGB}`

2.12.59 *quadrangle3d*

Arguments:

- $x_0, y_0, z_0, x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3$

Input 3D quadrangle at specified coordinates.



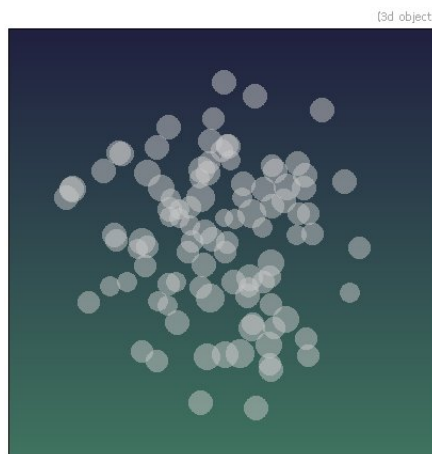
Example 523 : `quadrangle3d -10,-10,10,10,-10,10,10,10,-10,10,10 repeat 10 +rotate3d[-1]
0,1,0,30 color3d[-1] ${-RGB},0.6 done add3d mode3d 2`

2.12.60 *random3d*

Arguments:

- `nb_points >= 0`

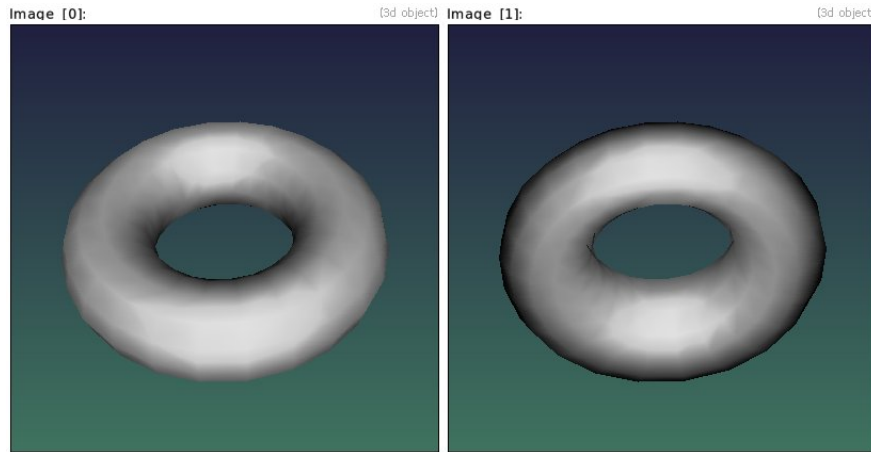
Input random 3D point cloud in $[0,1]^3$.



Example 524 : `random3d 100 circles3d 0.1 opacity3d 0.5`

2.12.61 *reverse3d* (+)

Reverse primitive orientations of selected 3D objects.
(eq. to 'rv3d').

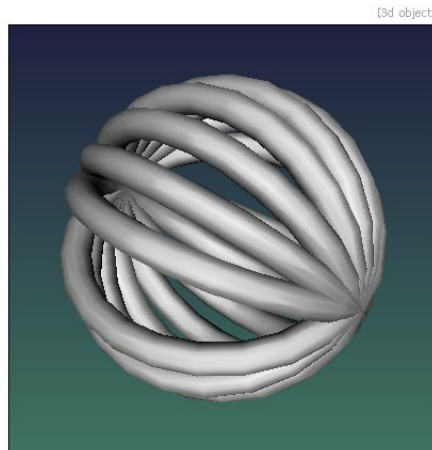


Example 525 : `torus3d 100,40 double3d 0 +reverse3d`

2.12.62 *rotate3d* (+)**Arguments:**

- `u,v,w,angle`

Rotate selected 3D objects around specified axis with specified angle (in deg.).
(eq. to 'r3d').

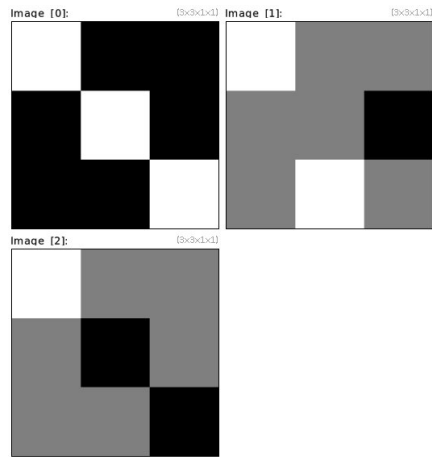


Example 526 : `torus3d 100,10 double3d 0 repeat 7 +rotate3d[-1] 1,0,0,20 done add3d`

2.12.63 *rotation3d***Arguments:**

- `u,v,w,angle`

Input 3x3 rotation matrix with specified axis and angle (in deg).



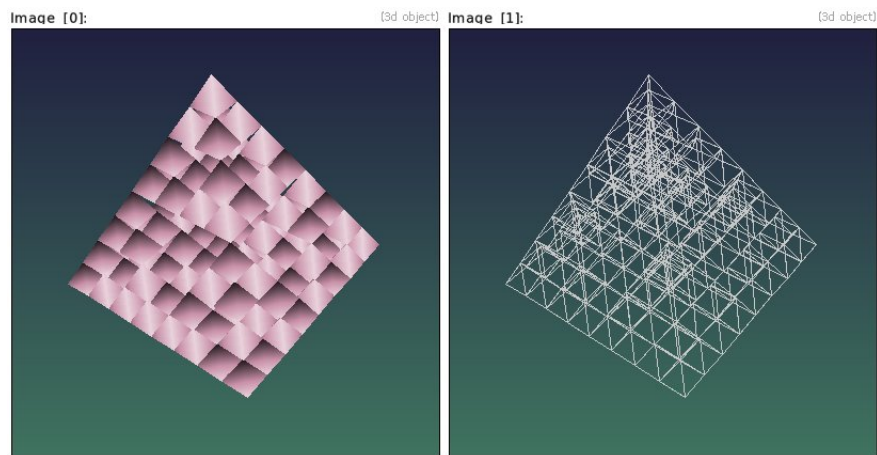
Example 527 : `rotation3d 1,0,0,0 rotation3d 1,0,0,90 rotation3d 1,0,0,180`

2.12.64 *sierpinski3d*

Arguments:

- `_recursion_level>=0, _width, _height`

Input 3d Sierpinski pyramid.



Example 528 : `sierpinski3d 3,100,-100 +primitives3d 1 color3d[-2] ${-RGB}`

2.12.65 *size3d*

Return bounding box size of the last selected 3D object.

2.12.66 *skeleton3d*

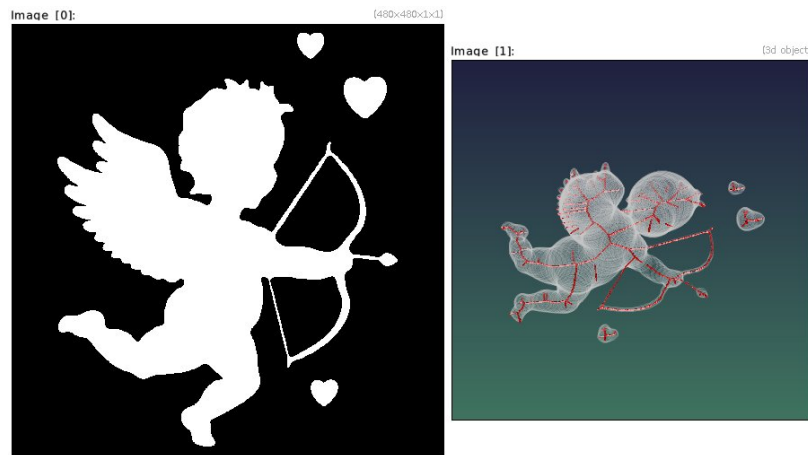
Arguments:

- `_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 529 : `shape_cupid 480 +skeleton3d` ,

2.12.67 *snapshot3d*

Arguments:

- `_size>0, _zoom>=0, _backgroundR, _backgroundG, _backgroundB, _backgroundA`
`[background_image], zoom>=0`

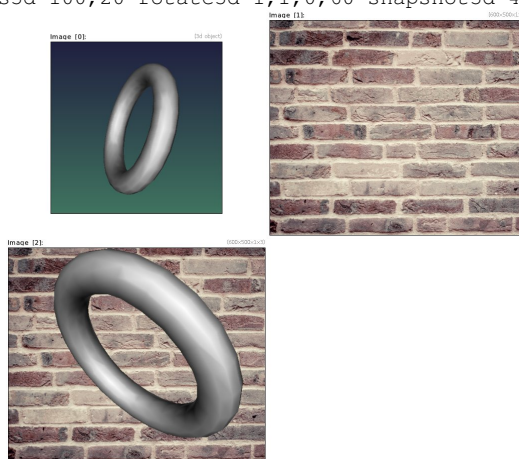
Take 2d snapshots of selected 3D objects.
 Set 'zoom' to 0 to disable object auto-scaling.

Default values:

- 'size=512', 'zoom=1' and '[background_image]=(default)'.



Example 530 : `torus3d 100,20 rotate3d 1,1,0,60 snapshot3d 400,1.2,128,64,32`



Example 531 : `torus3d 100,20 rotate3d 1,1,0,60 sample ? +snapshot3d[0] [1],1.2`

2.12.68 *spec3d* (+)

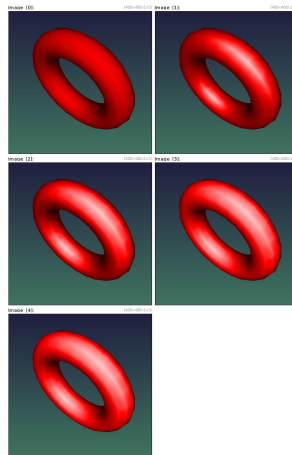
Arguments:

- `value` ≥ 0

Set lightness of 3D specular light.
(*eq. to* 's13d').

Default value:

- 'value=0.15'.



Example 532 : `(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]`

2.12.69 *specs3d* (+)

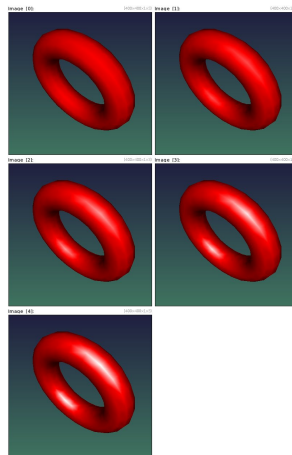
Arguments:

- `value` ≥ 0

Set shininess of 3D specular light.
(eq. to 'ss3d').

Default value:

- `'value=0.8'`.



Example 533 : `(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]`

2.12.70 *sphere3d* (+)

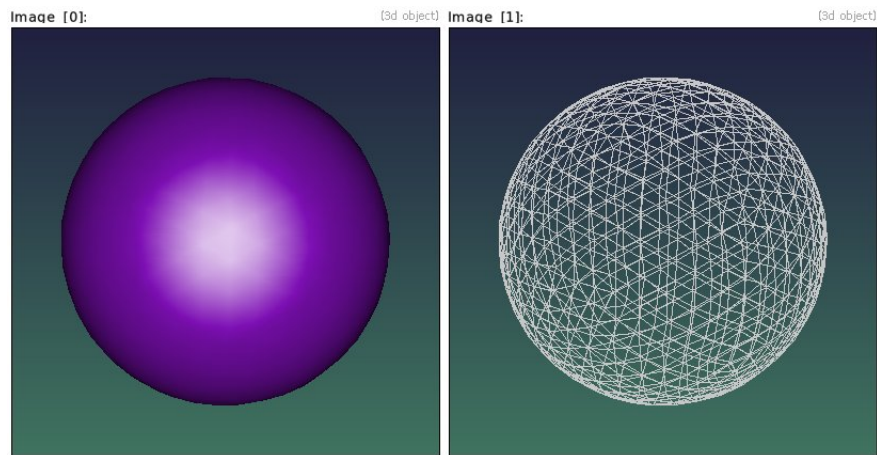
Arguments:

- `radius, _nb recursions>=0`

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

Default value:

- `'nb recursions=3'`.



Example 534 : `sphere3d 100 +primitives3d 1 color3d[-2] ${-RGB}`

2.12.71 *spherical3d*

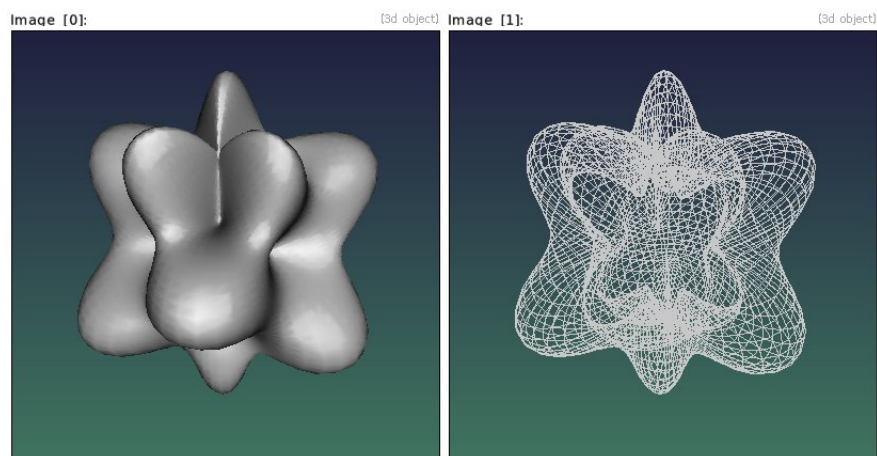
Arguments:

- `_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(4*theta))"'`.



Example 535 : `spherical3d 64 +primitives3d 1`

2.12.72 *spline3d*

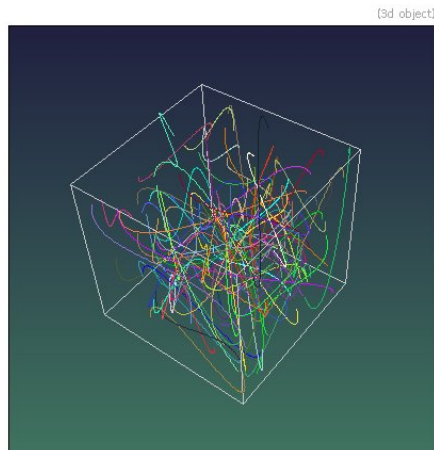
Arguments:

- `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 536 : `repeat 100 spline3d {u},{u},{u},{u},{u},{u},{u},{u},{u},{u},{u},{u},128
color3d[-1] ${-RGB} done box3d 1 primitives3d[-1] 1 add3d`

2.12.73 *split3d (+)*

Arguments:

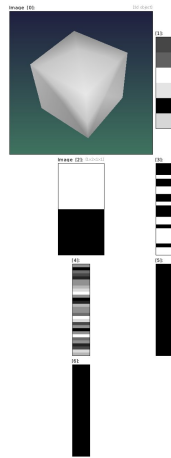
- `_keep_shared_data={ 0 | 1 }`

Split selected 3D objects into 6 feature vectors : { header, sizes, vertices, primitives, colors, opacities }.
(*eq. to 's3d'*) . \n).

To recreate the 3D object, append these 6 images along the y-axis.

Default value:

- `'keep_shared_data=1'`.

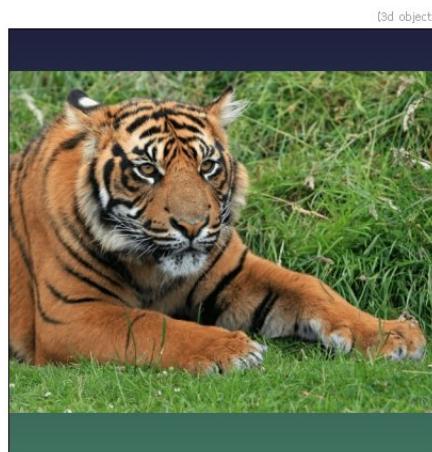


Example 537 : `box3d 100 +split3d`

2.12.74 *sprite3d*

Convert selected images as 3D sprites.

Selected images with alpha channels are managed.



Example 538 : `image.jpg sprite3d`

2.12.75 *sprites3d*

Arguments:

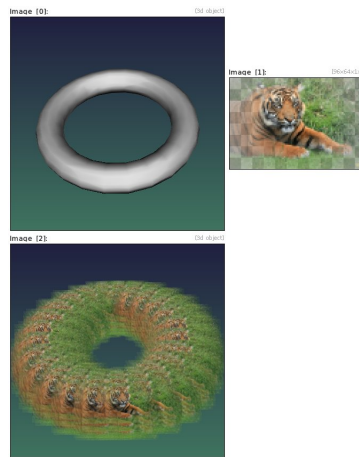
- `[sprite],_sprite_has_alpha_channel={ 0 | 1 }`

Convert selected 3D objects as a sprite cloud.

Set 'sprite_has_alpha_channel' to 1 to make the last channel of the selected sprite be a transparency mask.

Default value:

- `'mask_has_alpha_channel=0'`.



