The Handbook

Version 2.9.3 (pre-release #201024)

David Tschumperlé

October 24, 2020
2.25 Examples of Use ................................................. 658
License

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

An online version of this documentation is available at: https://gmic.eu/reference.shtml.

Motivations

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

G'MIC user interfaces are:

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

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

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

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

Version


Version 2.9.3 (pre-release #201024), Copyright (c) 2008-2020, David Tschumperlé (https://gmic.eu)
Chapter 1

Usage

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

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

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

1.1 Overall Context

− At any time, G'MIC manages one list of numbered (and optionally named) pixel−based images, entirely stored in computer memory (uncompressed).

− The first image of the list has index '0' and is denoted by '[0]'. The second image of the list is denoted by '[1]', the third by '[2]' and so on.

− Negative indices are treated in a periodic way: '[-1]' refers to the last image of the list, '[-2]' to the penultimate one, etc. Thus, if the list has 4 images, '[1]' and '[-3]' both designate the second image of the list.

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

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

− Such a pipeline can 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.
1.2 Image Definition and Terminology

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

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

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

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

- G'MIC stores all the image data as buffers of 'float' values (32 bits, value range \([-3.4E38,+3.4E38]\)). It performs all its image processing operations with floating point numbers. Each image pixel takes then 32bits/channel (except if double−precision buffers have been enabled during the compilation of the software, in which case 64bits/channel can be the default).

- Considering 'float'−valued pixels ensure to keep the numerical precision when executing image processing pipelines. For image input/output operations, you may want to prescribe the image datatype to be different than 'float' (like 'bool', 'char', 'int', etc...). This is possible by specifying it as a file option when using I/O commands. (see section 'Input/Output Properties' to learn more about file options).

1.3 Items of a Processing Pipeline

- In G'MIC, an image processing pipeline is described as a sequence of items separated by the space character ' '. Such items are interpreted and executed from the left to the right. For instance, the expression:
  
  filename.jpg blur 3,0 sharpen 10 resize 200%,200% output file_out.jpg

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

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

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

1.4. Input Data Items

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

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

- The following special input strings may be used as G'MIC items to create and insert new images with prescribed values, at the end of the image list:

  . ’[selection]’ or ’[selection]xN’: Insert 1 or N copies of already existing images. ’selection’ may represent one or several images (see section ‘Command Items and Selections’ to learn more about selections).

  . ’width[%], height[%], depth[%], spectrum[%], values[xN]’: Insert one or N images with specified size and values (adding ‘%’ to a dimension means ‘percentage of the size along the same axis, taken from the last image ‘[−1]’). Any specified dimension can be also written as ’[image]’ , and is then set to the size (along the same axis) of the existing specified image [image]. ‘values’ can be either a sequence of numbers separated by commas ’,’ or a mathematical expression, as e.g. in input item ‘256,256,1,3,[x,y,128]’ which creates a 256x256 RGB color image with a spatial shading on the red and green channels. (see section ‘Mathematical Expressions’ to learn more about mathematical expressions).

  . ’(v1,v2,...)[xN]’: Insert one or N new images from specified prescribed values. Value separator inside parentheses can be ‘,’ (column separator), ‘;’ (row separator), ‘/’ (slice separator) or ‘ˆ’ (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 ’ˆ’). When specified delimiter is ’,’ , ’;’ , ’/’ or ’ˆ’, the expression is actually equivalent to ’((’string’[:delimiter])’xN]’ (see section ‘Substitution Rules’ for more information on the syntax).

  . ’0[xN]’: Insert one or N new ‘empty’ images, containing no pixel data. Empty images are used only in rare occasions.

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

- A variable definition is always local to the current command except when it starts by the underscore character ‘_’. In that case, it becomes also accessible by any command invoked
outside the current command scope (global variable).

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

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

− Input item ‘name.=string’ concatenates specified ‘string’ to the end of variable ‘name’.

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

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

1.5 Command Items and Selections

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

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

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

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

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

- indices in selections are always sorted in increasing order, and duplicate indices are discarded. For instance, selections \([3-1,1-3]\) and \([1,1,1,3,2]\) are both equivalent to \([1-3]\). If you want to repeat a single command multiple times on an image, use a `repeat...done` loop instead. Inverting the order of images for a command is achieved by explicitly inverting the order of the images in the list, with command `reverse[selection]`.

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

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

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

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

- There are two different types of commands that can be run by the G'MIC interpreter:

  - Built-in commands, are the hard-coded functionalities in the interpreter core. They are thus compiled as binary code and run fast, most of the time. Omitting an argument when invoking a built-in command is not permitted, except if all following arguments are also omitted. For instance, invoking `plasma 10,,5` is invalid but `plasma 10` is correct.

  - Custom commands, are defined as G'MIC pipelines of built-in or other custom commands. They are parsed by the G'MIC interpreter, and thus run a bit slower than built-in commands. Omitting arguments when invoking a custom command is permitted. For instance, expressions `flower ,,,100,,2` or `flower ,` are correct.

- Most of the existing commands in G'MIC are actually defined as custom commands.

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

1.6 Input/Output Properties

- G'MIC is able to read/write most of the classical image file formats, including:

  - 2D grayscale/color files: `.png`, `.jpeg`, `.gif`, `.pnm`, `.tif`, `.bmp`, ...
  - 3D volumetric files: `.dcm`, `.hdr`, `.nii`, `.pan`, `.inr`, `.pnk`, ...
  - video files: `.mpeg`, `.avi`, `.mov`, `.ogg`, `.flv`, ...
  - Generic text or binary data files: `.gmz`, `.cimg`, `.cimgz`, `.dlm`, `.asc`, `.pfm`, `.raw`, `.txt`, `.h`
  - 3D object files: `.off`

- When dealing with color images, G'MIC generally reads, writes and displays data using the usual sRGB color space.

- G'MIC is able to manage 3D objects that may be read from files or generated by G'MIC commands.
A 3D object is stored as a one-column scalar image containing the object data, in the following order: `{ magic_number; sizes; vertices; primitives; colors; opacities }`. These 3D representations can be then processed as regular images.

(see command `split3d` for accessing each of these 3D object data separately).

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

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

  - **Video files:** Only sub—frames of an image sequence may be loaded, using the input expression `filename.ext,[first_frame],[last_frame],[step]`. Set `last_frame==−1` to tell it must be the last frame of the video. Set `step` to 0 to force an opened video file to be opened/closed. Output framerate and codec can be also set by using the output expression `filename.avi, fps, codec, keep_open=0|1`. `codec` is a 4-char string (see http://www.fourcc.org/codecs.php) or ‘0’ for the default codec. ‘keep_open’ tells if the output video file must be kept open for appending new frames afterwards.

  - **.cimg[z] files:** Only crops and sub—images of .cimg files can be loaded, using the input expressions `filename.cimg,N0,N1`, `filename.cimg,N0,N1,x0,x1`, `filename.cimg,N0,N1,x0,y0,x1,y1,z1` or `filename.cimg,N0,N1,x0,y0,z0,c0,x1,y1,z1,c1`. Specifying ‘−1’ for one coordinates stands for the maximum possible value. Output expression `filename.cimg[],datatype` can be used to force the output pixel type. ‘datatype’ can be `{auto | uchar | char | ushort | short | uint | uint64 | int | int64 | float | double).

  - **.raw binary files:** Image dimensions and input pixel type may be specified when loading .raw files with input expression `filename.raw[,datatype[,width[,height[,depth[,dim[,offset]]]]]]`. If no dimensions are specified, the resulting image is a one—column vector with maximum possible height. Pixel type can also be specified with the output expression `filename.raw[,datatype]`. ‘datatype’ can be the same as for .cimg[z] files.

  - **.yuv files:** Image dimensions must be specified when loading, and only sub—frames of an image sequence may be loaded, using the input expression `filename.yuv,width,height[,chroma_subsampling,[first_frame,[last_frame,[step]]]]`. ‘chroma_subsampling’ can be `{ 420 | 422 | 444 }`. When saving, chroma subsampling mode can be specified with output expression `filename.yuv[,chroma_subsampling]`.

  - **.tiff files:** Only sub—images of multi—pages tiff files can be loaded, using the input expression `filename.tif,[first_frame,[last_frame,[step]]]`. Output expression `filename.tiff[,datatype[,compression[,force_multipage,[use_bigtiff]]]]` can be used to specify the output pixel type, as well as the compression method. ‘datatype’ can be the same as for .cimg[z] files. ‘compression’ can be `{ none (default) | lzw | jpeg }`, ‘force_multipage’ can be `{ 0=no (default) | 1=yes }`, ‘use_bigtiff’ can be `{ 0=no | 1=yes (default) }.

  - **.gif files:** Animated gif files can be saved, using the input expression `filename.gif,fps>0,nb_loops`. Specify ‘nb_loops=0’ to get an infinite number of animation loops (this is the default behavior).
1.7. SUBSTITUTION RULES

- **.jpeg files**: The output quality may be specified (in %), using the output expression 'filename.jpg,30' (here, to get a 30% quality output). '100' is the default.

- **.mnc files**: The output header can set from another file, using the output expression 'filename.mnc,header_template.mnc'.

- **.pan, .cpp, .hpp, .c and .h files**: The output datatype can be selected with output expression 'filename[,datatype]'. 'datatype' can be the same as for .cimg[z] files.

- **.gmic files**: These filenames are assumed to be G'MIC custom commands files. Loading such a file will add the commands it defines to the interpreter. Debug information can be enabled/disabled by the input expression 'filename.gmic[,add_debug_info={ 0 | 1 }]'.

- Inserting 'ext:' on the beginning of a filename (e.g. 'jpg:filename') forces G'MIC to read/write the file as it would have been done if it had the specified extension '.ext'.

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

### 1.7 Substitution Rules

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

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

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

- '$!': The current number of images in the list.
- '$>' and '$<': The increasing/decreasing index of the latest (currently running) 'repeat...done' loop. '$>' goes from '0' (first loop iteration) to 'nb_iterations − 1' (last iteration). '$<' does the opposite.
- '$/': The current call stack. Stack items are separated by slashes '/'.
- '$%': The current value (expressed in seconds) of a millisecond precision timer.
- '$^': The current verbosity level.
- '$cpus': The number of computation cores available on your machine.
- '$pid': The current process identifier, as an integer.
- '$prerelease': For pre-releases, the date of the pre-release as 'ymmd'. For stable releases, this variable is set to 0.
- '$version': A 3−digits number telling about the current version of the G'MIC interpreter (e.g. '293').
- '$host': A string telling about the host running the G'MIC interpreter (e.g. 'cli' or 'gimp').
- '$vt100': Set to 1 if colored text output is allowed on the console. Otherwise, set to 0.
- '$path_rc': The path to the G'MIC folder used to store configuration files (its value is OS−dependent).
. `$.path_user`: The path to the G'MIC user file `.gmic` or `user.gmic` (its value is OS-dependent).
. `$.path_commands`: A list of all imported command files (stored as a list-valued variable).

- `$\{name\}` and `$\{\{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 `AAAAAAA`. 

- `'{\{\{\}\}'}\}'` 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:
  
  - 'w': display width (i.e. width of the display area managed by the window).
1.8. MATHEMATICAL EXPRESSIONS

- 'h': display height (i.e. height of the display area managed by the window).
- 'wh': display width x display height.
- 'd': window width (i.e. width of the window widget).
- 'e': window height (i.e. height of the window widget).
- 'de': window width x window height.
- 'u': screen width (actually independent on the window size).
- 'v': screen height (actually independent on the window size).
- 'uv': screen width x screen height.
- 'n': current normalization type of the instant display.
- 't': window title of the instant display.
- 'x': X-coordinate of the mouse position (or -1, if outside the display area).
- 'y': Y-coordinate of the mouse position (or -1, if outside the display area).
- 'b': state of the mouse buttons \{ 1=left-but. | 2=right-but. | 4=middle-but. \}.
- 'o': state of the mouse wheel.
- 'k': decimal code of the pressed key if any, 0 otherwise.
- 'c': boolean (0 or 1) telling if the instant display has been closed recently.
- 'r': boolean telling if the instant display has been resized recently.
- 'm': boolean telling if the instant display has been moved recently.

Any other 'feature' stands for a keycode name (in capital letters), and is substituted by a boolean describing the current key state \{ 0=pressed | 1=released \}.

You can also prepend a hyphen '-' to a 'feature' (that supports it) to flush the corresponding event immediately after reading its state (works for keys, mouse and window events).

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

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

1.8 Mathematical Expressions

- G'MIC has an embedded mathematical parser, used to evaluate (possibly complex) expressions inside braces '{ }', or formulas in commands that may take one as an argument (e.g. 'fill').

- When the context allows it, a formula is evaluated for each pixel of the selected images (e.g. 'fill').

- A math expression may return a scalar or vector-valued result (with a fixed number of components).

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

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

  _Usual math functions:_ abs(), acos(), acosh(), arg(), argkth(), argmax(), argmaxabs(),
  argmin(), argminabs(), asin(), asinh(), atan(), atan2(), atanh(), avg(), bool(), cbrt(), ceil(),
  clog(), clog2(), clog10(), cmax(), cmaxabs(),
  cmin(), cminabs(), cmod(), cpow(), cround(),
  csqrt(), fmax(), fmaxabs(),
  fmin(), fminabs(), fmod(),
  floor(), fround(), fsum(),
  isfinite(), isinf(), isnan(),
  log(), log10(), log2(),
  pow(), rint(), round(),
  saturate(), sign(),
  sqrt(), trunc().
cos(), cosh(), cut(), exp(), fact(), fibonacci(), floor(), gauss(), int(), isnan(), isnumber(), isnan(), isint(), isbool(), isexpr(), isdir(), kth(), log(), log2(), log10(), max(), maxabs(), med(), min(), minabs(), prod(), roll() (left bit rotation), rol() (right bit rotation), round(), sign(), sinc(), sinh(), sqrt(), std(), srand(), sum(), tan(), tanh(), var(), xor().

'atan2(y,x)' is the version of 'atan()' with two arguments 'y' and 'x' (as in C/C++).

'permute(k,n,with order)' computes the number of permutations of k objects from a set of n objects.

'gauss(x,sigma,is_normalized)' returns \(\exp(-x^2/(2+\pi^2))/\sqrt{(2+\pi^2)}\).

'cut(value, min, max)' returns value if it is in range [min, max], or min or max otherwise.

'narg(a,1,...,a,N)' returns the number of specified arguments (here, N).

'argl(a,1,...,a,N)' returns the ith argument a_i.

'isnum()', 'isint()', 'isfloat()', 'isstring()', 'isbool()' test the type of the given number or expression, and return 0 (false) or 1 (true).

'isfile(path)' (resp. 'isdir(path)') returns 0 (false) or 1 (true) whether its string argument is a path to an existing file (resp. to a directory) or not.

'sin(x,v,a,...,a,n)' returns 0 (false) or 1 (true) whether the first value 'v' appears in the set of other values 'a_i'.

'inrange(value,m,M,include_boundaries)' returns 0 (false) or 1 (true) whether the specified value lies in range [m,M] or not (checked range is [m,M] if 'include_boundaries'=0).

'argkth()', 'argmin()', 'argmax()', 'argminabs()', 'argmaxabs()', 'argminabs()', 'min()', 'max()', 'minabs()', 'maxabs()', 'med()', 'prod()', 'std()', 'sum()' and 'var()' can be called with an arbitrary number of scalar/vector arguments.

'vergkth()', 'vargmin()', 'vargmax()', 'vargminabs()', 'vargmaxabs()', 'vargminabs()', 'vmax()', 'vminabs()', 'vminabs()', 'vmaxabs()', 'vmed()', 'vprod()', 'vstd()', 'vsum()' and 'vvar()' are the versions of the previous function with vector-valued arguments.

'round(value, rounding_value, direction)' returns a rounded value. 'direction' can be {−1=to−lowest | 0=to−nearest | 1=to−highest }.

'lerp(a,b,t)' returns 'a+(t−1)*b'.

'swap(a,b)' swaps the values of the given arguments.

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

'.l': length of the associated list of images.

'.k': index of the associated image, in [0..1−1].

'.w': width of the associated image, if any (0 otherwise).

'.h': height of the associated image, if any (0 otherwise).

'.d': depth of the associated image, if any (0 otherwise).

'.s': spectrum of the associated image, if any (0 otherwise).

'.r': shared state of the associated image, if any (0 otherwise).

