Usage

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

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

This reference documentation describes all the technical aspects of the G'MIC framework, in its current version 3.4.0.

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

Overall Context

- At any time, G'MIC manages one list of numbered (and optionally named) pixel-based images, entirely stored in computer memory (uncompressed).
- The first image of the list has index 0 and is denoted by [0]. The second image of the list is denoted by [1], the third by [2] and so on.
- Negative indices are treated in a periodic way: [-1] refers to the last image of the list, [-2] to the penultimate one, etc. Thus, if the list has 4 images, [1] and [-3] both designate the second image.
of the list.

- A named image may be also indicated by [name], if name uses the character set [a-zA-Z0-9_] and does not start with a number. Image names can be set or reassigned at any moment during the processing pipeline (see command name for this purpose).

- **G'MIC** defines a set of various commands and substitution mechanisms to allow the design of complex pipelines and operators managing this list of images, in a very flexible way: You can insert or remove images in the list, rearrange image order, process images (individually or grouped), merge image data together, display and output image files, etc.

- Such a pipeline can define a new custom **G'MIC** command (stored in a user command file), and re-used afterwards as a regular command, in a larger pipeline if necessary.

### Image Definition and Terminology

- In **G'MIC**, each image is modeled as a 1D, 2D, 3D or 4D array of scalar values, uniformly discretized on a rectangular/parallelepipedic domain.

- The four dimensions of this array are respectively denoted by:
  - **width**, the number of image columns (size along the \( x \)-axis).
  - **height**, the number of image rows (size along the \( y \)-axis).
  - **depth**, the number of image slices (size along the \( z \)-axis). The depth is equal to 1 for usual color or grayscale 2D images.
  - **spectrum**, the number of image channels (size along the \( c \)-axis). The spectrum is respectively equal to 3 and 4 for usual **RGB** and **RGBA** color images.

- There are no hard limitations on the size of the image along each dimension. For instance, the number of image slices or channels can be of arbitrary size within the limits of the available memory.

- The width, height, and depth of an image are considered as spatial dimensions, while the spectrum has a multi-spectral meaning. Thus, a 4D image in **G'MIC** should be most often regarded as a 3D dataset of multi-spectral voxels. Most of the **G'MIC** commands will stick with this idea (e.g. command blur blurs images only along the spatial \( xyz \)-axes).

- **G'MIC** stores all the image data as buffers of float values (32 bits, value range \([-3.4\text{E38},+3.4\text{E38}]\]). 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 numerical precision when executing image processing pipelines. For image input/output operations, you may want to prescribe the image datatype to be different than float (like bool, char, int, etc.). This is possible by specifying it as a file option when using I/O commands (see section [Output Properties](#output-properties) to learn more about file options).

### Items of a Processing Pipeline

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

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

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

Input Data

- If a specified G'MIC item appears to be an existing filename, the corresponding image data are loaded and inserted at the end of the image list (which is equivalent to the use of input filename).

- Special filenames - and -.ext stand for the standard input/output streams, optionally forced to be in a specific ext file format (e.g. -.jpg or -.png).

- The following special input strings may be used as G'MIC items to create and insert new images with prescribed values, at the end of the image list:
  - [selection] or [selection]xN: Insert 1 or N copies of already existing images. selection may represent one or several images (see section Command Items and Selections to learn more about selections).
  - width[%], height[%], depth[%], spectrum[%], values[xN]: Insert one or N images with specified size and values (adding % to a dimension means "percentage of the size along the same axis", taken from the last image [-1]). Any specified dimension can be also written as [image], and is then set to the size (along the same axis) of the existing specified image [image]. values can be either a sequence of numbers separated by commas ,, or a mathematical expression, as e.g. in input item 256,256,1,3,[x,y,128] which creates a 256x256 RGB color image with a spatial shading on the red and green channels. (see section Mathematical Expressions to learn more about mathematical expressions).
  - (v1,v2,...[delimiter | axis_order])xN: Insert one or N new images from specified values. Value separator inside parentheses can be , (column separator), ; (row separator), / (slice separator) or ^ (channel separator). For instance, expression (1,2,3;4,5,6;7,8,9) creates a 3x3 matrix (scalar image), with values running from 1 to 9.
  - ('string'[delimiter])xN: Insert one or N new images from specified string, by filling the images with the character codes composing the string. When specified, delimiter tells about the main orientation of the image. Delimiter can be x (eq. to , which is the default), y (eq. to .), z (eq. to /) or c (eq. to ^). When specified delimiter is ,, or / or ^, the expression is actually equivalent to ({'string'[delimiter]})xN (see section Substitution Rules for more information on the syntax).
  - 0[xN]: Insert one or N new empty images, containing no pixel data. Empty images are used only in rare occasions.

- Input item name=value declares a new variable name, or assign a new string value to an existing variable. Variable names must use the character set [a-zA-Z0-9_] and cannot start with a number.

- A variable definition is always local to the current command except:
  - When it starts by the underscore character _. In that case, it becomes also accessible by any command invoked outside the current command scope (global variable).
- When defined in a subcommand of the current command, a variable becomes also accessible in the parent command. A subcommand of a command $foo$ is a command whose name starts with _foo (e.g. _foo_sub) and that is called from $foo$.

- If a variable name starts with two underscores __, the global variable is also shared among different threads and can be read/set by commands running in parallel (see command parallel for this purpose). Otherwise, it remains local to the thread that defined it.

- Numerical variables can be updated with the use of these special operators: += (addition), -= (subtraction), *= (multiplication), /= (division), %= (modulo), &= (bitwise and), |= (bitwise or), ^= (power), <<= and >>= (bitwise left and right shifts). For instance, foo=1 foo+=3.

- Input item name.=string appends specified string at the end of variable name.

- Input item name..=string prepends specified string at the beginning of variable name.

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

- Variables usually store numbers or strings. Use command store to assign variables from image data (and syntax input $variable$ to bring them back on the image list afterwards).

### Command Items and Selections

- A G'MIC item that is not a filename nor a special input string designates a command most of the time. Generally, commands perform image processing operations on one or several available images of the list.

- Recurrent commands have two equivalent names (regular and short). For instance, command names resize and r refer to the same image resizing action.

- A G'MIC command may have mandatory or optional arguments. Command arguments must be specified in the next item on the command line. Commas , are used to separate multiple arguments of a single command, when required.

- The execution of a G'MIC command may be restricted only to a subset of the image list, by appending [selection] to the command name. Examples of valid syntaxes for selection are:

  - command[-2]: Apply command only on the penultimate image [-2] of the list.
  - command[0,1,3]: Apply command only on images [0], [1] and [3].
  - command[3-6]: Apply command only on images [3] to [6] (i.e. [3], [4], [5] and [6]).
  - command[50%-100%]: Apply command only on the second half of the image list.
  - command[0,4-1]: Apply command only on the first image and the last four images.
  - command[0-9:3]: Apply command only on images [0] to [9], with a step of 3 (i.e. on images [0], [3], [6] and [9]).
  - command[0--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 [1,1,3,2] are both equivalent to [1-3]. If you want to repeat a single command multiple times on an image, use a repeat..done loop instead. Inverting the
order of images for a command is achieved by explicitly inverting the order of the images in the list, with command reverse[selection].

- Command selections [-1], [-2] and [-3] are so often used they have their own shortcuts, respectively ,, .. and .... For instance, command blur.. is equivalent to blur[-2]. These shortcuts work also when specifying command arguments.

- **G'MIC** commands invoked without [selection] are applied on all images of the list, i.e. the default selection is [0--1] (except for command input whose default selection is [-1]).

- Prepending a single hyphen - to a **G'MIC** command is allowed. This may be useful to recognize command items more easily in a one-liner pipeline (typically invoked from a shell).

- A **G'MIC** command prepended with a plus sign + does not act in-place but inserts its result as one or several new images at the end of the image list.

- There are two different types of commands that can be run by the **G'MIC** interpreter:
  - **Built-in commands** are the hard-coded functionalities in the interpreter core. They are thus compiled as binary code and run fast, most of the time. Omitting an argument when invoking a built-in command is not permitted, except if all following arguments are also omitted. For instance, invoking blur 1,,1 is invalid but blur 1 is correct.
  - **Custom commands**, are defined as **G'MIC** pipelines of built-in or other custom commands. They are parsed by the **G'MIC** interpreter, and thus run a bit slower than built-in commands. Omitting arguments when invoking a custom command is permitted. For instance, expressions flower,,,100,,2 or flower,, are correct.

- Most of the existing commands in **G'MIC** are actually defined as custom commands.
- A user can easily add its own custom commands to the **G'MIC** interpreter (see section Adding Custom Commands for more details). New built-in commands cannot be added (unless you modify the **G'MIC** interpreter source code and recompile it).

---

**Input/Output Properties**

- **G'MIC** is able to read/write most of the classical image file formats, including:
  - 2D grayscale/color files: .png, .jpeg, .gif, .pm, .tif, .bmp, ...
  - 3D volumetric files: .dcm, .hdr, .ni, .cube, .pan, .in, .pnk, ...
  - Video files: .mpeg, .avi, .mp4, .mov, .ogg, .flv, ...
  - Generic text or binary data files: .gmz, .cimg, .cimgz, .flo, .ggr, .gpl, .dlm, .asc, .p, .raw, .txt, .h.
  - 3D mesh files: .off, .obj.

- When dealing with color images, **G'MIC** generally reads, writes and displays data using the usual sRGB color space.
- When loading a .png and .tiff file, the bit-depth of the input image(s) is returned to the status.

- **G'MIC** is able to manage 3D mesh objects that may be read from files or generated by **G'MIC** commands. A 3D object is stored as a one-column scalar image containing the object data, in the following order: { magic_number; sizes; vertices; primitives; colors; opacities }. These 3D representations can be then processed as regular images (see command split3d for accessing each of these 3D object data separately).

- Be aware that usual file formats may be sometimes not adapted to store all the available image
data, since G'MIC uses float-valued image buffers. For instance, saving an image that was initially loaded as a 16bits/channel image, as a .jpg file will result in a loss of information. Use the G'MIC-specific file extension .gmz to ensure that all data precision is preserved when saving images.

- Sometimes, file options may/must be set for file formats:
  - **Video files:** Only sub-frames of an image sequence may be loaded, using the input expression filename.ext,[first_frame,...,last_frame,step]]. Set last_frame=-1 to tell it must be the last frame of the video. Set step to 0 to force an opened video file to be opened/closed. Output framerate and codec can be also set by using the output expression filename.avi,_fps,_codec,_keep_open where keep_open can be { 0: no | 1: yes (default) }.
  - **.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,y0,0,x1,y1,z1] or filename.cimg[N0,N1,x0,y0,z0,0,x1,y1,z1,c1]. Specifying -1 for one coordinates stands for the maximum possible value. Output expression filename.cimg[z][,datatype] can be used to force the output pixel type. datatype can be { auto | bool | uint8 | int8 | uint16 | int16 | uint32 | int32 | uint64 | int64 | float32 | float64 }.
  - **.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.tif[,datatype,,compression,...,use_bigtiff] can be used to specify the output pixel type, as well as the compression method. datatype can be the same as for .cimg[z] files. compression can be { none (default) | lzw | jpeg }. force_multipage can be { 0:no (default) | 1:yes }. use_bigtiff can be { 0:no | 1:yes (default) }.
  - **.pdf files:** When loading a file, the rendering resolution can be specified using the input expression filename.pdf,resolution, where resolution is an unsigned integer value.
  - **.gif files:** Animated gif files can be saved, using the input expression filename.gif,fps>0,nb_loops. Specify nb_loops=0 to get an infinite number of animation loops (this is the default behavior).
  - **.jpeg files:** The output quality may be specified (in %), using the output expression filename.jpeg,quality. Set quality=30 for a 30% quality output. 100 is the default.
  - **.mnc files:** The output header can set from another file, using the output expression filename.mnc,header_template.mnc.
  - **.pan,.cpp,.hpp,.c and .h files:** The output datatype can be selected with output expression filename[datatype]. datatype can be the same as for .cimg[z] files.
  - **.gmic files:** These filenames are assumed to be G'MIC custom commands files. Loading such a file will add the commands it defines to the interpreter. Debug information can be enabled/disabled by the input expression filename.gmic[add_debug_info] where debug_info can be { 0:false | 1:true }.
○ Inserting ext: on the beginning of a filename (e.g. jpg:filename) forces G'MIC to read/write the file as it would have been done if it had the specified extension .ext.

○ Some input/output formats and options may not be supported, depending on the configuration flags that have been set during the build of the G'MIC software.

Substitution Rules

• G'MIC items containing $ or {} are substituted before being interpreted. Use these substituting expressions to access various data from the interpreter environment.

• ${name} and $(name) are both substituted by the value of the specified named variable (set previously by the item name=value). If this variable has not been already set, the expression is substituted by the highest positive index of the named image [name]. If no image has this name, the expression is substituted by the value of the OS environment variable with same name (it may be thus an empty string if it is not defined).

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

  ○ $!: The current number of images in the list.
  ○ $> and $<: The increasing/decreasing index of the latest (currently running) repeat...done loop. $> goes from 0 (first loop iteration) to nb_iterations - 1 (last iteration). $< does the opposite.
  ○ $/: The current call stack. Stack items are separated by slashes /.
  ○ $|: The current value (expressed in seconds) of a millisecond precision timer.
  ○ $^: The current verbosity level.
  ○ $_cpus: The number of computation cores available on your machine.
  ○ $_flags: The list of enabled flags when G'MIC interpreter has been compiled.
  ○ $_host: A string telling about the host running the G'MIC interpreter (e.g. cli or gimp).
  ○ $_os: A string describing the running operating system.
  ○ $_path_rc: The path to the G'MIC folder used to store configuration files (its value is OS-dependent).
  ○ $_path_user: The path to the G'MIC user file .gmic or user.gmic (its value is OS-dependent).
  ○ $_path_commands: A list of all imported command files (stored as an image list).
  ○ $_pid: The current process identifier, as an integer.
  ○ $_pixeltype: The type of image pixels (default: float32).
  ○ $_prerelease: For pre-releases, the date of the pre-release as yymmdd. For stable releases, this variable is set to 0.
  ○ $_version: A 3-digits number telling about the current version of the G'MIC interpreter (e.g. 340).
  ○ $_vt100: Set to 1 if colored text output is allowed on the console. Otherwise, set to 0.

• $$name and $$${name} are both substituted by the G'MIC script code of the specified named custom command, or by an empty string if no custom command with specified name exists.

• $("-pipeline") is substituted by the status value after the execution of the specified G'MIC pipeline (see command status ). Expression $() thus stands for the current status value.

• {``string} (starting with two backquotes) is substituted by a double-quoted version of the
specified string.

• `{/string}` is substituted by the escaped version of the specified string.

• `{string[delimiter]}` (between single quotes) is substituted by the sequence of character codes that composes the specified string, separated by specified delimiter. Possible delimiters are , (default), ;, /, ^ or . For instance, item `{foo}` is substituted by `102,111,111` and `{foo;;}` by `102;111;111`.

• `{image,feature[delimiter]}` is substituted by a specific feature of the image `[image]`. `image` can be either an image number or an image name. It can be also eluded, in which case, the last image [-1] of the list is considered for the requested feature. Specified feature can be one of:
  ◦ `b`: The image basename (i.e. filename without the folder path nor extension).
  ◦ `f`: The image folder name.
  ◦ `n`: The image name or filename (if the image has been read from a file).
  ◦ `t`: The text string from the image values regarded as character codes.
  ◦ `x`: The image extension (i.e the characters after the last `. in the image name).
  ◦ `^`: The sequence of all image values, separated by commas ,.
  ◦ `@subset`: The sequence of image values corresponding to the specified subset, and separated by commas ,.
  ◦ Any other feature is considered as a mathematical expression associated to the image [image] and is substituted by the result of its evaluation (float value). For instance, expression `{0,w+h}` is substituted by the sum of the width and height of the first image (see section Mathematical Expressions for more details). If a mathematical expression starts with an underscore _, the resulting value is truncated to a readable format. For instance, item `_pi` is substituted by 3.14159 (while `{pi}` is substituted by 3.141592653589793).
  ◦ A feature delimited by backquotes is replaced by a string whose character codes correspond to the list of values resulting from the evaluation of the specified mathematical expression. For instance, item `{`[102,111,111]`}` is substituted by foo and item `{vector8(65)}` by "AAAAAAAA".

• `{*}` is substituted by the visibility state of the instant display window #0 (can be `{0:closed | 1:visible}`).

• `{*[index],feature1,...,featureN[delimiter]}` is substituted by a specific set of features of the instant display window #0 (or #index, if specified). Requested features can be:
  ◦ `u`: screen width (actually independent on the window size).
  ◦ `v`: screen height (actually independent on the window size).
  ◦ `uv`: screen width*screen height.
  ◦ `d`: window width (i.e. width of the window widget).
  ◦ `e`: window height (i.e. height of the window widget).
  ◦ `de`: window width*window height.
  ◦ `w`: display width (i.e. width of the display area managed by the window).
  ◦ `h`: display height (i.e. height of the display area managed by the window).
  ◦ `wh`: display width*display height.
  ◦ `i`: X-coordinate of the display window.
  ◦ `j`: Y-coordinate of the display window.
  ◦ `f`: current fullscreen state of the instant display.
  ◦ `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.
- \text{Any other feature stands for a keycode name (in capital letters), and is substituted by a boolean describing the current key state \{ 0:pressed | 1:released \}.}
- You can also prepend a hyphen - to a feature (that supports it) to flush the corresponding event immediately after reading its state (works for keys, mouse and window events).

- Item substitution is \textbf{never} performed in items between double quotes. One must break the quotes to enable substitution if needed, as in "3+8 kg = "(3+8)" kg". Using double quotes is then a convenient way to disable the substitutions mechanism in items, when necessary.

- One can also disable the substitution mechanism on items outside double quotes, by escaping the \{, \} or $ characters, as in \{3+4\}$ doesn’t evaluate.

### Mathematical Expressions

- \textbf{G’MIC} has an embedded \textit{mathematical parser}, used to evaluate (possibly complex) math expressions specified inside braces {}, or formulas in commands that may take one as an argument (e.g. \texttt{fill} or \texttt{eval}).

- When the context allows it, a formula is evaluated \textit{for each pixel} of the selected images (e.g. \texttt{fill} or \texttt{eval}).

- A \textit{math expression may return or take as an argument a scalar or a vector-valued result (with a fixed number of components).}

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

#### Usual math operators:

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

#### Usual math functions:

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

- \texttt{cov(A,B,_avgA,_avgB)} estimates the covariance between vectors A and B (estimated averages of these vectors may be specified as arguments).
• \text{mse}(A, B) \text{ returns the mean-squared error between vectors } A \text{ and } B.
• \text{atan2}(y, x) \text{ is the version of } \text{atan}() \text{ with two arguments } y \text{ and } x \text{ (as in C/C++)}.
• \text{perm}(k, n, \_\text{with_order}) \text{ computes the number of permutations of } k \text{ objects from a set of } n \text{ objects}.
• \text{gauss}(x, _\text{sigma}, _\text{is_normalized}) \text{ returns } \frac{\exp(-x^2/(2*\text{sigma}^2))}{(\text{is_normalized)? \sqrt(2*pi*\text{sigma}^2):1)}.
• \text{cut}(\text{value}, \text{min}, \text{max}) \text{ returns value if it is in range } [\text{min}, \text{max}], \text{ or min or max otherwise}.
• \text{narg}(a_1, \ldots, a_N) \text{ returns the number of specified arguments (here, } N\).
• \text{arg}(i, a_1, \ldots, a_N) \text{ returns the } i\text{-th argument } a_i.
• \text{isnum}(), \text{isnan}(), \text{isinf}(), \text{isint}(), \text{isbool}() \text{ test the type of the given number or expression, and return 0 (false) or 1 (true).}
• \text{isfile}('\text{path}') \text{ (resp. } \text{isdir}('\text{path}') \text{ returns '0 (false) or 1 (true) whether its string argument is a path to an existing file (resp. to a directory) or not.}
• \text{ispercentage}(\text{arg}) \text{ returns 1 (true) or 0 (false) whether } \text{arg} \text{ ends with a } % \text{ or not.}
• \text{isvarname}('\text{str}') \text{ returns 0 (false) or 1 (true) whether its string argument would be a valid to name a variable or not.}
• \text{isin}(v, a_1, \ldots, a_n) \text{ returns 0 (false) or 1 (true) whether the first argument } v \text{ appears in the set of other argument } a_i.
• \text{isint}(x, _\text{xmin}, _\text{xmax}) \text{ returns 1 (true), if } x \text{ is an integer in range } [\text{xmin}, \text{xmax}], \text{ otherwise 0 (false).}
• \text{inrange}(\text{value}, \text{m}, \text{M}, \_\text{include_m}, \_\text{include_M}) \text{ returns 0 (false) or 1 (true) whether the specified value lies in range } [\text{m}, \text{M}] \text{ or not (include_m and includeM tells how boundaries } \text{m} \text{ and } \text{M} \text{ are considered).}
• \text{argkth}(), \text{argmin}(), \text{argmax}(), \text{argminabs}(), \text{argmaxabs}(), 'avg', \text{avg}(), \text{kth}(), \text{min}(), \text{max}(), \text{minabs}(), \text{maxabs}(), \text{med}(), \text{prod}(), \text{std}(), \text{sum}() \text{ and } \text{var}() \text{ can be called with an arbitrary number of scalar/vector arguments.}
• \text{vargkth}(), \text{vargmin}(), \text{vargmax}(), \text{vargminabs}(), \text{vargmaxabs}(), 'vavg', \text{vavg}(), \text{vkth}(), \text{vmin}(), \text{vmax}(), \text{vminabs}(), \text{vmaxabs}(), \text{vmed}(), \text{vprod}(), \text{vstd}(), \text{vsum}() \text{ and } \text{vvar}() \text{ are the versions of the previous function with vector-valued arguments.}
• \text{round}(\text{value}, \text{rounding_value}, \text{direction}) \text{ returns a rounded value. direction can be } \{-1: \text{to-lowest} \mid 0: \text{to-nearest} \mid 1: \text{to-highest}\}.
• \text{lerp}(a, b, t) \text{ returns } a*(1-t)+b*t.
• \text{swap}(a, b) \text{ swaps the values of the given arguments.}

\textbf{Predefined variable names:}

Variable names below are pre-defined. They can be overridden though.
• l: length of the associated list of images.
• k: index of the associated image, in [0, l-1].
• w: width of the associated image, if any (0 otherwise).
• h: height of the associated image, if any (0 otherwise).
• d: depth of the associated image, if any (0 otherwise).
• s: spectrum of the associated image, if any (0 otherwise).
• r: shared state of the associated image, if any (0 otherwise).
• wh: shortcut for width*height.
- whd: shortcut for width*height*depth.
- whds: shortcut for width*height*depth*spectrum (i.e. number of image values).
- im, iM, ia, iv, id, is, ip, ic, in: Respectively the minimum, maximum, average, variance, standard deviation, sum, product, median value and L2-norm of the associated image, if any (0 otherwise).
- xm, ym, zm, cm: The pixel coordinates of the minimum value in the associated image, if any (0 otherwise).
- xM, yM, zM, cM: The pixel coordinates of the maximum value in the associated image, if any (0 otherwise).
- All these variables are considered as constant values by the math parser (for optimization purposes) which is indeed the case most of the time. Anyway, this might not be the case, if function resize(#ind,..) is used in the math expression. If so, it is safer to invoke functions l(), w(#ind), h(#ind), ... s(#ind) and in(#ind) instead of the corresponding named variables.
- i: current processed pixel value (i.e. value located at $(x, y, z, c)$) in the associated image, if any (0 otherwise).
- iN: N-th channel value of current processed pixel (i.e. value located at $(x, y, z, N)$ in the associated image, if any (0 otherwise). N must be an integer in range $[0,9]$.
- R, G, B and A are equivalent to $i0, i1, i2$ and $i3$ respectively.
- I: current vector-valued processed pixel in the associated image, if any (0 otherwise). The number of vector components is equal to the number of image channels (e.g. $I = [R, G, B]$ for a RGB image).
- You may add #ind to any of the variable name above to retrieve the information for any numbered image $[ind]$ of the list (when this makes sense). For instance $ia#0$ denotes the average value of the first image of the list).
- x: current processed column of the associated image, if any (0 otherwise).
- y: current processed row of the associated image, if any (0 otherwise).
- z: current processed slice of the associated image, if any (0 otherwise).
- c: current processed channel of the associated image, if any (0 otherwise).
- t: thread id when an expression is evaluated with multiple threads (0 means master thread).
- n: maximum number of threads when expression is evaluated in parallel (so that t goes from 0 to n-1).
- e: value of e, i.e. $2.71828\ldots$.
- pi: value of pi, i.e. $3.1415926\ldots$.
- eps: value of machine epsilon, that is the difference between 1.0 and the next value representable by a double.
- u: a random value between $[0,1]$, following a uniform distribution.
- v: a random integer that is either 0 or 1, following a uniform distribution.
- g: a random value, following a gaussian distribution of variance 1 (roughly in $[-6,6]$).
- interpolation: value of the default interpolation mode used when reading pixel values with the pixel access operators (i.e. when the interpolation argument is not explicitly specified, see below for more details on pixel access operators). Its initial default value is 0.
- boundary: value of the default boundary conditions used when reading pixel values with the pixel access operators (i.e. when the boundary condition argument is not explicitly specified, see below for more details on pixel access operators). Its initial default value is 0.
• The last image of the list is always associated to the evaluations of expressions, e.g. G’MIC sequence

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

will create a 256x128 image filled with value 256.

**Vector-valued functions and operators:**

The math evaluator is able to work with vector-valued elements. A math function applied on a vector-valued argument usually returns a vector with same dimension, where each element of the input vector has been passed to the specified function (e.g. abs([-1,2,-3]) returns [1,2,3]).

There are specific functions and operators to define or compute vector-valued elements though:
- \([a_0,a_1,...,a_{N-1}]\) defines a N-dimensional vector with scalar coefficients \(a_k\).
- \(\text{vectorN}(a_0,a_1,...,a_{N-1})\) does the same, with the \(a_k\) being repeated periodically if only a few are specified.
- \(\text{vector}(\#N,a_0,a_1,...,a_{N-1})\) does the same, and can be used for any constant expression \(N\).
- In previous expressions, the \(a_k\) can be vectors themselves, to be concatenated into a single vector.
- The scalar element \(a_k\) 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: \(\text{cross}(X,Y)\) (cross product), \(\text{dot}(X,Y)\) (dot product), \(\text{size}(X)\) (vector dimension), \(\text{sort}(X,\text{is\_increasing},\text{nb\_elts},\text{size\_elt})\) (sorted values), \(\text{reverse}(A)\) (reverse order of components), \(\text{map}(X,P,\text{nb\_channels}X,\text{nb\_channels}P,\text{boundary\_conditions})\), \(\text{shift}(A,\text{length},\text{boundary\_conditions})\) and \(\text{same}(A,B,\text{nb\_vals},\text{is\_case\_sensitive})\) (vector equality test).
- Function \(\text{normP}(u_1,...,u_n)\) computes the LP-norm of the specified vector \((P\) being a constant or inf, as in e.g. \(\text{norm1}()\)).
- Function \(\text{normp}(V,p)\) computes the Lp-norm of the specified vector \(V\). Here, \(p\) can be variable. Default value for \(p\) is 2.
- Function \(\text{unitnorm}(V,p)\) returns a normalized version \(V/\text{normp}(V)\) of specified vector \(V\). Default value for \(p\) is 2.
- Function \(\text{resize}(A,\text{size},\text{interpolation},\text{boundary\_conditions})\) 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 \(\text{find}(A,B,\text{starting\_index},\text{search\_step})\) returns the index where sub-vector \(B\) appears in vector \(A\), (or \(-1\) if \(B\) is not contained in \(A\)). Argument \(A\) can be also replaced by an image index \#ind.
- Specifying a vector-valued math expression as an argument of a command that operates on image values (e.g. \(\text{fill}\)) 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.

**Complex-valued functions:**
A 2-dimensional vector may be seen as a complex number and used in those particular functions/operators: ** (complex multiplication), // (complex division), ^^ (complex exponentiation), **= (complex self-multiplication), //= (complex self-division), ^= (complex self-exponentiation), cabs() (complex modulus), carg() (complex argument), cconj() (complex conjugate), cexp() (complex exponential), clog() (complex logarithm), ccos() (complex cosine), csin() (complex sine), csqr() (complex square), csqrt() (complex square root), ctn() (complex tangent), ccosh() (complex hyperbolic cosine), csinh() (complex hyperbolic sine) and ctanh() (complex hyperbolic tangent).

Matrix-valued functions:

A \(M \times N\)-dimensional vector may be seen as a \(M\times N\) matrix and used in those particular functions/operators: * (matrix-vector multiplication), det(A) (determinant), diag(V) (diagonal matrix from a vector), eig(A) (eigenvalues/eigenvectors), eye(n) (n\times n identity matrix), invert(A_nb_colsA_use_LU_lambda) (matrix inverse), mul(A,B_nb_colsB) (matrix-matrix multiplication), rot(u,v,\theta,angle) (3D rotation matrix), rot(\theta) (2D rotation matrix), solve(A,B_nb_colsB_use_LU) (solver of linear system \(A.X = B\)), svd(A_nb_colsA) (singular value decomposition), trace(A) (matrix trace) and transpose(A_nb_colsA) (matrix transpose). Argument nb_colsB may be omitted if it is equal to 1.

Image-valued functions:

Some functions takes vector-valued arguments that represent image data:
- Function expr(formula,w,h,d,s) outputs a vector of size \(w*\text{h}*d*s\) with values generated from the specified formula, as if one were filling an image with dimensions (w,h,d,s).
- Function resize(A,wA,hA,dA,sA,nwA_nhA_ndA_nsA_interpolation_boundary_conditions_ax_ay_az_ac) is an extended version of the resize() function. It allows to resize the vector A, seen as an image of size (ow,oh,od,os) as a new image of size (nw,nh,nd,ns), with specified resizing options.
- Function warp(A,wA,hA,dA,sA,B,wB,hB,dB,sB_mode_interpolation_boundary_conditions) returns the warped version of the image A (of size (wA,hA,dA,sA), viewed as a vector of size \(wA*hA*dA*sA\)) by the warping field B (of size (wB,hB,dB,sB)). The resulting image has size (wB,hB,dB,sA).
- Function index(A,P_nb_channelsP_dithering_map_colors) returns the indexed version of the image A by the colormap P. This is the math evaluator analog to command index.
- Function permute(A,wA,hA,dA,sA,permutation_string) returns a permuted version of the image A (of size (wA,hA,dA,sA), viewed as a vector of size \(wA*hA*dA*sA\)). This is the math evaluator analog to command permute.
- Function mirror(A,wA,hA,dA,sA,axes_string) returns a mirrored version of the image A (of size (wA,hA,dA,sA), viewed as a vector of size \(wA*hA*dA*sA\)). This is the math evaluator analog to command mirror.
- Function cumulate(A,wA,hA,dA,sA_axes_string) returns a cumulated version of the image A (of size (wA,hA,dA,sA), viewed as a vector of size \(wA*hA*dA*sA\)). This is the math evaluator analog to command cumulate.
- Function histogram(A_nb_levels_min_value_max_value) returns the histogram of the vector A. This is the math evaluator analog to command histogram.
- Function equalize(A_nb_levels_min_value_max_value) returns the equalized version of the vector A. This is the math evaluator analog to command equalize.
- Function normalize(A_min_value_max_value) returns the normalized version of the vector A. This is the math evaluator analog to command normalize.
- mproj(S,nb_colsS,D_nb_colsD,method_max_iter_max_residual) projects a matrix S onto a dictionary (matrix) D. This is the math evaluator analog to command mproj.
• Function noise(A,amplitude,_noise_type) returns the noisy version of the vector A. This is the math evaluator analog to command noise.
• Function rand(#size,_min_value,_max_value,_pdf,_precision) returns the a vector of size random values. This is the math evaluator analog to command rand.

String manipulation:

Character strings are defined as vectors objects and can be then managed as is. Dedicated functions and initializers to manage strings exist:
• ['string'] and 'string' define a vector whose values are the character codes of the specified character string (e.g. 'foo' is equal to [ 102,111,111 ]).
• _'character'_ returns the (scalar) byte code of the specified character (e.g. _'A' is equal to 65).
• A special case happens for empty strings: Values of both expressions [''] and '' are 0.
• Functions lowercase() and uppercase() return string with all string characters lowercased or uppercased.
• Function s2v(str,_starting_index,_is_strict) parses specified string str and returns the value contained in it.
• Function v2s(expr,_nb_digits,_siz) returns a vector of size siz which contains the character representation of values described by expression expr. nb_digits can be { <-1:0-padding of integers | -1:auto-reduced | 0:all | >0:max number of digits }.
• Function echo(str1,str2,...,strN) prints the concatenation of given string arguments on the console.
• Function string(_#siz,str1,str2,...,strN) generates a vector corresponding to the concatenation of given string/number arguments.

Dynamic arrays:

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

A dynamic array can be manipulated in a math expression, with the following functions:
• da_size(_#ind): Return the number of elements in dynamic array [ind].
• da_back(_#ind): Return the last element of the dynamic array [ind].
• da_insert(_#ind,pos,elt_1,_,elt_2,_,...,elt_N): Insert N new elements elt_k starting from index pos in dynamic array [ind].
• da_push(_#ind,elt1,_,elt2,_,...,eltN): Insert N new elements elt_k at the end of dynamic array [ind].
• da_pop(_#ind): Same as da_back() but also remove last element from the dynamic array [ind].
• da_push_heap(_#ind,elt1,_,elt2,_,...,eltN) and da_pop_heap(_#ind) does the same but for a dynamic array viewed as a min-heap structure.
• da_remove(_#ind,_start,_end): Remove elements located between indices start and end (included) in dynamic array [ind].
• da_freeze(_#ind): Convert a dynamic array into a 1-column image with height da_size(#ind).
• The value of the k-th element of dynamic array [ind] is retrieved with i[_#ind,k] (if the element is a scalar value), or I[_#ind,k] (if the element is a vector).

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

- `;`: expression separator. The returned value is always the last encountered expression. For instance expression `1;2;pi` is evaluated as `pi`.

- `=`: variable assignment. Variables in mathematical parser can only refer to numerical values (vectors or scalars). Variable names are case-sensitive. Use this operator in conjunction with `;` to define more complex evaluable expressions, such as

\[
t = \cos(x); 3^t + 2 + 2^t + 1
\]

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

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

Specific functions:

- `addr(expr)`: return the pointer address to the specified expression `expr`.

- `o2c(_ind,offset)` and `c2o(_ind,x,y,z,c)`: Convert image offset to image coordinates and vice-versa.

- `fill(target,expr)` or `fill(target,index_name,expr)` fill the content of the specified target (often vector-valued) using a given expression, e.g. `V = vector16(); fill(V,k,k^2 + k + 1);`. For a vector-valued target, it is basically equivalent to:

```c
for (index_name = 0, index_name<size(target), +
    +index_name, target[index_name] = expr);
```

- `u(max)` or `u(min,max,include_min,include_max)`: return a random value in range `0...max` or `min...max`, following a uniform distribution. Each range extremum can be included (default) in the distribution or not.

- `v(max)` or `v(min,max,include_min,include_max)` do the same but returns an integer in specified range.

- `f2ui(value)` and `ui2f(value)`: Convert a large unsigned integer as a negative floating point value (and vice-versa), so that 32bits floats can be used to store large integers while keeping a unitary precision.

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

\[
\text{fill } 0.5*(i(x+1)-i(x-1))
\]

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[#ind, offset, boundary_conditions] and I/J(#ind, x, y, z, interpolation, boundary_conditions) do the same as i/j[#ind, offset, boundary_conditions] and i/j(#ind, x, y, z, c, interpolation, boundary_conditions) but return a vector instead of a scalar (e.g. a vector [ R, G, B ] for a pixel at (a, b, c) in a color image).

• crop(#ind, x, y, z, c, dx, dy, dz, dc, boundary_conditions) returns a vector whose values come from the cropped region of image [ind] (or from default image selected if ind is not specified). Cropped region starts from point (x, y, z, c) and has a size of (dx, dy, dz, 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).

* crop(S, w, h, d, s, x, y, z, c, dx, dy, dz, dc, boundary_conditions) does the same but extracts the cropped data from a vector S, viewed as an image of size (w, h, d, s).

• draw(_#ind, S, x, y, z, c, dx, dy, dz, dc, opacity, opacity_mask, max_opacity_mask) draws a sprite S in image [ind] (or in default image selected if ind is not specified) at coordinates (x, y, z, c).

• draw(D, w, h, s, d, S, x, y, z, c, dx, dy, dz, dc, opacity, M, max_M) does the same but draw the sprite S in the vector D, viewed as an image of size (w, h, d, 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 an 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).

• flood(_#ind, x, y, z, tolerance, is_high_connectivity, opacity, color) performs a flood fill in image [ind] (or in default image selected if ind is not specified) with specified coordinates. This is the math evaluator analog to command flood.

• 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

```gmic
fill if(!(x%10),255,i)
```

and

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

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

• do(expression, condition) repeats the evaluation of expression until condition vanishes (or until expression vanishes if no condition is specified). For instance, the expression:
if(N<2,N,n=N-1;F0=0;F1=1;do(F2=F0+F1;F0=F1;F1=F2,n=n-1))

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

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

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

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

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

- while(condition,expression) is exactly the same as for(init,condition,expression) without the specification of an initializing expression.

- repeat(nb_iters,expr) or fill(nb_iters,iter_name,expr) run nb_iters iterations of the specified expression expr, e.g. V = vector16(); repeat(16,k,V[k] = k^2 + k + 1); It is basically equivalent to: for (iter_name = 0, iter_name<nb_iters, ++iter_name, expr);

- break() and continue() respectively breaks and continues the current running block.

- fsize('filename') returns the size of the specified filename (or -1 if file does not exist).

- date(attr,'path') returns the date attribute for the given path (file or directory), with attr being { 0:year | 1:month | 2:day | 3:day of week | 4:hour | 5:minute | 6:second }, or a vector of those values.

- date(_attr) returns the specified attribute for the current (locale) date (attributes being { 0...6:same meaning as above | 7:milliseconds })

- print(expr1,expr2,...) or print(#ind) prints the value of the specified expressions (or image information) on the console, and returns the value of the last expression (or nan in case of an image). Function prints(expr) also prints the string composed of the character codes defined by the vector-valued expression (e.g. prints('Hello')).

- debug(expression) prints detailed debug info about the sequence of operations done by the math parser to evaluate the expression (and returns its value).

- display(X,_w,_h,_d,_s) or display(#ind) display the contents of the vector X (or specified image) and wait for user events. if no arguments are provided, a memory snapshot of the math parser environment is displayed instead.

- begin(expression) and end(expression) evaluates the specified expressions only once, respectively at the beginning and end of the evaluation procedure, and this, even when multiple evaluations are required (e.g. in 'fill "begin(foo = 0); ++foo"').

- copy(dest,src,_nb_elts,_inc_d,_inc_s,_opacity) copies an entire memory block of nb_elts elements starting from a source value src to a specified destination dest, with increments defined by inc_d and inc_s respectively for the destination and source pointers.

- stats(_#ind) returns the statistics vector of the running image [ind], i.e the vector [ im,iM,ia,iv,xm,ym,zm,cm,xM,yM,zM,cM,is,ip ] (14 values).

- ref(expr,a) references specified expression expr as variable name a.

- unref(a,b,...) destroys references to the named variable given as arguments.

- breakpoint() inserts a possible computation breakpoint (useless with the cli interface).

- _(comment) expr just returns expression expr (useful for inserting inline comments in math expressions).

- run('pipeline)' executes the specified G'MIC pipeline as if it was called outside the currently
evaluated expression.

- set('variable_name', A) sets the G'MIC variable $variable_name with the value of expression A. If A is a vector-valued variable, it is assumed to encode a string.

- store('variable_name', A, _w, _h, _d, _s, _is_compressed) transfers the data of vector A as a \((w, h, d, s)\) image to the G'MIC variable $variable_name. Thus, the data becomes available outside the math expression (that is equivalent to using the regular command store, but directly in the math expression).

- get('variable_name', _size, _return_as_string) returns the value of the specified variable, as a vector of size values, or as a scalar (if size is zero or not specified).

- name(_#ind, size) returns a vector of size size, whose values are the characters codes of the name of image [ind] (or default image selected if ind is not specified).

- correlate(I, wI, hI, dI, sI, K, wK, hK, dK, sK, _boundary_conditions, _is_normalized, _channel_mode, _xcenter, _ycenter) returns the correlation, unrolled as a vector, of the \((wI, hI, dI, sI)\) -sized image I with the \((wK, hK, dK, sK)\) -sized kernel K (the meaning of the other arguments are the same as in command correlate). Similar function convolve(...) is also defined for computing the convolution between I and K.

User-defined macros:

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

  
  \[
  \text{foo}(x, y) = \cos(x + y); \text{result} = \text{foo}(1, 2) + \text{foo}(2, 3)
  \]

- 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,

  \[
  \text{foo}(x) = x + x; z = 0; \text{foo}(++z)
  \]

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

  \[
  \text{foo}(x, y) = x*y; \text{foo}(1+2, 3)
  \]

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

  \[
  \text{foo}(x, y) = x#*y#; \text{foo}(1+2, 3)
  \]

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

  \[
  \text{str}(N) = ['I like N#']
  \]

- Macros with variadic arguments can be defined, by specifying a single argument name followed by . . . . For instance,
foo(args...) = sum([ args ]^2);

defines a macro that returns the sum of its squared arguments, so foo(1, 2, 3) returns 14 and foo(4, 5) returns 41.

Multi-threaded and in-place evaluation:

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

Adding Custom Commands

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

```
command_name: command_definition
```

or

```
command_definition (continuation)
```

- At startup, G'MIC automatically includes user's command file $HOME/.gmic (on Unix) or %USERPROFILE%\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 verbatim copy of the specified string of arguments (do not include arguments set to default values).
- $**$ is substituted by the sequence of specified arguments, separated by commas ,, each being double-quoted (include arguments set to default values).
- $#$ is substituted by the maximum index of known arguments (either specified by the user or set to a default value in the custom command).
- $[]$ is substituted by the list of selected image indices that have been specified in the command invocation.
- $? is substituted by a printable version of $[]$ to be used in command descriptions.
- $i$ and ${i}$ are both substituted by the $i$-th specified argument. Negative indices such as ${-j}$ are allowed and refer to the $j$-th latest argument. $0$ is substituted by the custom command name.
- ${i=default}$ is substituted by the value of $i$ (if defined) or by its new value set to default otherwise (default may be a $-expression as well).
- ${subset}$ is substituted by the argument values (separated by commas ,) of a specified argument subset. For instance expression ${2--2}$ is substituted by all specified command arguments except the first and the last one. Expression ${^0}$ is then substituted by all arguments of the invoked command (eq. to $*$ if all arguments have been indeed specified).
- $=var$ is substituted by the set of instructions that will assign each argument $i$ to the named variable var$i$ (for i in $[0...$#]). This is particularly useful when a custom command want to manage variable numbers of arguments. Variables names must use character set [a-zA-Z0-9_] and cannot start with a number.

These particular $-expressions for custom commands are always substituted, even in double-quoted items or when the dollar sign $ is escaped with a backslash $. To avoid substitution, place an empty double quoted string just after the $ (as in $"$).

Specifying arguments may be skipped when invoking a custom command, by replacing them by commas , as in expression

```
flower , , 3
```

Omitted arguments are set to their default values, which must be thus explicitly defined in the code of the corresponding custom command (using default argument expressions as ${1=default}$).

If one numbered argument required by a custom command misses a value, an error is thrown by the G'MIC interpreter.

It is possible to specialize the invocation of a +command by defining it as

```
+command_name: command_definition
```

A +-specialization takes priority over the regular command definition when the command is invoked with a prepended +.

When only a +-specialization of a command is defined, invoking command is actually equivalent to +command.

List of Commands

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

The examples accompanying this List of Commands illustrate the use of the \texttt{G'MIC} language and are written as they would appear in a custom command. While some examples may work if entered directly at a shell prompt, there is no guarantee. No attempt has been made to escape special characters in these examples, which many shells reserve.