Example 539 : `torus3d 100,20 image.jpg resize2dy[-1] 64 100%,100% gaussian[-1] 30%,30% *[-1] 255 append[-2,-1] c +sprites3d[0] [1],1 display-rgba[-2]`

2.12.76 *star3d*

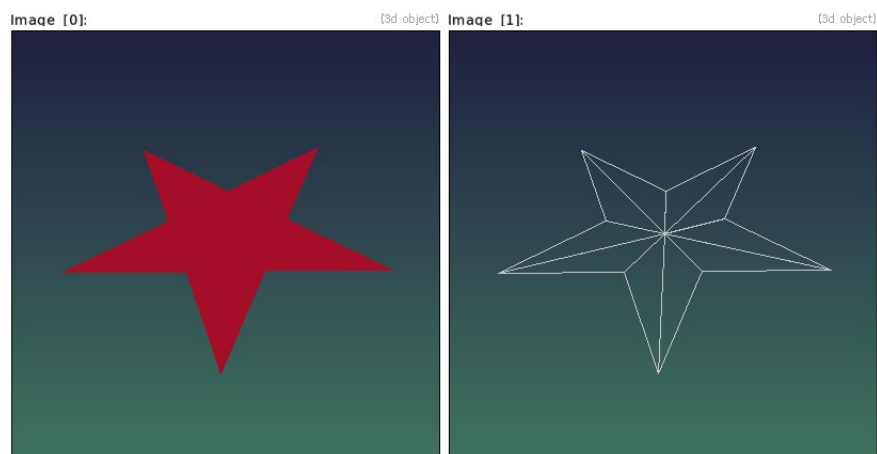
Arguments:

- `_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 540 : `star3d , +primitives3d 1 color3d[-2] ${-RGB}`

2.12.77 *streamline3d (+)*

Arguments:

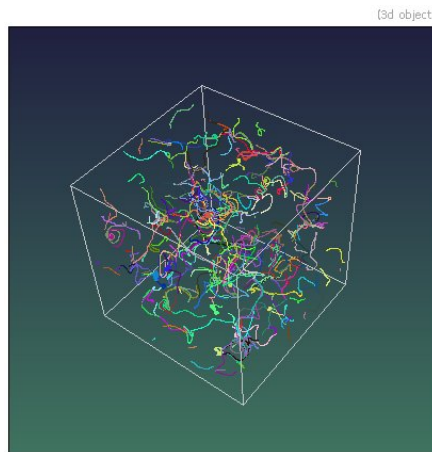
- `x[%],y[%],z[%],_L>=0,_dl>0,_interpolation,_is_backward={ 0 | 1 }`
`},_is_oriented={ 0 | 1 }`
- `'formula',x,y,z,_L>=0,_dl>0,_interpolation,_is_backward={ 0 | 1 }`
`},_is_oriented={ 0 | 1 }`

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 541 : `100,100,100,3 rand -10,10 blur 3 repeat 300 +streamline3d[0]
 {u(100)}, {u(100)}, {u(100)}, 1000,1,1 color3d[-1] ${-RGB} done remove[0] box3d 100
 primitives3d[-1] 1 add3d`

2.12.78 *sub3d* (+)

Arguments:

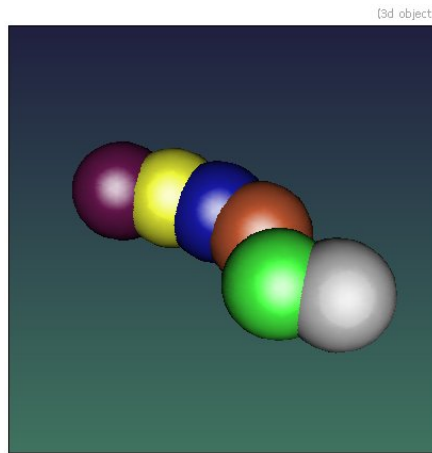
- `tx,ty,tz`

Shift selected 3D objects with the opposite of specified displacement vector.

(eq. to '3d').

Default values:

- `'ty=tz=0'.`



Example 542 : `sphere3d 10 repeat 5 +sub3d[-1] 10,{u(-10,10)},0 color3d[-1] ${-RGB} done add3d`

2.12.79 *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 543 : `superformula3d ,`

2.12.80 *tensors3d*

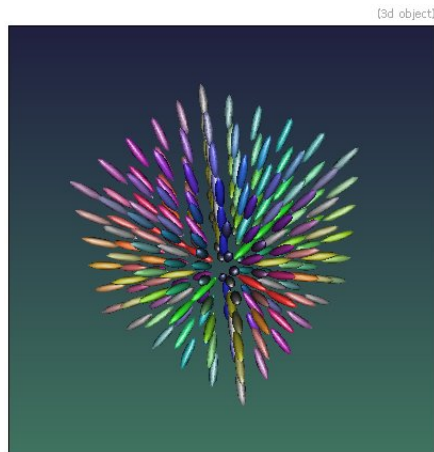
Arguments:

- `_radius_factor>=0,_shape={ 0=box | >=N=ellipsoid },_radius_min>=0`

Generate 3D tensor fields from selected images. when 'shape'>0, it gives the ellipsoid shape precision.

Default values:

- 'radius_factor=1', 'shape=2' and 'radius_min=0.05'.



Example 544 : `6,6,6,9,"U = [x,y,z] - [w,h,d]/2; U/=norm(U); mul(U,U,3) + 0.3*eye(3)"`
`tensors3d 0.8`

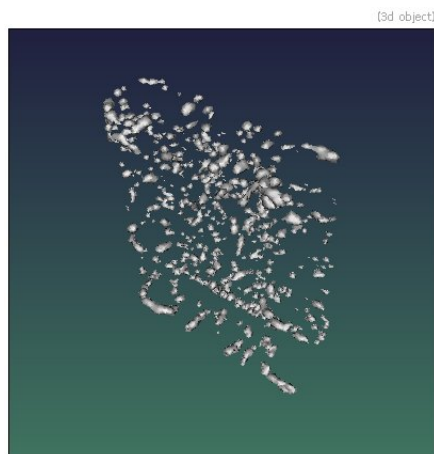
2.12.81 *text_pointcloud3d***Arguments:**

- `_"text1",_"text2",_smoothness`

Input 3D text pointcloud from the two specified strings.

Default values:

- 'text1="text1"', 'text2="text2"' and 'smoothness=1'.



Example 545 : `text_pointcloud3d "G'MIC", "Rocks!"`

2.12.82 *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 546 : `text3d "G'MIC as a\n3D logo!"`

2.12.83 *texturize3d*

Arguments:

- `[ind_texture], _[ind_coords]`

Texturize selected 3D objects with specified texture and coordinates.

(eq. to `'t3d'`) .\n).

When `'[ind_coords]'` is omitted, default XY texture projection is performed.

Default value:

- `'ind_coords=(undefined)'`.



Example 547: `image.jpg torus3d 100,30 texturize3d[-1] [-2] keep[-1]`

2.12.84 *torus3d*

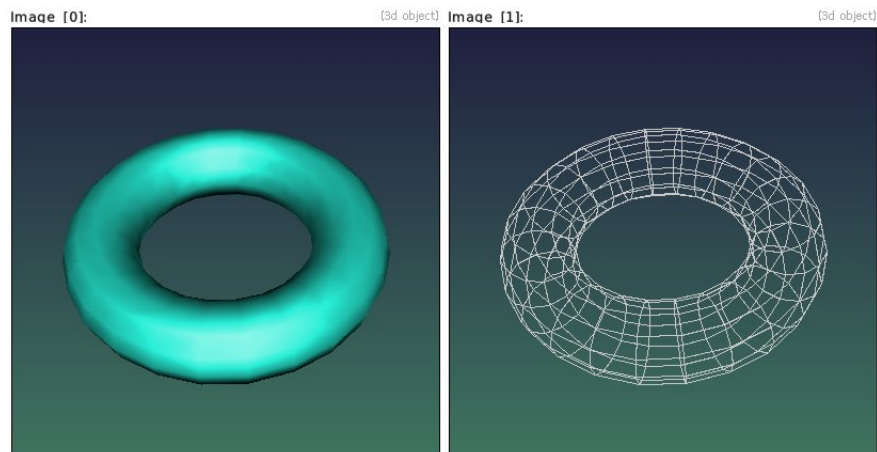
Arguments:

- `_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'.`



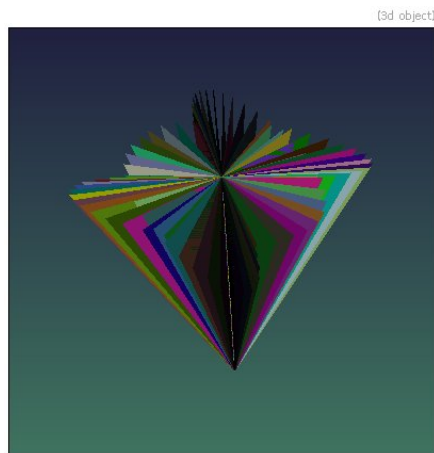
Example 548: `torus3d 10,3 +primitives3d 1 color3d[-2] ${-RGB}`

2.12.85 *triangle3d*

Arguments:

- `x0, y0, z0, x1, y1, z1, x2, y2, z2`

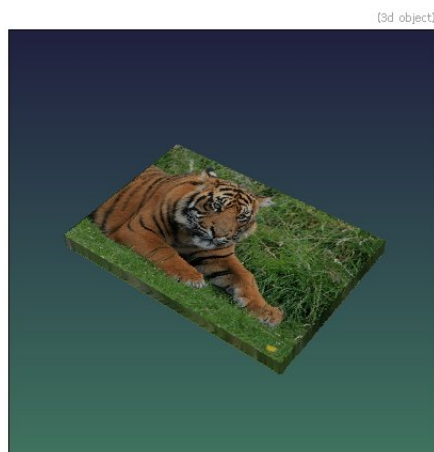
Input 3D triangle at specified coordinates.



Example 549 : `repeat 100 a={ $\pi/50$ } triangle3d 0,0,0,0,0,3,{ $\cos(3*a)$ },{ $\sin(2*a)$ },0
color3d[-1] $\{-RGB\}$ done add3d`

2.12.86 *volume3d*

Transform selected 3D volumetric images as 3D parallelepipedic objects.



Example 550 : `image.jpg animate blur,0,5,30 append z volume3d`

2.12.87 *weird3d*

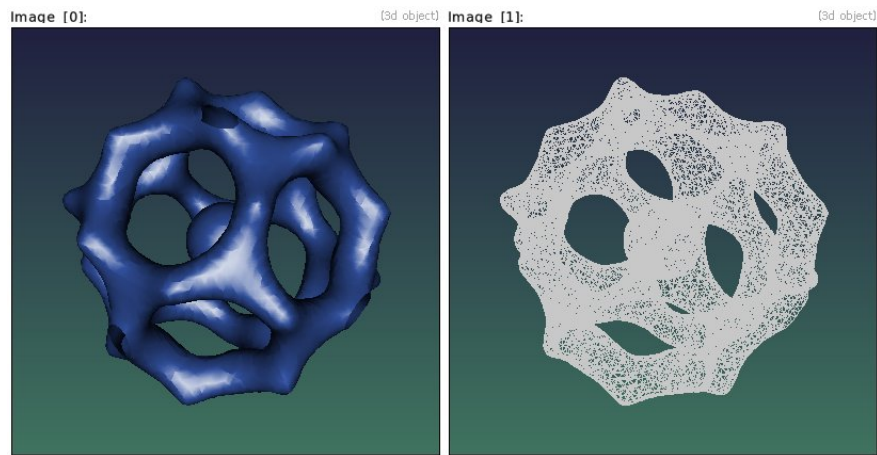
Arguments:

- `_resolution>0`

Input 3D weird object at (0,0,0), with specified resolution.

Default value:

- `'resolution=32'.`



Example 551 : `weird3d 48 +primitives3d 1 color3d[-2] ${-RGB}`

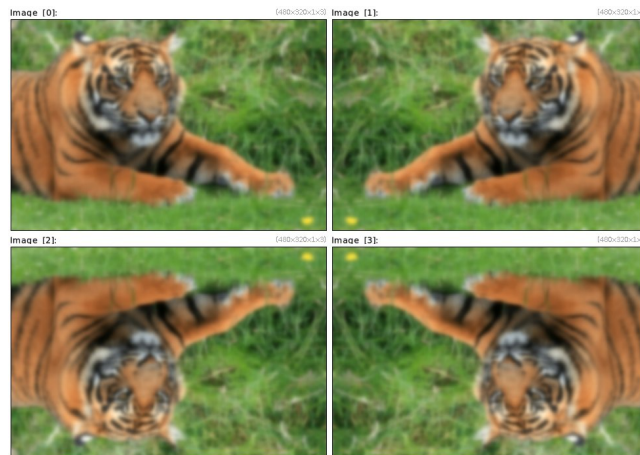
2.13 Control Flow

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 552 : `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 553 : `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'*) . \n).

'nb_threads' must be a power of 2.

Default values:

- `'overlap=0', 'nb_threads=0'`.



Example 554 : `image.jpg +apply_parallel_overlap "smooth 500,0,1",1`

2.13.4 *apply_tiles*

Arguments:

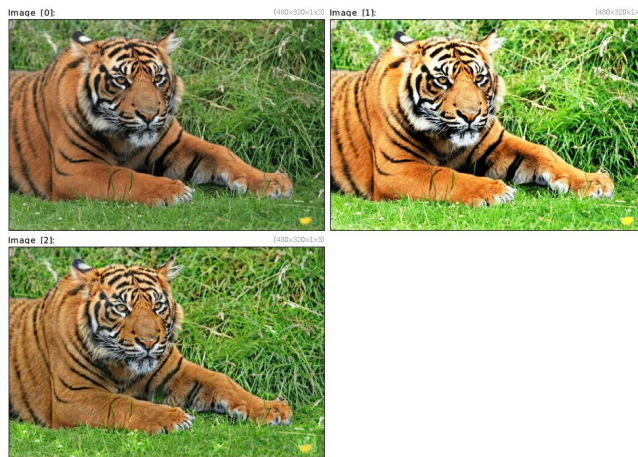
- "command", _tile_width[%]>0, _tile_height[%]>0, _tile_depth[%]>0, _overlap_width[%]>0, _overlap_height[%]>0, _overlap_depth[%]>0, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }

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

(eq. to 'at').

Default values:

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



Example 555 : `image.jpg +equalize[0] 256 +apply_tiles[0] "equalize 256",16,16,1,50%,50%`

2.13.5 *apply_timeout*

Arguments:

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

Apply a command with a timeout.

2.13.6 *check (+)*

Arguments:

- expression

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

If 'expression' is not a math expression, it is regarded as a filename and checked if it exists.

2.13.7 *check3d* (+)

Arguments:

- `_is_full_check={ 0 | 1 }`

Check validity of selected 3D vector objects, and display an error message if one of the selected images is not a valid 3D vector object.

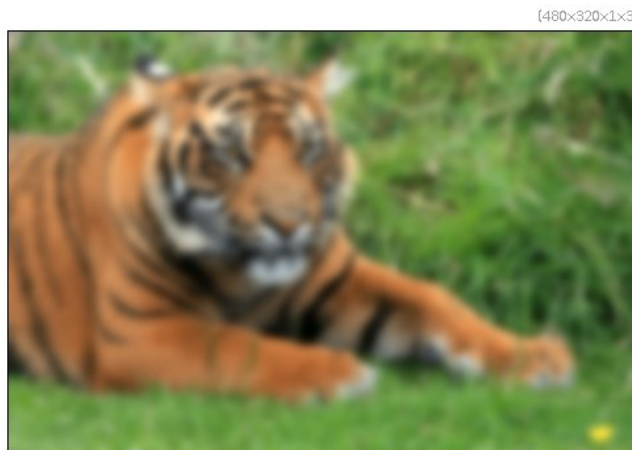
Full 3D object check is slower but more precise.

Default value:

- `'is_full_check=1'`.

2.13.8 *continue* (+)

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



Example 556 : `image.jpg repeat 10 blur 1 if {1==1} continue fi deform 10 done`

2.13.9 *break* (+)

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



Example 557 : `image.jpg repeat 10 blur 1 if {1==1} break fi deform 10 done`

2.13.10 *do* (+)

Start a 'do...while' block.



Example 558 : `image.jpg luminance i={ia+2} do set 255,{u(100)}%,{u(100)}% while {ia<$i}`

2.13.11 *done* (+)

End a 'repeat/for...done' block, and go to associated 'repeat/for' position, if iterations remain.

2.13.12 *elif* (+)**Arguments:**

- `boolean`
- `filename`

Start a 'elif...[else]...fi' 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 }.

2.13.13 *else* (+)

Execute following commands if previous 'if' or 'elif' conditions failed.

2.13.14 *endif* (+)

End a 'if...[elif]...[else]...endif' block.

(eq. to 'fi').

2.13.15 *endlocal* (+)

End a 'local...endlocal' block.

(eq. to 'endl').

2.13.16 *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.

2.13.17 *eval* (+)

Arguments:

- *expression*

Evaluate specified math expression. - If no command selection is specified, the expression is evaluated once and its result is set to status. - If command selection is specified, the evaluation is looped over selected images. Status is not modified.

(in this latter case, 'eval' is similar to 'fill' without assigning the image values).

2.13.18 *exec* (+)

Arguments:

- `_is_verbose={ 0 | 1 }, "command"`

Execute external command using a system call.

The status value is then set to the error code returned by the system call.

If 'is_verbose=1', the executed command is allowed to output on stdout/stderr.

(eq. to 'x').

Default value:

- `'is_verbose=1'`.

2.13.19 *for* (+)

Arguments:

- *condition*

Start a 'for...done' block.



Example 559 : `image.jpg resize2dy 32 400,400,1,3 x=0 for {$x<400} image[1] [0], $x, $x x+=40 done`

2.13.20 *if* (+)

Arguments:

- `boolean`
- `filename`

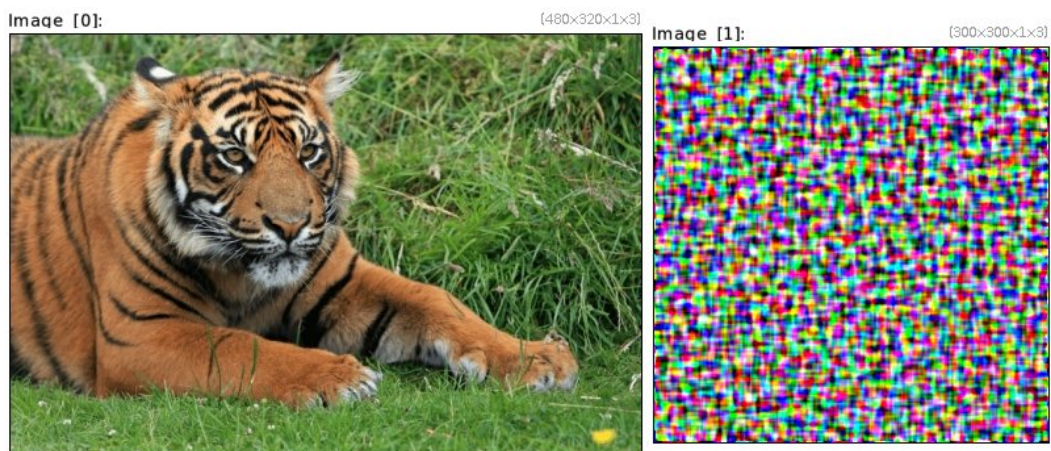
Start a 'if...[elif]...[else]...fi' block and test if specified boolean is true, or if specified filename exists. 'boolean' can be a float number standing for { 0=false | other=true }.



Example 560 : `image.jpg if {ia<64} add 50% elif {ia<128} add 25% elif {ia<192} sub 25% else sub 50% fi cut 0,255`

2.13.21 *local* (+)

Start a 'local...[onfail]...endlocal' block, with selected images. (eq. to '1').



Example 561 : `image.jpg local[] 300,300,1,3 rand[0] 0,255 blur 4 sharpen 1000 endlocal`



Example 562 : `image.jpg +local repeat 3 deform 20 done endlcal`

Tutorial page:

https://gmic.eu/tutorial/_local.shtml

2.13.22 *mutex* (+)

Arguments:

- `index, _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. 'index' designates the mutex index, in [0,255].

Default value:

- `'action=1'`.