'.wh': shortcut for width x height x depth.

'.whd': shortcut for width x height x depth x spectrum (i.e. number of image values).

'.im', '.iM', '.ia', '.iv', '.is', '.ip', '.ic', '.in': Respectively the minimum, maximum, average, variance, sum, product, median value and L2−norm of the associated image, if any (0 otherwise).

'.xM', '.ym', '.zm', '.cm': The pixel coordinates of the minimum value in the associated image, if any (0 otherwise).

'.xM', '.yM', '.zM', '.cM': The pixel coordinates of the maximum value in the associated image, if any (0 otherwise).

All these variables are considered as **constant values** by the math parser for optimization
1.8. MATHEMATICAL EXPRESSIONS

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': Nth channel value of current processed pixel (i.e. value located at (x,y,z,N)) in the associated image, if any (0 otherwise). ’N’ must be an integer in range [0,9].

'.R’, '.G’, '.B’ and ‘.A’ are equivalent to ’i0’, ’i1’, ’i2’ and ’i3’ respectively.

'.i': current vector—valued processed pixel in the associated image, if any (0 otherwise).

The number of vector components is equal to the number of image channels (e.g. I = [ R,G,B ] for a RGB image).

You may add ‘#ind’ to any of the variable name above to retrieve the information for any numbered image [ind] of the list (when this makes sense). For instance ’ia#0’ denotes the average value of the first image of the list).

'.x’: current processed column of the associated image, if any (0 otherwise).

'.y’: current processed row of the associated image, if any (0 otherwise).

'.z’: current processed slice of the associated image, if any (0 otherwise).

'.c’: current processed channel of the associated image, if any (0 otherwise).

’.t’: thread id when an expression is evaluated with multiple threads (0 means ’master thread’).

’.e’: value of e, i.e. 2.71828...

’.pi’: value of pi, i.e. 3.1415926...

’.u’: a random value between [0,1], following a uniform distribution.

’.g’: a random value, following a gaussian distribution of variance 1 (roughly in [−6,6]).

’.interpolation’: value of the default interpolation mode used when reading pixel values with the pixel access operators (i.e. when the interpolation argument is not explicitly specified, see below for more details on pixel access operators). Its initial default value is 0.

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

Vector calculus: Most operators are also able to work with vector—valued elements.

’.a0,a1,...,aN−1’ defines a N—dimensional vector with scalar coefficients ak.

’.vectorN(a0,a1,,...,aN−1)’ does the same, with the ak being repeated periodically if only a few are specified.

’.vector(#N,a0,a1,,...,aN−1)’ does the same, and can be used for any constant expression N.

In previous expressions, the ak can be vectors themselves, to be concatenated into a single vector.

The scalar element ak of a vector X is retrieved by ’X[k]’.

The sub—vector [ X[p],X[p+s]...X[p+s×(q−1)] ] (of size q) of a vector X is retrieved by ’X[p:q:s]’.

.expr(‘formula’,_w_,h,d,s) outputs a vector of size w+h+d+s with values generated from the specified formula, as if one were filling an image with dimensions (w,h,d,s).

Equality/inequality comparisons between two vectors is done with operators ’==’ and ’!=’.

Some vector—specific functions can be used on vector values: ‘cross(X,Y)’ (cross product), ‘dot(X,Y)’ (dot product), ’size(X)’ (vector dimension), ‘sort(X,is_increasing,chunk_size)’ (sorting values), ’reverse(A)’ (reverse order of components), ’shift(A,length,boundary_conditions)’ and ’same(A,B, nb_vals,is_case_sensitive)’ (vector equality test).
. Function `normP(u1,...,un)` computes the LP–norm of the specified vector 
(P being an unsigned integer constant or "inf"). If P is omitted, the L2 norm is used.

. Function `resize(A,siz,interpolation,boundary_conditions)` returns a resized version of 
a vector ‘A’ with specified interpolation mode. ‘interpolation’ can be 
{−1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 
6=lanczos }, and ‘boundary_conditions’ can be 
{ 0=dirichlet | 1=neumann | 2=periodic | 
3=mirror }.

. Function `find(A,B,starting_index,search_step)` returns the index where 
sub–vector B appears in vector A, (or −1 if B is not found in A).
Argument A can be also replaced by an image index #find.

A 2–dimensional vector may be seen as a complex number and used in those particular 
functions/operators:
‘*’ (complex multiplication), ‘/’ (complex division), ‘**’ (complex exponentiation),
‘*+*’ (complex self–multiplication), ‘/=/’ (complex self–division), ‘**=’ (complex 
self–exponentiation), ‘cabs’ (complex modulus), ‘carg’ (complex argument), ‘cconj’
(complex conjugate), ‘cexp’ (complex exponential), ‘clog’ (complex logarithm)
‘ccos’ (complex cosine), ‘csin’ (complex sine), ‘ctan’ (complex tangent),
‘ccosh’ (complex hyperbolic cosine), ‘csinh’ (complex hyperbolic sine)
and ‘ctanh’ (complex hyperbolic tangent).

A MN–dimensional vector may be seen as a M x N matrix and used in those particular functions/operators:
‘*’ (matrix–vector multiplication), ‘det’ (determinant), ‘diag’ (diagonal matrix
from a vector), ‘eig’ (eigenvalues/eigenvectors), ‘eye’ (n x n identity matrix),
‘invert’ (matrix inverse), ‘mul’ (matrix multiplication),
‘pinv’ (matrix pseudoinverse), ‘rot’ (3D rotation matrix), ‘rot’ (2D rotation matrix),
‘solve’ (solver of linear system A.X = B),
‘svd’ (singular value decomposition), ‘trace’ (matrix trace) and
‘transpose’ (matrix transpose). Argument ‘nb_colsB’ may be omitted if it is 
equal to 1.

‘mproj(S.nb_colsS,D.nb_colsD.method,max_iter,max_residual)’ projects a matrix S onto
a dictionary (matrix) D. Equivalent to command ‘mproj’ but inside the math evaluator.

Specifying a vector–valued math expression as an argument of a command that operates on
image values (e.g. ‘fill’) modifies the whole spectrum range of the processed image(s),
for each spatial coordinates (x,y,z). The command does not loop over the C–axis in this
case.

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

- `"[ 'string' ]` and `"string"` define a vector whose values are the character codes of the
  specified character string (e.g. ‘foo’ is equal to [ 102,111,111 ]).
- `"character"` returns the (scalar) byte code of the specified character (e.g. ‘A’ is
equal to 65).

A special case happens for empty strings: Values of both expressions [ ] and ‘’ are 0.
Functions ‘lowercase’ and ‘uppercase’ return string with all string characters
lowercased or uppercased.

Function `str(str,starting_index, is strict)` parses specified string ‘str’ and
returns the value contained in it.

Function `vtos(expr,nb_digits,siz)` returns a vector of size ‘siz’ which contains
the character representation of values described by expression ‘expr’.
‘nb_digits’ can be { −1=auto—reduced | 0=all | >0=max number of digits }.

Function ‘echo(str1,str2,...,strN)’ prints the concatenation of given string arguments
on the console.

Function ‘string(#siz,str1,str2,...,strN)’ generates a vector corresponding to the
concatenation of given string/number arguments.

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

- The following **specific functions** are also defined:
  - `'u(max)` or `'u(min,max)`': return a random value between `[0,max]` or `[min,max]`, following a uniform distribution.
  - `f2ui(value)` and `ui2f(value)`: Convert a large unsigned integer as a negative floating point value (and vice-versa), so that 32bits floats can be used to store large integers while keeping a unitary precision.
  - `'i(a,b,c,d,interpolation_type,boundary_conditions)`: return the value of the pixel located at position `(a,b,c,d)` in the associated image, if any (0 otherwise).
    `interpolation_type` can be `{ 0=nearest neighbor | 1=linear | 2=cubic }`.
    `boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`.
    Omitted coordinates are replaced by their default values which are respectively x, y, z, c, interpolation and boundary.
    For instance command `'fill 0.5*(i(x+1)−i(x−1))'` will estimate the X−derivative of an image with a classical finite difference scheme.
  - `'j(dx,dy,dz,dc,interpolation_type,boundary_conditions)’ does the same for the pixel located at position `(x+dx,y+dy,z+dz,c+dc)` (pixel access relative to the current coordinates).
  - `'i[offset,boundary_conditions]’ returns the value of the pixel located at specified ‘offset’ in the associated image buffer (or 0 if offset is out−of−bounds).
  - `'j[offset,boundary_conditions]’ does the same for an offset relative to the current pixel coordinates (x,y,z,c).
  - `'i[#ind,x,y,z,c,interpolation,boundary_conditions]’,
. 'draw(#ind,S,x,y,z,c,dx,dy,dz,dc,opacity,M,\_max=M)' draws a sprite \$S\$ in image \$[\text{ind}]\$ (or in default image selected if \$\text{ind}\$ is not specified) at coordinates \$(x,y,z,c)\$.

The size of the sprite \$dx \times dy \times dz \times dc\$ must be specified. You can also specify a corresponding opacity mask \$M\$ if its size matches \$S\$.

. 'polygon(#ind,nb vertices,coords,opacity,color)' draws a filled polygon in image \$[\text{ind}]\$ (or in default image selected if \$\text{ind}\$ is not specified) at specified coordinates. It draws a single line if \$\text{nb vertices}\$ is set to 2.

. 'polygon(#ind,nb vertices,coords,opacity_pattern,color)' draws an outlined polygon in image \$[\text{ind}]\$ (or in default image selected if \$\text{ind}\$ is not specified) at specified coordinates and with specified line pattern.

It draws a single line if \$\text{nb vertices}\$ is set to 2.

. 'ellipse(#ind,xc,yc,\_radius1,\_radius2,angle,opacity,color)' draws a filled ellipse in image \$[\text{ind}]\$ (or in default image selected if \$\text{ind}\$ is not specified) with specified coordinates.

. 'ellipse(#ind,xc,yc,\_radius1,\_radius2,angle,opacity_pattern,color)' draws an outlined ellipse in image \$[\text{ind}]\$ (or in default image selected if \$\text{ind}\$ is not specified).

. 'resize(#ind,w,h,d,s,interp,boundary conditions,ex,ey,ez,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 \$\text{width(#ind)}\$ and \$\text{width(#ind)}\$ instead of the corresponding constant variables.

. 'if(condition,expr \_then,expr \_else)' returns value of \$\text{expr \_then}\$ or \$\text{expr \_else}\$ depending on the value of \$\text{condition}\$. \$\text{expr \_else}\$ can be omitted in which case 0 is returned if the condition does not hold. Using the ternary operator \$\text{condition}\text{?expr \_then}\text{?expr \_else}\$ gives an equivalent expression.

For instance, G'MIC commands 'fill if(x%10==0,255)\$i\$' and 'fill x%10?i:255' both draw blank vertical lines on every 10th column of an image.

. 'do(expression,condition)' repeats the evaluation of \$\text{expression}\$ until \$\text{condition}\$ vanishes (or until \$\text{expression}\$ vanishes if no \$\text{condition}\$ is specified). For instance, the expression: \$\text{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 \$\text{expression}\$.

. 'for(init,condition,procedure,\_body)' first evaluates the expression \$\text{init}\$, then iteratively evaluates \$\text{body}\$ (followed by \$\text{procedure}\$ if specified) while \$\text{condition}\$ is verified (i.e. not zero). It may happen that no iteration is done, in which case the function returns \$\text{nan}\$. Otherwise, it returns the last value of \$\text{body}\$.

For instance, the expression: \$\text{if}(N<2,N,for(n=N;F0=0;F1=1;n=\text{\_for}\=n−1,F2=F0+F1;F0=F1;F1=F2);F0=F1,F2=0))\$ 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 \$\text{for(init,condition,procedure,\_body)}\$ without the specification of an initializing expression.

. 'break()' and 'continue()' respectively breaks and continues the current running bloc (loop, init or main environment).

. 'size(filename)' returns the size of the specified \$\text{filename}\$ (or \$\text{nan}\$ if file does not exist).

. 'date(attr,\_path)' returns the date attribute for the given \$\text{path}\$ (file or directory), with \$\text{attr}\$ being \{ \$\text{year}|\text{month}|\text{day}|\text{week} | \text{hour}|\text{minute}|\text{second}\$ \}, or a vector of those values.

. 'date(attr)' returns the specified attribute for the current (locale) date (attributes being \{ \$0..7\text{same meaning as above}|7\text{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 \$\text{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")').
1.8. MATHEMATICAL EXPRESSIONS

- `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.x, im.y, im.z, 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).
- `(expr)` just ignores its arguments (mainly useful for debugging).
- `run('pipeline')` executes the specified G'MIC pipeline as if it was called outside the currently evaluated expression.
- `store(A,'varname', w, h, d, s, is_compressed)` transfers the data of vector `A` as a `w x h x d x s` image to the G'MIC variable `$varname`. 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, to_numbers)` 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, zcenter, xstart, ystart, zstart, xend, yend, zend, xstride, ystride, zstride, xdilation, ydilation, zdilation)` returns the correlation, unrolled as a vector, of the `wI x hI x dI x sI`−sized image `I` with the `wK x hK x dK x sK`−sized kernel `K` (the meaning of the other arguments are the same as in command `correlate`). Similar function `convolve(...)` is also defined for computing the convolution between `I` and `K`.

- User—defined macros:

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

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

. Starting an expression with ‘:’ or ‘∗’ forces the evaluations required for an image to be run in parallel, even if the amount of data to process is small (beware, it may be slower to evaluate in this case!). Specify ‘:’ (instead of ‘∗’) to avoid possible image copy done before evaluating the expression (this saves memory, but do this only if you are sure this step is not required!)

. If the specified expression starts with ‘>’ or ‘<’, the pixel access operators ‘i(), i[], j()' and ‘j[]’ return values of the image being currently modified, in forward (‘>’) or backward (‘<’) order. The multi-threading evaluation of the expression is also disabled in this case.

. Function ‘critical(operands)’ forces the execution of the given operands in a single thread at a time.

. ‘begin(t(expression))’ and ‘end(t(expression))’ evaluates the specified expressions once for each running thread (so possibly several times) at the beginning and the end of the evaluation procedure.

. ‘merge(variable,operator)’ tells to merge the local variable value computed by threads, with the specified operator, when all threads have finished computing.

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

— The last image of the list is always associated to the evaluations of ‘{expressions}’, e.g. G’MIC sequence ‘256,128 fill {w}’ will create a 256x128 image filled with value 256.

## 1.9 Image and Data Viewers

— G’MIC has some very handy embedded visualization modules, for 1D signals (command ‘plot’), 1D/2D/3D images (command ‘display’) and 3D objects (command ‘display3d’). It manages interactive views of the selected image data.

— The following actions are available in the interactive viewers:

. (mousewheel): Zoom in/out.
. ESC: Close window.
. CTRL+D: Increase window size.
. CTRL+C: Decrease window size.
. CTRL+R: Reset window size.
. CTRL+F: Toggle fullscreen mode.
. CTRL+S: Save current view as a numbered file ‘gmic xxxx.ext’.
. CTRL+O: Save copy of the viewed data, as a numbered file ‘gmic xxxx.ext’.

— Actions specific to the 1D/2D image viewer (command ‘display’) are:
### 1.10 Adding Custom Commands

New custom commands can be added by the user, through the use of G'MIC custom commands files.

A command file is a simple text file, where each line starts either by
'command_name: command_definition' or 'command_definition (continuation)'.

At startup, G'MIC automatically includes user’s command file
$HOME/gmic (on Unix) or %APPDATA%/user.gmic (on Windows). The CLI tool 'gmic'
automatically runs the command 'cli_start' if defined.

---

- Left mouse button: Create an image selection and zoom into it.
- Middle mouse button, or CTRL+left mouse button: Move image.
- Mouse wheel or PADD+/-: Zoom in/out.
- Arrow keys: Move image left/right/up/down.
- CTRL+A: Enable/disable transparency (show alpha channel).
- CTRL+N: Change normalization mode (can be 'none', 'normal' or 'channel—by—channel').
- CTRL+SPACE: Reset view.
- CTRL+X: Show/hide axes.
- CTRL+Z: Hold/release aspect ratio.