Categories:

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

Global Options:

\begin{center}
\begin{tabular}{|c|c|c|}
\hline
\texttt{debug} & \texttt{help} & \texttt{version} \\
\hline
\end{tabular}
\end{center}

Input / Output:

\begin{center}
\begin{tabular}{|c|c|c|c|c|}
\hline
\texttt{camera} & \texttt{command} & \texttt{compress_to_keypoints} & \texttt{cursor} & \texttt{delete} \\
\hline
\texttt{display} & \texttt{display0} & \texttt{display_array} & \texttt{display_camera} & \texttt{display_clut} \\
\hline
\texttt{display_fft} & \texttt{display_graph} & \texttt{display_histogram} & \texttt{display_parametric} & \texttt{display_polar} \\
\hline
\texttt{display_quiver} & \texttt{display_rgb} & \texttt{display_tensors} & \texttt{display_voxels_3d} & \texttt{display_warp} \\
\hline
\texttt{echo} & \texttt{echo_file} & \texttt{font} & \texttt{font2gmz} & \texttt{function1d} \\
\hline
\texttt{identity} & \texttt{input} & \texttt{input_565} & \texttt{input_bytes} & \texttt{input_csv} \\
\hline
\end{tabular}
\end{center}
<table>
<thead>
<tr>
<th>input_cube</th>
<th>input_flo</th>
<th>input_glob</th>
<th>input_gpl</th>
<th>input_cached</th>
</tr>
</thead>
<tbody>
<tr>
<td>input_obj</td>
<td>input_text</td>
<td>lorem</td>
<td>network</td>
<td>output</td>
</tr>
<tr>
<td>output_565</td>
<td>output_cube</td>
<td>output_flo</td>
<td>output_ggr</td>
<td>output_gmz</td>
</tr>
<tr>
<td>output_obj</td>
<td>output_text</td>
<td>outputn</td>
<td>outputp</td>
<td>outputw</td>
</tr>
<tr>
<td>outputx</td>
<td>parse_cli</td>
<td>parse_gmd</td>
<td>gmd2html</td>
<td>gmd2ascii</td>
</tr>
<tr>
<td>parse_gui</td>
<td>pass</td>
<td>plot</td>
<td>poincare_disk</td>
<td>portrait</td>
</tr>
<tr>
<td>print</td>
<td>random_pattern</td>
<td>screen</td>
<td>select</td>
<td>serialize</td>
</tr>
<tr>
<td>shape_circle</td>
<td>shape_cupid</td>
<td>shape_diamond</td>
<td>shape_dragon</td>
<td>shape_fern</td>
</tr>
<tr>
<td>shape_gear</td>
<td>shape_heart</td>
<td>shape_menger</td>
<td>shape_mosely</td>
<td>shape_polygon</td>
</tr>
<tr>
<td>shape_rays</td>
<td>shape_snowflake</td>
<td>shape_star</td>
<td>shared</td>
<td>sample</td>
</tr>
<tr>
<td>srand</td>
<td>store</td>
<td>testimage2d</td>
<td>uncommand</td>
<td>uniform_distribution</td>
</tr>
<tr>
<td>unserializable</td>
<td>update</td>
<td>verbose</td>
<td>wait</td>
<td>warn</td>
</tr>
<tr>
<td>window</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**List Manipulation:**

<table>
<thead>
<tr>
<th>keep</th>
<th>keep_named</th>
<th>move</th>
<th>name</th>
<th>remove</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>remove_duplicates</td>
<td>remove_empty</td>
<td>remove_named</td>
<td>reverse</td>
<td>sort_list</td>
</tr>
</tbody>
</table>

**Mathematical Operators:**

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

**Values Manipulation:**

<table>
<thead>
<tr>
<th>apply_curve</th>
<th>apply_gamma</th>
<th>balance_gamma</th>
<th>cast</th>
<th>complex2polar</th>
</tr>
</thead>
<tbody>
<tr>
<td>compress_clut</td>
<td>compress_huffman</td>
<td>compress_rle</td>
<td>cumulate</td>
<td>cut</td>
</tr>
<tr>
<td>decompress_clut</td>
<td>decompress_from_keypoints</td>
<td>decompress_huffman</td>
<td>decompress_rle</td>
<td>discard</td>
</tr>
<tr>
<td>eigen2tensor</td>
<td>endian</td>
<td>equalize</td>
<td>fill</td>
<td>index</td>
</tr>
<tr>
<td>inrange</td>
<td>map</td>
<td>mix_channels</td>
<td>negate</td>
<td>noise_perlin</td>
</tr>
<tr>
<td>noise_poisson_disk</td>
<td>normp</td>
<td>norm1</td>
<td>norm2</td>
<td>normalize</td>
</tr>
<tr>
<td>normalize_l2</td>
<td>normalize_sum</td>
<td>orientation</td>
<td>oneminus</td>
<td>otsu</td>
</tr>
<tr>
<td>polar2complex</td>
<td>quantize</td>
<td>quantize_area</td>
<td>rand</td>
<td>rand_sum</td>
</tr>
<tr>
<td>replace</td>
<td>replace_inf</td>
<td>replace_infna</td>
<td>replace_nan</td>
<td>replace_seq</td>
</tr>
<tr>
<td>replace_str</td>
<td>round</td>
<td>roundify</td>
<td>set</td>
<td>threshold</td>
</tr>
<tr>
<td>vector2tensor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Colors:**

| adjust_colors | apply_channels | autoindex | bayer2rgb | cm | clut |
| clut2hald | hald2clut | cmy2rgb | cmyk2rgb | colorblind |
| colormap | compose_channels | count_colors | deltaE | direction2rgb |
| ditheredbw | fill_color | gradient2rgb | hcy2rgb | hsi2rgb |
| hsi82rgb | hsl2rgb | hsi82rgb | hsv2rgb | hsv82rgb |
| int2rgb | ipremula | jzazbz2rgb | jzazbz2xyz | lab2lch |
| lab2rgb | lab2srgb | lab82srgb | lab2xyz | lab82rgb |
| lch2lab | lch2rgb | lch82rgb | luminance | lightness |
| lut_contrast | map_clut | match_histogram | match_icp | match_pca |
| match_rgb | mix_rgb | oklab2rgb | palette | premula |
| pseudogray | random_clut | random_clut | replace_color | retinex |
| rgb2bayer | rgb2cmy | rgb2cmyk | rgb2hcy | rgb2hsi |
| rgb2hsi8 | rgb2hsi | rgb2hsi8 | rgb2hsv | rgb2hsv8 |
| rgb2int | rgb2jzazbz | rgb2lab | rgb2lab8 | rgb2lch |
| rgb2lch8 | rgb2luv | rgb2oklab | rgb2ryb | rgb2srgb |
| rgb2xyz | rgb2xyz8 | rgb2yiq | rgb2yiq8 | rgb2ycbcr |
| rgb2yuv | rgb2yuv8 | remove_opacity | ryb2rgb | select_color |
| sepia | solarize | split_colors | split_opacity | split_vector |
| srgb2lab | srgb2lab8 | srgb2rgb | to_a | to_color |
| to_colormode | to_gray | to_graya | to_pseudogray | to_rgb |
| to_rgb | to_automode | xyz2jzazbz | xyz2lab | xyz2rgb |
| xyz82rgb | ycbcr2rgb | yiq2rgb | yiq82rgb | yuv2rgb |
| yuv82rgb |
Geometry Manipulation:

<table>
<thead>
<tr>
<th>append</th>
<th>append_tiles</th>
<th>apply_scales</th>
<th>autocrop</th>
<th>autocrop_components</th>
</tr>
</thead>
<tbody>
<tr>
<td>autocrop_seq</td>
<td>channels</td>
<td>columns</td>
<td>crop</td>
<td>diagonal</td>
</tr>
<tr>
<td>elevate</td>
<td>expand</td>
<td>extract</td>
<td>extract_region</td>
<td>montage</td>
</tr>
<tr>
<td>mirror</td>
<td>permute</td>
<td>rescale2d</td>
<td>rescale3d</td>
<td>resize</td>
</tr>
<tr>
<td>resize_as_image</td>
<td>resize_mn</td>
<td>resize_pow2</td>
<td>rotate</td>
<td>rotate_tileable</td>
</tr>
<tr>
<td>rows</td>
<td>scale2x</td>
<td>scale3x</td>
<td>scale_dcci2x</td>
<td>seamcarve</td>
</tr>
<tr>
<td>shift</td>
<td>shrink</td>
<td>slices</td>
<td>sort</td>
<td>split</td>
</tr>
<tr>
<td>split_tiles</td>
<td>undistort</td>
<td>unroll</td>
<td>upscale_smart</td>
<td>volumetric2d</td>
</tr>
</tbody>
</table>

Filtering:

<table>
<thead>
<tr>
<th>bandpass</th>
<th>bilateral</th>
<th>blur</th>
<th>blur_angular</th>
<th>blur_bloom</th>
</tr>
</thead>
<tbody>
<tr>
<td>blur_linear</td>
<td>blur_radial</td>
<td>blur_selective</td>
<td>boxfilter</td>
<td>bump2normal</td>
</tr>
<tr>
<td>closing</td>
<td>closing_circ</td>
<td>compose_freq</td>
<td>convolve</td>
<td>convolve_fft</td>
</tr>
<tr>
<td>correlate</td>
<td>cross_correlation</td>
<td>curvature</td>
<td>dct</td>
<td>deblur</td>
</tr>
<tr>
<td>deblur_golden_el</td>
<td>deblur_richardson</td>
<td>deconvolve_fft</td>
<td>deinterlace</td>
<td>denoise</td>
</tr>
<tr>
<td>denoise_haar</td>
<td>denoise_cnn</td>
<td>denoise_patch_pca</td>
<td>deriche</td>
<td>dilate</td>
</tr>
<tr>
<td>dilate_circ</td>
<td>dilate_oct</td>
<td>dilate_threshold</td>
<td>divergence</td>
<td>dog</td>
</tr>
<tr>
<td>diffusion_tensor</td>
<td>edges</td>
<td>erode</td>
<td>erode_circ</td>
<td>erode_oct</td>
</tr>
<tr>
<td>erode_threshold</td>
<td>fft</td>
<td>gradient</td>
<td>gradient_normal</td>
<td>gradient_orientation</td>
</tr>
<tr>
<td>guided</td>
<td>haar</td>
<td>heat_flow</td>
<td>hessian</td>
<td>idct</td>
</tr>
<tr>
<td>iee</td>
<td>ifft</td>
<td>haar</td>
<td>ilaplacian</td>
<td>inn</td>
</tr>
<tr>
<td>inpaint</td>
<td>inpaint_pde</td>
<td>inpaint_flow</td>
<td>inpaint_holes</td>
<td>inpaint_morpho</td>
</tr>
<tr>
<td>inpaint_match_patch</td>
<td>kuwahara</td>
<td>laplacian</td>
<td>lic</td>
<td>map_tones</td>
</tr>
<tr>
<td>map_tones_fast</td>
<td>meancurvature_flow</td>
<td>median</td>
<td>merge_alpha</td>
<td>nlmeans</td>
</tr>
<tr>
<td>nlmeans_core</td>
<td>normalize_local</td>
<td>normalized_cross_correlation</td>
<td>opening</td>
<td>opening_circ</td>
</tr>
<tr>
<td>percentile</td>
<td>peronamalik_flow</td>
<td>phase_correlation</td>
<td>pde_flow</td>
<td>periodize_poison</td>
</tr>
<tr>
<td>rbf</td>
<td>red_eye</td>
<td>remove_hotpixels</td>
<td>remove_pixels</td>
<td>rolling_guidance</td>
</tr>
<tr>
<td>sharpen</td>
<td>sharpen_alph a</td>
<td>smooth</td>
<td>split_freq</td>
<td>solve_poisson</td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
<td>--------</td>
<td>------------</td>
<td>---------------</td>
</tr>
<tr>
<td>split_alpha</td>
<td>split_details</td>
<td>structuretensors</td>
<td>solidify</td>
<td>syntexturize</td>
</tr>
<tr>
<td>syntexturize_matchpatch</td>
<td>tv_flow</td>
<td>unsharp</td>
<td>unsharp_octave</td>
<td>vanvliet</td>
</tr>
<tr>
<td>voronoi</td>
<td>watermark_fourier</td>
<td>watershed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Features Extraction:

<table>
<thead>
<tr>
<th>area</th>
<th>area_fg</th>
<th>at_line</th>
<th>at_quadrangle</th>
<th>barycenter</th>
<th>displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>betti</td>
<td>canny</td>
<td>delaunay</td>
<td>detect_skin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>distance</td>
<td>edgels</td>
<td>fftpolar</td>
<td>histogram</td>
<td>histogram_masked</td>
<td></td>
</tr>
<tr>
<td>histogram_nd</td>
<td>histogram_cumul</td>
<td>histogram_pointwise</td>
<td>hough</td>
<td>huffman_tree</td>
<td></td>
</tr>
<tr>
<td>ifftpolar</td>
<td>img2patches</td>
<td>isophotes</td>
<td>label</td>
<td>label_fg</td>
<td></td>
</tr>
<tr>
<td>laar</td>
<td>max_patch</td>
<td>min_patch</td>
<td>minimal_path</td>
<td>mse</td>
<td></td>
</tr>
<tr>
<td>mse_matrix</td>
<td>patches2img</td>
<td>patches</td>
<td>matchpatch</td>
<td>plot2value</td>
<td></td>
</tr>
<tr>
<td>pointcloud</td>
<td>psnr</td>
<td>psnr_matrix</td>
<td>segment_watershed</td>
<td>shape2bump</td>
<td></td>
</tr>
<tr>
<td>skeleton</td>
<td>slic</td>
<td>ssd_patch</td>
<td>ssim</td>
<td>ssim_matrix</td>
<td></td>
</tr>
<tr>
<td>thinning</td>
<td>tones</td>
<td>topographic_map</td>
<td>tsp</td>
<td>variance_patch</td>
<td></td>
</tr>
</tbody>
</table>

### Image Drawing:

<table>
<thead>
<tr>
<th>arrow</th>
<th>axes</th>
<th>ball</th>
<th>chessboard</th>
<th>cie1931</th>
</tr>
</thead>
<tbody>
<tr>
<td>circle</td>
<td>close_binary</td>
<td>curve</td>
<td>ellipse</td>
<td>flood</td>
</tr>
<tr>
<td>gaussian</td>
<td>graph</td>
<td>grid</td>
<td>image</td>
<td>imagealpha</td>
</tr>
<tr>
<td>line</td>
<td>line_aa</td>
<td>spline</td>
<td>thickline</td>
<td>thick spline</td>
</tr>
<tr>
<td>mandelbrot</td>
<td>marble</td>
<td>maze</td>
<td>maze_mask</td>
<td>newton_fractal</td>
</tr>
<tr>
<td>object3d</td>
<td>pack_sprites</td>
<td>piechart</td>
<td>plasma</td>
<td>point</td>
</tr>
<tr>
<td>polka_dots</td>
<td>polygon</td>
<td>quiver</td>
<td>rectangle</td>
<td>rorschach</td>
</tr>
<tr>
<td>sierpinski</td>
<td>spiralbw</td>
<td>tetraedron_shade</td>
<td>text</td>
<td>text_outline</td>
</tr>
<tr>
<td>triangle_shade</td>
<td>truchet</td>
<td>turbulence</td>
<td>yinyang</td>
<td></td>
</tr>
</tbody>
</table>

### Matrix Computation:

<table>
<thead>
<tr>
<th>dijkstra</th>
<th>eigen</th>
<th>eye</th>
<th>fitsamples</th>
<th>invert</th>
</tr>
</thead>
<tbody>
<tr>
<td>meigen</td>
<td>mproj</td>
<td>orthogonalize</td>
<td>poweriteratio</td>
<td>solve</td>
</tr>
</tbody>
</table>
3D Meshes:

<table>
<thead>
<tr>
<th>add3d</th>
<th>animate3d</th>
<th>apply_camera3d</th>
<th>apply_matrix3d</th>
<th>array3d</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrow3d</td>
<td>axes3d</td>
<td>boundingbox3d</td>
<td>box3d</td>
<td>center3d</td>
</tr>
<tr>
<td>chainring3d</td>
<td>circle3d</td>
<td>circles3d</td>
<td>color3d</td>
<td>colorcube3d</td>
</tr>
<tr>
<td>colorize3d</td>
<td>cone3d</td>
<td>cubes3d</td>
<td>cup3d</td>
<td>curve3d</td>
</tr>
<tr>
<td>cylinder3d</td>
<td>delaunay3d</td>
<td>distribution3d</td>
<td>div3d</td>
<td>double3d</td>
</tr>
<tr>
<td>elevation3d</td>
<td>empty3d</td>
<td>extract_textures3d</td>
<td>extrude3d</td>
<td>focale3d</td>
</tr>
<tr>
<td>fov3d</td>
<td>gaussians3d</td>
<td>gmic3d</td>
<td>gyroid3d</td>
<td>histogram3d</td>
</tr>
<tr>
<td>image6cube3d</td>
<td>imageblocks3d</td>
<td>imagecube3d</td>
<td>imageplane3d</td>
<td>imagepyramid3d</td>
</tr>
<tr>
<td>imagerubik3d</td>
<td>imagesphere3d</td>
<td>isolate3d</td>
<td>isosurface3d</td>
<td>label3d</td>
</tr>
<tr>
<td>label_points3d</td>
<td>lathe3d</td>
<td>light3d</td>
<td>line3d</td>
<td>lines3d</td>
</tr>
<tr>
<td>lissajous3d</td>
<td>mode3d</td>
<td>moded3d</td>
<td>mul3d</td>
<td>normalize3d</td>
</tr>
<tr>
<td>opacity3d</td>
<td>parametric3d</td>
<td>pca_patch3d</td>
<td>plane3d</td>
<td>point3d</td>
</tr>
<tr>
<td>pointcloud3d</td>
<td>pose3d</td>
<td>primitives3d</td>
<td>projections3d</td>
<td>pyramid3d</td>
</tr>
<tr>
<td>quadrangle3d</td>
<td>random3d</td>
<td>reverse3d</td>
<td>rotate3d</td>
<td>rotation3d</td>
</tr>
<tr>
<td>sierpinski3d</td>
<td>size3d</td>
<td>skeleton3d</td>
<td>snapshot3d</td>
<td>spec3d</td>
</tr>
<tr>
<td>specs3d</td>
<td>sphere3d</td>
<td>spherical3d</td>
<td>spline3d</td>
<td>split3d</td>
</tr>
<tr>
<td>sprite3d</td>
<td>sprites3d</td>
<td>star3d</td>
<td>streamline3d</td>
<td>sub3d</td>
</tr>
<tr>
<td>subdivide3d</td>
<td>superformula3d</td>
<td>surfels3d</td>
<td>tensors3d</td>
<td>text_pointcloud3d</td>
</tr>
<tr>
<td>text3d</td>
<td>texturize3d</td>
<td>torus3d</td>
<td>triangle3d</td>
<td>volume3d</td>
</tr>
<tr>
<td>voxelize3d</td>
<td>weird3d</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Flow Control:

<table>
<thead>
<tr>
<th>apply_parallel</th>
<th>apply_parallel_channels</th>
<th>apply_parallel_overlap</th>
<th>apply_tiles</th>
<th>apply_timeout</th>
</tr>
</thead>
<tbody>
<tr>
<td>check</td>
<td>check3d</td>
<td>continue</td>
<td>break</td>
<td>do</td>
</tr>
<tr>
<td>done</td>
<td>elif</td>
<td>else</td>
<td>fi</td>
<td>error</td>
</tr>
<tr>
<td>eval</td>
<td>exec</td>
<td>exec_out</td>
<td>for</td>
<td>foreach</td>
</tr>
<tr>
<td>if</td>
<td>local</td>
<td>mutex</td>
<td>noarg</td>
<td>onfail</td>
</tr>
<tr>
<td>parallel</td>
<td>progress</td>
<td>quit</td>
<td>repeat</td>
<td>return</td>
</tr>
<tr>
<td>rprogress</td>
<td>run</td>
<td>skip</td>
<td>status</td>
<td>while</td>
</tr>
</tbody>
</table>

Neural Networks:
### Arrays, Tiles and Frames:

<table>
<thead>
<tr>
<th>array</th>
<th>array_fade</th>
<th>array_mirror</th>
<th>array_random</th>
<th>frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>frame_blur</td>
<td>frame_cube</td>
<td>frame_fuzzy</td>
<td>frame_painting</td>
<td>frame_pattern</td>
</tr>
<tr>
<td>frame_round</td>
<td>frame_seamles</td>
<td>img2ascii</td>
<td>imagegrid_hexagonal</td>
<td>pack</td>
</tr>
<tr>
<td>imagegrid_triangular</td>
<td>linearize_tiles</td>
<td>map_sprites</td>
<td>puzzle</td>
<td></td>
</tr>
<tr>
<td>quadratize_tiles</td>
<td>rotate_tiles</td>
<td>shift_tiles</td>
<td>taquin</td>
<td>tunnel</td>
</tr>
</tbody>
</table>

### Artistic:

<table>
<thead>
<tr>
<th>boxfitting</th>
<th>brushify</th>
<th>cartoon</th>
<th>color_ellipses</th>
<th>cubism</th>
</tr>
</thead>
<tbody>
<tr>
<td>draw_whirl</td>
<td>drawing</td>
<td>drop_shadow</td>
<td>drop_shadow</td>
<td>ellipsionism</td>
</tr>
<tr>
<td>fire_edges</td>
<td>fractalize</td>
<td>glow</td>
<td>halftone</td>
<td>hardsketchbw</td>
</tr>
<tr>
<td>hearts</td>
<td>houghsketchbw</td>
<td>lightrays</td>
<td>light_relief</td>
<td>linify</td>
</tr>
<tr>
<td>mosaic</td>
<td>old_photo</td>
<td>pencilbw</td>
<td>pixelsort</td>
<td>polaroid</td>
</tr>
<tr>
<td>polygonize</td>
<td>poster_edges</td>
<td>poster_hope</td>
<td>rodilius</td>
<td>sketchbw</td>
</tr>
<tr>
<td>sponge</td>
<td>stained_glass</td>
<td>stars</td>
<td>stencil</td>
<td>stencilbw</td>
</tr>
<tr>
<td>stylize</td>
<td>tetriss</td>
<td>warhol</td>
<td>weave</td>
<td>whirls</td>
</tr>
</tbody>
</table>

### Warping:

<table>
<thead>
<tr>
<th>nn_lib</th>
<th>nn_init</th>
<th>nn_check_layere</th>
<th>nn_add</th>
<th>nn_append</th>
</tr>
</thead>
<tbody>
<tr>
<td>nn_avgpool2d</td>
<td>nn_avgpool3d</td>
<td>nn_clone</td>
<td>nn_conv2d</td>
<td>nn_conv2dnnl</td>
</tr>
<tr>
<td>nn_conv2dnnl</td>
<td>nn_conv3d</td>
<td>nn_conv3dnnl</td>
<td>nn_conv3dnnl</td>
<td>nn_crop</td>
</tr>
<tr>
<td>nn_distance</td>
<td>nn_dropout</td>
<td>nn_fc</td>
<td>nn_nlf</td>
<td>nn_fcnl</td>
</tr>
<tr>
<td>nn_fcnl</td>
<td>nn_input</td>
<td>nn_maxpool2d</td>
<td>nn_maxpool3d</td>
<td>nn_mul</td>
</tr>
<tr>
<td>nn_nl</td>
<td>nn_normalize</td>
<td>nn_patchdown2d</td>
<td>nn_patchdown3d</td>
<td>nn_patchup2d</td>
</tr>
<tr>
<td>nn_patchup3d</td>
<td>nn_rename</td>
<td>nn_resconv2d</td>
<td>nn_resconv2dnl</td>
<td>nn_resconv2dnnl</td>
</tr>
<tr>
<td>nn_resconv3d</td>
<td>nn_resconv3dnl</td>
<td>nn_resfc</td>
<td>nn_resfcnl</td>
<td></td>
</tr>
<tr>
<td>nn_resfcnnl</td>
<td>nn_reshape</td>
<td>nn_resize</td>
<td>nn_run</td>
<td>nn_split</td>
</tr>
<tr>
<td>nn_loss_binary_crossentropy</td>
<td>nn_loss_crossentropy</td>
<td>nn_loss_mse</td>
<td>nn_loss_normal</td>
<td>nn_loss_softmax_crossentropy</td>
</tr>
<tr>
<td>nn_print</td>
<td>nn_trainer</td>
<td>nn_size</td>
<td>nn_load</td>
<td>nn_save</td>
</tr>
<tr>
<td>nn_store</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>deform</td>
<td>euclidean2polar</td>
<td>equirectangular2nadirzenith</td>
<td>fisheye</td>
<td>flower</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------</td>
<td>-----------------------------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>kaleidoscope</td>
<td>map_sphere</td>
<td>nadirzenith2equirectangular</td>
<td>polar2euclidean</td>
<td>raindrops</td>
</tr>
<tr>
<td>ripple</td>
<td>rotoidoscope</td>
<td>spherize</td>
<td>symmetrize</td>
<td>transform_polar</td>
</tr>
<tr>
<td>twirl</td>
<td>warp</td>
<td>warp_patch</td>
<td>warp_perspective</td>
<td>warp_rbf</td>
</tr>
<tr>
<td>warp_seamless</td>
<td>water</td>
<td>wave</td>
<td>wind</td>
<td>zoom</td>
</tr>
</tbody>
</table>

### Degradations:

<table>
<thead>
<tr>
<th>cracks</th>
<th>light_patch</th>
<th>noise</th>
<th>noise_hurl</th>
<th>pixelize</th>
</tr>
</thead>
<tbody>
<tr>
<td>scanlines</td>
<td>shade_stripes</td>
<td>shadow_patch</td>
<td>spread</td>
<td>stripes_y</td>
</tr>
<tr>
<td>texturize_canvas</td>
<td>texturize_paper</td>
<td>vignette</td>
<td>watermark_visible</td>
<td></td>
</tr>
</tbody>
</table>

### Blending and Fading:

<table>
<thead>
<tr>
<th>blend</th>
<th>nblend</th>
<th>blend_edges</th>
<th>blend_fade</th>
<th>blend_median</th>
</tr>
</thead>
<tbody>
<tr>
<td>blend_seamless</td>
<td>fade_diamond</td>
<td>fade_linear</td>
<td>fade_radial</td>
<td>fade_x</td>
</tr>
<tr>
<td>fade_y</td>
<td>fade_z</td>
<td>sub_alpha</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Image Sequences and Videos:

<table>
<thead>
<tr>
<th>animate</th>
<th>apply_camera</th>
<th>apply_files</th>
<th>apply_video</th>
<th>average_files</th>
</tr>
</thead>
<tbody>
<tr>
<td>average_video</td>
<td>fade_files</td>
<td>fade_video</td>
<td>files2video</td>
<td>median_files</td>
</tr>
<tr>
<td>median_video</td>
<td>morph</td>
<td>morph_files</td>
<td>morph_rbf</td>
<td>morph_video</td>
</tr>
<tr>
<td>register_nonrigid</td>
<td>register_rigid</td>
<td>transition</td>
<td>transition3d</td>
<td>video2files</td>
</tr>
</tbody>
</table>

### Convenience Functions:

<table>
<thead>
<tr>
<th>add_copymarker</th>
<th>alert</th>
<th>arg</th>
<th>arg0</th>
<th>arg2img</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg2var</td>
<td>autocrop_coords</td>
<td>average_vectors</td>
<td>base642img</td>
<td>base642uint8</td>
</tr>
<tr>
<td>basename</td>
<td>bin</td>
<td>bin2dec</td>
<td>cat</td>
<td>color2name</td>
</tr>
<tr>
<td>covariance_vectors</td>
<td>da_freeze</td>
<td>date</td>
<td>dec</td>
<td>dec2str</td>
</tr>
<tr>
<td>dec2bin</td>
<td>dec2hex</td>
<td>dec2oct</td>
<td>fibonacci</td>
<td>file_mv</td>
</tr>
<tr>
<td>filename</td>
<td>filename_rand</td>
<td>filename_date</td>
<td>files</td>
<td>files2img</td>
</tr>
</tbody>
</table>
## Table:

<table>
<thead>
<tr>
<th>Trait</th>
<th>Path_type</th>
<th>Trait</th>
<th>Path_type</th>
<th>Trait</th>
<th>Path_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td><em>fitscreen</em></td>
<td><em>fontchart</em></td>
<td><em>fps</em></td>
<td><em>hex</em></td>
<td></td>
</tr>
<tr>
<td>hex2dec</td>
<td><em>hex2img</em></td>
<td><em>hex2str</em></td>
<td><em>img2base64</em></td>
<td><em>img2hex</em></td>
<td></td>
</tr>
<tr>
<td>img2str</td>
<td><em>img2text</em></td>
<td>is_mesh3d</td>
<td><em>is_change</em></td>
<td><em>is_half</em></td>
<td></td>
</tr>
<tr>
<td>is_ext</td>
<td><em>is_image_arg</em></td>
<td><em>is_pattern</em></td>
<td><em>is_varname</em></td>
<td><em>is_video_file_name</em></td>
<td></td>
</tr>
<tr>
<td>is_macos</td>
<td><em>is_windows</em></td>
<td><em>lof</em></td>
<td><em>math_lib</em></td>
<td><em>mad</em></td>
<td></td>
</tr>
<tr>
<td>max_w</td>
<td><em>max_h</em></td>
<td><em>max_d</em></td>
<td><em>max_s</em></td>
<td><em>max_wh</em></td>
<td></td>
</tr>
<tr>
<td>max_whd</td>
<td><em>max_whds</em></td>
<td><em>median_vectors</em></td>
<td><em>min_w</em></td>
<td><em>min_h</em></td>
<td></td>
</tr>
<tr>
<td>min_d</td>
<td><em>min_s</em></td>
<td><em>min_wh</em></td>
<td><em>min_whd</em></td>
<td><em>min_whds</em></td>
<td></td>
</tr>
<tr>
<td>name2color</td>
<td><em>named</em></td>
<td><em>narg</em></td>
<td>normalize_filename</td>
<td><em>oct</em></td>
<td></td>
</tr>
<tr>
<td>oct2dec</td>
<td>padint</td>
<td><em>path_cache</em></td>
<td><em>path_current</em></td>
<td><em>path_gimp</em></td>
<td></td>
</tr>
<tr>
<td>path_tmp</td>
<td>remove_copy</td>
<td>mark</td>
<td>reset</td>
<td><em>rgb</em></td>
<td><em>rgba</em></td>
</tr>
<tr>
<td>shell_cols</td>
<td><em>size_value</em></td>
<td><em>std_noise</em></td>
<td><em>str</em></td>
<td><em>strbuffer</em></td>
<td></td>
</tr>
<tr>
<td>str2hex</td>
<td>strcapitalize</td>
<td>strcontains</td>
<td>strclut</td>
<td><em>strlen</em></td>
<td></td>
</tr>
<tr>
<td>strreplace</td>
<td>strlowercase</td>
<td>struppercase</td>
<td>strvar</td>
<td><em>strcasevar</em></td>
<td></td>
</tr>
<tr>
<td>strver</td>
<td>tic</td>
<td>time</td>
<td><em>toc</em></td>
<td><em>uint82base64</em></td>
<td></td>
</tr>
</tbody>
</table>

## Other Interactive Commands:

- **demos**
  - x_2048
  - x_blobs
  - x_bouncing
  - x_color_curves
- **x_colorize**
  - x_connect4
  - x_crop
  - x_cut
  - x_fire
- **x_fireworks**
  - x_fisheye
  - x_fourier
  - x_grab_color
  - x_hanoi
- **x_histogram**
  - x_hough
  - x_jawbreaker
  - x_landscape
  - x_life
- **x_light**
  - x_mandelbrot
  - x_mask_color
  - x_metaballs3d
  - x_minesweeper
- **x_minimal_path**
  - x_morph
  - x_pacman
  - x_paint
  - x_plasma
- **x_quantize_rgb**
  - x_reflection3d
  - x_rubber3d
  - x_segment
  - x_select_color
- **x_select_function1d**
  - x_select_palette
  - x_shadebobs
  - x_spline
  - x_starfield3d
- **x_tetris**
  - x_threshold
  - x_tictactoe
  - x_tixy
  - x_warp
- **x_waves**
  - x_whirl

## Command Shortcuts:

<table>
<thead>
<tr>
<th>Shortcut name</th>
<th>Equivalent command name</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>help</td>
</tr>
<tr>
<td>m</td>
<td>command</td>
</tr>
<tr>
<td>d</td>
<td>display</td>
</tr>
<tr>
<td>d0</td>
<td>display0</td>
</tr>
<tr>
<td>da</td>
<td>display_array</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>dc</td>
<td>display_camera</td>
</tr>
<tr>
<td>dclut</td>
<td>display_clut</td>
</tr>
<tr>
<td>dfft</td>
<td>display_fft</td>
</tr>
<tr>
<td>dg</td>
<td>display_graph</td>
</tr>
<tr>
<td>dh</td>
<td>display_histogram</td>
</tr>
<tr>
<td>dq</td>
<td>display_quiver</td>
</tr>
<tr>
<td>drgba</td>
<td>display_rgba</td>
</tr>
<tr>
<td>dt</td>
<td>display_tensors</td>
</tr>
<tr>
<td>dv3d</td>
<td>display_voxels3d</td>
</tr>
<tr>
<td>dw</td>
<td>display_warp</td>
</tr>
<tr>
<td>e</td>
<td>echo</td>
</tr>
<tr>
<td>i</td>
<td>input</td>
</tr>
<tr>
<td>ib</td>
<td>input_bytes</td>
</tr>
<tr>
<td>ig</td>
<td>input_glob</td>
</tr>
<tr>
<td>it</td>
<td>input_text</td>
</tr>
<tr>
<td>o</td>
<td>output</td>
</tr>
<tr>
<td>ot</td>
<td>output_text</td>
</tr>
<tr>
<td>on</td>
<td>outputn</td>
</tr>
<tr>
<td>op</td>
<td>outputp</td>
</tr>
<tr>
<td>ow</td>
<td>outputw</td>
</tr>
<tr>
<td>ox</td>
<td>outputx</td>
</tr>
<tr>
<td>p</td>
<td>print</td>
</tr>
<tr>
<td>sh</td>
<td>shared</td>
</tr>
<tr>
<td>sp</td>
<td>sample</td>
</tr>
<tr>
<td>um</td>
<td>uncommand</td>
</tr>
<tr>
<td>up</td>
<td>update</td>
</tr>
<tr>
<td>v</td>
<td>verbose</td>
</tr>
<tr>
<td>w</td>
<td>window</td>
</tr>
<tr>
<td>k</td>
<td>keep</td>
</tr>
<tr>
<td>kn</td>
<td>keep_named</td>
</tr>
<tr>
<td>mv</td>
<td>move</td>
</tr>
<tr>
<td>nm</td>
<td>name</td>
</tr>
<tr>
<td>=&gt;</td>
<td>name</td>
</tr>
<tr>
<td>rm</td>
<td>remove</td>
</tr>
<tr>
<td>rmn</td>
<td>remove_named</td>
</tr>
<tr>
<td>rv</td>
<td>reverse</td>
</tr>
<tr>
<td>+</td>
<td>add</td>
</tr>
<tr>
<td>&amp;</td>
<td>and</td>
</tr>
<tr>
<td>&lt;&lt;</td>
<td>bsl</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>bsr</td>
</tr>
<tr>
<td>/</td>
<td>div</td>
</tr>
<tr>
<td>==</td>
<td>eq</td>
</tr>
<tr>
<td>&gt;=</td>
<td>ge</td>
</tr>
<tr>
<td>&gt;</td>
<td>gt</td>
</tr>
<tr>
<td>&lt;=</td>
<td>le</td>
</tr>
<tr>
<td>&lt;</td>
<td>lt</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>m/</td>
<td>mdiv</td>
</tr>
<tr>
<td>%</td>
<td>mod</td>
</tr>
<tr>
<td>m*</td>
<td>mmul</td>
</tr>
<tr>
<td>*</td>
<td>mul</td>
</tr>
<tr>
<td>!=</td>
<td>neq</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>^</td>
<td>pow</td>
</tr>
<tr>
<td>-</td>
<td>sub</td>
</tr>
<tr>
<td>c</td>
<td>cut</td>
</tr>
<tr>
<td>f</td>
<td>fill</td>
</tr>
<tr>
<td>ir</td>
<td>inrange</td>
</tr>
<tr>
<td>norm</td>
<td>norm2</td>
</tr>
<tr>
<td>n</td>
<td>normalize</td>
</tr>
<tr>
<td>=</td>
<td>set</td>
</tr>
<tr>
<td>ac</td>
<td>apply_channels</td>
</tr>
<tr>
<td>fc</td>
<td>fill_color</td>
</tr>
<tr>
<td>a</td>
<td>append</td>
</tr>
<tr>
<td>z</td>
<td>crop</td>
</tr>
<tr>
<td>rs</td>
<td>rescale2d</td>
</tr>
<tr>
<td>rs3d</td>
<td>rescale3d</td>
</tr>
<tr>
<td>r</td>
<td>resize</td>
</tr>
<tr>
<td>ri</td>
<td>resize_as_image</td>
</tr>
<tr>
<td>s</td>
<td>split</td>
</tr>
<tr>
<td>y</td>
<td>unroll</td>
</tr>
<tr>
<td>b</td>
<td>blur</td>
</tr>
<tr>
<td>g</td>
<td>gradient</td>
</tr>
<tr>
<td>j</td>
<td>image</td>
</tr>
<tr>
<td>ja</td>
<td>imagealpha</td>
</tr>
<tr>
<td>j3d</td>
<td>object3d</td>
</tr>
<tr>
<td>t</td>
<td>text</td>
</tr>
<tr>
<td>to</td>
<td>text_outline</td>
</tr>
<tr>
<td>+3d</td>
<td>add3d</td>
</tr>
<tr>
<td>c3d</td>
<td>center3d</td>
</tr>
<tr>
<td>col3d</td>
<td>color3d</td>
</tr>
<tr>
<td>/3d</td>
<td>div3d</td>
</tr>
<tr>
<td>db3d</td>
<td>double3d</td>
</tr>
<tr>
<td>f3d</td>
<td>focale3d</td>
</tr>
<tr>
<td>l3d</td>
<td>light3d</td>
</tr>
<tr>
<td>m3d</td>
<td>mode3d</td>
</tr>
<tr>
<td>md3d</td>
<td>moded3d</td>
</tr>
<tr>
<td>*3d</td>
<td>mul3d</td>
</tr>
<tr>
<td>n3d</td>
<td>normalize3d</td>
</tr>
<tr>
<td>o3d</td>
<td>opacity3d</td>
</tr>
<tr>
<td>p3d</td>
<td>primitives3d</td>
</tr>
<tr>
<td>rv3d</td>
<td>reverse3d</td>
</tr>
<tr>
<td>r3d</td>
<td>rotate3d</td>
</tr>
</tbody>
</table>
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:

\[
gmic \text{echo} \ "\cos(\pi/4)^2+\sin(\pi/4)^2=\{\cos(\pi/4)^2+\sin(\pi/4)^2\}"
\]

Plot a 2D function:

\[
gmic 1000,1,1,2 \ \text{fill} \ "X=3*(x-500)/500;X^2*\sin(3*X^2)+(!c?u(0,-1):\cos(X*10))" \ \text{plot}
\]

Plot a 3D elevated function in random colors:

\[
gmic 128,128,1,3,"u(0,255)" \ \text{plasma} \ 10,3 \ \text{blur} \ 4 \ \text{sharpen} \ 10000 \ n \ 0,255 \ \text{elevation} \ 64)/6;Y=(y-64)/6;100*\exp(-(X^2+Y^2)/30)*\abs(\cos(X)*\sin(Y))"'
\]

Plot the isosurface of a 3D volume:

\[
gmic \text{mode3d} 5 \ \text{moded3d} 5 \ \text{double3d} 0 \ \text{isosurface3d} \ "'x^2+y^2+\abs(z)^\abs(4*\cos(x^2+y^2+z^2))/24000"'
\]

Render a \textit{G'MIC} 3D logo:

\[
gmic 0 \ \text{text} \ G'MIC,0,0,53,1,1,1,1 \ \text{expand} \ xy,10 \ \text{blur} \ 1 \ \text{normalize} \ 0,100 \ +\text{plasma}
\]
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 ad

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[-

Launch a set of interactive demos:

$ gmic demos

abs

No arguments

Description:

Compute the pointwise absolute values of selected images.
Examples of use:

• Example #1

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

• Example #2

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

**acos**

**Built-in command**

**No arguments**

**Description:**

Compute the pointwise arccosine of selected images.

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

**Examples of use:**

• Example #1

```
image.jpg +normalize -1,1 acos[-1]
```
Example #2

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

acosh

Built-in command
No arguments

Description:
Compute the pointwise hyperbolic arccosine of selected images.

add

Built-in command

Arguments:

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

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

(equivalent to shortcut command +).
Examples of use:

• Example #1

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

• Example #2

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

• Example #3

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

• Example #4

```
image.jpg repeat 9 { +rotate[0] {>*36},1,0,50%,50% } add div 10
```
add3d

Arguments:

- tx, ty, tz  
or  
- [object3d]  
or  
- (no arg)

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

(equivalent to shortcut command +3d).

Default values:
ty=tz=0.

Examples of use:

- Example #1

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

- Example #2

```plaintext
repeat 20 { torus3d 15,2 color3d[-1] ${-rgb} mul3d[-1] 0.5,1 if $>%2
add3d
```

add_copymark

No arguments

Description:
Add copymark suffix in names of selected images.

adjust_colors

Arguments:
- \(-100\leq\text{brightness}\leq100, -100\leq\text{contrast}\leq100, -100\leq\text{gamma}\leq100, -100\leq\text{hue\_shift}\leq100\)

Description:
Perform a global adjustment of colors on selected images.

Range of correct image values are considered to be in \([\text{value\_min},\text{value\_max}]\) (e.g. \([0,255]\)). If \(\text{value\_min}==\text{value\_max}==0\), value range is estimated from min/max values of selected images. Processed images have pixel values constrained in \([\text{value\_min},\text{value\_max}]\).

Default values:
brightness=0, contrast=0, gamma=0, hue_shift=0, saturation=0, value_min=value_max=0.

Example of use:
```
image.jpg +adjust\_colors 0,30,0,0,30
```
alert

Arguments:

- title, message, label_button1, label_button2,...

Description:

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

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

Default values:

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

and

Arguments:

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

Description:

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

(equivalent to shortcut command '&').

Examples of use:

- Example #1
  
  `image.jpg and {128+64}`
• Example #2

image.jpg +mirror x and animate

animate

Arguments:

- `filter_name,"param1_start,...,paramN_start","param1_end,...,paramN_end",nb_frames={0 | 1},output_filename` or
- `delay>0,back and forth={0 | 1}`

Description:

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

Default values:

delay=30.

Example of use:

image.jpg animate flower,"0,3","20,8",9
animate3d

Arguments:

- \(nb\_frames>0, _step\_angle\_x, _step\_angle\_y, _step\_angle\_z, _zoom\_factor, 0<=_fake\_shadow\_level<=100\)

Description:

Generate 3D animation frames of rotating 3D objects.

Frames are stacked along the z-axis (volumetric image). Frame size is the same as the size of the [background] image (or 800x800 if no background specified).

Default values:

- \(step\_angle\_x=0, step\_angle\_y=5, step\_angle\_z=0, zoom\_factor=1, fake\_shadow\_level=50\) and background=(undefined).

append

Arguments:

- \([image], axis, centering\) or \(axis, centering\)

Description:

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

(equivalent to shortcut command \(a\)).

- \(axis\) can be \{\(x\mod{y\mod{z\mod{c}}}\).
- Usual centering values are \{0:left-justified | 0.5:centred | 1:right-justified\}.

Default values:

- centering=0.

Examples of use:

- Example #1
  
  `image.jpg split y,10 reverse append y`
append_tiles

Arguments:

- \( M \geq 0, N \geq 0, 0 \leq \text{centering}_x \leq 1, 0 \leq \text{centering}_y \leq 1 \)

Description:

Append MxN selected tiles as new images.

If N is set to 0, number of rows is estimated automatically.
If M is set to 0, number of columns is estimated automatically.
If M and N are both set to 0, auto-mode is used.
If M or N is set to 0, only a single image is produced.
centering_x and centering_y tells about the centering of tiles when they have different sizes.
Default values:
M=0, N=0, centering_x=centering_y=0.5.

Example of use:
```
image.jpg split xy,4 append_tiles ,
```

---

**apply_camera**

**Arguments:**
- "command",_camera_index>=0,_skip_frames>=0,_output_filename

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

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

**Default values:**
command="", camera_index=0 (default camera), skip_frames=0 and output_filename="".

---

**apply_camera3d**

**Arguments:**
- pos_x,pos_y,pos_z,target_x,target_y,target_z,up_x,up_y,up_z

**Description:**
Apply 3D camera matrix to selected 3D objects.

**Default values:**
target_x=0, target_y=0, target_z=0, up_x=0, up_y=-1 and up_z=0.
**apply_channels**

**Arguments:**

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

**Description:**

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

(equivalent to shortcut command |ac|).

Argument color_channels refers to a colorspace, and can be basically one of

{ all | rgba | [s]rgb | ryb | lrgb | ycbcr | lab | lch | hsv | hsi | hsl | cmy | cmyk | yiq }.

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

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

**Default values:**

value_action=0.

**Example of use:**

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

---

**apply_curve**

**Arguments:**

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

**Description:**

Apply curve transformation to image values.

**Default values:**

smoothness=1, x0=0, y0=100.
Example of use:

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

---

**apply_files**

**Arguments:**

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

**Description:**

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

If a display window is opened, rendered frames are displayed in it during processing.

The output filename may have extension .avi or .mp4 (saved as a video), or any other usual image file extension (saved as a sequence of images).

**Default values:**

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

---

**apply_gamma**

**Arguments:**

- `gamma>=0`

**Description:**

Apply gamma correction to selected images.

**Example of use:**

```
image.jpg +apply_gamma 2
```
apply_matrix3d

**Arguments:**

- $a_{11}, a_{12}, a_{13}, \ldots, a_{31}, a_{32}, a_{33}$

**Description:**

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

**Example of use:**

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

apply_parallel

**Arguments:**

- "command"

**Description:**

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

image.jpg +mirror x +mirror y apply_parallel "blur 3"

apply_parallel_channels

Arguments:

- "command"

Description:

Apply specified command on each of the selected images, by parallelizing it for all channel of the images independently.

(equivalent to shortcut command `apc`).

Example of use:

image.jpg apply_parallel_channels "blur 3"
apply_parallel_overlap

Arguments:

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

Description:

Apply specified command on each of the selected images, by parallelizing it on nb_threads overlapped sub-images.

(equivalent to shortcut command apo).

nb_threads must be a power of 2.

Default values:

overlap=0, nb_threads=0.

Example of use:

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

apply_scales

Arguments:
**"command",number_of_scales>0,_min_scale[%]>=0,_max_scale[%]>=0,_scale_gamma>0,_interpolation**

**Description:**
Apply specified command on different scales of selected images.

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

**Default values:**
min_scale=25%, max_scale=100% and interpolation=3.

**Example of use:**
```
image.jpg apply_scales "blur 5 sharpen 1000",4
```

---

**apply_tiles**

**Arguments:**
- "command",_tile_width[%]>0,_tile_height[%]>0,_tile_depth[%]>0,_overlap_width[%]:
  0:dirichlet | 1:neumann | 2:periodic | 3:mirror }

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

(equivalent to shortcut command `at`).
Default values:

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

Example of use:

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

apply_timeout

Arguments:

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

Description:

Apply a command with a timeout.

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

Default values:

timeout=20.

apply_video
Arguments:

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

Description:

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

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

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

Default values:

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

---

**area**

Arguments:

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

Description:

Compute area of connected components in selected images.

Default values:

is_high_connectivity=0.

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

Example of use:

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

---

**area_fg**
Arguments:

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

Description:

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

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

Default values:

`is_high_connectivity=0.`

Example of use:

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

---

`arg`

Arguments:

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

Description:

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

---

`arg0`

Arguments:

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

Description:

Return the `n`-th argument of the specified argument list (where `n` starts from 0).
arg2img

Arguments:
- argument_1,...,argument_N

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

arg2var

Arguments:
- variable_name,argument_1,...,argument_N

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

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

argmax

No arguments

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

Example of use:

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

![Example images](image.jpg sample lena,lion,square +argmax)
argmaxabs

No arguments

Description:
Compute the argmaxabs of selected images. Returns a single image with each pixel value being the index of the input image with maxabs value.

argmin

No arguments

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

Example of use:

```
image.jpg sample lena, lion, square +argmin
```
argminabs

No arguments

Description:
Compute the argminabs of selected images. Returns a single image with each pixel value being the index of the input image with minabs value.

array

Arguments:
- $M>0$, $N>0$, expand_type={0:min | 1:max | 2:all}

Description:
Create MxN array from selected images.

**Default values:**

N=M and expand_type=0.

**Example of use:**

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

---

**array3d**

**Arguments:**

- `size_x>=1, size_y>=1, size_z>=1, offset_x[%], offset_y[%], offset_z[%]`

**Description:**

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

**Default values:**

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

**Example of use:**

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

---

**array_fade**
Arguments:

- \( M > 0, N > 0, 0 \leq \text{fade}_\text{start} \leq 100, 0 \leq \text{fade}_\text{end} \leq 100, \_\text{expand}_\text{type} = \{ 0: \text{min} | 1: \text{max} | 2: \text{all} \} \)

Description:
Create \( M \times N \) array from selected images.

Default values:
\( N = M, \text{fade}_\text{start} = 60, \text{fade}_\text{end} = 90 \) and \( \text{expand}_\text{type} = 1 \).

Example of use:

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

---

array\_mirror

Arguments:

- \( N \geq 0, \_\text{dir} = \{ 0: \text{x} | 1: \text{y} | 2: \text{xy} | 3: \text{tri-xy} \}, \_\text{expand}_\text{type} = \{ 0 | 1 \} \)

Description:
Create \( 2^N \times 2^N \) array from selected images.

Default values:
dir=2 and expand_type=0.

Example of use:

```
image.jpg array_mirror 2
```
array_random

Arguments:
• Ms>0, Ns>0, Md>0, Nd>0

Description:
Create MdxNd array of tiles from selected MsxNs source arrays.

Default values:
Ns=Ms, Md=Ms and Nd=Ns.

Example of use:
image.jpg +array_random 8,8,15,10

arrow

Arguments:
• x0[], y0[], x1[], y1[], thickness[], head_length[], head_thickness[]

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

**Default values:**

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

**Example of use:**

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

---

**arrow3d**

**Arguments:**

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

**Description:**

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

**Default values:**

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

**Example of use:**

```
repeat 10 { a:=-pi/10 arrow3d 0,0,0,{cos($a)},{sin($a)},-0.5 }
+3d
```
Asin

No arguments

Description:
Compute the pointwise arcsine of selected images.

This command has a tutorial page.

Examples of use:

• Example #1

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

• Example #2

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

No arguments

Description:
Compute the pointwise hyperbolic arcsine of selected images.

at_line

Arguments:
- \(x_0\%\), \(y_0\%\), \(z_0\%\), \(x_1\%\), \(y_1\%\), \(z_1\%\)

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

Example of use:

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

at_quadrangle
Arguments:

- \( x_0\%), y_0\%), x_1\%), y_1\%), x_2\%), y_2\%), x_3\%), y_3\%), \_interpolation, \_boundary_conditions \)
or
- \( x_0\%), y_0\%), z_0\%), x_1\%), y_1\%), z_1\%), x_2\%), y_2\%), z_2\%), x_3\%), y_3\%), z_3\%), \_interpolation, \_boundary_conditions \)

Description:

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

interpolation can be \{0: nearest-neighbor \mid 1: linear \mid 2: cubic \}. boundary_conditions can be \{0: dirichlet \mid 1: neumann \mid 2: periodic \mid 3: mirror \}.

Example of use:

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

atan

No arguments

Description:

Compute the pointwise arctangent of selected images.

This command has a [tutorial page].

Examples of use:

• Example #1

```bash
image.jpg +normalize 0,8 atan[-1]
```
atan2

**Arguments:**
- \([x\_argument]\)

**Description:**
Compute the pointwise oriented arctangent of selected images.

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

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

**Example of use:**

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

No arguments

**Description:**
Compute the pointwise hyperbolic arctangent of selected images.

autocrop

**Arguments:**
- `value1,value2,...` or
- `(no arg)`

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

**Example of use:**
```
400,400,1,3 fill_color 64,128,255 ellipse 50%,50%,120,120,0,1,255
+autocrop
```
autocrop_components

Arguments:

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

Description:

Autocrop and extract connected components in selected images, according to a mask given as the last channel of each of the selected image (e.g. alpha-channel).

Default values:

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

Example of use:

```
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 ,
```
autocrop_coords

Arguments:

• value1,value2,... | auto

Description:
Return coordinates (x0,y0,z0,x1,y1,z1) of the autocrop that could be performed on the latest of the selected images.

Default values:
auto

autocrop_seq

Arguments:

• value1,value2,... | auto

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

Default values:
auto mode.

Example of use:

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

Arguments:

- `nb_colors>0`, `0<=dithering<=1`, `method={ 0:median-cut | 1:k-means }`

Description:

Index selected vector-valued images by adapted colormaps.

Default values:

dithering=0 and method=1.

Example of use:

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

average_files

Arguments:

- `filename_pattern`, `first_frame>=0`, `last_frame={ >=0 | -1=last }`, `frame_step>=1`, `output_filename`

Description:

Average specified input image files, in a streamed way.

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

**Default values:**
first_frame=0, last_frame=-1, frame_step=1 and output_filename=(undefined).

---

**average_vectors**

**No arguments**

**Description:**
Return the vector-valued average of the latest of the selected images.

---

**average_video**

**Arguments:**

- video_filename, first_frame>=0, last_frame={ >=0 | -1=last }, frame_step>=1, output_filename

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

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

**Default values:**
first_frame=0, last_frame=-1, frame_step=1 and output_filename=(undefined).

---

**axes**

**Arguments:**

- x0,x1,y0,y1, _font_height_>=0, _opacity_, _pattern_, _color1_,...

**Description:**
Draw xy-axes on selected images.

pattern is an hexadecimal number starting with 0x which can be omitted even if a color is specified.
To draw only one x-axis at row Y, set both y0 and y1 to Y.
To draw only one y-axis at column X, set both x0 and x1 to X.
Default values:
font_height=14, opacity=1, pattern=(undefined) and color1=0.

Example of use:

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

---

**axes3d**

**Arguments:**

- `size_x, size_y, size_z, font_size>0, label_x, label_y, label_z, is_origin={0:no | 1:yes }`

**Description:**

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

**Default values:**

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

**Example of use:**

```
axes3d
```

---
balance_gamma

Arguments:

• ref_color1,...

Description:

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

Default values:

ref_color1=128.

Example of use:

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

ball

Arguments:

• size>0, R, G, B, 0<=specular_light<=8, 0<=specular_size<=8, _shadow>=0

Description:

Input a 2D RGBA colored ball sprite.

Default values:

size=64, R=255, G=R, B=R, specular_light=0.8, specular_size=1 and shading=1.5.

Example of use:

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

Arguments:
• _min_freq[%, max_freq[%]

Description:
Apply bandpass filter to selected images.

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

This command has a tutorial page.

Example of use:

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

barycenter

No arguments

Description:
Compute the barycenter vector of pixel values.

Example of use:

```
256,256 ellipse 50%,50%,20%,20%,0,1,1 deform 20 +barycenter +ellipse[-2] {@0,1},5,5,0,10
```
base642img

**Arguments:**
- "base64_string"

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

The argument string must have been generated using command img2base64.

base642uint8

**Arguments:**
- "base64_string"

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

The argument string must have been generated using command uint82base64.

basename

**Arguments:**
- file_path, variable_name_for_folder

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

When specified variable_name_for_folder must starts by an underscore (global variable accessible from calling function).
bayer2rgb

Arguments:

- \_GM\_smoothness, \_RB\_smoothness1, \_RB\_smoothness2

Description:
Transform selected RGB-Bayer sampled images to color images.

Default values:
GM\_smoothness=RB\_smoothness=1 and RB\_smoothness2=0.5.

Example of use:

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

betti

No arguments

Description:
Compute Betti numbers B0,B1 and B2 from selected 3D binary shapes.

Values B0,B1 and B2 are returned in the status. When multiple images are selected, the B0,B1,B2 of each image are concatenated in the status. (see https://en.wikipedia.org/wiki/Betti_number for details about Betti numbers).

bilateral

Arguments:

- \[guide\],std\_deviation\_s[\%]>=0,std\_deviation\_r[\%]>=0,_sampling\_s>=0,_sampling\_r>=
Description:
Blur selected images by anisotropic (eventually joint/cross) bilateral filtering.

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

Example of use:

```
image.jpg repeat 5 { bilateral 10,10 }
```

---

**bin**

**Arguments:**

- `binary_int1,...`

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

---

**bin2dec**

**Arguments:**

- `binary_int1,...`

**Description:**
Convert specified binary integers into their decimal representations.

---

**blend**

**Arguments:**
Description:

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

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

opacity must be in range [0,1] (or [0%,100%]).

Default values:

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

Examples of use:

• Example #1

```latex
image.jpg +drop_shadow , rescale2d[-1] ,200 rotate[-1] 20 +blend alpha display_rgba[-2]
```

• Example #2
Example #3

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 #4

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 #5

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 #6

```plaintext
command "ex : "$"="arg repeat "$"# +blend[0,1] ${arg{$>+1}}
text_outline[-1] Mode:" "${arg{$>+1}},2,2,23,2,1,255 done"
image.jpg testimage2d {w},{h} ex
linearburn,linearlight,luminance,multiply,negation,or,overlay
```
Example #7
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 #8

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

**Arguments:**

- `smoothness[%] >= 0`

**Description:**
Blend selected images togethers using edges mode.

**Example of use:**

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

Arguments:

- [fading_shape]

Description:

Blend selected images together using specified fading shape.

Example of use:

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

blend_median
No arguments

Description:
Blend selected images together using median mode.

Example of use:

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

blend_seamless

Arguments:
- `is_mixed_mode={ 0 | 1 }, inner_fading[%]>=0, outer_fading[%]>=0`

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

Default values:
is_mixed=0, inner_fading=0 and outer_fading=100%.

blur

Arguments:
Description:
Blur selected images by a deriche or gaussian filter (recursive implementation).
(equivalent to shortcut command `b`).

boundary_conditions can be `{ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror }`. 
kernel can be `{ 0:deriche | 1:gaussian }`. 
When specified, argument axes is a sequence of `{ x | y | z | c }`. 
Specifying one axis multiple times apply also the blur multiple times.