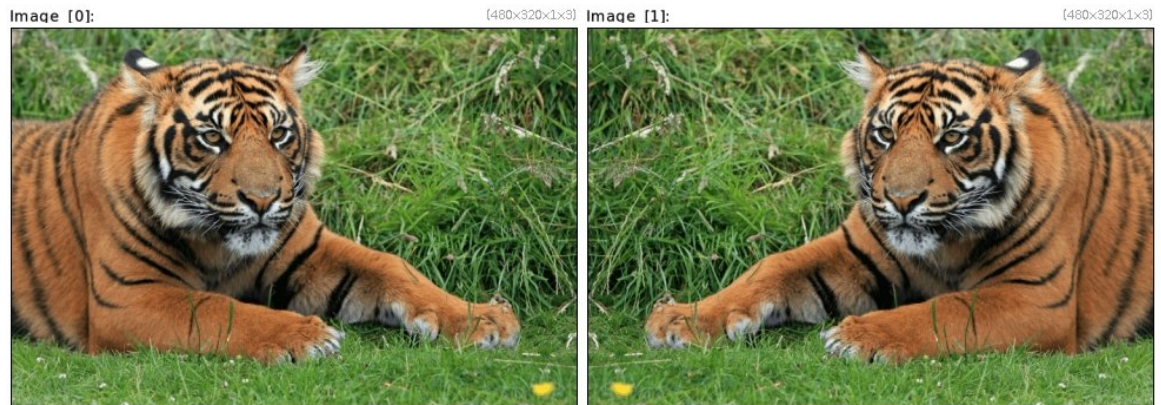
2.13.23 *noarg* (+)

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.

2.13.24 *onfail* (+)

Execute following commands when an error is encountered in the body of the 'local...endlcal' block. The status value is set with the corresponding error message.



Example 563 : `image.jpg +local blur -3 onfail mirror x endl local`

2.13.25 *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 564 : `image.jpg [0] parallel "blur[0] 3", "mirror[1] c"`

2.13.26 *progress* (+)

Arguments:

- `0<=value<=100`
- `-1`

Set the progress index of the current processing pipeline.

This command is useful only when G'MIC is used by an embedding application.

2.13.27 quit (+)

Quit G'MIC interpreter.
(eq. to 'q').

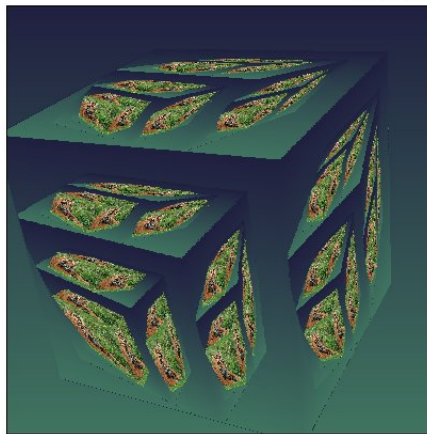
2.13.28 repeat (+)**Arguments:**

- `nb_iterations, _variable_name`

Start iterations of a 'repeat...done' block.



Example 565 : `image.jpg split y repeat $!,n shift[$n] $<,0,0,0,2 done append y`



Example 566 : `image.jpg mode3d 2 repeat 4 imagecube3d rotate3d 1,1,0,40 snapshot3d 400,1.4 done`

Tutorial page:

https://gmic.eu/tutorial/_repeat.shtml

2.13.29 return (+)

Return from current custom command.

2.13.30 *rprogress*

Arguments:

- `0<=value<=100 | -1 | "command", 0<=value_min<=100, 0<=value_max<=100`

Set the progress index of the current processing pipeline (relatively to previously defined progress bounds), or call the specified command with specified progress bounds.

2.13.31 *run*

Arguments:

- `"G'MIC pipeline"`

Run specified G'MIC pipeline.

This is only useful when used from a shell, e.g. to avoid shell substitutions to happen in argument.

2.13.32 *skip* (+)

Arguments:

- `item`

Do nothing but skip specified item.

2.13.33 *status* (+)

Arguments:

- `status_string`

Set the current status. Used to define a returning value from a function.
(*eq. to 'u'*).



Example 567 : `image.jpg command "foo : u0=Dark u1=Bright status ${u{ia}>=128}}" text-outline ${-foo},2,2,23,2,1,255`

2.13.34 *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 568 : `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 569 : `image.jpg array.fade 3,2`

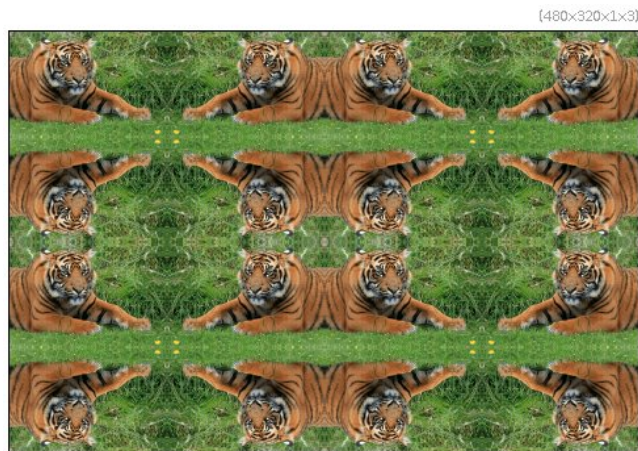
2.14.3 array_mirror**Arguments:**

- $N \geq 0$, `_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 570 : `image.jpg array.mirror 2`

2.14.4 *array_random*

Arguments:

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

Create MdxNd array of tiles from selected MxxNs source arrays.

Default values:

- `'Ns=Ms', 'Md=Ms' and 'Nd=Ns'.`



Example 571 : `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 572 : `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 573: `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 574 : `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 575 : `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 576 : `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 577 : `image.jpg frame.round 10`

2.14.11 *frame_seamless*

Arguments:

- `frame_size>=0, _patch_size>0, _blend_size>=0, _frame_direction={ 0=inner (preserve image size) | 1=outer }`

Insert frame in selected images, so that tiling the resulting image makes less visible seams.

Default values:

- `'patch_size=7', 'blend_size=5' and 'frame_direction=1'.`



Example 578 : `image.jpg +frame_seamless 30 array 2,2`

2.14.12 *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 579 : `image.jpg frame_x 20,255,0,255`

2.14.13 *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 580 : `image.jpg frame_xy 1,1,0 frame_xy 20,10,255,0,255`

2.14.14 *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'.`

2.14.15 *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 581 : `image.jpg frame_y 20,255,0,255`

2.14.16 *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.

Default values:

- 'charset=[ascii charset]', 'analysis_scale=16', 'analysis_smoothness=20', 'synthesis_scale=16' and '_output_ascii_filename=[undefined]'.



Example 582 : image.jpg img2ascii ,

2.14.17 imagegrid**Arguments:**

- $M > 0, N > 0$

Create $M \times N$ image grid from selected images.

Default value:

- 'N=M'.



Example 583 : image.jpg imagegrid 16

2.14.18 *imagegrid_hexagonal*

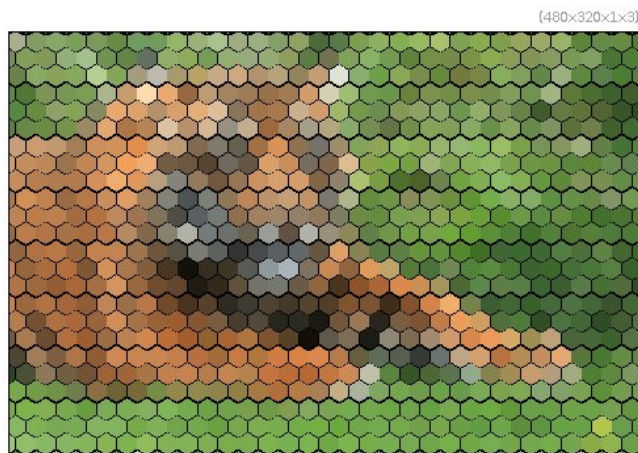
Arguments:

- `_resolution>0, 0<=_outline<=1`

Create hexagonal grids from selected images.

Default values:

- `'resolution=32', 'outline=0.1' and 'is_antialiased=1'.`



Example 584 : `image.jpg imagegrid.hexagonal 24`

2.14.19 *imagegrid_triangular*

Arguments:

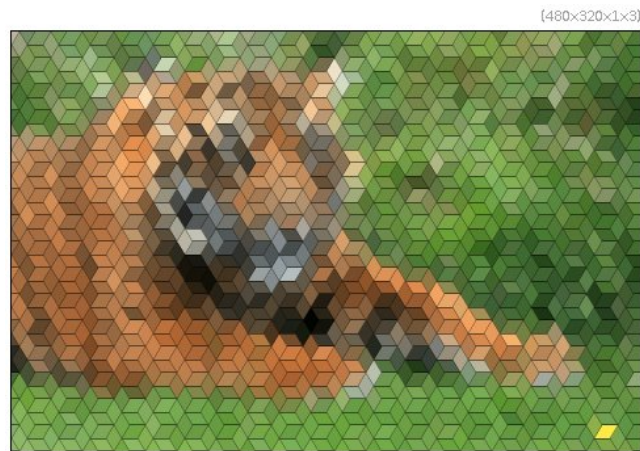
- `pattern_width>=1, _pattern_height>=1, _pattern_type, 0<=_outline_opacity<=1, _outline_color1, ...`

Create triangular grids from selected images.

'pattern type' can be { 0=horizontal | 1=vertical | 2=crossed | 3=cube | 4=decreasing | 5=increasing }.

Default values:

- `'pattern_width=24', 'pattern_height=pattern_width', 'pattern_type=0', 'outline_opacity=0.1' and 'outline_color1=0'.`



Example 585 : `image.jpg imagegrid_triangular 6,10,3,0.5`

2.14.20 *linearize_tiles*

Arguments:

- $M > 0, N > 0$

Linearize MxN tiles on selected images.

Default value:

- 'N=M'.



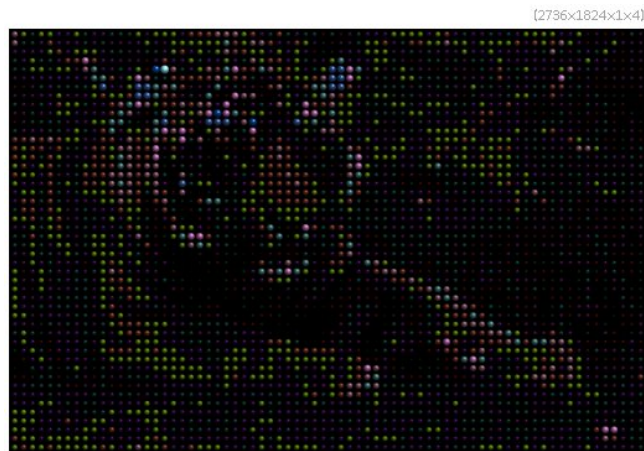
Example 586 : `image.jpg +linearize_tiles 16`

2.14.21 *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 587: `image.jpg resize2dy 48 repeat 16 ball {8+2*$>}, ${-RGB} mul[-1] {(1+$>)/16} done
map_sprites 16`

2.14.22 *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 588: `image.jpg repeat 10 +resize2dx[-1] 75% balance_gamma[-1] ${-RGB} done pack 0`

2.14.23 *puzzle*

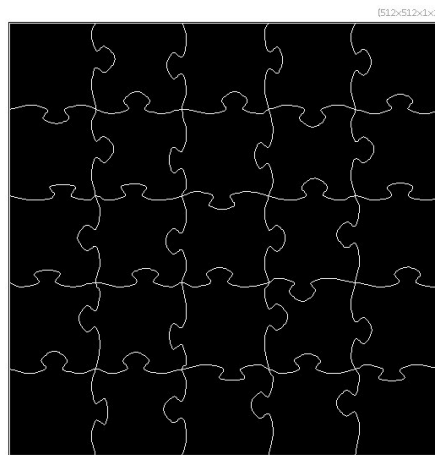
Arguments:

- `_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 589 : puzzle ,

2.14.24 *quadratize tiles*

Arguments:

- `M>0, _N>0`

Quadratize MxN tiles on selected images.

Default value:

- `'N=M'.`



Example 590 : image.jpg +quadratize-tiles 16

2.14.25 *rotate_tiles*

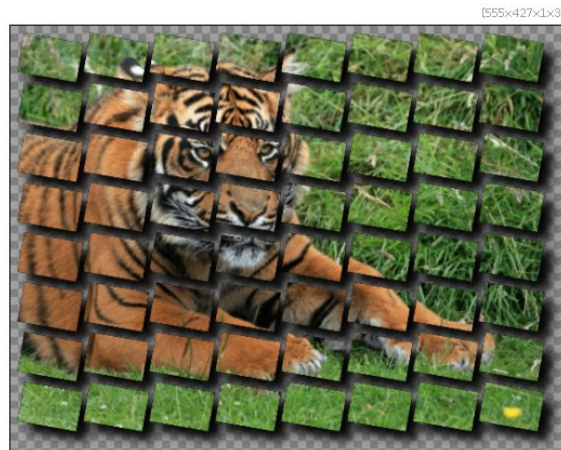
Arguments:

- `angle, _M>0, _N>0`

Apply MxN tiled-rotation effect on selected images.

Default values:

- `'M=8' and 'N=M' .`



Example 591 : `image.jpg to_rgba rotate_tiles 10,8 drop_shadow 10,10 display_rgba`

2.14.26 *shift_tiles*

Arguments:

- `_M>0, _N>0, _amplitude`

Apply MxN tiled-shift effect on selected images.

Default values:

- `'N=M' and 'amplitude=20' .`



Example 592 : `image.jpg +shift_tiles 8,8,10`

2.14.27 *taquin*

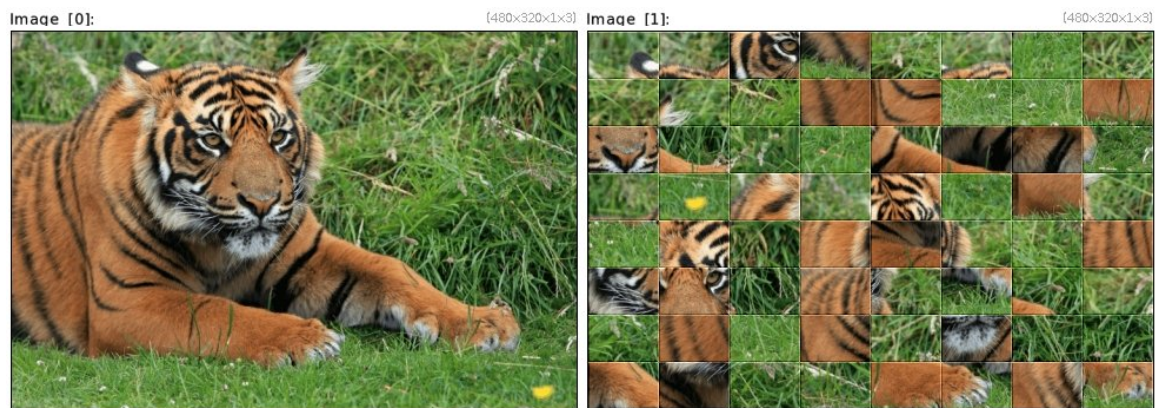
Arguments:

- `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 593 : image.jpg +taquin 8

2.14.28 *tunnel*

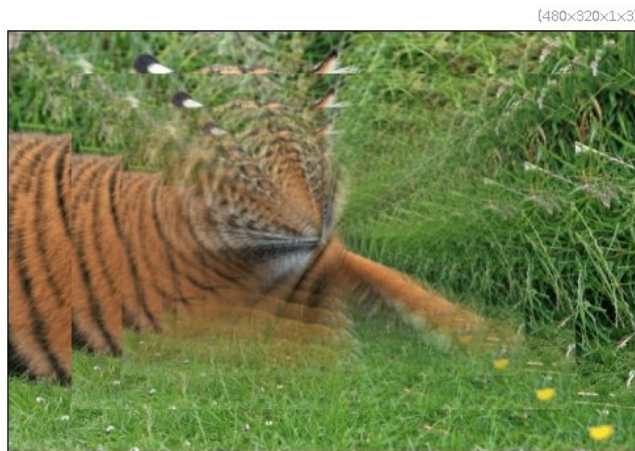
Arguments:

- `_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 594 : image.jpg tunnel 20

2.15 Artistic

2.15.1 *boxfitting*

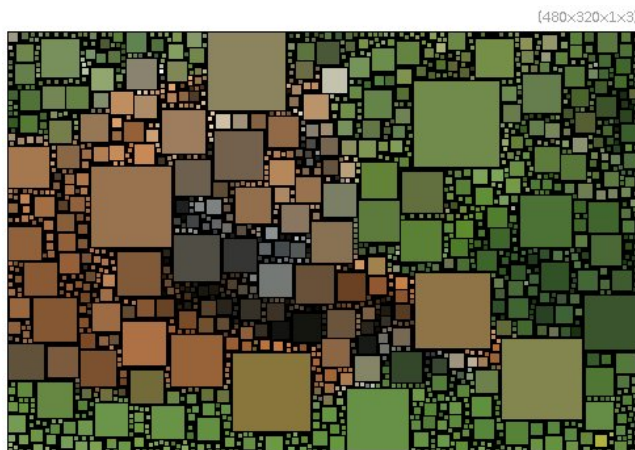
Arguments:

- `_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_attempts=3'.`



Example 595 : image.jpg boxfitting ,

2.15.2 *brushify*

Arguments:

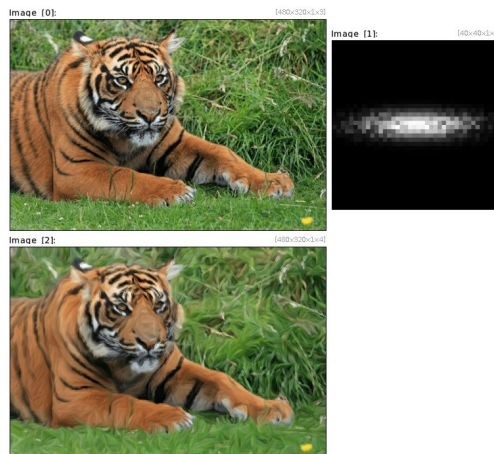
- `[brush], _brush_nb_sizes>=1, 0<=_brush_min_size_factor<=1, _brush_nb_orientations>=1, _brush_light_type, 0<=_brush_light_strength<=1, _brush_opacity, _painting_density[%]>=0, 0<=_painting_contours_coherence<=1, 0<=_painting_orientation_coherence<=1, _painting_coherence_alpha[%]>=0, _painting_coherence_sigma[%]>=0, _painting_primary_angle, 0<=_painting_angle_dispersion<=1`

Apply specified brush to create painterly versions of specified images.