- Actions specific to the 3D volumetric image viewer (command 'display') are:
  - CTRL+P: Play z—stack of frames as a movie.
  - CTRL+V: Show/hide 3D view on bottom right zone.
  - CTRL+X: Show/hide axes.
  - CTRL+(mousewheel): Go up/down.
  - SHIFT+(mousewheel): Go left/right.
  - Numeric PAD: Zoom in/out (+/−) and move through zoomed image (digits).
  - BACKSPACE: Reset zoom scale.

- Actions specific to the 3D object viewer (command 'display3d') are:
  - (mouse)+(left mouse button): Rotate 3D object.
  - (mouse)+(right mouse button): Zoom 3D object.
  - (mouse)+(middle mouse button): Shift 3D object.
  - F1 ... F6: Toggle between different 3D rendering modes.
  - F7/F8: Decrease/increase focale.
  - F9: Select animation mode.
  - F10: Select animation speed.
  - SPACE: Start/stop animation.
  - CTRL+A: Show/hide 3D axes.
  - CTRL+B: Switch between available background.
  - CTRL+G: Save 3D object, as numbered file 'gmic_xxxx.off'.
  - CTRL+L: Show/hide outline.
  - CTRL+P: Print current 3D pose on stderr.
  - CTRL+T: Switch between single/double—sided 3D modes.
  - CTRL+V: Start animation with video output.
  - CTRL+X: Show/hide 3D bounding box.
  - CTRL+Z: Enable/disable z—buffered rendering.
− Custom command names must use character set \([a−zA−Z0−9_]\) and cannot start with a number.

− Any ’# comment’ expression found in a custom commands file is discarded by the G’MIC parser, wherever it is located in a line.

− In a custom command, the following $—expressions are recognized and substituted:

  . ’$*’ is substituted by a copy of the specified string of arguments.
  . ’$"*’ is substituted by a copy of the specified string of arguments, each being double—quoted.
  . ’$#’ is substituted by the maximum index of known arguments (either specified by the user or set to a default value in the custom command).
  . ’$[i]’ is substituted by the list of selected image indices that have been specified in the command invocation.
  . ’$?’ is substituted by a printable version of ’$[i]’ to be used in command descriptions.
  . ’$i’ and ’$[i]’ are both substituted by the i’th specified argument. Negative indices such as ’$[i−j]’ are allowed and refer to the j’th latest argument. ’$0’ is substituted by the custom command name.
  . ’$[default]’ is substituted by the value of $i (if defined) or by its new value set to ’default’ otherwise (’default’ may be a $—expression as well).
  . ’$[subset]’ is substituted by the argument values (separated by commas ’,’) of a specified argument subset. For instance expression ’$[2−2]’ is substituted by all specified command arguments except the first and the last one. Expression ’$[‘0’]’ is then substituted by all arguments of the invoked command (eq. to ’$*’ if all specified arguments have indeed a value).
  . ’$=var’ is substituted by the set of instructions that will assign each argument $i to the named variable ‘var$i’ (for i in \([0...$#]\)). This is particularly useful when a custom command want to manage variable numbers of arguments. Variables names must use character set \([a−zA−Z0−9_]\) and cannot start with a number.

− These particular $—expressions for custom commands are always substituted, even in double—quoted items or when the dollar sign ’$’ is escaped with a backslash ’\’. To avoid substitution, place an empty double quoted string just after the ’$’ (as in ’$''1’).

− Specifying arguments may be skipped when invoking a custom command, by replacing them by commas ’,’ as in expression ’flower ,,3’. Omitted arguments are set to their default values, which must be thus explicitly defined in the code of the corresponding custom command (using default argument expressions as ’$[1=default]’).

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

### 1.11 List of Commands

All available G’MIC commands are listed below, classified by themes. When several choices of command arguments are possible, they appear separated by ’|’. An argument specified inside ’[]’ or starting by ’\_’ is optional except when standing for an existing image [image], where ’image’ can be either an index number or an image name. In this case, the ’[]’ characters are mandatory when writing the item. A command marked with ’(+’ is one of the built—in commands. Note also that all images that serve as illustrations in this reference documentation are normalized in
range \([0,255]\) before being displayed. You may need to do this explicitly (command
'normalize 0,255’) if you want to save and view images with the same aspect than those
illustrated in the example codes.
Chapter 2

List of commands

2.1 Global Options

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

2.1.2 help
Arguments:
- command
- (no arg)
Display help (optionally for specified command only) and exit.
(eq. to ‘h’).

2.1.3 version
Display current version number on stdout.

2.2 Input / Output

2.2.1 camera (+)
Arguments:
- _camera_index>=0
- _nb_frames>=0
- _skip_frames>=0
- _capture_width>=0
- _capture_height>=0
Insert one or several frames from specified camera.
When ‘nb_frames==0’, the camera stream is released instead of capturing new images.
Default values:

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

2.2.2 clut

Arguments:

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

Insert one of the 862 pre-defined CLUTs at the end of the image list.
2.2. INPUT / OUTPUT

Default values:

- ‘resolution=33’ and ‘cut_and_round=1’.

Example 1: clut summer

2.2.3 command (+)

Arguments:

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

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

(eq. to ’m’). \(\text{\textbackslash n}\).

Imported commands are available directly after the `command` invocation.
Default value:

- ‘add_debug_info=1’.

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

2.2.4 cursor (+)

Arguments:

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

Show or hide mouse cursor for selected instant display windows. Command selection (if any) stands for instant display window indices instead of image indices.

Default value:

- ‘mode=1’.

2.2.5 display (+)

Arguments:

- `X[%]>=0,Y[%]>=0,Z[%]>=0,exit_on_anykey={ 0 | 1 }`

Display selected images in an interactive viewer (use the instant display window [0] if opened). (eq. to ‘d’). Arguments ‘X’, ‘Y’, ‘Z’ determine the initial selection view, for 3D volumetric images.
2.2. INPUT / OUTPUT

Default value:

- ‘X=Y=Z=0’ and ‘exit_on_anykey=0’.

Tutorial page:
https://gmic.eu/tutorial/_display.shtml

2.2.6 display0
Display selected images without value normalization.
(eq. to ‘d0’).

2.2.7 display2d
Display selected 2d images in an interactive window.
(eq. to ‘d2d’).
This command is used by default by command ’display’ when displaying 2d images.
If selected image is a volumetric image, each slice is displayed on a separate display window (up to 10 images
can be displayed simultaneously this way), with synchronized moves.
When interactive window is opened, the following actions are possible:
- Left mouse button: Create an image selection and zoom into it. - Middle mouse button, or CTRL+left mouse button: Move image. - Mouse wheel or PADD+/-: Zoom in/out. - Arrow keys: Move image left/right/up/down. - CTRL + A: Enable/disable transparency (show/hide alpha channel). - CTRL + C: Decrease window size. - CTRL + D: Increase window size. - CTRL + F: Toggle fullscreen mode. - CTRL + N: Change normalization mode (can be 'none' | 'normal' | 'channel-by-channel'). - CTRL + O: Save a copy of the input image, as a numbered file 'gmic_xxxxxx.gmz'. - CTRL + R: Reset both window size and view. - CTRL + S: Save a screenshot of the current view, as a numbered file 'gmic_xxxxxx.png'. - CTRL + SPACE: Reset view. - CTRL + X: Show/hide axes. - -CTRL + Z: Hold/release aspect ratio.

2.2.8 display3d
Arguments:
- _background_image, exit_on_anykey={ 0 | 1 }
- _exit_on_anykey={ 0 | 1 }
Display selected 3D objects in an interactive viewer (use the instant display window [0] if opened).
(eq. to ‘d3d’).
Default values:

- ‘[background_image]=(default)’ and ‘exit_on_anykey=0’.

2.2.9 display_array
Arguments:
- _width>0, _height>0
Display images in interactive windows where pixel neighborhoods can be explored.
Default values:

- ‘width=13’ and ‘height=width’.
2.2.10  display fft

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

Example 3: image.jpg +display fft

2.2.11  display_graph

Arguments:

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

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

Default values:

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

Example 4: 128,1,1,1,'cos(x/10+u)' +display graph 400,300,3

2.2.12 display_histogram

Arguments:

- \_width>0, \_height>0, \_clusters>0, \_min_value[\%], \_max_value[\%], \_show_axes={0 | 1}, \_expression.

Render a channel-by-channel histogram.
If selected images have several slices, the rendering is performed for all input slices.
'expression' is a mathematical expression used to transform the histogram data for visualization purpose.
(eq. to 'dh'). if specified 'width' or 'height' is '0', then image size is set to half the screen size.

Default values:

- 'width=0', 'height=0', 'clusters=256', 'min_value=0%', 'max_value=100%', 'show_axes=1' and 'expression=i'.
Example 5: image.jpg +display_histogram 512,300

2.2.13 display_parametric

Arguments:

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

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

Default values:

- ‘width=512’, ‘height=width’, ‘outline_opacity=3’, ‘vertex_radius=0’, ‘is_antialiased=1’, ‘is_decorated=1’, ‘xlabel="x-axis"’ and ‘ylabel="y-axis"’. 
Example 6: $1024,1,1,2,'t=x/40; if(c==0,\sin(t)\cdot\cos(t)^2\cdot\exp(\cos(t))-2\cdot\cos(4\cdot t)-\sin(t/12)^5)';$ display \texttt{parametric} 512,512

Example 7: $1000,1,1,2,u(-100,100) \text{ quantize } 4,1 \text{ noise } 12 \text{ channels } 0,2 +\text{normalize } 0,255 \text{ append c display \texttt{parametric} 512,512,0.1,8}$
2.2.14 *display parallel*
Display each selected image in a separate interactive display window.  
(*eq. to ‘dp’*).

2.2.15 *display parallel0*
Display each selected image in a separate interactive display window, without value normalization.  
(*eq. to ‘dp0’*).

2.2.16 *display polar*

**Arguments:**

- \_width\_>32, \_height\_>32, \_outline\_type\_, \_fill\_R\_, \_fill\_G\_, \_fill\_B\_, \_theta\_start\_, \_theta\_end\_, \_xlabel\_, \_ylabel\_.

Render polar curve from selected image data.
- \_outline\_type\_ can be \{ r\_<0=\text{dots with radius -}r \mid 0=\text{no outline} \mid r\>_0=\text{lines+dots with radius r} \}.  
- \_fill\_color\_ can be \{ -1=\text{no fill} \mid R,G,B=\text{fill with specified color} \}.

**Default values:**


**Example 8:** 300,1,1,’0.3+\text{abs}(\cos(10*\pi*x/w))+u(0.4)’ display_polar 512,512,4,200,255,200
2.2.17 *display_quiver*

Arguments:

- \( \text{size_factor} > 0 \), \( \text{arrow_size} > 0 \), \( \text{color_mode} = \{ 0 = \text{monochrome} \mid 1 = \text{grayscale} \mid 2 = \text{color} \} \)

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

Default values:

- \( '\text{size_factor}=16', \ '\text{arrow_size}=1.5' \) and \( '\text{color_mode}=1' \).
Example 10: `image.jpg +luminance gradient[-1] xy r\text{v}[2,-1] *[-2] -1 a[-2,-1] c\ crop 60,10,90,30 +display\ quiver[1] ,`

2.2.18  \textit{display\_rgba}

Arguments:

- \texttt{\_background\_RGB\_color}

Render selected RGBA images over a checkerboard or colored background. \textit{(eq. to ‘drgba’).}

Default values:

- ‘\texttt{\_background\_RGB\_color=undefined}’ (checkerboard).
2.2.19 display_tensors

Arguments:

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

Render selected images of tensors as a field of 2D ellipses. (\textit{eq. to ’dt’}).

Default values:

- ‘size_factor=16’, ‘ellipse_size=1.5’, ‘color_mode=2’ and ‘outline=2’.
Example 12: `image.jpg diffusiontensors 0.1,0.9 resize2dx 32 +display_tensors 64,2`

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

### 2.2.20 `display_warp`

**Arguments:**

- `_cell_size>0`

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

**Default value:**

- `'cell_size=15'`. 
2.2. INPUT / OUTPUT

Example 13:
400,400,1,2,'x=x-w/2;y=y-h/2;r=sqrt(x*x+y*y);a=atan2(y,x);5*sin(r/10)*[cos(a),sin(a)]' +display warp 10

2.2.21 document_gmic

Arguments:

- _format={ ascii | bash | html | images | latex }, _image_path, _write_wrapper={ 0 | 1 }

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

Default values:

- ‘format=ascii’, ‘image_path=""’ and ‘write_wrapper=1’.

Example(s) : input_text filename.gmic document_gmic html.img

2.2.22 echo (+)

Arguments:

- message

Output specified message on the error output. (eg. to ‘e’).

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

**Arguments:**

- filename, message

Output specified message, appending it to specified output file.  
(similar to `echo` for specified output file stream).

2.2.24  **echo stdout**

**Arguments:**

- message

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

2.2.25  **function1d**

**Arguments:**

- $0 \leq \text{smoothness} \leq 1$, $x_0=0, y_0=0, x_i=0, y_i, \ldots, x_n=0, y_n$

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

**Default values:**

- ’smoothness=1’ and ’$x_0=0, y_0=0$’.

---

Example 14: `function1d 1,0,0,10,30,40,20,70,30,80,0 +display graph 400,300`
2.2. INPUT / OUTPUT

2.2.26  *input* (+)

Arguments:

- [type:]filename
- [type:]http://URL
- [selection]nb_copies>0
- { width>0[%] | [image.d] }, { height>0[%] | [image.h] }, { depth>0[%] | [image.d] }, { spectrum>0[%] | [image.s] }, { value1, value2,... | 'formula' }
- (value1{, | ; | / | ^}value2{, | ; | / | ^}...[:{x | y | z | c | , | ; | / | ^}])
- 0

Insert a new image taken from a filename or from a copy of an existing image [index], or insert new image with specified dimensions and values. Single quotes may be omitted in 'formula'. Specifying argument '0' inserts an ‘empty’ image.

(eq. to ‘i’ | (no arg).)

**Default values:**

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

Example 15: input image.jpg
Example 16: input (1,2,3;4,5,6;7,8,9ˆ9,8,7;6,5,4;3,2,1)

Example 17: image.jpg (1,2,3;4,5,6;7,8,9) (255ˆ128ˆ64) 400,400,1,3,'if(x>w/2,x,y)*c'

Tutorial page:
https://gmic.eu/tutorial/_input.shtml
2.2. INPUT / OUTPUT

2.2.27 input_565

Arguments:
- filename, width>0, height>0, reverse_endianness={ 0 | 1 }

Insert image data from a raw RGB-565 file, at the end of the list.

Default value:
- ‘reverse_endianness=0’.

2.2.28 input_cube

Arguments:
- "filename", convert_1d_cluts_to_3d={ 0 | 1 }.

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

Default value:
- ‘convert_1d_cluts_to_3d=1’.

2.2.29 input_flo

Arguments:
- "filename"

Insert optical flow data from a .flo filename (vision.middlebury.edu file format).

2.2.30 input_glob

Arguments:
- pattern

Insert new images from several filenames that match the specified glob pattern. (eq. to ‘ig’).

2.2.31 input_gpl

Arguments:
- filename

Input specified filename as a .gpl palette data file.
2.2.32  *input text*

**Arguments:**

- `filename`

Input specified text-data filename as a new image.

*(eq. to 'it').*

2.2.33  *network (+)*

**Arguments:**

- `mode={ -1=disabled | 0=enabled w/o timeout | >0=enabled w/ specified timeout in seconds }

Enable/disable load-from-network and set corresponding timeout.

(Default mode is 'enabled w/o timeout').

2.2.34  *output (+)*

**Arguments:**

- `[type:]filename,.format_options`

Output selected images as one or several numbered file(s).

*(eq. to 'o').*

**Default value:**

- `'format_options'=(undefined).

2.2.35  *output.565*

**Arguments:**

- "filename", reverse_endianness={ 0=false | 1=true }

Output selected images as raw RGB-565 files.

**Default value:**

- `'reverse_endianness=0'.

2.2.36  *output.cube*

**Arguments:**

- "filename"

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

Arguments:

- "filename"

Output selected optical flow as a .flo file (vision.middlebury.edu file format).

2.2.38 output_ggr

Arguments:

- filename, gradient_name

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

2.2.39 output_text

Arguments:

- filename

Output selected images as text-data filenames.  
(eq. to ‘ot’).