Default values:
boundary_conditions=1 and kernel=1.

This command has a tutorial page.

Examples of use:
• Example #1

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

• Example #2

```plaintext
image.jpg +blur y,10%
```
blur_angular

Arguments:

- amplitude[%], center_x[%], center_y[%]

Description:
Apply angular blur on selected images.

Default values:
center_x=center_y=50%.

This command has a tutorial page.

Example of use:

```
image.jpg blur_angular 2%
```

blur_bloom

Arguments:

- amplitude>=0, ratio>=0, nb_iter>=0, blend_operator={ + | max | min }, kernel={ 0:deriche | 1:gaussian | 2:box | 3:triangle | 4:quadratic }, normalize_scales={ 0 | 1 }, axes
**Description:**
Apply a bloom filter that blend multiple blur filters of different radii, resulting in a larger but sharper glare than a simple blur.
When specified, argument axes is a sequence of \{ x | y | z | c \}.
Specifying one axis multiple times apply also the blur multiple times.

**Default values:**
- amplitude=1, ratio=2, nb_iter=5, blend_operator=+, kernel=1, normalize_scales=0 and axes=(all)

**Example of use:**
```
image.jpg blur_bloom ,
```

---

**blur_linear**

**Arguments:**
- \_amplitude2[%], \_angle, \_boundary_conditions={ 0:dirichlet | 1:neumann }

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

**Default values:**
- amplitude2=0, angle=0 and boundary_conditions=1.

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

**Example of use:**
```
image.jpg blur_linear 10,0,45
```
blur_radial

**Arguments:**

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

**Description:**

Apply radial blur on selected images.

**Default values:**

center_x=center_y=50%.

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

**Example of use:**

```
image.jpg blur_radial 2%
```

---

blur_selective

**Arguments:**

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

**Description:**
Blur selected images using selective gaussian scales.

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

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

**Example of use:**

```plaintext
image.jpg noise 20 cut 0,255 +local[-1] repeat 4 { blur_selective , }
done
```

---

**boundingbox3d**

**No arguments**

**Description:**

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

**Example of use:**

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

---

**box3d**
Arguments:

- \(size_x, size_y, size_z\)

Description:

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

Default values:

\(size_x=1\) and \(size_z=size_y=size_x\).

Example of use:

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

---

**boxfilter**

Built-in command

Arguments:

- \(size\geq0\%\), \(_order\), \(_boundary_conditions\), \(_nb_iter\geq0\) or
- \(axes, size\geq0\%\), \(_order\), \(_boundary_conditions\), \(_nb_iter\geq0\)

Description:

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

order can be \(\{0: smooth \mid 1: \text{1st-derivative} \mid 2: \text{2nd-derivative}\}\).

boundary_conditions can be \(\{0: \text{dirichlet} \mid 1: \text{neumann} \mid 2: \text{periodic} \mid 3: \text{mirror}\}\).

When specified, argument axes is a sequence of \(\{x \mid y \mid z \mid c\}\).

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

Default values:

order=0, \_boundary_conditions=1 and \_nb_iter=1.

Examples of use:

- Example #1
  
  ```
  image.jpg +boxfilter 5%
  ```
Example #2

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

boxfitting

Arguments:

- `min_box_size>=1, max_box_size>=0, initial_density>=0, min_spacing>0`

Description:

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

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

Default values:

`min_box_size=1, max_box_size=0, initial_density=0.25 and min_spacing=1`.

Example of use:

```
image.jpg boxfitting
```
**break**

*Built-in command*

**No arguments**

**Description:**

Break current do...while, for...done, foreach...done, local...done or repeat...done block.

**Example of use:**

```bash
image.jpg repeat 10 blur 1 if 1==1 break fi deform 10 done
```

---

**brushify**

**Arguments:**

- `[brush], _brush_nb_sizes>=1, 0<=_brush_min_size_factor<=1, _brush_nb_orientations>=1, _brush_light_type`:  

**Description:**

Apply specified brush to create painterly versions of specified images.

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

**Default values:**

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

Example of use:

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

bsl

**Arguments:**

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

**Description:**

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

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

**Example of use:**

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

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

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

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

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

---

**bump2normal**

**No arguments**

**Description:**
Convert selected bumpmaps to normalmaps.

**Example of use:**
```
300,300 circle 50%,50%,128,1,1 blur 5% bump2normal
```
camera

Arguments:

- camera_index\[\geq 0\], nb_frames\[> 0\], skip_frames\[\geq 0\], capture_width\[\geq 0\], capture_height\[\geq 0\]

Description:

Insert one or several frames from specified camera.

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

Default values:

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

---

canny

Arguments:

- sigma\[\geq 0\], low_threshold\[\geq 0\], high_threshold\[\geq 0\]

Description:

Locate image edges using Canny edge detector.

Default values:

sigma=1, low_threshold=0.05, high_threshold=0.15.

Example of use:

image.jpg canny 1
cartoon

Arguments:

- smoothness, sharpening, threshold>=0, thickness>=0, color>=0, quantization>0

Description:

Apply cartoon effect on selected images.

Default values:

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

Example of use:

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

---

cast

Arguments:

- datatype_source, datatype_target

Description:

Cast datatype of image buffer from specified source type to specified target type.
datatype_source and datatype_target can be \{ uint8 \mid int8 \mid uint16 \mid int16 \mid uint32 \mid int32 \mid uint64 \mid int64 \mid float32 \mid float64 \}.

---

cat

**Arguments:**

- `filename, _display_line_numbers={ 0 | 1 }, _line_selection`

**Description:**

Print specified line selection of given filename on stdout.

**Default values:**

display_line_numbers=1 and line_selection=^.

---

center3d

**No arguments**

**Description:**

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

(equivalent to shortcut command `c3d`).

**Example of use:**

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

---

chainring3d

**Arguments:**
• \( \text{nb_links} \geq 3, x \_\text{scale} > 0, y \_\text{scale} > 0, z \_\text{scale} > 0 \)

**Description:**

Input 3D chain ring with specified geometry.

\( \text{nb_links} \) should be preferably even.

**Default values:**

\( \text{nb_links}=16, x \_\text{scale}=0.5, y \_\text{scale}=1 \) and \( z \_\text{scale}=1 \).

**Example of use:**

```python
chainring3d
```

---

**channels**

**Arguments:**

• \( c0\% \), \( c1\% \), \( \text{boundary\_conditions} \)

**Description:**

Keep only specified channels of selected images.

boundary\_conditions can be \{ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror \}.

**Default values:**

\( c1=c0 \) and boundary\_conditions=0.

**Examples of use:**

• Example #1

```python
image\_jpg \ channels 1
```
• Example #2

image.jpg luminance channels 0,2

---

**check**

**Arguments:**

- condition

**Description:**

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

---

**check3d**

**Arguments:**

- is_full_check={ 0 | 1 }

**Description:**

Check validity of selected 3D vector objects, and display an error message if one of the selected images is not a valid 3D vector object.

Full 3D object check is slower but more precise.

**Default values:**
chessboard

Arguments:

- size1 > 0, size2 > 0, offset1, offset2, angle, opacity, color1, ..., color2, ...

Description:

Draw chessboard on selected images.

Default values:

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

Example of use:

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

![Chessboard example](image.jpg)

cie1931

No arguments

Description:

Draw CIE-1931 chromaticity diagram on selected images.

Example of use:

```
500,400,1,3 cie1931
```
circle

Arguments:

• x[%], y[%], R[%], _opacity, _pattern, _color1,...

Description:

Draw specified colored circle on selected images.

A radius of 100% stands for sqrt(width^2+height^2).
pattern is an hexadecimal number starting with 0x which can be omitted even if a color is specified. If a pattern is specified, the circle is drawn outlined instead of filled.

Default values:

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

Example of use:

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

circle3d

Arguments:
• \(x_0, y_0, z_0, \text{radius} \geq 0\)

**Description:**
Input 3D circle at specified coordinates.

**Default values:**
\(x_0 = y_0 = z_0 = 0\) and \(\text{radius} = 1\).

**Example of use:**

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

---

circles3d

**Arguments:**

• \(\text{radius} \geq 0, \text{is_outlined} = \{ 0 \mid 1 \}\)

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

**Default values:**
\(\text{radius} = 1\) and \(\text{is_outlined} = 1\).

**Example of use:**

```plaintext
image.jpg luminance rescale2d ,40 threshold 50% * 255 pointcloud3d
color3d[-1] 255,255,255 circles3d 0.7
```
**close_binary**

**Arguments:**

- $0 \leq \_\text{endpoint\_rate} \leq 100$, $\_\text{endpoint\_connectivity} \geq 0$, $\_\text{spline\_distmax} \geq 0$, $\_\text{segment\_distmax} \geq 0$, $0 \mid 1$

**Description:**

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

**Default values:**

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

**closing**

**Arguments:**

- $\text{size} \geq 0$ or $\text{size}_x \geq 0, \text{size}_y \geq 0, \text{size}_z \geq 0$ or $[\text{kernel}], \_\text{boundary\_conditions}, \_\text{is\_real} = \{ 0:\text{binary\_mode} \mid 1:\text{real\_mode} \}$

**Description:**

Apply morphological closing to selected images.

boundary_conditions can be $\{ 0:\text{dirichlet} \mid 1:\text{neumann} \mid 2:\text{periodic} \mid 3:\text{mirror} \}$.

**Default values:**

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

**Example of use:**

```plaintext
image.jpg +closing 10
```
closing_circ

Arguments:
• \_size\geq0, _is_real={ 0 \mid 1 }

Description:
Apply circular dilation of selected images by specified size.

Default values:
boundary_conditions=1 and is_real=0.

Example of use:

```
image.jpg +closing_circ 7
```

clut

Arguments:
• "clut_name", _resolution>0, _cut_and_round={ 0:no \mid 1:yes }

Description:
Insert one of the 1149 pre-defined CLUTs at the end of the image list.
clut_name can be { 12_years_a_slave | 1917 | 2-strip-process | 60s | 60s_faded |
60s_faded_alt | 7drk_21 | action_magenta_01 | action_red_01 | ad_astra |
adventure_1453 | agfa_apx_100 | agfa_apx_25 | agfa_precisa_100 |
agfa_ultra_color_100 | agfa_vista_200 | agressive_highlightes_recovery_5 |
aladdin | alberto_street | alien_green | ampio | amstragram | amstragram+ |
alogfis_vf_1 | analogfx_ann0_1870_color | analogfx_old_style_i |
alogfis_vf_2 | analogfx_old_style_ii | analogfx_old_style_iii | analogfx_sepia_color |
alogfis_vf_3 | analogfx_soft_sepia_i | analogfx_soft_sepia_ii | anime | ant-man |
apocalypse_this_very_moment | aqua | aqua_and_orange_dark | aquaman |
arabica_12 | asistas | atomic_pink | atusa | autumn | autumn_leaves | ava_614 |
avanche | avengers_endgame | azrael_93 | baby_driver | bad_boys_for_life |
basuco | bboyz_2 | bc_darkum | beach_aqua_orange | beach_faded_analog |
beati | beauty_and_the_beast | berlin_sky | bisogno | black_and_white |
black_panther | black_star | black_white_01 | black_white_02 | black_white_03 |
black_white_04 | black_white_05 | black_white_06 | blade_runner |
bleach_bypass | bleachbypass_1 | bleachbypass_2 | bleachbypass_3 |
bleachbypass_4 | bleachbypass_green | bleachbypass_yellow_01 |
blue_cold_fade | blue_dark | blue_house | blue_ice | blue_love_39 | blue_mono |
blue_shadows_01 | bluearchitecture | bluehour | blues | bob_ford |
bohemian_rhapsody | bombshell | bourbon_64 | boyado | bright_green_01 |
bright_teal_orange | bright_warm | brightgreen | brown_mobster | brownbm |
brownish | bw_1 | bw_10 | bw_2 | bw_3 | bw_4 | bw_5 | bw_6 | bw_7 | bw_8 |
bw_9 | bw_but_yellow | byers_11 | calidum | candlelight | captain_marvel |
caribe | chemical_168 | chrome_01 | cineblue | cinebm_4k | cinema | cinema_2 |
cinema_3 | cinema_4 | cinema_5 | cinema_noir | cinematic-1 | cinematic-10 |
cinematic-2 | cinematic-3 | cinematic-4 | cinematic-5 | cinematic-6 |
cinematic-7 | cinematic-8 | cinematic-9 | cinematic_01 | cinematic_02 |
cinematic_03 | cinematic_04 | cinematic_05 | cinematic_06 | cinematic_07 |
cinematic_for_flog | cinematic_forest | cinematic_lady_bird |
cinematic_mexico | city | city_7 | city_dust | city_of_god | classic_films_01 |
classic_films_02 | classic_films_03 | classic_films_04 | classic_films_05 |
classic_teal_and_orange | clayton_33 | clear | clear_teal_fade | clouseau_54 |
cobi_3 | coffee_44 | cold_clear_blue | cold_clear_blue_1 | cold_ice |
cold_simplicity_2 | coldchrome | color_rich | colore | colorful_0209 |
colornegative | conflict_01 | contrail_35 |
contrast_with_highlights_protection | contrasty_afternoon | contrasty_green | convold |
cosa | creed_2 | crispaumtn | crisppromance | crispswarm |
crispwinter | cross_process_cp_130 | cross_process_cp_14 |
cross_process_cp_15 | cross_process_cp_16 | cross_process_cp_18 |
cross_process_cp_2 | cross_process_cp_4 | cross_process_cp_6 | crushin |
cubicle_99 | culor | d_o_1 | dark_blues_in_sunlight | dark_green_02 |
dark_green_1 | dark_man_x | dark_orange_teal | dark_place_01 | darkandsomber |
darkness | date_39 | day_4nite | day_for_night | day_to_night_kings_blue |
deep | deep_blue | deep_dark | deep_high_contrast | deep_teal_fade |
dep Warm_fade | deepskintones_2 | deepskintones_3 | delicatessen |
denoiser_simple_40 | desert_gold_37 | dimension | dimmer | directions_23 |
django_25 | doctor_strange | domino_145 | dream_1 | dream_85 |
drop_green_tint_14 | dropblues | dunkirk | duotone_blue_red |
earth_tone_boost | eda_0_2 | edgeyember | elegance_38 | enchanted | ensaya |
eterna_for_flog | expired_69 | expired_fade | expired_polaroid | extreme |
fade | fade_to_green | faded | faded_47 | faded_alt | faded_analog |
faded_extreme | faded_green | faded_pink-ish | faded_print | faded_retro_01 |
faded_retro_02 | faded_vivid | fadedlook | fallcolors | falua | farkling |
fatos | faux_infrared | faux_infrared_bw_1 | faux_infrared_color_p_2 |
| retro_summer_3 | retro_yellow_01 | rocketman | rollei_ir_400 |
| rollei_ortho_25 | rollei_retro_100_tonal | rollei_retro_80s | rotate_muted |
| rotate_vibrant | rotated | rotated_crush | satid | saturated_blue |
| saving_private_damon | scala | science_fiction | scribble | sea | seges |
| selor | sensum | separation | serenity | seringe_4 | serpent |
| seventies_magazine | sevsuz | shade_kings_ink | shadow_king_39 | shine |
| sicario | sino | skin_tones | slog_to_rec709_basic | slog_to_rec709_contrasty |
| smart_contrast | smokey | smooth_clear | smooth_cromeish | smooth_fade |
| smooth_green_orange | smooth_sailing | smooth_teal_orange | soft_fade |
| softblackandwhite | softwarming | solarized_color | solarized_color_2 | soldi |
| spider-man_far_from_home | spotlight | springmorning | sprocket_231 |
| spy_29 | standard | star_wars_the_rise_of_skywalker | strano | street |
| stringa | studio_skin_tone_shaper | subtle_blue | subtle_green |
| subtle_yellow | sully | summer | summer_alt | sunlight_love_11 |
| sunny | sunny_alt | sunny_rich | sunny_warm | sunset |
| sunset_intensity_violet_blue | sunset_violet_mood | super_warm |
| super_warm_rich | sutro_fx | sweet_bubblegum | sweet_gelatto | taşimırır_1 |
| taiga | tarraco | teal-orange_for_flog | teal_fade | teal_moonlight |
| tealmagentagold | tealorange_1 | tealorange_2 | tealorange_3 |
| technicalfx_backlight_filter | teigen_28 | tenet | tensiongreen_1 |
| tensiongreen_2 | tensiongreen_3 | tensiongreen_4 | terra_4 | the_dark_knight |
| the_darkest_hour | the_gentlemens | the_grand_budapest_hotel |
| the_hurt_locker | the_ishman | the_lighthouse | the_lobster |
| the_calamity | the_matrices | the_revenant | the_shape_of_water | the_social_network |
| the_two_popes | the_way_back | thor_ragnarok | thriller_2 |
| toastedgarden | top_gun_maverick | trent_18 | true_colors_8 |
| turkjest_42 | tutto | tweed_71 | ultra_water |
| uncut_gems | undeniable | undeniable_2 |
| underwater | unknown | uplow | urban_01 |
| urban_02 | urban_03 | urban_04 | urban_05 |
| urban_cowboy | uzbek_bukhara | uzbek_marriage | uzbek_samarande |
| valize | valsky | velvetia | venom |
| very_warm_greenish | vfb_21 | vibrant |
| vibrant_alien | vibrant_contrast | vibrant_cromeish | victory |
| vintage_01 | vintage_02 | vintage_03 | vintage_04 |
| vintage_05 | vintage_163 | vintage_alt | vintage_brighter |
| vintage_chrome | vintage_mob | vintage_warmoth_1 | violet_taste |
| vireo_37 | vita | vivid | vubes |
| war_for_the_planet_of_the_apes | warm | warm_dark_contrasty | warm_fade |
| warm_fade_1 | warm_highlight | warm_neutral | warm_sunset_red |
| warm_teal | warm_vintage | warm_yellow | wavefire |
| waves | well_see | western_6 |
| westernlut_2 | westernlut_2_13 | whiter_whites |
| winterlighthouse | wipe | wolf_of_wall_street | wonder_woman |
| wooden_gold_20 | x-men_dark_phoenix | yangabuz_8 | yellow_55b |
| yellowstone | you_can_do_it | zed_32 | zeke_39 |
| zilverfx_bw_solarization | zilverfx_infrared | zilverfx_vintage_bw | zombieland_double_tap |

**Default values:**

resolution=33 and cut_and_round=1.

**Example of use:**

```plaintext
clut summer clut alien_green,17 clut orange_dark4,48
```
clut2hald

No arguments

Description:
Convert selected 3D CLUTs to 2D HaldCLUTs.

Example of use:

| clut summer +clut2hald |

-----------

cmy2rgb

No arguments

Description:
Convert color representation of selected images from CMY to RGB.

cmyk2rgb

No arguments

Description:
Convert color representation of selected images from CMYK to RGB.

**color2name**

**Arguments:**
- R, G, B

**Description:**
Return the name (as a string, in English) that most matches the specified color.

**color3d**

**Arguments:**
- \( R, G, B, \text{opacity} \)

**Description:**
Set color (and optionally opacity) of selected 3D objects.

*(equivalent to shortcut command* \( \text{col3d} \)).

**Default values:**
B=\( G=R \) and \( \text{opacity}=(\text{undefined}) \).

**Example of use:**

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

**color_ellipses**

**Arguments:**
• \_count>0, \_radius>=0, \_opacity>=0

**Description:**
Add random color ellipses to selected images.

**Default values:**
count=400, radius=5 and opacity=0.1.

**Example of use:**
```
image.jpg +color_ellipses ,,0.15
```

---

**colorblind**

**Arguments:**
• type={ 0:protanopia | 1:protanomaly | 2:deuteranopia | 3:deuteranomaly | 4:tritanopia | 5:tritanomaly | 6:achromatopsia | 7:achromatamaly }

**Description:**
Simulate color blindness vision.

Simulation method of Vienot, Brettel & Mollon 1999, "Digital video colourmaps for checking the legibility of displays by dichromats". The dichromacy matrices of the paper were adapted to sRGB (RGB->XYZ). Anomalous trichromacy simulated via linear interpolation with the identity and a factor of 0.6.

**Example of use:**
```
image.jpg +colorblind 0
```
**colorcube3d**

**Arguments:**
- \(_\text{is\_wireframe} = \{0 \mid 1\}\)

**Description:**
Input 3D color cube.

**Default values:**
is\_wireframe=0.

**Example of use:**
```
colorcube3d mode3d 2 +primitives3d 1
```

**colorize3d**

**Arguments:**
- \(_\text{color\_function}, _\text{passed\_images\_for\_color\_function}\)

**Description:**
Colorize primitives of selected 3D objects, according to a specified function.
- \(_\text{color\_function}\) returns a G, GA, RGB or RGBA vector that can depend on variables x, y and z, which
are defined as the barycenter coordinates for each primitive.

- passed_images_for_color_function can be specified as a selection (e.g. [0,2]) of images that will be inserted at the end of the image list while modifying 3D objects, so that the color_function can have access to their content.

**Default values:**

color_function=[x,y,z] and passed_images_for_color_function=.

**Example of use:**

torus3d 100,40,640,100 c3d n3d mul3d 256 +3d 128,128,128 sample colorful,257 colorize3d[0] “I(#-1,x,y,0)”,[1]

colormap

**Arguments:**

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

**Description:**

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

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

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

**Default values:**

method=1 and sort_vectors=1.

This command has a [tutorial page].

**Example of use:**

image.jpg +colormap[0] 4 +colormap[0] 8 +colormap[0] 16
columns

Arguments:

- $x_0[\%], x_1[\%], \text{boundary\_conditions}$

Description:

Keep only specified columns of selected images.

boundary_conditions can be \{ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror \}.

Default values:

$x_1=x_0$ and boundary_conditions=0.

Example of use:

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

command

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

**Description:**

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

*(equivalent to shortcut command \m*).

Imported commands are available directly after the command invocation. Specified filename is not allowed to contain colons :.

**Default values:**

add_debug_info=1 (except for a "string" argument, in which case add_debug_info=0).

**Example of use:**

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

---

**complex2polar**

**No arguments**

**Description:**

Compute complex to polar transforms of selected images.

**Example of use:**

```plaintext
image.jpg +fft complex2polar[-2,-1] log[-2] shift[-2] 50%,50%,0,0,2
```
compose_channels

No arguments

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

Default values:
1=+

This command has a tutorial page.

Example of use:

image.jpg +compose_channels and

compose_freq

No arguments

Description:
Compose selected low and high frequency parts into new images.
Example of use:

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

compress_clut

No arguments

Description:
Compress selected color LUTs as sequences of colored keypoints.

compress_huffman

Arguments:
- [huffman_tree], max_leaf_value

Description:
Compress selected images with Huffman coding.

See also:
- decompress_huffman, huffman_tree

compress_rle

Arguments:
- is_binary_data={ 0 | 1 }, maximum_sequence_length>=0

Description:
Compress selected images as 2xN data matrices, using RLE algorithm.
Set maximum_sequence_length=0 to disable maximum length constraint.
Default values:
is_binary_data=0 and maximum_sequence_length=0.

Example of use:

```
image.jpg rescale2d ,100 quantize 4 round +compress_rle , +decompress_rle[-1]
```

**compress_to_keypoints**

**Arguments:**

- `method, max_keypoints>=0, err_avg[\%]>=0, err_max[\%]>=0, err_command`

**Description:**

Compress each of the selected images into a set of keypoints that can be further decompressed using command `decompress_from_keypoints`.

**Beware:** This type of compression is effective only for images with very smooth content.

- `method` can be `{ 0:PDE | 1:RBF }`. Add 2 to `method` to skip the point removal step.
- `max_keypoints` is the maximal number of keypoints generated by the compression method. If `max_keypoints<0`, the removal step is not done when number of maximal keypoints has been reached. `max_keypoints=0` means no limits.
- `err_avg` is the desired average compression error.
- `err_max` is the desired pointwise max compression error.
- `err_command` is the code of a command that inputs the two images [reference] and [compressed] and compute a single error map as a last image.

Defaults values: `method=3`, `max_keypoints=0`, `err_avg=1\%`, `err_max=5\%` and `err_command=-. [0]`
**cone3d**

**Arguments:**
- \_radius, \_height, \_nb_subdivisions>0

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

**Default values:**
radius=1, height=1 and nb_subdivisions=24.

**Example of use:**
```
cone3d 10,40 +primitives3d 1 color3d[-2] ${-rgb}
```

---

**continue**

**Built-in command**

**No arguments**

**Description:**
Go to end of current do...while, for...done, foreach...done, local...done or repeat...done block.

**Example of use:**
```
image.jpg repeat 10 blur 1 if 1==1 continue fi deform 10 done
```
convolve

**Arguments:**

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

**Description:**

Convolve selected images by specified mask.

- `boundary_conditions` can be `{ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror }`.
- `channel_mode` can be `{ 0:all | 1:one-for-one | 2:partial sum | 3:full sum }`.
- `interpolation_type` can be `{ 0:nearest-neighbor | 1:linear }`.

**Default values:**

`boundary_conditions=1, _is_normalized=0, _channel_mode=1, _xcenter=_ycenter=_zcenter=(undefined), _xstart=_ystart=_zstart=0, _xend=_yend=_zend=(max-coordinates), _xstride=_ystride=_zstride=1, _xdilation=_ydilation=_zdilation=1` and `interpolation_type=0`.

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

**Examples of use:**

- **Example #1**

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

- **Example #2**

  ```
  ```
convolve_fft

Arguments:
- [mask], boundary_conditions

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

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

Example of use:
image.jpg 100%,100% gaussian[-1] 20,1,45 +convolve_fft[0] [1]
correlate

Arguments:

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

Description:

Correlate selected images by specified mask.

boundary_conditions can be `{ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror }.
channel_mode can be `{ 0:all | 1:one-for-one | 2:partial sum | 3:full sum }.
interpolation_type can be `{ 0:nearest-neighbor | 1:linear }.

Default values:

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

Examples of use:

• Example #1

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

• Example #2
**COS**

Built-in command

No arguments

**Description:**

Compute the pointwise cosine of selected images.

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

**Examples of use:**

- **Example #1**

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

  ![Example #1](image.jpg +crop 40%,40%,60%,60% +correlate[0] [-1],0,1)
**Example #2**

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

---

cosh

**Built-in command**

**No arguments**

**Description:**

Compute the pointwise hyperbolic cosine of selected images.

**Examples of use:**

**• Example #1**

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

**• Example #2**

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

Arguments:

- \_count_until={ 0 or none | >0 or max number of counted colors }

Description:

Count number of distinct colors in selected images until it reaches the specified max number of counted colors.

Set count_until to 0 to disable limit on counted colors.

This command returns the number of distinct colors for each image (separated by commas).

covariance_vectors

Arguments:

- \_avg_outvarname

Description:

Return the covariance matrix of the vector-valued colors in the latest of the selected images (for arbitrary number of channels).

Parameter avg_outvarname is used as a variable name that takes the value of the average vector-value.

cracks

Arguments:

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

Description:
Draw random cracks on selected images with specified color.

**Default values:**

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

**Example of use:**

```
image.jpg +cracks ,
```

---

crop

**Arguments:**

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

**Description:**

Crop selected images with specified region coordinates.

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

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

**Default values:**

boundary_conditions=0.

**Examples of use:**

- Example #1

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

Arguments:

• [mask]

Description:

Compute cross-correlation of selected images with specified mask.

Example of use:

```plaintext
image.jpg +shift -30,-20 +cross_correlation[0] [1]
```
cubes3d

Arguments:
• \_size\geq0

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

Default values:
size=1.

Example of use:

```
image.jpg luminance rescale2d ,40 threshold 50% * 255 pointcloud3d
color3d[-1] 255,255,255 cubes3d 1
```

cubism

Arguments:
• \_density\geq0,0\leq\_thickness\leq50,\_max\_angle,\_opacity,\_smoothness\geq0

Description:
Apply cubism effect on selected images.
Default values:
density=50, thickness=10, max_angle=75, opacity=0.7 and smoothness=0.

Example of use:

```
image.jpg cubism ,
```

---

**cumulate**

**Built-in command**

**Arguments:**

- `\{ x | y | z | c \}...\{ x | y | z | c \}` or
- `(no arg)`

**Description:**

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

**Example of use:**

```
image.jpg +histogram 256 +cumulate[-1] display_graph[-2,-1] 400,300,3
```
**cup3d**

**Arguments:**
- \( \text{resolution} > 0 \)

**Description:**
Input 3D cup object.

**Default values:**
resolution=128.

**Example of use:**
```
cup3d ,
```

---

**cursor**

**Arguments:**
- \( \text{mode} = \{ \text{0:hide} | \text{1:show} \} \)

**Description:**
Show or hide mouse cursor for selected instant display windows.
Default values:

mode=1.

---

**curvature**

No arguments

**Description:**
Compute isophote curvatures on selected images.

**Example of use:**

```
image.jpg blur 10 curvature
```

---

**curve**

**Arguments:**
- `[xy_coordinates],_thickness>0,_tilt,_tilt_strength[%],_is_closed={ 0:no | 1:yes },_opacity,_color1,...`

**Description:**
Draw specified parameterized curve on selected images.

Arguments are:
- `[xy_coordinates]` is the set of XY-coordinates of the curve, specified as a 2-channels image.
- `thickness` is the thickness of the drawing, specified in pixels.
- `tilt` is an angle, specified in degrees.
- `tilt_strength` must be a float value in [0,1] (or in [0,100] if specified as a percentage).
- `is_closed` is a boolean which tells if the curve is closed or not.

**Default values:**
厚度=0，倾斜角=45°

例用示例:

```plaintext
image.jpg srand 3 16,1,1,4,u s. c,2 rbf[-2,-1] 1000,0,1 n[-2] 10, {w#0-10} n[-1] 10, {h#0-10} a[-2,-1] c curve[-2] [-1],6,0,0,0,1,0,128,0
```

**curve3d**

**Arguments:**

- "x(t)", "y(t)", "z(t)", "r(t)", _resolution>1, _tmin, _tmax, _nb_sides>=0, _is_closed={0 | 1}

**Description:**

输入3D曲线，具有指定参数化。

如果r(t)==0或nb_sides<3，生成的3D对象仅由段组成。

**Default values:**

x(t)=cos(2*pi*t), y(t)=sin(2*pi*t), z(t)=t, r(t)=0.025, resolution=128, tmin=0, tmax=1, nb_sides=16和is_closed_curve=0。

**Example of use:**

```plaintext
curve3d ,
```
**cut**

**Arguments:**

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

**Description:**

Cut values of selected images in specified range.

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

**Examples of use:**

- **Example #1**
  
  ```plaintext
  image.jpg +add 30% cut[-1] 0,255
  ```

- **Example #2**
  
  ```plaintext
  image.jpg +cut 25%,75%
  ```

---

**cylinder3d**

**Arguments:**
Arguments:
- \( _{\text{radius, height, nb_subdivisions}>0} \)

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

Default values:
radius=1, height=1 and nb_subdivisions=24.

Example of use:
```
cylinder3d 10,40 +primitives3d 1 color3d[-2] ${-rgb}
```

\[ \text{[3D cylinder] (50 vert, 72 prim)} \]
\[ \text{[3D cylinder]ct (50 vert, 70 prim)} \]

---

**do_freeze**

No arguments

Description:
Convert each of the selected dynamic arrays into a 1-column image whose height is the number of array elements.

---

**date**

No arguments

Description:
Return current date as a string YYYY/MM/DD.

---

**dct**

Arguments:
- \( \{ x | y | z \} \ldots \{ x | y | z \} \) or
- (no arg)
**Description:**

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

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

**Default values:**

(no arg)

**See also:**

idct.

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

**Example of use:**

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

---

**deblur**

**Arguments:**

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

**Description:**

Deblur image using a regularized Jansson-Van Cittert algorithm.
Default values:
nb_iter=10, dt=20, regul=0.7 and regul_type=1.

Example of use:

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

---

deblur_goldmeinel

**Arguments:**

- sigma>=0, nb_iter>=0, acceleration>=0, kernel_type={ 0:deriche | 1:gaussian }.

**Description:**

Deblur selected images using Gold-Meinel algorithm

**Default values:**

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

**Example of use:**

```
image.jpg +blur 1 +deblur_goldmeinel[-1] 1
```
deblur_richardsonlucy

Arguments:

- \( \sigma \geq 0 \), \( \text{nb\_iter} \geq 0 \), \_kernel\_type={ 0:deriche | 1:gaussian }.

Description:

Deblur selected images using Richardson-Lucy algorithm.

Default values:

\( \text{nb\_iter}=50 \) and \_kernel\_type=1.

Example of use:

```
image.jpg +blur 1 +deblur_richardsonlucy[-1] 1
```
debug

No arguments

**Description:**
Activate debug mode.

When activated, the G'MIC interpreter becomes very verbose and outputs additional log messages about its internal state on the standard output (stdout). This option is useful for developers or to report possible bugs of the interpreter.

---

dec

**Arguments:**

- `decimal_int1,...`

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

---

dec2bin

**Arguments:**

- `decimal_int1,...`

**Description:**
Convert specified decimal integers into their binary representations.

---

dec2hex

**Arguments:**

- `decimal_int1,...`

**Description:**
Convert specified decimal integers into their hexadecimal representations.

---

dec2oct

**Arguments:**

- `decimal_int1,...`

**Description:**
Convert specified decimal integers into their octal representations.
• *decimal_int1,...*

**Description:**
Convert specified decimal integers into their octal representations.

---

**dec2str**

**Arguments:**
• *decimal_int1,...*

**Description:**
Convert specified decimal integers into its string representation.

---

**decompress_clut**

**Arguments:**
• *width>0,_height>0,_depth>0*

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

**Default values:**
width=height=depth=33 and reconstruction_colors=0.

---

**decompress_from_keypoints**

**Arguments:**
• *width>0,_height>0,_depth>0* or
• *(no arg)*

**Description:**
Decompress selected sets of keypoints as images (opt. of specified size).

A set of keypoints is defined as a vector-valued image, such that:
• The first pixel is a vector which encodes the \([ Width,Height,Depth ]\) of the decompressed image.
• The second pixel is a vector which encodes \([ Min,Max,Use_RBF ]\), where Min and Max defines the value range of the decompressed image, and Use_RBF tells is the decompression scheme must use RBFs (Use_RBF=1) or Multiscale Diffusion PDE's (Use_RBF=0).
• The remaining pixels define the keypoint coordinates and values, as:
[ x_k, y_k, z_k, v_1_k, ..., v_N_k ] for a 3D target image of N-valued vectors.

[ x_k, y_k, v_1_k, ..., v_N_k ] for a 2D target image of N-valued vectors.

[ x_k, v_1_k, ..., v_N_k ] for a 1D target image of N-valued vectors.

where the coordinates x_k, y_k and z_k are defined respectively in ranges [0,Width-1], [0,Height-1] and [0,Depth-1].

If the width, height and depth arguments are provided, they define the size of the decompressed image; overriding then the original image size [Width,Height,Depth] defined in the keypoints header.

decompress_huffman

**Arguments:**

- [huffman_tree]

**Description:**

Decompress selected images with Huffman decoding.

**See also:**

compress_huffman, huffman_tree.

**Example of use:**

```plaintext
image.jpg huffman_tree compress_huffman... +decompress_huffman...
```

decompress_rle

**No arguments**

**Description:**

Decompress selected data vectors, using RLE algorithm.

decovolve_fft
Arguments:
  • [kernel], _regularization>=0

Description:
Deconvolve selected images by specified mask in the fourier space.

Default values:
regularization>=0.

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

---

deform

Arguments:
  • amplitude>=0, _interpolation

Description:
Apply random smooth deformation on selected images.

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

Default values:
amplitude=10.
Example of use:

image.jpg +deform[0] 10 +deform[0] 20

deg2rad
No arguments
Description:
Convert pointwise angle values of selected images, from degrees to radians (apply j*pi/180).

deinterlace
Arguments:
- method={ 0 | 1 }

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

Default values:
method=0.

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

---

delaunay

**Arguments:**

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

**Description:**

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

Input images must be scalar.

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

**Examples of use:**

- **Example #1**

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

- **Example #2**

  image.jpg 100%,100% noise. 2,2 eq. 1 delaunay. +blend shapeaverage0
delaunay3d

No arguments

Description:
Generate 3D Delaunay triangulations from selected images.

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

Example of use:

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

**Arguments:**

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

**Description:**

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

---

deltaE

**Arguments:**

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

**Description:**

Compute the CIE DeltaE color difference between selected images and specified image. Argument to_Lab_command is a command able to convert colors of image into a Lab representation.

**Default values:**

metric=1 and to_Lab_command="srgb2lab".

**Example of use:**

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

Arguments:

- \_run\_in\_parallel={ 0:no | 1:yes | 2:auto }

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

denoise

Arguments:

- \[guide\],std\_deviation\_s\[%\]>=0,std\_deviation\_r\[%\]>=0,\_patch\_size>0,\_lookup\_size>0,\_smoothness,0 | 1 }
  or
- \ \_std\_deviation\_s\[%\]>=0,\_std\_deviation\_r\[%\]>=0,\_patch\_size>0,\_lookup\_size>0,\_smoothness,0 | 1 }

Description:
Denoise selected images by non-local patch averaging.

Default values:
patch\_size=5, lookup\_size=6 and smoothness=1.

Example of use:

image.jpg +denoise 5,5,8
denoise_cnn

Arguments:

- _noise_type={ 0:soft | 1:heavy | 2:heavy (faster) | 3:poisson+gaussian | 4:poisson+gaussian2 }, _patch_size>0

Description:

Denoise selected images using a convolutional neural network (CNN).

Input value range should be [0,255]. Output value range is [0,255].

Default values:

patch_size=64.

Example of use:

```
image.jpg noise 20 cut 0,255 +denoise_cnn
```

denoise_haar

Arguments:

- _threshold>=0, _nb_scales>=0, _cycle_spinning>0
**Description:**
Denoise selected images using haar-wavelet thresholding with cycle spinning. Set nb_scales==0 to automatically determine the optimal number of scales.

**Default values:**
threshold=1.4, nb_scale=0 and cycle_spinning=10.

**Example of use:**

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

---

denoise_patchpca

**Arguments:**

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

**Description:**
Denoise selected images using the patch-pca algorithm.

**Default values:**
patch_size=7, lookup_size=11, details=1.8 and spatial_sampling=5.

**Example of use:**

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

**Arguments:**

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

**Description:**

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

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

**Default values:**

`boundary_conditions=1`.

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

**Examples of use:**

- **Example #1**

  ```
  image.jpg deriche 3,1,x
  ```
detect_skin

Arguments:

- $0 \leq \text{tolerance} \leq 1$, skin_x, skin_y, skin_radius >= 0

Description:

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 values:

tolerance=0.5 and skin_x=skin_y=radius=-1.

diagonal

No arguments

Description:
Transform selected vectors as diagonal matrices.

**Example of use:**

```
1,10,1,1,'y' +diagonal
```

---

**diffusiontensors**

**Arguments:**

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

**Description:**

Compute the diffusion tensors of selected images for edge-preserving smoothing algorithms.

**Default values:**

- sharpness=0.7, anisotropy=0.3, alpha=0.6, sigma=1.1 and is_sqrt=0.

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

**Example of use:**

```
image.jpg diffusiontensors 0.8 abs pow 0.2
```
**dijkstra**

**Arguments:**

- \( \text{starting}_\text{vertex} \geq 0, \_\text{ending}_\text{vertex} = \{ -1: \text{none} | \geq 0 \} \)

**Description:**

Compute minimal distances/paths in selected graphs, from specified starting_vertex to all other vertices (opt. only until ending_vertex has been reached).

A graph of N vertices is specified as a NxN adjacency matrix giving the weights of all edges connecting vertices (set to inf when two vertices are not connected).

This command return a 1xNx1x2 image containing the \([\text{distance}, \text{parent}]\) information:

- \( \text{distance} \) is the minimal distance from vertex \( y \) to the starting_vertex (i.e. the sum of edge weights composing the minimal path between these two vertices).
- \( \text{parent} \) is the index of the next vertex that must be followed to reaches the starting_vertex through the minimal path.

**Default values:**

- \( \text{ending}_\text{vertex} = -1 \)

---

**dilate**

**Built-in command**

**Arguments:**

- \( \text{size} \geq 0 \) or
- \( \text{size}_x \geq 0, \text{size}_y \geq 0, \text{size}_z \geq 0 \) or
- \( [\text{kernel}], \_\text{boundary}_\text{conditions}, \_\text{is}_\text{real} = \{ 0: \text{binary-mode} | 1: \text{real-mode} \} \)

**Description:**

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

boundary_conditions can be \( \{ 0: \text{dirichlet} | 1: \text{neumann} | 2: \text{periodic} | 3: \text{mirror} \} \).

**Default values:**

- \( \text{size}_z = 1, \_\text{boundary}_\text{conditions} = 1 \) and \( \_\text{is}_\text{real} = 0 \).

**Example of use:**

```
image.jpg +dilate 10
```
dilate_circ

Arguments:
- \( \text{size} \geq 0, \text{boundary_conditions}, \text{is_real} = \{ 0 \mid 1 \} \)

Description:
Apply circular dilation of selected images by specified size.

boundary_conditions can be \{ \text{0:dirichlet} \mid \text{1:neumann} \mid \text{2:periodic} \mid \text{3:mirror} \}.

Default values:
boundary_conditions=1 and is_real=0.

Example of use:

```
image.jpg +dilate_circ 7
```

dilate_oct

Arguments:
- \( \text{size} \geq 0, \text{boundary_conditions}, \text{is_real} = \{ 0 \mid 1 \} \)

Description:
Apply octagonal dilation of selected images by specified size.

**Default values:**

boundary_conditions=1 and is_real=0.

**Example of use:**

```
image.jpg +dilate_oct 7
```

---

**dilate_threshold**

**Arguments:**

- size_x\geq 1, size_y\geq 1, size_z\geq 1, threshold\geq 0, boundary_conditions

**Description:**

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

boundary_conditions can be \{ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror \}.

**Default values:**

size_y=size_x, size_z=1, threshold=255 and boundary_conditions=1.

---

**direction2rgb**

**No arguments**

**Description:**

Compute RGB representation of selected 2D direction fields.

**Example of use:**

```
image.jpg luminance gradient append c blur 2 orientation +direction2rgb
```
discard

Built-in command

Arguments:

• _value1, _value2,... or
• { x | y | z | c}...{ x | y | z | c}, _value1, _value2,... or
• (no arg)

Description:

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

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

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

Examples of use:

• Example #1

(1;2;3;4;3;2;1) +discard 2

• Example #2

(1,2,2,3,3,3,4,4,4,4) +discard x
displacement

Arguments:

• [source_image],_smoothness,_precision>=0,_nb_scales>=0,_iteration_max>=0,is_backward={0 | 1 },,[guide]

Description:

Estimate displacement field between specified source and selected target images.

If smoothness>=0, regularization type is set to isotropic, else to anisotropic. If nb_scales==0, the number of scales used is estimated from the image size.

Default values:

smoothness=0.1, precision=5, nb_scales=0, iteration_max=10000, is_backward=1 and [guide]=(unused).

Example of use:

```
image.jpg +rotate 3,1,0,50%,50% +displacement[-1] [-2] quiver[-1] [-1],15,1,1,1,{1.5*iM}
```
display

No arguments

Description:
Display selected images in an interactive window.
(equivalent to shortcut command \( d \)).

When invoked with a + prefix (i.e. +display), the command outputs its log messages on stdout rather than on stderr.
Display window #0 is used as the default window for the display, if already opened.

Available controls are shown below (where LMB = Left mouse button, RMB = Right mouse button, MMB = Middle mouse button and MW = Mouse wheel).

- **Thumbnail navigation bar:**
  TAB: Show/hide thumbnails - LMB: Select thumbnail or shift thumbnail bar - 0-9,ARROWS (opt. +SHIFT),B,BACKSPACE,C,E,END,H,HOME,SPACE: Navigate and select thumbnails (add CTRL if mouse pointer is outside thumbnail bar).

- **Image view:**

- **3D mesh view:**

- **2D images specific:**
  CTRL+LMB: Rectangular selection.

- **3D volumetric images specific:**
  CTRL+MW: Pan along orthogonal axis - X: Reset area layout.

- **Window size, decoration and data I/O:**

- **Configuration variables:**
  The viewer configuration can be tuned by assigning the following variables:
  - \_display\_selected is an integer or an image name that tells which image is selected by default.
  - \_display\_alpha can be \{ 0:off | 1:on | 2:over black | 3:over gray | 4:over \}. 
display0

No arguments

Description:
Display selected images in an interactive window, without normalization and alpha mode activated.
**display_array**

**Arguments:**
- \_width>0, \_height>0

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

**Default values:**
width=13 and height=width.

---

**display_camera**

**No arguments**

**Description:**
Open camera viewer.

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

---

**display_clut**

**Arguments:**
- \_image\_resolution>0, \_clut\_resolution>0

**Description:**
Display selected 3D color LUTs.

**Default values:**
image\_resolution=320 and clut\_resolution=33.

**Example of use:**

```
clut teal orange clut summer clut 60s display\_clut 400
```
display_fft

No arguments

Description:
Display fourier transform of selected images, with centered log-module and argument.

(equivalent to shortcut command `dfft`).

Example of use:

```plaintext
image.jpg +display_fft
```

display_graph
**Arguments:**

- `width>=0, height>=0, plot_type, vertex_type, xmin, xmax, ymin, ymax, xlabel, ylabel, frame_size`

**Description:**

Render graph plot from selected image data.

plot_type can be `{ 0: none | 1: lines | 2: splines | 3: bar }.
vertex_type can be `{ 0: none | 1: points | 2,3: crosses | 4,5: circles | 6,7: squares }.
xmin, xmax, ymin, ymax set the coordinates of the displayed xy-axes.
if specified width or height is 0, then image size is set to half the screen size.

**Default values:**

width=0, height=0, plot_type=1, vertex_type=1, xmin=xmax=ymin=ymax=0 (auto), xlabel="x-axis", ylabel="y-axis" and frame_size=32.

**Example of use:**

```plaintext
128,1,1,1,'cos(x/10+u)' +display_graph 400,300,3
```

**display_histogram**

**Arguments:**

- `width>=0, height>=0, clusters>0, _min_value[%], _max_value[%], _show_axes={ 0 | 1 }, _expression`

**Description:**

Render a channel-by-channel histogram.

If selected images have several slices, the rendering is performed for all input slices.
expression is a mathematical expression used to transform the histogram data for visualization purpose.

*(equivalent to shortcut command dh).*

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

**Default values:**
Example of use:

```plaintext
image.jpg +display_histogram 512,300
```

---

display_parametric

**Arguments:**

- `width>0, height>0, outline_opacity, vertex_radius>=0, is_antialiased={ 0 | 1 }, is_decorated={ 0 | 1 }, xlabel, ylabel`

**Description:**

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

Curve points are defined as pixels of a 2 or 3-channel image. If the point image contains more than 3 channels, additional channels define the (R,G,B) color for each vertex. If `outline_opacity>1`, the outline is colored according to the specified vertex colors and `outline_opacity-1` is used as the actual drawing opacity.

**Default values:**

- `width=512, height=width, outline_opacity=3, vertex_radius=0, is_antialiased=1, is_decorated=1, xlabel="x-axis" and ylabel="y-axis"`

**Examples of use:**

- **Example #1**

  ```plaintext
  1024,1,1,2,'t=x/40;(!c?sin(t):cos(t))*(exp(cos(t))-2*cos(4*t)-sin(t/12)^5)' display_parametric 512,512
  ```
Example #2

```
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
```

**display_polar**

**Arguments:**

- `width>32, height>32, outline_type, fill_R, fill_G, fill_B, theta_start, theta_end`

**Description:**

Render polar curve from selected image data.

- `outline_type` can be `{ r<0: dots with radius -r | 0: no outline | r>0: lines+dots with radius r }`.
- `fill_color` can be `{ -1: no fill | R,G,B: fill with specified color }`.

**Default values:**

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

**Examples of use:**

- Example #1
Example #2

\[ 3000,1,1,1,'x^3/10' \] display_polar 400,400,1,-1,,0,\{15*360\}

---

**display_quiver**

**Arguments:**

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

**Description:**

Render selected images of 2D vectors as a field of 2D arrows.

*(equivalent to shortcut command dq).*

**Default values:**

size_factor=16, arrow_size=1.5 and color_mode=1.

**Example of use:**

\[ \text{image.jpg +luminance gradient[-1] xy rv[-2,-1] *[-2] -1 a[-2,-1] c crop 60,10,90,30 +display_quiver[1]} \]
**display_rgba**

**Arguments:**

- **background_RGB_color**

**Description:**

Render selected RGBA images over a checkerboard or colored background.  

*(equivalent to shortcut command drgba).*

**Default values:**

`background_RGB_color=undefined` (checkerboard).

**Example of use:**

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

**Arguments:**

- `size_factor > 0`, `ellipse_size >= 0`, `color_mode` = { 0: monochrome | 1: grayscale | 2: color }, `outline` = 0

**Description:**

Render selected images of tensors as a field of 2D ellipses.

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

**Default values:**

`size_factor=16`, `ellipse_size=1.5`, `color_mode=2` and `outline=2`.

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

**Example of use:**

```
image.jpg +diffusontensors 0.1,0.9 rescale2d. 64 +display_tensors. 16,2
```
display_voxels3d

No arguments

Description:
Display selected images as set of 3D voxels.
(equivalent to shortcut command `dv3d`).

display_warp

Arguments:
- `cell_size>0`

Description:
Render selected 2D warping fields.
(equivalent to shortcut command `dw`).

Default values:
- `cell_size=15`.

Example of use:
```
400,400,1,2,'x=x-w/2;y=y-h/2;r=sqrt(x*x+y*y);a=atan2(y,x);5*sin(r/10)*[cos(a),sin(a)]'
+display_warp 10
```
distance

Arguments:

• `isovalue[%], metric` or
• `isovalue[%],[metric],method`

Description:

Compute the unsigned distance function to specified isovalue, opt. according to a custom metric.

metric can be `{ 0:chebyshev | 1:manhattan | 2:euclidean | 3:squared-euclidean }.
method can be `{ 0:fast-marching | 1:low-connectivity dijkstra | 2:high-connectivity dijkstra | 3:1+return path | 4:2+return path }.

Default values:

metric=2 and method=0.

This command has a [tutorial page].

Examples of use:

• Example #1

  image.jpg threshold 20% distance 0 pow 0.3

• Example #2

  400,400 set 1,50%,50% +distance[0] 1,2 +distance[0] 1,1 distance[0]
distribution3d

No arguments

Description:
Get 3D color distribution of selected images.

Example of use:

```
image.jpg distribution3d colorcube3d primitives3d[-1] 1 add3d
```

ditheredbw

No arguments

Description:
Create dithered B&W version of selected images.

Example of use:

```
image.jpg +equalize ditheredbw[-1]
```
div

Arguments:

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

Description:

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

(equivalent to shortcut command /).

Examples of use:

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

- Example #2
  
  ```
  image.jpg +norm add[-1] 1 +div
  ```
div3d

Arguments:

- factor or
- factor_x, factor_y, factor_z

Description:

Scale selected 3D objects isotropically or anisotropically, with the inverse of specified factors.

(equivalent to shortcut command /3d).

Default values:

factor_z = 1.