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

Default values:

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



Example 596 : `image.jpg 40,40 gaussian[-1] 8,2 spread[-1] 4,0 +brushify[0] [1]`

2.15.3 *cartoon*

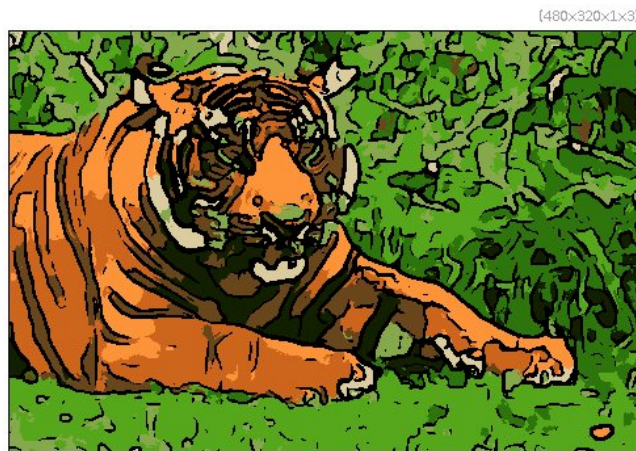
Arguments:

- `_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 597 : `image.jpg cartoon 3,80,15`

2.15.4 *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 598 : `image.jpg +color_ellipses , ,0.15`

2.15.5 *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 599 : `image.jpg cubism` ,

2.15.6 *draw_whirl***Arguments:**

- `_amplitude>=0`

Apply whirl drawing effect on selected images.

Default value:

- `'amplitude=100'`.



Example 600 : `image.jpg draw.whirl` ,

2.15.7 *drawing*

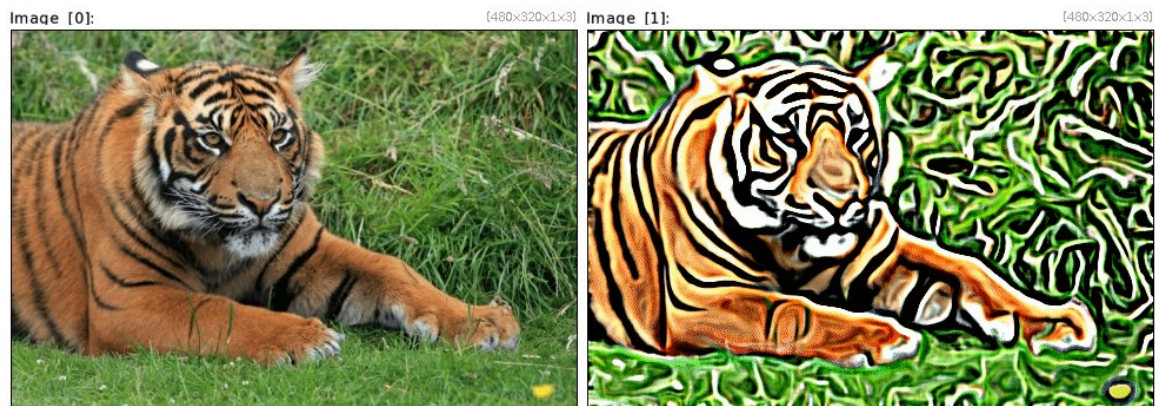
Arguments:

- `_amplitude>=0`

Apply drawing effect on selected images.

Default value:

- `'amplitude=200'`.



Example 601 : `image.jpg +drawing ,`

2.15.8 *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 602 : `image.jpg drop-shadow 10,20,5,0.5 expand-xy 20,0 display-rgba`

2.15.9 *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 603 : `image.jpg +ellipsionism ,`

2.15.10 *fire_edges*

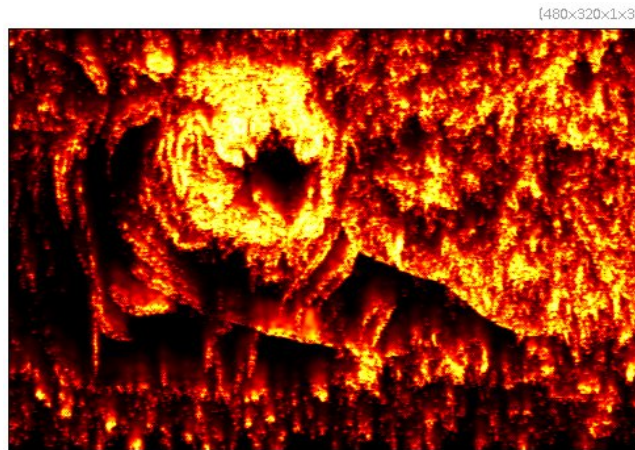
Arguments:

- `_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 604 : `image.jpg fire.edges ,`

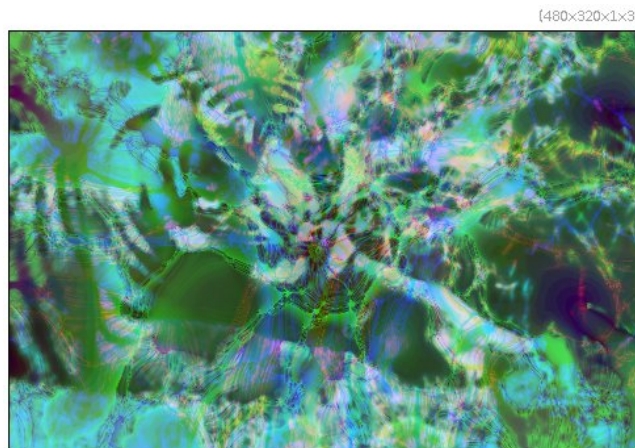
2.15.11 *fractalize***Arguments:**

- `0<=detail_level<=1`

Randomly fractalize selected images.

Default value:

- `'detail_level=0.8'`



Example 605 : `image.jpg fractalize ,`

2.15.12 *glow*

Arguments:

- `_amplitude>=0`

Add soft glow on selected images.

Default value:

- `'amplitude=1%'`.



Example 606 : `image.jpg glow ,`

2.15.13 *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 607 : `image.jpg halftone ,`

2.15.14 *hardsketchbw*

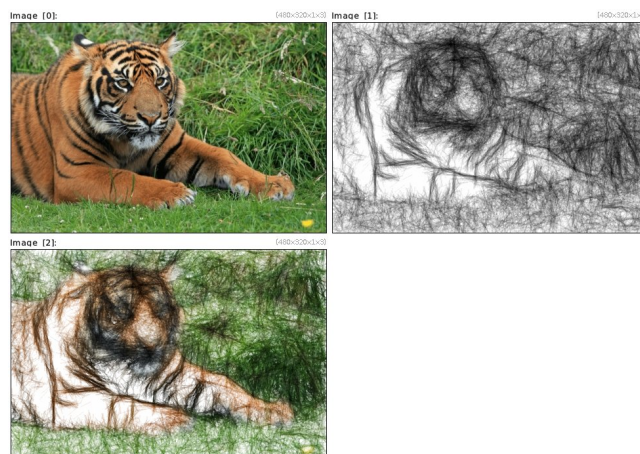
Arguments:

- `_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 608 : `image.jpg +hardsketchbw 200,70,0.1,10 median[-1] 2 +local reverse blur[-1] 3
blend[-2,-1] overlay endlocal`

2.15.15 *hearts*

Arguments:

- `_density>=0`

Apply heart effect on selected images.

Default value:

- `'density=10'`.



Example 609 : `image.jpg +hearts` ,

2.15.16 *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 610 : `image.jpg +houghsketchbw` ,

2.15.17 *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 611 : `image.jpg +lightrays , + cut 0,255`

2.15.18 *light_relief*

Arguments:

- `_ambient_light, _specular_lightness, _specular_size, _light_smoothness, _darkness, _x1, _y1, _z1, _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', 'x1=0.2', 'y1=z1=0.5',`

`'zscale=1', 'opacity=1' and 'opacity_is_heightmap=0'.`



Example 612 : `image.jpg blur 2 light-relief 0.3,4,0.1,0`

2.15.19 *linify*

Arguments:

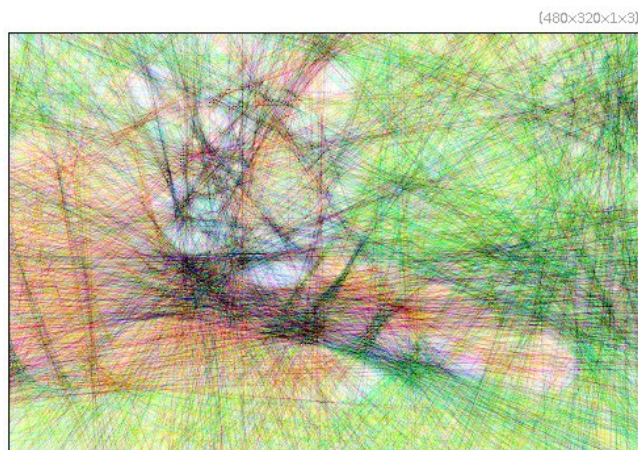
- `0<=_density<=100, _spreading>=0, _resolution[%]>0, _line_opacity>=0, _line_precision>0, mode={ 0=subtractive | 1=additive }`

Apply linify effect on selected images.

The algorithm is inspired from the one described on the webpage '<http://linify.me/about>'.

Default values:

- `'density=50', 'spreading=2', 'resolution=40%', 'line_opacity=10', 'line_precision=24' and 'mode=0'.`



Example 613 : `image.jpg linify 40`

2.15.20 *mosaic*

Arguments:

- `0<=_density<=100`

Create random mosaic from selected images.

Default values:

- `'density=30'`.



Example 614 : `image.jpg mosaic ,`

2.15.21 *old_photo*

Apply old photo effect on selected images.



Example 615 : `image.jpg old_photo`

2.15.22 *pencilbw*

Arguments:

- `_size>=0, _amplitude>=0`

Apply B&W pencil effect on selected images.

Default values:

- 'size=0.3' and 'amplitude=60'.



Example 616 : `image.jpg pencilbw ,`

2.15.23 pixelsort**Arguments:**

- `_ordering={ + | - }, _axis={ x | y | z | xy | yx }, _[sorting_criterion], _[mask]`

Apply a 'pixel sorting' algorithm on selected images, as described in the page : <http://satyarth.me/articles/pixel-sorting/>

Default values:

- 'ordering=+', 'axis=x' and 'sorting_criterion=mask=(undefined)'.



Example 617 : `image.jpg +norm +ge[-1] 30% +pixelsort[0] +,y,[1],[2]`

2.15.24 *polaroid*

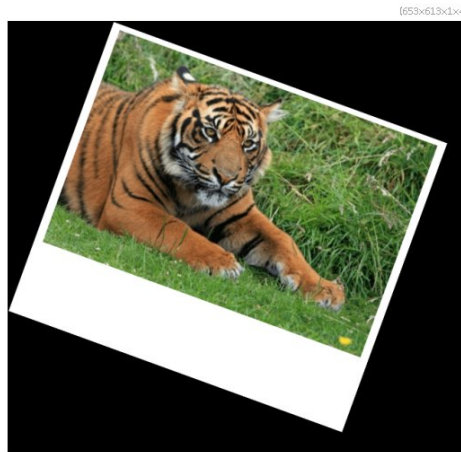
Arguments:

- `_size1>=0, _size2>=0`

Create polaroid effect in selected images.

Default values:

- `'size1=10'` and `'size2=20'`.



Example 618 : `image.jpg to_rgba polaroid 5,30 rotate 20 drop_shadow ,`

2.15.25 *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 619 : `image.jpg polygonize 300,1%,0.1%,3%,3%`

2.15.26 *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 620 : `image.jpg +poster_edges ,`

2.15.27 *poster_hope*

Arguments:

- `_smoothness>=0`

Apply Hope stencil poster effect on selected images.

Default value:

- `'smoothness=3'`.



Example 621 : `image.jpg poster.hope ,`

2.15.28 *rodilius*

Arguments:

```
• 0<=_amplitude<=100, _0<=_thickness<=100, _sharpness>=0, _nb_orientations>0,
_offset, _color_mode={ 0=darker | 1=brighter
}
```

Apply rodilius (fractalus-like) filter on selected images.

Default values:

- `'amplitude=10', 'thickness=10', 'sharpness=400', 'nb_orientations=7', 'offset=0' and 'color_mode=1'`.



Example 622: `image.jpg rodilius 12,10,300,10 normalize-local 10,6`
[480x320x1x3]



Example 623: `image.jpg normalize-local 10,16 rodilius 10,4,400,16 smooth 60,0,1,1,4`
`normalize-local 10,16`

2.15.29 *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 624: `image.jpg stained.glass 20%,0.1`

2.15.30 *stars*

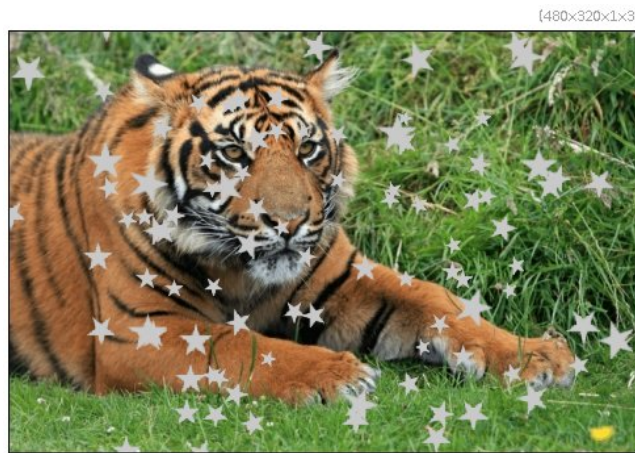
Arguments:

- `_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 625 : image.jpg stars ,

2.15.31 *sketchbw*

Arguments:

- `_nb_angles>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_angles=2', 'start_angle=45', 'angle_range=180', 'length=30', 'threshold=3', 'opacity=0.03',`

`'bgfactor=0', 'density=0.6', 'sharpness=0.1', 'anisotropy=0.6', 'smoothness=0.25', 'coherence=1', 'is_boost=0' and 'is_curved=1'.`



Example 626 : `image.jpg +sketchbw 1 reverse blur[-1] 3 blend[-2,-1] overlay`

2.15.32 *sponge*

Arguments:

- `_size>0`

Apply sponge effect on selected images.

Default value:

- `'size=13'`



Example 627 : `image.jpg +sponge ,`

2.15.33 *stencil*

Arguments:

- `_radius[%]>=0, _smoothness>=0, _iterations>=0`

Apply stencil filter on selected images.

Default values:

- 'radius=3', 'smoothness=1' and 'iterations=8'.



Example 628 : image.jpg stencil 1,10,3

2.15.34 stencilbw**Arguments:**

- _edges>=0, _smoothness>=0

Apply B&W stencil effect on selected images.

Default values:

- 'edges=15' and 'smoothness=10'.



Example 629 : image.jpg +stencilbw 40,4

2.15.35 tetris**Arguments:**

- `_scale>0`

Apply tetris effect on selected images.

Default value:

- `'scale=10'`.



Example 630 : `image.jpg +tetris 10`

2.15.36 *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 631 : `image.jpg warhol 3,3,3,40`

2.15.37 *weave*

Arguments:

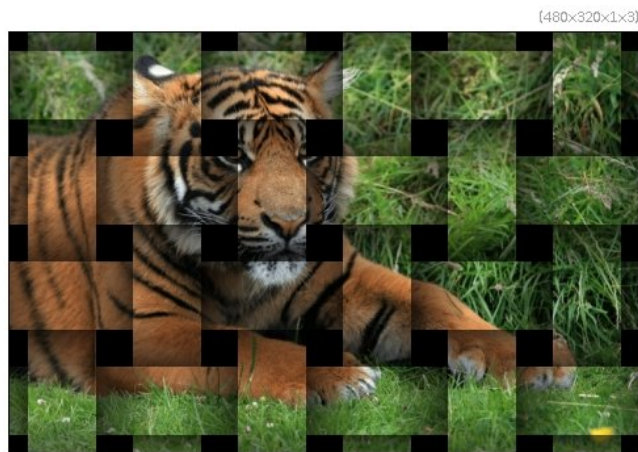
- `_density>=0, 0<=_thickness<=100, 0<=_shadow<=100, _shading>=0, _fibers_amplitude>=0, _fibers_smoothness>=0, _angle, -1<=_x_curvature<=1, -1<=_y_curvature<=1`

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 632 : image.jpg weave ,

2.15.38 *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 633 : `image.jpg +whirls ,`

2.16 Warpings

2.16.1 *deform*

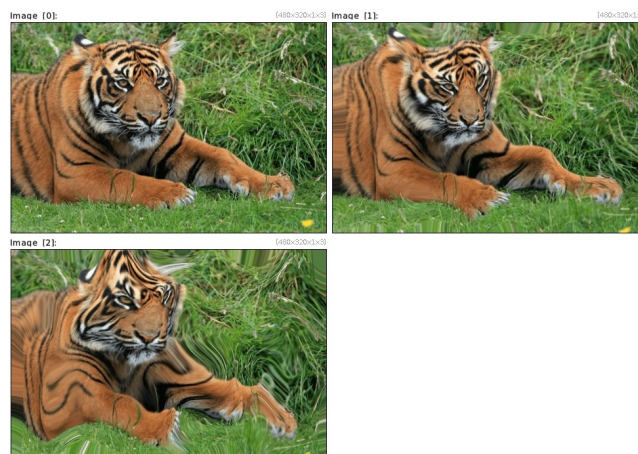
Arguments:

- `_amplitude>=0, _interpolation`

Apply random smooth deformation on selected images.
'interpolation' can be { 0=none | 1=linear | 2=bicubic }.

Default value:

- `'amplitude=10'`.



Example 634 : `image.jpg +deform[0] 10 +deform[0] 20`

2.16.2 *euclidean2polar*

Arguments:


```
• _center_x[_center_y[_stretch_factor>0,_boundary_conditions={ 0=dirichlet
| 1=neumann | 2=periodic | 3=mirror }
```

Apply euclidean to polar transform on selected images.

Default values:

- 'center_x=center_y=50%', 'stretch_factor=1' and 'boundary_conditions=1'.



Example 635 : image.jpg +euclidean2polar ,

2.16.3 *equirectangular2nadirzenith*

Transform selected equirectangular images to nadir/zenith rectilinear projections.

2.16.4 *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 636 : image.jpg +fisheye ,

2.16.5 *flower*

Arguments:

- `_amplitude, _frequency, _offset_r[_], _angle, _center_x[_], _center_y[_],`
`_boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic`
`| 3=mirror}`

Apply flower deformation on selected images.

Default values:

- `'amplitude=30', 'frequency=6', 'offset_r=0', 'angle=0',`
`'center_x=center_y=50%' and 'boundary_conditions=3'.`



Example 637 : `image.jpg flower ,`

2.16.6 *kaleidoscope*

Arguments:

- `_center_x[_], _center_y[_], _radius, _angle, _boundary_conditions={ 0=dirichlet`
`| 1=neumann | 2=periodic | 3=mirror }`

Create kaleidoscope effect from selected images.