2.2.40 outputn

Arguments:

- filename, index

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

2.2.41 outputp

Arguments:

- prefix

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

**Default value:**

- ‘prefix=’

2.2.42 outputw

Output selected images by overwriting their original location.  
(eq. to ’ow’).
2.2.43  \textit{outputx}

Arguments:

- \texttt{extension1, extension2, \ldots, extensionN, output\_at\_same\_location=\{ 0 | 1 \}}

Output selected images with same base filenames but for N different extensions. (\textit{eq. to '@ox'}).  

Default value:

- \texttt{’output\_at\_same\_location=0’}.

2.2.44  \textit{pass (+)}

Arguments:

- \texttt{\_shared\_state=\{ -1=status only | 0=non-shared (copy) | 1=shared | 2=adaptive \}}

Insert images from parent context of a custom command or a local environment. Command selection (if any) stands for a selection of images in the parent context. By default (adaptive shared state), selected images are inserted in a shared state if they do not belong to the context (selection) of the current custom command or local environment as well.  

Typical use of command ‘pass’ concerns the design of custom commands that take images as arguments. This commands return the list of corresponding indices in the status.  

Default value:

- \texttt{’shared\_state=2’}. 

\begin{figure}
\centering
\includegraphics[width=0.5\linewidth]{landscape.png}
\caption{’landscape’ (1024x576x1x3)}
\end{figure}

\begin{figure}
\centering
\includegraphics[width=0.5\linewidth]{landscape_t1.png}
\caption{’landscape_t1’ (1024x576x1x3)}
\end{figure}

\begin{figure}
\centering
\includegraphics[width=0.5\linewidth]{landscape_t2.png}
\caption{’landscape_t2’ (1024x576x1x3)}
\end{figure}
+average[0] [1]

2.2.45 plot (+)

Arguments:

- plot_type, vertex_type, xmin, xmax, ymin, ymax, exit_on_anykey={ 0 | 1 }
- 'formula', resolution>=0, plot_type, vertex_type, xmin, xmax, ymin, ymax,
  exit_on_anykey={ 0 | 1 }

Display selected images or formula in an interactive viewer (use the instant display window [0] if opened).
'plot_type' can be { 0=none | 1=lines | 2=splines | 3=bar }.
'vertex_type' can be { 0=none | 1=points | 2,3=crosses | 4,5=circles | 6,7=squares }.
'xmin','xmax','ymin','ymax' set the coordinates of the displayed xy-axes.

Default values:

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

2.2.46 print (+)

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

2.2.47 random_pattern

Arguments:

- _width>0, _height>0, _min_detail_level>=0

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'.

2.2.48  **screen (+)**

Arguments:

- \(x0\%,[y0\%],[x1\%],[y1\%]\)

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

2.2.49  **select (+)**

Arguments:

- \(\text{feature.type, }X[\%]>0, Y[\%]>0, Z[\%]>0, \text{exit.on.anykey=0 } | 1\), is.deep.selection=0 | 1 \}

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

- \(\text{feature.type}\) can be \(\{0=\text{point} | 1=\text{segment} | 2=\text{rectangle} | 3=\text{ellipse}\}\).

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

- The retrieved feature is returned as a 3D vector (if \(\text{feature.type==0}\)) or as a 6d vector (if \(\text{feature.type!=0}\)) containing the feature coordinates.

Default values:

- \('X=Y=Z=\text{undefined}', 'exit.on.anykey=0' and 'is.deep.selection=0'.\)
2.2. INPUT / OUTPUT

2.2.50 serialize (+)

Arguments:

- `_datatype`, `is_compressed={0 | 1}`, `store_names={0 | 1}`

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

- 'datatype' can be `{ auto | uchar | char | ushort | short | uint | int | uint64 | int64 | float | double }.

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

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

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

Default values:

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

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

2.2.51 shape circle

Arguments:

- `_size>=0`

Input a 2D circle binary shape with specified size.

Default value:

- `size=512`.
2.2.52  *shape_cupid*

**Arguments:**

- \( \_\text{size} \geq 0 \)

Input a 2D cupid binary shape with specified size.

**Default value:**

- ‘size=512’.
2.2.53 \textit{shape\_diamond}

Arguments:

- \_size\geq 0

Input a 2D diamond binary shape with specified size.

Default value:

- `size=512`. 
2.2.54 \textit{shape\_dragon}

\textbf{Arguments:}

\begin{itemize}
  \item \texttt{\_size}>0, \texttt{\_recursion\_level}>0, \texttt{\_angle}
\end{itemize}

Input a 2D Dragon curve with specified size.

\textbf{Default value:}

\begin{itemize}
  \item \texttt{\_size=512, \_recursion\_level=18 and \_angle=0}.
\end{itemize}
2.2.55 \textit{shape fern}

Arguments:

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

Input a 2D Barnsley fern with specified size.

Default value:

- ‘size=512’, ‘density=50\%’, ‘angle=30’, ‘opacity=0.3’ and ‘type=0’.
2.2.56  \textit{shape\_gear}

\textbf{Arguments:}

\begin{itemize}
  \item \texttt{size}>0, \texttt{nb\_teeth}>0, 0<\texttt{height\_teeth}\leq100, 0<\texttt{offset\_teeth}\leq100, 0<\texttt{inner\_radius}\leq100
\end{itemize}

Input a 2D gear binary shape with specified size.

\textbf{Default value:}

\begin{itemize}
  \item \texttt{size=512}, \texttt{nb\_teeth=12}, \texttt{height\_teeth=20}, \texttt{offset\_teeth=0} and \texttt{inner\_radius=40}.
\end{itemize}
2.2.57  \textit{shape\_heart}

Arguments:

- $\_size \geq 0$

Input a 2D heart binary shape with specified size.

Default value:

- ‘$size=512$’.
2.2.58  *shape_polygon*

**Arguments:**

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

Input a 2D polygonal binary shape with specified geometry.

**Default value:**

- ‘_size=512’, ‘_nb_vertices=5’ and ‘_angle=0’.
Example 28: \texttt{repeat 6 \texttt{shape\_polygon} 256,\{3+6\} done}

2.2.59 \texttt{shape\_snowflake}

Arguments:

- \texttt{size} \geq 0, 0 \leq \texttt{nb\_recursions} \leq 6

Input a 2D snowflake binary shape with specified size.

Default values:

- \texttt{`size=512'} and \texttt{`nb\_recursions=5'}. 
2.2.60 \textit{shape\_star}

\textbf{Arguments:}

- \texttt{\_size} \geq 0, \texttt{\_nb\_branches} \geq 0, 0 \leq \texttt{\_thickness} \leq 1

Input a 2D star binary shape with specified size.

\textbf{Default values:}

- ‘\texttt{size=512}, \texttt{nb\_branches=5} \texttt{and \_thickness=0.38}'.

\begin{center}
\includegraphics[width=\textwidth]{example29.png}
\end{center}

\textbf{Example 29:} \texttt{repeat 6 shape\_snowflake 256, \$> \texttt{done}}
2.2.61  \textit{shared} (+)

Arguments:

- x0[\%], x1[\%], y[\%], z[\%], c[\%]
- y0[\%], y1[\%], z[\%], c[\%]
- z0[\%], z1[\%], c[\%]
- c0[\%], c1[\%]
- c0[\%]
- (no arg)

Insert shared buffers from (opt. points/rows/planes/channels of) selected images. Shared buffers cannot be returned by a command, nor a local environment. (\textit{eq. to \textasciitilde sh\textasciitilde}).
Example 31: `image.jpg shared 1 blur[-1] 3 remove[-1]`

Example 32: `image.jpg repeat s shared 25%,75%,0,$ mirror[-1] x remove[-1] done`

Tutorial page:
https://gmic.eu/tutorial/_shared.shtml
2.2. INPUT / OUTPUT

2.2.62 sample

Arguments:

- \texttt{name1=} \{ ? \mid \text{apples} \mid \text{balloons} \mid \text{barbara} \mid \text{boats} \mid \text{bottles} \mid \text{butterfly} \mid \text{cameraman} \mid \text{car} \mid \text{cat} \mid \text{cliff} \mid \text{chick} \mid \text{colorful} \mid \text{david} \mid \text{dog} \mid \text{duck} \mid \text{eagle} \mid \text{elephant} \mid \text{earth} \mid \text{flower} \mid \text{fruits} \mid \text{gmicky} \mid \text{gmicky.mahvin} \mid \text{gmicky.wilber} \mid \text{greece} \mid \text{gummy} \mid \text{house} \mid \text{inside} \mid \text{landscape} \mid \text{leaf} \mid \text{lena} \mid \text{leno} \mid \text{lion} \mid \text{mandrill} \mid \text{monalisa} \mid \text{monkey} \mid \text{parrots} \mid \text{pencils} \mid \text{peppers} \mid \text{portrait0} \mid \text{portrait1} \mid \text{portrait2} \mid \text{portrait3} \mid \text{portrait4} \mid \text{portrait5} \mid \text{portrait6} \mid \text{portrait7} \mid \text{portrait8} \mid \text{portrait9} \mid \text{roddy} \mid \text{rooster} \mid \text{rose} \mid \text{square} \mid \text{swan} \mid \text{teddy} \mid \text{tiger} \mid \text{tulips} \mid \text{wall} \mid \text{waterfall} \mid \text{zelda} \}\), \texttt{name2,\ldots,nameN,width=} \{ \geq 0 \mid 0 \ ( \text{auto} ) \}, \texttt{height=} \{ \geq 0 \mid 0 \ ( \text{auto} ) \}
- (\text{no arg})

Input a new sample RGB image (opt. with specified size).
\texttt{(eq. to 'sp').}\texttt{\textbackslash n}).
Argument 'name' can be replaced by an integer which serves as a sample index.

\begin{figure}[h]
\centering
\begin{subfigure}{0.3\textwidth}
\includegraphics[width=\textwidth]{tiger.png}
\caption{[0]: 'tiger' (750x500x1x3)}
\end{subfigure} \quad
\begin{subfigure}{0.3\textwidth}
\includegraphics[width=\textwidth]{landscape.png}
\caption{[1]: 'landscape' (1024x576x1x3)}
\end{subfigure} \quad
\begin{subfigure}{0.3\textwidth}
\includegraphics[width=\textwidth]{duck.png}
\caption{[2]: 'duck' (640x480x1x3)}
\end{subfigure}
\end{figure}

\begin{figure}[h]
\centering
\begin{subfigure}{0.3\textwidth}
\includegraphics[width=\textwidth]{tiger.png}
\caption{[3]: 'tiger' (750x500x1x3)}
\end{subfigure} \quad
\begin{subfigure}{0.3\textwidth}
\includegraphics[width=\textwidth]{monkey.png}
\caption{[4]: 'monkey' (640x600x1x3)}
\end{subfigure} \quad
\begin{subfigure}{0.3\textwidth}
\includegraphics[width=\textwidth]{car.png}
\caption{[5]: 'car' (1024x683x1x3)}
\end{subfigure}
\end{figure}

Example 33: repeat 6 sample done

2.2.63 srand (+)

Arguments:

- \texttt{value}
- (\text{no arg})

Set random generator seed.
If no argument is specified, a random value is used as the random generator seed.
2.2.64  store (+)

Arguments:

- \_is\_compressed=\{ 0 | 1 \},variable\_name1,variable\_name2,...

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' to bring the stored images back in the list.

Default value:

- 'is\_compressed=0'.

Example 34: sample eagle, earth store img1, img2 input $img2 $img1

2.2.65  testimage2d

Arguments:

- \_width>0,\_height>0,\_spectrum>0

Input a 2D synthetic image.

Default values:

- 'width=512', 'height=width' and 'spectrum=3'.
2.2. INPUT / OUTPUT

2.2.66 uncommand (+)

Arguments:

- \texttt{command\_name[,]\_command\_name2,...}
- \texttt{*}

Discard definition of specified custom commands. Set argument to \texttt{*} for discarding all existing custom commands. (\textit{eq. to ‘um’}).

2.2.67 uniform\_distribution

Arguments:

- \texttt{nb\_levels}>=1, \texttt{spectrum}>=1

Input set of uniformly distributed spectrum-d points in \([0,1]\) spectrum.
2.2.68  **unserialize (+)**
Recreate lists of images from serialized image buffers, obtained with command 'serialize'.

2.2.69  **update**
Update commands from the latest definition file on the G’MIC server.
*(eq. to ‘up’).*

2.2.70  **parse CLI**

**Arguments:**
- `{ * | command_name }, output_mode`

Parse definition of ‘%@cli’-documented commands and output info about them in specified output mode.
'output_mode' can be { bashcompletion | help | html | images | print }.

**Default values:**
- 'command_name=*' and 'output_mode=print'.

2.2.71  **parse GUI**

**Arguments:**
- _filter_name, outputmode

Parse selected filter definitions and generate info about filters in selected output mode.
'outputmode' can be { json | list | print | strings | update | zart }.
2.2. INPUT / OUTPUT

It is possible to define a custom output mode, by implementing the following commands
('outputmode' must be replaced by the name of the custom user output mode): . 'parse_gui_outputmode' : A command that outputs the parsing information with a custom format. . 'parse_gui_parseparams_outputmode' (optional): A simple command that returns 0 or 1. It tells the parser whether parameters of matching filter must be analyzed (slower) or not. . 'parse_gui_trigger_outputmode' (optional): A command that is called by the parser just before parsing the set of each matching filters.

Here is the list of global variables set by the parser, accessible in command 'parse_gui_outputmode':

- '$nbfilters': Number of matching filters.
- '$nongui' (stored as an image): All merged lines in the file that do not correspond to '%@gui' lines.

For each filter \%F ('F' in range [0,$nbfilters-1]): . '$fF_name': Filter name. . '$fF_path': Full path. . '$fF_locale': Filter locale (empty, if not specified). . '$fF_command': Filter command. . '$fF_commandpreview': Filter preview command (empty, if not specified). . '$fF_zoomfactor': Default zoom factor (empty, if not specified). . '$fF_zoomaccurate': Is preview accurate when zoom changes? (can be {0=false | 1=true}). . '$fF_inputmode': Default preferred input mode (empty, if not specified). . '$fF_hide': Path of filter hid by current filter (for localized filters, empty if not specified). . '$fF_nbparams': Number of parameters.

For each parameter \%P of the filter \%F ('P' in range [0,$fF_nbparams-1]): . '$fF_pP_name': Parameter name. . '$fF_pP_type': Parameter type. . '$fF_pP_responsivity': Parameter responsivity (can be {0 | 1}). . '$fF_pP_visibility': Parameter visibility. . '$fF_pP_propagation': Propagation of the parameter visibility. . '$fF_pP_nbargs': Number of parameter arguments.

For each argument \%A of the parameter \%P ('A' in range [0,$fF_pP_nbargs-1]): . '$fF_pP_aA': Argument value

Default parameters: 'filter_name='*'' and 'output_format=print'.

### 2.2.72 verbose (+)

**Arguments:**

- `level`
- `{ + | - }

Set or increment/decrement the verbosity level. Default level is 0. 
*(eq. to 'v').

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

**Default value:**

- 'level=1'.

### 2.2.73 wait (+)

**Arguments:**

- `delay`
- `{ + | - }

Wait for a given delay (in ms), optionally since the last call to 'wait'. or wait for a user event occurring on the selected instant display windows.

'delay' can be {<0=delay+flush events | 0=event | >0=delay}

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

If no window indices are specified and if 'delay' is positive, the command results in a 'hard' sleep during specified delay.
Default value:

- ‘delay=0’.

### 2.2.74 \textit{warn} (+)

**Arguments:**

- \texttt{force.visible}\{ 0 | 1 \}, \texttt{message}

Print specified warning message, on the standard error (stderr). Command selection (if any) stands for displayed call stack subset instead of image indices.

### 2.2.75 \textit{window} (+)

**Arguments:**

- \texttt{width\%}>=-1, \texttt{height\%}>=-1, \texttt{normalization}, \texttt{fullscreen}, \texttt{pos.x\%}, \texttt{pos.y\%}, \texttt{title}

Display selected images into an instant display window with specified size, normalization type, fullscreen mode and title. (\texttt{eq. to \textquoteleft w\textquoteright} \texttt{)}.  
If \texttt{‘width’} or \texttt{‘height’} is set to -1, the corresponding dimension is adjusted to the window or image size. Specify \texttt{‘pos.x’} and \texttt{‘pos.y’} arguments only if the window has to be moved to the specified coordinates. Otherwise, they can be avoided. \texttt{‘width’}=0 or \texttt{‘height’}=0 closes the instant display window. \texttt{‘normalization’} can be \{ -1=keep same | 0=none | 1=always | 2=1st-time | 3=auto \}. \texttt{‘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, \texttt{eq. to \textquoteleft w\textquoteright} ), \texttt{‘w1’} \textsl{...}, \texttt{‘w9’} of the command \texttt{‘w’}.  
Invoke \texttt{‘window’} with no selection to make the window visible, if it has been closed by the user.