Example of use:

torus3d 5,2 repeat 5 { +add3d[-1] 12,0,0 div3d[-1] 1.2 color3d[-1] ${-rgb} } add3d
### divergence

**No arguments**

**Description:**

Compute divergence of selected vector fields.

**Example of use:**

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

---

### do

**No arguments**

**Description:**

Start a do...while block.

**Example of use:**

```
image.jpg luminance i:=ia+2 do set 255,{u(100)}%,{u(100)}% while ia<i
```
**dog**

**Arguments:**
- \( \sigma_1 \geq 0\% \), \( \sigma_2 \geq 0\% \)

**Description:**
Compute difference of gaussian on selected images.

**Default values:**
\( \sigma_1 = 2\% \) and \( \sigma_2 = 3\% \).

**Example of use:**
```
image.jpg dog 2,3
```

---

**done**

**Built-in command**

**No arguments**

**Description:**
End a for/foreach/local/repeat...done block, and go to associated for/foreach/repeat if iterations remain.

*(equivalent to shortcut command \{}\).*
**double3d**

**Arguments:**

- `is_double_sided={ 0 | 1 }

**Description:**

Enable/disable double-sided mode for 3D rendering.

*(equivalent to shortcut command `db3d`)*.

**Default values:**

is_double_sided=1.

**Example of use:**

```plaintext
mode3d 1 repeat 2 { torus3d 100,30 rotate3d[-1] 1,1,0,60 double3d $>
snapshot3d[-1] 400 }
```

---

**draw_whirl**

**Arguments:**

- `amplitude>=0`

**Description:**

Apply whirl drawing effect on selected images.

**Default values:**

amplitude=100.

**Example of use:**

```plaintext
image.jpg draw_whirl ,
```
**drawing**

**Arguments:**
- \_amplitude>=0

**Description:**
Apply drawing effect on selected images.

**Default values:**
amplitude=200.

**Example of use:**
```
image.jpg +drawing ,
```

**drop_shadow**

**Arguments:**
- \_offset_x[%], \_offset_y[%], \_smoothness[%]>=0, curvature_x>=0, curvature_y>=0, \_expand_size={ 0 | 1 }, \_output_separate_layers={ 0 | 1 }

**Description:**
Drop shadow behind selected images.

**Default values:**

offset_x=20, offset_y=offset_x, smoothness=5, curvature_x=curvature_y=0, expand_size=1 and output_separate_layers=0.

**Example of use:**

```
image.jpg drop_shadow 10,20,5,0.5 display_rgba
```

---

drop\_shadow

**Arguments:**

- offset\_x[\%], offset\_y[\%], smoothness[\%]>=0, curvature\_x>=0, curvature\_y>=0, expand\_size={0 | 1 }, output\_separate\_layers={ 0 | 1 }

**Description:**

Drop shadow behind selected images.

**Default values:**

offset_x=20, offset_y=offset_x, smoothness=5, curvature_x=curvature_y=0, expand_size=1 and output_separate_layers=0.

**Example of use:**

```
image.jpg drop_shadow 10,20,5,0.5 display_rgba
```
**echo**

**Arguments:**
- `message`

**Description:**
Output specified message on the error output.

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

Command selection (if any) stands for displayed call stack subset instead of image indices. When invoked with a + prefix (i.e. `+echo`), the command output its message on stdout rather than stderr.

**echo_file**

**Arguments:**
- `filename,message`

**Description:**
Output specified message, appending it to specified output file.

(similar to echo for specified output file stream).

**edgels**

**Arguments:**
- `x0,y0,_n0,_is_high_connectivity={ 0 | 1 }`

**Description:**
Extract one or several lists of edgels (and their normals) that defines a 2D binary silhouette.
When specified (i.e. !=-1), arguments x0,y0,n0 are the coordinates of the starting edgel, which must be located on an edge of the binary silhouette.

- If x0,y0 and n0 are specified, only a single list of edgels is returned.
- If only x0,y0 are specified (meaning n0=-1), up to 4 lists of edgels can be returned, all starting from the same point (x0,y0).
- If no arguments are specified (meaning x0=y0=n0=-1), all possible lists of edgels are returned.

A list of edgels is returned as an image with 3 channels [x,y,n] where x and y are the 2D coordinates of the edgel pixel, and n is the orientation of its associated canonical normal (which can be \{ 0: [1,0] | 1:[0,1] | 2:[-1,0] | 3:[0,-1] \}).

**Default values:**

x0=y0=n0=-1 and is_high_connectivity=1.

---

**edges**

**Arguments:**

- **threshold[%]>=0**

**Description:**

Estimate contours of selected images.

**Default values:**

edges=15%

**Example of use:**

```
image.jpg +edges 15%
```

---

**eigen**

**Built-in command**

**No arguments**

**Description:**
Compute the eigenvalues and eigenvectors of selected symmetric matrices or matrix fields.

If one selected image has 3 or 6 channels, it is regarded as a field of 2x2 or 3x3 symmetric matrices, whose eigen elements are computed at each point of the field.

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

**Examples of use:**

- **Example #1**

\[
\begin{pmatrix}
1, & 0, & 0; \\
0, & 2, & 0; \\
0, & 0, & 3
\end{pmatrix}
+ \text{eigen}
\]

- **Example #2**

```bash
image.jpg structuretensors blur 2 eigen split[0] c
eigen2tensor
```

---

**eigen2tensor**
No arguments

Description:
Recompose selected pairs of eigenvalues/eigenvectors as 2x2 or 3x3 tensor fields.
This command has a tutorial page.

elevate

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

Description:
Elevate selected 2D images into 3D volumes.

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

elevation3d

Arguments:
- \{ z-factor | [elevation_map] | 'formula' \}, base_height={ -1 | >=0 } or (no arg)

Description:
Generate 3D elevation of selected images, opt. with a specified elevation map.
When invoked with (no arg) or z-factor, the elevation map is computed as the pointwise L2 norm of the pixel values. Otherwise, the elevation map is taken from the specified image or formula.

Examples of use:
- Example #1

  image.jpg +blur 5 elevation3d. 0.75
• **Example #2**

```
128,128,1,3,u(255) plasma 10,3 blur 4 sharpen 10000 n 0,255
elevation3d[-1] 'X=(x-64)/6;Y=(y-64)/6;-100*exp(-
(X^2+Y^2)/30)*abs(cos(X)*sin(Y))'
```

**elif**

**Arguments:**

- `condition`

**Description:**

Start a `elif`...[else]...fi block if previous if was not verified and test if specified condition holds.

Condition is a mathematical expression, whose evaluation is interpreted as `{ 0: false | other: true }`.

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

**ellipse**

**Arguments:**

- `x[%],y[%],R[%],r[%],_angle,_opacity,_pattern,_color1,...`
Description:
Draw specified colored ellipse on selected images.

A radius of 100% stands for $\sqrt{\text{width}^2 + \text{height}^2}$.
Pattern is an hexadecimal number starting with 0x which can be omitted even if a color is specified. If a pattern is specified, the ellipse is drawn outlined instead of filled.

Default values:
opacity=1, pattern=(undefined) and color1=0.

Example of use:
```
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
```

ellipsionism

Arguments:
- R>0[%, r>0[%, smoothness>=0[%], opacity, outline>0, density>0

Description:
Apply ellipsionism filter to selected images.

Default values:
R=10, r=3, smoothness=1%, opacity=0.7, outline=8 and density=0.6.

Example of use:
```
image.jpg ellipsionism
```
else

No arguments

**Description:**
Execute following commands if previous if or elif conditions failed.
This command has a [tutorial page](#).

empty3d

No arguments

**Description:**
Input empty 3D object.

**Example of use:**

```plaintext
empty3d
```

endian

Built-in command
**Arguments:**

- `datatype`

**Description:**
Reverse data endianness of selected images, eventually considering the pixel being of the specified datatype.

datatype can be `{ bool | uint8 | int8 | uint16 | int16 | uint32 | int32 | uint64 | int64 | float32 | float64 }.

This command does nothing for bool, uint8 and int8 datatypes.

---

**eq**

**Built-in command**

**Arguments:**

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

**Description:**
Compute the boolean equality of selected images with specified value, image or mathematical expression, or compute the boolean equality of selected images.

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

**Examples of use:**

- **Example #1**

  ```plaintext```
  image.jpg round 40 eq {round(ia,40)}
  ```plaintext```

- **Example #2**

  ```plaintext```
  image.jpg +mirror x eq
  ```plaintext```
equalize

**Arguments:**

- \( \text{nb_levels}>0\% \), \( \text{value_min}\% \), \( \text{value_max}\% \)  or
- (no arg)

**Description:**

Equalize histograms of selected images.

If value range is specified, the equalization is done only for pixels in the specified value range.

**Default values:**

\( \text{nb_levels}=256 \), \( \text{value_min}=0\% \) and \( \text{value_max}=100\% \).

**Examples of use:**

- **Example #1**

  ```plaintext
  image.jpg +equalize
  ```

- **Example #2**

  ```plaintext
  image.jpg +equalize 4,0,128
  ```
equirectangular2nadirzenith

No arguments

Description:
Transform selected equirectangular images to nadir/zenith rectilinear projections.

erf

No arguments

Description:
Compute the pointwise error function of selected images.

Examples of use:

• Example #1

  image.jpg +normalize 0,2 erf[-1]

• Example #2

  300,1,1,1,'7*x/w-3.5+u' +erf display_graph 400,300
erode

**Arguments:**

- $\text{size} \geq 0$ or
- $\text{size}_x \geq 0, \text{size}_y \geq 0, \text{size}_z \geq 0$ or
- $[\text{kernel}], \text{boundary_conditions}, \text{is_real} = \{ 0: \text{binary-mode} | 1: \text{real-mode} \}$

**Description:**

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

boundary_conditions can be $\{ 0: \text{dirichlet} | 1: \text{neumann} | 2: \text{periodic} | 3: \text{mirror} \}$.

**Default values:**

$\text{size}_z = 1, \text{boundary_conditions} = 1$ and $\text{is_real} = 0$.

**Example of use:**

```
image.jpg +erode 10
```

erode_circ

**Arguments:**

- $\text{size} \geq 0, \text{boundary_conditions}, \text{is_real} = \{ 0 | 1 \}$
**Description:**
Apply circular erosion of selected images by specified size.

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

**Default values:**
boundary_conditions=1 and is_real=0.

**Example of use:**
```
image.jpg +erode_circ 7
```

**erode_oct**

**Arguments:**
- `size>=0, boundary_conditions, is_real={ 0 | 1 }`

**Description:**
Apply octagonal erosion of selected images by specified size.

**Default values:**
boundary_conditions=1 and is_real=0.

**Example of use:**
```
image.jpg +erode_oct 7
```
erode_threshold

**Arguments:**

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

**Description:**

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

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

**Default values:**

size_y=size_x, size_z=1, threshold=255 and boundary_conditions=1.

---

error

**Arguments:**

- message

**Description:**

Print specified error message on the standard error (stderr) and exit interpreter, except if error is caught by a onfail command.

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

---

euclidean2polar

**Arguments:**

- center_x[], center_y[], stretch_factor>0, boundary_conditions={ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror }

**Description:**
Apply euclidean to polar transform on selected images.

**Default values:**

center_x=center_y=50%, stretch_factor=1 and boundary_conditions=3.

**Example of use:**

```
image.jpg +euclidean2polar ,
```

---

**eval**

**Arguments:**

- expression

**Description:**

Evaluate specified math expression.
- If no command selection is specified, the expression is evaluated once and its result is set to status.
- If command selection is specified, the evaluation is looped over selected images. Status is unchanged. In this case, eval is similar to fill without assigning the image values.

---

**exec**

**Arguments:**

- is_verbose={ 0 | 1 },"command"

**Description:**

Execute external command using a system call.

The status value is then set to the error code returned by the system call.
If is_verbose=1, the executed command is allowed to output on stdout/stderr.

*(equivalent to shortcut command \x).*
**Default values:**
is_verbose=1.

---

**exec_out**

**Arguments:**

- `_mode","command"

**Description:**

Execute external command using a system call, and return resulting stdout and/or stderr.

mode can be `{ 0:stdout | 1:stderr | 2:stdout+stderr }`.

---

**exp**

Built-in command

**No arguments**

**Description:**

Compute the pointwise exponential of selected images.

**Examples of use:**

- **Example #1**

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

- **Example #2**

  ```
  300,1,1,1,'7*x/w+u' +exp display_graph 400,300
  ```
**expand**

**Arguments:**

- axes, size[%], boundary_conditions={ 0: dirichlet | 1: neumann | 2: periodic | 3: mirror }

**Description:**

Expand selected images along the specified axes.

axes can be { x | y | z | c | xy | xz | xc | yz | yc | zc | xyz | xyc | xzc | yzc | xyzc }.

**Default values:**

boundary_conditions=0.

**Example of use:**

```plaintext
image.jpg expand xy,30
```

**extract**

**Arguments:**

- "condition", output_type={ 0: xyzc-coords | 1: xyz-coords | 2: scalar-values
Description:
Extract a list of coordinates or values from selected image, where specified mathematical condition holds. For N coordinates matching, result is a 1xNx1x4 image.

Default values:
output_type=0.

Example of use:

```
sp lena +extract "norm(I)>128",3
```

---

**extract_region**

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

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

Default values:
extract_xyz_coordinates=0.

Example of use:

```
image.jpg +blur 3 quantize. 4,0 +extract_region[0] [1],0,1,3
```
extract_textures3d

No arguments

Description:
Extract texture data from selected 3D objects.

Example of use:

```
image.jpg imagesphere3d 10,10 +extract_textures3d
```

extrude3d

Arguments:
• \text{depth} > 0, \text{resolution} > 0, \text{smoothness} \% \geq 0

\textbf{Description:}

Generate extruded 3D object from selected binary XY-profiles.

\textbf{Default values:}

depth=16, resolution=1024 and smoothness=0.5%.

\textbf{Example of use:}

\begin{verbatim}
image.jpg threshold 50% extrude3d 16
\end{verbatim}

\begin{center}
\includegraphics[width=0.5\textwidth]{image.jpg}
\end{center}

\textbf{Argument:}

\begin{itemize}
  \item \textsize > 0
\end{itemize}

\textbf{Description:}

Insert an identity matrix of given size at the end of the image list.

\textbf{Example of use:}

\begin{verbatim}
eye 3 eye 7 eye 10
\end{verbatim}

\begin{center}
\includegraphics[width=0.5\textwidth]{eye.png}
\end{center}
fade_diamond

Arguments:
• 0<=_start<=100,0<=_end<=100

Description:
Create diamond fading from selected images.

Default values:
start=80 and end=90.

Example of use:

```plaintext
image.jpg testimage2d {w},{h} +fade_diamond 80,85
```

fade_files

Arguments:
• "filename_pattern",_nb_inner_frames>0,_first_frame>=0,_last_frame={ >=0 | -1=last },_frame_step>=1,_output_filename

Description:
Generate a temporal fading from specified input image files, in a streamed way.

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

**Default values:**

nb_inner_frames=10, first_frame=0, last_frame=-1, frame_step=1 and output_filename=(undefined).

---

**fade_linear**

**Arguments:**

- \_angle, 0<= \_start<=100, 0<= \_end<=100

**Description:**

Create linear fading from selected images.

**Default values:**

angle=45, start=30 and end=70.

**Example of use:**

```plaintext
image.jpg testimage2d {w},{h} +fade_linear 45,48,52
```

---

**fade_radial**

**Arguments:**
Description:
Create radial fading from selected images.

Default values:
start=30 and end=70.

Example of use:
```
image.jpg testimage2d {w},{h} +fade_radial 30,70
```

Fade_video

Arguments:
- `video_filename`, `nb_inner_frames>=0`, `first_frame>=0`, `last_frame={ >=0 | -1=last }`, `frame_step>=1`, `output_filename`

Description:
Create a temporal fading sequence from specified input video file, in a streamed way.

If a display window is opened, rendered frames are displayed in it during processing.
This command requires features from the OpenCV library (not enabled in G'MIC by default).

Default values:
nb_inner_frames=10, first_frame=0, last_frame=-1, frame_step=1 and output_filename=(undefined).
**fade_x**

**Arguments:**
- $0 \leq _{start} \leq 100, 0 \leq _{end} \leq 100$

**Description:**
Create horizontal fading from selected images.

**Default values:**
start=30 and end=70.

**Example of use:**

```
image.jpg testimage2d {w},{h} +fade_x 30,70
```

![Example of fade_x](image.jpg)

---

**fade_y**

**Arguments:**
- $0 \leq _{start} \leq 100, 0 \leq _{end} \leq 100$

**Description:**
Create vertical fading from selected images.
**Default values:**

start=30 and end=70.

**Example of use:**

```
image.jpg testimage2d {w},{h} +fade_y 30,70
```

![Example images](image.jpg) ![Example images](testimage2d.jpg)

**fade_z**

**Arguments:**

- \(0 \leq \text{start} \leq 100, 0 \leq \text{end} \leq 100\)

**Description:**

Create transversal fading from selected images.

**Default values:**

start=30 and end=70.

**fft**

**Arguments:**

- \(\{x|y|z\} \ldots \{x|y|z\}\)
Description:
Compute the direct fourier transform (real and imaginary parts) of selected images, optionally along the specified axes only.

See also:
ifft.
This command has a tutorial page.

Examples of use:
• Example #1

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

• Example #2

```plaintext
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]
```

fftpolar
No arguments
Description:
Compute Fourier transform of selected images, as centered magnitude/phase images.

**Example of use:**

```plaintext
image.jpg fftpolar ellipse 50%,50%,10,10,0,1,0 ifftpolar
```

---

**fi**

**No arguments**

**Description:**

End a if...[elif]...[else]...fi block.

*(equivalent to shortcut command [fi].)*

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

---

**fibonacci**

**Arguments:**

- \( N \geq 0 \)

**Description:**

Return the \( N \)th number of the Fibonacci sequence.

**Example of use:**

```plaintext
echo \${"fibonacci 10"}
```

---

[gmic]-0./ Start G'MIC interpreter.
[gmic]-0./ 55
[gmic]-0./ End G'MIC interpreter.
file_mv

Arguments:

• filename_src, filename_dest

Description:

Rename or move a file from a location $1 to another location $2.

filename

Arguments:

• filename, number1, number2, ..., numberN

Description:

Return a filename numbered with specified indices.

filename_dated

Arguments:

• filename

Description:

Convert specified filename to one stamped with the current date (filename_YYYYMMDD_HHMMSS.ext).

filename_rand

No arguments

Description:

Return a random filename for storing temporary data.

files

Built-in command

Arguments:
• _mode,path

**Description:**

Return the list of files and/or subfolders from specified path.

path can be eventually a matching pattern.

mode can be { 0: files only | 1: folders only | 2: files + folders }.

Add 3 to mode to return full paths instead of filenames only.

**Default values:**

mode=5.

---

**files2img**

**Arguments:**

• _mode,path

**Description:**

Insert a new image where each vector-valued pixel is a string encoding the filenames returned by command `files`.

Useful to manage list of filenames containing characters that have a special meaning in the G'MIC language, such as spaces or commas.

---

**files2video**

**Arguments:**

• "filename_pattern",_output_filename,fps>0,_codec

**Description:**

Convert several files into a single video file.

**Default values:**

output_filename=output.mp4, fps=25 and codec=mp4v.

---

**fill**

**Built-in command**

**Arguments:**

• value1,value2,... or
• [image] or
• 'formula'
Description:
Fill selected images with values read from the specified value list, existing image or mathematical expression. Single quotes may be omitted in formula.
(equivalent to shortcut command f).
This command has a tutorial page.

Examples of use:

• Example #1

\[ 4,4 \text{ fill } 1,2,3,4,5,6,7 \]

• Example #2

\[ 4,4 (1,2,3,4,5,6,7) \text{ fill[-2] [-1]} \]

• Example #3

\[ 400,400,1,3 \text{ fill } "X=x-w/2; Y=y-h/2; R=\sqrt{X^2+Y^2}; a=\arctan2(Y,X); R<=180?255*\text{abs}(\cos(c+200*(x/w-0.5)*(y/h-0.5))):850*(a%(0.1*(c+1)))" \]
**fill_color**

**Arguments:**
- col1,...,colN

**Description:**
Fill selected images with specified color.

*(equivalent to shortcut command \texttt{fc}).*

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

**Example of use:**

```
image.jpg +fill_color 255,0,255
```

---

**fire_edges**

**Arguments:**
- edges>=0,0<=attenuation<=1,smoothness>=0,threshold>=0,nb_frames>0,starting=

**Description:**
Generate fire effect from edges of selected images.
Default values:
edges=0.7, attenuation=0.25, smoothness=0.5, threshold=25, nb_frames=1, starting_frame=20 and frame_skip=0.

Example of use:

```
image.jpg fire_edges ,
```

---

fisheye

**Arguments:**

- `center_x, center_y, 0<=_radius<=100, _amplitude>=0`

**Description:**

Apply fish-eye deformation on selected images.

**Default values:**

x=y=50, radius=50 and amplitude=1.2.

**Example of use:**

```
image.jpg +fisheye ,
```
**fitratio_wh**

**Arguments:**
- `min_width, min_height, ratio_wh`

**Description:**
Return a 2D size width, height which is bigger than `min_width, min_height` and has the specified width/height ratio.

**fitsamples**

**Arguments:**
- `nb_samples>0, relevant_dimension[%]>0, average_vector_varname, dilation_vector_varname`

**Description:**
Generate `nb_samples` vectors having the same multivariate gaussian distribution as the vectors of the selected images.

Each input represents a set of M vectors of dimension N (with M>1) (specified as an image with size `MxNx1x1, Mx1xNx1, Mx1x1xN, 1xMxNx1, 1xMx1xN or 1x1xMxN`). The command returns a new set of random vectors with similar geometry.

**Default values:**
- `relevant_dimension=100%`, and
- `average_vector_varname=orientation_matrix_varname=dilation_matrix_varname=(undefined)`.

**fitscreen**

**Arguments:**
- `width, height, depth, minimal_size[%], maximal_size[%]` or
- `[image], minimal_size[%], maximal_size[%]`

**Description:**
Return the ideal size WxH for a window intended to display an image of specified size on screen.

**Default values:**
- `depth=1`, `minimal_size=128` and `maximal_size=85%`.

**flood**

Built-in command
Arguments:

- \(x\%\), \(y\%\), \(z\%\), \(\text{tolerance} \geq 0\), \(\text{is high connectivity} = \{0 \mid 1\}\), \(\text{opacity}\), \(\text{color1}\)... 

Description:

Flood-fill selected images using specified value and tolerance.

Default values:

\(y=z=0\), \(\text{tolerance}=0\), \(\text{is high connectivity}=0\), \(\text{opacity}=1\) and \(\text{color1}=0\).

Example of use:

```bash
image.jpg repeat 1000 flood \{u(100)\}\%,\{u(100)\}\%,0,20,0,1,${-rgb} done
```

---

**flower**

Arguments:

- \(\text{amplitude}\), \(\text{frequency}\), \(\text{offset_r}\%\), \(\text{angle}\), \(\text{center_x}\%\), \(\text{center_y}\%\), \(\text{boundary_conditions} = \{0: \text{dirichlet} \mid 1: \text{neumann} \mid 2: \text{periodic} \mid 3: \text{mirror}\}\)

Description:

Apply flower deformation on selected images.

Default values:

\(\text{amplitude}=30\), \(\text{frequency}=6\), \(\text{offset_r}=0\), \(\text{angle}=0\), \(\text{center_x}=\text{center_y}=50\%\) and \(\text{boundary_conditions}=3\).

Example of use:

```bash
image.jpg +flower,
```
focale3d

Arguments:

• focale

Description:

Set 3D focale.

(equivalent to shortcut command f3d).

Set focale to 0 to enable parallel projection (instead of perspective).
Set negative focale will disable 3D sprite zooming.

Default values:

focale=700.

Example of use:

```
repeat 5 { torus3d 100,30 rotate3d[-1] 1,1,0,60 focale3d {$<*90}
snapshot3d[-1] 400 } remove[0]
```

[Images of torus3d examples]
font

Arguments:

- { 'Font_name' | font_number | font.gmz }, _font_height[%]>0, _is_bold={ 0 | 1 }

Description:

Return font identifier (variable name) that can be further used in command text as a custom font.

Font name can be { Acme | Arial | ArialBlack | BlackOpsOne | BlackChancery | CabinSketch | Caprasimo | CarnevaleFreakshow | CheeseBurger | Cheque | ChequeBlack | Chlorinar | ComicSansMS | CourierNew | Creepster | Georgia | Hidayatullah | Impact | Jaro | Lobster | LuckiestGuy | Macondo | MedievalSharp | OdinRounded | Oswald | PalatinoLinotype | PlayfairDisplay | Roboto | Satisfy | Sofia | SundayMilk | TexGyreAdventor | TimesNewRoman | TitanOne | Typewriter | Verdana }.

If a filename font.gmz is specified, it must be a file converted with command font2gmz.

Default values:

font_height=64 and is_bold=0.

Example of use:

400,300,1,3 text "Hello World!",0.5~,0.5~,${"font \"Cheese Burger\",80"},1,255,255,128
font2gmz

Arguments:
• \texttt{\_font\_name,\_font\_size>0,\_font\_qualifier}

Description:
Convert specified font to \texttt{G'MIC} format, so that it can be used as a custom font for command \texttt{text}.

\texttt{font\_name} can be either a filename as \texttt{font.ttf}, or a Google Font Name. This command requires the command line tool \texttt{cutycapt} to be installed on your system. Beware, \texttt{font\_size} is the size of font used for the rendering, it does \texttt{not} correspond to the font height.

Default values:
\texttt{font\_name=Sofia, font\_size=24 and font\_qualifier=""}.

---

fontchart

Arguments:
• \texttt{display\_mode}.

Description:
Insert \texttt{G'MIC} font chart at the end of the image list.

\texttt{display\_mode} can be \{ 0: List of characters | N: List of fonts with height 'N'}.

Default values:
\texttt{display\_mode=0}.

Example of use:
\texttt{fontchart 0 fontchart 64}

---
for

Arguments:

- condition

Description:

Start a `for...done` block.

Example of use:

```bash
image.jpg rescale2d,32,400,400,1,3 x=0 for $x<400 image[1] [0],$x,$x x+=40 done
```

foreach

No arguments

Description:

Start a `foreach...done` block, that iterates over all images in the selection, with a separate local environment for each one.

Example of use:

```bash
sample colorful,earth,duck,dog foreach[^2] +blur 10 sub normalize 0,255 done
```
fov3d

Arguments:
• $\text{fov\_angle} \geq 0, \text{image\_resolution} > 0$

Description:
Set 3D focal to match specified field of vision angle (in degree) for rendering a 3D object in an image with specified resolution.

Return corresponding value of the focale in status.

Default values:
$\text{fov\_angle}=45$ and $\text{image\_size}=\text{max}(w,h)$ (max size of the latest image).

fps

No arguments

Description:
Return the number of time this function is called per second, or -1 if this info is not yet available.

Useful to display the framerate when displaying animations.
fractalize

**Arguments:**
- \(0 \leq \text{detail}_\text{level} \leq 1\)

**Description:**
Randomly fractalize selected images.

**Default values:**
detail_level=0.8

**Example of use:**
```
image.jpg fractalize ,
```

---

frame

**Arguments:**
- \(\text{axes, size\%}, \text{col1}, \ldots, \text{colN}\)

**Description:**
Insert outer frame in selected images, along the specified axes.

axes can be \(\{ x \mid y \mid z \mid xy \mid xz \mid yz \mid xyz \}\).

**Default values:**
axes=xy, size=10%, col1=col2=col3=col4=255.

**Example of use:**
```
image.jpg frame xy,10%,255,128,0
```
frame_blur

Arguments:
• sharpness>0, size>=0, smoothness, shading, blur

Description:
Draw RGBA-colored round frame in selected images.

Default values:
sharpness=10, size=30, smoothness=0, shading=1 and blur=3%.

Example of use:

image.jpg frame_blur 3,30,8,10%

frame_cube

Arguments:
• depth>=0, centering_x, centering_y, left_side={ 0:normal | 1:mirror-x | 2:mirror-y | 3:mirror-xy }, right_side, lower_side, upper_side

Description:
Insert 3D frames in selected images.
Default values:
depth=1, centering_x=centering_y=0 and left_side=right_side, lower_side=upper_side=0.

Example of use:

```
image.jpg frame_cube
```

---

**frame_fuzzy**

**Arguments:**

- `size_x[%]>=0, size_y[%]>=0, fuzzyness>=0, smoothness[%]>=0, R, G, B, A`

**Description:**

Draw RGBA-colored fuzzy frame in selected images.

**Default values:**

size_y=size_x, fuzzyness=5, smoothness=1 and R=G=B=A=255.

**Example of use:**

```
image.jpg frame_fuzzy 20
```
frame_painting

Arguments:
- \( size[\%] \geq 0, 0 \leq contrast \leq 1, \text{profile smoothness}[\%] \geq 0, R, G, B, \text{vignette size}[\%] \geq 0 \)

Description:
Add a painting frame to selected images.

Default values:
- size=10\%, contrast=0.4, profile_smoothness=6\%, R=225, G=200, B=120, vignette_size=2\%,
- vignette_contrast=400, defects_contrast=50, defects_density=10, defects_size=1,
- defects_smoothness=0.5\% and serial_number=123456789.

Example of use:
```
image.jpg frame_painting ,
```

frame_pattern

Arguments:
- \( M \geq 3, \text{constrain size} = \{ 0 \mid 1 \} \) or
- \( M \geq 3, [\text{frame_image}], \text{constrain size} = \{ 0 \mid 1 \} \)

Description:
Insert selected pattern frame in selected images.

Default values:
- pattern=0 and constrain_size=0.

Example of use:
```
image.jpg frame_pattern 8
```

![Image with frame](image.jpg)
frame_round

Arguments:

- size_x[%]>=0, size_y[%]>=0, radius[%]>=0, smoothness[%]>=0, col1,...,colN

Description:

Insert an inner round frame in selected images.

Default values:

size_x=size_y=5%, radius=30%, smoothness=0 and col=0,0,0,255.

frame_seamless

Arguments:

- frame_size>=0, patch_size>0, blend_size>=0, frame_direction={ 0:inner (preserve image size) | 1:outer }

Description:

Insert frame in selected images, so that tiling the resulting image makes less visible seams.

Default values:

patch_size=7, blend_size=5 and frame_direction=1.

Example of use:

```
image.jpg +frame_seamless 30 array 2,2
```
**function1d**

**Arguments:**
- $0\leq\text{smoothness}\leq1, x_0=0, y_0=x_1=0, y_1, \ldots, x_n=0, y_n$

**Description:**
Insert continuous 1D function from specified list of keypoints $(x_k,y_k)$ in range $[0,\max(x_k)]$ ($x_k$ are positive integers).

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

---

**gaussian**

**Arguments:**
- $\sigma_1[\%], \sigma_2[\%], \angle$

**Description:**
Draw a centered gaussian on selected images, with specified standard deviations and orientation.

**Default values:**
sigma1=3, sigma2=sigma1 and angle=0.

This command has a tutorial page.

Example of use:

```
400,400 gaussian 100,30,45
```

---

**gaussians3d**

**Arguments:**

- `size>0, opacity`

**Description:**

Convert selected 3D objects into set of 3D gaussian-shaped sprites.

**Example of use:**

```
image.jpg rescale2d ,32 distribution3d gaussians3d 20 colorcube3d primitives3d[-1] 1 +3d
g
```

---

**ge**

**Arguments:**

```
Built-in command
```
Description:
Compute the boolean 'greater or equal than' of selected images with specified value, image or mathematical expression, or compute the boolean 'greater or equal than' of selected images. (equivalent to shortcut command \texttt{>=}).

Examples of use:

• Example #1

image.jpg ge {ia}

• Example #2

image.jpg +mirror x ge

Arguments:

• \texttt{amplitude}>=0

\texttt{glow}
Description:
Add soft glow on selected images.

Default values:
amplitude=1%.

Example of use:

```
image.jpg glow
```

---

gmd2ascii

Arguments:

- \_max_line_length>0, _indent_forced_newlines>=0 or
- (no arg)

Description:
Convert selected gmd-formatted text images to ascii format.

Default values:
max_line_length=80 and indent_forced_newline=0.

---

gmd2html

Arguments:

- include_default_header_footer={ 0: none | 1: Reference | 2: Tutorial | 3: News } or
- (no arg)

Description:
Convert selected gmd-formatted text images to html format.
Default values:
include_default_header_footer=1.

---

gmic3d

No arguments

Description:
Input a 3D G'MIC logo.

Example of use:

```
gmic3d +primitives3d 1
```

---

gradient

Arguments:

- `{ x | y | z | c }...{ x | y | z | c },_scheme,_boundary_conditions` or
- `(no arg)`

Description:
Compute the gradient components (first derivatives) of selected images, along specified axes.

(equivalent to shortcut command `g`).

scheme can be `{ -1:backward | 0:centered | 1:forward | 2:sobel | 3:rotation-invariant (default) | 4:deriche | 5:vanvliet }.
boundary_conditions can be `{ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror }.
(no arg) compute all significant components.

Default values:

scheme=0 and boundary_conditions=1.
This command has a [tutorial page](#).

**Example of use:**

```
image.jpg gradient
```

---

**gradient2rgb**

**Arguments:**

- `is_orientation={ 0 | 1 }`

**Description:**
Compute RGB representation of 2D gradient of selected images.

**Default values:**

`is_orientation=0`.

**Example of use:**

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

---

**gradient_norm**
No arguments

Description:

Compute gradient norm of selected images.

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

Example of use:

```plaintext
image.jpg gradient_norm equalize
```

---

**gradient_orientation**

**Arguments:**

- `dimension={ 1 | 2 | 3 }`

**Description:**

Compute N-d gradient orientation of selected images.

**Default values:**

dimension=3.

**Example of use:**

```plaintext
image.jpg +gradient_orientation 2
```
graph

Arguments:

- [function_image], plot_type, vertex_type, ytop, ybottom, opacity, pattern, color

Description:

Draw specified function graph on selected images.

plot_type can be `{ 0:none | 1:lines | 2:splines | 3:bar }`.
vertex_type can be `{ 0:none | 1:points | 2,3:crosses | 4,5:circles | 6,7:squares }`.
pattern is an hexadecimal number starting with 0x which can be omitted even if a color is specified.

Default values:

plot_type=1, vertex_type=1, ytop=ybottom=0 (auto), opacity=1, pattern=(undefined)
and color1=0.

Example of use:

```
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]
```
grid

Arguments:

- \( \text{size}_x[\%] \geq 0, \text{size}_y[\%] \geq 0, \text{offset}_x[\%], \text{offset}_y[\%], \text{opacity}, \text{pattern}, \text{color}_1, \ldots \)

Description:

Draw xy-grid on selected images.

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

Default values:

\( \text{offset}_x = \text{offset}_y = 0, \text{opacity} = 1, \text{pattern} \) (undefined) and \( \text{color}_1 = 0. \)

Examples of use:

- Example #1

  \[
  \text{image.jpg grid 10\%, 10\%, 0, 0, 0.5, 255}
  \]

- Example #2

  \[
  400, 400, 1, 3, 255 \text{ grid 10\%, 10\%, 0, 0, 0.3, 0xCCCCCCCC, 128, 32, 16}
  \]
Arguments:

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

Description:

Compute the boolean greater than of selected images with specified value, image or mathematical expression, or compute the boolean greater than of selected images.

(equivalent to shortcut command \gt). 

Examples of use:

- Example #1

  \[ \text{image.jpg \ gt \ {ia}} \]

- Example #2

  \[ \text{image.jpg \ +mirror \ x \ gt} \]

---

**guided**

Built-in command

Arguments:

- [guide], radius[%] \geq 0, regularization[%] \geq 0 or
Description:
Blur selected images by guided image filtering.

If a guide image is provided, it is used to drive the smoothing process. A guide image must be of the same xyz-size as the selected images.
This command implements the filtering algorithm described in:
He, Kaiming; Sun, Jian; Tang, Xiaou, "Guided Image Filtering", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.35, no.6, pp.1397,1409, June 2013

Example of use:

```
image.jpg +guided 5,400
```

---

**gyroid3d**

Arguments:

- \( \text{resolution}\geq 0, \text{zoom}\geq 0 \)

Description:
Input 3D gyroid at (0,0,0), with specified resolution.

Default values:
resolution=32 and zoom=5.

Example of use:

```
gyroid3d 48 +primitives3d 1
```
haar

**Arguments:**

- `scale > 0`

**Description:**

Compute the direct haar multiscale wavelet transform of selected images.

**See also:**

[ihu](#)

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

hald2clut

**No arguments**

**Description:**

Convert selected 2D HaldCLUTs to 3D CLUTs.

halftone

**Arguments:**

- `nb_levels >= 2, size_dark >= 2, size_bright >= 2, shape = { 0:square | 1:diamond | 2:circle | 3:inv-square | 4:inv-diamond | 5:inv-circle }, smoothness[\%] >= 0`

**Description:**

Apply halftone dithering to selected images.

**Default values:**
Example of use:

```
image.jpg halftone
```

---

**hardsketchbw**

**Arguments:**

- \_amplitude>=0, \_density>=0, \_opacity, 0<= \_edge_threshold<=100, \_is_fast={ 0 | 1 }

**Description:**

Apply hard B&W sketch effect on selected images.

**Default values:**

amplitude=1000, sampling=3, opacity=0.1, edge_threshold=20 and is_fast=0.

**Example of use:**

```
image.jpg +hardsketchbw 200,70,0.1,10 median[-1] 2 +local reverse blur[-1] 3 blend[-2,-1] overlay done
```
**hcy2rgb**

*No arguments*

**Description:**
Convert color representation of selected images from HCY to RGB.

**hearts**

**Arguments:**
- `density`\(\geq 0\)

**Description:**
Apply heart effect on selected images.

**Default values:**

density=10.

**Example of use:**

```
image.jpg hearts ,
```
**heat_flow**

**Arguments:**
- \( \text{nb_iter} \geq 0, \text{dt}, \text{keep_sequence} = \{ 0 | 1 \} \)

**Description:**
Apply iterations of the heat flow on selected images.

**Default values:**
\( \text{nb_iter}=10, \text{dt}=30 \) and \( \text{keep_sequence}=0 \).

**Example of use:**
```
image.jpg +heat_flow 20
```

**help**

**Arguments:**
- \( \text{command} \) or \( \text{no arg} \)

**Description:**
Display help (optionally for specified command only) and exit.

*(equivalent to shortcut command \h).*

**hessian**

**Arguments:**
- \( \{ xx | xy | xz | yy | yz | zz \} \ldots \{ xx | xy | xz | yy | yz | zz \}, \text{boundary_conditions} \) or 
- \( \text{no arg} \)
Description:
Compute the hessian components (second derivatives) of selected images along specified axes.

boundary_conditions can be {0:dirichlet | 1:neumann | 2:periodic | 3:mirror}.
(no arg) compute all significant components.

Default values:
boundary_conditions=1.

Example of use:
```
image.jpg hessian
```

---

hex

Arguments:
- hexadecimal_int1,...

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

---

hex2dec

Arguments:
Description:
Convert specified hexadecimal integers into their decimal representations.

hex2img

Arguments:
- "hexadecimal_string"

Description:
Insert new image 1xN at the end of the list with values specified by the given hexadecimal-encoded string.

hex2str

Arguments:
- hexadecimal_string

Description:
Convert specified hexadecimal string into a string.

See also:
str2hex.

histogram

Arguments:
- nb_levels>0[%],_min_value[%],_max_value[%]

Description:
Compute the histogram of selected images.

If value range is set, the histogram is estimated only for pixels in the specified value range. Argument max_value must be specified if min_value is set.

Default values:
min_value=0% and max_value=100%.

Example of use:
**histogram3d**

No arguments

**Description:**
Get 3D color histogram of selected images.

**Example of use:**

```plaintext
image.jpg rescale2d 64 histogram3d circles3d 3 opacity3d. 0.75
colorcube3d primitives3d[-1] 1 add3d
```

**histogram_cumul**

**Arguments:**

- `nb_levels>0, is_normalized={0 | 1}, val0[%, val1[%]

**Description:**
Compute cumulative histogram of selected images.

**Default values:**
nb_levels=256, is_normalized=0, val0=0% and val1=100%.

Example of use:

```plaintext
image.jpg +histogram_cumul 256 histogram[0] 256 display_graph 400,300,3
```

---

**histogram_masked**

**Arguments:**

- `[mask],nb_levels>0, min_value, max_value`

**Description:**

Compute the masked histogram of selected images.

**Default values:**

min_value=0% and max_value=100%.

---

**histogram_nd**

**Arguments:**

- `nb_levels>0, value0, value1`

**Description:**

Compute the 1D, 2D or 3D histogram of selected multi-channels images (having 1, 2 or 3 channels).

If value range is set, the histogram is estimated only for pixels in the specified value range.

**Default values:**

value0=0% and value1=100%.

**Example of use:**
histogram_pointwise

Arguments:
• \texttt{nb\_levels>0}, \texttt{value0}, \texttt{value1}

Description:
Compute the histogram of each vector-valued point of selected images. If value range is set, the histogram is estimated only for values in the specified value range.

Default values:
\texttt{value0=0\%} and \texttt{value1=100\%}.

hough

Arguments:
• \texttt{width>0}, \texttt{height>0}, \texttt{gradient\_norm\_voting={ 0 | 1 }}

Description:
Compute hough transform (theta, rho) of selected images.

Default values:
width=512, height=width and \texttt{gradient\_norm\_voting=1}.

Example of use:
\texttt{image.jpg +blur 1.5 hough[-1] 400,400 blur[-1] 0.5 add[-1] 1 log[-1]}
houghsketchbw

Arguments:

- \_density\geq 0, \_radius>0, 0\leq \_threshold\leq 100, 0\leq \_opacity\leq 1, \_votesize[\%]>0

Description:

Apply hough B&W sketch effect on selected images.

Default values:

density=100, radius=3, threshold=100, opacity=0.1 and votesize=100%.

Example of use:

```plaintext
image.jpg +houghsketchbw ,
```

hsi2rgb

No arguments

Description:

Convert color representation of selected images from HSI to RGB.
hsi82rgb

No arguments

Description:
Convert color representation of selected images from HSI8 to RGB.

hsl2rgb

No arguments

Description:
Convert color representation of selected images from HSL to RGB.

hsl82rgb

No arguments

Description:
Convert color representation of selected images from HSL8 to RGB.

hsv2rgb

No arguments

Description:
Convert color representation of selected images from HSV to RGB.

Example of use:

```
(0,360;0,360^0,0;1,1^1,1;1,1) resize 400,400,1,3,3 hsv2rgb
```
hsv82rgb

No arguments

Description:
Convert color representation of selected images from HSV8 to RGB.

huffman_tree

No arguments

Description:
Generate Huffman coding tree from the statistics of all selected images.

Huffman tree is returned as a 1xN image inserted at the end of the image list, representing the N vector-valued leafs/nodes of the tree, encoded as `[value, parent, child0, child1]`. Last row of the returned image corresponds to the tree root. Selected images must contain only positive integer values. Return maximal value of the input data in the status.

See also:
compress_huffman, decompress_huffman.

idct

Arguments:

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

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

Output images are always evenly sized, so this command may change the size of the selected images. (dct images obtained with the dct command are evenly sized anyway).

Default values:

(no arg)

See also:
dct.

This command has a tutorial page.
identity

Arguments:

- \texttt{width}\geq0, \texttt{height}\geq0, \texttt{depth}\geq0

Description:
Insert an identity map of given size at the end of the image list.

Default values:

height=width and depth=1.

Example of use:

\begin{verbatim}
identity 5,1 identity 8,8
\end{verbatim}

iese

No arguments

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

Example of use:

\begin{verbatim}
image.jpg iee
\end{verbatim}
if

Arguments:

- condition

Description:

Start a if...[elif]...[else]...fi block and test if specified condition holds.

(condition is a mathematical expression, whose evaluation is interpreted as \( 0:\text{false} | \text{other:}\text{true} \).

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

Example of use:

```plaintext
image.jpg if ia<64 add 50% elif ia<128 add 25% elif ia<192 sub 25% else sub 50% fi cut 0,255
```

iift

Arguments:

- \( \{x | y | z\}...\{x | y | z\} \)

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

See also:

fft.

This command has a tutorial page.

---

ifftpolar

No arguments

Description:

Compute inverse fourier transform of selected images, from centered magnitude/phase images.

---

ihaar

Arguments:

- scale>0

Description:

Compute the inverse haar multiscale wavelet transform of selected images.

See also:

haar.

This command has a tutorial page.

---

ilaplacian

Arguments:

- \{ nb_iterations>0 | 0 \}, [initial_estimate]

Description:

Invert selected Laplacian images.

If given nb_iterations is 0, inversion is done in Fourier space (single iteration),
otherwise, by applying nb_iterations of a Laplacian-inversion PDE flow.
Note that the resulting inversions are just estimation of possible/approximated solutions.

Default values:
nb_iterations=0, axes=(undefined) and [initial_estimated]=(undefined).

**Example of use:**

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

![Image](image.jpg) ![Image](image.jpg) ![Image](image.jpg)

---

**image**

*Built-in command*

**Arguments:**

- `[sprite],_x[~%|~],_y[~%|~],_z[~%|~],_c[~,~],_opacity,[opacity_mask],_max_opacity_mask`

**Description:**

Draw specified sprite on selected images. *(equivalent to shortcut command [ ]).*

If one of the x,y,z or c argument ends with a ~, its value is expected to be a centering ratio (in [0,1]) rather than a position. Usual centering ratio are `{ 0: left-justified | 0.5: centered | 1: right-justified }`.

**Default values:**

x=y=z=c=0, opacity=1, opacity_mask=(undefined) and max_opacity_mask=1.

**Example of use:**

```
image.jpg +crop 40%,40%,60%,60% resize[-1] 200%,200%,1,3,5 frame[-1]
```
image6cube3d

No arguments

Description:
Generate 3D mapped cubes from 6-sets of selected images.

Example of use:
```
image.jpg animate flower,"30,0","30,5",6 image6cube3d
```

imagealpha

Arguments:

- `[sprite],_x[%~],_y[%~],_z[%~],_c[%~],_opacity`

Description:
Draw specified sprite on selected images, considering that the sprite's last channel is the drawing's alpha.

(equivalent to shortcut command `ja`).
If one of the x,y,z or c argument ends with a ~, its value is expected to be a centering ratio (in \([0,1]\)) rather than a position. Usual centering ratios are \( \{ 0: \text{left-justified} | 0.5: \text{centered} | 1: \text{right-justified} \} \).

**Default values:**

\(x=y=z=c=0\) and \(\text{opacity}=1\).

---

**imageblocks3d**

**Arguments:**

- \(_{\text{maximum_elevation}}, _{\text{smoothness}}\% \geq 0\)

**Description:**

Generate 3D blocks from selected images.

Transparency of selected images is taken into account.

**Default values:**

\(\text{maximum_elevation}=10\) and \(\text{smoothness}=0\).

**Example of use:**

```
image.jpg rescale2d ,32 imageblocks3d -20 mode3d 3
```

---

**imagecube3d**

**No arguments**

**Description:**

Generate 3D mapped cubes from selected images.

**Example of use:**
imagegrid

**Arguments:**

- \( M > 0, \ N > 0 \)

**Description:**

Create \( M \times N \) image grid from selected images.

**Default values:**

\( N = M \).

**Example of use:**

```
image.jpg imagegrid 16
```

---

imagegrid_hexagonal

**Arguments:**

- \( _\text{resolution} > 0, 0 \leq _\text{outline} \leq 1 \)
**Description:**
Create hexagonal grids from selected images.

**Default values:**
resolution=32, outline=0.1 and is_antialiased=1.

**Example of use:**
```
image.jpg imagegrid_hexagonal 24
```

---

**imagegrid_triangular**

**Arguments:**
- pattern_width\(\geq 1\), pattern_height\(\geq 1\), pattern_type, 0\(\leq\) outline_opacity\(\leq 1\), outline_color1...

**Description:**
Create triangular grids from selected images.

pattern type can be \{ 0:horizontal | 1:vertical | 2:crossed | 3:cube | 4:decreasing | 5:increasing \}.

**Default values:**
pattern_width=24, pattern_height=pattern_width, pattern_type=0, outline_opacity=0.1 and outline_color1=0.

**Example of use:**
```
image.jpg imagegrid_triangular 6,10,3,0.5
```
imageplane3d

No arguments

Description:
Generate 3D mapped planes from selected images.

Example of use:
```
image.jpg imageplane3d
```

imagepyramid3d

No arguments

Description:
Generate 3D mapped pyramids from selected images.

Example of use:
```
image.jpg imagepyramid3d
```
imagerubik3d

Arguments:

- \( \_xy\_tiles \geq 1, 0 \leq xy\_shift \leq 100, 0 \leq z\_shift \leq 100 \)

Description:
Generate 3D mapped Rubik's cubes from selected images.

Default values:

\( xy\_tiles = 3, xy\_shift = 5 \) and \( z\_shift = 5 \).

Example of use:

```plaintext
image.jpg imagerubik3d ,
```

imagesphere3d

Arguments:

- \( \_resolution1 \geq 3, \_resolution2 \geq 3 \)

Description:
Generate 3D mapped sphere from selected images.
Default values:

resolution1=32 and resolutions2=16.

Example of use:

```
image.jpg imagesphere3d 32,16
```

---

**img2ascii**

**Arguments:**

- `_charset,_analysis_scale>0,_analysis_smoothness[%]>=0,_synthesis_scale>0,_output_ascii_filename=

**Description:**

Render selected images as binary ascii art.

This command returns the corresponding the list of widths and heights (expressed as a number of characters) for each selected image.

**Default values:**

charset=[ascii charset], analysis_scale=16, analysis_smoothness=20%, synthesis_scale=16 and _output_ascii_filename=[undefined].

**Example of use:**

```
image.jpg img2ascii ,
```
**img2base64**

**Arguments:**
- \_encoding={ 0:base64 | 1:base64url }, \_store_names={ 0 | 1 }

**Description:**
Encode selected images as a base64-encoded string.

The images can be then decoded using command base642img.

**Default values:**
encoding=0 and store_names=1.

---

**img2hex**

**No arguments**

**Description:**
Return representation of last image as an hexadecimal-encoded string.

Input image must have values that are integers in [0,255].

---

**img2patches**

**Arguments:**
- patch_size>0, overlap[%]>0, boundary_conditions

**Description:**
Decompose selected 2D images into (possibly overlapping) patches and stack them along the z-axis.

overlap must be in range [0,patch_size-1].
boundary_conditions can be \{ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror \}.

**Default values:**

overlap=0 and boundary_conditions=0.

**See also:**

patches2img.

**Example of use:**

```plaintext
image.jpg img2patches 64
```

---

**img2str**

**No arguments**

**Description:**

Return the content of the selected images, as special G'MIC input strings.

---

**img2text**

**Arguments:**

- **line_separator**

**Description:**

Return text contained in a multi-line image.

**Default values:**

line_separator= .

**Preamble**

- This document is distributed under the GNU Free Documentation License, version 1.3.
inn

No arguments

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

**Example of use:**
```
image.jpg inn
```
inpaint

Arguments:

- [mask] or
- [mask],0,fast_method or
- [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}

Description:

Inpaint selected images by specified mask.

If no patch size (or 0) is specified, inpainting is done using a fast average or median algorithm. Otherwise, it used a patch-based reconstruction method, that can be very time consuming. fast_method can be {0: low-connectivity average | 1: high-connectivity average | 2: low-connectivity median | 3: high-connectivity median}.