Default values:

- `'center_x=center_y=50%', 'radius=100', 'angle=30' and 'boundary_conditions=3'.`



Example 638 : `image.jpg +kaleidoscope ,`

2.16.7 *map_sphere*

Arguments:

- `_width>0, _height>0, _radius, _dilation>0, _fading>=0, _fading_power>=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 639 : `image.jpg map_sphere ,`

2.16.8 *nadirzenith2equirectangular*

Transform selected nadir/zenith rectilinear projections to equirectangular images.

2.16.9 *polar2euclidean*

Arguments:

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

Apply euclidean to polar transform on selected images.

Default values:

- `'center_x=center_y=50%', 'stretch_factor=1' and 'boundary_conditions=1'.`



Example 640 : `image.jpg +euclidean2polar ,`

2.16.10 *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 641 : `image.jpg +raindrops ,`

2.16.11 *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 642 : image.jpg +ripple ,

2.16.12 *rotoidoscope*

Arguments:

```
• _center_x[_], _center_y[_], _tiles>0, _smoothness[_]>=0, _boundary_conditions={
0=dirichlet | 1=neumann | 2=periodic | 3=mirror }
```

Create rotational kaleidoscope effect from selected images.

Default values:

- 'center_x=center_y=50%', 'tiles=10', 'smoothness=1' and 'boundary_conditions=3'.



Example 643 : `image.jpg +rotoidoscope ,`

2.16.13 *spherize*

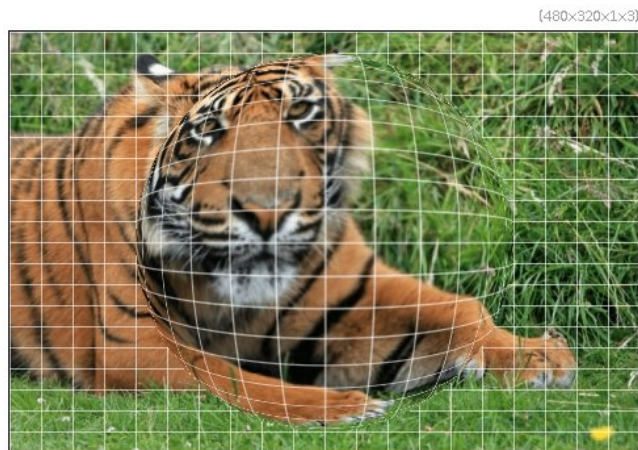
Arguments:

- `_radius[%]>=0, _strength, _smoothness[%]>=0, _center_x[_], _center_y[_], _ratio_x/y>0, _angle, _interpolation`

Apply spherize effect on selected images.

Default values:

- `'radius=50%', 'strength=1', 'smoothness=0', 'center_x=center_y=50%', 'ratio_x/y=1', 'angle=0' and 'interpolation=1'.`



Example 644 : `image.jpg grid 5%,5%,0,0,0.6,255 spherize ,`

2.16.14 *symmetrize*

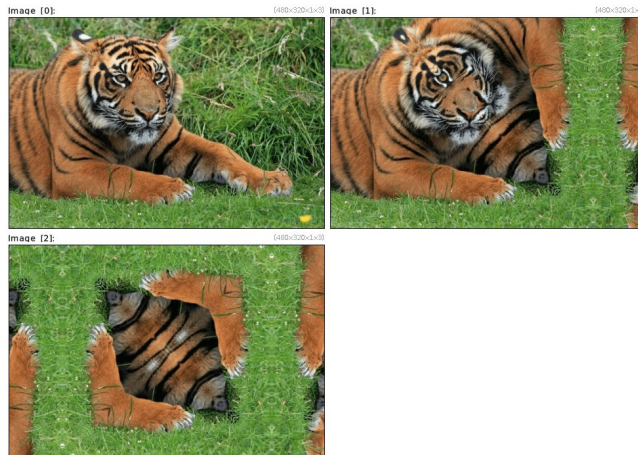
Arguments:

- `-x[_], -y[_], -angle, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }, _is_antisymmetry={ 0 | 1 }, _swap_sides={ 0 | 1 }`

Symmetrize selected images regarding specified axis.

Default values:

- `'x=y=50%', 'angle=90', 'boundary_conditions=3', 'is_antisymmetry=0'` and `'swap_sides=0'`.



Example 645 : `image.jpg +symmetrize 50%,50%,45 +symmetrize[-1] 50%,50%,-45`

2.16.15 *transform_polar*

Arguments:

- `"expr_radius",_"expr_angle",_center_x[_],_center_y[_],_boundary_conditions={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_conditions=1'`.



Example 646 : `image.jpg +transform.polar[0] R*(r/R)^2,a +transform.polar[0] r,2*a`

2.16.16 *twirl*

Arguments:

- `_amplitude, _center_x[%], _center_y[%], _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

Apply twirl deformation on selected images.

Default values:

- `'amplitude=1', 'center_x=center_y=50%' and 'boundary_conditions=3'.`



Example 647: `image.jpg twirl 0.6`

2.16.17 *warp perspective*

Arguments:

- `_x-angle, _y-angle, _zoom>0, _x-center, _y-center, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

Warp selected images with perspective deformation.

Default values:

- `'x-angle=1.5', 'y-angle=0', 'zoom=1', 'x-center=y-center=50' and 'boundary_conditions=2'.`



Example 648 : `image.jpg warp-perspective ,`

2.16.18 *water*

Arguments:

- `_amplitude, _smoothness >= 0, _angle`

Apply water deformation on selected images.

Default values:

- `'amplitude=30', 'smoothness=1.5' and 'angle=45'.`



Example 649 : `image.jpg water ,`

2.16.19 *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 650 : `image.jpg wave` ,

2.16.20 *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 651 : `image.jpg +wind` ,

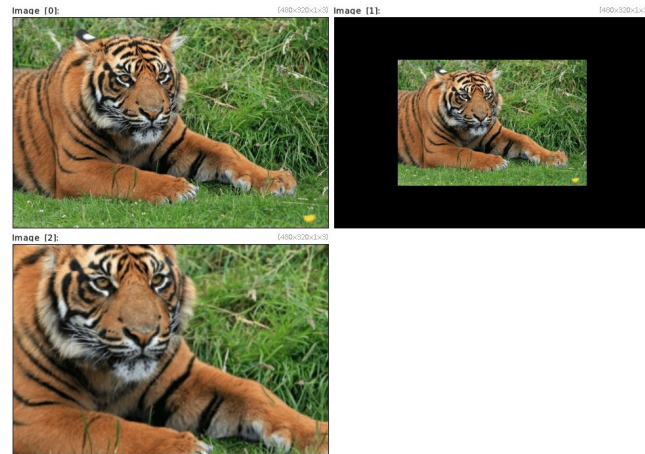
2.16.21 *zoom***Arguments:**

- `_factor, _cx, _cy, _cz, _boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

Apply zoom factor to selected images.

Default values:

- `'factor=1', 'cx=cy=cz=0.5' and 'boundary_conditions=0'.`



Example 652 : `image.jpg +zoom[0] 0.6 +zoom[0] 1.5`

2.17 Degradations

2.17.1 cracks

Arguments:

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

Draw random cracks on selected images with specified color.

Default values:

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



Example 653 : `image.jpg +cracks ,`

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 654 : `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 655 : `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 656 : `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 657 : `image.jpg +scanlines ,`

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 658: `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 659: `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 660 : `image.jpg +spread 3`

2.17.9 *stripes_y*

Arguments:

- `_frequency>=0`

Add vertical stripes to selected images.

Default value:

- `'frequency=10'`.



Example 661 : `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 662 : `image.jpg +texturize_canvas ,`

2.17.11 *texturize_paper*

Add paper texture to selected images.



Example 663 : `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 664 : 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 665 : image.jpg watermark_visible ,0.7

2.18 Blending and Fading

2.18.1 *blend*

Arguments:

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

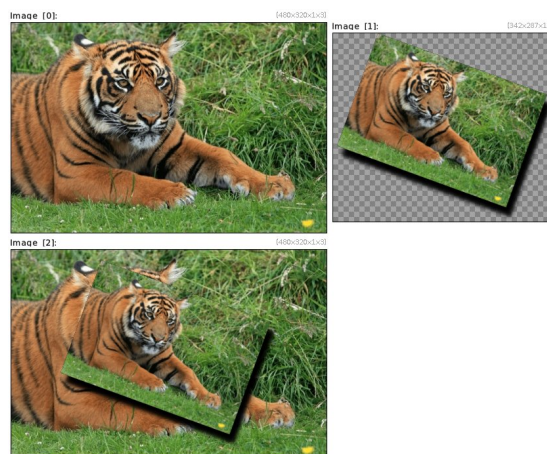
Blend selected G,GA,RGB or RGBA images by specified layer or blend all selected images together, using specified blending mode.

'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 | shapeareamax | shapeareamax0 | shapeareamin | shapeareamin0 | shapeaverage | shapeaverage0 | shapemedian | shapemedian0 | shapemin | shapemin0 | shapemax | shapemax0 | softburn | softdodge | softlight | stamp | subtract | value | vividlight | xor }.

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

Default values:

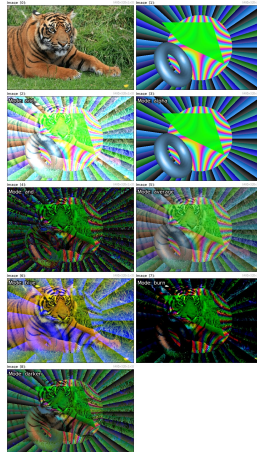
- 'blending_mode=alpha', 'opacity=1' and 'selection_is=0'.



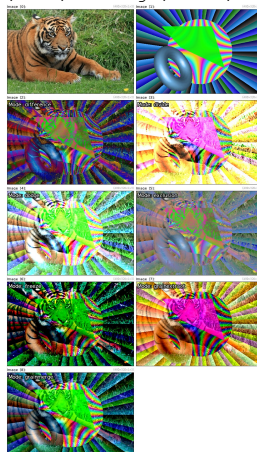
Example 666 : image.jpg +drop.shadow , resize2dy[-1] 200 rotate[-1] 20 +blend alpha
display_rgba[-2]



Example 667 : `image.jpg testimage2d {w},{h} blend overlay`

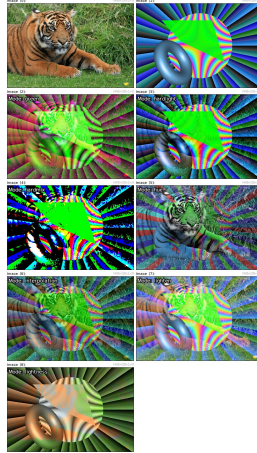


Example 668 : `command "ex : $"=arg repeat $"% +blend[0,1] ${arg{${>+1}}} text.outline[-1]
Mode:\ " \ "${arg{${>+1}}},2,2,23,2,1,255 done" image.jpg testimage2d {w},{h} ex
add,alpha,and,average,blue,burn,darken`

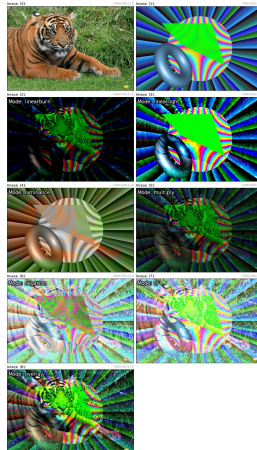


Example 669 : `command "ex : $"=arg repeat $"% +blend[0,1] ${arg{${>+1}}} text.outline[-1]
Mode:\ " \ "${arg{${>+1}}},2,2,23,2,1,255 done" image.jpg testimage2d {w},{h} ex`

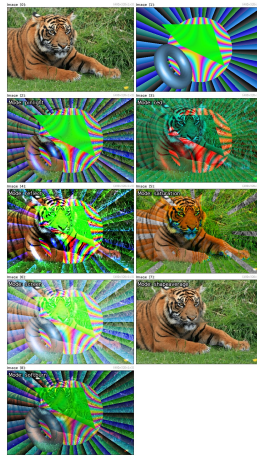
difference,divide,dodge,exclusion,freeze,grainextract,grainmerge



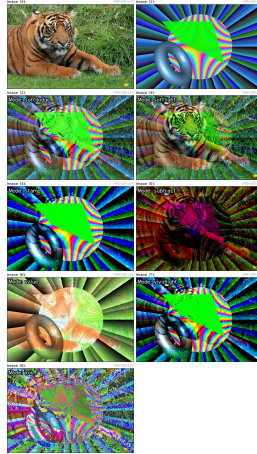
Example 670 : command "ex : \$"=arg repeat \$" "% +blend[0,1] \${arg{\$>+1}} text_outline[-1]
 Mode:\ " \ "\${arg{\$>+1}},2,2,23,2,1,255 done" image.jpg testimage2d {w},{h} ex
 green,hardlight,hardmix,hue,interpolation,lighten,lightness



Example 671 : command "ex : \$"=arg repeat \$" "% +blend[0,1] \${arg{\$>+1}} text_outline[-1]
 Mode:\ " \ "\${arg{\$>+1}},2,2,23,2,1,255 done" image.jpg testimage2d {w},{h} ex
 linearburn,linearlight,luminance,multiply,negation,or,overlay



Example 672 : command "ex : \$"=arg repeat \$"% +blend[0,1] \${arg{\$>+1}} text.outline[-1]
 Mode:\ " \ "\${arg{\$>+1}},2,2,23,2,1,255 done" image.jpg testimage2d {w},{h} ex
 pinlight,red,reflect,saturation,screen,shapeaverage,softburn



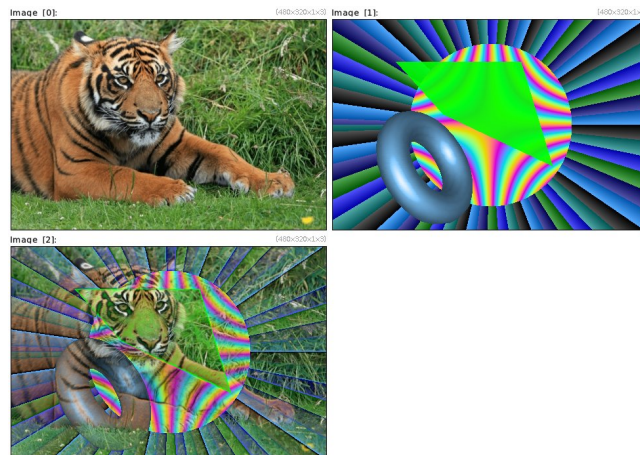
Example 673 : command "ex : \$"=arg repeat \$"% +blend[0,1] \${arg{\$>+1}} text.outline[-1]
 Mode:\ " \ "\${arg{\$>+1}},2,2,23,2,1,255 done" image.jpg testimage2d {w},{h} ex
 softdodge,softlight,stamp,subtract,value,vividlight,xor

2.18.2 *blend_edges*

Arguments:

- smoothness[%]>=0

Blend selected images together using 'edges' mode.



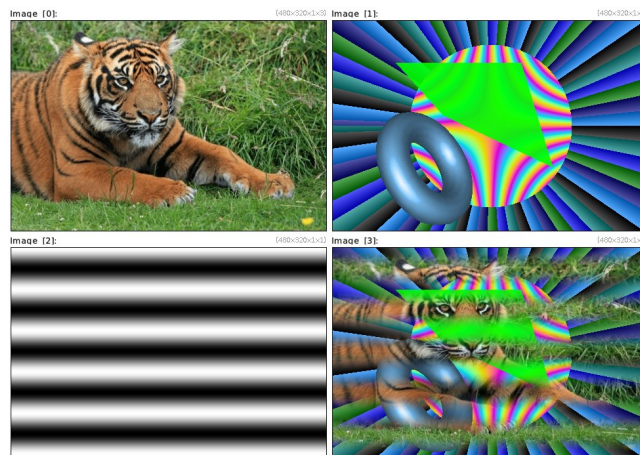
Example 674 : `image.jpg testimage2d {w},{h} +blend_edges 0.8`

2.18.3 *blend_fade*

Arguments:

- `[fading_shape]`

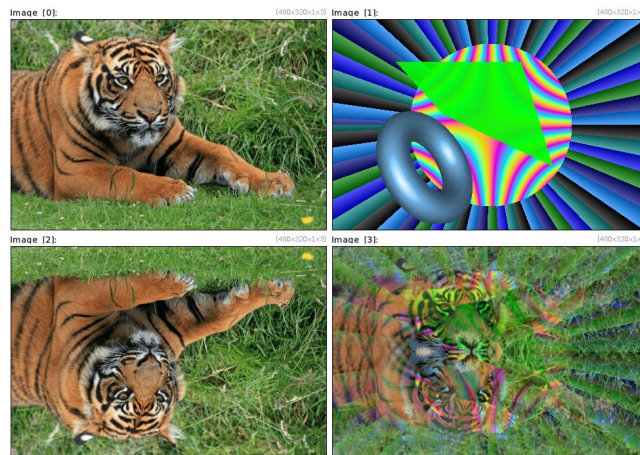
Blend selected images together using specified fading shape.



Example 675 : `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 676 : `image.jpg testimage2d {w},{h} +mirror[0] y +blend.median`

2.18.5 *blend_seamless*

Arguments:

- `_is_mixed_mode={ 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*

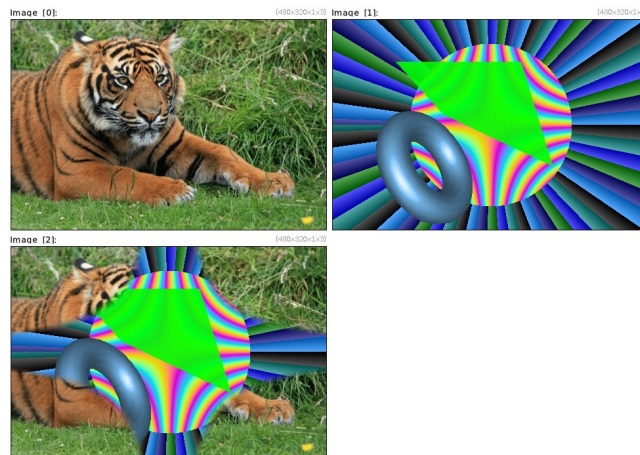
Arguments:

- `0<=_start<=100,0<=_end<=100`

Create diamond fading from selected images.

Default values:

- `'_start=80' and '_end=90'`.