**Default values:**

- \texttt{‘width=height=normalization=fullscreen=-1’ and \textquoteleft title=\{undefined\}’}.

### 2.3 List Manipulation

#### 2.3.1 \textit{keep} (+)

Keep only selected images. (\texttt{eq. to \textquoteleft k\textquoteright} ).
2.3. LIST MANIPULATION

Example 37: `image.jpg split x keep[0-50%:2] append x`

Example 38: `image.jpg split x keep[30%-70%] append x`

2.3.2 move (+)

Arguments:
• **position[%]**

Move selected images at specified position.  
(*eq. to ‘mv’*).

**Example 39:** `image.jpg split x,3 move[1] 0`

**Example 40:** `image.jpg split x move[50%--1:2] 0 append x`
2.3. LIST MANIPULATION

2.3.3 name (+)

Arguments:

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

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

Example 41: image.jpg name image blur[image] 2

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

2.3.4 remove (+)

Remove selected images. (eq. to 'rm').
2.3.5 *remove_duplicates*
Remove duplicates images in the selected images list.
2.3. LIST MANIPULATION

Example 44: \((1,2,3,4,2,4,3,1,3,4,2,1)\) split x remove_duplicates append x

2.3.6 remove_empty

Remove empty images in the selected image list.

2.3.7 remove_named

Arguments:

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

Remove all images with specified names from the list of images. Does nothing if no images with those names exist. (eq. to ‘rmn’).

2.3.8 reverse (+)

Reverse positions of selected images. (eq. to ‘rv’).
CHAPTER 2. LIST OF COMMANDS

Example 45: `image.jpg split x,3 reverse[-2,-1]`

Example 46: `image.jpg split x,-16 reverse[50%-100%] append x`

2.3.9  sort_list

Arguments:
2.4. MATHEMATICAL OPERATORS

- _ordering={ + | - }, _criterion

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

**Default values:**

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

**Example 47:**

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

---

2.4 Mathematical Operators

2.4.1 *abs (+)*

Compute the pointwise absolute values of selected images.
Example 48: `image.jpg +sub {ia} abs[-1]`

Example 49: `300,1,1,1,’cos(20*x/w)’ +abs display,graph 400,300`

### 2.4.2 acos (+)

Compute the pointwise arccosine of selected images.
2.4. MATHEMATICAL OPERATORS

Example 50: image.jpg +normalize -1,1 acos[-1]

Example 51: 300,1,1,'cut(x/w+0.1*u,0,1)’ +acos display graph 400,300

Tutorial page:
2.4.3  *acosh* (+)

Compute the pointwise hyperbolic arccosine of selected images.

2.4.4  *add* (+)

**Arguments:**

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

Add specified value, image or mathematical expression to selected images, or compute the pointwise sum of selected images. (*eq. to ’+’*).

Example 52: `image.jpg +add 30% cut 0,255`
Example 53: `image.jpg +blur 5 normalize 0,255 add[1] [0]`

Example 54: `image.jpg add '80*cos(80*(x/w-0.5)*(y/w-0.5)+c)' cut 0,255`
Example 55: `image.jpg repeat 9 +rotate[0] {5>36},1,0,50%,50% done add div 10`

2.4.5 \textit{and} (+)

Arguments:

- \texttt{value[\%]}
- \texttt{[image]}
- \texttt{‘formula’}
- \texttt{(no arg)}

Compute the bitwise AND of selected images with specified value, image or mathematical expression, or compute the pointwise sequential bitwise AND of selected images. 
(eq. to ‘\&’).
Example 56: `image.jpg` and `{128+64}`

Example 57: `image.jpg + mirror x and`

**2.4.6 argmax**

Compute the argmax of selected images. Returns a single image with each pixel value being the index of the input image with maximal value.
Example 58: image.jpg sample lena,lion,square +argmax

2.4.7 argmaxabs

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

2.4.8 argmin

Compute the argmin of selected images. Returns a single image with each pixel value being the index of the input image with minimal value.
2.4.9 \textit{argminabs}

Compute the \textit{argminabs} of selected images. Returns a single image with each pixel value being the index of the input image with \textit{minabs} value.

2.4.10 \textit{asin} (+)

Compute the pointwise arcsine of selected images.
CHAPTER 2. LIST OF COMMANDS

Example 60: `image.jpg +normalize -1,1 asin[-1]`

Example 61: `300,1,1,'cut(x/w+0.1*u,0,1)' +asin display graph 400,300`

Tutorial page:
2.4.11 \textit{asinh (+)}
Compute the pointwise hyperbolic arcsine of selected images.

2.4.12 \textit{atan (+)}
Compute the pointwise arctangent of selected images.

Example 62: \texttt{image.jpg +normalize 0,8 atan[-1]}
Example 63: 300,1,1,1,'4*x/w+u' +atan display graph 400,300

Tutorial page:

2.4.13 atan2 (+)

Arguments:
- \([x_{\text{argument}}]\)

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

Example 64: (-1,1) (-1;1) resize 400,400,1,1,3 atan2[1] [0] keep[1] mod \{\pi/8\}

Tutorial page:

2.4.14 atanh (+)

Compute the pointwise hyperbolic arctangent of selected images.
2.4.15  \textit{bsl} (+)

\textbf{Arguments:}

- \texttt{value[\%]}
- [image]
- \texttt{‘formula’}
- (no arg)

Compute the bitwise left shift of selected images with specified value, image or mathematical expression, or compute the pointwise sequential bitwise left shift of selected images.  
(eq. to \texttt{‘<<’}).

\begin{example}
image.jpg bsl ‘round(3*x/w,0)’ cut 0,255
\end{example}

2.4.16  \textit{bsr} (+)

\textbf{Arguments:}

- \texttt{value[\%]}
- [image]
- \texttt{‘formula’}
- (no arg)

Compute the bitwise right shift of selected images with specified value, image or mathematical expression, or compute the pointwise sequential bitwise right shift of selected images.  
(eq. to \texttt{‘>>’}).
Example 66: `image.jpg bsr 'round(3*x/w,0)' cut 0,255`

2.4.17 \textit{cos (±)}

Compute the pointwise cosine of selected images.

Example 67: `image.jpg +normalize 0,\{2*pi\} cos[-1]`
Example 68: 300,1,1,1,'20*x/w+u' + cos display graph 400,300

Tutorial page:

2.4.18  \textit{cosh} (+)

Compute the pointwise hyperbolic cosine of selected images.
CHAPTER 2. LIST OF COMMANDS

Example 69: `image.jpg +normalize -3,3 cosh[-1]`

Example 70: `300,1,1,1,'4*x/w+u' +cosh display graph 400,300`

2.4.19 \textit{div (+)}

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

(eq. to `/`).

Example 71: `image.jpg div '1+abs(cos(x/10)*sin(y/10))'`
2.4.20 div_complex

Arguments:

- \([\text{divider}\_\text{real}, \text{divider}\_\text{imag}], \epsilon > 0\)

Perform division of the selected complex pairs \((\text{real1}, \text{imag1}), \ldots, (\text{realN}, \text{imagN})\) of images by specified complex pair of images \((\text{divider}\_\text{real}, \text{divider}\_\text{imag})\).

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

Default value:

- ‘\(\epsilon = 1e-8\)’.

2.4.21 eq (+)

Arguments:

- value\[%\]
- \([\text{image}]\)
- ‘\text{formula}\’
- (no arg)

Compute the boolean equality of selected images with specified value, image or mathematical expression, or compute the boolean equality of selected images.

\((eq. \text{ to } ‘=‘)\).
Example 73: image.jpg round 40 eq \{round(ia,40)\}

Example 74: image.jpg +mirror x eq

2.4.22 \textit{exp} (+)

Compute the pointwise exponential of selected images.
Example 75: image.jpg +normalize 0,2 exp[-1]

Example 76: 300,1,1,1,'7*x/w+u' +exp display.graph 400,300

2.4.23  ge (+)

Arguments:
2.4. MATHEMATICAL OPERATORS

- value[\%]
- [image]
- ‘formula’
- (no arg)

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

Example 77: image.jpg ge {ia}
2.4.24  \texttt{gt (+)}

Arguments:

- \texttt{value[\%]}
- \texttt{[image]}
- \texttt{’formula’}
- \texttt{(no arg)}

Compute the boolean ‘greater than’ of selected images with specified value, image or mathematical expression, or compute the boolean ‘greater than’ of selected images.  
\textit{(eq. to ‘>’)}.
2.4. MATHEMATICAL OPERATORS

Example 79: `image.jpg gt {ia}`

Example 80: `image.jpg +mirror x gt`

2.4.25 le (+)

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

Compute the boolean ‘less or equal than’ of selected images with specified value, image or mathematical expression, or compute the boolean ‘less or equal than’ of selected images.

(eq. to ‘<=').

Example 81: image.jpg le {is}
2.4. MATHEMATICAL OPERATORS

Example 82: image.jpg +mirror x le

2.4.26 \textit{lt} (+)

Arguments:

- \texttt{value}\% 
- \texttt{image} 
- \texttt{’formula’} 
- (no arg)

Compute the boolean 'less than' of selected images with specified value, image or mathematical expression, or compute the boolean 'less than' of selected images. 
\textit{(eq. to ’<’).}
Example 83: \texttt{image.jpg lt \{ia\}}

Example 84: \texttt{image.jpg +mirror x lt}

2.4.27 \texttt{log (+)}

Compute the pointwise base-e logarithm of selected images.
2.4. MATHEMATICAL OPERATORS

Example 85: `image.jpg +add 1 log[-1]`

Example 86: `300,1,1,1,'7*x/w+u' +log display graph 400,300`

2.4.28 \textit{\textbf{log10}} (+)

Compute the pointwise base-10 logarithm of selected images.
Example 87: \texttt{image.jpg +add 1 log10[-1]}

Example 88: \texttt{300,1,1,1,'7*x/w+u' +log10 display graph 400,300}

2.4.29 \texttt{log2 (+)}

Compute the pointwise base-2 logarithm of selected images
2.4.30 $\textit{max}$ (+)

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

Example 91: `image.jpg +mirror x max`
2.4. MATHEMATICAL OPERATORS

Example 92: image.jpg max ‘R=((x/w-0.5)ˆ2+(y/h-0.5)ˆ2)ˆ0.5;255*R’

2.4.31 maxabs (+)

Arguments:

- value[%]
- [image]
- ‘formula’
- (no arg)

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

2.4.32 mdiv (+)

Arguments:

- value[%]
- [image]
- ‘formula’
- (no arg)

Compute the matrix division of selected matrices/vectors by specified value, image or mathematical expression, or compute the matrix division of selected images.

(eq. to ‘m/^’).
2.4.33  med

Compute the median of selected images.

Example 93: image.jpg sample lena,lion,square +med

2.4.34  min (+)

Arguments:

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

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

Example 94: `image.jpg +mirror x min`

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

2.4.35 \textit{minabs} (+)

Arguments:
Computing the minabs between selected images and specified value, image or mathematical expression, or compute the pointwise minabs between selected images.

2.4.36  \textit{mod \,(+)}

Arguments:

- \textit{value[\%]}
- \textit{[image]}
- \textit{‘formula’}
- \textit{(no arg)}

Computing the modulo of selected images with specified value, image or mathematical expression, or compute the pointwise sequential modulo of selected images. \quad (\textit{eq. to ‘\%’}).

Example 96: \texttt{image.jpg \_mirror \_x mod}
2.4. MATHEMATICAL OPERATORS

Example 97: `image.jpg` mod `R=((x/w-0.5)^2+(y/h-0.5)^2)^0.5;255*R`

2.4.37 **mmul (+)**

**Arguments:**

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

Compute the matrix right multiplication of selected matrices/vectors by specified value, image or mathematical expression, or compute the matrix right multiplication of selected images. *(eq. to `m*`).*
2.4.38 \textit{mul} (+)

Arguments:

- \textit{value[]}  
- \textit{[image]}  
- ‘\textit{formula}’  
- (no arg)

Multiply selected images by specified value, image or mathematical expression, or compute the pointwise product of selected images.
\textit{(eq. to ‘∗’).}
Example 99: `image.jpg +mul 2 cut 0,255`

Example 100: `image.jpg (1,2,3,4,5,6,7,8) ri[-1] [0] mul[0] [-1]

Example 100: `image.jpg (1,2,3,4,5,6,7,8) ri[-1] [0] mul[0] [-1]`
Example 101: image.jpg mul '1-3*abs(x/w-0.5)' cut 0,255

Example 102: image.jpg +luminance negate[-1] +mul

2.4.39  *mul_channels*

**Arguments:**
2.4. MATHEMATICAL OPERATORS

- value1, value2, ..., valueN

Multiply channels of selected images by specified sequence of values.

Example 103: `image.jpg +mul_channels 1,0.5,0.8`

2.4.40 `mul_complex`

Arguments:

- `[multiplier_real, multiplier_imag]`

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

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

2.4.41 `neq (+)`

Arguments:

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

Compute the boolean inequality of selected images with specified value, image or mathematical expression, or compute the boolean inequality of selected images. 

(`eq. to '!=').`
Example 104: image.jpg round 40 neq \{\text{round}(ia, 40)\}

2.4.42 \textit{or (+)}

Arguments:

- \text{value}[\%]
- [image]
- ‘formula’
- (no arg)

Compute the bitwise OR of selected images with specified value, image or mathematical expression, or compute the pointwise sequential bitwise OR of selected images. (\textit{eq. to ‘|’}).
2.4.43  \texttt{pow (+)}

\textbf{Arguments:}
• value[%]
• [image]
• ’formula’
• (no arg)

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

Example 107: image.jpg div 255 +pow 0.5 mul 255
2.4. MATHEMATICAL OPERATORS

Example 108: `image.jpg gradient pow 2 add pow 0.2`

2.4.44 \textit{rol (+)}

\textbf{Arguments:}

- \texttt{value[\%]}
- \texttt{[image]}
- \texttt{’formula’}
- \texttt{(no arg)}

Compute the bitwise left rotation of selected images with specified value, image or mathematical expression, or compute the pointwise sequential bitwise left rotation of selected images.
CHAPTER 2. LIST OF COMMANDS

Example 109: `image.jpg rol 'round(3*x/w,0)' cut 0,255`

2.4.45  ror (+)

Arguments:

- `value[%]`
- `[image]`
- `'formula'`
- `{no arg}`

Compute the bitwise right rotation of selected images with specified value, image or mathematical expression, or compute the pointwise sequential bitwise right rotation of selected images.
2.4. MATHEMATICAL OPERATORS

Example 110: `image.jpg` ror `round(3*x/w,0)` cut 0,255

2.4.46 **sign (+)**

Compute the pointwise sign of selected images.

Example 111: `image.jpg` +sub `{ia}` sign[-1]
2.4.47 \textit{sin} (+)

Compute the pointwise sine of selected images.

Example 112: \texttt{300,1,1,1,’cos(20*x/w+u)’ +sign display_graph 400,300}

Example 113: \texttt{image.jpg +normalize 0,{2*pi} sin[-1]}
2.4. MATHEMATICAL OPERATORS

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

Tutorial page:

2.4.48 $sinc$ (+)

Compute the pointwise sinc function of selected images.
2.4.49 \textit{sinh} (+)

Compute the pointwise hyperbolic sine of selected images.
Example 117: `image.jpg +normalize -3,3 sinh[-1]`

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

2.4.50 $sqr$ (+)

Compute the pointwise square function of selected images.
2.4.51 \textit{sqrt (+)}

Compute the pointwise square root of selected images.
Example 121: `image.jpg + sqrt`

Example 122: `300,1,1,1,'40*x/w+u' + sqrt display_graph 400,300`

### 2.4.52 `sub (+)`

**Arguments:**
• value[%]
• [image]
• ‘formula’
• (no arg)

Subtract specified value, image or mathematical expression to selected images, or compute the pointwise difference of selected images. 
(eq. to ‘-’).