Default values:

patch_size=0, fast_method=1, lookup_size=22, lookup_factor=0.5, lookup_increment=1, blend_size=0, blend_threshold=0, blend_decay=0.05, blend_scales=10 and is_blend_outer=1.

Examples of use:

- Example #1

```
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 #2

inpaint_flow

Arguments:

- [mask], nb_global_iter>=0, nb_local_iter>=0, dt>0, alpha>=0, sigma>=0

Description:

Apply iteration of the inpainting flow on selected images.

Default values:

nb_global_iter=10, nb_local_iter=100, dt=5, alpha=1 and sigma=3.

Example of use:

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

inpaint_holes
Arguments:

- maximal_area[%]>=0, tolerance>=0, is_high_connectivity={ 0 | 1 }

Description:

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

Default values:

maximal_area=4, tolerance=0 and is_high_connectivity=0.

Example of use:

```
image.jpg noise 5%,2 +inpaint_holes 8,40
```

---

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 }

Description:

Inpaint selected images by specified binary mask, using a multi-scale matchpatch algorithm.

Default values:

nb_scales=0, patch_size=9, nb_iterations_per_scale=10, blend_size=5, allow_outer_blending=1 and is_already_initialized=0.

Example of use:

```
image.jpg 100%,100% ellipse[-1] 30%,30%,40,30,0,1,255 +inpaint_matchpatch[0] [1]
```
inpaint_morpho

Arguments:

- [mask]

Description:

Inpaint selected images by specified mask using morphological operators.

Example of use:

```
image.jpg 100%,100% ellipse[-1] 30%,30%,40,30,0,1,255
+inpaint_morpho[0] [1]
```
inpaint_pde

Arguments:

- `[mask], nb_scales[%, diffusion_type={ 0: isotropic | 1: Delaunay-guided | 2: edge-guided | 3: mask-guided }, diffusion_iter>=0` 

Description:

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

Argument `nb_scales` sets the number of scales used in the multi-scale resolution scheme.
- When the `%` qualifier is used for `nb_scales`, the number of used scales is relative to `nb_scales_max = ceil(log2(max(w,h,d)))`.
- When `nb_scales<0`, it determines the minimum image size encountered at the lowest scale.

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=-9, diffusion_type=1` and `diffusion_iter=20`.

Example of use:

```
image.jpg 100%,100% ellipse[-1] 30%,30%,40,30,0,1,255 +inpaint_pde[0] [1]
```
input

Arguments:

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

Description:

Insert a new image taken from a filename or from a copy of an existing image [index], 
or insert new image with specified dimensions and values. Single quotes may be omitted in 
formula. Specifying argument 0 inserts an empty image.

(equivalent to shortcut command i).

Default values:

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

This command has a tutorial page.

Examples of use:

- Example #1

  input image.jpg
• Example #2

```plaintext
input (1,2,3;4,5,6;7,8,9^9,8,7;6,5,4;3,2,1)
```

• Example #3

```plaintext
image.jpg (1,2,3;4,5,6;7,8,9) (255^128^64) 400,400,1,3,'(x>w/2?
x:y)*c'
```
input_565

Arguments:
• filename, width>0, height>0, reverse_endianness={ 0 | 1 }

Description:
Insert image data from a raw RGB-565 file, at the end of the list.

Default values:
reverse_endianness=0.

input_bytes

Arguments:
• filename

Description:
Input specified filename as a 1D array of bytes.
(equivalent to shortcut command ib).

input_cached

Arguments:
• "basename.ext", try_downloading_from_gmic_server={ 0 | 1 }

Description:
Input specified filename, assumed to be stored in one of the G'MIC resource folder.
If file not found and try_downloading=1, file is downloaded from the G'MIC server and stored in the ${-path_cache} folder.

Default values:
try_downloading_from_gmic_server=1.

input_csv
Arguments:

- "filename", _read_data_as={ 0:numbers | 1:strings | _variable_name }

Description:

Insert number of string array from specified .csv file.

If variable_name is provided, the string of each cell is stored in a numbered variable _variable_name_x_y, where x and y are the indices of the cell column and row respectively (starting from 0).
Otherwise, a WxH image is inserted at the end of the list, with each vector-valued pixel I(x,y) encoding the number or the string of each cell.
This command returns the W,H dimension of the read array, as the status.

Default values:

read_data_as=1.

input_cube

Arguments:

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

Description:

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

Default values:

convert_1d_cluts_to_3d=1.

input_flo

Arguments:

- "filename"

Description:

Insert optical flow data from a .flo filename (vision.middlebury.edu file format).

input_glob

Arguments:

- pattern
Description:
Insert new images from several filenames that match the specified glob pattern.
(equivalent to shortcut command `ig`).

input_gpl

Arguments:
• `filename`

Description:
Input specified filename as a .gpl palette data file.

input_obj

Arguments:
• `filename`

Description:
Input specified 3D mesh from a .obj Wavefront file.

input_text

Arguments:
• `filename`

Description:
Input specified text-data filename as a new image.
(equivalent to shortcut command `it`).

inrange

Arguments:
• `min[%],max[%],_include_min_boundary={ 0:no | 1:yes },_include_max_boundary={ 0:no | 1:yes }`

Description:
Detect pixels whose values are in specified range \([\text{min, max}]\), in selected images.

(equivalent to shortcut command \texttt{ir}).

**Default values:**

include\_min\_boundary=include\_max\_boundary=1.

**Example of use:**

```
image.jpg +inrange 25\%,75%
```

---

**int2rgb**

**No arguments**

**Description:**

Convert color representation of selected images from INT24 to RGB.

---

**invert**

**Built-in command**

**Arguments:**

- use\_LU={ 0:SVD | 1:LU }, _lambda>=0

**Description:**

Inverse selected matrices (or compute Moore-Penrose pseudoinverse for non-square matrices).

SVD solver is slower but more precise than LU.
Lambda is used only in the Moore-Penrose pseudoinverse, by estimating \(A^t.(A^t.A + \lambda \cdot \text{Id})^{-1}\).

**Default values:**

use\_LU=0 and lambda=0.

**Example of use:**
ipremula

No arguments

Description:
Convert selected images with premultiplied alpha colors to normal colors.

See also:
premula.

is_change

Arguments:
• value={ 0: false | 1: true }

Description:
Set or unset the is_change flag associated to the image list.
This flag tells the interpreter whether or not the image list should be displayed when the pipeline ends.

Default values:
value=1.

is_ext

Arguments:
• filename, extension
is_half

No arguments

Description:
Return 1 if the type of image pixels is limited to half-float.

is_image_arg

Arguments:
• string

Description:
Return 1 if specified string looks like [ind].

is_macos

No arguments

Description:
Return 1 if current computer OS is Darwin (MacOS), 0 otherwise.

is_mesh3d

No arguments

Description:
Return 1 if all of the selected images are 3D meshes, 0 otherwise.

is_pattern

Arguments:
• string

Description:
is_varname

Arguments:
- string

Description:
Return 1 if specified string can be considered as a valid variable name.

is_videofilename

Arguments:
- filename

Description:
Return 1 if extension of specified filename is typical from video files.

is_windows

No arguments

Description:
Return 1 if current computer OS is Windows, 0 otherwise.

isoline3d

Arguments:
- isovalue[%] or
  - 'formula',value,_x0,_y0,_x1,_y1,_size_x>0[%],_size_y>0[%]

Description:
Extract 3D isolines with specified value from selected images or from specified formula.

Default values:
x0=y0=-3, x1=y1=3 and size_x=size_y=256.

Examples of use:
- Example #1
Example #2

```
isoline3d 'X=x-w/2;Y=y-h/2;(X^2+Y^2)%20',10,-10,-10,10,10
```

---

**isophotes**

**Arguments:**

- **nb_levels>0**

**Description:**

Render isophotes of selected images on a transparent background.

**Default values:**

nb_levels=64

**Example of use:**

```
image.jpg blur 2 isophotes 6 dilate_circ 5 display_rgba
```
**isosurface3d**

**Arguments:**

- `isovalue[%]` or
- `'formula', value, x0, y0, z0, x1, y1, z1, size_x>0[%], size_y>0[%], size_z>0[%]`

**Description:**

Extract 3D isosurfaces with specified value from selected images or from specified formula.

**Default values:**

x0=y0=z0=-3, x1=y1=z1=3 and size_x=size_y=size_z=32.

**Examples of use:**

- **Example #1**

```
image.jpg rescale2d ,128 luminance threshold 50% expand z,2 blur 1
isosurface3d 50% mul3d 1,1,30
```

- **Example #2**

```
isosurface3d 'x^2+y^2+abs(z)^abs(4*cos(x*y*z*3))', 3
```
jzazbz2rgb

Arguments:

- **illuminant**={ 0:D50 | 1:D65 | 2:E } or
- **(no arg)**

Description:

Convert color representation of selected images from RGB to Jzazbz.

Default values:

illuminant=2.

jzazbz2xyz

No arguments

Description:

Convert color representation of selected images from RGB to XYZ.

kaleidoscope

Arguments:

- **center_x[%]**, **center_y[%]**, **radius**, **angle**, **boundary_conditions**={ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror }

Description:

Create kaleidoscope effect from selected images.

Default values:

center_x=center_y=50%, radius=100, angle=30 and boundary_conditions=3.
Example of use:

| image.jpg kaleidoscope |

keep

No arguments

Description:
Keep only selected images.

(equivalent to shortcut command k).

Examples of use:

• Example #1

| image.jpg split x keep[0-50%:2] append x |

• Example #2

| image.jpg split x keep[^30%-70%] append x |
**keep_named**

**Arguments:**
- "name1", "name2", ...

**Description:**
Keep all images with specified names from the list of images.
Remove all images if no images with those names exist.

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

---

**kuwahara**

**Arguments:**
- `size>0`

**Description:**
Apply Kuwahara filter of specified size on selected images.

**Example of use:**

```
image.jpg kuwahara 9
```
laar

No arguments

Description:
Extract the largest axis-aligned rectangle in non-zero areas of selected images.
Rectangle coordinates are returned in status, as a sequence of numbers x0,y0,x1,y1.

Example of use:
```
shape_cupid 256 coords=${-laar} normalize 0,255 to_rgb rectangle $coords,0.5,0,128,0
```

lab2lch

No arguments

Description:
Convert color representation of selected images from Lab to Lch.

lab2rgb

Arguments:
- illuminant= { 0:D50, 1:D65, 2:E } or
- (no arg)

Description:
Convert color representation of selected images from Lab to RGB.

Default values:
illuminant=2.

Example of use:

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

lab2srgb

Arguments:

- `illuminant={ 0:D50 | 1:D65 | 2:E }` or `no arg`

Description:

Convert color representation of selected images from Lab to sRGB.

Default values:

illuminant=2.

Example of use:

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

lab2xyz
Arguments:

- illuminant={ 0:D50 | 1:D65 | 2:E } or (no arg)

Description:
Convert color representation of selected images from Lab to XYZ.

Default values:
illuminant=2.

lab82rgb

Arguments:

- illuminant={ 0:D50 | 1:D65 | 2:E } or (no arg)

Description:
Convert color representation of selected images from Lab8 to RGB.

Default values:
illuminant=2.

lab82srgb

Arguments:

- illuminant={ 0:D50 | 1:D65 | 2:E } or (no arg)

Description:
Convert color representation of selected images from Lab8 to sRGB.

Default values:
illuminant=2.

Example of use:

(50,50;50,50^-3,3;-3,3^-3,-3;3,3) resize 400,400,1,3,3 lab2rgb
**label**

**Arguments:**

- \(_\text{tolerance} \geq 0, \text{is\_high\_connectivity} = \{0, 1\}, \text{is\_L2\_norm} = \{0, 1\}\)

**Description:**

Label connected components in selected images.

If \(\text{is\_L2\_norm} = 1\), tolerances are compared against L2-norm, otherwise L1-norm is used.

**Default values:**

tolerance=0, \(\text{is\_high\_connectivity}\)=0 and \(\text{is\_L2\_norm}\)=1.

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

**Examples of use:**

- **Example #1**

  ```
  image.jpg luminance threshold 60\% label normalize 0,255 map 0
  ```

- **Example #2**

  ```
  400,400 set 1,50\%,50\% distance 1 mod 16 threshold 8 label mod 255 map 2
  ```
**label3d**

**Arguments:**

- "text", font_height>=0, _opacity, _color1,...

**Description:**

Generate 3D text label.

**Default values:**

font_height=13, opacity=1 and color=255,255,255.

---

**label_fg**

**Arguments:**

- tolerance>=0, is_high_connectivity={ 0 | 1 }, _is_L2_norm={ 0 | 1 }

**Description:**

Label connected components for non-zero values (foreground) in selected images.

Similar to label except that 0-valued pixels are not labeled.

If is_L2_norm=1, tolerances are compared against L2-norm, otherwise L1-norm is used.

**Default values:**

is_high_connectivity=0.

---

**label_points3d**

**Arguments:**

- _label_size>0, _opacity

---
**Description:**
Add a numbered label to all vertices of selected 3D objects.

**Default values:**
label_size=13 and opacity=0.8.

**Example of use:**
```
torus3d 100,40,6,6 label_points3d 23,1 mode3d 1
```

---

**laplacian**

**No arguments**

**Description:**
Compute Laplacian of selected images.

**Example of use:**
```
image.jpg laplacian
```

---

**lathe3d**
Arguments:

- resolution > 0, smoothness [%] ≥ 0, max_angle ≥ 0

Description:

Generate 3D object from selected binary XY-profiles.

Default values:

resolution=128, smoothness=0.5% and max_angle=361.

Example of use:

```
300,300 rand -1,1 blur 40 sign normalize 0,255 lathe3d ,
```

---

**lch2lab**

No arguments

Description:

Convert color representation of selected images from Lch to Lab.

---

**lch2rgb**

Arguments:

- illuminant={ 0:D50 | 1:D65 | 2:E } or (no arg)

Description:

Convert color representation of selected images from Lch to RGB.

Default values:

illuminant=2.
**lch82rgb**

**Arguments:**
- `illuminant={ 0:D50 | 1:D65 | 2:E }` or
- `(no arg)`

**Description:**
Convert color representation of selected images from Lch8 to RGB.

**Default values:**
`illuminant=2`.

---

**le**

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

**Description:**
Compute the boolean 'less or equal than' of selected images with specified value, image or mathematical expression, or compute the boolean 'less or equal than' of selected images.

(equivalent to shortcut command `<=`).

**Examples of use:**

- **Example #1**
  
  ```
  image.jpg le {ia}
  ```

- **Example #2**
lic

Arguments:
• _amplitude>0,_channels>0

Description:
Render LIC representation of selected vector fields.

Default values:
amplitude=30 and channels=1.

Example of use:

```
400,400,1,2,'!c?x-w/2:y-h/2' +lic 200,3 quiver[-2] [-2],10,1,1,1,255
```

light3d

Arguments:
• position_x,position_y,position_z or
• [texture] or
**Description:**
Set the light coordinates or the light texture for 3D rendering.
(equivalent to shortcut command `l3d`).
(no arg) resets the 3D light to default.

**Example of use:**
```plaintext
torus3d 100,30 double3d 0 specs3d 1.2 repeat 5 { light3d {$>*100},0,-300 +snapshot3d[0] 400 } remove[0]
```

**Arguments:**
- `density>0`, `darkness>=0`, `lightness>=0`

**Description:**
Add light patches to selected images.

**Default values:**
density=10, darkness=0.9 and lightness=1.7.

**Example of use:**
light_relief

Arguments:

- ambient_light, specular_lightness, specular_size, darkness, light_smoothness, xl, yl, zl, zs
  0 | 1 }

Description:

Apply relief light to selected images.

Default values(s): ambient_light=0.3, specular_lightness=0.5, specular_size=0.2, darkness=0, xl=0.2, yl=zl=0.5, zscale=1, opacity=1 and opacity_is_heightmap=0.

Example of use:

```
image.jpg blur 2 light_relief 0.3,4,0.1,0
```

lightness

No arguments

Description:
Compute lightness of selected sRGB images.

**Example of use:**
```
image.jpg +lightness
```

**lightrays**

**Arguments:**
- `100<=density<=0,center_x[%,center_y[%,ray_length]>=0,ray_attenuation>=0`

**Description:**
Generate ray lights from the edges of selected images.

**Default values:**
density=50%, center_x=50%, center_y=50%, ray_length=0.9 and ray_attenuation=0.5.

**Example of use:**
```
image.jpg +lightrays , + cut 0,255
```

**line**

Built-in command
**Arguments:**

- \(x_0\%\), \(y_0\%\), \(x_1\%\), \(y_1\%\), _opacity_, _pattern_, _color1_, ...

**Description:**

Draw specified colored line on selected images.

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

**Default values:**

Opacity=1, pattern=(undefined) and color1=0.

**Example of use:**

```plaintext
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
```

![Image](image.jpg)

---

**line3d**

**Arguments:**

- \(x_0\), \(y_0\), \(z_0\), \(x_1\), \(y_1\), \(z_1\)

**Description:**

Input 3D line at specified coordinates.

**Example of use:**

```plaintext
repeat 100 { a:=$>*pi/50 line3d 0,0,0,\{cos(3*$a}\},\{sin(2*$a)\},0 color3d. ${-rgb} } add3d
```
**line_aa**

**Arguments:**

- \( x0[:\%], y0[:\%], x1[:\%], y1[:\%], \_opacity, \_color1, \ldots \)

**Description:**

Draw specified antialiased colored line on selected images.

**Default values:**

opacity=1 and color1=0.

**Example of use:**

```plaintext
512,512,1,3 repeat 100 line_aa {v([w,h,w,h])-1},1,${-RGB} done
```

**linearize_tiles**

**Arguments:**

- \( M>0, N>0 \)

**Description:**

Linearize MxN tiles on selected images.
Default values:
N=M.

Example of use:

```
image.jpg +linearize_tiles 16
```

![Example of use](image.jpg)

---

**lines3d**

**Arguments:**

- `length>=0`

**Description:**

Convert specified 3D objects to sets of 3D horizontal segments with specified length.

**Default values:**

`length=1.`

**Example of use:**

```
torus3d 100,40 +lines3d 20
```

![Example of use](torus3d_100_40_lines3d_20.png)
linify

Arguments:
• \(0\leq \text{density}\leq 100, \text{spreading}\geq 0, \text{resolution}\%>0, \text{line_opacity}\geq 0, \text{line_precision}>0, \text{mode}=\{0: \text{subtractive} | 1: \text{additive}\}\)

Description:
Apply linify effect on selected images.

The algorithm is inspired from the one described on the webpage http://linify.me/about.

Default values:
density=50, spreading=2, resolution=40%, line_opacity=10, line_precision=24 and mode=0.

Example of use:

```
image.jpg linify 60
```

lissajous3d

Arguments:
• resolution>1,a,A,b,B,c,C

Description:
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 of use:

```
lissajous3d ,
```
local

Built-in command

No arguments

Description:
Start a local...[onfail]...done block, with selected images.

(equivalent to shortcut command \l). This command has a tutorial page.

Examples of use:

• Example #1

  image.jpg local[] 300,300,1,3 rand[0] 0,255 blur 4 sharpen 1000 done

  ![Example #1](image1.jpg)

• Example #2

  image.jpg +local repeat 3 { deform 20 } done

  ![Example #2](image2.jpg)
lof

Arguments:
• feature

Description:
Return the list of specified features (separated by commas) for each selected images.

log

Built-in command

No arguments

Description:
Compute the pointwise base-e logarithm of selected images.

Examples of use:
• Example #1

```plaintext
image.jpg +add 1 log[-1]
```

• Example #2

```plaintext
300,1,1,1,'7*x/w+u' +log display_graph 400,300
```
**log10**

**Built-in command**

**No arguments**

**Description:**

Compute the pointwise base-10 logarithm of selected images.

**Examples of use:**

- **Example #1**

  ```
  image.jpg +add 1 log10[-1]
  ```

- **Example #2**

  ```
  300,1,1,1,'7*x/w+u' +log10 display_graph 400,300
  ```
**log2**

*Built-in command*

No arguments

**Description:**

Compute the pointwise base-2 logarithm of selected images

**Examples of use:**

• **Example #1**

```
image.jpg +add 1 log2[-1]
```

• **Example #2**

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

- \_width>0, _height>0

Description:

Input random image of specified size, retrieved from Internet.

Default values:

width=height=800.

---

lt

Built-in command

Arguments:

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

Description:

Compute the boolean less than of selected images with specified value, image or mathematical expression, or compute the boolean less than of selected images.

(equivalent to shortcut command <).

Examples of use:

• Example #1

  image.jpg lt {ia}

• Example #2

  ![Example Image](image.jpg)

  ![Example Image](image.jpg)
luminance

No arguments

Description:
Compute luminance of selected sRGB images.

This command has a tutorial page.

Example of use:

image.jpg +luminance

lut_contrast

Arguments:

- nb_colors>1, min_rgb_value

Description:
Generate a RGB colormap where consecutive colors have high contrast.

This function performs a specific score maximization to generate the result, so
it may take some time when nb_colors is high.

**Default values:**

nb_colors=256 and min_rgb_value=64.

---

**mad**

**No arguments**

**Description:**

Return the MAD (Maximum Absolute Deviation) of the last selected image.

The MAD is defined as \( \text{MAD} = \text{med}_j |x_i - \text{med}_j(x_j)| \)

---

**mandelbrot**

**Arguments:**

- \( z0r, z0i, z1r, z1i, \_iteration\_max\geq0, \_is\_julia=\{0 \mid 1\}, c0r, c0i, \_opacity \)

**Description:**

Draw mandelbrot/julia fractal on selected images.

**Default values:**

iteration_max=100, is_julia=0, c0r=c0i=0 and opacity=1.

**Example of use:**

```
400,400 mandelbrot -2.5,-2,2,2,1024 map 0 +blur 2 elevation3d[-1]
-0.2
```

---

**map**

Built-in command
Arguments:

- [palette],_boundary_conditions or
- palette_name,_boundary_conditions

Description:

Map specified vector-valued palette to selected indexed images.

Each output image has M*N channels, where M and N are the numbers of channels of, respectively, the corresponding input image and the palette image.

palette_name can be { default | hsv | lines | hot | cool | jet | flag | cube | rainbow | algae | amp | balance | curl | deep | delta | dense | diff | gray | haline | ice | matter | oxy | phase | rain | solar | speed | tarn | tempo | thermal | topo | turbid | aurora | hocuspocus | srb2 | uzebox }

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

Default values:

boundary_conditions=0.

This command has a [tutorial page].

Examples of use:

- Example #1

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

- Example #2

  ```
  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]
  ```
map_clut

**Arguments:**

- `[clut] | "clut_name"

**Description:**

Map specified RGB color LUT to selected images.

**Example of use:**

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

map_sphere
Arguments:

- \texttt{width>0, height>0, radius, dilation>0, fading>=0, fading_power>=0}

Description:

Map selected images on a sphere.

Default values:

\texttt{width=height=512, radius=100, dilation=0.5, fading=0 and fading_power=0.5.}

Example of use:

\begin{verbatim}
image.jpg map_sphere ,
\end{verbatim}

\begin{center}
\includegraphics[width=0.5\textwidth]{sphere.png}
\end{center}

---

\textbf{map_sprites}

Arguments:

- \texttt{nbSprites\geq1, allow_rotation=\{ 0:none | 1:90 deg. | 2:180 deg. \}}

Description:

Map set of sprites (defined as the \texttt{nbSprites} latest images of the selection) to other selected images,

according to the luminosity of their pixel values.

Example of use:

\begin{verbatim}
image.jpg rescale2d ,48 repeat 16 ball {8+2*$>},$\{-rgb\} mul[-1] \{(1+ $>)/16\} done map_sprites 16
\end{verbatim}
map_tones

Arguments:

- \_threshold\geq0, \_gamma\geq0, \_smoothness\geq0, \_nb\_iter\geq0

Description:

Apply tone mapping operator on selected images, based on Poisson equation.

Default values:

threshold=0.1, gamma=0.8, smoothness=0.5 and nb_iter=30.

Example of use:

```
image.jpg +map_tones ,
```

map_tones_fast

Arguments:

- \_radius[\%]\geq0, \_power\geq0

Description:

Apply fast tone mapping operator on selected images.
Default values:

radius=3% and power=0.3.

Example of use:

```
image.jpg +map_tones_fast ,
```

marble

Arguments:

- image_weight, pattern_weight, angle, amplitude, sharpness>=0, anisotropy>=0, alpha, sigma, cut_low=cut_high=0.

Description:

Render marble like pattern on selected images.

Default values:

image_weight=0.2, pattern_weight=0.1, angle=45, amplitude=0, sharpness=0.4 and anisotropy=0.8,

alpha=0.6, sigma=1.1 and cut_low=cut_high=0.

Example of use:

```
image.jpg +marble ,
```
match_histogram

Arguments:

- \([reference\_image],_nb\_levels>0,_color\_channels\]

Description:

Transfer histogram of the specified reference image to selected images.

Argument color channels is the same as with command apply_channels.

Default values:

nb_levels=256 and color_channels=all.

Example of use:

```
image.jpg 100,100,1,3,"u([256,200,100])" +match_histogram[0] [1]
```

match_icp

Arguments:

- \([reference\_image],_precision>0,_transformation\_variable\]

Description:


Transform selected set of d-dimensional vectors to match specified set of reference vectors, using ICP (Iterative Closest Point) algorithm.


Default values:

precision=1e-2 and transformation_variable=(undefined).

sample lena,earth +match_icp[0] [1]

match_pca

Arguments:

• [reference_image],_color_channels

Description:

Transfer mean and covariance matrix of specified vector-valued reference image to selected images.

Argument color channels is the same as with command apply_channels.

Default values:

color_channels=all.

Example of use:

sample lena,earth +match_pca[0] [1]

match_rgb

Arguments:

• [target],_gamma>=0,_regularization>=0,_luminosity_constraints>=0,_rgb_resolution>=0,_is_const

Description:
Transfer colors from selected source images to selected reference image (given as argument).

gamma determines the importance of color occurrences in the matching process (0:none to 1:huge).
regularization determines the number of guided filter iterations to remove quantization effects.
luminosity_constraints tells if luminosity constraints must be applied on non-confident matched colors.
is_constraints tells if additional hard color constraints must be set (opens an interactive window).

**Default values:**
gamma=0.3, regularization=8, luminosity_constraints=0.1, rgb_resolution=64 and is_constraints=0.

**Example of use:**

```
sample pencils, wall +match_rgb[0] [1], 0, 0.01
```

---

**matchpatch**

**Arguments:**

- `[patch_image],patch_width>=1, patch_height>=1, patch_depth>=1, nb_iterations>=0, 0 | 1`, `[guide]`

**Description:**

Estimate correspondence map between selected images and specified patch image, using a patch-matching algorithm.
Each pixel of the returned correspondence map gives the location (p,q) of the closest patch in
the specified patch image. If output_score=1, the third channel also gives the corresponding matching score for each patch as well. If patch_penalization is >=0, SSD is penalized with patch occurrences. If patch_penalization is <0, SSD is inf-penalized when distance between patches are less than -patch_penalization.

**Default values:**

patch_height=patch_width, patch_depth=1, nb_iterations=5, nb_randoms=5, patch_penalization=0, output_score=0 and guide=(undefined).

**Example of use:**

```
image.jpg sample colorful +matchpatch[0] [1],3 +warp[-2] [-1],0
```

---

**math_lib**

**No arguments**

**Description:**

Return string that defines a set of several useful macros for the embedded math evaluator.

---

**max**

**Arguments:**

- `value[%]` or
Description:

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

Examples of use:

• Example #1

```
image.jpg +mirror x max
```

• Example #2

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

max_d

No arguments

Description:

Return the maximal depth between selected images.
**max_h**

No arguments

**Description:**
Return the maximal height between selected images.

---

**max_patch**

**Arguments:**
- \(\text{patch\_size} \geq 1\)

**Description:**
Return locations of maximal values in local patch-based neighborhood of given size for selected images.

**Default values:**
\(\text{patch\_size}=16\).

**Example of use:**
```
image.jpg norm +max_patch 16
```

---

**max_s**

No arguments

**Description:**
Return the maximal spectrum between selected images.

---

**max_w**
**max_wh**

No arguments

**Description:**

Return the maximal width between selected images.

---

**max_whd**

No arguments

**Description:**

Return the maximal wxhxd size of selected images.

---

**max_whds**

No arguments

**Description:**

Return the maximal wxhxd xs size of selected images.

---

**maxabs**

**Arguments:**

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

**Description:**

Compute the maxabs between selected images and specified value, image or mathematical expression, or compute the pointwise maxabs between selected images.
Arguments:
- \_width>0, \_height>0, \_cell_size>0

Description:
Input maze with specified size.

Example of use:
```
maze 30,20 negate normalize 0,255
```

maze_mask

Arguments:
- \_cellsize>0

Description:
Input maze according to size and shape of selected mask images.
Mask may contain disconnected shapes.

Example of use:
```
0 text "G'MIC",0,0,53,1,1 dilate 3 autocrop 0 frame xy,1,0 maze_mask
8 dilate 3 negate mul 255
```

mdiv

Built-in command
Arguments:

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

Description:
Compute the matrix division of selected matrices/vectors by specified value, image or mathematical expression, or compute the matrix division of selected images.

(equivalent to shortcut command m/).

meancurvature_flow

Arguments:

- _nb_iter>=0, _dt, _keep_sequence={0 | 1}

Description:
Apply iterations of the mean curvature flow on selected images.

Default values:

_nb_iter=10, _dt=30 and _keep_sequence=0.

Example of use:

```
image.jpg +meancurvature_flow 20
```

med

No arguments

Description:
Compute the median of selected images.
Example of use:

```
image.jpg sample lena,lion,square +med
```

---

**median**

**Built-in command**

**Arguments:**

- `size>=0, _threshold>0`

**Description:**

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

**Example of use:**

```
image.jpg +median 5
```
median_files

Arguments:

- "filename_pattern", _first_frame>=0, _last_frame={ >=0 | -1=last }, _frame_step>=1, _frame_rows[%]>=1, _is_fast_approximation={ 0 | 1 }

Description:

Compute the median frame of specified input image files, in a streamed way.

If a display window is opened, rendered frame is displayed in it during processing.

Default values:

first_frame=0, last_frame=-1, frame_step=1, frame_rows=20% and is_fast_approximation=0.

median_vectors

No arguments

Description:

Return the median vector value of the last selected image (median computed channel by channel)

median_video

Arguments:

- video_filename, _first_frame>=0, _last_frame={ >=0 | -1=last }, _frame_step>=1, _frame_rows[%]>=1, _is_fast_approximation={ 0 | 1 }

Description:

Compute the median of all frames of an input video file, in a streamed way.

If a display window is opened, rendered frame is displayed in it during processing.
This command requires features from the OpenCV library (not enabled in G'MIC by default).

**Default values:**

first\_frame=0, last\_frame=-1, frame\_step=1, frame\_rows=100% and is\_fast\_approximation=1.

---

### meigen

**Arguments:**

- \( m \geq 1 \)

**Description:**


**Example of use:**

```
(1,0,0;0,2,0;0,0,3) +meigen 3
```

---

### merge\_alpha

**No arguments**

**Description:**

Merge selected alpha detail scales into a single image. Alpha detail scales have been obtained with command \texttt{split\_alpha}.

---

### min

**Built-in command**
Arguments:

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

Description:

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

Examples of use:

- **Example #1**

  ```
  image.jpg +mirror x min
  ```

- **Example #2**

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

---

**min_d**

No arguments

Description:

Return the minimal depth between selected images.
**min_h**

No arguments

**Description:**
Return the minimal height between selected images.

---

**min_patch**

**Arguments:**
- `patch_size`\(\geq 1\)

**Description:**
Return locations of minimal values in local patch-based neighborhood of given size for selected images.

**Default values:**
patch_size=16.

**Example of use:**

```
image.jpg norm +min_patch 16
```

---

**min_s**

No arguments

**Description:**
Return the minimal s size of selected images.
**min_w**

No arguments

**Description:**
Return the minimal width between selected images.

---

**min_wh**

No arguments

**Description:**
Return the minimal wxh size of selected images.

---

**min_whd**

No arguments

**Description:**
Return the minimal wxhxd size of selected images.

---

**min_whds**

No arguments

**Description:**
Return the minimal wxhxdxs size of selected images.

---

**minabs**

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

**Description:**
Compute the minabs between selected images and specified value, image or mathematical expression, or compute the pointwise minabs between selected images.
**minimal_path**

**Arguments:**

- \( x_0[\%] \geq 0, y_0[\%] \geq 0, z_0[\%] \geq 0, x_1[\%] \geq 0, y_1[\%] \geq 0, z_1[\%] \geq 0, \_is\_high\_connectivity = \{ 0 | 1 \} \)

**Description:**
Compute minimal path between two points on selected potential maps.

**Default values:**

- is_high_connectivity = 0.

**Example of use:**

```
image.jpg +gradient_norm fill[-1] 1/(1+i) minimal_path[-1] 0,0,0,100\%,100\%,0 pointcloud[-1] 0 *[1] 280 to_rgb[-1] ri[-1] [-2],0 or
```

---

**mirror**

**Built-in command**

**Arguments:**

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

**Description:**
Mirror selected images along specified axes.

**Examples of use:**

- **Example #1**

```
image.jpg +mirror y +mirror[0] c
```
• Example #2

```plaintext
image.jpg +mirror x +mirror y append_tiles 2,2
```

---

**mix_channels**

**Arguments:**

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

**Description:**

Apply specified matrix to channels of selected images.

**Example of use:**

```plaintext
image.jpg +mix_channels (0,1,0;1,0,0;0,0,1)
```
mix_rgb

Arguments:

• $a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}, a_{31}, a_{32}, a_{33}$

Description:

Apply 3x3 specified matrix to RGB colors of selected images.

Default values:

$a_{11}=1, a_{12}=a_{13}=a_{21}=0, a_{22}=1, a_{23}=a_{31}=a_{32}=0$ and $a_{33}=1$.

This command has a tutorial page.

Example of use:

```
image.jpg +mix_rgb 0,1,0,1,0,0,0,0,1
```

mmul

Built-in command

Arguments:

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

**Description:**

Compute the matrix right multiplication of selected matrices/vectors by specified value, image or mathematical expression, or compute the matrix right multiplication of selected images.

If the right-hand side image is vector-valued, this command multiplies each vector-valued pixels by the specified left-hand matrix.

*(equivalent to shortcut command \(m^*\)).*

**Examples of use:**

• **Example #1**

\[
\begin{pmatrix}
0,1,0;0,0,1;1,0,0
\end{pmatrix}
(1;2;3)
\]

• **Example #2**

\[
\begin{pmatrix}
0,1,0;0,0,1;1,0,0
\end{pmatrix}
image.jpg
\]
**mod**

**Arguments:**

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

**Description:**

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

*(equivalent to shortcut command %).*

**Examples of use:**

- **Example #1**

  ```plaintext
  image.jpg +mirror x n. 1,255 round. mod
  ```

- **Example #2**

  ```plaintext
  image.jpg mod 'R=((x/w-0.5)^2+(y/h-0.5)^2)^0.5;255*R'
  ```
mode3d

Arguments:

- _mode

Description:

Set static 3D rendering mode.

(equivalent to shortcut command m3d).

mode can be 

Bounding-box mode (mode==1) is active only for the interactive 3D viewer.

Default values:

mode=4.

Example of use:

```plaintext
(0,1,2,3,4,5) double3d 0 repeat w { torus3d 100,30 rotate3d[-1] 1,1,0,60 mode3d {0,@$>} snapshot3d[-1] 300 } remove[0]
```

moded3d

Arguments:

- _mode
Description:
Set dynamic 3D rendering mode for interactive 3D viewer.
(equivalent to shortcut command `md3d`).

mode can be \{ -1:bounding-box | 0:dots | 1:wireframe | 2:flat | 3:flat-shaded | 4:gouraud-shaded | 5:phong-shaded \}.

Default values:
mode=-1.

---

montage

Arguments:
- "layout_code", montage_mode={ 0<=centering<=1 | 2<=scale+2<=3 }, output_mode={ 0:single layer | 1:multiple layers }, "processing_command"

Description:
Create a single image montage from selected images, according to specified layout code:
- X to assemble all images using an automatically estimated layout.
- H to assemble all images horizontally.
- V to assemble all images vertically.
- A to assemble all images as an horizontal array.
- B to assemble all images as a vertical array.
- Ha:b to assemble two blocks a and b horizontally.
- Va:b to assemble two blocks a and b vertically.
- Ra to rotate a block a by 90 deg. (RRa for 180 deg. and RRRa for 270 deg.).
- Ma to mirror a block a along the X-axis (MRRa for the Y-axis).

A block a can be an image index (treated periodically) or a nested layout expression Hb:c,Vb:c,Rb or Mb itself.
For example, layout code H0:V1:2 creates an image where image [0] is on the left, and images [1] and [2] vertically packed on the right.

Default values:
layout_code=X, montage_mode=2, output_mode='0' and processing_command="".

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

**Arguments:**
- \texttt{nb\_inner\_frames}\geq 1, \texttt{smoothness}\geq 0, \texttt{precision}\geq 0

**Description:**
Create morphing sequence between selected images.

**Default values:**
smoothness=0.1 and precision=4.

**Example of use:**
```
image.jpg +rotate 20,1,1,50%,50% morph 9
```
**morph_files**

**Arguments:**

- "filename_pattern", nb_inner_frames>=0, smoothness>=0, precision>=0, first_frame: >=0 | -1=last }, frame_step>=1, output_filename

**Description:**

Generate a temporal morphing from specified input image files, in a streamed way.

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

**Default values:**

nb_inner_frames=10, smoothness=0.1, precision=4, first_frame=0, last_frame=-1, frame_step=1 and output_filename=(undefined).

---

**morph_rbf**

**Arguments:**

- nb_inner_frames>=1,xs0[%],ys0[%],xt0[%],yt0[%],...,xsN[%],ysN[%],xtN[%],ytN[%]

**Description:**

Create morphing sequence between selected images, using RBF-based interpolation.

Each argument (xsk,ysk)-(xtk,ylk) corresponds to the coordinates of a keypoint respectively on the source and target images. The set of all keypoints define the overall image deformation.

---

**morph_video**

**Arguments:**
video_filename, nb_inner_frames>0, smoothness>=0, precision>=0, first_frame>=0, >=0 | -1=last }, frame_step>=1, output_filename

Description:

Generate a temporal morphing from specified input video file, in a streamed way.

If a display window is opened, rendered frames are displayed in it during processing.

The output filename may have extension .avi or .mp4 (saved as a video), or any other usual image file extension (saved as a sequence of images).

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

Default values:

nb_inner_frames=10, smoothness=0.1, precision=4, first_frame=0, last_frame=-1, frame_step=1 and output_filename=(undefined).

---

mosaic

Arguments:

• 0<=density<=100

Description:

Create random mosaic from selected images.

Default values:

density=30.

Example of use:

```
image.jpg mosaic , +fill "I!=J(1) || I!=J(0,1)?[0,0,0]:I"
```

---

move

Built-in command

Arguments:
• **position[%]**

**Description:**

Move selected images at specified position.

Images are actually inserted between current positions position-1 and position.

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

**Examples of use:**

• **Example #1**

```plaintext
image.jpg split x,3 move[1] 0
```

• **Example #2**

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

---

**mproj**

**Built-in command**

**Arguments:**

• `[dictionary],_method,_max_iter={ 0:auto | >0 },_max_residual>=0`

**Description:**

Find best matching projection of selected matrices onto the span of an over-complete
dictionary D, using the orthogonal projection or Matching Pursuit algorithm. Selected images are 2D-matrices in which each column represents a signal to project. [dictionary] is a matrix in which each column is an element of the dictionary D. method tells what projection algorithm must be applied. It can be:
- 0 = orthogonal projection (least-squares solution using LU-based solver).
- 1 = matching pursuit.
- 2 = matching pursuit, with a single orthogonal projection step at the end.
- >=3 = orthogonal matching pursuit where an orthogonal projection step is performed every method-2 iterations.
max_iter sets the max number of iterations processed for each signal. If set to 0 (default), max_iter is equal to the number of columns in D. (only meaningful for matching pursuit and its variants).
max_residual gives a stopping criterion on signal reconstruction accuracy. (only meaningful for matching pursuit and its variants).
For each selected image, the result is returned as a matrix W whose columns correspond to the weights associated to each column of D, such that the matrix product D*W is an approximation of the input matrix.

**Default values:**

method=0, max_iter=0 and max_residual=1e-6.

---

**mse**

**Arguments:**

- [reference]

**Description:**

Return the MSE (Mean-Squared Error) between selected images and specified reference image.

This command does not modify the images. It returns a value or a list of values in the status.

---

**mse_matrix**

**No arguments**

**Description:**

Compute MSE (Mean-Squared Error) matrix between selected images.

**Example of use:**

```
image.jpg +noise 30 +noise[0] 35 +noise[0] 38 cut. 0,255 +mse_matrix
```
mul

Built-in command

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

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

(equivalent to shortcut command *).

See also:
add, sub, div.

Examples of use:
• Example #1

```
image.jpg +mul 2 cut 0,255
```

• Example #2

```
image.jpg (1,2,3,4,5,6,7,8) ri[-1] [0] mul[0] [-1]
```

• Example #3

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

• Example #4

```
image.jpg +luminance negate[-1] +mul
```
mul3d

Arguments:

- factor
- factor_x, factor_y, factor_z

Description:

Scale selected 3D objects isotropically or anisotropically, with specified factors.

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

Default values:

factor_z=1.

Example of use:

torus3d 5,2 repeat 5 { +add3d[-1] 10,0,0 mul3d[-1] 1.2 color3d[-1] ${-rgb} } add3d
mutex

Arguments:

- index, _action={ 0:unlock | 1:lock }

Description:

Lock or unlock specified mutex for multi-threaded programming.

A locked mutex can be unlocked only by the same thread. All mutexes are unlocked by default. index designates the mutex index, in [0,255].

Default values:

action=1.

nadirzenith2equirectangular

No arguments

Description:

Transform selected nadir/zenith rectilinear projections to equirectangular images.

name

Arguments:

- “name1”, “name2”, ..., “nameN”

Description:

Set names of selected images.

- If no explicit image selection is given, image selection is assumed to be [-N--1], where N is the number of specified arguments.

- If N is higher than the number of images in selection, an error is thrown.
• If N is lower than the number of images in selection, image names are assigned in a periodic way, i.e. `name(selection[k]) = arg[k%N]`.

(equivalent to shortcut command `=>`).

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

**Example of use:**

```verbatim
image.jpg name image blur[image] 2
```

---

**name2color**

**Arguments:**

- name

**Description:**

Return the R,G,B color that matches the specified color name.

---

**named**

**Arguments:**

- mode,"name1","name2",...

**Description:**

Return the set of indices corresponding to images of the selection with specified names.

After this command returns, the status contains a list of indices (unsigned integers), separated by commas (or an empty string if no images with those names have been found).

(equivalent to shortcut command `nmd`).

mode can be `{ 0: all indices (default) | 1: lowest index | 2: highest index | 3: all indices (case insensitive) | 4: lowest index (case insensitive) | 5: highest index (case insensitive)}`
narg

Arguments:
- arg1, arg2, ..., argN

Description:
Return number of specified arguments.

nblend

Arguments:
- [layer], blending_mode, opacity[%], selection_is={ 0:base-layers | 1:top-layers } or
- blending_mode, opacity[%]

Description:

negate

Arguments:
- base_value or
- (no arg)

Description:
Negate image values.

Default values:
base_value=(undefined).

Example of use:

image.jpg +negate
neq

Arguments:

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

Description:

Compute the boolean inequality of selected images with specified value, image or mathematical expression, or compute the boolean inequality of selected images.

(equivalent to shortcut command !=).

Example of use:

image.jpg round 40 neq {round(ia,40)}

network

Arguments:

• mode={ -1=disabled | 0:enabled w/o timeout | >0:enabled w/ specified timeout in seconds }
Description:
Enable/disable load-from-network and set corresponding timeout.
(Default mode is enabled w/o timeout).

newton_fractal

Arguments:
• $z_0r, z_0i, z_1r, z_1i, \_\text{angle}, 0\leq \_\text{descent\_method}\leq 2, \_\text{iteration\_max}\geq 0, \_\text{convergence\_precision}$

Description:
Draw newton fractal on selected images, for complex numbers in range $(z_0r, z_0i) - (z_1r, z_1i)$.
Resulting images have 3 channels whose meaning is $[\text{last}_z; \text{last}_z; \text{nb\_iter\_used\_for\_convergence}]$.
descent\_method can be $\{0: \text{secant} | 1: \text{newton} | 2: \text{householder}\}$.

Default values:
angle=0, descent\_method=1, iteration\_max=200, convergence\_precision=0.01, expr\_p(z)=z\^{3}-1,
expr\_dp(z)=3*z\^{2} and expr\_d2z(z)=6*z.$

Example of use:
```
400,400 newton_fractal -1.5,-1.5,1.5,1.5,0,2,200,0.01,"z\^{6} + z\^{3} - 1","6*z\^{5} + 3*z\^{2}","30*z\^{4} + 6*z" f \
atan2(i1,i0)*90+20,1,cut(i2/30,0.2,0.7) ]" hsl2rgb
```

nlmeans

Arguments:
• $[\text{guide}], _\text{patch\_radius}>0, _\text{spatial\_bandwidth}>0, _\text{tonal\_bandwidth}>0, _\text{patch\_measure\_command}$

or
• $ _\text{patch\_radius}>0, _\text{spatial\_bandwidth}>0, _\text{tonal\_bandwidth}>0, _\text{patch\_measure\_command}$

Description:
Apply non local means denoising of Buades et al, 2005. on selected images.

The patch is a gaussian function of \( \text{std\_patch\_radius} \).
The spatial kernel is a rectangle of radius \( \text{spatial\_bandwidth} \).
The tonal kernel is exponential \( \exp(-d^2/\text{tonal\_bandwidth}^2) \)
with \( d \) the euclidean distance between image patches.

**Default values:**

\( \text{patch\_radius}=4, \text{spatial\_bandwidth}=4, \text{tonal\_bandwidth}=10 \) and \( \text{patch\_measure\_command}=\text{norm} \).

**Example of use:**