Example 677 : `image.jpg testimage2d {w},{h} +fade.diamond 80,85`

2.18.7 *fade_linear*

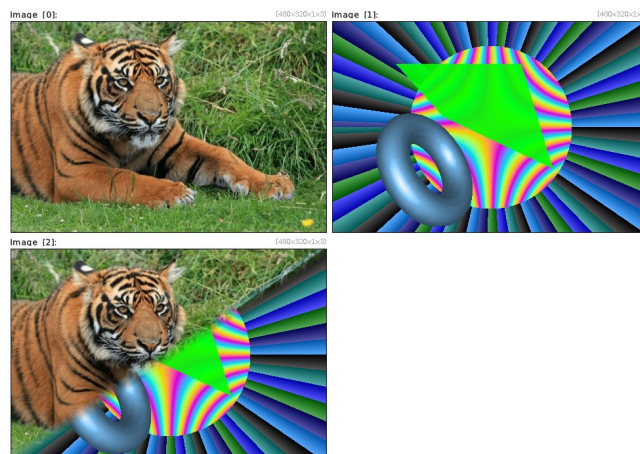
Arguments:

- `_angle, 0<=_start<=100, 0<=_end<=100`

Create linear fading from selected images.

Default values:

- `'angle=45', 'start=30' and 'end=70'.`



Example 678 : `image.jpg testimage2d {w},{h} +fade.linear 45,48,52`

2.18.8 *fade_radial*

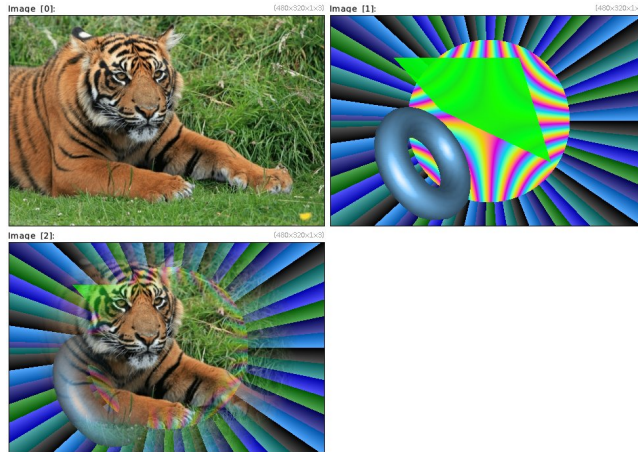
Arguments:

- `0<=_start<=100, 0<=_end<=100`

Create radial fading from selected images.

Default values:

- 'start=30' and 'end=70'.



Example 679 : `image.jpg testimage2d {w},{h} +fade-radial 30,70`

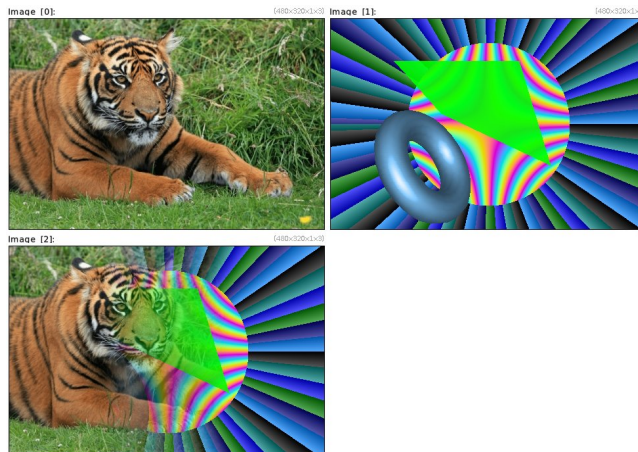
2.18.9 *fade_x***Arguments:**

- `0<=_start<=100, 0<=_end<=100`

Create horizontal fading from selected images.

Default values:

- 'start=30' and 'end=70'.



Example 680 : `image.jpg testimage2d {w},{h} +fade-x 30,70`

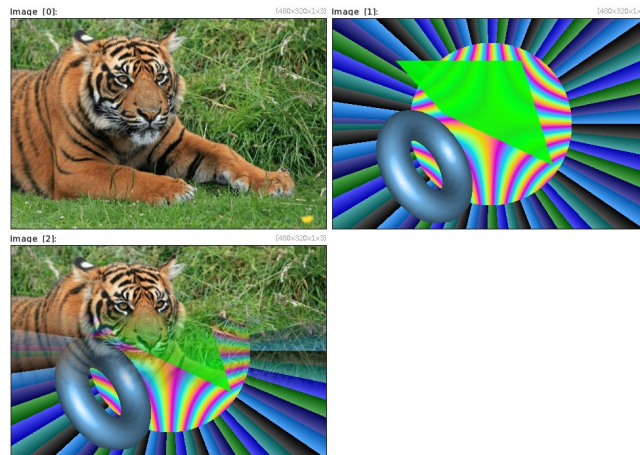
2.18.10 *fade_y***Arguments:**

- `0<=_start<=100,0<=_end<=100`

Create vertical fading from selected images.

Default values:

- `'start=30'` and `'end=70'`.



Example 681 : `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:

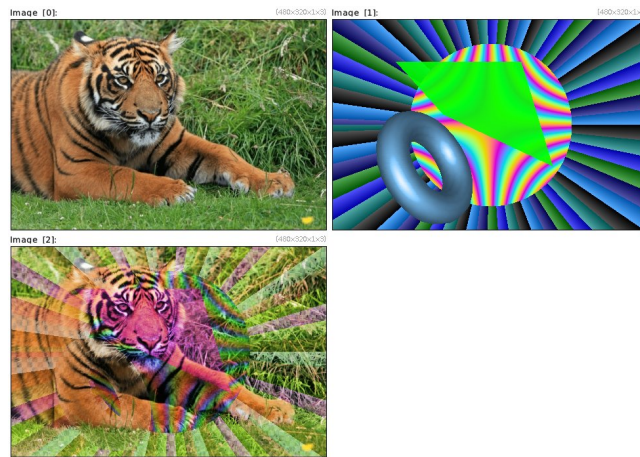
- `[base_image],_opacity_gain>=1`

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 682 : `image.jpg testimage2d {w},{h} +sub_alpha[0] [1] display_rgba`

2.19 Image Sequences and Videos

2.19.1 *animate*

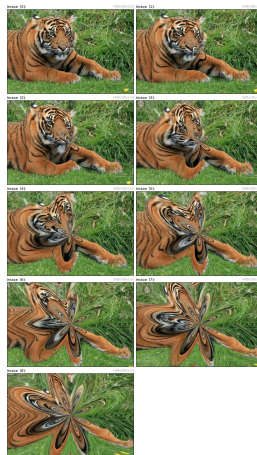
Arguments:

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

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

Default value:

- `'delay=30'`.



Example 683 : `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:

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

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

If a display window is opened, rendered frames are displayed in it during processing.

The output filename may have extension '.avi' (saved as a video), or any other usual image file extension (saved as a sequence of images).

Default values:

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

2.19.4 *apply_video*

Arguments:

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

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

If a display window is opened, rendered frames are displayed in it during processing.

The output filename may have extension '.avi' (saved as a video), or any other usual image file extension (saved as a sequence of images).

Default values:

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

2.19.5 *average_files*

Arguments:

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

Average specified input image files, in a streamed way.

If a display window is opened, rendered frames are displayed in it during processing.

The output filename may have extension '.avi' (saved as a video), or any other usual image file extension (saved as a sequence of images).

Default values:

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

2.19.6 *average_video*

Arguments:

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

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

If a display window is opened, rendered frames are displayed in it during processing.

The output filename may have extension '.avi' (saved as a video), or any other usual image file extension (saved as a sequence of images).

Default values:

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

2.19.7 *fade_files*

Arguments:

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

Generate a temporal fading from specified input image files, in a streamed way.

If a display window is opened, rendered frames are displayed in it during processing.

The output filename may have extension '.avi' (saved as a video), or any other usual image file extension (saved as a sequence of images).

Default values:

- 'nb_inner_frames=10', 'first_frame=0', 'last_frame=-1', 'frame_step=1' and 'output_filename=(undefined)' .

2.19.8 *fade_video*

Arguments:

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

Create a temporal fading sequence from specified input video file, in a streamed way.

If a display window is opened, rendered frames are displayed in it during processing.

Default values:

- `'nb_inner_frames=10', 'first_frame=0', 'last_frame=-1', 'frame_step=1'` and `'output_filename=(undefined)'`.

2.19.9 *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'`.

2.19.10 *median_files***Arguments:**

- `"filename_pattern", _first_frame>=0, _last_frame={ >=0 | -1=last }, _frame_step>=1, _frame_rows[%]>=1, _is_fast_approximation={ 0 | 1 }`

Compute the median frame of specified input image files, in a streamed way.

If a display window is opened, rendered frame is displayed in it during processing.

Default values:

- `'first_frame=0', 'last_frame=-1', 'frame_step=1', 'frame_rows=20%'` and `'is_fast_approximation=0'`.

2.19.11 *median_video***Arguments:**

- `video_filename, _first_frame>=0, _last_frame={ >=0 | -1=last }, _frame_step>=1, _frame_rows[%]>=1, _is_fast_approximation={ 0 | 1 }`

Compute the median of all frames of an input video file, in a streamed way.

If a display window is opened, rendered frame is displayed in it during processing.

Default values:

- `'first_frame=0', 'last_frame=-1', 'frame_step=1', 'frame_rows=100%'` and `'is_fast_approximation=1'`.

2.19.12 *morph*

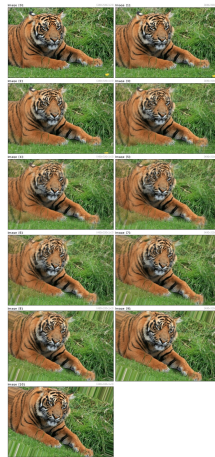
Arguments:

- `nb_inner_frames>=1, _smoothness>=0, _precision>=0`

Create morphing sequence between selected images.

Default values:

- `'smoothness=0.1'` and `'precision=4'`.



Example 684 : `image.jpg +rotate 20,1,1,50%,50% morph 9`

2.19.13 *morph_files*

Arguments:

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

Generate a temporal morphing from specified input image files, in a streamed way.

If a display window is opened, rendered frames are displayed in it during processing.

The output filename may have extension `'avi'` (saved as a video), or any other usual image file extension (saved as a sequence of images).

Default values:

- `'nb_inner_frames=10', 'smoothness=0.1', 'precision=4', 'first_frame=0', 'last_frame=-1', 'frame_step=1' and 'output_filename=(undefined)'`.

2.19.14 *morph_video*

Arguments:

```

• video_filename, _nb_inner_frames>0, _smoothness>=0, _precision>=0,
_first_frame>=0, _last_frame={ >=0 | -1=last
}, _frame_step>=1, _output_filename

```

Generate a temporal morphing from specified input video file, in a streamed way.

If a display window is opened, rendered frames are displayed in it during processing.

The output filename may have extension '.avi' (saved as a video), or any other usual image file extension (saved as a sequence of images).

Default values:

```

• 'nb_inner_frames=10', 'smoothness=0.1', 'precision=4', 'first_frame=0',
'last_frame=-1', 'frame_step=1' and 'output_filename=(undefined)'.

```

2.19.15 *register_nonrigid*

Arguments:

```

• [destination], _smoothness>=0, _precision>0, _nb_scale>=0

```

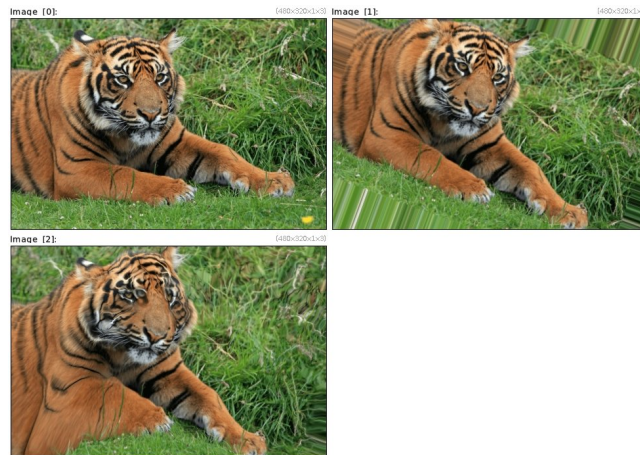
Register selected source images with specified destination image, using non-rigid warp.

Default values:

```

• 'smoothness=0.2', 'precision=6' and 'nb_scale=0 (auto)'.

```



Example 685 : image.jpg +rotate 20,1,1,50%,50% +register_nonrigid[0] [1]

2.19.16 *register_rigid*

Arguments:

```

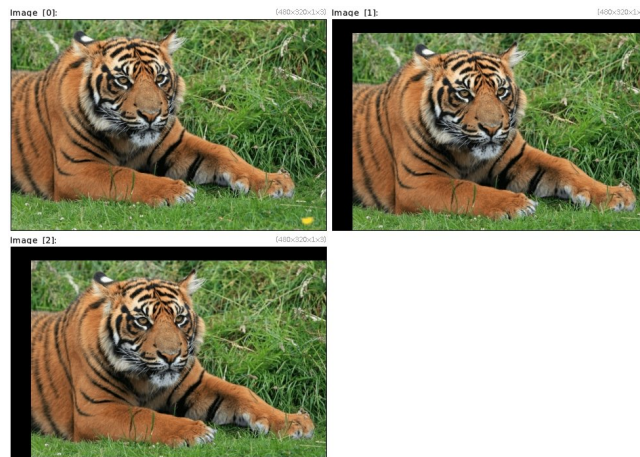
• [destination], _smoothness>=0, _boundary_conditions={ 0=dirichlet | 1=neumann
| 2=periodic | 3=mirror }

```

Register selected source images with specified destination image, using rigid warp (shift).

Default values:

- 'smoothness=1' and 'boundary_conditions=0'.



Example 686 : `image.jpg +shift 30,20 +register_rigid[0] [1]`

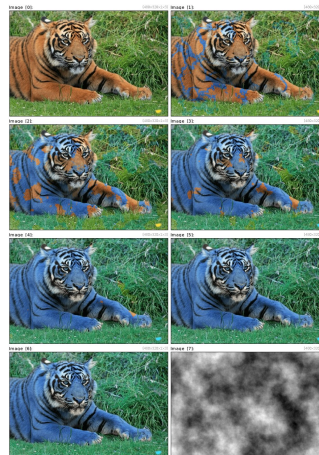
2.19.17 transition**Arguments:**

- `[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 687 : `image.jpg +mirror c 100%,100% plasma[-1] 1,1,6 transition[0,1] [2],5`

2.19.18 *transition3d*

Arguments:

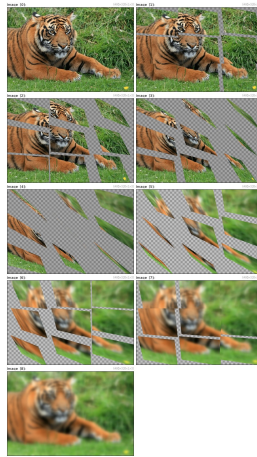
- `_nb_frames>=2, _nb_xtiles>0, _nb_ytiles>0, _axis_x, _axis_y, _axis_z, _is_antialias={0 | 1}`

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 688 : `image.jpg +blur 5 transition3d 9 display_rgba`

2.19.19 *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 Convenience Functions

2.20.1 *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.'`

2.20.2 *arg*

Arguments:

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

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

2.20.3 *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).

2.20.4 *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'`

2.20.5 *base64img*

Arguments:

- `"base64_string"`

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

The argument string must have been generated using command `'img2base64'`.

2.20.6 *base64uchar*

Arguments:

- `"base64_string"`

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

The argument string must have been generated using command `'uchar2base64'`.

2.20.7 *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).

2.20.8 *bin*

Arguments:

- `binary_int1, ...`

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

2.20.9 *bin2dec*

Arguments:

- `binary_int1, ...`

Convert specified binary integers into their decimal representations.

2.20.10 *dec*

Arguments:

- `decimal_int1, ...`

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

2.20.11 *dec2str*

Arguments:

- `decimal_int1, ...`

Convert specifical decimal integers into its string representation.

2.20.12 *dec2bin*

Arguments:

- `decimal_int1, ...`

Convert specified decimal integers into their binary representations.

2.20.13 *dec2hex***Arguments:**

- `decimal_int1, ...`

Convert specified decimal integers into their hexadecimal representations.

2.20.14 *dec2oct***Arguments:**

- `decimal_int1, ...`

Convert specified decimal integers into their octal representations.

2.20.15 *fact***Arguments:**

- `value`

Return the factorial of the specified value.

2.20.16 *fibonacci***Arguments:**

- `N >= 0`

Return the Nth number of the Fibonacci sequence.

2.20.17 *file_mv***Arguments:**

- `filename_src, filename_dest`

Rename or move a file from a location \$1 to another location \$2.

2.20.18 *file_rand*

Return a random filename for storing temporary data.

2.20.19 *file_rm***Arguments:**

- `filename`

Delete a file.

2.20.20 *filename*

Arguments:

- `filename, _number1, _number2, ..., _numberN`

Return a filename numbered with specified indices.

2.20.21 *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 filenames only.

Default value:

- `'mode=5'`.

2.20.22 *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.

2.20.23 *fitscreen*

Arguments:

- `width, height, _depth, _minimal_size[%], _maximal_size[%]`

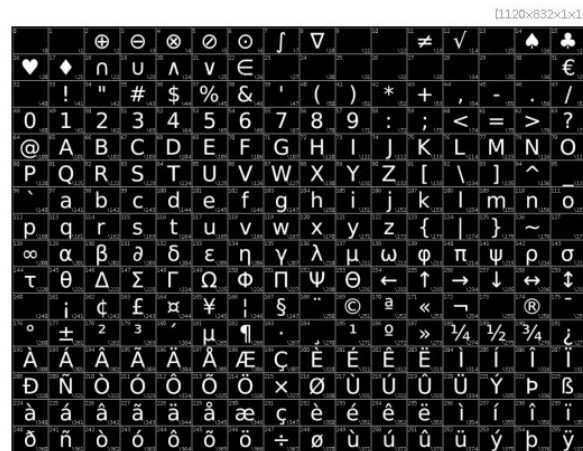
Return the 'ideal' size WxH for a window intended to display an image of specified size on screen.

Default values:

- `'depth=1', 'minimal_size=128' and 'maximal_size=85%'`.

2.20.24 *fontchart*

Insert G'MIC font chart at the end of the image list.



Example 689 : fontchart

2.20.25 *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.

2.20.26 *gcd***Arguments:**

- `a, b`

Return the GCD (greatest common divisor) between a and b.

2.20.27 *hex***Arguments:**

- `hexadecimal_int1, ...`

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

2.20.28 *hex2dec***Arguments:**

- `hexadecimal_int1, ...`

Convert specified hexadecimal integers into their decimal representations.

2.20.29 *hex2img***Arguments:**

- `"hexadecimal_string"`

Insert new image 1xN at the end of the list with values specified by the given hexadecimal-encoded string.

2.20.30 *hex2str*

Arguments:

- `hexadecimal_string`

Convert specified hexadecimal string into a string.

2.20.31 *img2hex*

Return representation of last image as an hexadecimal-encoded string.

Input image must have values that are integers in [0,255].