Example 123: image.jpg +sub 30% cut 0,255
Example 124: `image.jpg +mirror x sub[-1] [0]`

Example 125: `image.jpg sub 'i(w/2+0.9*(x-w/2),y)'

[0]: 'image_cl.jpg' (480x320x1x3)
2.4.53 $\tan$ (+)

Compute the pointwise tangent of selected images.

Example 126: image.jpg +mirror x sub

Example 127: image.jpg +normalize {-0.47*pi}, {0.47*pi} tan[-1]
2.4. MATHEMATICAL OPERATORS

Example 128: \[300,1,1,1,'20*x/w+u' + \tan\ display\ graph\ 400,300\]

Tutorial page:

2.4.54 \textit{tanh} (+)

Compute the pointwise hyperbolic tangent of selected images.
2.4.55 \textit{xor} (+)

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

Example 131: \texttt{image.jpg xor 128}
2.5 Values Manipulation

2.5.1 apply_curve

Arguments:

- $0 \leq \text{smoothness} \leq 1, x_0, y_0, x_1, y_1, x_2, y_2, \ldots, x_N, y_N$

Apply curve transformation to image values.

Default values:

- '$\text{smoothness}=1$', '$x_0=0$', '$y_0=100$'.

Example 132: `image.jpg +mirror x xor`
Example 133: `image.jpg +apply_curve 1,0,0,128,255,255,0`

2.5.2 *apply_gamma*

Arguments:

- `gamma > 0`

Apply gamma correction to selected images.
2.5.3  \textit{balance}\_gamma

Arguments:

- \texttt{\_ref\_color1,...}

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

Default value:

- \texttt{\_ref\_color1=128}'.

2.5. VALUES MANIPULATION

Example 135: image.jpg +balance,gamma 128,64,64

2.5.4 cast

Arguments:

- `datatype_source,datatype_target`

Cast datatype of image buffer from specified source type to specified target type. `datatype_source` and `datatype_target` can be `{ uchar | char | ushort | short | uint | int | uint64 | int64 | float | double }.

2.5.5 complex2polar

Compute complex to polar transforms of selected images.
Example 136: \texttt{image.jpg +fft complex2polar[-2,-1] log[-2] shift[-2] 50\%50\%0,0,2 remove[-1]}

2.5.6 \textit{compress \texttt{clut}}

Arguments:

\begin{itemize}
  \item $\maxerror > 0, \avgerror > 0, \maxnbpoints > -8 \mid 0$ (unlimited), \textit{error} \textit{metric} = \{0=L2-norm \mid 1=\text{deltaE	extsubscript{1976}} \mid 2=\text{deltaE	extsubscript{2000}} \}, \textit{reconstruction} \textit{color} \textit{space} = \{0=srgb \mid 1=rgb \mid 2=lab \}, \textit{try} \textit{rbf} \textit{first} = \{0 \mid 1\}
\end{itemize}

Compress selected color LUTs as sequences of colored keypoints.

Default values:

\begin{itemize}
  \item 'max\textunderscore error=1.5', 'avg\textunderscore error=0.75', 'max\textunderscore nb\textunderscore points=2048', 'error\textunderscore metric=2', 'reconstruction\textunderscore colorspace=0' and 'try\textunderscore rbf\textunderscore first=1'.
\end{itemize}

2.5.7 \textit{compress \texttt{rle}}

Arguments:

\begin{itemize}
  \item $\isbinarydata = \{0 \mid 1\}, \textit{maximum\textunderscore sequence\textunderscore length} \geq 0$
\end{itemize}

Compress selected images as 2xN data matrices, using RLE algorithm. Set 'maximum\textunderscore sequence\textunderscore length=0' to disable maximum length constraint.

Default values:

\begin{itemize}
  \item 'is\textunderscore binary\textunderscore data=0' and 'maximum\textunderscore sequence\textunderscore length=0'.
\end{itemize}
2.5. VALUES MANIPULATION

Example 137: `image.jpg resize2dy 100 quantize 4 round +compress_rle , +decompress_rle[-1]`

2.5.8  _cumulate_ (+)

Arguments:

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

Compute the cumulative function of specified image data, optionally along the specified axes.
Example 138: image.jpg +histogram +cumulate[-1] display_graph[-2,-1] 400,300,3

2.5.9  cut (+)

Arguments:

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

Cut values of selected images in specified range.
(eq. to 'cut').
2.5. VALUES MANIPULATION

2.5.10  

**decompress_clut**

**Arguments:**
• \_width>0, \_height>0, \_depth>0, \_reconstruction\_colorspace=\{ 0=srgb | 1=rgb | 2=lab \}

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

**Default values:**

• ‘width=height=depth=33’ and ‘reconstruction\_colorspace=0’.

### 2.5.11 `decompress\_clut\_rbf`

**Arguments:**

• \_width>0, \_height>0, \_depth>0, \_reconstruction\_colorspace=\{ 0=srgb | 1=rgb | 2=lab \}

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

**Default value:**

• ‘width=height=depth=33’ and ‘reconstruction\_colorspace=0’.

### 2.5.12 `decompress\_clut\_pde`

**Arguments:**

• \_width>0, \_height>0, \_depth>0, \_reconstruction\_colorspace=\{ 0=srgb | 1=rgb | 2=lab \}

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

**Default values:**

• ‘width=height=depth=33’ and ‘reconstruction\_colorspace=0’.

### 2.5.13 `decompress\_rle`

Decompress selected data vectors, using RLE algorithm.

### 2.5.14 `discard (+)`

**Arguments:**

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

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

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

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

Example 141: \((1;2;3;4;3;2;1) + \text{discard 2}\)

Example 142: \((1,2,2,2,3,3,3,4,4,4,4,4) + \text{discard x}\)

2.5.15 \textit{eigen2tensor}

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

**Tutorial page:**
https://gmic.eu/tutorial/_eigen2tensor.shtml
2.5.16  **endian (+)**

**Arguments:**

- `_datatype`

Reverse data endianness of selected images, eventually considering the pixel being of the specified datatype. 
'datatype' can be `{ uchar | char | ushort | short | uint | int | uint64 | int64 | float | double }`.

2.5.17  **equalize (+)**

**Arguments:**

- `_nb_levels>`
- `_value_min`
- `_value_max`

Equalize histograms of selected images.
If value range is specified, the equalization is done only for pixels in the specified value range.

**Default values:**

- `nb_levels=256`, `value_min=0%` and `value_max=100%`.

**Example 143:** `image.jpg +equalize`
2.5. VALUES MANIPULATION

Example 144: `image.jpg +equalize 4,0,128`

2.5.18  *fill (+)*

**Arguments:**

- `value1, value2,...`
- `[image]`
- `'formula’`

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

*(eq. to ‘f’)*.
Example 145: 4,4 fill 1,2,3,4,5,6,7

Example 146: 4,4 (1,2,4,5,6,7) fill[-2] [-1]
Example 147: 400,400,1,3 fill "X=x-w/2; Y=y-h/2; R=sqrt(X^2+Y^2); a=atan2(Y,X);
                 if(R<=180,255*abs(cos(c+200*(x/w-0.5)*(y/h-0.5))),850*(a%(0.1*(c+1))))"

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

2.5.19  index (+)

Arguments:

  • { [palette] | palette_name },0<=dithering<=1,map_palette={ 0 | 1 }

Index selected vector-valued images by specified vector-valued palette.
'palette_name' can be { default | hsv | lines | hot | cool | jet | flag | cube | rainbow | algae | amp |
balance | curl | deep | delta | dense | diff | haline | ice | matter | oxy | phase | rain | solar |
speed | tarn | tempo | thermal | topo | turbid | aurora | hocuspocus | srb2 | uzebox }

Default values:

  • ‘dithering=0’ and ‘map_palette=0’.
Example 148: `image.jpg +index 1,1,1`

Example 149: `image.jpg (0;255;255^0;128;255^0;0;255) +index[-2] [-1],1,1`

Tutorial page: [https://gmic.eu/tutorial/_index.shtml](https://gmic.eu/tutorial/_index.shtml)
2.5. VALUES MANIPULATION

2.5.20  *inrange*

**Arguments:**

- `min[%],max[%],include_boundaries={ 0=no | 1=yes }`

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

**Default value:**

- `'include_boundaries=1'`.

![Image](image.jpg)

Example 150: `image.jpg +inrange 25%,75%`

2.5.21  *map* (+)

**Arguments:**

- `[palette],boundary_conditions`
- `palette_name,boundary_conditions`

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

'palette_name' can be { default | hsv | lines | hot | cool | jet | flag | cube | rainbow | algae | amp | balance | curl | deep | delta | dense | diff | gray | haline | ice | matter | oxy | phase | rain | solar | speed | tarn | tempo | thermal | topo | turbid | aurora | hocuspocus | srb2 | uzebox }

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

**Default value:**

- `'boundary_conditions=0'`.

![Image](image_cl.jpg)
Example 151: `image.jpg +luminance map[-1] 3`

Example 152: `image.jpg +rgb2ycbcr split[-1] c (0,255,0) resize[-1] 256,1,1,1,3 map[-4] [-1] remove[-1] append[-3--1] c ycbcr2rgb[-1]`

**Tutorial page:**
https://gmic.eu/tutorial/_map.shtml
2.5.22 mix_channels

Arguments:

- \((a_0, \ldots, a_{MN})\)
- \([\text{matrix}]\)

Apply specified matrix to channels of selected images.

Example 153: `image.jpg +mix_channels (0,1,0;1,0,0;0,0,1)`

2.5.23 negate

Arguments:

- `base_value`
- `(no arg)`

Negate image values.

Default value:

- `'base_value=(undefined)'`
2.5.24  *noise* (+)

**Arguments:**

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

Add random noise to selected images.
`noise_type` can be `{0=gaussian | 1=uniform | 2=salt&pepper | 3=poisson | 4=rice}`.

**Default value:**

- `'noise_type=0'`.
2.5. VALUES MANIPULATION

Example 155: `image.jpg +noise[0] 50,0 +noise[0] 50,1 +noise[0] 10,2 cut 0,255`

Example 156: `300,300,1,3 [0] noise[0] 20,0 noise[1] 20,1 +histogram 100 display graph[-2,-1] 400,300,3`
2.5.25  \textit{noise perlin}

Arguments:

- \texttt{\_scale.x[\%]>0,\_scale.y[\%]>0,\_scale.z[\%]>0,\_seed.x,\_seed.y,\_seed.z}

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

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

Default values:

- \texttt{\_scale.x=\_scale.y=\_scale.z=16} and \texttt{\_seed.x=\_seed.y=\_seed.z=0}.

Example 157: 500,500,1,3 noise\_perlin.

2.5.26  \textit{noise poissondisk}

Arguments:

- \texttt{\_radius[\%]>0,\_max_sample_attempts>0}

Add poisson disk sampling noise to selected images.

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

Default values:

- \texttt{\_radius=8} and \texttt{\_max_sample_attempts=30}.
2.5. VALUES MANIPULATION

Example 158: 300,300 noise poissondisk 8

2.5.27 normp

Arguments:

- \( p \geq 0 \)

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

Default value:

- ‘p=2’.
Chapter 2. List of Commands

Example 159: `image.jpg +normp[0] 0 +normp[0] 1 +normp[0] 2 +normp[0] inf`

2.5.28 norm

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

Example 160: `image.jpg +norm`

Tutorial page:
https://gmic.eu/tutorial/_norm.shtml
2.5. VALUES MANIPULATION

2.5.29  **normalize (+)**

**Arguments:**

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

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

---

**Example 161:** image.jpg split x,2 normalize[-1] 64,196 append x

---

**Tutorial page:**
https://gmic.eu/tutorial/_normalize.shtml

2.5.30  **normalize_sum**

Normalize selected images with a unitary sum.
2.5.31 \textit{not}

Apply boolean not operation on selected images.

\textbf{Example 163}: image.jpg +ge 50\% +not[-1]
2.5.32 orientation

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


Tutorial page: https://gmic.eu/tutorial/_orientation.shtml

2.5.33 oneminus

For each selected image, compute one minus image.
Example 165: `image.jpg normalize 0,1 +oneminus`

### 2.5.34 otsu

**Arguments:**

- `_nb_levels>0`

Hard-threshold selected images using Otsu’s method.
The computed thresholds are returned as a list of values in the status.

**Default value:**

- ‘`nb_levels=256`.’
2.5.35 *polar2complex*

Compute polar to complex transforms of selected images.

2.5.36 *quantize*

Arguments:

- `nb_levels>=1`, `keep_values={ 0 | 1 }`, `quantization_type={ -1=median-cut | 0=k-means | 1=uniform }`

Quantize selected images.

Default value:

- `'keep_values=1' and 'quantization_type=0'`. 
Example 167: `image.jpg luminance +quantize 3`

Example 168: `200,200,1,1,'cos(x/10)+sin(y/10)' +quantize[0] 6 +quantize[0] 4 +quantize[0] 3 +quantize[0] 2`
2.5. VALUES MANIPULATION

2.5.37  quantize_area

Arguments:

- \texttt{min\_area}>0

Quantize selected images such that each flat region has an area greater or equal to `	exttt{min\_area}`.

Default value:

- `	exttt{min\_area=10}`.

Example 169: image.jpg quantize 3 +blur 1 round[-1] +quantize_area[-1] 2

2.5.38  rand (+)

Arguments:

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

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

Arguments:

- source, target

Replace pixel values in selected images.
2.5.40 **replace inf**

**Arguments:**

- _expression_

Replace all infinite values in selected images by specified expression.

**Example 171:** \((1;2;3;4) \text{ replace } 2,3\)

![Image of example 171](image1.png)

2.5.41 **replace nan**

**Arguments:**

- _expression_

Replace all NaN values in selected images by specified expression.

**Example 172:** \((0;1;2) \text{ log } \text{replace inf } 2\)
2.5.42  replace naninf

Arguments:

- _expression

Replace all NaN and infinite values in selected images by specified expression.

2.5.43  replace_seq

Arguments:

- "search_seq","replace_seq"

Search and replace a sequence of values in selected images.
Example 174: \((1,2,3,4,5) +\text{replace}_\text{seq} "2,3,4", "7,8"\)

### 2.5.44 `replace_str`

**Arguments:**

- "search_str","replace_str"

Search and replace a string in selected images (viewed as strings, i.e. sequences of character codes).
Example 175: ('"Hello there, how are you ?"') +replace_str "Hello there","Hi David"

2.5.45  \textit{round} (+)

\textbf{Arguments:}

\begin{itemize}
  \item \texttt{rounding\_value}>0, \texttt{rounding\_type}
  \item (no arg)
\end{itemize}

Round values of selected images.
\texttt{\textasciitilde rounding\_type} can be \{ -1=backward \mid 0=nearest \mid 1=forward \}.

\textbf{Default value:}

\begin{itemize}
  \item \texttt{\textasciitilde rounding\_type=0}'.
\end{itemize}

Example 176: image.jpg +round 100
2.5.46  **roundify**

Arguments:

- \( \gamma \geq 0 \)

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

Default value:

- \('\gamma = 0'\).

Example 177: `image.jpg mul {pi/180} sin +round`
Example 178: 1000 fill '4*x/w' repeat 5 +roundify[0] {$>*0.2} done append c display, graph 400,300

2.5.47 set (+)

Arguments:

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

Set pixel value in selected images, at specified coordinates. 
(*eq. to '='*).
If specified coordinates are outside the image bounds, no action is performed.

Default values:

- 'x=y=z=c=0'.
Example 179: 2,2 set 1,0,0 set 2,1,0 set 3,0,1 set 4,1,1

Example 180: image.jpg repeat 10000 set 255, {u(100)}%, {u(100)}%, 0, {u(100)}% done

2.5.48 threshold

Arguments:
• `value[%,`is_soft={ 0 | 1 }:

Threshold values of selected images.
'soft' can be { 0=hard-thresholding | 1=soft-thresholding }.

**Default value:**

• 'is_soft=0'.

Example 181: image.jpg +threshold[0] 50% +threshold[0] 50%,1

**Tutorial page:**
https://gmic.eu/tutorial/_threshold.shtml

### 2.5.49 `vector2tensor`

Convert selected vector fields to corresponding tensor fields.