```
image.jpg +noise 10 nlmeans[-1] 4,4,\{0.6*\{-\text{std\_noise}\}\}
```

---

### nlmeans_core

**Arguments:**

- \( \text{reference\_image}, \text{scaling\_map}, \text{patch\_radius}>0, \text{spatial\_bandwidth}>0 \)

**Description:**

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

---

### nn_add

**Arguments:**

- \( \text{out}, \text{in0}, \text{in1} \)

**Description:**

Add an add layer to the network.

**Default values:**

\( \text{in1}=\text{. (previous layer)} \).
**nn_append**

**Arguments:**
- `out, in0, _in1`

**Description:**
Add an append layer to the network.

**Default values:**
`in1=.' (previous layer).

---

**nn_avgpool2d**

**Arguments:**
- `out, in, _patch_size>1`

**Description:**
Add a avgpool2d layer (2D average pooling) to the network.

**Default values:**
`in=.' (previous layer).

---

**nn_avgpool3d**

**Arguments:**
- `out, in, _patch_size>1`

**Description:**
Add a avgpool3d layer (3D average pooling) to the network.

**Default values:**
`in=.' (previous layer).

---

**nn_check_layer**

**Arguments:**

**Check that the layer with specified name already exists in the network.**

**Add a clone layer to the network.**

**Add a conv2d layer (2D convolutional layer) to the network.**

**Add a conv2dnl (2D convolutional layer followed by a non-linearity) to the network.**
learning_mode can be \{ 0:\text{no learning} \mid 1:\text{weights only} \mid 2:\text{biases only} \mid 3:\text{weights+biases} \}. 

**Default values:**

kernel_size=3, stride=1, dilation=1, border_shrink=0, boundary_conditions=1, activation=leakyrelu and learning_mode=3.

---

\texttt{nn\_conv2dnnl}

**Arguments:**

- out, in, nb_channels>0, _kernel_size>0, _stride>0, _dilation>0, _border_shrink>=0, _boundary_conditions

**Description:**

Add a conv2dnnl (2D convolutional layer followed by a normalization layer, then a non-linearity) to the network.

boundary_conditions can be \{ 0: \text{dirichlet} \mid 1: \text{neumann} \mid 2: \text{periodic} \mid 3: \text{mirror} \}. 

learning_mode can be \{ 0: \text{no learning} \mid 1: \text{weights only} \mid 2: \text{biases only} \mid 3: \text{weights+biases} \}.

**Default values:**

kernel_size=3, stride=1, dilation=1, border_shrink=0, boundary_conditions=1, activation=leakyrelu and learning_mode=3.

---

\texttt{nn\_conv3d}

**Arguments:**

- out, in, nb_channels>0, _kernel_size>0, _stride>0, _dilation, _border_shrink>=0, _boundary_conditions

**Description:**

Add a conv3d layer (3D convolutional layer) to the network.

boundary_conditions can be \{ 0: \text{dirichlet} \mid 1: \text{neumann} \mid 2: \text{periodic} \mid 3: \text{mirror} \}. 

learning_mode can be \{ 0: \text{no learning} \mid 1: \text{weights only} \mid 2: \text{biases only} \mid 3: \text{weights+biases} \}.

**Default values:**

kernel_size=3, stride=1, dilation=1, border_shrink=0, boundary_conditions=1 and learning_mode=3.

---

\texttt{nn\_conv3dnl}
**Arguments:**

- `out, in, nb_channels>0, kernel_size>0, stride>0, dilation>0, border_shrink>=0, boundary_conditions`<br><br>**Description:**

Add a conv3dnl (3D convolutional layer followed by a non-linearity) to the network.

boundary_conditions can be `{ 0: dirichlet | 1: neumann | 2: periodic | 3: mirror }`. learning_mode can be `{ 0: no learning | 1: weights only | 2: biases only | 3: weights+biases }`.

**Default values:**

kernel_size=3, stride=1, dilation=1, border_shrink=0, boundary_conditions=1, activation=leakyrelu and learning_mode=3.

---

**nn_conv3dnnl**

**Arguments:**

- `out, in, nb_channels>0, kernel_size>0, stride>0, dilation>0, border_shrink>=0, boundary_conditions`<br><br>**Description:**

Add a conv3dnnl (3D convolutional layer followed by a normalization layer, then a non-linearity) to the network.

boundary_conditions can be `{ 0: dirichlet | 1: neumann | 2: periodic | 3: mirror }`. learning_mode can be `{ 0: no learning | 1: weights only | 2: biases only | 3: weights+biases }`.

**Default values:**

kernel_size=3, stride=1, dilation=1, border_shrink=0, boundary_conditions=1, activation=leakyrelu and learning_mode=3.

---

**nn_crop**

**Arguments:**

- `out, in, x0, y0, z0, c0, x1, y1, z1, c1, _boundary_conditions`<br><br>**Description:**

Add a crop layer to the network.

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

**Default values:**
**nn_distance**

**Arguments:**

- \textit{out}, \textit{in0}, \textit{in1}, \textit{metric} = \{0: squared-L2 | \text{p}>0: Lp-norm\}

**Description:**
Add a distance layer to the network (distance between two inputs, with specified metric).

**Default values:**

\textit{in} = (previous layer),

---

**nn_dropout**

**Arguments:**

- \textit{out}, \textit{in}, 0 \leq \textit{dropout\_rate} < 1

**Description:**
Add a dropout layer to the network.

---

**nn_fc**

**Arguments:**

- \textit{out}, \textit{in}, nb\_channels > 0, 0 \leq \textit{learning\_mode} \leq 3

**Description:**
Add a fc layer (fully connected layer) to the network.

\textit{learning\_mode} can be \{0: no learning | 1: weights only | 2: biases only | 3: weights+biases\}.

**Default values:**

\textit{learning\_mode} = 3.

---

**nn_fcnl**

**Arguments:**
Add a fcnl layer (fully connected layer followed by a non-linearity) to the network.

learning_mode can be \{ 0:\textit{no learning} | 1:\textit{weights only} | 2:\textit{biases only} | 3:\textit{weights+biases} \}.

**Default values:**
activation=leakyrelu and learning_mode=3.

---

**nn_fcnnl**

**Arguments:**
- out, in, nb_neurons>0, _activation, 0<=_learning_mode<=3

**Description:**
Add a fcnl layer (fully connected layer followed by a normalization layer, then a non-linearity) to the network.

learning_mode can be \{ 0:\textit{no learning} | 1:\textit{weights only} | 2:\textit{biases only} | 3:\textit{weights+biases} \}.

**Default values:**
activation=leakyrelu and learning_mode=3.

---

**nn_init**

**No arguments**

**Description:**
Initialize a new network.

---

**nn_input**

**Arguments:**
- name, width, _height, _depth, _spectrum

**Description:**
Add a new input to the network.
Default values:
height=1, depth=1 and spectrum=1.

### nn_lib

No arguments

**Description:**
Return the list of library functions that has to be included in a math expression, in order to use the neural network library.

### nn_load

**Arguments:**
- `'filename.gmz', include_trainer_data={ 0: no | 1: yes }`

**Description:**
Load and initialize network saved as a .gmz file.

Neural network files can be only loaded in .gmz format.

**Default values:**
include_trainer_data=1.

### nn_loss_binary_crossentropy

**Arguments:**
- `out, in, ground_truth`

**Description:**
Add a binary_crossentropy loss to the network (binary cross entropy).

### nn_loss_crossentropy

**Arguments:**
- `out, in, ground_truth`

**Description:**
Add a crossentropy loss to the network (cross entropy).

**nn_loss_mse**

**Arguments:**
- $\text{out, in, ground\_truth}$

**Description:**
Add a mse loss to the network (mean-squared error).

**nn_loss_normp**

**Arguments:**
- $\text{out, in, ground\_truth, _metric={ 0:squared-L2 | p>0:Lp-norm }}$

**Description:**
Add a normp loss to the network ($\| out - ground\_truth \|_p$).

**Default values:**
metric=1.

**nn_loss_softmax_crossentropy**

**Arguments:**
- $\text{out, in, ground\_truth}$

**Description:**
Add a softmax_crossentropy loss to the network (softmax followed by cross entropy).

**nn_maxpool2d**

**Arguments:**
- $\text{out, in, _patch\_size>1}$

**Description:**
Add a maxpool2d layer (2D max pooling) to the network.

**Default values:**
### nn_maxpool3d

**Arguments:**
- `out, in, patch_size>1`

**Description:**
Add a maxpool3d layer (3d max pooling) to the network.

**Default values:**
- `in=. (previous layer) and patch_size=2.`

### nn_mul

**Arguments:**
- `out, in0, in1`

**Description:**
Add an mul layer to the network.

**Default values:**
- `in1=. (previous layer)`

### nn_nl

**Arguments:**
- `out, in, activation`

**Description:**
Add a nl (nonlinearity) layer to the network.

**activation can be** `{ elu | gelu | leakyrelu | linear | relu | sigmoid | sin | sinc | softmax | sqr | sqrt | swish | tanh }.

**Default values:**
- `in=. (previous layer) and activation=leakyrelu.`
**nn_nlfc**

**Arguments:**
- `out, in, nb_channels>0, _activation, 0<=_learning_mode<=3`

**Description:**
Add a nlfc layer (nonlinear fully connected layer) to the network.

Learning_mode can be `{ 0: no learning | 1: weights only | 2: biases only | 3: weights+biases }`.

**Default values:**
- `activation=leakyrelu` and `learning_mode=3`.

---

**nn_normalize**

**Arguments:**
- `out, in, _normalization_mode_, 0<=_learning_mode<=3`

**Description:**
Add a normalize layer to the network.

Normalization_mode can be `{ 0: global parameters | 1: channel-by-channel parameters }`

Learning_mode can be `{ 0: no learning | 1: alpha only | 2: beta only | 3: alpha+beta }`

**Default values:**
- `in=. (previous layer), normalization_mode=0` and `learning_mode=3`.

---

**nn_patchdown2d**

**Arguments:**
- `out, in, _patch_size>1`

**Description:**
Add a patchdown2d (2D downscale by patch) layer to the network.

**Default values:**
- `in=. (previous layer)` and `patch_size=2`.
**nn_patchdown3d**

**Arguments:**
- `out, in, patch_size>1`

**Description:**
Add a patchdown3d (3D downscale by patch) layer to the network.

**Default values:**
in=. (previous layer) and patch_size=2.

---

**nn_patchup2d**

**Arguments:**
- `out, in, patch_size>1`

**Description:**
Add a patchup2d (2D upscale by patch) layer to the network.

**Default values:**
in=. (previous layer) and patch_size=2.

---

**nn_patchup3d**

**Arguments:**
- `out, in, patch_size>1`

**Description:**
Add a patchup3d (3D upscale by patch) layer to the network.

**Default values:**
in=. (previous layer) and patch_size=2.

---

**nn_print**

No arguments
Description:
Print info on current neural network.

---

**nn_rename**

**Arguments:**
- `out, in`

**Description:**
Add a rename layer to the network.

**Default values:**
in=. (previous layer).

---

**nn_resconv2d**

**Arguments:**
- `out, in, kernel_size>0, dilation>0, boundary_conditions, 0<= learning_mode<=3`

**Description:**
Add a resconv2d (residual 2D convolutional layer) to the network.

boundary_conditions can be `{ 0: dirichlet | 1: neumann | 2: periodic | 3: mirror }.
learning_mode can be `{ 0: no learning | 1: weights only | 2: biases only | 3: weights+biases }.

**Default values:**
in= (previous layer), kernel_size=3, dilation=1, boundary_conditions=1 and learning_mode=3.

---

**nn_resconv2dnl**

**Arguments:**
- `out, in, kernel_size>0, dilation>0, boundary_conditions, activation, 0<= learning_mode<=3`

**Description:**
Add a resconv2dnl (residual 2D convolutional layer followed by a non-linearity) to the network.

boundary_conditions can be `{ 0: dirichlet | 1: neumann | 2: periodic | 3: mirror }.
learning_mode can be `{ 0: no learning | 1: weights only | 2: biases only | 3: weights+biases }.
Default values:
in=. (previous layer), kernel_size=3, dilation=1, boundary_conditions=1, activation=leakyrelu and learning_mode=3.

---

**nn_resconv2dnnl**

Arguments:

- `out, in, kernel_size>0, dilation>0, boundary_conditions, activation, 0<=learning_mode<=3`

Description:

Add a resconv2dnnl (residual 2D convolutional layer followed by a normalization layer, then a non-linearity) to the network.

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

learning_mode can be `{0:no learning | 1:weights only | 2:biases only | 3:weights+biases}`.

Default values:

in=. (previous layer), kernel_size=3, dilation=1, boundary_conditions=1, activation=leakyrelu and learning_mode=3.

---

**nn_resconv3d**

Arguments:

- `out, in, kernel_size>0, dilation>0, boundary_conditions, 0<=learning_mode<=3`

Description:

Add a resconv3d (residual 3D convolutional layer) to the network.

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

learning_mode can be `{0:no learning | 1:weights only | 2:biases only | 3:weights+biases}`.

Default values:

in=. (previous layer), kernel_size=3, dilation=1, boundary_conditions=1 and learning_mode=3.

---

**nn_resconv3dnl**

Arguments:

- `out, in, kernel_size>0, dilation>0, boundary_conditions, activation, 0<=learning_mode<=3`
Description:
Add a resconv3dnl (residual 3D convolutional layer followed by a non-linearity) to the network.

boundary_conditions can be \{ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror \}.
learning_mode can be \{ 0:no learning | 1:weights only | 2:biases only | 3:weights+biases \}.

Default values:
in=. (previous layer), kernel_size=3, dilation=1, boundary_conditions=1, activation='leakyrelu' and learning_mode=3.

\textbf{nn\_resconv3dnnl}

Arguments:
- \texttt{out, in, kernel\_size>0, dilation>0, boundary\_conditions, activation, 0<=learning\_mode<=3}

Description:
Add a resconv3dnnl (residual 3D convolutional layer followed by a normalization layer, then a non-linearity) to the network.

boundary_conditions can be \{ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror \}.
learning_mode can be \{ 0:no learning | 1:weights only | 2:biases only | 3:weights+biases \}.

Default values:
in=. (previous layer), kernel_size=3, dilation=1, boundary_conditions=1, activation=leakyrelu and learning_mode=3.

\textbf{nn\_resfc}

Arguments:
- \texttt{out, in, 0<=learning\_mode<=3}

Description:
Add a resfc (residual fully connecter layer) to the network.

learning_mode can be \{ 0:no learning | 1:weights only | 2:biases only | 3:weights+biases \}.

Default values:
in=. (previous layer) and learning_mode=3.
nn_resfcnl

Arguments:

- \( \text{out, in, activation, 0<=learning\_mode<=3} \)

Description:

Add a resfcnl (residual fully connecter layer followed by a non-linearity) to the network.

learning\_mode can be \{ 0: no learning | 1: weights only | 2: biases only | 3: weights+biases \}.

Default values:

in=. (previous layer), activation=leakyrelu and learning\_mode=3.

---

nn_resfcnnl

Arguments:

- \( \text{out, in, activation, 0<=learning\_mode<=3} \)

Description:

Add a resfcnnl (residual fully connecter layer followed by a normalization layer, then a non-linearity) to the network.

learning\_mode can be \{ 0: no learning | 1: weights only | 2: biases only | 3: weights+biases \}.

Default values:

in=. (previous layer), activation=leakyrelu and learning\_mode=3.

---

nn_reshape

Arguments:

- \( \text{out, in, width>0, height>0, depth>0, spectrum>0} \)

Description:

Add a reshape layer to the network.

---

nn_resize

Arguments:
Description:
Add a resize layer to the network.

Default values:
height=depth=spectrum=100% and interpolation=3.

**nn_run**

Arguments:
- `out, in, "command", width[%]>0, height[%]>0, depth[%]>0, spectrum[%]>0`

Description:
Add a run layer to the network.

Default values:
width=height=depth=spectrum=100%.

**nn_save**

Arguments:
- `'filename.gmz', include_trainer_data={ 0:no | 1:yes }`

Description:
Save current network as a .gmz file.

*.gmz* is mandatory extension, specifying another file extension will throw an error.

Default values:
include_trainer_data=1.

**nn_size**

No arguments

Description:
Return size of the current network (i.e. number of stored parameters).
nn_split

**Arguments:**
- name0, name1, in, nb_channels0

**Description:**
Add a split layer to the network.

nn_store

**Arguments:**
- `variable_name`, _include_trainer_data={ 0:no | 1:yes }

**Default values:**
- include_trainer_data=1.

nn_trainer

**Arguments:**
- name, _loss, _learning_rate>0, _optimizer, _scheduler

**Description:**
Add a network trainer to the network.

- optimizer can be `{ sgd | rmsprop | adam | adamax }.
- scheduler can be `{ constant | linear | exponential | adaptive }.

**Default values:**
- loss=. (previous loss), learning_rate=2e-4, optimizer=rmsprop and scheduler=constant.

noarg

**Built-in command**

**No arguments**

**Description:**
Used in a custom command, noarg tells the command that its argument list have not been used.
finally, and so they must be evaluated next in the G'MIC pipeline, just as if the custom command takes no arguments at all. Use this command to write a custom command which can decide if it takes arguments or not.

---

noise

Built-in command

**Arguments:**

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

**Description:**

Add random noise to selected images.

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

**Default values:**

`noise_type=0`.

**Examples of use:**

- **Example #1**

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

- **Example #2**

  ```
  300,300,1,3 [0] noise[0] 20,0 noise[1] 20,1 +histogram 100
  ```
noise_hurl

Arguments:

- \_amplitude\geq0

Description:

Add hurl noise to selected images.

Default values:

amplitude=10.

Example of use:

image.jpg +noise_hurl ,
### noise_perlin

**Arguments:**

- \_scale\_x[%]>0, \_scale\_y[%]>0, \_scale\_z[%]>0, \_seed\_x, \_seed\_y, \_seed\_z

**Description:**

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


**Default values:**

scale\_x=scale\_y=scale\_z=16 and seed\_x=seed\_y=seed\_z=0.

**Example of use:**

```
500,500,1,3 noise_perlin ,
```

### noise_poissondisk

**Arguments:**

- \_radius[%]>0, \_max\_sample\_attempts>0, \_p\_norm>0

**Description:**
Add poisson disk sampling noise to selected images.

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

**Default values:**

radius=8, max_sample_attempts=30 and p_norm=2.

**Example of use:**

```
300,300 noise_poissondisk 8
```

---

**norm1**

No arguments

**Description:**

Compute the pointwise L1-norm of vector-valued pixels in selected images.

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

**Example of use:**

```
image.jpg +norm1
```
norm2

No arguments

Description:
Compute the pointwise L2-norm (euclidean norm) of vector-valued pixels in selected images.

This command has a tutorial page.

Example of use:

```
image.jpg +norm
```

normalize

Arguments:

• `{ value0[%] | [image0] },{ value1[%] | [image1] },_constant_case_ratio` or
• `[image]`

Description:
Linearly normalize values of selected images in specified range.

(equivalent to shortcut command `in`).

This command has a tutorial page.

Example of use:

```
image.jpg split x,2 normalize[-1] 64,196 append x
```
**normalize3d**

**No arguments**

**Description:**

Normalize selected 3D objects to unit size.

*(equivalent to shortcut command)*

**Example of use:**

```
repeat 100 { circle3d {u(3)},{u(3)},{u(3)},0.1 } add3d color3d[-1] 255,0,0 +normalize3d[-1] color3d[-1] 0,255,0 add3d
```

---

**normalize_filename**

**Arguments:**

- **filename**

**Description:**

Return a "normalized" version of the specified filename, without spaces and capital letters.
**normalize_l2**

No arguments

**Description:**
Normalize selected images such that they have a unit L2 norm.

**normalize_local**

**Arguments:**
- \( \text{amplitude} \geq 0, \text{radius} > 0, \text{n_smooth} \geq 0 \%, \text{a_smooth} \geq 0 \%, \text{is_cut} \in \{0, 1\}, \text{min} = 0, \text{max} = 255 \)

**Description:**
Normalize selected images locally.

**Default values:**
- \( \text{amplitude} = 3, \text{radius} = 16, \text{n_smooth} = 4 \%, \text{a_smooth} = 2 \%, \text{is_cut} = 1, \text{min} = 0 \text{ and max} = 255. \)

**Example of use:**
```
image.jpg normalize_local 8,10
```

**normalize_sum**

No arguments

**Description:**
Normalize selected images such that they have a unit sum.

**Example of use:**
```
image.jpg +histogram 256 normalize_sum[-1] display_graph[-1] 400,300
```
normalized_cross_correlation

**Arguments:**

- [mask]

**Description:**
Compute normalized cross-correlation of selected images with specified mask.

**Example of use:**

```
image.jpg +shift -30,-20 +normalized_cross_correlation[0] [1]
```
Arguments:

• \( p \geq 0 \)

Description:

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

Default values:

\( p = 2 \).

Example of use:

```plaintext
image.jpg +normp[0] 0 +normp[0] 1 +normp[0] 2 +normp[0] inf
```

---

**object3d**

Built-in command

Arguments:
Description:
Draw specified 3D object on selected images.
(equivalent to shortcut command j3d).

rendering_mode can be { 0:dots | 1:wireframe | 2:flat | 3:flat-shaded | 4:gouraud-shaded | 5:phong-shaded }.

Default values:
x=y=z=0, opacity=1 and is_zbuffer=1. All other arguments take their default values from the 3D environment variables.

Example of use:
```
image.jpg torus3d 100,10 cone3d 30,-120 add3d[-2,-1] rotate3d.
1,1,0,60 object3d[0] [-1],50%,50% keep[0]
```

oct

Arguments:
- octal_int1,...

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

oct2dec

Arguments:
- octal_int1,...
Description:
Convert specified octal integers into their decimal representations.

oklab2rgb
No arguments
Description:
Convert color representation of selected images from OKlab to RGB.
(see colorspace definition at: https://bottosson.github.io/posts/oklab/).
See also:
rgb2oklab.

old_photo
No arguments
Description:
Apply old photo effect on selected images.
Example of use:

image.jpg old_photo

oneminus
No arguments
Description:
For each selected image, compute one minus image.
Example of use:

```
image.jpg normalize 0,1 +oneminus
```

---

**onfail**

No arguments

**Description:**

Execute following commands when an error is encountered in the body of the local...done block.

The status value is set with the corresponding error message.

**Example of use:**

```
image.jpg +local blur -3 onfail mirror x done
```

---

**opacity3d**

**Arguments:**

- `opacity`

**Description:**
Set opacity of selected 3D objects.

*(equivalent to shortcut command \(o3d\)).*

**Example of use:**

```
torus3d 100,10 double3d 0 repeat 7 \{ +rotate3d[-1] 1,0,0,20
opacity3d[-1] \{u\} \} add3d
```

---

**opening**

**Arguments:**

- \(size\geq0\) or
- \(size_x\geq0, size_y\geq0, size_z\geq0\) or
- \([kernel], _boundary\_conditions, _is\_real={ 0:binary-mode | 1:real-mode }\)

**Description:**

Apply morphological opening to selected images.

boundary\_conditions can be \{ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror \}.

**Default values:**

size\_z=1, boundary\_conditions=1 and is\_real=0.

**Example of use:**

```
image.jpg +opening 10
```
opening_circ

**Arguments:**

- size\(\geq 0\), is\_real\{ 0 | 1 \}

**Description:**

Apply circular opening of selected images by specified size.

**Default values:**

boundary\_conditions=1 and is\_real=0.

**Example of use:**

```
image.jpg +opening_circ 7
```

or

**Built-in command**

**Arguments:**

- value\[%\] or [image] or ‘formula’ or (no arg)
Description:
Compute the bitwise OR of selected images with specified value, image or mathematical expression, or compute the pointwise sequential bitwise OR of selected images.

(equivalent to shortcut command ||).

Examples of use:

• Example #1

```
image.jpg or 128
```

• Example #2

```
image.jpg +mirror x or
```

orientation

No arguments

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

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

Example of use:
orthogonalize

Arguments:

- \textit{mode} = \{ 0:orthogonalize | 1:orthonormalize \}

Description:

Orthogonalize or orthonormalize selected matrices, using Modified Gram-Schmidt process.

Default values:

mode=0.

otsu

Arguments:

- \textit{nb\_levels} $>$ 0

Description:

Hard-threshold selected images using Otsu's method.

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

nb_levels=256.

Example of use:

```
image.jpg luminance +otsu ,
```

output

**Arguments:**

- `[type:]filename, _format_options`

**Description:**

Output selected images as one or several numbered file(s).

*(equivalent to shortcut command 'o').*

**Default values:**

'format_options'=(undefined).

output_565

**Arguments:**

- `filename`, `reverse_endianness={ 0:false | 1:true }`

**Description:**

Output selected images as raw RGB-565 files.

**Default values:**

reverse_endianness=0.
output_cube

Arguments:

- filename

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

output_flo

Arguments:

- filename

Description:
Output selected optical flow as a .flo file (vision.middlebury.edu file format).

output_ggr

Arguments:

- filename, gradient_name

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

output_gmz

Arguments:

- filename, datatype

Description:
Output selected images as .gmz files (G'MIC native file format).

datatype can be { bool | uint8 | int8 | uint16 | int16 | uint32 | int32 | uint64 | int64 | float32 | float64 }.

output_obj
Arguments:

- filename, save_materials={ 0:no | 1:yes }

Description:

Output selected 3D meshes as Wavefront 3D object files.

Set save_materials to 1 to produce a corresponding material file (.mtl) and eventually texture files. Beware, the export to .obj files may be quite slow for large 3D objects.

Default values:

save_materials=1.

---

**output_text**

Arguments:

- filename

Description:

Output selected images as text-data filenames.

(equivalent to shortcut command `ot`).

---

**outputn**

Arguments:

- filename, _index

Description:

Output selected images as automatically numbered filenames in repeat...done loops.

(equivalent to shortcut command `on`).

---

**outputp**

Arguments:

- prefix

Description:

Output selected images as prefixed versions of their original filenames.
Default values:

prefix=_.

outputw

No arguments

Description:
Output selected images by overwriting their original location.

(output equivalent to shortcut command op).

outputx

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

Description:
Output selected images with same base filenames but for N different extensions.

(output equivalent to shortcut command ox).

Default values:
output_at_same_location=0.

pack

Arguments:
- is_ratio_constraint={ 0 | 1 }, sort_criterion

Description:
Pack selected images into a single image.

The returned status contains the list of new (x,y) offsets for each input image.
Parameter is_ratio_constraint tells if the resulting image must tend to a square image.

Default values:
is_ratio_constraint=0 and sort_criterion=max(w,h).
pack_sprites

**Arguments:**

- \( nb_scales \geq 0 \), \( 0 \leq min\_scale \leq 100 \), \( allow\_rotation= \{ 0:0 \text{ deg.} | 1:180 \text{ deg.} \mid 2:90 \text{ deg.} | 3:\text{any} \} \), \( spacing \), \( precision \geq 0 \), \( max\_iterations \geq 0 \)

**Description:**

Try to randomly pack as many sprites as possible onto the empty areas of an image.

Sprites can be eventually rotated and scaled during the packing process.
First selected image is the canvas that will be filled with the sprites.
Its last channel must be a binary mask whose zero values represent potential locations for drawing the sprites.
All other selected images represent the sprites considered for packing.
Their last channel must be a binary mask that represents the sprite shape (i.e. a 8-connected component).
The order of sprite packing follows the order of specified sprites in the image list.
Sprite packing is done on random locations and iteratively with decreasing scales.
\( nb_scales \) sets the number of decreasing scales considered for all specified sprites to be packed.
\( min\_scale \) (in %) sets the minimal size considered for packing (specified as a percentage of the original sprite size).
\( spacing \) can be positive or negative.
\( precision \) tells about the desired number of failed trials before ending the filling process.

**Default values:**

\( nb_scales=5 \), \( min\_scale=25 \), \( allow\_rotation=3 \), \( spacing=1 \), \( precision=7 \) and \( max\_iterations=256 \).

**Example of use:**

```
512,512,1,3,"min(255,y*c/2)" 100%,100% circle 50%,50%,100,1,255
append c image.jpg rescale2d[-1],24 to_rgba pack_sprites 3,25
```
**padint**

**Arguments:**
- number, size>0

**Description:**
Return an integer with size digits (eventually left-padded with 0).

---

**palette**

**Arguments:**
- palette_name | palette_number

**Description:**
Input specified color palette at the end of the image list.

Palette names can be:
- default | hsv | lines | hot | cool | jet | flag | cube | rainbow | parula | spring | summer | autumn | winter | bone | copper | pink | vga | algae | amp | balance | curl | deep | delta | dense | diff | gray | haline | ice | matter | oxy | phase | rain | solar | speed | tarn | tempo | thermal | topo | turbid | aurora | hocuspocus | srb2 | uzebox | amiga7800 | amiga7800mess | fornaxvoid1

**Example of use:**

```plaintext
palette hsv
```
**parallel**

**Arguments:**
- `wait_threads,"command1","command2",...`

**Description:**
Execute specified commands in parallel, each in a different thread.

Parallel threads share the list of images.
wait_threads can be `{ 0:when current environment ends | 1:immediately }`.

**Default values:**
`wait_threads=1`.

**Example of use:**
```
image.jpg [0] parallel "blur[0] 3","mirror[1] c"
```

---

**parametric3d**

**Arguments:**
- `x(a,b), y(a,b), z(a,b), amin, amax, bmin, bmax, res_a>0, res_b>0, res_x>0, res_y>0, res_z>0, smoothness, isovalue`

**Description:**
Input 3D object from specified parametric surface \( (a,b) \rightarrow (x(a,b),y(a,b),z(a,b)) \).

**Default values:**
\[\begin{align*}
    x &= (2+\cos(b))\cdot\sin(a), \\
    y &= (2+\cos(b))\cdot\cos(a), \\
    c &= \sin(b), \\
    a_{\text{min}} &= -\pi, \\
    a_{\text{max}} &= \pi, \\
    b_{\text{min}} &= -\pi, \\
    b_{\text{max}} &= \pi, \\
    \text{res}_a &= 512, \\
    \text{res}_b &= \text{res}_a, \\
    \text{res}_x &= 64, \\
    \text{res}_y &= \text{res}_x, \\
    \text{res}_z &= \text{res}_y, \\
    \text{smoothness} &= 2\% \\ \\
    \text{isovalue} &= 10\%
\end{align*}\]

**Example of use:**
```
parametric3d
```

---
parse_cli

Arguments:

- `_output_mode, _{ * | command_name }

Description:

Parse definition of @cli-documentated commands and output info about them in specified output mode.

output_mode can be `{ ascii | bashcompletion | html | images | print }.

Default values:

output_mode=print and command_name=*.

parse_gmd

No arguments

Description:

Parse and tokenize selected images, viewed as text strings formatted with the G'MIC markdown syntax.

parse_gui

Arguments:

- `_outputmode, _{ * | filter_name }

Description:

Parse selected filter definitions and generate info about filters in selected output mode.
outputmode can be \{ gmicol | images | json | list | print | strings | update | zart \}.

It is possible to define a custom output mode, by implementing the following commands (outputmode must be replaced by the name of the custom user output mode):

- parse_gui_outputmode : A command that outputs the parsing information with a custom format.
- parse_gui_parseparams_outputmode (optional): A simple command that returns 0 or 1. It tells the parser whether parameters of matching filter must be analyzed (slower) or not.
- parse_gui_trigger_outputmode (optional): A command that is called by the parser just before parsing the set of each matching filters.

Here is the list of global variables set by the parser, accessible in command parse_gui_outputmode:

- \$_nb_filters: Number of matching filters.
- \$_nongui (stored as an image): All merged lines in the file that do not correspond to #@gui lines.

For each filter \#F (F in range \[0,\$_nb_filters-1\]):

- \$_fF_name : Filter name.
- \$_fF_path : Full path.
- \$_fF_locale : Filter locale (empty, if not specified).
- \$_fF_command : Filter command.
- \$_fF_command_preview : Filter preview command (empty, if not specified).
- \$_fF_zoom_factor : Default zoom factor (empty, if not specified).
- \$_fF_preview_accuracy : Preview accuracy (can be \{ 0:does not support zoom in/out | 1:support zoom in/out | 2:pixel-perfect \}).
- \$_fF_input_mode : Default preferred input mode (empty, if not specified).
- \$_fF_hide : Path of filter hid by current filter (for localized filters, empty if not specified).
- \$_fF_nb_params : Number of parameters.

For each parameter \#P of the filter \#F (P in range \[0,\$_fF_nb_params-1\]):

- \$_fF_pP_name : Parameter name.
- \$_fF_pP_type : Parameter type.
- \$_fF_pP_responsivity : Parameter responsivity (can be \{ 0 | 1 \}).
- \$_fF_pP_randomizable : Randomizable property of the parameter (can be \{ 0 | 1 \}).
- \$_fF_pP_visibility : Parameter visibility.
- \$_fF_pP_propagation : Propagation of the parameter visibility.
- \$_fF_pP_nb_args : Number of parameter arguments.

For each argument \#A of the parameter \#P (A in range \[0,\$_fF_pP_nb_args-1\]):

- \$_fF_pP_aA : Argument value

Default parameters: filter_name=\* and output_format=print.
Arguments:

- \texttt{shared\_state=\{-1:status\ only | 0:non-shared (copy) | 1:shared | 2:adaptive \}}

Description:

Insert images from parent context of a custom command or a local environment.

Command selection (if any) stands for a selection of images in the parent context. By default (adaptive shared state), selected images are inserted in a shared state if they do not belong to the context (selection) of the current custom command or local environment as well. Typical use of command pass concerns the design of custom commands that take images as arguments. This commands return the list of corresponding indices in the status.

Default values:

\texttt{shared\_state=2}.

Example of use:

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

patches

Arguments:

- \texttt{patch\_width>0,patch\_height>0,patch\_depth>0,x0,y0,z0,x1,y1,z1,\ldots,xN,yN,zN}

Description:

Extract \(N+1\) patches from selected images, centered at specified locations.

Example of use:

```
image.jpg +patches 64,64,1,153,124,0,184,240,0,217,126,0,275,38,0
```
patches2img

**Arguments:**

- \text{width}>0, \text{height}>0, \text{overlap[%]}>0, \text{overlap\_std[%]}

**Description:**

Recompose 2D images from their selected patch representations.

\text{overlap} must be in range \([0, \text{patch\_size}-1]\) where \text{patch\_size} is the width/height of the selected image.

\text{overlap\_std} is the standard deviation of the gaussian weights used for reconstructing overlapping patches.

If \text{overlap\_std} is set to -1, uniform weights are used rather than gaussian.

**Default values:**

\text{overlap}=0 and \text{overlap\_std}=-1.

**See also:**

\text{img2patches}.

**Example of use:**

```

image.jpg +img2patches 32,0,3 mirror[-1] xy patches2img[-1] {0,[w,h]}

```
path_cache

No arguments

Description:
Return a path to store G'MIC data files for one user (whose value is OS-dependent).

path_current

No arguments

Description:
Return current folder from where G'MIC has been run.

path_gimp

No arguments

Description:
Return a path to store GIMP configuration files for one user (whose value is OS-dependent).

path_tmp

No arguments

Description:
Return a path to store temporary files (whose value is OS-dependent).

pca_patch3d
Arguments:

- \texttt{patch\_size}>0, \texttt{M}>0, \texttt{N}>0, \texttt{normalize\_input}={ 0 \mid 1 }, \texttt{normalize\_output}={ 0 \mid 1 }, \texttt{lambda\_xy}

Description:

Get 3D patch-pca representation of selected images.

The 3D patch-pca is estimated from \texttt{M} patches on the input image, and displayed as a cloud of \texttt{N} 3D points.

Default values:

\texttt{patch\_size}=7, \texttt{M}=1000, \texttt{N}=3000, \texttt{normalize\_input}=1, \texttt{normalize\_output}=0, and \texttt{lambda\_xy}=0.

Example of use:

\begin{verbatim}
image.jpg pca\_patch3d 7
\end{verbatim}

---

\texttt{pde\_flow}

Arguments:

- \texttt{nb\_iter}>=0, \texttt{dt}, \texttt{velocity\_command}, \texttt{keep\_sequence}={ 0 \mid 1 }

Description:

Apply iterations of a generic PDE flow on selected images.

Default values:

\texttt{nb\_iter}=10, \texttt{dt}=30, \texttt{velocity\_command}=laplacian and \texttt{keep\_sequence}=0.

Example of use:

\begin{verbatim}
image.jpg +pde\_flow 20
\end{verbatim}
pencilbw

Arguments:

- \( \text{size} \geq 0, \text{amplitude} \geq 0 \)

Description:

Apply B&W pencil effect on selected images.

Default values:

size=0.3 and amplitude=60.

Example of use:

```
image.jpg pencilbw ,
```

percentile

Arguments:

- \([\text{mask}], 0 \leq \text{min\_percentile}[] \leq 100, 0 \leq \text{max\_percentile}[] \leq 100\).

Description:

Apply percentile averaging filter to selected images.
Default values:

\texttt{min\_percentile=0 and max\_percentile=100.}

Example of use:

\begin{verbatim}
   image.jpg shape_circle 11,11 +percentile[0] [1],25,75
\end{verbatim}

\begin{verbatim}
   image.jpg +periodize_poisson array 2,2,2
\end{verbatim}

**periodize_poisson**

No arguments

**Description:**

Periodize selected images using a Poisson solver in Fourier space.

Example of use:

\begin{verbatim}
   image.jpg +periodize_poisson array 2,2,2
\end{verbatim}
permute

Arguments:

- permutation_string

Description:

Permute selected image axes by specified permutation.

permutation is a combination of the character set \{x|y|z|c\}, e.g. xycz, cxyz, ...

Example of use:

```
image.jpg permute yxzc
```

peronamalik_flow

Arguments:

- K_factor>0, _nb_iter>=0, _dt, _keep_sequence={ 0 | 1 }

Description:

Apply iterations of the Perona-Malik flow on selected images.
Default values:

K_factor=20, nb_iter=5, dt=5 and keep_sequence=0.

Example of use:

```
image.jpg +heat_flow 20
```

---

**phase_correlation**

**Arguments:**

- `[destination]`

**Description:**

Estimate translation vector between selected source images and specified destination.

**Example of use:**

```
image.jpg +shift -30,-20 +phase_correlation[0] [1] unroll[-1] y
```
**piechart**

**Arguments:**
- `label_height>=0,label_R,label_G,label_B,"label1",value1,R1,G1,B1,...,"labelN",valueN,RN,GN,BN`

**Description:**
Draw pie chart on selected (RGB) images.

**Example of use:**
```
image.jpg piechart
25,0,0,0,"Red",55,255,0,0,"Green",40,0,255,0,"Blue",30,128,128,255,"Other",5,12
```

---

**pixelize**

**Arguments:**
- `_scale_x>0,_scale_y>0,_scale_z>0`

**Description:**
Pixelize selected images with specified scales.

**Default values:**
scale_x=20 and scale_y=scale_z=scale_x.
Example of use:

```
image.jpg +pixelize ,
```

**pixelsort**

**Arguments:**

- `ordering={ + | - }`, `axis={ x | y | z | xy | yx }`, `[sorting_criterion]`, `[mask]`

**Description:**

Apply a pixel sorting algorithm on selected images, as described in the page:

http://satyarth.me/articles/pixel-sorting/.

**Default values:**

ordering=+, axis=x and sorting_criterion=mask=(undefined).

**Example of use:**

```
image.jpg +norm +ge[-1] 30% +pixelsort[0] +,y,[1],[2]
```
**plane3d**

**Arguments:**
- \texttt{size\_x, size\_y, nb\_subdivisions\_x>0, nb\_subdivisions\_y>0}

**Description:**
Input 3D plane at (0,0,0), with specified geometry.

**Default values:**
- \texttt{size\_x=1, size\_y=size\_x and nb\_subdivisions\_x=nb\_subdivisions\_y=24.}

**Example of use:**
```
plane3d 50,30 +primitives3d 1 color3d[-2] ${-rgb}
```

**plasma**

**Arguments:**
- \texttt{alpha, beta, scale>=0}

**Description:**
Draw a random colored plasma fractal on selected images.
This command implements the so-called Diamond-Square algorithm.

**Default values:**

alpha=1, beta=1 and scale=8.

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

**Example of use:**

```
400,400,1,3 plasma 1
```

---

**plot**

**Arguments:**

- `plot_type, vertex_type, xmin, xmax, ymin, ymax, exit_on_anykey={ 0 | 1 }`

**Description:**

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

- `plot_type` can be `{ 0: none | 1: lines | 2: splines | 3: bar }.
- `vertex_type` can be `{ 0: none | 1: points | 2,3: crosses | 4,5: circles | 6,7: squares }.
- `xmin, xmax, ymin, ymax` set the coordinates of the displayed xy-axes.

**Default values:**

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

---

**plot2value**

**No arguments**

**Description:**

Retrieve values from selected 2D graph plots.
Example of use:

```
400,300,1,1,'y>300*abs(cos(x/10+2*u))' +plot2value +display_graph[-1]
400,300
```

---

**poincare_disk**

**Arguments:**

- `size>=0, p>2, q>2, angle, tiling={0:triangular | 1:polygonal }, nb_max_iter>=0, xmin, ymin, xmax, ymax`

**Description:**

Return a 3-channels image of a poincare disk. Output channels are [x,y,it].

**Default values:**

- `size=1024, p=5, q=3, angle=0, tiling=0, nb_max_iter=20, xmin=ymin=-1 and xmax=ymax=1`. 

```
 repeat 4{ poincare_disk 1024,{3+}$\} channels[-1] 2 mod[-1] 3 neq[-1] 2 } rescale2d 50%
```

---

**point**

**Built-in command**

**Arguments:**

- `x[%, y[%, z[%, opacity, color1,...]`
**Description:**
Set specified colored pixel on selected images.

**Default values:**
z=0, opacity=1 and color1=0.

**Example of use:**

```plaintext
image.jpg repeat 10000 point {u(100)}%, {u(100)}%, 0, 1, ${-rgb} done
```

![Image](image.jpg)

**point3d**

**Arguments:**

- $x0, y0, z0$

**Description:**
Input 3D point at specified coordinates.

**Example of use:**

```plaintext
repeat 1000 { a:=$>*pi/500 point3d {cos(3*$a)}, {sin(2*$a)}, 0 color3d[-1] ${-rgb} } add3d
```

![Image](point3d.png)
pointcloud

Arguments:
- **type** = { -X:-X-opacity | 0:binary | 1:cumulative | 2:label | 3:retrieve coordinates }, _width, _height>0, _depth>0

Description:
Render a set of point coordinates, as a point cloud in a 1D/2D or 3D binary image (or do the reverse, i.e. retrieve coordinates of non-zero points from a rendered point cloud).

Input point coordinates can be a NxMx1x1, Nx1x1xM or 1xNx1xM image, where N is the number of points, and M the point coordinates.

If 'M'>3, the 3-to-M components sets the (M-3)-dimensional color at each point.

Parameters width,height and depth are related to the size of the final image:
- If set to 0, the size is automatically set along the specified axis.
- If set to N>0, the size along the specified axis is N.
- If set to N<0, the size along the specified axis is at most N.

Points with coordinates that are negative or higher than specified (width,height,depth) are not plotted.

Default values:
type=0 and max_width=max_height=max_depth=0.

Examples of use:
- **Example #1**
  
  ```
  3000,2 rand 0,400 +pointcloud 0 dilate[-1] 3
  ```

- **Example #2**
  
  ```
  3000,2 rand 0,400 {w} {w},3 rand[-1] 0,255 append y +pointcloud 0 dilate[-1] 3
  ```
pointcloud3d

No arguments

**Description:**
Convert selected planar or volumetric images to 3D point clouds.

**Example of use:**
```
image.jpg luminance rescale2d,100 threshold 50% mul 255 pointcloud3d
color3d[-1] 255,255,255
```

polar2complex

No arguments

**Description:**
Compute polar to complex transforms of selected images.

polar2euclidean

**Arguments:**
• \( \text{center}_x[\%], \text{center}_y[\%], \text{stretch}_factor>0, \text{boundary}_conditions=\{0:\text{dirichlet} | 1:\text{neumann} | 2:\text{periodic} | 3:\text{mirror} \} \)

**Description:**
Apply euclidean to polar transform on selected images.

**Default values:**
\( \text{center}_x=\text{center}_y=50\%, \text{stretch}_factor=1 \) and \( \text{boundary}_conditions=3 \).

**Example of use:**

```
image.jpg +euclidean2polar ,
```

---

**polaroid**

**Arguments:**

• \( size1\geq0, size2\geq0 \)

**Description:**
Create polaroid effect in selected images.

**Default values:**
\( size1=10 \) and \( size2=20 \).

**Example of use:**

```
image.jpg to_rgba polaroid 5,30 rotate 20 drop_shadow , drgba
```
polka_dots

Arguments:

• diameter>=0, _density, _offset1, _offset2, _angle, _aliasing, _shading, _opacity, _color...

Description:

Draw dots pattern on selected images.

Default values:

density=20, offset1=offset2=50, angle=0, aliasing=10, shading=1, opacity=1 and color=255.

Example of use:

```
image.jpg polka_dots 10,15,0,0,20,10,1,0.5,0,128,255
```

polygon

Arguments:

• N>=1, x1[%], y1[%], ..., xN[%], yN[%], _opacity, _pattern, _color1,...

Description:

Draw specified colored N-vertices polygon on selected images.
pattern is an hexadecimal number starting with 0x which can be omitted even if a color is specified. If a pattern is specified, the polygon is drawn outlined instead of filled. Adding a - sign before pattern makes the command draw an open polyline rather than a closed polygon.

**Default values:**

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

**Examples of use:**

- **Example #1**

  ```
  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 #1](image.jpg)

- **Example #2**

  ```
  image.jpg 2,16,1,1,'u(x?h#0:w#0)' polygon[-2] [-1],0.6,255,0,255
  remove[-1]
  ```

  ![Example #2](image.jpg)

**polygonize**

**Arguments:**

- `warp_amplitude>=0`, `smoothness[%]>=0`, `min_area[%]>=0`, `resolution_x[%]>0`, `resolution_y[%]>0`
**Description:**
Apply polygon effect on selected images.

**Default values:**
warp_amplitude=300, smoothness=2%, min_area=0.1%, resolution_x=resolution_y=10%.

**Example of use:**
```
image.jpg +polygonize 100,10 fill[-1] "I!=J(1) || I!=J(0,1)? [0,0,0]:I"
```

**portrait**

**Arguments:**
- ![size>0](#)

**Description:**
Input random portrait image of specified size, retrieved from Internet.

**Default values:**
size=800.

**pose3d**

**Arguments:**
- ![p1,...,p12](#)

**Description:**
Apply 3D pose matrix to selected 3D objects.

**Example of use:**
Torus 3D

Arguments:
- `torus3d 100,20 pose3d 0.152437,1.20666,-0.546366,0,-0.535962,0.559129,1.08531,0,1.21132,0.0955431,0.5`

Snapshot 3D

Arguments:
- `snapshot3d 400`

Poster Edges

Arguments:
- `poster_edges`

Description:
Apply poster edges effect on selected images.

Default values:
- `edge_threshold=40, edge_shade=5, edge_thickness=0.5, edge_antialiasing=10, posterization_level=12, posterization_antialiasing=0`

Example of use:
- `image.jpg poster_edges`

Poster Hope

Arguments:
• smoothness $\geq 0$

**Description:**

Apply Hope stencil poster effect on selected images.

**Default values:**

smoothness=3.

**Example of use:**

```markdown
image.jpg poster_hope ,
```

---

## pow

**Built-in command**

**Arguments:**

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

**Description:**

Raise selected images to the power of specified value, image or mathematical expression, or compute the pointwise sequential powers of selected images.

*(equivalent to shortcut command ^)*.

**Examples of use:**

- **Example #1**

```markdown
image.jpg div 255 +pow 0.5 mul 255
```
Example #2

image.jpg gradient pow 2 add pow 0.2

---

poweriteration

**Arguments:**

- \(\text{nb_eigenvectors} > 0, \epsilon > 0, \text{max_iter} > 0\)

**Description:**

Compute the \(\text{nb_eigenvectors}\) largest eigenvectors of the selected symmetric matrices, using the power iteration algorithm.

**Default values:**

\(\text{nb_eigenvectors}=1, \epsilon=1e-5\) and \(\text{max_iter}=100\).

---

premula

**No arguments**

**Description:**

Convert selected images with normal colors to premultiplied alpha colors.
After conversion, alpha channel of resulting images has value in \([0,1]\) range.

**See also:**

| ipremula |

---

**primitives3d**

**Arguments:**

- mode

**Description:**

Convert primitives of selected 3D objects.

*(equivalent to shortcut command* \(\text{p3d}\)).

mode can be \{ 0: points | 1: outlines | 2: non-textured \}.

**Example of use:**

```
  sphere3d 30 primitives3d 1 torus3d 50,10 color3d[-1] ${-rgb} add3d
```

![Image of primitives3d example](image.png)

---

**print**

**No arguments**

**Description:**

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

*(equivalent to shortcut command* \(\text{p}\)).

When invoked with a + prefix (i.e. +print), the command outputs on stdout rather than on stderr.
progress

Arguments:

- $0 \leq \text{value} \leq 100$
- $-1$

Description:
Set the progress index of the current processing pipeline.

This command is useful only when G'MIC is used by an embedding application.

projections3d

Arguments:

- $x[\%], y[\%], z[\%], \_\text{is\_bounding\_box} = \{ 0 \mid 1 \}, nb_{\_\text{subdivisions}} > 0$

Description:
Generate 3D xy,xz,yz projection planes from specified volumetric images.

Default values:
$x=y=z=50\%, \text{is\_bounding\_box}=1$ and $nb_{\_\text{subdivisions}}=5$

pseudogray

Arguments:

- $\_\text{max\_increment} \geq 0, \_\text{JND\_threshold} \geq 0, \_\text{bits\_depth} > 0$

Description:
Generate pseudogray colormap with specified increment and perceptual threshold.

If $\_\text{JND\_threshold}$ is 0, no perceptual constraints are applied.

Default values:
$max_{\_\text{increment}}=5, \_\text{JND\_threshold}=2.3$ and $\_\text{bits\_depth}=8$.

Example of use:

```
pseudogray 5
```
**psnr**

**Arguments:**

- ![reference], \( \text{max}\_\text{value}>0 \)

**Description:**

Return PSNR (Peak Signal-to-Noise Ratio) between selected images and specified reference image.

This command does not modify the images. It returns a value or a list of values in the status.

**Default values:**

\( \text{max}\_\text{value}=255. \)

**psnr\_matrix**

**Arguments:**

- \( \_\text{max}\_\text{value}>0 \)

**Description:**

Compute PSNR (Peak Signal-to-Noise Ratio) matrix between selected images.

**Default values:**

\( \text{max}\_\text{value}=255. \)

**Example of use:**

```
image.jpg +noise 30 +noise[0] 35 +noise[0] 38 cut. 0,255 +psnr\_matrix
```
puzzle

Arguments:
• \_width>0, \_height>0, \_M>=1, \_N>=1, \_curvature, \_centering, \_connectors\_variability, \_resolution>=1

Description:
Input puzzle binary mask with specified size and geometry.

Default values:
width=height=512, M=N=5, curvature=0.5, centering=0.5, connectors_variability=0.5 and resolution=64.

Example of use:

    puzzle ,
pyramid3d

Arguments:

• width, height

Description:

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

Example of use:

```
pyramid3d 100,-100 +primitives3d 1 color3d[-2] ${-rgb}
```

---

quadrangle3d

Arguments:

• x0,y0,z0,x1,y1,z1,x2,y2,z2,x3,y3,z3

Description:

Input 3D quadrangle at specified coordinates.

Example of use:

```
quadrangle3d -10,-10,10,10,-10,10,10,10,10,10,10,10 repeat 10 {
  +rotate3d[-1] 0,1,0,30 color3d[-1] ${-rgb},0.6 } add3d mode3d 2
```
quadratize_tiles

**Arguments:**

- $M > 0, \ N > 0$

**Description:**

Quadratize $M \times N$ tiles on selected images.

**Default values:**

$N = M$.

**Example of use:**

```
image.jpg +quadratize_tiles 16
```

---

quantize

**Arguments:**

- `nb_levels>=1`, `keep_values={ 0 | 1 }`, `quantization_type={ -1:median-cut | 0:k-means | 1:uniform }`

**Description:**

Quantize selected images.
Default values:

keep_values=1 and quantization_type=0.

Examples of use:

• Example #1

  image.jpg luminance +quantize 3

• Example #2

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

quantize_area

Arguments:
• \texttt{min\_area>0}

**Description:**
Quantize selected images such that each flat region has an area greater or equal to \texttt{min\_area}.

**Default values:**
\texttt{min\_area=10}.

**Example of use:**

\begin{verbatim}
image.jpg quantize 3 +blur 1 round[-1] +quantize_area[-1] 2
\end{verbatim}

---

**quit**

**No arguments**

**Description:**
Quit \texttt{G'MIC} interpreter.

(*equivalent to shortcut command *\texttt{q}.*).

---

**quiver**

**Arguments:**
- [function_image], _sampling[%]>0, _factor>=0, _is_arrow={ 0 | 1 }, _opacity, _color1,...

**Description:**

Draw specified 2D vector/orientation field on selected images.

**Default values:**

sampling=5%, factor=1, is_arrow=1, opacity=1, pattern=(undefined)

and color1=0.

**Examples of use:**

- **Example #1**

  \[
  100,100,1,2,'!c?x-w/2:y-h/2' 500,500,1,3,255 \text{ quiver[-1] [-2],10}
  \]

- **Example #2**

  \[
  \text{image.jpg +rescale2d ,600 luminance[0] gradient[0] mul[1] -1 reverse[0,1] append[0,1] c blur[0] 8 orientation[0] quiver[1] [0],20,1,1,0.8,255}
  \]

**rad2deg**

No arguments
Description:
Convert pointwise angle values of selected images, from radians to degrees (apply \(i \times 180/\pi\)).

---

**raindrops**

**Arguments:**
- \(\text{amplitude, density} \geq 0, \text{wavelength} \geq 0, \text{merging steps} \geq 0\)

**Description:**
Apply raindrops deformation on selected images.

**Default values:**
amplitude=80, density=0.1, wavelength=1 and merging steps=0.

**Example of use:**
```
image.jpg +raindrops ,
```

---

**rand**

**Arguments:**
- \({ \text{value0\%} | \text{image0} } , _{ \text{value1\%} | \text{image1} } , _{ \text{pdf} } , _{ \text{precision} } \) or
- \([\text{image}]\)

**Description:**
Fill selected images with random values in the specified range.

If no [pdf] (probability density function) is specified, random values follow a uniform distribution. Argument precision tells about the number of distinct values that can be generated when a [pdf] is specified.

**Examples of use:**
- Example #1
Example #2

(8,2,1) 50,50 rand[-1] 0,255,[-2]

Example #3

256 gaussian[-1] 30 line[-1] 47%,0,53%,0,1,0 500,500 rand[-1] 0,255,[-2] +histogram[-1] 256 display_graph[0,2] 640,480,3,0
**rand_sum**

**Arguments:**
- sum>0, random_function

**Description:**
Fill selected images with strictly positive, random, integer values, that sums to sum.

For each image, sum must be greater or equal than width*height*depth*spectrum.

**Default values:**
random_function=u.

**Example of use:**

```
100 rand_sum 1000
```

**random3d**

**Arguments:**
- nb_points>=0

**Description:**
Input random 3D point cloud in [0,1]^3.

**Example of use:**

```
random3d 100 circles3d 0.1 opacity3d 0.5
```
**random_clut**

**Arguments:**
- _seed = { >0 | -1 }

**Description:**
Generate a 33x33x33 random 3D color LUT.

If specified seed is positive, it is used as a seed for the random number generator @cli : (so that using the same seed will return the same CLUT).

**Example of use:**
```
image.jpg random_clut +map_clut...
```
**random_clut**

**Arguments:**

- \_seed = \{ \geq 0 \mid -1 \}

**Description:**

Generate a 33x33x33 random 3D color LUT.

If specified seed is positive, it is used as a seed for the random number generator @cli : (so that using the same seed will return the same CLUT).

**Example of use:**

```plaintext
image.jpg random_clut +map_clut.. .
```

**random_pattern**

**Arguments:**

- \width>0, \height>0, \min\_detail\_level\geq0

**Description:**

Insert a new RGB image of specified size at the end of the image list, rendered with a random pattern.