2.20.32 *img2str*

Return the content of the latest of the selected images as a special G'MIC input string.

2.20.33 *img2text*

Arguments:

- `_line_separator`

Return text contained in a multi-line image.

Default value:

- `'line_separator= '`

2.20.34 *img82hex*

Convert selected 8bits-valued vectors into their hexadecimal representations (ascii-encoded).

2.20.35 *hex2img8*

Convert selected hexadecimal representations (ascii-encoded) into 8bits-valued vectors.

2.20.36 *is_3d*

Return 1 if all of the selected images are 3D objects, 0 otherwise.

2.20.37 *is_ext*

Arguments:

- `filename, _extension`

Return 1 if specified filename has a given extension.

2.20.38 *is_image_arg*

Arguments:

- `string`

Return 1 if specified string looks like '[ind]'.

2.20.39 *is_pattern***Arguments:**

- `string`

Return 1 if specified string looks like a drawing pattern '0x.....'.

2.20.40 *is_percent***Arguments:**

- `string`

Return 1 if specified string ends with a '%', 0 otherwise.

2.20.41 *is_videofilename*

Return 1 if extension of specified filename is typical from video files.

2.20.42 *is_windows*

Return 1 if current computer OS is Windows, 0 otherwise.

2.20.43 *math_lib*

Return string that defines a set of several useful macros for the embedded math evaluator.

2.20.44 *mad*

Return the MAD (Maximum Absolute Deviation) of the last selected image.

The MAD is defined as $MAD = \text{med}_i |x_i - \text{med}_j(x_j)|$

2.20.45 *max_w*

Return the maximal width between selected images.

2.20.46 *max_h*

Return the maximal height between selected images.

2.20.47 *max_d*

Return the maximal depth between selected images.

2.20.48 *max_s*

Return the maximal spectrum between selected images.

2.20.49 *max_wh*

Return the maximal wxh size of selected images.

2.20.50 *max_whd*

Return the maximal wxhd size of selected images.

2.20.51 *max_whds*

Return the maximal wxhxdxs size of selected images.

2.20.52 *med*

Return the median value of the last selected image.

2.20.53 *median_color*

Return the median color value of the last selected image.

2.20.54 *min_w*

Return the minimal width between selected images.

2.20.55 *min_h*

Return the minimal height between selected images.

2.20.56 *min_d*

Return the minimal depth between selected images.

2.20.57 *min_s*

Return the minimal s size of selected images.

2.20.58 *min_wh*

Return the minimal wxh size of selected images.

2.20.59 *min_whd*

Return the minimal wxhxd size of selected images.

2.20.60 *min_whds*

Return the minimal wxhxdxs size of selected images.

2.20.61 *normalize_filename***Arguments:**

- `filename`

Return a "normalized" version of the specified filename, without spaces and capital letters.

2.20.62 *oct***Arguments:**

- `octal_int1, ...`

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

2.20.63 *oct2dec***Arguments:**

- `octal_int1, ...`

Convert specified octal integers into their decimal representations.

2.20.64 *padint***Arguments:**

- `number, _size>0`

Return a integer with 'size' digits (eventually left-padded with '0').

2.20.65 *path_gimp*

Return a path to store GIMP configuration files for one user (whose value is OS-dependent).

2.20.66 *path_tmp*

Return a path to store temporary files (whose value is OS-dependent).

2.20.67 *reset*

Reset global parameters of the interpreter environment.

2.20.68 *RGB*

Return a random int-valued RGB color.

2.20.69 *RGBA*

Return a random int-valued RGBA color.

2.20.70 *std_noise*

Return the estimated noise standard deviation of the last selected image.

2.20.71 *str***Arguments:**

- `string`

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

2.20.72 *str2hex***Arguments:**

- `string`

Convert specified string into a sequence of hexadecimal values.

2.20.73 *strcontains***Arguments:**

- `string1, string2`

Return 1 if the first string contains the second one.

2.20.74 *strlen***Arguments:**

- `string1`

Return the length of specified string argument.

2.20.75 *strreplace***Arguments:**

- `string, search, replace`

Search and replace substrings in an input string.

2.20.76 *strlowercase***Arguments:**

- `string`

Return a lower-case version of the specified string.

2.20.77 *strvar***Arguments:**

- `string`

Return a simplified version of the specified string, that can be used as a variable name.

2.20.78 *strver***Arguments:**

- `_version`

Return the specified version number of the G'MIC interpreter, as a string.

Default value:

- `'version=$_version'`

2.20.79 *tic*

Initialize tic-toc timer.
Use it in conjunction with 'toc'.

2.20.80 *toc*

Display elapsed time of the tic-toc timer since the last call to 'tic'.
This command returns the elapsed time in the status value.
Use it in conjunction with 'tic'.

2.20.81 *to_clutname***Arguments:**

- "string"

Return simplified name that can be used as a CLUT name, from specified input string.

2.20.82 *uchar2base64***Arguments:**

- `_encoding={ 0=base64 | 1=base64url }`

Encode the values of the latest of the selected images as a base64-encoded string.
The string can be decoded using command 'base642uchar'.
Selected images must have values that are integers in [0,255].

Default values:

- `'encoding=0'`.

2.20.83 *img2base64***Arguments:**

- `_encoding={ 0=base64 | 1=base64url }`

Encode selected images as a base64-encoded string.
The images can be then decoded using command 'base642img'.

Default values:

- `'encoding=0'`.

2.20.84 *average_colors*

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

2.20.85 *covariance_colors*

Arguments:

- `_avg_outvarname`

Return the covariance matrix of the vector-valued colors in the latest of the selected images (for arbitrary number of channels).

Parameter '`avg_outvarname`' is used as a variable name that takes the value of the average vector-value.

2.21 Other Interactive Commands

2.21.1 *demo*

Arguments:

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

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

2.21.2 *x_2048*

Launch the 2048 game.

2.21.3 *x_blobs*

Launch the blobs editor.

2.21.4 *x_bouncing*

Launch the bouncing balls demo.

2.21.5 *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`'.

2.21.6 *x_colorize*

Arguments:

- `_is_lineart={ 0 | 1 }, _max_resolution={ 0 | >=128 }, _multichannels_output={ 0 | 1 }, -[palette1], -[palette2], -[grabber1]`

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_linear=1', 'max_resolution=1024' and 'multichannels_output=0'.`

2.21.7 *x_connect4*

Launch the Connect Four game.

2.21.8 *x_fire*

Launch the fire effect demo.

2.21.9 *x_fireworks*

Launch the fireworks demo.

2.21.10 *x_fisheye*

Launch the fish-eye effect demo.

2.21.11 *x_fourier*

Launch the fourier filtering demo.

2.21.12 *x_grab_color*

Arguments:

- `_variable_name`

Open a color grabber widget from the first selected image.

Argument `'variable_name'` specifies the variable that contains the selected color values at any time.

Assigning `'-1'` to it forces the interactive window to close.

Default values:

- `'variable_name=xgc_variable'.`

2.21.13 *x_hanoi*

Launch the Tower of Hanoi game.

2.21.14 *x_histogram*

Launch the histogram demo.

2.21.15 *x_hough*

Launch the hough transform demo.

2.21.16 *x_jawbreaker***Arguments:**

- `0<_width<20,0<_height<20,0<_balls<=8`

Launch the Jawbreaker game.

2.21.17 *x_landscape*

Launch the virtual landscape demo.

2.21.18 *x_life*

Launch the game of life.

2.21.19 *x_light*

Launch the light effect demo.

2.21.20 *x_mandelbrot***Arguments:**

- `_julia={ 0 | 1 },_c0r,_c0i`

Launch Mandelbrot/Julia explorer.

2.21.21 *x_mask_color***Arguments:**

- `_colorspace={ all | rgb | lrgb | ycbr | lab | lch | hsv | hsi | hsl | cmy | cmyk | yiq },_spatial-tolerance>=0,_color-tolerance>=0`

Interactively select a color, and add an alpha channel containing the corresponding color mask.

Argument 'colorspace' refers to the color metric used to compute color similarities, and can be basically one of { `rgb | lrgb | ycbr | lab | lch | hsv | hsi | hsl | cmy | cmyk | yiq` }.

You can also select one particular channel of this colorspace, by setting 'colorspace' as 'colorspace.channel' (e.g. 'hsv_h' for the hue).

Default values:

- `'colorspace=all', 'spatial-tolerance=5' and 'color-tolerance=5'.`

2.21.22 *x_metaballs3d*

Launch the 3D metaballs demo.

2.21.23 *x_minesweeper***Arguments:**

- `8<=_width=<20,8<=_height<=20`

Launch the Minesweeper game.

2.21.24 *x_minimal_path*

Launch the minimal path demo.

2.21.25 *x_pacman*

Launch pacman game.

2.21.26 *x_paint*

Launch the interactive painter.

2.21.27 *x_plasma*

Launch the plasma effect demo.

2.21.28 *x_quantize_rgb***Arguments:**

- `_nbcolors>=2`

Launch the RGB color quantization demo.

2.21.29 *x_reflection3d*

Launch the 3D reflection demo.

2.21.30 *x_rubber3d*

Launch the 3D rubber object demo.

2.21.31 *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'`.

2.21.32 *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'`.

2.21.33 *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=xf-variable`', '`background_curve_R=220`', '`background_curve_G=background_curve_B=background_curve_T`'.

2.21.34 *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 (as R,G,B,[A]) at any time.

Assigning '-1' to it forces the interactive window to close.

Default values:

- '`variable_name=xsp-variable`' and '`number_of_columns=2`'.

2.21.35 *x_shadebobs*

Launch the shade bobs demo.

2.21.36 *x_spline*

Launch spline curve editor.

2.21.37 *x_starfield3d*

Launch the 3D starfield demo.

2.21.38 *x_tetris*

Launch tetris game.

2.21.39 *x_tictactoe*

Launch tic-tac-toe game.

2.21.40 *x_waves*

Launch the image waves demo.

2.21.41 *x_whirl***Arguments:**

- `_opacity>=0`

Launch the fractal whirls demo.

Default values:

- `'opacity=0.2'`.

2.22 Command shortcuts

- `'!=` (+) is equivalent to `'neq'`.
- `'<` (+) is equivalent to `'lt'`.
- `'<<` (+) is equivalent to `'bsl'`.
- `'<=` (+) is equivalent to `'le'`.
- `'>` (+) is equivalent to `'gt'`.
- `'>>` (+) is equivalent to `'bsr'`.
- `'>=` (+) is equivalent to `'ge'`.
- `'*` (+) is equivalent to `'mul'`.
- `'*3d` (+) is equivalent to `'mul3d'`.
- `'+` (+) is equivalent to `'add'`.
- `'+3d` (+) is equivalent to `'add3d'`.
- `'-` (+) is equivalent to `'sub'`.
- `'-3d` (+) is equivalent to `'sub3d'`.
- `'/'` (+) is equivalent to `'div'`.
- `'/3d` (+) is equivalent to `'div3d'`.
- `'=` (+) is equivalent to `'set'`.
- `'=='` (+) is equivalent to `'eq'`.
- `'%` (+) is equivalent to `'mod'`.
- `'&` (+) is equivalent to `'and'`.
- `'^` (+) is equivalent to `'pow'`.
- `'a` (+) is equivalent to `'append'`.
- `'ac` is equivalent to `'apply_channels'`.
- `'ap` is equivalent to `'apply_parallel'`.
- `'apc` is equivalent to `'apply_parallel_channels'`.
- `'apo` is equivalent to `'apply_parallel_overlap'`.
- `'at` is equivalent to `'apply_tiles'`.
- `'b` (+) is equivalent to `'blur'`.
- `'c` (+) is equivalent to `'cut'`.
- `'c3d` is equivalent to `'center3d'`.
- `'col3d` (+) is equivalent to `'color3d'`.
- `'d` (+) is equivalent to `'display'`.
- `'d0` is equivalent to `'display0'`.
- `'d3d` (+) is equivalent to `'display3d'`.

- `'da'` is equivalent to `'display_array'`.
- `'db3d'` (+) is equivalent to `'double3d'`.
- `'dfft'` is equivalent to `'display_fft'`.
- `'dg'` is equivalent to `'display_graph'`.
- `'dh'` is equivalent to `'display_histogram'`.
- `'dp'` is equivalent to `'display_parallel'`.
- `'dp0'` is equivalent to `'display_parallel0'`.
- `'dq'` is equivalent to `'display_quiver'`.
- `'drgba'` is equivalent to `'display_rgba'`.
- `'dt'` is equivalent to `'display_tensors'`.
- `'dw'` is equivalent to `'display_warp'`.
- `'e'` (+) is equivalent to `'echo'`.
- `'endl'` (+) is equivalent to `'endlocal'`.
- `'f'` (+) is equivalent to `'fill'`.
- `'f3d'` (+) is equivalent to `'focale3d'`.
- `'fc'` is equivalent to `'fill_color'`.
- `'fi'` (+) is equivalent to `'endif'`.
- `'frame'` is equivalent to `'frame_xy'`.
- `'g'` (+) is equivalent to `'gradient'`.
- `'h'` is equivalent to `'help'`.
- `'i'` (+) is equivalent to `'input'`.
- `'ig'` is equivalent to `'input_glob'`.
- `'ir'` is equivalent to `'inrange'`.
- `'j'` (+) is equivalent to `'image'`.
- `'j3d'` (+) is equivalent to `'object3d'`.
- `'k'` (+) is equivalent to `'keep'`.
- `'l'` (+) is equivalent to `'local'`.
- `'l3d'` (+) is equivalent to `'light3d'`.
- `'m'` (+) is equivalent to `'command'`.
- `'m*' (+)` is equivalent to `'mmul'`.
- `'m/' (+)` is equivalent to `'mdiv'`.
- `'m3d'` (+) is equivalent to `'mode3d'`.
- `'md3d'` (+) is equivalent to `'moded3d'`.
- `'mv'` (+) is equivalent to `'move'`.
- `'n'` (+) is equivalent to `'normalize'`.
- `'n3d'` is equivalent to `'normalize3d'`.
- `'nm'` (+) is equivalent to `'name'`.
- `'o'` (+) is equivalent to `'output'`.
- `'o3d'` (+) is equivalent to `'opacity3d'`.
- `'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'`.
- `'p3d'` (+) is equivalent to `'primitives3d'`.
- `'q'` (+) is equivalent to `'quit'`.
- `'r'` (+) is equivalent to `'resize'`.
- `'r2dx'` is equivalent to `'resize2dx'`.
- `'r2dy'` is equivalent to `'resize2dy'`.
- `'r3d'` (+) is equivalent to `'rotate3d'`.

- `'r3dx'` is equivalent to `'resize3dx'`.
- `'r3dy'` is equivalent to `'resize3dy'`.
- `'r3dz'` is equivalent to `'resize3dz'`.
- `'rm'` (+) is equivalent to `'remove'`.
- `'rr2d'` is equivalent to `'resize_ratio2d'`.
- `'rv'` (+) is equivalent to `'reverse'`.
- `'rv3d'` (+) is equivalent to `'reverse3d'`.
- `'s'` (+) is equivalent to `'split'`.
- `'s3d'` (+) is equivalent to `'split3d'`.
- `'sh'` (+) is equivalent to `'shared'`.
- `'sl3d'` (+) is equivalent to `'spec13d'`.
- `'sp'` is equivalent to `'sample'`.
- `'ss3d'` (+) is equivalent to `'specs3d'`.
- `'t'` (+) is equivalent to `'text'`.
- `'t3d'` is equivalent to `'texturize3d'`.
- `'to'` is equivalent to `'text_outline'`.
- `'u'` (+) is equivalent to `'status'`.
- `'up'` is equivalent to `'update'`.
- `'v'` (+) is equivalent to `'verbose'`.
- `'w'` (+) is equivalent to `'window'`.
- `'x'` (+) is equivalent to `'exec'`.
- `'y'` (+) is equivalent to `'unroll'`.
- `'z'` (+) is equivalent to `'crop'`.
- `'|'` (+) is equivalent to `'or'`.

2.23 Examples of use

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

- View a list of images:
`gmic file1.bmp file2.jpeg`
- Convert an image file:
`gmic input.bmp output output.jpg`
- Create a volumetric image from a movie sequence:
`gmic input.mpg append z output output.hdr`
- Compute image gradient norm:
`gmic input.bmp gradient_norm`
- Denoise a color image:
`gmic image.jpg denoise 30,10 output denoised.jpg`
- Compose two images using overlay layer blending:
`gmic image1.jpg image2.jpg blend overlay output blended.jpg`

- Evaluate a mathematical expression:
`gmic echo "cos(pi/4)^2+sin(pi/4)^2={cos(pi/4)^2+sin(pi/4)^2}"`
- Plot a 2D function:
`gmic 1000,1,1,2 fill "X=3*(x-500)/500;X^2*sin(3*X^2)+if(c==0,u(0,-1),cos(X*10))" plot`
- Plot a 3D elevated function in random colors:
`gmic 128,128,1,3,"u(0,255)" plasma 10,3 blur 4 sharpen 10000 \
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 mode3d 5 moded3d 5 double3d 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 normalize 0,100 +plasma 0.4 add \
blur 1 elevation3d -0.1 moded3d 4`
- Generate a 3D ring of torii:
`gmic repeat 20 torus3d 15,2 color3d[-1] ""{u(60,255)},{u(60,255)},{u(60,255)}"" \
*3d[-1] 0.5,1 if ""{$>%2}" rotate3d[-1] 0,1,0,90 fi add3d[-1] 70 add3d \
rotate3d 0,0,1,18 done moded3d 3 mode3d 5 double3d 0`
- Create a vase from a 3D isosurface:
`gmic moded3d 4 isosurface3d ""x^2+2*abs(y/2)*sin(2*y)^2+z^2-3',0" sphere3d 1.5 \
sub3d[-1] 0,5 plane3d 15,15 rotate3d[-1] 1,0,0,90 center3d[-1] add3d[-1] 0,3,2 \
color3d[-1] 180,150,255 color3d[-2] 128,255,0 color3d[-3] 255,128,0 add3d`
- Display filtered webcam stream:
`gmic apply_camera \"+mirror x +mirror y add div 4\"`
- Launch a set of **G'MIC** interactive demos:
`gmic demo`