### 2.6 Colors Manipulation

#### 2.6.1 `adjust_colors`

**Arguments:**

- `-100<_brightness<_100, -100<_contrast<_100, -100<_gamma<_100, -100<_hue_shift<_100, -100<_saturation<_100, value_min, value_max`

Perform a global adjustment of colors on selected images.
Range of correct image values are considered to be in [value_min,value_max] (e.g. [0,255]).
2.6. COLORS MANIPULATION

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

**Default values:**

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

**Example 182:** image.jpg +adjust_colors 0,30,0,0,30

### 2.6.2 apply_channels

**Arguments:**

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

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

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 value:**

- `'value_action=0'`. 
Example 183: `image.jpg +apply_channels "equalize blur 2",ycbcr`, cbcr

2.6.3 **autoindex**

**Arguments:**

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

Index selected vector-valued images by adapted colormaps.

**Default values:**

- `'dithering=0’ and ‘method=1’`.
2.6. COLORS MANIPULATION

Example 184: `image.jpg +autoindex[0] 4 +autoindex[0] 8 +autoindex[0] 16`

2.6.4 \textit{bayer2rgb}

Arguments:

- \texttt{GM\_smoothness}, \texttt{RB\_smoothness1}, \texttt{RB\_smoothness2}

Transform selected RGB-Bayer sampled images to color images.

Default values:

- ‘\texttt{GM\_smoothness=RB\_smoothness1=1’ and ‘RB\_smoothness2=0.5’}’. 
2.6.5 \textit{deltaE}

Arguments:

\begin{itemize}
  \item $\{\text{image}\}, \text{metric}=\{0=\text{deltaE}\_1976 \mid 1=\text{deltaE}\_2000\}$, \text{“to\_Lab\_command”}
\end{itemize}

Compute the CIE DeltaE color difference between selected images and specified [image]. Argument \text{“to\_Lab\_command”} is a command able to convert colors of [image] into a Lab representation.

Default values:

\begin{itemize}
  \item \text{“metric=1” and “to\_Lab\_command=“srgb2lab””}.
\end{itemize}
2.6. COLORS MANIPULATION

Example 186: image.jpg +blur 2 +deltaE[0] [1],1,srgb2lab

2.6.6 cmy2rgb
Convert color representation of selected images from CMY to RGB.

2.6.7 cmyk2rgb
Convert color representation of selected images from CMYK to RGB.

2.6.8 colorblind

Arguments:

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

Simulate color blindness vision.
2.6.9 colormap

Arguments:

- `nb_levels>=0`, `method={ 0=median-cut | 1=k-means }, sort_vectors`

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 value:

- 'method=1' and 'sort_vectors=1'.
2.6. COLORS MANIPULATION

2.6.10 compose_channels

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

Default value:

- '+'

Example 188: image.jpg +colormap[0] 4 +colormap[0] 8 +colormap[0] 16

Tutorial page:
https://gmic.eu/tutorial/_colormap.shtml
2.6.11  \textit{direction2rgb}

Compute RGB representation of selected 2D direction fields.
2.6. COLORS MANIPULATION

Example 190: image.jpg luminance gradient append c blur 2 orientation +direction2rgb

2.6.12  

*ditheredbw*

Create dithered B&W version of selected images.

Example 191: image.jpg +equalize ditheredbw[-1]
2.6.13 \textit{fill\_color}

\textbf{Arguments:}

\begin{itemize}
  \item \texttt{col1,...,colN}
\end{itemize}

Fill selected images with specified color. \textit{(eq. to \textit{fc}).}

Example 192: \texttt{image.jpg +fill\_color 255,0,255}

\textbf{Tutorial page:}
https://gmic.eu/tutorial/_fill\_color.shtml

2.6.14 \textit{gradient2rgb}

\textbf{Arguments:}

\begin{itemize}
  \item \texttt{_is\_orientation=} \{ 0 \mid 1 \}
\end{itemize}

Compute RGB representation of 2D gradient of selected images.

\textbf{Default value:}

\begin{itemize}
  \item \texttt{‘is\_orientation=0’}.
\end{itemize}
2.6. COLORS MANIPULATION

### 2.6.15 hcy2rgb
Convert color representation of selected images from HCY to RGB.

### 2.6.16 hsi2rgb
Convert color representation of selected images from HSI to RGB.

### 2.6.17 hsi82rgb
Convert color representation of selected images from HSI8 to RGB.

### 2.6.18 hsl2rgb
Convert color representation of selected images from HSL to RGB.

### 2.6.19 hsl82rgb
Convert color representation of selected images from HSL8 to RGB.

### 2.6.20 hsv2rgb
Convert color representation of selected images from HSV to RGB.
2.6.21  **hsv82rgb**
Convert color representation of selected images from HSV8 to RGB.

2.6.22  **int2rgb**
Convert color representation of selected images from INT24 to RGB.

2.6.23  **jzazbz2rgb**

**Arguments:**

- **illuminant**={ 0=D50  |  1=D65  |  2=E  }
- **(no arg)**

Convert color representation of selected images from RGB to Jzazbz.

**Default value:**

- `'illuminant=2'`.

2.6.24  **jzazbz2xyz**
Convert color representation of selected images from RGB to XYZ.

2.6.25  **lab2lch**
Convert color representation of selected images from Lab to Lch.
2.6. COLORS MANIPULATION

2.6.26  *lab2rgb*

**Arguments:**

- **illuminant** =
  0=D50  |  1=D65  |  2=E  
- (no arg)

Convert color representation of selected images from Lab to RGB.

**Default value:**

- 'illuminant=2'.

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

2.6.27  *lab2srgb*

**Arguments:**

- **illuminant** =
  0=D50  |  1=D65  |  2=E  
- (no arg)

Convert color representation of selected images from Lab to sRGB.

**Default value:**

- 'illuminant=2'.

Example 195: (50,50;50,50^-3,3;-3,3^-3,-3;3,3) resize 400,400,1,3,3 lab2srgb
Example 196: \((50,50;50,50^{-3,3};-3,3^{-3,3};3,3)\) resize 400,400,1,3,3 lab2rgb

2.6.28 \textit{lab82srgb}

Arguments:

- \texttt{illuminant=} \{ 0=D50 | 1=D65 | 2=E \}
- (no arg)

Convert color representation of selected images from Lab8 to sRGB.

Default value:

- ‘illuminant=2’.
2.6. COLORS MANIPULATION

2.6.29  \textit{lab2xyz}

\textbf{Arguments:}

- \texttt{illuminant=} \{ 0=D50 \mid 1=D65 \mid 2=E \}
- \texttt{(no arg)}

Convert color representation of selected images from Lab to XYZ.

\textbf{Default value:}

- \texttt{illuminant=2}.

2.6.30  \textit{lab82rgb}

\textbf{Arguments:}

- \texttt{illuminant=} \{ 0=D50 \mid 1=D65 \mid 2=E \}
- \texttt{(no arg)}

Convert color representation of selected images from Lab8 to RGB.

\textbf{Default value:}

- \texttt{illuminant=2}.

2.6.31  \textit{lch2lab}

Convert color representation of selected images from Lch to Lab.
2.6.32  \textit{lch2rgb}

Arguments:

- \texttt{illuminant=\{ 0=D50 | 1=D65 | 2=E \}}
- (no arg)

Convert color representation of selected images from Lch to RGB.

Default value:

- ‘illuminant=2’.

2.6.33  \textit{lch82rgb}

Arguments:

- \texttt{illuminant=\{ 0=D50 | 1=D65 | 2=E \}}
- (no arg)

Convert color representation of selected images from Lch8 to RGB.

Default value:

- ‘illuminant=2’.

2.6.34  \textit{luminance}

Compute luminance of selected sRGB images.

Example 198: \texttt{image.jpg +luminance}
2.6. COLORS MANIPULATION

2.6.35  lightness
Compute lightness of selected sRGB images.

Example 199: image.jpg +lightness

2.6.36  lut_contrast

Arguments:

- \_nb\_colors>1, \_min\_rgb\_value

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'.

2.6.37  map_clut

Arguments:

- [clut] | "clut\_name"

Map specified RGB color LUT to selected images.
2.6.38  \textit{mix\_rgb}

Arguments:

\begin{itemize}
  \item $a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}, a_{31}, a_{32}, a_{33}$
\end{itemize}

Apply 3x3 specified matrix to RGB colors of selected images.

Default values:

\begin{itemize}
  \item $a_{11}=1$, $a_{12}=a_{13}=a_{21}=0$, $a_{22}=1$, $a_{23}=a_{31}=a_{32}=0$ and $a_{33}=1$.
\end{itemize}
2.6. COLORS MANIPULATION

Example 201: image.jpg +mix_rgb 0,1,0,1,0,0,0,0,1

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

2.6.39  palette

Arguments:

- `palette_name` | `palette_number`

Input specified color palette at the end of the image list.
'palette_name' can be { default | hsv | lines | hot | cool | jet | flag | cube | rainbow | parula | spring |
| summer | autumn | winter | bone | copper | pink | vga | algae | amp | balance | curl | deep |
| delta | dense | diff | gray | haline | ice | matter | oxy | phase | rain | solar | speed | tarn | tempo |
| thermal | topo | turbid | aurora | hocuspocus | srb2 | uzebox | amiga7800 | amiga7800mess |
| fornaxvoid1 |
2.6.40  pseudogray

Arguments:

- \texttt{max\_increment} \geq 0, \texttt{JND\_threshold} \geq 0, \texttt{bits\_depth} > 0

Generate pseudogray colormap with specified increment and perceptual threshold. If \texttt{JND\_threshold} is 0, no perceptual constraints are applied.

Default values:

- \texttt{max\_increment=5'}, \texttt{JND\_threshold=2.3'} and \texttt{bits\_depth=8'}.

2.6.41  replace\_color

Arguments:

- \texttt{tolerance\[%\]} \geq 0, \texttt{smoothness\[%\]} \geq 0, \texttt{src1, src2,...,dest1, dest2,...}

Replace pixels from/to specified colors in selected images.
2.6. COLORS MANIPULATION

2.6.42 retinex

Arguments:

- `_value_offset>0, _colors=hei | hsv | lab | lrgb | rgb | ycbcr`, `0<=_min_cut<=100, 0<=_max_cut<=100, _sigma_low>0, _sigma_mid>0, _sigma_high>0`

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

Default values:

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

2.6.43 rgb2bayer

Arguments:

- `_start_pattern=0, _color_grid=0`

Transform selected color images to RGB-Bayer sampled images.

Default values:

- `'start_pattern=0 and 'color_grid=0'`. 
2.6.44 *rgb2cmy*

Convert color representation of selected images from RGB to CMY.

Example 206: image.jpg rgb2cmy split c
2.6. COLORS MANIPULATION

2.6.45 rgb2cmyk

Convert color representation of selected images from RGB to CMYK.

Example 207: image.jpg rgb2cmyk split c

Example 208: image.jpg rgb2cmyk split c fill[3] 0 append c cmyk2rgb
2.6.46 rgb2hcy

Convert color representation of selected images from RGB to HCY.

Example 209: image.jpg rgb2hcy split c

2.6.47 rgb2hsi

Convert color representation of selected images from RGB to HSI.
2.6. COLORS MANIPULATION

Example 210: image.jpg rgb2hsi split c

2.6.48  *rgb2hsi8*

Convert color representation of selected images from RGB to HSI8.

Example 211: image.jpg rgb2hsi8 split c
2.6.49  *rgb2hsl*

Convert color representation of selected images from RGB to HSL.

**Example 212:** `image.jpg rgb2hsl split c`

**Example 213:** `image.jpg rgb2hsl +split c add[-3] 100 mod[-3] 360 append[-3--1] c hsl2rgb`
2.6.50  *rgb2hsl8*

Convert color representation of selected images from RGB to HSL8.

**Example 214**: `image.jpg rgb2hsl8 split c`

2.6.51  *rgb2hsv*

Convert color representation of selected images from RGB to HSV.
2.6.52  **rgb2hsv8**

Convert color representation of selected images from RGB to HSV8.
2.6. COLORS MANIPULATION

Example 217: `image.jpg rgb2hsv split c`

2.6.53 rgb2int

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

Example 218: `image.jpg rgb2int`
2.6.54  rgb2jzazbz

Arguments:

- illuminant=\{ 0=D50 \ | \ 1=D65 \ | \ 2=E \}
- (no arg)

Convert color representation of selected images from RGB to Jzazbz.

Default value:

- ‘illuminant=2’.

2.6.55  rgb2lab

Arguments:

- illuminant=\{ 0=D50 \ | \ 1=D65 \ | \ 2=E \}
- (no arg)

Convert color representation of selected images from RGB to Lab.

Default value:

- ‘illuminant=2’.

2.6.56  rgb2lab8

Arguments:

- illuminant=\{ 0=D50 \ | \ 1=D65 \ | \ 2=E \}
- (no arg)

Convert color representation of selected images from RGB to Lab8.

Default value:

- ‘illuminant=2’.
Example 219: image.jpg rgb2lab8 split c

2.6.57 rgb2lch

Arguments:

- **illuminant**={ 0=D50 | 1=D65 | 2=E }  
- (no arg)

Convert color representation of selected images from RGB to Lch.

Default value:

- ‘illuminant=2’.
2.6.58  \texttt{rgb2lch8}

Arguments:

\begin{itemize}
\item \texttt{illuminant=} \{ 0=D50 | 1=D65 | 2=E \}
\item (no arg)
\end{itemize}

Convert color representation of selected images from RGB to Lch8.

Default value:

\begin{itemize}
\item \texttt{‘illuminant=2’}.
\end{itemize}
2.6. COLORS MANIPULATION

2.6.59 *rgb2luv*
Convert color representation of selected images from RGB to LUV.

Example 222: `image.jpg rgb2luv split c`
2.6.60  *rgb2ryb*

Convert color representation of selected images from RGB to RYB.

Example 223: `image.jpg rgb2ryb split c`

2.6.61  *rgb2srgb*

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

2.6.62  *rgb2xyz*

**Arguments:**

- `illuminant={ 0=D50 | 1=D65 | 2=E }`
- `(no arg)`

Convert color representation of selected images from RGB to XYZ.

**Default value:**

- `'illuminant=2'`. 
2.6. COLORS MANIPULATION

Example 224: image.jpg rgb2xyz split c

2.6.63 rgb2xyz8

Arguments:

- `illuminant={ 0=D50 | 1=D65 | 2=E }`
- `(no arg)`

Convert color representation of selected images from RGB to XYZ8.

Default value:

- `'illuminant=2'`.
2.6.64  *rgb2yiq*

Convert color representation of selected images from RGB to YIQ.

Example 225: image.jpg rgb2xyz8 split c
2.6. COLORS MANIPULATION

2.6.65  rgb2yiq8

Convert color representation of selected images from RGB to YIQ8.

Example 227: image.jpg rgb2yiq8 split c

2.6.66  rgb2ycbcr

Convert color representation of selected images from RGB to YCbCr.
2.6.67  *rgb2yuv*

Convert color representation of selected images from RGB to YUV.

Example 228: `image.jpg rgb2ycbcr split c`

Example 229: `image.jpg rgb2yuv split c`
2.6.68 \textit{rgb2yuv8}

Convert color representation of selected images from RGB to YUV8.

Example 230: \texttt{image.jpg rgb2yuv8 split c}

2.6.69 \textit{remove\_opacity}

Remove opacity channel of selected images.

2.6.70 \textit{ryb2rgb}

Convert color representation of selected images from RYB to RGB.

2.6.71 \textit{select\_color}

\textbf{Arguments:}

- \texttt{tolerance[\%] = 0, col1, \ldots, colN}

Select pixels with specified color in selected images.
Example 231: `image.jpg +select_color 40,204,153,110`

**Tutorial page:**

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

### 2.6.72 sepia

Apply sepia tones effect on selected images.
2.6. COLORS MANIPULATION

Example 232: `image.jpg sepia`

2.6.73 solarize
Solarize selected images.

Example 233: `image.jpg solarize`
2.6.74  *split_colors*

**Arguments:**

- \( \text{tolerance} \geq 0, \text{max\_nb\_outputs} > 0, \text{min\_area} > 0 \)

Split selected images as several images containing a single color. One selected image can be split as at most \( \text{max\_nb\_outputs} \) images. Output images are sorted by decreasing area of extracted color regions and have an additional alpha-channel.

**Default values:**

- ‘tolerance=0’, ‘max\_nb\_outputs=256’ and ‘min\_area=8’.

Example 234: `image.jpg quantize 5 +split_colors , display_rgba`

2.6.75  *split_opacity*

Split color and opacity parts of selected images.