**Default values:**
width=height=512 and min_detail_level=2.

**Example of use:**

```plaintext
repeat 6 { random_pattern 256 }
```

---

**rbf**

**Arguments:**

- `dx, x0, x1, phi(r)` or
- `dx, dy, x0, y0, x1, y1, phi(r)` or
- `dx, dy, dz, x0, y0, z0, x1, y1, z1, phi(r)`

**Description:**

Reconstruct 1D/2D or 3D image from selected sets of keypoints, by RBF-interpolation.

A set of keypoints is represented by a vector-valued image, where each pixel represents a single keypoint.

Vector components of a keypoint have the following meaning:

- For 1D reconstruction: `[ x_k, f1(k),...,fN(k) ]`.
- For 2D reconstruction: `[ x_k, y_k, f1(k),...,fN(k) ]`.
- For 3D reconstruction: `[ x_k, y_k, z_k, f1(k),...,fN(k) ]`.

Values `x_k, y_k` and `z_k` are the spatial coordinates of keypoint `k`.
Values `f1(k),...,fN(k)` are the `N` components of the vector value of keypoint `k`.

The command reconstructs an image with specified size `dx'x'dy'x'dz`, with `N` channels.

**Default values:**
\(x_0=y_0=z_0=0, \ x_1=dx-1, \ y_1=dy-1, \ z_1=dz-1, \ \phi(r)=r^2\log(1e^{-5}+r)\).

**Examples of use:**

- **Example #1**

  ```plaintext
  sample colorful,400 100%,100% noise_poissondisk. 10 1,[is],1,5
  eval[-2] "begin(p=0);i?(I[#-1,p++]=[x,y,I(#0)])" to_rgb[1] mul[0,1]
  dilate_circ[0] 5 +rbf[-1] {0,[w,h]} c[-1] 0,255
  ```

- **Example #2**

  ```plaintext
  32,1,1,5,u([400,400,255,255,255]) rbf 400,400 c 0,255
  ```

---

**rectangle**

**Arguments:**

- \(x_0\%\), \(y_0\%\), \(x_1\%\), \(y_1\%\), _opacity, _pattern, _color1,...

**Description:**

Draw specified colored rectangle on selected images.

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

**Default values:**
Example of use:

```
image.jpg repeat 30 { rectangle {u(100)}%,{u(100)}%,{u(100)}%,
{u(100)}%,0.3,${-rgb} }
```

red_eye

**Arguments:**

- $0<= \text{threshold}<=100$, $\text{smoothness}>=0$, $0<=\text{attenuation}<=1$

**Description:**

Attenuate red-eye effect in selected images.

**Default values:**

threshold=75, smoothness=3.5 and attenuation=0.1.

**Example of use:**

```
image.jpg +red_eye
```

register_nonrigid
Arguments:

- [destination], _smoothness>=0, _precision>0, _nb_scale>=0

Description:

Register selected source images with specified destination image, using non-rigid warp.

Default values:

smoothness=0.2, precision=6 and nb_scale=0(auto).

Example of use:

image.jpg +rotate 20,1,1,50%,50% +register_nonrigid[0] [1]

---

register_rigid

Arguments:

- [destination], _smoothness>=0, _boundary_conditions={ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror }

Description:

Register selected source images with specified destination image, using rigid warp (shift).

Default values:

smoothness=0.1% and boundary_conditions=0.
Example of use:

image.jpg +shift 30,20 +register_rigid[0] [1]

remove

Built-in command

No arguments

Description:

Remove selected images.

(equivalent to shortcut command \texttt{rm}).

Examples of use:

• Example #1

image.jpg split x remove[30%-70%] append x
• Example #2

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

--

remove_copymark

No arguments

**Description:**
Remove copymark suffix in names of selected images.

remove_duplicates

No arguments

**Description:**
Remove duplicates images in the selected images list.

**Example of use:**

(1,2,3,4,2,4,3,1,3,4,2,1) split x remove_duplicates append x
remove_empty

No arguments

Description:
Remove empty images in the selected image list.

remove_hotpixels

Arguments:
- \_mask\_size>0, \_threshold[\%]>0

Description:
Remove hot pixels in selected images.

Default values:
mask\_size=3 and threshold=10%.

Example of use:

```plaintext
image.jpg noise 10,2 +remove_hotpixels ,
```

remove_named

Arguments:
- "name1", "name2", ...
**Description:**
Remove all images with specified names from the list of images. Does nothing if no images with those names exist. (*equivalent to shortcut command* `rmn`).

---

**remove_opacity**

*No arguments*

**Description:**
Remove opacity channel of selected images.

---

**remove_pixels**

**Arguments:**
- `number_of_pixels[%]>=0`

**Description:**
Remove specified number of pixels (i.e. set them to 0) from the set of non-zero pixels in selected images.

**Example of use:**

```markdown
image.jpg +remove_pixels 50%
```

---

**repeat**

**Built-in command**

**Arguments:**
- `nb_iterations`
Description:
Start nb_iterations iterations of a repeat...done block.
nb_iterations is a mathematical expression that will be evaluated.
This command has a tutorial page.

Examples of use:

• Example #1

```bash
image.jpg split y repeat $! n=>$ shift[$n] $<,0,0,0,2 done append y
```

![Example #1 Image](image.jpg)

• Example #2

```bash
image.jpg mode3d 2 repeat 4 imagecube3d rotate3d 1,1,0,40 snapshot3d 400,1.4 done
```

![Example #2 Image](image.jpg)

replace

Arguments:

• source,target

Description:
Replace pixel values in selected images.
Example of use:

(1;2;3;4) +replace 2,3

replace_color

Arguments:

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

Description:

Replace pixels from/to specified colors in selected images.

Example of use:

image.jpg +replace_color 40,3,204,153,110,255,0,0

replace_inf

Arguments:

• expression

Description:
Replace all infinite values in selected images by specified expression.

Example of use:

\[(0;1;2) \text{ log } +\text{replace}_\text{inf} \ 2\]

---

replace\_infnan

Arguments:

- \text{expression}

Description:

Replace all NaN and infinite values in selected images by specified expression.

---

replace\_nan

Arguments:

- \text{expression}

Description:

Replace all NaN values in selected images by specified expression.

Example of use:

\[(-1;0;2) \text{ sqrt } +\text{replace}_\text{nan} \ 2\]
**replace_seq**

**Arguments:**

- "search_seq","replace_seq"

**Description:**

Search and replace a sequence of values in selected images.

**Example of use:**

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

**replace_str**

**Arguments:**

- "search_str","replace_str"

**Description:**

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

**Example of use:**

```
('"Hello there, how are you ?"') +replace_str "Hello there","Hi
```
rescale2d

Arguments:

- \( width[\%] = \{ 0: \text{any} \mid >0 \}, height[\%] = \{ 0: \text{any} \mid >0 \} \), \(-1 = \leq \text{interpolation} \leq 6\), \( \text{mode} = \{ 0: \text{inside} \mid 1: \text{padded-inside} \mid 2: \text{outside} \mid 3: \text{cropped-outside} \}\)

Description:

Resize selected 2D images while preserving aspect ratio.

Interpolation can be \( \{ -1: \text{status only} \mid 0: \text{none} \mid 1: \text{nearest} \mid 2: \text{average} \mid 3: \text{linear} \mid 4: \text{grid} \mid 5: \text{bicubic} \mid 6: \text{lanczos} \} \).

When \( \text{interpolation} = -1 \), image size is actually not modified, but the size that would have been used for the last selected image is returned in the status value.

Each resized image size is computed according to the specified mode:
- If \( \text{mode} = 0 \), image size is \textbf{at most} \((\text{width}, \text{height})\).
- If \( \text{mode} = 1 \) or \( \text{mode} = 3 \), image size is \textbf{exactly} \((\text{width}, \text{height})\).
- If \( \text{mode} = 2 \), image size is \textbf{at least} \((\text{width}, \text{height})\).

\(\text{equivalent to shortcut command} \ |rs|\).

Default values:

width=height=0, interpolation=2 and mode=0.

rescale3d

Arguments:

- \( width[\%] = \{ 0: \text{any} \mid >0 \}, height[\%] = \{ 0: \text{any} \mid >0 \}, depth[\%] = \{ 0: \text{any} \mid >0 \} \), \(-1 = \leq \text{interpolation} \leq 6\), \( \text{mode} = \{ 0: \text{inside} \mid 1: \text{padded-inside} \mid 2: \text{outside} \mid 3: \text{cropped-outside} \}\)
**Description:**
Resize selected 3D images while preserving aspect ratio.

Interpolation can be \{-1:status only | 0:none | 1:nearest | 2:average | 3:linear | 4=grid | 5=bicubic | 6=lanczos \}.

When interpolation=-1, image size is actually not modified, but the size that would have been used for the last selected image is returned in the status value.

Each resized image size is computed according to the specified mode:
- If mode=0, image size is **at most** (width, height).
- If mode=1 or mode=3, image size is **exactly** (width, height).
- If mode=2, image size is **at least** (width, height).

*(equivalent to shortcut command \texttt{rs3d}).*

**Default values:**
width=height=depth=0, interpolation=2 and mode=0.

---

**reset**

**No arguments**

**Description:**
Reset global parameters of the interpreter environment.

---

**resize**

**Arguments:**
- \{[[image\_w] | width>0\[%\]}, [[image\_h] | height>0\[%\]}, [[image\_d] | depth>0\[%\]}, [[image\_s] | spectrum>0\[%\]}, interpolation, boundary_conditions, ax, ay, az, ac

**Description:**
Resize selected images with specified geometry.

*(equivalent to shortcut command \texttt{r})*.

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

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

\texttt{ax,ay,az,ac} set the centering along each axis when interpolation=0 or 4 (set to 0 by default, must be defined in range [0,1]).
**Default values:**

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

**Example of use:**

```bash
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
```

**resize_as_image**

**Arguments:**

- [reference], _interpolation, _boundary_conditions, _ax, _ay, _az, _ac

**Description:**

Resize selected images to the geometry of specified [reference] image.

(equivalent to shortcut command `ri`).

**Default values:**
interpolation=1, boundary_conditions=0 and ax=ay=az=ac=0.

**Example of use:**

```
image.jpg sample duck +resize_as_image[-1] [-2]
```

---

**resize_mn**

**Arguments:**

- `width[%]>=0,_height[%]>=0,_depth[%]>=0,_B_value,_C_value`

**Description:**

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

For details about the method, see: [https://de.wikipedia.org/wiki/Mitchell-Netravali-Filter](https://de.wikipedia.org/wiki/Mitchell-Netravali-Filter).

**Default values:**

height=100%, depth=100%, B=0.3333 and C=0.3333.

**Example of use:**

```
image.jpg rescale2d 32 resize_mn 800%,800%
```
resize_pow2

**Arguments:**

- `interpolation, boundary_conditions, ax, ay, az, ac`

**Description:**

Resize selected images so that each dimension is a power of 2.

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

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

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

**Default values:**

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

**Example of use:**

```
image.jpg +resize_pow2[-1] 0
```
retinex

Arguments:

- \_value\_offset>0, \_colorspace={hsi | hsv | lab | lrgb | rgb | ycbcr}, 0<=\_min\_cut<=100, 0<=\_max\_cut<=100, \_sigma\_low>0, \_sigma\_mid>0, \_sigma\_high>0

Description:
Apply multi-scale retinex algorithm on selected images to improve color consistency.

Default values:
offset=1, colorspace=hsv, min_cut=1, max_cut=1, sigma_low=15, sigma_mid=80 and sigma_high=250.

---

return

Built-in command

No arguments

Description:
Return from current custom command.

---

reverse

Built-in command

No arguments

Description:
Reverse positions of selected images.

(equivalent to shortcut command \textit{rv}).

Examples of use:

- Example #1

```
image.jpg split x,3 reverse[-2,-1]
```
**Example #2**

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

---

**reverse3d**

**No arguments**

**Description:**
Reverse primitive orientations of selected 3D objects.

*(equivalent to shortcut command* `rv3d`*).*

**Example of use:**

```
torus3d 100,40 double3d 0 +reverse3d
```
**rgb**

No arguments

**Description:**

Return a random int-valued RGB color.

---

**rgb2bayer**

**Arguments:**

- `start_pattern=0, color_grid=0`

**Description:**

Transform selected color images to RGB-Bayer sampled images.

**Default values:**

`start_pattern=0` and `color_grid=0`.

**Example of use:**

```plaintext
image.jpg +rgb2bayer 0
```

---

**rgb2cmy**

No arguments

**Description:**

Convert color representation of selected images from RGB to CMY.

**Example of use:**
rgb2cmyk

No arguments

Description:
Convert color representation of selected images from RGB to CMYK.

Examples of use:
• Example #1

image.jpg rgb2cmy split c
- Example #2

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

**rgb2hcy**

No arguments

**Description:**
Convert color representation of selected images from RGB to HCY.

**Example of use:**

    ```
    image.jpg rgb2hcy split c
    ```
rgb2hsi

No arguments

Description:
Convert color representation of selected images from RGB to HSI.

Example of use:

```plaintext
image.jpg rgb2hsi split c
```

rgb2hsi8

No arguments
Description:
Convert color representation of selected images from RGB to HSI8.

Example of use:

```
image.jpg rgb2hsi8 split c
```

---

**rgb2hsl**

No arguments

Description:
Convert color representation of selected images from RGB to HSL.

Examples of use:

• Example #1

```
image.jpg rgb2hsl split c
```
Example #2

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

rgb2hsl8

**No arguments**

**Description:**

Convert color representation of selected images from RGB to HSL8.

**Example of use:**

```
image.jpg rgb2hsl8 split c
```
rgb2hsv

No arguments

**Description:**
Convert color representation of selected images from RGB to HSV.

**Examples of use:**

• Example #1

```
image.jpg rgb2hsv split c
```
Example #2

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

**rgb2hsv8**

No arguments

**Description:**
Convert color representation of selected images from RGB to HSV8.

**Example of use:**

```plaintext
image.jpg rgb2hsv8 split c
```
rgb2int

No arguments

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

Example of use:

```
image.jpg rgb2int
```

rgb2jzazbz

Arguments:

- `illuminant={ 0:D50 | 1:D65 | 2:E }` or `no_arg`

Description:
Convert color representation of selected images from RGB to Jzazbz.

Default values:
illuminant=2.
rgb2lab

**Arguments:**

- `illuminant={ 0:D50 | 1:D65 | 2:E }` or
- `no arg`

**Description:**

Convert color representation of selected images from RGB to Lab.

**Default values:**

`illuminant=2`.

rgb2lab8

**Arguments:**

- `illuminant={ 0:D50 | 1:D65 | 2:E }` or
- `no arg`

**Description:**

Convert color representation of selected images from RGB to Lab8.

**Default values:**

`illuminant=2`.

**Example of use:**

```shell
image.jpg rgb2lab8 split c
```
rgb2lch

Arguments:
- \textbf{illuminant}\{ 0:D50 | 1:D65 | 2:E \} or
- (no arg)

Description:
Convert color representation of selected images from RGB to Lch.

Default values:
illuminant=2.

Example of use:

\texttt{image.jpg rgb2lch split c}
rgb2lch8

Arguments:

• illuminant={ 0:D50 | 1:D65 | 2:E } or
• (no arg)

Description:

Convert color representation of selected images from RGB to Lch8.

Default values:

illuminant=2.

Example of use:

image.jpg rgb2lch8 split c

rgb2luv

No arguments

Description:

Convert color representation of selected images from RGB to LUV.

Example of use:
rgb2oklab

No arguments

Description:
Convert color representation of selected images from RGB to Oklab.

(see colorspace definition at: https://bottosson.github.io/posts/oklab/).

See also:
oklab2rgb.

rgb2ryb

No arguments

Description:
Convert color representation of selected images from RGB to RYB.

Example of use:
image.jpg rgb2ryb split c
rgb2srgb

No arguments

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

rgb2xyz

Arguments:
- illuminant={ 0:D50 | 1:D65 | 2:E } or (no arg)

Description:
Convert color representation of selected images from RGB to XYZ.

Default values:
illuminant=2.

Example of use:
image.jpg rgb2xyz split c
rgb2xyz8

**Arguments:**

- illuminant={ 0:D50 | 1:D65 | 2:E }  or
- (no arg)

**Description:**

Convert color representation of selected images from RGB to XYZ8.

**Default values:**

illuminant=2.

**Example of use:**

```
image.jpg rgb2xyz8 split c
```
**rgb2ycbcr**

No arguments

**Description:**
Convert color representation of selected images from RGB to YCbCr.

**Example of use:**

```
image.jpg rgb2ycbcr split c
```

---

**rgb2yiq**

No arguments
Description:
Convert color representation of selected images from RGB to YIQ.

Example of use:

```
image.jpg rgb2yiq split c
```

---

**rgb2yiq8**

No arguments

Description:
Convert color representation of selected images from RGB to YIQ8.

Example of use:

```
image.jpg rgb2yiq8 split c
```
rgb2yuv

No arguments

Description:
Convert color representation of selected images from RGB to YUV.

Example of use:

```bash
image.jpg rgb2yuv split c
```
**rgb2yuv8**

No arguments

**Description:**
Convert color representation of selected images from RGB to YUV8.

**Example of use:**

```
image.jpg rgb2yuv8 split c
```

**rgba**

No arguments
Description:
Return a random int-valued RGBA color.

**ripple**

**Arguments:**
- \[\text{amplitude, bandwidth, shape} = \{0:block, 1:triangle, 2:sine, 3:sine+, 4:random\}, \angle, \text{offset}\]

**Description:**
Apply ripple deformation on selected images.

**Default values:**
amplitude=10, bandwidth=10, shape=2, angle=0 and offset=0.

**Example of use:**
```
image.jpg +ripple ,
```

---

**rodilius**

**Arguments:**
- \[0<=\text{amplitude}<=100, 0<=\text{thickness}<=100, \text{sharpness}>=0, \text{nb orientations}>0, \text{offset}, 0:darken | 1:brighten\]

**Description:**
Apply rodilius (fractalius-like) filter on selected images.

**Default values:**
amplitude=10, thickness=10, sharpness=400, nb_orientations=7, offset=0 and color_mode=1.

**Examples of use:**
• Example #1

image.jpg rodilius 12,10,300,10 normalize_local 10,6

• Example #2

image.jpg normalize_local 10,16 rodilius 10,4,400,16 smooth 60,0,1,4 normalize_local 10,16

**rol**

**Arguments:**

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

**Description:**

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

**Example of use:**

image.jpg rol 'round(3*x/w,0)' cut 0,255
rolling_guidance

Arguments:

• std_deviation_s[\%] >= 0, std_deviation_r[\%] >= 0, precision >= 0

Description:

Apply the rolling guidance filter on selected image.

Rolling guidance filter is a fast image abstraction filter, described in: "Rolling Guidance Filter", Qi Zhang Xiaoyong, Shen Li, Xu Jiaya Jia, ECCV'2014.

Default values:

std_deviation_s=4, std_deviation_r=10 and precision=0.5.

Example of use:

image.jpg +rolling_guidance , +
ror

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

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

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

rorschach

Arguments:
- `'smoothness[%]>=0'`, `'mirroring={ 0:none | 1:x | 2:y | 3:xy }`

Description:
Render rorschach-like inkblots on selected images.
Default values:

smoothness=5% and mirroring=1.

Example of use:

400,400 rorschach 3%

---

**rotate**

**Built-in command**

**Arguments:**

• angle, interpolation, boundary_conditions, center_x[%], center_y[%] or 
  u,v,w,angle,interpolation,boundary_conditions,center_x[%],center_y[%],_center_z[%]

**Description:**

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

interpolation can be { 0: none | 1: linear | 2: bicubic }.
boundary_conditions can be { 0: dirichlet | 1: neumann | 2: periodic | 3: mirror }.
When a rotation center (cx, cy, _cz) is specified, the size of the image is preserved.

**Default values:**

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

**Example of use:**

image.jpg +rotate -25,1,2,50%,50% rotate[0] 25
rotate3d

**Arguments:**

- u, v, w, angle

**Description:**

Rotate selected 3D objects around specified axis with specified angle (in deg.).

*(equivalent to shortcut command r3d).*

**Example of use:**

```
 torus3d 100,10 double3d 0 repeat 7 { +rotate3d[-1] 1,0,0,20 } add3d
```

rotate_tileable

**Arguments:**

- angle, max_size_factor>=0

**Description:**

Rotate selected images by specified angle and make them tileable.

If resulting size of an image is too big, the image is replaced by a 1x1 image.
Default values:
max_size_factor=8.

rotate_tiles

Arguments:
• angle, M>0, N>0

Description:
Apply MxN tiled-rotation effect on selected images.

Default values:
M=8 and N=M.

Example of use:
image.jpg to_rgba rotate_tiles 10,8 drop_shadow 10,10 display_rgba

rotation3d

Arguments:
• u,v,w,angle

Description:
Input 3x3 rotation matrix with specified axis and angle (in deg).

Example of use:
rotation3d 1,0,0,0 rotation3d 1,0,0,90 rotation3d 1,0,0,180
rotoidoscope

Arguments:

- _center_x[%],_center_y[%],_tiles>0,_smoothness[%]>=0,_boundary_conditions={0:dirichlet | 1:neumann | 2:periodic | 3:mirror }

Description:
Create rotational kaleidoscope effect from selected images.

Default values:
center_x=center_y=50%, tiles=10, smoothness=1 and boundary_conditions=3.

Example of use:

image.jpg +rotoidoscope ,

round

Arguments:

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

Description:
Round values of selected images.

rounding_type can be \{-1:backward \mid 0:nearest \mid 1:forward \}.

**Default values:**

rounding_type=0.

**Examples of use:**

- **Example #1**

  ```
  image.jpg +round 100
  ```

- **Example #2**

  ```
  image.jpg mul {\pi/180} sin +round
  ```

---

**roundify**

**Arguments:**

- \textit{gamma}>=0

**Description:**

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

**Default values:**
gamma=0.

**Example of use:**

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

**rows**

**Arguments:**

- \textit{y0[\%], y1[\%], boundary_conditions}

**Description:**

Keep only specified rows of selected images.

boundary_conditions can be \{ \textit{0:dirichlet | 1:neumann | 2:periodic | 3:mirror} \}.

**Default values:**

\(y1=y0\) and \textit{boundary_conditions}=0.

**Example of use:**

```
image.jpg rows -25\%,50\%
```

**rprogress**
Arguments:

- \(0\leq\text{value}\leq100\) | -1 | "command", \(0\leq\text{value_min}\leq100, 0\leq\text{value_max}\leq100\)

Description:

Set the progress index of the current processing pipeline (relatively to previously defined progress bounds), or call the specified command with specified progress bounds.

---

**run**

Arguments:

- "G'MIC pipeline"

Description:

Run specified G'MIC pipeline.

---

**ryb2rgb**

No arguments

Description:

Convert color representation of selected images from RYB to RGB.

---

**sample**

Arguments:

- \_name1={ ? | apples | balloons | barbara | boats | bottles | butterfly | cameraman | car | cat | cliff | chick | colorful | david | dog | duck | eagle | elephant | earth | flower | fruits | gmicky | gmicky_mahvin | gmicky_wilber | greece | gummy | house | inside | landscape | leaf | lena | leno | lion | mandrill | monalisa | monkey | parrots | pencils | peppers | portrait0 | portrait1 | portrait2 | portrait3 | portrait4 | portrait5 | portrait6 | portrait7 | portrait8 | portrait9 | roddy | rooster | rose | square | swan | teddy | tiger | tulips | wall | waterfall | zelda }, \_name2,...,\_nameN,\_width={ >=0 | 0 (auto) }, \_height = { >=0 | 0 (auto) }
  - or
  - (no_arg)

Description:

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

*(equivalent to shortcut command 'sp').*
Argument name can be replaced by an integer which serves as a sample index.

Example of use:

```
repeat 6 { sample }
```

---

**scale2x**

No arguments

**Description:**

Resize selected images using the Scale2x algorithm.

**Example of use:**

```
image.jpg threshold 50% resize 50%,50% +scale2x
```
scale3x

No arguments

Description:
Resize selected images using the Scale3x algorithm.

Example of use:

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

scale_dcci2x

Arguments:
- `edge_threshold>=0, _exponent>0, _extend_1px={ 0:false | 1:true }

Description:
Double image size using directional cubic convolution interpolation,

Default values:

edge_threshold=1.15, exponent=5 and extend_1px=0.
Example of use:

image.jpg +scale_dcci2x ,

scanlines

Arguments:

- \_amplitude, \_bandwidth, \_shape={ 0:block | 1:triangle | 2:sine | 3:sine+ | 4:random }, \_angle, \_offset

Description:

Apply ripple deformation on selected images.

Default values:

amplitude=60, bandwidth=2, shape=0, angle=0 and offset=0.

Example of use:

image.jpg +scanlines ,

screen

Arguments:
• \texttt{x0[\%], y0[\%], x1[\%], y1[\%]}

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

---

**seamcarve**

**Arguments:**

- \texttt{width[\%] \geq 0, height[\%] \geq 0, is\_priority\_channel={ 0 | 1 }, is\_antialiasing={ 0 | 1 }, maximum\_seams[\%] \geq 0}

**Description:**
Resize selected images with specified 2D geometry, using the seam-carving algorithm.

**Default values:**

height=100\%, is\_priority\_channel=0, is\_antialiasing=1 and maximum\_seams=25\%.

**Example of use:**

```plaintext
image.jpg seamcarve 60%
```

---

**segment\_watershed**

**Arguments:**

- \texttt{threshold \geq 0}

**Description:**
Apply watershed segmentation on selected images.

**Default values:**

threshold=2.
Example of use:

```
image.jpg segment_watershed 2
```

---

**select**

**Arguments:**

- `feature_type, X[%]=0, Y[%]=0, Z[%]=0, exit_on_anykey={0 | 1}, is_multiaxes_selection={0 | 1}`

**Description:**

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

- `feature_type` can be `{0:point | 1:segment | 2:rectangle | 3:ellipse}`.
- Arguments `X,Y,Z` determine the initial selection view, for 3D volumetric images.
- The retrieved feature is returned as a 3D vector (if `feature_type==0`) or as a 6d vector (if `feature_type!=0`) containing the feature coordinates.

**Default values:**

- `feature_type=2, X=Y=Z=50%, exit_on_anykey=0 and is_multiaxes_selection=1`.

---

**select_color**

**Arguments:**

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

**Description:**

Select pixels with specified color in selected images.

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

**Example of use:**
**sephia**

**No arguments**

**Description:**
Apply sephia tones effect on selected images.

**Example of use:**

image.jpg sephia

**serialize**

**Arguments:**
- **datatype, is_compressed={ 0 | 1 }, store_names={ 0 | 1 }**

**Description:**
Serialize selected list of images into a single image, optionally in a compressed form.

datatype can be { auto | uint8 | int8 | uint16 | int16 | uint32 | int32 | uint64 | int64 | float32 | float64 }.
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 integer values in \([0,255]\) and can then be saved as a raw image of unsigned chars (doing so will output a valid .cimg[z] or .gmz file). If store_names is set to 1, serialization uses the .gmz format to store data in memory (otherwise the .cimg[z] format).

**Default values:**

datatype=auto, is_compressed=1 and store_names=1.

**Example of use:**

```
image.jpg +serialize uint8 +unserialize[-1]
```

---

**set**

**Built-in command**

**Arguments:**

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

**Description:**

Set pixel value in selected images, at specified coordinates.

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

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

**Default values:**
Examples of use:

• Example #1

```
2,2 set 1,0,0 set 2,1,0 set 3,0,1 set 4,1,1
```

• Example #2

```
image.jpg repeat 10000 { set 255,{u(100)}%,{u(100)}%,0,{u(100)}% }
```

shade_stripes

Arguments:

- `frequency>=0`, `direction={0:horizontal | 1:vertical}`, `darkness>=0`, `lightness>=0`

Description:

Add shade stripes to selected images.

Default values:

- frequency=5, direction=1, darkness=0.8 and lightness=2.

Example of use:
**shadow_patch**

**Arguments:**
- `opacity>=0`

**Description:**
Add shadow patches to selected images.

**Default values:**
opacity=0.7.

**Example of use:**

```
image.jpg +shadow_patch 0.4
```

**shape2bump**

**Arguments:**
- `resolution>=0`, `0<= weight_std_max_avg<=1`, `dilation`, `smoothness>=0`
Description:
Estimate bumpmap from binary shape in selected images.

Default values:
resolution=256, weight_std_max=0.75, dilation=0 and smoothness=100.

---

**shape_circle**

**Arguments:**
- `size>=0`

**Description:**
Input a 2D circle binary shape with specified size.

**Default values:**
size=512.

**Example of use:**
```plaintext
shape_circle
```

![Circle Image](image)

---

**shape_cupid**

**Arguments:**
- `size>=0`

**Description:**
Input a 2D cupid binary shape with specified size.

**Default values:**
size=512.
Example of use:

```
shape_cupid ,
```

![Shape Cupid](image1)

---

**shape_diamond**

**Arguments:**

- `size >= 0`

**Description:**

Input a 2D diamond binary shape with specified size.

**Default values:**

`size = 512`.

**Example of use:**

```
shape_diamond ,
```

![Shape Diamond](image2)

---

**shape_dragon**

**Arguments:**
• \texttt{size=0, recursion_level=0, angle=0}

**Description:**

Input a 2D Dragon curve with specified size.

**Default values:**

size=512, recursion_level=18 and angle=0.

**Example of use:**

```shape_dragon
```

---

**shape_fern**

**Arguments:**

• \texttt{size=0, density[\%]=0, angle, 0<=\_opacity<=1, _type={ 0: Asplenium adiantum-nigrum | 1: Thelypteridaceae }}

**Description:**

Input a 2D Barnsley fern with specified size.

**Default values:**

size=512, density=50\%, angle=30, opacity=0.3 and type=0.

**Example of use:**

```shape_fern```
shape_gear

Arguments:

- \_size\geq 0, \_nb\_teeth\geq 0, 0\leq \_height\_teeth\leq 100, 0\leq \_offset\_teeth\leq 100, 0\leq \_inner\_radius

Description:

Input a 2D gear binary shape with specified size.

Default values:

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

Example of use:

```
shape_gear ,
```

shape_heart

Arguments:

- \_size\geq 0

Description:

Input a 2D heart binary shape with specified size.
Default values:
size=512.

Example of use:

```
shape_heart ,
```

---

**shape_menger**

**Arguments:**

- \( nb\_\text{iterations} \geq 0 \)

**Description:**
Input a 3D voxelized representation of the Menger sponge.

**Default values:**

\( nb\_\text{iterations} = 3 \).

**Example of use:**

```
shape_menger 4 surfels3d , color3d 200 m3d 3
```
shape_mosely

**Arguments:**
- `nb_iterations>=0`

**Description:**
Input a 3D voxelized representation of the Mosely snowflake.

**Default values:**
`nb_iterations=3`.

**Example of use:**
```
shape_mosely 4 surfels3d, color3d 200 m3d 3
```

---

shape_polygon

**Arguments:**
- `size>=0, nb_vertices>=3, angle`

**Description:**
Input a 2D polygonal binary shape with specified geometry.

**Default values:**
`size=512, nb_vertices=5 and angle=0`.

**Example of use:**
```
repeat 6 { shape_polygon 256, {3+} }
```
shape_rays

Arguments:

• \_size>=0, _xcenter[\%], _ycenter[\%], _branches>0, _angle[\%], _twist, 0<=_perspective<=1, _is_antialias \in \{0 | 1\}

Description:

Input a 3D binary spiral with specified size and attributes.

Default values:

size=512, xcenter=50\%, ycenter=50\%, branches=7, angle=50\%, twist=0, perspective=0.35 and is_antialias=0.

Example of use:

| shape_rays 400,50\%,50\%,7  shape_rays 400,50\%,50\%,3,0,3 |
shape_snowflake

Arguments:

- \texttt{size} \geq 0, 0 \leq \texttt{nb\_recursions} \leq 6

Description:

Input a 2D snowflake binary shape with specified size.

Default values:

\texttt{size}=512 and \texttt{nb\_recursions}=5.

Example of use:

\begin{verbatim}
repeat 6 { shape_snowflake 256, $> }
\end{verbatim}

shape_star

Arguments:

- \texttt{size} \geq 0, \texttt{nb\_branches} > 0, 0 \leq \texttt{thickness} \leq 1

Description:

Input a 2D star binary shape with specified size.
Default values:
size=512, nb_branches=5 and thickness=0.38.

Example of use:

```plaintext
repeat 9 { shape_star 256,{$>+2} }
```

Arguments:

- `x0[%],x1[%],y[%],z[%],c[%]` or
- `y0[%],y1[%],z[%],c[%]` or
- `z0[%],z1[%],c[%]` or
- `c0[%],c1[%]` or
- `c0[%]` or
- `(no arg)`

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

Shared buffers cannot be returned by a command, nor a local environment.  
(equivalent to shortcut command 'sh').

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

**Examples of use:**

- **Example #1**

  ```
  image.jpg shared 1 blur[-1] 3 remove[-1]
  ```

- **Example #2**

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

---

**sharpen**

**Arguments:**

- amplitude>=0
- amplitude>=0,edge>=0,_alpha[%],_sigma[%]

**Description:**

Sharpen selected images by inverse diffusion or shock filters methods.
edge must be specified to enable shock-filter method.

**Default values:**
edge=0, alpha=0 and sigma=0.

**Examples of use:**
- **Example #1**
  ```plaintext
  image.jpg sharpen 300
  ```

- **Example #2**
  ```plaintext
  image.jpg blur 5 sharpen 300,1
  ```

---

**sharpen_alpha**

**Arguments:**
- `amplitude[%]>=0, nb_scales>0,0<=anisotropy<=1,0<=minimize_alpha<=1`

**Description:**
Sharpen selected images using a multi-scale and alpha boosting algorithm.

**Default values:**
amplitude=1, nb_scales=5, anisotropy=0 and minimize_alpha=1.
shell_cols

No arguments

Description:
Return the estimated number of columns of the current shell.

shift

Built-in command

Arguments:

• vx[%, vy[%, vz[%, vc[%, boundary_conditions, interpolation={
  0:nearest_neighbor | 1:linear }

Description:
Shift selected images by specified displacement vector.

Displacement vector can be non-integer in which case linear interpolation should be chosen.
boundary_conditions can be { 0:dirichlet | 1:neumann | 2:periodic | 3:mirror }.

Default values:
boundary_conditions=0 and interpolation=0.

Example of use:

image.jpg +shift[0] 50%,50%,0,0,0 +shift[0] 50%,50%,0,0,1 +shift[0]
50%,50%,0,0,2
shift_tiles

Arguments:
- M>0, N>0, amplitude

Description:
Apply MxN tiled-shift effect on selected images.

Default values:
N=M and amplitude=20.

Example of use:
```
image.jpg +shift_tiles 8,8,10
```

shrink

Arguments:
- axes, size[\%], boundary_conditions={ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror }

Description:
Shrink selected images along the specified axes.
axes can be \{ x | y | z | c | xy | xz | xc | yz | yc | zc | xyz | xyc | xzc | yzc | xyzc \}.

Default values:

boundary_conditions=0.

Example of use:

```
image.jpg shrink xy,100
```

---

sierpinski

**Arguments:**

- \texttt{recursion\_level}>=0

**Description:**

Draw Sierpinski triangle on selected images.

**Default values:**

recursion\_level=7.

**Example of use:**

```
image.jpg sierpinski 7
```
sierpinski3d

Arguments:
• _recursion_level>=0,_width,_height

Description:
Input 3D Sierpinski pyramid.

Example of use:

```
sierpinski3d 3,100,-100 +primitives3d 1 color3d[-2] ${-rgb}
```

--

sign

No arguments

Description:
Compute the pointwise sign of selected images.

Examples of use:

• Example #1

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

--
• Example #2

\[
300,1,1,1,'\cos(20*x/w+u)' +\text{sign} \quad \text{display_graph} \quad 400,300
\]

sin

Built-in command
No arguments

Description:
Compute the pointwise sine of selected images.
This command has a tutorial page.

Examples of use:

• Example #1

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

• Example #2

\[
300,1,1,1,'20*x/w+u' +\sin \quad \text{display_graph} \quad 400,300
\]
**sinc**

*Built-in command*

*No arguments*

**Description:**

Compute the pointwise sinc function of selected images.

**Examples of use:**

- **Example #1**

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

- **Example #2**

  ```plaintext
  300,1,1,1,'20*x/w+u' +sinc display_graph 400,300
  ```
sinh

No arguments

Description:
Compute the pointwise hyperbolic sine of selected images.

Examples of use:
• Example #1
   
   image.jpg +normalize -3,3 sinh[-1]

• Example #2
   
   300,1,1,1,'4*x/w+u' +sinh display_graph 400,300

size3d

No arguments

Description:
Return bounding box size of the last selected 3D object.
size_value

No arguments

Description:
Return the size (in bytes) of image values.

skeleton

Arguments:
- \texttt{boundary_conditions=\{ 0:dirichlet | 1:neumann \}}

Description:
Compute skeleton of binary shapes using distance transform and constrained thinning.

Default values:
boundary_conditions=1.

Example of use:

\begin{verbatim}
shape_cupid 320 +skeleton 0
\end{verbatim}

skeleton3d

Arguments:
- \texttt{metric, frame_type=\{ 0:squares | 1:diamonds | 2:circles | 3:auto \}}, \texttt{skeleton_opacity, frame_opacity, is_frame_wireframe=\{ 0 | 1 \}}

Description:
Build 3D skeletal structure object from 2d binary shapes located in selected images.
metric can be \{ \texttt{0:chebyshev} \mid \texttt{1:manhattan} \mid \texttt{2:euclidean} \}.

**Default values:**

metric=2, bones_type=3, skeleton_opacity=1 and frame_opacity=0.1.

**Example of use:**

```
shape_cupid 480 +skeleton3d ,
```

---

**sketchbw**

**Arguments:**

- \_nb_angles>0\_, \_start_angle\_, \_angle_range\_>=0\_, \_length\_>=0\_, \_threshold\_>=0\_, \_opacity\_, \_bg\_factor\_>=0\_, \_density\_=0\_, \_sharpness\_=0\_.
- \_is_curved\_={ 0 \mid 1 }\_, \_is_boost\_={ 0 \mid 1 }\_,

**Description:**

Apply sketch effect to selected images.

**Default values:**

nb_angles=2, start_angle=45, angle_range=180, length=30, threshold=3, opacity=0.03, bgfactor=0, density=0.6, sharpness=0.1, anisotropy=0.6, smoothness=0.25, coherence=1, is_boost=0 and is_curved=1.

**Example of use:**

```
image.jpg +sketchbw 1 reverse blur[-1] 3 blend[-2,-1] overlay
```
**skip**

**Built-in command**

**Arguments:**

- item

**Description:**

Do nothing but skip specified item.

---

**slic**

**Arguments:**

- size>0, _regularity>=0, _nb_iterations>0

**Description:**

Segment selected 2D images with superpixels, using the SLIC algorithm (Simple Linear Iterative Clustering).

Scalar images of increasingly labeled pixels are returned.


**Default values:**

size=16, regularity=10 and nb_iterations=10.

**Example of use:**

```
image.jpg +srgb2lab slic[-1] 16 +blend shapeaverage f[-2] "j(1,0)==i && j(0,1)==i" *[1] [-2]
```
slices

Arguments:

- z0[%, z1[%], boundary_conditions

Description:

Keep only specified slices of selected images.

boundary_conditions can be \{ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror \}.

Default values:

z1=z0 and boundary_conditions=0.

---

smooth

Arguments:

- amplitude[%]=0, sharpness=0, 0<=anisotropy<=1, alpha[%, sigma[%, dl>0, da>0, 0 | 1] or
- nb_iterations>=0, sharpness=0, anisotropy, alpha, sigma, dt>0, 0 or
- [tensor_field], amplitude=0, dl>0, da>0, precision>0, interpolation, fast_approx=0 | 1] or
- [tensor_field], nb_iters>=0, dt>0, 0

Description:
Smooth selected images anisotropically using diffusion PDE's, with specified field of diffusion tensors.
Interpolation can be \{ 0: nearest | 1: linear | 2: 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.

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

**Examples of use:**

- **Example #1**
  
  ```
  image.jpg repeat 3 smooth 40,0,1,1,2 done
  ```

  ![Example #1](image.jpg)

- **Example #2**

  ```
  image.jpg 100%,100%,1,2 rand[-1] -100,100 repeat 2 smooth[-1]
  100,0.2,1,4,4 done warp[0] [-1],1,1,1
  ```

  ![Example #2](image.jpg)

**snapshot3d**

**Arguments:**

- \_size>0, \_zoom>=0, \_backgroundR, \_backgroundG, \_backgroundB, \_backgroundA, \_fov_angle:

  or
Description:
Take 2D snapshots of selected 3D objects.
Set zoom to 0 to disable object auto-scaling.

Default values:
size=1024, zoom=1, [background_image]=(default) and fov_angle=45.

Examples of use:
• Example #1
  torus3d 100,20 rotate3d 1,1,0,60 snapshot3d 400,1.2,128,64,32

• Example #2
  torus3d 100,20 rotate3d 1,1,0,60 sample ? +snapshot3d[0] [1],1.2

solarize
No arguments
Description:
Solarize selected images.
**solidify**

**Arguments:**

- `_smoothness[%]`\(\geq 0\), `_diffusion_type`\(\{\ 0: \text{isotropic} \ | \ 1: \text{Delaunay-guided} \ | 
2: \text{edge-oriented} \} \), `_diffusion_iter`\(\geq 0\)

**Description:**

Solidify selected transparent images.

**Default values:**

smoothness=75%, diffusion_type=1 and diffusion_iter=20.

**Example of use:**

```
image.jpg 100%,100% circle[-1] 50%,50%,25%,1,255 append c +solidify, display_rgba
```

**solve**
**Arguments:**

- \( [\text{image}], \_\text{use\_LU}=\{0:\text{SVD} | 1:\text{LU}\} \)

**Description:**

Solve linear system \( AX = B \) for selected \( B \)-matrices and specified \( A \)-matrix.

If the system is under- or over-determined, the least squares solution is returned.

**Default values:**

\( \_\text{use\_LU}=0 \).

**Example of use:**

\[
\begin{pmatrix}
0,1,0;1,0,0;0,0,1
\end{pmatrix}
(1;2;3) +\text{solve}[-1][-2]
\]

---

**solve_poisson**

**Arguments:**

- "\text{laplacian\_command}" , \_\text{nb\_iterations}>=0, \_\text{time\_step}>0, \_\text{nb\_scales}>=0

**Description:**

Solve Poisson equation so that applying \( \text{laplacian}[n] \) is close to the result of \( \text{laplacian\_command}[n] \).

Solving is performed using a multi-scale gradient descent algorithm.

If \( \_\text{nb\_scales}=0 \), the number of scales is automatically determined.

**Default values:**

\( \_\text{nb\_iterations}=60, \_\text{dt}=5 \) and \( \_\text{nb\_scales}=0 \).

**Example of use:**

image.jpg command "foo : gradient x" +solve_poisson foo +foo[0] +laplacian[1]
sort

Arguments:

- \texttt{ordering\{ + | - \}, axis\{ x | y | z | c \}}

Description:

Sort pixel values of selected images.

If axis is specified, the sorting is done according to the data of the first column/row/slice/channel of selected images.

Default values:

ordering=+ and axis=(undefined).

Example of use:

\begin{verbatim}
64 rand 0,100 +sort display_graph 400,300,3
\end{verbatim}
sort_list

Arguments:
• ordering={+ | -}, criterion

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

Default values:
ordering=+, criterion=i.

Example of use:

\[(1;4;7;3;9;2;4;7;6;3;9;1;0;3;3;2)\] split y sort_list +,i append y

specl3d

Arguments:
• value\geq0

Description:
Set lightness of 3D specular light.
Default values:

value=0.15.

Example of use:

```plaintext
(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 } remove[0]
```
sphere3d

Arguments:

- \( \text{radius, } \text{nb\_recursions} \neq 0 \) or
- \( \text{radius, } \text{nb\_phi} \geq 3, \text{nb\_theta} \geq 3 \)

Description:

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

- If 2 arguments are specified:
  - If \( \text{nb\_recursions} > 0 \), the sphere is generated using recursive subdivisions of an icosahedron.
  - If \( \text{nb\_recursions} < 0 \), the sphere is generated using recursive subdivisions of a cube.
- If 3 arguments are specified, the sphere is generated using spherical coordinates discretization.

Default values:

\( \text{nb\_recursions} = 3. \)

Example of use:

```
sphere3d 100 +primitives3d 1 color3d[-2] $\{-\text{rgb}\}
```
spherical3d

Arguments:

- "radius_function(phi,theta)", nb_recursions!="0 or
- "radius_function(phi,theta)", nb_phi>=3, nb_theta>=3

Description:

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

Second and third arguments are the same as in command sphere3d.

Default values:

nb_recursions=5.

Example of use:

```latex
spherical3d "abs(1+0.5*cos(3*phi)*sin(4*theta))" +primitives3d 1
```

spherize

Arguments:

- radius[\%]>=0, strength, smoothness[\%]>=0, center_x[\%], center_y[\%], ratio_x/y>=0, angle, interpolation
**Description:**
Apply spherize effect on selected images.

**Default values:**
radius=50%, strength=1, smoothness=0, center_x=center_y=50%, ratio_x/y=1, angle=0 and interpolation=1.

**Example of use:**
```
image.jpg grid 5%,5%,0,0,0.6,255 spherize ,
```

---

**spiralbw**

**Arguments:**
- `width>0, height>0, is_2dcoords={ 0 | 1 }`

**Description:**
Input a 2D rectangular spiral image with specified size.

**Default values:**
height=width and is_2dcoords=0.

**Examples of use:**
- **Example #1**
```
spiralbw 16
```
spline

Arguments:

- \(x_0, y_0, u_0, v_0, x_1, y_1, u_1, v_1, \_opacity, \_color1, \ldots\)

Description:

Draw specified colored spline curve on selected images (cubic hermite spline).

Default values:

opacity=1 and color1=0.

Example of use:
spline3d

**Arguments:**

- \(x0\%\), \(y0\%\), \(z0\%\), \(u0\%\), \(v0\%\), \(w0\%\), \(x1\%\), \(y1\%\), \(z1\%\), \(u1\%\), \(v1\%\), \(w1\%\), \(\_nb\_vertices\) \(\geq 2\)

**Description:**

Input 3D spline with specified geometry.

**Default values:**

\(\_nb\_vertices=128\).

**Example of use:**

```
repeat 100 \{ spline3d \{u\},\{u\},\{u\},\{u\},\{u\},\{u\},\{u\},\{u\},\{u\},\{u\},\{u\},128 color3d[-1] ${-rgb} \} box3d 1 primitives3d[-1] 1 add3d
```

---

**split**

**Built-in command**

**Arguments:**
• \{ x | y | z | c \}...{ x | y | z | c }, \_split\_mode or 
• keep\_splitting\_values={ + | - },\_\{ x | y | z | c \}...{ x | y | z | c \},value1, value2,... or
• (no arg)

**Description:**
Split selected images along specified axes, or regarding to a sequence of scalar values (optionally along specified axes too).
*(equivalent to shortcut command \$s\$).*

**Default values:**
\split\_mode=-1.

**Examples of use:**
• Example #1

```
image.jpg split c
```

• Example #2

```
image.jpg split y,3
```
• Example #3

```plaintext
image.jpg split x,-128
```

• Example #4

```plaintext
1,20,1,1,"1,2,3,4" +split -,2,3 append[1--1] y
```

• Example #5

```plaintext
(1,2,3,3,3,4,4,4,4) +split x,0 append[1--1] y
```
split3d

No arguments

**Description:**

Split selected 3D objects into feature vectors:

```
{ header, sizes, vertices, primitives, colors, opacities }
```

(equivalent to shortcut command `s3d`).

To recreate the 3D object, append all produced images along the y-axis (with command `append y`).

**Example of use:**

```
box3d 100 +split3d
```
split_alpha

**Arguments:**

- \texttt{nb\_scales\%}={ 0:auto | -S<0 | N>0 }, \_subsample={ 0:no | 1:yes }, 0<=\_anisotropy<=1, 0<=\_minimize\_alpha<=1

**Description:**

Split selected images into alpha detail scales.

If \texttt{nb\_scales}=-S, the lowest scale has a size of at least SxS. Parameter anisotropy is only considered when subsample=0. Image reconstruction is done with command \texttt{merge\_alpha}.

**Default values:**

\texttt{nb\_scales=0, subsample=0, anisotropy=0 and minimize\_alpha=1}.

---

split_colors

**Arguments:**

- \texttt{tolerance}>=0, \_max\_nb\_outputs>0, \_min\_area>0

**Description:**

Split selected images as several image containing a single color.

One selected image can be split as at most \_max\_nb\_outputs images. Output images are sorted by decreasing area of extracted color regions and have an additional alpha-channel.

**Default values:**

\texttt{tolerance=0, max\_nb\_outputs=256 and min\_area=8}.

**Example of use:**

\texttt{image.jpg quantize 5 +split\_colors , display\_rgba}
split_details

Arguments:

- \texttt{nb\_scales[%] = \{ 0: auto | -S < 0 | N > 0 \}}, \texttt{base\_scale[%]]} = \geq 0, \texttt{detail\_scale[%]} \geq 0

Description:

Split selected images into \texttt{nb\_scales} detail scales.

If \texttt{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:

\texttt{nb\_scales=0, base\_scale=0 and detail\_scale=0}.

Example of use:

\texttt{image.jpg split\_details ,}
split_freq

Arguments:

• smoothness>0[%]

Description:
Split selected images into low and high frequency parts.
Example of use:

image.jpg split_freq 2%

split_opacity

No arguments

Description:
Split color and opacity parts of selected images.

This command returns 1 or 2 images for each selected image, whether it has an opacity channel or not.

split_tiles

Arguments:

- \( M \neq 0, N \neq 0, \_is\_homogeneous={ 0 | 1 } \)

Description:
Split selected images as a \( M \times N \) array of tiles.

If \( M \) or \( N \) is negative, it stands for the tile size instead.

Default values:
\( N=M \) and \_is\_homogeneous=0.

Example of use:

image.jpg +local split_tiles 5,4 blur 3,0 sharpen 700 append_tiles 4,5 done
split_vector

Arguments:

- keep_splitting_values={ + | - },value1,_value2,...

Description:
Split selected images into multiple parts, where specified vector [value1,_value2,...] is the separator.

sponge

Arguments:

- _size>0

Description:
Apply sponge effect on selected images.

Default values:
size=13.

Example of use:

image.jpg sponge ,
spread

Arguments:

- \( dx \geq 0, \ dy \geq 0, \ dz \geq 0 \)

Description:

Spread pixel values of selected images randomly along \( x, y \) and \( z \).

Default values:

\( dx = 3, \ dy = dx \) and \( dz = 0 \).

Example of use:

```
image.jpg +spread 3
```

sprite3d

No arguments

Description:

Convert selected images as 3D sprites.

Selected images with alpha channels are managed.

Example of use:

```
image.jpg sprite3d
```
sprites3d

**Arguments:**

- `[sprite], _sprite_has_alpha_channel={ 0 | 1 }`

**Description:**

Convert selected 3D objects as a sprite cloud.

Set `sprite_has_alpha_channel` to 1 to make the last channel of the selected sprite be a transparency mask.

**Default values:**

`mask_has_alpha_channel=0`.

**Example of use:**