Index of commands

abs, 60
acos, 61
acosh, 62
add, 62
add3d, 290
adjust_colors, 125
alert, 422
and, 63
animate, 415
animate3d, 291
append, 155
append_tiles, 156
apply_camera, 416
apply_camera3d, 291
apply_channels, 125
apply_curve, 98
apply_files, 416
apply_gamma, 98
apply_matrix3d, 291
apply_parallel, 338
apply_parallel_channels, 338
apply_parallel_overlap, 339
apply_scales, 157
apply_tiles, 340
apply_timeout, 340
apply_video, 416
area, 241
area_fg, 241
arg, 423
arg2var, 423
argmax, 64
argmin, 65
array, 349
array3d, 292
array_fade, 349
array_mirror, 350
array_random, 351
arrow, 262
arrow3d, 292
asin, 65
asinh, 66
at_line, 242
at_quadrangle, 242
atan, 66
atan2, 67
atanh, 68
autocrop, 157
autocrop_components, 158
autocrop_coords, 423
autocrop_seq, 159
autoindex, 126
average_colors, 433
average_files, 416
average_video, 417
axes, 263
axes3d, 293

balance_gamma, 99
ball, 264
bandpass, 185
barycenter, 243
base64img, 423
base64uchar, 423
basename, 424
bayer2rgb, 126
bilateral, 185
bin, 424
bin2dec, 424
blend, 406
blend_edges, 409
blend_fade, 410
blend_median, 410
blend_seamless, 411
blur, 186
blur_angular, 187
blur_bloom, 188
blur_linear, 188
blur_radial, 189
blur_selective, 190
blur_x, 190
blur_xy, 191
blur_xyz, 192

- blur_y, 192
- blur_z, 192
- box3d, 294
- boxfilter, 193
- boxfitting, 365
- break, 341
- brushify, 365
- bsl, 68
- bsr, 68
- bump2normal, 194

- camera, 25
- cartoon, 366
- cast, 99
- center3d, 294
- channels, 159
- check, 340
- check3d, 341
- chessboard, 264
- cie1931, 265
- circle, 265
- circle3d, 295
- circles3d, 295
- close_binary, 266
- clut, 26
- cmy2rgb, 127
- cmyk2rgb, 127
- color3d, 296
- color_ellipses, 367
- colorblind, 127
- colorcube3d, 296
- colormap, 128
- columns, 160
- command, 28
- complex2polar, 100
- compose_channels, 128
- compose_freq, 194
- compress_clut, 100
- compress_rle, 100
- cone3d, 297
- continue, 341
- convolve, 195
- convolve_fft, 196
- correlate, 196
- cos, 69
- cosh, 70
- covariance_colors, 434
- cracks, 399
- crop, 160
- cross_correlation, 197

- cubes3d, 297
- cubism, 367
- cumulate, 101
- cup3d, 298
- cursor, 29
- curvature, 198
- cut, 101
- cylinder3d, 298

- dct, 198
- deblur, 199
- deblur_goldmeinel, 199
- deblur_richardsonlucy, 200
- debug, 25
- dec, 424
- dec2bin, 424
- dec2hex, 425
- dec2oct, 425
- dec2str, 424
- decompress_clut, 102
- decompress_clut_rbf, 102
- decompress_rle, 103
- deconvolve_fft, 201
- deform, 388
- deinterlace, 201
- delaunay, 243
- delaunay3d, 299
- demo, 434
- denoise, 202
- denoise_haar, 202
- denoise_patchpca, 203
- deriche, 203
- detect_skin, 244
- diagonal, 161
- diffusionensors, 207
- dijkstra, 286
- dilate, 204
- dilate_circ, 205
- dilate_oct, 206
- dilate_threshold, 206
- direction2rgb, 129
- discard, 103
- displacement, 244
- display, 29
- display0, 29
- display3d, 30
- display_array, 30
- display_fft, 30
- display_graph, 30
- display_histogram, 31

- display_parallel, 33
- display_parallel0, 33
- display_parametric, 32
- display_polar, 33
- display_quiver, 34
- display_rgba, 35
- display_tensors, 35
- display_warp, 36
- distance, 245
- distribution3d, 299
- ditheredbw, 129
- div, 70
- div3d, 300
- div_complex, 71
- divergence, 206
- do, 342
- document_gmic, 37
- dog, 207
- done, 342
- double3d, 300
- draw_whirl, 368
- drawing, 369
- drop_shadow, 369

- echo, 37
- echo_file, 37
- echo_stdout, 38
- edges, 208
- eigen, 287
- eigen2tensor, 104
- elevate, 162
- elevation3d, 301
- elif, 342
- ellipse, 266
- ellipsionism, 370
- else, 342
- empty3d, 302
- endian, 104
- endif, 342
- endlocal, 342
- eq, 71
- equalize, 104
- equiangular2nadirzenith, 389
- erode, 209
- erode_circ, 209
- erode_oct, 210
- erode_threshold, 210
- error, 342
- euclidean2polar, 388
- eval, 343

- exec, 343
- exp, 72
- expand_x, 162
- expand_xy, 163
- expand_xyz, 163
- expand_y, 164
- expand_z, 164
- extract_region, 165
- extrude3d, 303

- fact, 425
- fade_diamond, 411
- fade_files, 417
- fade_linear, 412
- fade_radial, 412
- fade_video, 417
- fade_x, 413
- fade_y, 413
- fade_z, 414
- fft, 211
- fftpolar, 246
- fibonacci, 425
- file_mv, 425
- file_rand, 425
- file_rm, 425
- filename, 426
- files, 426
- files2video, 418
- fill, 105
- fill_color, 129
- fire_edges, 370
- fish-eye, 389
- fitratio_wh, 426
- fitscreen, 426
- flood, 267
- flower, 390
- focale3d, 303
- fontchart, 426
- for, 343
- fps, 427
- fractalize, 371
- frame_blur, 351
- frame_cube, 352
- frame_fuzzy, 352
- frame_painting, 353
- frame_pattern, 354
- frame_round, 354
- frame_seamless, 355
- frame_x, 355
- frame_xy, 356

- frame_xyz, 357
- frame_y, 357
- function1d, 38
- gaussian, 268
- gaussians3d, 304
- gcd, 427
- ge, 73
- glow, 372
- gmic3d, 304
- gradient, 211
- gradient2rgb, 130
- gradient_norm, 212
- gradient_orientation, 213
- graph, 268
- grid, 269
- gt, 74
- guided, 213
- gyroid3d, 305
- haar, 214
- halftone, 372
- hardsketchbw, 373
- hcy2rgb :, 130
- hearts, 373
- heat_flow, 214
- help, 25
- hessian, 215
- hex, 427
- hex2dec, 427
- hex2img, 427
- hex2img8, 428
- hex2str, 428
- histogram, 247
- histogram3d, 305
- histogram_cumul, 248
- histogram_nd, 247
- histogram_pointwise, 249
- hough, 249
- houghsketchbw, 374
- hsi2rgb, 130
- hsi82rgb, 131
- hsl2rgb, 131
- hsl82rgb, 131
- hsv2rgb, 131
- hsv82rgb, 131
- idct, 215
- iee, 215
- if, 344
- ifft, 216
- ifftpolar, 249
- ihaar, 216
- ilaplacian, 216
- image, 270
- image6cube3d, 306
- imageblocks3d, 306
- imagecube3d, 307
- imagegrid, 358
- imagegrid_hexagonal, 359
- imagegrid_triangular, 359
- imageplane3d, 307
- imagepyramid3d, 308
- imagerubik3d, 308
- imagesphere3d, 309
- img2ascii, 357
- img2base64, 433
- img2hex, 428
- img2str, 428
- img2text, 428
- img82hex, 428
- index, 106
- inn, 217
- inpaint, 217
- inpaint_diffusion, 218
- inpaint_flow, 219
- inpaint_holes, 219
- inpaint_matchpatch, 220
- inpaint_morpho, 220
- input, 38
- input_cube, 40
- input_glob, 40
- input_gpl, 40
- inrange, 107
- int2rgb, 131
- invert, 287
- is_3d, 428
- is_ext, 428
- is_image_arg, 428
- is_pattern, 429
- is_percent, 429
- is_videofilename, 429
- is_windows, 429
- isoline3d, 309
- isophotes, 249
- isosurface3d, 310
- kaleidoscope, 390
- keep, 55
- kuwahara, 221

- lab2lch, 131
- lab2rgb, 131
- lab2srgb, 132
- lab2xyz, 133
- lab82rgb, 133
- lab82srgb, 132
- label, 250
- label3d, 311
- label_fg, 251
- label_points3d, 311
- laplacian, 222
- lathe3d, 312
- lch2lab, 133
- lch2rgb, 134
- lch82rgb, 134
- le, 75
- lic, 222
- light3d, 313
- light_patch, 400
- light_relief, 375
- lightrays, 375
- line, 270
- line3d, 313
- linearize_tiles, 360
- linethick, 271
- linify, 376
- lissajous3d, 314
- local, 344
- log, 77
- log10, 78
- log2, 79
- lt, 76
- luminance, 134

- mad, 429
- mandelbrot, 272
- map, 108
- map_clut, 109
- map_sphere, 391
- map_sprites, 360
- map_tones, 222
- map_tones_fast, 223
- marble, 272
- matchpatch, 254
- math_lib, 429
- max, 79
- max_d, 429
- max_h, 429
- max_patch, 251
- max_s, 429
- max_w, 429
- max_wh, 429
- max_whd, 429
- max_whds, 430
- maze, 273
- maze_mask, 273
- mdiv, 80
- meancurvature_flow, 224
- med, 430
- median, 224
- median_color, 430
- median_files, 418
- median_video, 418
- min, 80
- min_d, 430
- min_h, 430
- min_patch, 252
- min_s, 430
- min_w, 430
- min_wh, 430
- min_whd, 430
- min_whds, 430
- minimal_path, 252
- mirror, 166
- mix_channels, 109
- mix_rgb, 134
- mmul, 82
- mod, 81
- mode3d, 314
- moded3d, 315
- montage, 165
- morph, 419
- morph_files, 419
- morph_video, 419
- mosaic, 376
- move, 55
- mse, 253
- mul, 83
- mul3d, 315
- mul_channels, 84
- mul_complex, 85
- mutex, 345

- nadirzenith2equirectangular, 391
- name, 56
- negate, 110
- neq, 85
- nlmeans, 225
- nlmeans_core, 225
- noarg, 345

- noise, 110
- noise_hurl, 400
- noise_perlin, 111
- noise_poissondisk, 112
- norm, 113
- normalize, 113
- normalize3d, 316
- normalize_filename, 430
- normalize_local, 225
- normalize_sum, 114
- normalized_cross_correlation, 226
- normp, 112
- not, 114

- object3d, 274
- oct, 430
- oct2dec, 431
- old_photo, 377
- oneminus, 115
- onfail, 345
- opacity3d, 316
- or, 86
- orientation, 115
- otsu, 116
- output, 41
- output_cube, 41
- output_ggr, 41
- outputn, 41
- outputp, 41
- outputw, 42
- outputx, 42

- pack, 361
- pack_sprites, 275
- padint, 431
- parallel, 346
- parametric3d, 317
- pass, 42
- patches, 253
- path_gimp, 431
- path_tmp, 431
- pca_patch3d, 318
- pde_flow, 227
- pencilbw, 377
- periodize_poisson, 228
- permute, 167
- peronamalik_flow, 226
- phase_correlation, 227
- piechart, 275
- pixelize, 401

- pixelsort, 378
- plane3d, 318
- plasma, 276
- plot, 43
- plot2value, 255
- point, 277
- point3d, 319
- pointcloud, 255
- pointcloud3d, 319
- polar2complex, 116
- polar2euclidean, 391
- polaroid, 379
- polka_dots, 277
- polygon, 278
- polygonize, 379
- pose3d, 320
- poster_edges, 380
- poster_hope, 380
- pow, 87
- primitives3d, 320
- print, 43
- progress, 346
- projections3d, 321
- pseudogray, 135
- psnr, 256
- puzzle, 361
- pyramid3d, 321

- quadrangle3d, 322
- quadratize_tiles, 362
- quantize, 116
- quantize_area, 117
- quit, 347
- quiver, 279

- rainbow_lut, 43
- raindrops, 392
- rand, 118
- random3d, 322
- rectangle, 280
- red_eye, 228
- register_nonrigid, 420
- register_rigid, 420
- remove, 57
- remove_duplicates, 58
- remove_empty, 58
- remove_hotpixels, 229
- remove_opacity, 148
- remove_pixels, 229
- repeat, 347

- replace, 118
- replace_color, 136
- replace_inf, 119
- replace_nan, 119
- replace_seq, 120
- replace_str, 120
- reset, 431
- resize, 167
- resize2dx, 170
- resize2dy, 171
- resize3dx, 172
- resize3dy, 172
- resize3dz, 172
- resize_mn, 168
- resize_pow2, 169
- resize_ratio2d, 170
- retinex, 136
- return, 347
- reverse, 58
- reverse3d, 323
- RGB, 431
- rgb2bayer, 136
- rgb2cmy, 137
- rgb2cmyk, 137
- rgb2hcy, 138
- rgb2hsi, 139
- rgb2hsi8, 139
- rgb2hsl, 139
- rgb2hsl8, 140
- rgb2hsv, 141
- rgb2hsv8, 141
- rgb2int, 144
- rgb2lab, 142
- rgb2lab8, 142
- rgb2lch, 142
- rgb2lch8, 143
- rgb2luv, 144
- rgb2srgb, 144
- rgb2xyz, 144
- rgb2xyz8, 145
- rgb2ycbcr, 146
- rgb2yiq, 146
- rgb2yiq8, 146
- rgb2yuv, 147
- rgb2yuv8, 147
- RGBA, 431
- ripple, 393
- rodilius, 381
- rol, 88
- rolling_guidance, 230
- ror, 88
- rorschach, 280
- rotate, 173
- rotate3d, 323
- rotate_tileable, 173
- rotate_tiles, 363
- rotation3d, 323
- rotoidoscope, 393
- round, 121
- roundify, 122
- rows, 174
- rprogress, 348
- run, 348
- sample, 51
- scale2x, 174
- scale3x, 175
- scale_dcci2x, 175
- scanlines, 401
- screen, 43
- seamcarve, 176
- segment_watershed, 257
- select, 44
- select_color, 148
- sepia, 148
- serialize, 44
- set, 122
- set_vector_covariance, 123
- shade_stripes, 402
- shadow_patch, 402
- shape2bump, 257
- shape_circle, 45
- shape_cupid, 45
- shape_diamond, 46
- shape_fern, 46
- shape_gear, 47
- shape_heart, 47
- shape_polygon, 48
- shape_snowflake, 49
- shape_star, 49
- shared, 50
- sharpen, 230
- shift, 176
- shift_tiles, 363
- shrink_x, 177
- shrink_xy, 177
- shrink_xyz, 178
- shrink_y, 178
- shrink_z, 179
- sierpinski, 281

- sierpinski3d, 324
- sign, 89
- sin, 90
- sinc, 91
- sinh, 91
- size3d, 324
- skeleton, 258
- skeleton3d, 324
- sketchbw, 383
- skip, 348
- slic, 258
- slices, 179
- smooth, 231
- snapshot3d, 325
- solarize, 149
- solidify, 235
- solve, 288
- solve_poisson, 233
- sort, 179
- sort_list, 59
- sort_str, 60
- spec13d, 326
- specs3d, 327
- sphere3d, 327
- spherical3d, 328
- spherize, 394
- spiralbw, 282
- spline, 282
- spline3d, 329
- split, 179
- split3d, 329
- split_colors, 149
- split_details, 233
- split_freq, 232
- split_opacity, 150
- split_tiles, 182
- sponge, 384
- spread, 403
- sprite3d, 330
- sprites3d, 330
- sqr, 92
- sqrt, 93
- srnd, 51
- srgb2lab, 150
- srgb2lab8, 151
- srgb2rgb, 151
- ssd_patch, 259
- stained_glass, 382
- star3d, 331
- stars, 382
- status, 348
- std_noise, 431
- stencil, 384
- stencilbw, 385
- str, 431
- str2hex, 431
- strcontains, 432
- streamline3d, 331
- string, 51
- stripes_y, 403
- strlen, 432
- strlowercase, 432
- strreplace, 432
- structuretensors, 234
- strvar, 432
- strver, 432
- sub, 94
- sub3d, 332
- sub_alpha, 414
- superformula3d, 333
- svd, 288
- symmetrize, 394
- syntexturize, 235
- syntexturize_matchpatch, 236
- tan, 95
- tanh, 96
- taquin, 364
- tensors3d, 333
- testimage2d, 52
- tetraedron_shade, 283
- tetris, 385
- text, 283
- text3d, 335
- text_outline, 284
- text_pointcloud3d, 334
- texturize3d, 335
- texturize_canvas, 404
- texturize_paper, 404
- thinning, 259
- threshold, 123
- tic, 433
- to_a, 151
- to_clutname, 433
- to_color, 151
- to_colormode, 151
- to_gray, 152
- to_graya, 152
- to_pseudogray, 152
- to_rgb, 152

to_rgba, 152
toc, 433
tones, 260
topographic_map, 261
torus3d, 336
transfer_histogram, 152
transfer_rgb, 153
transform_polar, 395
transition, 421
transition3d, 422
transpose, 289
triangle3d, 336
triangle_shade, 284
trisolve, 289
truchet, 285
tsp, 261
tunnel, 364
turbulence, 285
tv_flow, 236
twirl, 396

uchar2base64, 433
uncommand, 52
undistort, 182
uniform_distribution, 53
unrepeat, 124
unroll, 182
unserialize, 53
unsharp, 237
unsharp_octave, 238
update, 53
upscale_smart, 183

vanvliet, 238
variance_patch, 262
vector2tensor, 125
verbose, 53
version, 25
video2files, 422
vignette, 404
volume3d, 337
voronoi, 239

wait, 54
warhol, 386
warn, 54
warp, 184
warp_patch, 184
warp_perspective, 396
water, 397

watermark_fourier, 239
watermark_visible, 405
watershed, 240
wave, 397
weave, 387
weird3d, 337
while, 349
whirls, 387
wind, 398
window, 54

x_2048, 434
x_blobs, 434
x_bouncing, 434
x_color_curves, 434
x_colorize, 434
x_connect4, 435
x_fire, 435
x_fireworks, 435
x_fisheye, 435
x_fourier, 435
x_grab_color, 435
x_hanoi, 435
x_histogram, 435
x_hough, 435
x_jawbreaker, 436
x_landscape, 436
x_life, 436
x_light, 436
x_mandelbrot, 436
x_mask_color, 436
x_metaballs3d, 436
x_minesweeper, 436
x_minimal_path, 437
x_pacman, 437
x_paint, 437
x_plasma, 437
x_quantize_rgb, 437
x_reflection3d, 437
x_rubber3d, 437
x_segment, 437
x_select_color, 437
x_select_function1d, 438
x_select_palette, 438
x_shadebobs, 438
x_spline, 438
x_starfield3d, 438
x_tetris, 438
x_tictactoe, 438
x_waves, 439

x_whirl, 439
xor, 97
xyz2lab, 154
xyz2rgb, 154
xyz82rgb, 154

ycbcr2rgb, 155
yinyang, 286
yiq2rgb, 155
yiq82rgb, 155
yuv2rgb, 155
yuv82rgb, 155

zoom, 398

☐ End of document.