```
torus3d 100,20 image.jpg rescale2d[-1],64 100%,100% gaussian[-1] 30%,30% *[-1] 255 append[-2,-1] c +sprites3d[0] [1],1 display_rgba[-2]
```
sqr

Built-in command
No arguments

Description:
Compute the pointwise square function of selected images.

Examples of use:

• Example #1

image.jpg +sqr

• Example #2

300,1,1,1,'40*x/w+u' +sqr display_graph 400,300
**sqrt**

**Built-in command**

**No arguments**

**Description:**

Compute the pointwise square root of selected images.

**Examples of use:**

- **Example #1**

  ```
  image.jpg +sqrt
  ```

- **Example #2**

  ```
  300,1,1,1,'40*x/w+u' +sqrt display_graph 400,300
  ```

---

**srand**

**Built-in command**

**Arguments:**

- **value** or
- **(no arg)**
Description:
Set random generator seed.
If no argument is specified, a random value is used as the random generator seed.

---

**srgb2lab**

**Arguments:**

- **illuminant={ 0:D50 | 1:D65 | 2:E }** or
- **(no arg)**

**Description:**
Convert color representation of selected images from sRGB to Lab.

**Default values:**
illuminant=2.

**Examples of use:**

- **Example #1**
  
  ```
  image.jpg srgb2lab split c
  ```

- **Example #2**
  
  ```
  image.jpg srgb2lab +split c mul[-2,-1] 2.5 append[-3--1] c lab2srgb
  ```
srgb2lab8

Arguments:
  • `illuminant={ 0:D50 | 1:D65 | 2:E }` or
  • `(no arg)`

Description:
Convert color representation of selected images from sRGB to Lab8.

Default values:
`illuminant=2`.

srgb2rgb

No arguments

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

ssd_patch

Arguments:
  • `[patch], _use_fourier={ 0 | 1 }, _boundary_conditions`

Description:
Compute fields of SSD between selected images and specified patch.

Argument `boundary_conditions` is valid only when `use_fourier=0`.
boundary_conditions can be `{ 0:dirichlet | 1:neumann | 2:periodic | 3:mirror }.`
Default values:
use_fourier=0 and boundary_conditions=0.

Example of use:

```
image.jpg +crop 20%,20%,35%,35% +ssd_patch[0] [1],0,0
```

ssim

Arguments:

- [reference], patch_size>0, max_value>0

Description:

Compute the Structural Similarity Index Measure (SSIM) between selected images and specified reference image.

This command does not modify the images, it just returns a value or a list of values in the status. When downsampling_factor is specified with a ending %, its value is equal to 1+(patch_size-1)*spatial_factor%.

SSIM is a measure introduced int the following paper:

The implementation of this command is a direct translation of the reference code (in Matlab), found at:
https://ece.uwaterloo.ca/~z70wang/research/ssim/
Default values:
patch_size=11, and max_value=255.

ssim_matrix

Arguments:
• _patch_size>0, _max_value>0

Description:
Compute SSIM (Structural Similarity Index Measure) matrix between selected images.

Default values:
patch_size=11, and max_value=255.

Example of use:
image.jpg +noise 30 +noise[0] 35 +noise[0] 38 cut. 0,255 +ssim_matrix
stained_glass

Arguments:

• \_edges[\%]\geq 0, \_shading\geq 0, \_is\_thin\_separators\{ 0 \mid 1 \}

Description:

Generate stained glass from selected images.

Default values:

edges=40%, shading=0.2 and is_precise=0.

Example of use:

image.jpg stained_glass 20%,1 cut 0,20

---

star3d

Arguments:

• \_nb\_branches\geq 0, 0\leq \_thickness\leq 1

Description:

Input 3D star at position \((0, 0, 0)\), with specified geometry.
Default values:
nb_branches=5 and thickness=0.38.

Example of use:

```
star3d , +primitives3d 1 color3d[-2] ${-rgb}
```

![Example image](image.jpg)

stars

Arguments:

- density[%]>=0, depth>=0, size>=0, nb_branches>=1, 0<=thickness<=1, smoothness[%]>=0, R=G=B=200 and opacity=1.

Description:

Add random stars to selected images.

Default values:

density=10%, depth=1, size=32, nb_branches=5, thickness=0.38, smoothness=0.5, R=G=B=200 and opacity=1.

Example of use:

```
image.jpg stars ,
```

![Example image](image.jpg)
status

Built-in command

Arguments:
• status_string

Description:
Set the current status. Used to define a returning value from a function.
(equivalent to shortcut command u).

Example of use:

```
image.jpg command "foo : u0=Dark u1=Bright status ${u{ia>=128}}"
```

std_noise

No arguments

Description:
Return the estimated noise standard deviation of the last selected image.

stencil

Arguments:
• _radius[\%]>=0, _smoothness>=0, _iterations>=0

Description:
Apply stencil filter on selected images.

Default values:
radius=3, smoothness=1 and iterations=8.
Example of use:

```
image.jpg +norm stencil. 2,1,4 +mul rm[0]
```

---

**stencilbw**

**Arguments:**

- `edges>=0, smoothness>=0`

**Description:**

Apply B&W stencil effect on selected images.

**Default values:**

edges=15 and smoothness=10.

**Example of use:**

```
image.jpg +stencilbw 40,4
```

---

**store**

**Arguments:**
**Description:**

Store selected images into one or several named variables.

Selected images are transferred to the variables, and are so removed from the image list. (except if the prepended variant of the command +store[selection] is used). If a single variable name is specified, all images of the selection are assigned to the named variable. Otherwise, there must be as many variable names as images in the selection, and each selected image is assigned to each specified named variable. Use command *input $variable_name* to bring the stored images back in the list.

**Default values:**

is_compressed=0.

This command has a [tutorial page].

**Example of use:**

```
sample eagle,earth store img1,img2 input $img2 $img1
```

---

**str**

**Arguments:**

- string

**Description:**

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

---

**str2hex**

**Arguments:**

- "string"

**Description:**
Convert specified string argument into a sequence of hexadecimal values (returned as a string).

**See also:**

`hex2str`.

**Example of use:**

```
hex=${"str2hex "Hello my friends""} echo $hex
```

```bash
[gmic]-0./ Start G'MIC interpreter.
[gmic]-0./ 48656c6c6f206d7920667269656e6473
[gmic]-0./ End G'MIC interpreter.
```

---

**strbuffer**

**Arguments:**

- `buffer_size`

**Description:**

Return a string describing a size for the specified buffer size.

---

**strcapitalize**

**Arguments:**

- `string`

**Description:**

Capitalize specified string.

---

**strcasevar**

**Arguments:**

- `"string"

**Description:**

Return a simplified version of the specified string, that can be used as a variable name.
strclut

Arguments:
• "string"

Description:
Return simplified version of the specified string that can be used as a CLUT name.

strcontains

Arguments:
• string1,string2

Description:
Return 1 if the first string contains the second one.

streamline3d

Arguments:
• x[%],y[%],z[%],_L>=0,_dl>0,_interpolation,_is_backward={ 0 | 1 },_is_oriented={ 0 | 1 } or
• 'formula',x,y,z,_L>=0,_dl>0,_interpolation,_is_backward={ 0 | 1 },_is_oriented={ 0 | 1 }

Description:
Extract 3D streamlines from selected vector fields or from specified formula.

interpolation can be { 0:nearest integer | 1:1st-order | 2:2nd-order | 3:4th-order }.

Default values:
dl=0.1, interpolation=2, is_backward=0 and is_oriented=0.

Example of use:

```
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} } remove[0] box3d 100 primitives3d[-1] 1 add3d
```
stripes_y

Arguments:

- _frequency>=0

Description:

Add vertical stripes to selected images.

Default values:

frequency=10.

Example of use:

```
image.jpg +stripes_y ,
```

strlen

Arguments:

- string1

Description:

Return the length of specified string argument.

Return the length of specified string argument.
strlowercase

Arguments:
- string

Description:
Return a lower-case version of the specified string.

strreplace

Arguments:
- string, search, replace

Description:
Search and replace substrings in an input string.

structuretensors

Arguments:
- scheme={ 0:centered | 1:forward/backward }

Description:
Compute the structure tensor field of selected images.

Default values:
scheme=0.

This command has a tutorial page.

Example of use:
```
image.jpg structuretensors abs pow 0.2
```
### struppercase

**Arguments:**
- string

**Description:**
Return an upper-case version of the specified string.

### strvar

**Arguments:**
- "string"

**Description:**
Return a simplified version of the specified string, that can be used as a variable name.
(version that creates a lowercase result, no longer than 128 chars).

### strver

**Arguments:**
- _version,_prerelease

**Description:**
Return the specified version number of the G'MIC interpreter, as a string.

**Default values:**
version=_${version} and prerelease=. 
**stylize**

**Arguments:**
- `[style_image], [fidelity_finest], [fidelity_coarsest], [fidelity_smoothness_finest]>=0

**Description:**
Transfer colors and textures from specified style image to selected images, using a multi-scale patch-matching algorithm.

If instant display window[0] is opened, the steps of the image synthesis are displayed on it.

`init_type` can be `{ 0:best-match | 1:identity | 2:randomized }`.

**Default values:**
- `fidelity_finest=0.5`, `fidelity_coarsest=2`, `fidelity_smoothness_finest=3`,
- `fidelity_smoothness_coarsest=0.5`, `fidelity_chroma=0.1`, `init_type=0`, `init_resolution=16`,
- `init_max_gradient=0`, `patch_size_analysis=5`, `patch_size_synthesis=5`, `patch_size_synthesis_final=5`,
- `nb_matches_finest=2`, `nb_matchesc_coarsest=30`, `penalize_repetitions=2`, `matching_precision=2`,
- `scale_factor=1.85`, `skip_finest_scales=0` and 'image_matching_command'="s c,-3 match_pca[0] [2] b[0,2] xy,0.7 n[0,2] 0,255 n[1,2] 0,200 a[0,1] c a[1,2] c".

---

**sub**

**Built-in command**

**Arguments:**
- `value[\%]`, `image` or `formula` or `(`no arg`)`

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

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

**Examples of use:**

- **Example #1**
  
  `image.jpg +sub 30\% cut 0,255`
• Example #2

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

• Example #3

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

• Example #4

```
image.jpg +mirror x sub
```
**sub3d**

**Arguments:**

- \( tx, ty, tz \)

**Description:**
Shift selected 3D objects with the opposite of specified displacement vector.

(equivalent to shortcut command \( 3d \)).

**Default values:**

\( ty= tz=0 \).

**Example of use:**

```text
sphere3d 10 repeat 5 { +sub3d[-1] 10,{u(-10,10)},0 color3d[-1] ${{-rgb}} } add3d
```

---

**sub_alpha**

**Arguments:**

- \([base_image], 0 <= \text{minimize\_alpha} <= 1\)

**Description:**
Compute the alpha-channel difference (opposite of alpha blending) between the selected images and the specified base image. The alpha difference $A-B$ is defined as the image having minimal opacity, such that $\text{alpha}_\text{blend}(B,A-B) = A$. The min_alpha argument is used to relax the alpha minimality constraint. When set to 1, alpha is constrained to be minimal. When set to 0, alpha is maximal (i.e. 255).

**Default values:**

minimize_alpha=1.

**Example of use:**

```
image.jpg testimage2d {w},{h} +sub_alpha[0] [1] display_rgba
```

---

**subdivide3d**

**No arguments**

**Description:**

Subdivide primitives of selected 3D objects.

---

**superformula3d**

**Arguments:**
Description:
Input 2D superformula curve as a 3D object.

Default values:
resolution=1024, m=8, n1=1, n2=5 and n3=8.

Example of use:
```
superformula3d ,
```

---

**surfels3d**

Arguments:
- \(0 \leq \text{left_right_attenuation} \leq 1, 0 \leq \text{top_bottom_attenuation} \leq 1, 0 \leq \text{closer_further_attenuation} \leq 1\)

Description:
Convert selected images to 3D objects composed of 3D surfels (or 2D edgels for 2D images).

The binary shape is composed of all non-zero voxels.
The resulting 3D object is colored according to the color of non-zero voxels.

Default values:
left_right_attenuation=1, top_bottom_attenuation=1 and closer_further_attenuation=1.

Example of use:
```
100,100,100 = 1,40%,40%,40% = 1,60%,60%,60% distance 1 lt 30% blur 3
gt 50% surfels3d 0.5,0.75,1
```
svd

No arguments

Description:
Compute SVD decomposition of selected matrices.

Example of use:

```
10,10,1,1,'x==y?x+u(-0.2,0.2):0' +svd
```

symmetrize

Arguments:
**Symmetrize:**
Symmetrize selected images regarding specified axis.

**Default values:**
x=y=50%, angle=90, boundary_conditions=3, is_antisymmetry=0 and swap_sides=0.

**Example of use:**

```
image.jpg +symmetrize 50%,50%,45 +symmetrize[-1] 50%,50%,-45
```

---

**syntexturize**

**Arguments:**
- **width[%]>0, height[%]>0**

**Description:**
Resynthetize width‘x’height versions of selected micro-textures by phase randomization.

The texture synthesis algorithm is a straightforward implementation of the method described in:

**Default values:**
width=height=100%.
Example of use:

image.jpg crop 2,282,50,328 syntexturize 320,320

----

**syntexturize_matchpatch**

**Arguments:**

- \_width\%>0, \_height\%>0, \_nb_scales>0, \_patch_size>0, \_blending_size>0, \_precision>0

**Description:**

Resynthetize width'x'height versions of selected micro-textures using a patch-matching algorithm.

If \_nb_scales==0, the number of scales used is estimated from the image size.

**Default values:**

width=height=100\%, \_nb_scales=0, \_patch_size=7, \_blending_size=5 and \_precision=1.

**Example of use:**

image.jpg crop 25\%,25\%,75\%,75\% syntexturize_matchpatch 512,512

---

**tan**

Built-in command
No arguments

Description:
Compute the pointwise tangent of selected images.

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

Examples of use:

• Example #1

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

• Example #2

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

tanh

No arguments

Description:
Compute the pointwise hyperbolic tangent of selected images.

Examples of use:

• Example #1
taquin

Arguments:

- \( M>0, \ N>0, \ remove\_tile=\{0:\text{none} \mid 1:\text{first} \mid 2:\text{last} \mid 3:\text{random}\}, \ _\text{relief}, \ _\text{border\_thickness\%}, \ _\text{border\_outline\%}, \ _\text{outline\_color} \)

Description:
Create \( M \times N \) taquin puzzle from selected images.

Default values:
\( N=M, \relief=50, \text{border\_thickness}=5, \text{border\_outline}=0 \) and \( \text{remove\_tile}=0 \).

Example of use:

\[
\text{image.jpg } +\text{taquin } 8
\]
tensors3d

Arguments:

- \_radius\_factor>=0, \_shape={ 0:box | >=N:ellipsoid }, \_radius\_min>=0

Description:

Generate 3D tensor fields from selected images.

when 'shape'>0, it gives the ellipsoid shape precision.

Default values:

radius\_factor=1, shape=2 and radius\_min=0.05.

Example of use:

```
6,6,6,9,"U = [x,y,z] - [w,h,d]/2; U/=norm(U); mul(U,U,3) + 0.3*eye(3)" tensors3d 0.8
```

testimage2d

Arguments:

- \_width>0, \_height>0, \_spectrum>0

Description:
Input a 2D synthetic image.

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

**Example of use:**
```
testimage2d 512
```

---

tetraedron_shade

**Arguments:**
- \(x0, y0, z0, x1, y1, z1, x2, y2, z2, x3, y3, z3, R0, G0, B0, \ldots, R1, G1, B1, \ldots, R2, G2, B2, \ldots, R3, G3\)

**Description:**
Draw tetraedron with interpolated colors on selected (volumetric) images.

---

tetris

**Arguments:**
- \(\text{scale}>0\)

**Description:**
Apply tetris effect on selected images.

**Default values:**
scale=10.

**Example of use:**
```
image.jpg +tetris 10
```
Built-in command

Arguments:

• `text, x[%|~], y[%|~], { font_height[%]>=0 | custom_font }, _opacity, _color1,...`

Description:

Draw specified colored text string on selected images.

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

If one of the x or y argument ends with a ~, its value is expected to be a centering ratio (in [0,1]) rather than a position.

Usual centering ratio are `{ 0: left-justified | 0.5: centered | 1: right-justified }`. Sizes 13 and 128 are special and correspond to binary fonts (no-antialiasing). Any other font size is rendered with anti-aliasing.

Specifying an empty target image resizes it to new dimensions such that the image contains the entire text string.

A custom font can be specified as a variable name that stores an image list of 256 or 512 items (512 for 256 character sprites + 256 associated opacities), or as an image selection that is a serialized version of such an image list.

Default values:

x=y=0.01~, font_height=16, opacity=1 and color1=0.

Examples of use:

• Example #1

```
image.jpg rescale2d ,600 div 2 y=0 repeat 30 { text {2*$>" : This is a nice text!",10,$y,{2*$>,0.9,255 y+={2*$>}} }
```
Example #2

```plaintext
0 text "G'MIC",0,0,23,1,255
```

---

**text3d**

**Arguments:**

- `text, { font_height>=0 | custom_font }, _depth>0, _smoothness`

**Description:**

Input a 3D text object from specified text.

**Default values:**

font_height=53, depth=10 and smoothness=1.5.

**Example of use:**

```plaintext
text3d "G'MIC as a
3D logo!"
```
text_outline

Arguments:

- `text, x[% | ~], y[% | ~], { _font_height[%] > 0 | custom_font }, outline >= 0, opacity, color1, ...`

Description:

Draw specified colored and outlined text string on selected images.

If one of the x or y argument ends with a ~, its value is expected to be a centering ratio (in [0,1]) rather than a position. Usual centering ratio are `{ 0: left-justified | 0.5: centered | 1: right-justified }`.

Default values:

`x=y=0.01 ~, font_height=7.5%, outline=2, opacity=1, color1=color2=color3=255` and `color4=255`.

Example of use:

```
image.jpg text_outline "Hi there!", 10, 10, 63, 3
```

---

text_pointcloud3d

Arguments:

- `"text1", "text2", smoothness`

Description:

Input 3D text pointcloud from the two specified strings.

Default values:

`text1=text1`, `text2=text2` and `smoothness=1`.

Example of use:
texturize3d

Arguments:

- \([\text{ind\_texture}], \_\text{[ind\_coords]}\)

Description:

Texturize selected 3D objects with specified texture and coordinates.

(equivalent to shortcut command \(t3d\)).

When \([\text{ind\_coords}]\) is omitted, default XY texture projection is performed.

Default values:

\(\text{ind\_coords}=(\text{undefined})\).

Example of use:

```
image.jpg torus3d 100,30 texturize3d[-1] [-2] keep[-1]
```
Arguments:
- \( \text{amplitude} \geq 0, \text{fibrousness} \geq 0, \text{emboss\_level} \geq 0 \)

Description:
Add paint canvas texture to selected images.

Default values:
- \( \text{amplitude}=20, \text{fibrousness}=3 \) and \( \text{emboss\_level}=0.6 \).

Example of use:
```
image.jpg +texturize_canvas
```

---

texturize\_paper

No arguments

Description:
Add paper texture to selected images.

Example of use:
```
image.jpg +texturize\_paper
```
thickline

Arguments:

- \( x_0[\%], y_0[\%], x_1[\%], y_1[\%], \text{thickness}, \text{opacity}, \text{color1} \)

Description:

Draw specified colored thick line on selected images.

Default values:

thickness=2, opacity=1 and color1=0.

Example of use:

```
400,400,1,3 repeat 100 thickline \{u([w,h,w,h,5])\},0.5,${-rgb} done
```

thickspline

Arguments:

- \( x_0[\%], y_0[\%], u_0[\%], v_0[\%], x_1[\%], y_1[\%], u_1[\%], v_1[\%], \text{thickness}, \text{opacity}, \text{color1}, \ldots \)

Description:

Draw specified colored thick spline curve on selected images (cubic hermite spline).

Default values:

thickness=3, opacity=1 and color1=0.

Example of use:

```
image.jpg repeat 30 \{ thickspline \{u(100)\},\{u(100)\},\{u(-600,600)\},\{u(-600,600)\},\{u(100)\},\{u(-600,600)\},3,1,${-RGB} \}
```
thinning

Arguments:

• \texttt{boundary\_conditions=\{ 0:dirichlet | 1:neumann \}}

Description:

Compute skeleton of binary shapes using morphological thinning
(beware, this is a quite slow iterative process)

Default values:

boundary\_conditions=1.

Example of use:

\begin{code}
\texttt{shape\_cupid 320 +thinning}
\end{code}

threshold

Arguments:

• \texttt{value[\%], is\_soft=\{ 0 | 1 \}} :

Description:
Threshold values of selected images.

soft can be \{ 0:hard-thresholding | 1:soft-thresholding \}.

**Default values:**

is_soft=0.

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

**Example of use:**

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

---

**tic**

**No arguments**

**Description:**

Initialize tic-toc timer.

Use it in conjunction with toc.

---

**time**

**No arguments**

**Description:**
Return current time as a string hh:mm:ss.

**to_a**

No arguments

**Description:**
Force selected images to have an alpha channel.

**to_automode**

No arguments

**Description:**
Force selected images to be in the most significant color mode.

This command checks for useless alpha channel (all values equal to 255), as well as detects grayscale images encoded as color images.

**to_color**

No arguments

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

**to_colormode**

**Arguments:**

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

**Description:**
Force selected images to be in a given color mode.

**Default values:**

mode=0.

**to_gray**
No arguments

**Description:**
Force selected images to be in GRAY mode.

**Example of use:**

```plaintext
image.jpg +to_gray
```

---

**to_graya**

No arguments

**Description:**
Force selected images to be in GRAYA mode.

---

**to_pseudogray**

**Arguments:**
- `max_step>=0, is_perceptual_constraint={ 0 | 1 }, bits_depth>0`

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

**Default values:**
- `max_step=5, is_perceptual_constraint=1 and bits_depth=8`

The original pseudo-gray technique has been introduced by Rich Franzen [http://r0k.us/graphics/pseudoGrey.html](http://r0k.us/graphics/pseudoGrey.html).
Extension of this technique to arbitrary increments for more tones, has been done by David Tschumperlé.
to_rgb

No arguments

**Description:**
Force selected images to be in RGB mode.

---

to_rgba

No arguments

**Description:**
Force selected images to be in RGBA mode.

---

toc

No arguments

**Description:**
Display elapsed time of the tic-toc timer since the last call to tic.

This command returns the elapsed time in the status value. Use it in conjunction with tic.

---

tones

**Arguments:**
- \( N > 0 \)

**Description:**
Get N tones masks from selected images.

**Example of use:**

```
image.jpg +tones 3
```
topographic_map

Arguments:

• _nb_levels>0, _smoothness

Description:
Render selected images as topographic maps.

Default values:

nb_levels=16 and smoothness=2.

Example of use:

image.jpg topographic_map 10
**torus3d**

**Arguments:**

- \(_{radius1, radius2, \text{nb\_subdivisions1}>2, \text{nb\_subdivisions2}>2}\)

**Description:**

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

**Default values:**

radius1=1, radius2=0.3, nb_subdivisions1=24 and nb_subdivisions2=12.

**Example of use:**

```
torus3d 10,3 +primitives3d 1 color3d[-2] ${-rgb}
```

---

**transform_polar**

**Arguments:**

- \("expr\_radius", "expr\_angle", center\_x[\%], center\_y[\%], boundary_conditions={0:dirichlet | 1:neumann | 2:periodic | 3:mirror}\)

**Description:**

Apply user-defined transform on polar representation of selected images.

**Default values:**

expr_radius=R-r, expr_rangle=a, center_x=center_y=50% and boundary_conditions=3.

**Example of use:**

```
image.jpg +transform_polar[0] R*(r/R)^2,a +transform_polar[0] r,2*a
```
transition

Arguments:

- \([transition\_shape], nb\_added\_frames>=0, 100>=shading>=0, _single\_frame\_only={-1=disabled | >=0}\)

Description:
Generate a transition sequence between selected images.

Default values:
shading=0 and single_frame_only=-1.

Example of use:

```plaintext
image.jpg +mirror c 100%,100% plasma[-1] 1,1,6 transition[0,1] [2],5
```
transition3d

Arguments:
- \_nb_frames>=2, \_nb_xtiles>0, \_nb_ytiles>0, \_axis_x, \_axis_y, \_axis_z, \_is_antialias={0 | 1}

Description:
Create 3D transition sequence between selected consecutive images.

axis_x, axis_y and axis_z can be set as mathematical expressions, depending on x and y.

Default values:
- nb_frames=10, nb_xtiles=nb_ytiles=3, axis_x=1, axis_y=1, axis_z=0 and is_antialias=1.

Example of use:
```
image.jpg +blur 5 transition3d 9 display_rgba
```
transpose

No arguments

Description:
Transpose selected matrices.

Example of use:

image.jpg +transpose

triangle3d

Arguments:

• x0, y0, z0, x1, y1, z1, x2, y2, z2

Description:
Input 3D triangle at specified coordinates.

Example of use:

repeat 100 { a:=$>*pi/50 triangle3d 0,0,0,0,0,3,\{cos(3*$a)\},\{sin(2*$a)\},0 color3d[-1] $\{-rgb\} } add3d
triangle_shade

Arguments:
- \(x_0, y_0, x_1, y_1, x_2, y_2, R_0, G_0, B_0, \ldots, R_1, G_1, B_1, \ldots, R_2, G_2, B_2, \ldots\)

Description:
Draw triangle with interpolated colors on selected images.

Example of use:

```
image.jpg triangle_shade
20,20,400,100,120,200,255,0,0,0,255,0,0,0,255
```

trisolve

Arguments:
- \([\text{image}]\)

Description:
Solve tridiagonal system \(AX = B\) for selected B-vectors and specified tridiagonal A-matrix.

Tridiagonal matrix must be stored as a 3 column vector, where 2nd column contains the diagonal coefficients, while 1st and 3rd columns contain the left and right coefficients.
Example of use:

\[(0,0,1;0,0,0;0,1,0) (1;2;3) +\text{trisolve}[-1] [-2]\]

truchet

**Arguments:**

- \(\text{scale}>0, \text{radius}>=0, \text{pattern}\_\text{type}={0: straight | 1: curved}\)

**Description:**

Fill selected images with random truchet patterns.

**Default values:**

scale=32, radius=5 and pattern\_type=1.

**Example of use:**

400,300 truchet ,

tsp

**Arguments:**
• _precision>=0

Description:

Try to solve the travelling salesman problem, using a combination of greedy search and 2-opt algorithms.

Selected images must have dimensions Nx1x1xC to represent N cities each with C-dimensional coordinates. This command re-order the selected data along the x-axis so that the point sequence becomes a shortest path.

Default values:

precision=256.

Example of use:

```
256,1,1,2 rand 0,512 tsp , 512,512,1,3 repeat w#0 circle[-1] {0,I[>]+},2,1,255,255,255 line[-1] {0,boundary=2; [I[>]+,1[$+1]]},1,255,128,0 done keep[-1]
```

---

tunnel

Arguments:

• _level>=0, _factor>0, _centering_x, _centering_y, _opacity, _angle

Description:

Apply tunnel effect on selected images.

Default values:

level=9, factor=80%, centering_x=centering_y=0.5, opacity=1 and angle=0

Example of use:

```
image.jpg tunnel 20
```
turbulence

Arguments:
- \( \text{radius}>0, \text{octaves}=\{1,2,3\ldots,12\}, \text{alpha}>0, \text{difference}=\{-10,10\}, \text{mode}=\{0,1,2,3\} \)

Description:
Render fractal noise or turbulence on selected images.

Default values:
radius=32, octaves=6, alpha=3, difference=0 and mode=0.

This command has a tutorial page.

Example of use:

400,400,1,3 turbulence 16

---

tv_flow

Arguments:
- \( \text{nb_iter}>0, \text{dt}, \text{keep_sequence}=\{0 | 1\} \)

Description:
Apply iterations of the total variation flow on selected images.

**Default values:**

\( \text{nb_iter}=10, \text{dt}=30 \) and \( \text{keep_sequence}=0 \).

**Example of use:**

```
image.jpg +tv_flow 40
```

---

twirl

**Arguments:**

- \( \text{amplitude}, \text{center}_x[\%], \text{center}_y[\%], \text{boundary_conditions} = \{ 0: \text{dirichlet} \mid \text{1: neumann} \mid \text{2: periodic} \mid \text{3: mirror} \} \)

**Description:**

Apply twirl deformation on selected images.

**Default values:**

\( \text{amplitude}=1, \text{center}_x=\text{center}_y=50\% \) and \( \text{boundary_conditions}=3 \).

**Example of use:**

```
image.jpg twirl 0.6
```
uint82base64

Arguments:
• encoding={ 0:base64 | 1:base64url }

Description:
Encode the values of the latest of the selected images as a base64-encoded string.

The string can be decoded using command base642uint8.
Selected images must have values that are integers in [0,255].

Default values:
encoding=0.

uncommand

Arguments:
• command_name[, command_name2,...] or *

Description:
Discard definition of specified custom commands.
Set argument to * for discarding all existing custom commands.
(equivalent to shortcut command um).

undistort

Arguments:
• -1<=_amplitude<=1,_aspect_ratio,_zoom,_center_x[%],_center_y[%],_boundary_conditions

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

References:
boundary_conditions can be { 0:dirichlet | 1:neumann | 2:periodic | 3:mirror }. 
Default values:

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

uniform_distribution

Arguments:

- nb_levels>=1, spectrum>=1

Description:

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

Example of use:

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

unroll

Arguments:

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

Description:

Unroll selected images along specified axis. 

(equivalent to shortcut command y).

Default values:

axis=y.

Example of use:

```
(1,2,3;4,5,6;7,8,9) +unroll y
```
unserialize  

Description:

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

unsharp

Arguments:

- \texttt{radius[%]} &ge; 0, \texttt{amount} &ge; 0, \texttt{threshold[%]} &ge; 0

Description:

Apply unsharp mask on selected images.

Default values:

amount=2 and threshold=0.

Example of use:

\begin{verbatim}
image.jpg blur 3 +unsharp 1.5,15 cut 0,255
\end{verbatim}
**unsharp_octave**

**Arguments:**
  - \( _\text{nb\_scales}>0, _\text{radius}[\%] \geq 0, _\text{amount} \geq 0, _\text{threshold}[\%] \geq 0 \)

**Description:**
Apply octave sharpening on selected images.

**Default values:**
\( \text{nb\_scales}=4, \text{radius}=1, \text{amount}=2 \) and \( \text{threshold}=0 \).

**Example of use:**

```
image.jpg blur 3 +unsharp_octave 4,5,15 cut 0,255
```

---

**update**

**No arguments**

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

*(equivalent to shortcut command up).*

---

**upscale_smart**

**Arguments:**
  - \( \text{width}[\%], \text{height}[\%], \text{depth}, \text{smoothness} \geq 0, \text{anisotropy}=[0,1], \text{sharpening} \geq 0 \)

**Description:**
Upscale selected images with an edge-preserving algorithm.

**Default values:**
Example of use:

```plaintext
image.jpg rescale2d,100 +upscale_smart 500%,500% append x
```

**vanvliet**

**Built-in command**

**Arguments:**

- `std_deviation>=0[\%]`, `order={0 | 1 | 2 | 3}`, `axis={x | y | z | c}`, `boundary_conditions`

**Description:**

Apply Vanvliet recursive filter on selected images, along specified axis and with specified standard deviation, order and boundary conditions. `boundary_conditions` can be `{0:dirichlet | 1:neumann | 2:periodic | 3:mirror}`.

**Default values:**

`boundary_conditions`=1.

**Examples of use:**

- **Example #1**
  ```plaintext
  image.jpg +vanvliet 3,1,x
  ```

- **Example #2**
variance_patch

Arguments:

• \( \text{patch\_size} \geq 1 \)

Description:

Compute variance of each images patch centered at \((x,y)\), in selected images.

Default values:

\( \text{patch\_size}=16 \)

Example of use:

```plaintext
image.jpg +variance_patch
```

vector2tensor

No arguments

Description:
Convert selected vector fields to corresponding tensor fields.

**verbose**

**Arguments:**
- `level` or `{ + | - }

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

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

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

**Default values:**
`level=1`.

**version**

**No arguments**

**Description:**
Display current version number on stdout.

**video2files**

**Arguments:**
- `input_filename, output_filename, first_frame>=0, last_frame={ >=0 | -1=last }, frame_step>=1`

**Description:**
Split specified input video file into image files, one for each frame.

First and last frames as well as step between frames can be specified.

**Default values:**
`output_filename=frame.png, first_frame=0, last_frame=-1 and frame_step=1`. 
vignette

Arguments:
- \(_{\text{strength}}\geq 0, 0 \leq \text{radius\_min} \leq 100, 0 \leq \text{radius\_max} \leq 100\)

Description:
Add vignette effect to selected images.

Default values:
strength=100, radius\_min=70 and radius\_max=90.

Example of use:
```
image.jpg vignette
```

---

volume3d

No arguments

Description:
Transform selected 3D volumetric images as 3D parallelepipedic objects.

Example of use:
```
image.jpg animate blur,0,5,30 append z volume3d
```
volumetric2d

Arguments:
- \( x_\% , y_\% , z_\% , \text{separator}_\% \geq 0 \)

Description:
Convert selected 3D volumetric images into a 2D representation.

Default values:
\( x=y=z=50\% \) and \( \text{separator}_\% = 0 \).

Example of use:
```
image.jpg rescale2d 64 animate noise,0,100,50 cut 0,255 append z
volumetric2d 50\%,50\%,50\%,1
```
voxelize3d

**Arguments:**
- \_max\_resolution>0, \_fill\_interior={ 0 | 1 }, \_preserve\_colors={ 0 | 1 }

**Description:**
Convert selected 3D objects as 3D volumetric images of binary voxels, using 3D mesh rasterization.

**Default values:**
\text{max\_resolution}=128, \text{fill\_interior}=1 and \text{preserve\_colors}=0.

wait

**Arguments:**
- delay or (no arg)

**Description:**
Wait for a given delay (in ms), optionally since the last call to wait.

or wait for a user event occurring on the selected instant display windows.
delay can be \{ <0:delay+flush events | 0:event | >0:delay \}.
Command selection (if any) stands for instant display window indices instead of image indices.
If no window indices are specified and if delay is positive, the command results in a hard sleep during specified delay.

**Default values:**
delay=0.
warhol

**Arguments:**

- \[_M>0, _N>0, _smoothness>=0, _color>=0\]

**Description:**
Create MxN Andy Warhol-like artwork from selected images.

**Default values:**

M=3, N=M, smoothness=2 and color=20.

**Example of use:**

```
image.jpg warhol 3,3,3,40
```

warn

**Built-in command**

**Arguments:**

- \[force_visible={ 0 | 1 }, _message\]

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

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

warp

**Built-in command**

**Arguments:**

- \[warping_field], _mode, _interpolation, _boundary_conditions, _nb_frames>0\]
Description:

Warp selected images with specified displacement field.

- mode can be { 0:backward-absolute | 1:backward-relative | 2:forward-absolute | 3:forward-relative }.
- interpolation can be { 0:nearest-neighbor | 1:linear | 2:cubic }.
- boundary_conditions can be { 0:dirichlet | 1:neumann | 2:periodic | 3:mirror }.

Default values:

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

This command has a tutorial page.

Example of use:

```
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*(!c?cos(4*A):sin(8*A))'
warp[-2] [-1],1,1,0 quiver[-1] [-1],10,1,1,1,100
```

---

warp_patch

Arguments:

- [displacement_map], patch_width>=1, patch_height>=1, patch_depth>=1, std_factor>0

Description:

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

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

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

Default values:

- std_factor=0.3 and boundary_conditions=3.
warp_perspective

Arguments:

• \( x\text{-angle}, y\text{-angle}, \text{zoom}>0, x\text{-center}, y\text{-center}, \text{boundary_conditions} = \{0: \text{dirichlet} | 1: \text{neumann} | 2: \text{periodic} | 3: \text{mirror} \} \)

Description:
Warp selected images with perspective deformation.

Default values:
x-angle=1.5, y-angle=0, zoom=1, x-center=y-center=50 and boundary_conditions=2.

Example of use:

```
image.jpg warp_perspective ,
```

---

warp_rbf

Arguments:

• \( x_{s0}[%], y_{s0}[%], x_{t0}[%], y_{t0}[%], \ldots, x_{SN}[%], y_{SN}[%], x_{TN}[%], y_{TN}[%] \)

Description:
Warp selected images using RBF-based interpolation.

Each argument \((x_{sk}, y_{sk})-(x_{tk}, y_{tk})\) corresponds to the coordinates of a keypoint respectively on the source and target images. The set of all keypoints define the overall image deformation.

Example of use:

```
image.jpg +warp_rbf 0,0,0,0,100%,0,100%,0,100%,100%,100%,100%,0,100%,0,100%,0,100%,50%,50%,70%,50%,25%,25%
```
**warp_seamless**

**Arguments:**

- \([\text{displacement\_map}], \sigma[\%] > 0, \text{blend\_dimension} = \{0: \text{auto} | 1:1D | 2:2D | 3:3D\}\)

**Description:**

Warp selected 2D or 3D images by specified displacement field, using seamless blending.

**Default values:**

\(\sigma = 5\% \) and \(\text{blend\_dimension} = 0\).

**Example of use:**

```
sp colorful,512 100%,100%,1,2,[x,y] l. {s xy,8 sort\_list +,u append\_tiles ,} +warp[0] [1] +warp\_seamless[0] [1]
```
water

Arguments:

- \_amplitude, \_smoothness\geq 0, \_angle

Description:

Apply water deformation on selected images.

Default values:

amplitude=30, smoothness=1.5 and angle=45.

Example of use:

```
image.jpg water ,
```

-watermark_fourier

Arguments:

- \text, \_size\geq 0

Description:

Add a textual watermark in the frequency domain of selected images.

Default values:

size=33.

Example of use:

```
image.jpg +watermark_fourier "Watermarked!" +display_fft
remove[-3,-1] normalize 0,255 append[-4,-2] y append[-2,-1] y
```
**watermark_visible**

**Arguments:**

- text, 0 < opacity < 1, { size > 0 | font }, angle, mode={ 0: remove | 1: add }, smoothness>=0

**Description:**

Add or remove a visible watermark on selected images (value range must be [0,255]).

**Default values:**

text=(c) G'MIC, opacity=0.3, size=53, angle=25, mode=1 and smoothness=0.

**Example of use:**

image.jpg watermark_visible ,0.7

---

**watershed**

**Arguments:**

- [priority_image], is_high_connectivity={ 0 | 1 }

**Description:**

Compute the watershed transform of selected images.
Default values:
is_high_connectivity=1.

Example of use:

```
400,400 noise 0.2,2 eq 1 +distance 1 mul[-1] -1 label[-2]
watershed[-2] [-1] mod[-2] 256 map[-2] 0 reverse
```

---

wave

Arguments:

- \_amplitude\_\geq0, \_frequency\_\geq0, \_center_x, \_center_y

Description:

Apply wave deformation on selected images.

Default values:

amplitude=4, frequency=0.4 and center_x=center_y=50.

Example of use:

```
image.jpg wave ,
```

---
weave

Arguments:

• \( _\text{density} \geq 0, 0 \leq _\text{thickness} \leq 100, 0 \leq _\text{shadow} \leq 100, _\text{shading} \geq 0, _\text{fibers\_amplitude} \geq 0, _\text{fibers\_smoothness} \geq 0 \),

Description:

Apply weave effect to the selected images.

angle can be \{ 0:0 \text{ deg.} \mid 1:22.5 \text{ deg.} \mid 2:45 \text{ deg.} \mid 3:67.5 \text{ deg.} \}.

Default values:

density=6, thickness=65, shadow=40, shading=0.5, fibers\_amplitude=0, fibers\_smoothness=0, angle=0 and curvature\_x=curvature\_y=0

Example of use:

```text
image.jpg weave ,
```

---

weird3d

Arguments:

• \( _\text{resolution} > 0 \)

Description:

Input 3D weird object at (0,0,0), with specified resolution.

Default values:

resolution=32.

Example of use:

```text
weird3d 48 +\text{primitives3d} 1 \text{color3d[-2]} \$\{-\text{rgb}\}
```
while

Arguments:

- condition

Description:

End a do...while block and go back to associated do if specified condition holds.

classification is a mathematical expression, whose evaluation is interpreted as \{ 0: \text{false} | \text{other: true} \}.

whirls

Arguments:

- \_texture>=0, \_smoothness>=0, \_darkness>=0, \_lightness>=0

Description:

Add random whirl texture to selected images.

Default values:

- texture=3, smoothness=6, darkness=0.5 and lightness=1.8.

Example of use:

```
image.jpg whirls ,
```
**wind**

**Arguments:**
- \_amplitude\geq 0, \_angle, 0 \leq \_attenuation \leq 1, \_threshold

**Description:**
Apply wind effect on selected images.

**Default values:**
amplitude=20, angle=0, attenuation=0.7 and threshold=20.

**Example of use:**
```
image.jpg +wind
```

![Example images](image.jpg)

---

**window**

**Arguments:**
- \texttt{width}\% \texttt{height}\% \_normalization, _fullscreen, _pos\_x\%\texttt{, pos\_y\%}, _title

**Description:**
Display selected images into an instant display window with specified size, normalization type,
fullscreen mode and title.

(equivalent to shortcut command \texttt{w}).

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

Specify pos\_x and pos\_y arguments only if the window has to be moved to the specified coordinates. Otherwise, they can be avoided.

'width'=0 or 'height'=0 closes the instant display window.

normalization can be \{ -1:keep same | 0:none | 1:always | 2:1st-time | 3:auto \}.

fullscreen can be \{ -1:keep same | 0:no | 1:yes \}.

You can manage up to 10 different instant display windows by using the numbered variants \texttt{w0} (default, eq. to \texttt{w}),\texttt{w1},...,\texttt{w9} of the command \texttt{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).

---

**\_2048**

No arguments

**Description:**

Launch the 2048 game.

---

**\_blobs**

No arguments

**Description:**

Launch the blobs editor.
**x_bouncing**

No arguments

**Description:**
Launch the bouncing balls demo.

---

**x_color_curves**

**Arguments:**

- \( \text{colorspace}=\{ \text{rgb} | \text{cmy} | \text{cmyk} | \text{hsi} | \text{hsl} | \text{hsv} | \text{lab} | \text{lch} | \text{ycbcr} | \text{last} \} \)

**Description:**
Apply color curves on selected RGB[A] images, using an interactive window.

Set `colorspace` to `last` to apply last defined color curves without opening interactive windows.

**Default values:**

`colorspace=rgb`. 
x_colorize

Arguments:

- \textit{is\_lineart} = \{0 \mid 1\}, \textit{max\_resolution} = \{0 \mid \geq 128\}, \textit{multichannels\_output} = \{0 \mid 1\}, \text{[palette1]}, \text{[palette2]}, \text{[grabber1]}

Description:

Colorized selected B&W images, using an interactive window.

When >0, argument max\_resolution defines the maximal image resolution used in the interactive window.

Default values:

is\_lineart = 1, max\_resolution = 1024 and multichannels\_output = 0.

x_connect4

No arguments

Description:

Launch the Connect Four game.

x_crop

No arguments

Description:

Crop selected images interactively.

If multiple input images are selected, the same crop is applied to all images.

(equivalent to shortcut command \texttt{xz}).

x_cut

No arguments

Description:

Cut selected images interactively.
x_fire

No arguments

Description:
Launch the fire effect demo.

x_fireworks

No arguments

Description:
Launch the fireworks demo.

x_fisheye

No arguments

Description:
Launch the fish-eye effect demo.

x_fourier

No arguments

Description:
Launch the fourier filtering demo.

x_grab_color

Arguments:

* variable_name

Description:
Open a color grabber widget from the first selected image.

Argument variable_name specifies the variable that contains the selected color values at any time.
Assigning -1 to it forces the interactive window to close.

Default values:
variable_name=xgc_variable.

---

x_hanoi

No arguments

**Description:**
Launch the Tower of Hanoi game.

---

x_histogram

No arguments

**Description:**
Launch the histogram demo.

---

x_hough

No arguments

**Description:**
Launch the hough transform demo.

---

x_jawbreaker

**Arguments:**

- $0 < \text{width} < 20, 0 < \text{height} < 20, 0 \leq \text{balls} \leq 8$

**Description:**
Launch the Jawbreaker game.

---

x_landscape

No arguments

**Description:**
Launch the virtual landscape demo.
**x_life**

No arguments

**Description:**
Launch the game of life.

---

**x_light**

No arguments

**Description:**
Launch the light effect demo.

---

**x_mandelbrot**

**Arguments:**

- `{ julia={ 0 | 1 }, _c0r, _c0i }

**Description:**
Launch Mandelbrot/Julia explorer.

---

**x_mask_color**

**Arguments:**

- `{ colorspace={ all | rgb | lrgb | ycbcr | lab | lch | hsv | hsi | hsl | cmy | cmyk | yiq }, _spatial_tolerance>=0, _color_tolerance>=0 }

**Description:**
Interactively select a color, and add an alpha channel containing the corresponding color mask.

Argument `colorspace` refers to the color metric used to compute color similarities, and can be basically one of `{ rgb | lrgb | ycbcr | lab | lch | hsv | hsi | hsl | cmy | cmyk | yiq }.
You can also select one particular channel of this colorspace, by setting `colorspace` as `colorspace_channel` (e.g. `hsv_h` for the hue).

**Default values:**

`colorspace=all`, `spatial_tolerance=5` and `color_tolerance=5`. 
x_metaballs3d

No arguments

**Description:**
Launch the 3D metaballs demo.

---

x_minesweeper

**Arguments:**
- \(8 \leq \text{width} \leq 20, 8 \leq \text{height} \leq 20\)

**Description:**
Launch the Minesweeper game.

---

x_minimal_path

No arguments

**Description:**
Launch the minimal path demo.

---

x_morph

**Arguments:**
- \(\text{nb\_frames} \geq 2, \text{preview\_fidelity}\{0: \text{coarsest} | 1: \text{coarse} | 2: \text{normal} | 3: \text{fine} | 4: \text{finest} \}\)

**Description:**
Launch the interactive image morpher.

**Default values:**
\(\text{nb\_frames}=16\) and \(\text{preview\_fidelity}=3\).

---

x_pacman

No arguments

**Description:**
Launch pacman game.

**x_paint**

No arguments

**Description:**
Launch the interactive painter.

**x_plasma**

No arguments

**Description:**
Launch the plasma effect demo.

**x_quantize_rgb**

**Arguments:**
- \_nbcolors\geq2

**Description:**
Launch the RGB color quantization demo.

**x_reflection3d**

No arguments

**Description:**
Launch the 3D reflection demo.

**x_rubber3d**

No arguments

**Description:**
Launch the 3D rubber object demo.
**x_segment**

**Arguments:**

- \_max_resolution={ 0 | >=128 }

**Description:**

Segment foreground from background in selected opaque RGB images, interactively.

Return RGBA images with binary alpha-channels.

**Default values:**

max_resolution=1024.

---

**x_select_color**

**Arguments:**

- \_variable_name

**Description:**

Display a RGB or RGBA color selector.

Argument variable_name specifies the variable that contains the selected color values (as R,G,B,[A]) at any time.

Its value specifies the initial selected color. Assigning -1 to it forces the interactive window to close.

**Default values:**

variable_name=xsc_variable.

---

**x_select_function1d**

**Arguments:**

- \_variable_name, \_background_curve_R, \_background_curve_G, \_background_curve_B

**Description:**

Open an interactive window, where the user can defined its own 1D function.

If an image is selected, it is used to display additional information:

- The first row defines the values of a background curve displayed on the window (e.g. an histogram).
- The 2nd, 3rd and 4th rows define the R,G,B color components displayed beside the X and Y axes.

Argument variable_name specifies the variable that contains the selected function keypoints at any
time. Assigning -1 to it forces the interactive window to close.

Default values:
variable_name=xsf_variable, background_curve_R=220, background_curve_G=background_curve_B=background_curve_T.

x_select_palette

Arguments:
- variable_name, _number_of_columns={ 0:auto | >0 }

Description:
Open a RGB or RGBA color selector widget from a palette.

The palette is given as a selected image. Argument variable_name specifies the variable that contains the selected color values (as R,G,B,[A]) at any time. Assigning -1 to it forces the interactive window to close.

Default values:
variable_name=xsp_variable and number_of_columns=2.

x_shadebobs

No arguments

Description:
Launch the shade bobs demo.

x_spline

No arguments

Description:
Launch spline curve editor.

x_starfield3d

No arguments
Description:
Launch the 3D starfield demo.

---

**x_tetris**

No arguments

**Description:**
Launch tetris game.

---

**x_threshold**

No arguments

**Description:**
Threshold selected images interactively.

---

**x_tictactoe**

No arguments

**Description:**
Launch tic-tac-toe game.

---

**x_tixy**

**Arguments:**
- "expression"

**Description:**
Animate specified mathematical expression with a 16x16 grid of circles, using the rules described at https://tixy.land.

---

**x_warp**

**Arguments:**
- \_nb_keypoints\_xgrid>=2,\_nb_keypoints\_ygrid>=2,\_nb_keypoints\_contours>=0,\_preview\_0:coarsest | 1:coarse | 2:normal | 3:fine | 4:finest
Description:
Launch the interactive image warper.

Default values:

nb_keypoints_xgrid=nb_keypoints_ygrid=2, nb_keypoints_contours=0 and preview_fidelity=1.

x_waves

No arguments

Description:
Launch the image waves demo.

x_whirl

Arguments:
• opacity=0

Description:
Launch the fractal whirls demo.

Default values:

opacity=0.2.

xor

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

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

Examples of use:
• Example #1
Example #2

```
image.jpg +mirror x xor
```

xyz2jzazbz

No arguments

**Description:**
Convert color representation of selected images from XYZ to RGB.

---

**xyz2lab**

**Arguments:**

- `illuminant={ 0:D50 | 1:D65 | 2:E }` or
- `(no arg)`

**Description:**
Convert color representation of selected images from XYZ to Lab.

**Default values:**
xyz2rgb

**Arguments:**
- `illuminant={ 0:D50 | 1:D65 | 2:E }` or
- `(no arg)`

**Description:**
Convert color representation of selected images from XYZ to RGB.

**Default values:**
`illuminant=2.`

xyz82rgb

**Arguments:**
- `illuminant={ 0:D50 | 1:D65 | 2:E }` or
- `(no arg)`

**Description:**
Convert color representation of selected images from XYZ8 to RGB.

**Default values:**
`illuminant=2.`

ycbcr2rgb

**No arguments**

**Description:**
Convert color representation of selected images from YCbCr to RGB.

yinyang

**No arguments**

**Description:**

Draw a yin-yang symbol on selected images.

Example of use:

```
400,400 yinyang
```

---

`yiq2rgb`

**No arguments**

**Description:**

Convert color representation of selected images from YIQ to RGB.

---

`yiq82rgb`

**No arguments**

**Description:**

Convert color representation of selected images from YIQ8 to RGB.

---

`yuv2rgb`

**No arguments**

**Description:**

Convert color representation of selected images from YUV to RGB.

---

`yuv82rgb`

**No arguments**
Description:
Convert selected images from YUV8 to RGB color bases.

---

**zoom**

**Arguments:**
- \( \text{factor, cx, cy, cz, boundary_conditions} = \{ 0: \text{dirichlet} | 1: \text{neumann} | 2: \text{periodic} | 3: \text{mirror} \} \)

**Description:**
Apply zoom factor to selected images.

**Default values:**
factor=1, cx=cy=cz=0.5 and boundary_conditions=0.

**Example of use:**

```plaintext
image.jpg +zoom[0] 0.6 +zoom[0] 1.5
```

---

End of document