2.6.76  *srgb2lab*

**Arguments:**

- \( \text{illuminant} = \{ 0=D50 | 1=D65 | 2=E \} \)
  - (no arg)

Convert color representation of selected images from sRGB to Lab.

**Default value:**

- ‘illuminant=2’.
2.6. COLORS MANIPULATION

Example 235: `image.jpg srgb2lab split c`

Example 236: `image.jpg srgb2lab +split c mul[-2,-1] 2.5 append[-3--1] c lab2srgb`

2.6.77 `srgb2lab8`

Arguments:
• **illuminant**={ 0=D50 | 1=D65 | 2=E }
  
  *(no arg)*

Convert color representation of selected images from sRGB to Lab8.

**Default value:**

• `'illuminant=2'`.

### 2.6.78  `srgb2rgb`

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

### 2.6.79  `to_a`

Force selected images to have an alpha channel.

### 2.6.80  `to_color`

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

### 2.6.81  `to_colormode`

**Arguments:**

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

Force selected images to be in a given color mode.

**Default value:**

• `'mode=0'`.

### 2.6.82  `to_gray`

Force selected images to be in GRAY mode.
2.6. COLORS MANIPULATION

2.6.83 \textit{to\_graya}
Force selected images to be in GRAYA mode.

2.6.84 \textit{to\_pseudogray}

\textbf{Arguments:}

\begin{itemize}
  \item \texttt{max\_step}>0, \texttt{is\_perceptual\_constraint}={\texttt{0} | \texttt{1}}, \texttt{bits\_depth}>0
\end{itemize}

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

\textbf{Default values:}

\begin{itemize}
  \item \texttt{max\_step=5}, \texttt{is\_perceptual\_constraint=1} and \texttt{bits\_depth=8}.
\end{itemize}

The original pseudo-gray technique has been introduced by Rich Franzen [http://r0k.us/graphics/pseudoGrey.html]. Extension of this technique to arbitrary increments for more tones, has been done by David Tschumperlé.

2.6.85 \textit{to\_rgb}
Force selected images to be in RGB mode.

2.6.86 \textit{to\_rgba}
Force selected images to be in RGBA mode.
2.6.87  \texttt{transfer\_histogram}

\textbf{Arguments:}

- \texttt{[reference\_image],\ nb\_levels>0,\ color\_channels}

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

\textbf{Default value:}

- 'nb\_levels=256' and 'color\_channels=all'.

\textbf{Example 238:} image.jpg 100,100,1,3,“u([256,200,100])” +transfer\_histogram[0] [1]

2.6.88  \texttt{transfer\_pca}

\textbf{Arguments:}

- \texttt{[reference\_image],\ color\_channels}

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'.

\textbf{Default value:}

- 'color\_channels=all'.

2.6. COLORS MANIPULATION

Example 239: sample lena, earth + transfer_pca[0] [1]

2.6.89 transfer_rgb

Arguments:

- [target], gamma>=0, regularization>=0, luminosity_constraints>=0, _rgb_resolution>=0, is_constraints={ 0 | 1 }

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

Default values:

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

[Images of lena, earth, and lena_c1]
2.6.90 \textit{xyz2jzazbz}
Convert color representation of selected images from XYZ to RGB.

2.6.91 \textit{xyz2lab}

\textbf{Arguments}:

- \texttt{illuminant=\{ 0=D50 | 1=D65 | 2=E \}}
- (no arg)

Convert color representation of selected images from XYZ to Lab.

\textbf{Default value}:

- ‘illuminant=2’.

2.6.92 \textit{xyz2rgb}

\textbf{Arguments}:

- \texttt{illuminant=\{ 0=D50 | 1=D65 | 2=E \}}
- (no arg)

Convert color representation of selected images from XYZ to RGB.

\textbf{Default value}:

- ‘illuminant=2’.
2.6.93  \textit{xyz2rgb}

\textbf{Arguments:}

- \texttt{illuminant=\{ 0=D50 | 1=D65 | 2=E \}}
- \texttt{(no arg)}

Convert color representation of selected images from XYZ to RGB.

\textbf{Default value:}

- ‘illuminant=2’.

2.6.94  \textit{ycbcr2rgb}

Convert color representation of selected images from YCbCr to RGB.

2.6.95  \textit{yiq2rgb}

Convert color representation of selected images from YIQ to RGB.

2.6.96  \textit{yiq82rgb}

Convert color representation of selected images from YIQ8 to RGB.

2.6.97  \textit{yuv2rgb}

Convert color representation of selected images from YUV to RGB.

2.6.98  \textit{yuv82rgb}

Convert selected images from YUV8 to RGB color bases.

2.7  Geometry Manipulation

2.7.1  \textit{append (+)}

\textbf{Arguments:}

- \{image\},axis,centering
- axis, centering

Append specified image to selected images, or all selected images together, along specified axis. (eq. to ‘+’).

‘axis’ can be \{ x | y | z | c \}.

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

\textbf{Default value:}

- ‘centering=0’.
Example 241: \texttt{image.jpg split y,10 reverse append y}

Example 242: \texttt{image.jpg repeat 5 +rows[0] 0,\{10+18*\} done remove[0] append x,0.5}
2.7. GEOMETRY MANIPULATION

Example 243: `image.jpg append[0] [0],y`

2.7.2 `append_tiles`

Arguments:

- `_M>=0, N>=0, 0<_centering.x<1, 0<_centering.y<1`

Append MxN selected tiles as new images.
If `N` is set to 0, number of rows is estimated automatically.
If `_M` is set to 0, number of columns is estimated automatically.
If `_M` and `N` are both set to '0', auto-mode is used.
If `_M` or `N` is set to 0, only a single image is produced.
`centering.x` and `centering.y` tells about the centering of tiles when they have different sizes.

Default values:

- `'M=0', 'N=0', 'centering_x=centering_y=0.5'.
2.7.3 apply_scales

Arguments:

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

Apply specified command on different scales of selected images. 'interpolation' can be { 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }.

Default value:

- ‘min_scale=25\%’, ‘max_scale=100\%’ and ‘interpolation=3’.
2.7. AUTOCROP (+)

Arguments:

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

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

Arguments:

- threshold[%], min_area[%] > 0, is_high_connectivity={ 0 | 1 }, output_type={ 0=crop | 1=segmentation | 2=coordinates }

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

Default values:

- ‘threshold=0%’, ‘min_area=0.1%’, ‘is_high_connectivity=0’ and ‘output_type=1’.
2.7.6 autocrop_seq

Arguments:

- value1, value2, ... | auto

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

Default value:

- auto mode.
Example 248: `image.jpg +fill[-1] 0 ellipse[-1] 50%,50%,30%,20%,0,1,1 autocrop_seq 0`

2.7.7 **channels (+)**

Arguments:

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

Keep only specified channels of selected images. Dirichlet boundary is used when specified channels are out of range.
2.7.8 \textit{columns} (+)

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

Example 251: 'image.jpg' columns -25%, 50%  

2.7.9  \textit{crop} (+)  

Arguments:  
\begin{itemize}  
\item x0(\%), x1(\%), \texttt{boundary\_conditions}  
\item x0(\%), y0(\%), x1(\%), y1(\%), \texttt{boundary\_conditions}  
\item x0(\%), y0(\%), z0(\%), x1(\%), y1(\%), z1(\%), \texttt{boundary\_conditions}  
\item x0(\%), y0(\%), z0(\%), c0(\%), x1(\%), y1(\%), z1(\%), c1(\%), \texttt{boundary\_conditions}  
\end{itemize}  

Crop selected images with specified region coordinates.  
(eq. to 'z').  
'boundary\_conditions' can be \{ 0=dirichlet \ | 1=neumann \ | 2=periodic \ | 3=mirror \}.  

Default value:  
\begin{itemize}  
\item 'boundary\_conditions=0'.  
\end{itemize}
Example 252: `image.jpg` +crop -230,-230,280,280,1
crop[0] -230,-230,280,280,0

Example 253: `image.jpg` crop 25%,25%,75%,75%

2.7.10 diagonal

Transform selected vectors as diagonal matrices.
2.7.11  *elevate*

Arguments:

- \texttt{\_depth, is\_plain=\{ 0 \mid 1 \}, is\_colored=\{ 0 \mid 1 \}}

Elevate selected 2D images into 3D volumes.

Default values:

- \texttt{‘depth=64’, ‘is\_plain=1’ and ‘is\_colored=1’}.

2.7.12  *expand* \_x

Arguments:

- \texttt{size\_x>0, boundary\_conditions=\{ 0=dirichlet \mid 1=neumann \mid 2=periodic \mid 3=mirror \}}

Expand selected images along the x-axis.

Default value:

- \texttt{‘boundary\_conditions=1’}.
2.7. GEOMETRY MANIPULATION

2.7.13 expand_xy

Arguments:

- size\geq0, boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }

Expand selected images along the xy-axes.

Default value:

- ‘boundary_conditions=1’.
2.7.14  \textit{expand} \textit{xyz}

**Arguments:**

- \textit{size} \geq 0, \textit{boundary_conditions}=\{ 0=\text{dirichlet} \mid 1=\text{neumann} \mid 2=\text{periodic} \mid 3=\text{mirror} \}

Expand selected images along the xyz-axes.

**Default value:**

- ‘\textit{boundary_conditions}=1’.

2.7.15  \textit{expand} \textit{y}

**Arguments:**

- \textit{size}_y \geq 0, \textit{boundary_conditions}=\{ 0=\text{dirichlet} \mid 1=\text{neumann} \mid 2=\text{periodic} \mid 3=\text{mirror} \}

Expand selected images along the y-axis.

**Default value:**

- ‘\textit{boundary_conditions}=1’.
2.7.16 expand $z$

Arguments:

- $size_z > 0$, $boundary\_conditions = \{ 0 = \text{dirichlet} | 1 = \text{neumann} | 2 = \text{periodic} | 3 = \text{mirror} \}$

Expand selected images along the $z$-axis.

Default value:

- ‘boundary\_conditions=1’.

2.7.17 extract

Arguments:

- "condition", $output\_type = \{ 0 = \text{xyzc-coordinates} | 1 = \text{xyz-coordinates} | 2 = \text{scalar-values} | 3 = \text{vector-values} \}$

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’.
2.7.18  \textit{extract}\_\textit{region}

\textbf{Arguments:}

- \texttt{[label\_image]}, \texttt{extract\_xyz\_coordinates={ 0 | 1 }, label\_1, ..., label\_M}

Extract all pixels of selected images whose corresponding label in \texttt{[label\_image]} is equal to \texttt{label\_m}, and output them as \texttt{M} column images.

\textbf{Default value:}

- \texttt{‘extract\_xyz\_coordinates=0’}.
2.7. GEOMETRY MANIPULATION

Example 259: image.jpg +blur 3 quantize. 4,0 +extract_region[0] [1],0,1,3

2.7.19 montage

Arguments:

- "layout code", montage_mode={ 0=centering<1 | 2=scale+2<=3 }, output_mode={ 0=single layer | 1=multiple layers }, "processing_command"

Create a single image montage from selected images, according to specified layout code: - 'X' to assemble all images using an automatically estimated layout. - 'H' to assemble all images horizontally. - 'V' to assemble all images vertically. - 'A' to assemble all images as an horizontal array. - 'B' to assemble all images as a vertical array. - 'Ha:b' to assemble two blocks 'a' and 'b' horizontally. - 'Va:b' to assemble two blocks 'a' and 'b' vertically. - 'Ra' to rotate a block 'a' by 90 deg. ('RRA' for 180 deg. and 'RRRA' for 270 deg.). - 'Ma' to mirror a block 'a' along the X-axis ('MRRa' for the Y-axis).

A block 'a' can be an image index (treated periodically) or a nested layout expression 'Hb:c', 'Vb:c', 'Rb' or 'Mb' itself.

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

Default values:

- 'layout_code=X', 'montage_mode=2', output_mode='0' and 'processing.command=""'.

2.7.20  mirror (+)

Arguments:

- \{ x \mid y \mid z \} \ldots \{ x \mid y \mid z \}

Mirror selected images along specified axes.
2.7. GEOMETRY MANIPULATION

Example 261: `image.jpg +mirror y +mirror[0] c`

Example 262: `image.jpg +mirror x +mirror y append,tiles 2,2`

2.7.21 permute (+)

Arguments:
• **permutation_string**

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

Example 263: image.jpg permute yxzc

2.7.22  **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

Resize selected images with specified geometry.

(eq. to ’r’).

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

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

’ax, ay, az, ac’ set the centering along each axis when ’interpolation=0 or 4’ (set to ’0’ by default, must be defined in range [0,1]).

Default values:

- ’interpolation=1’, ’boundary_conditions=0’ and ’ax=ay=az=ac=0’.
2.7.23  \textit{resize as image}

\textbf{Arguments:}

- \texttt{[reference], interpolation, boundary_conditions, ax, ay, az, ac}

Resize selected images to the geometry of specified \texttt{[reference]} image. 
\textit{(eq. to ‘xi’)}.

\textbf{Default values:}

- ‘interpolation=1’, ‘boundary_conditions=0’ and ‘ax=ay=az=ac=0’.
2.7.24 \textit{resize\_mn}

Arguments:

- \texttt{width[]}\%\geq 0, \texttt{height[]}\%\geq 0, \texttt{depth[]}\%\geq 0, \texttt{B\_value}, \texttt{C\_value}

Resize selected images with Mitchell-Netravali filter (cubic).
For details about the method, see: https://de.wikipedia.org/wiki/Mitchell-Netravali-Filter

Default values:

- \texttt{\'height=100\%'}, \texttt{\'depth=100\%'}, \texttt{\'B=0.3333'} \texttt{and \'C=0.3333\'}. 
2.7. GEOMETRY MANIPULATION

Example 266: `image.jpg resize2dx 32 resize_mn 800%,800%`

2.7.25 `resize_pow2`

Arguments:

- `_interpolation`, `_boundary_conditions`, `_ax`, `_ay`, `_az`, `_ac`

Resize selected images so that each dimension is a power of 2. 'interpolation' can be `{ -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }`. 'boundary_conditions' has different meanings, according to the chosen 'interpolation' mode: . When 'interpolation=={ -1 | 1 | 2 | 4 }', 'boundary_conditions' is meaningless. . When 'interpolation==0', 'boundary_conditions' can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`. . When 'interpolation=={ 3 | 5 | 6 }', 'boundary_conditions' can be `{ 0=none | 1=neumann }`. 'ax,ay,az,ac' set the centering along each axis when 'interpolation==0' (set to '0' by default, must be defined in range [0,1]).

Default values:

- 'interpolation=0', 'boundary_conditions=0' and 'ax=ay=az=ac=0'.
2.7.26 resize_ratio2d

Arguments:

- width>0, height>0, mode={ 0=inside | 1=outside | 2=padded }, 0=<interpolation<=6

Resize selected images while preserving their aspect ratio.
(eq. to ‘rr2d’).

Default values:

- ‘mode=0’ and ‘interpolation=6’.

2.7.27 resize2dx

Arguments:

- width[%]>0, interpolation, boundary_conditions, ax, ay, az, ac

Resize selected images along the x-axis, preserving 2D ratio.
(eq. to ‘r2dx’).

‘interpolation’ can be {-1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos}.
‘boundary_conditions’ has different meanings, according to the chosen ‘interpolation’ mode: . When ‘interpolation=={ -1 | 1 | 2 | 4 }’, ‘boundary_conditions’ is meaningless. . When ‘interpolation==0’, ‘boundary_conditions’ can be { 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }. . When ‘interpolation=={ 3 | 5 | 6 }’, ‘boundary_conditions’ can be { 0=none | 1=neumann }.
‘ax, ay, az, ac’ set the centering along each axis when ‘interpolation=0’ (set to ‘0’ by default, must be defined in range [0,1]).
2.7. GEOMETRY MANIPULATION

Default values:

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

Example 268: image.jpg +resize2dx 100,2 append x

2.7.28 resize2dy

Arguments:

- height[%] > 0, interpolation, boundary_conditions, ax, ay, az, ac

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

- `interpolation` can be {-1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos}.
- `boundary_conditions` has different meanings, according to the chosen `interpolation` mode: . When `interpolation={-1 | 1 | 2 | 4}`, `boundary_conditions` is meaningless. . When `interpolation=0`, `boundary_conditions` can be {0=dirichlet | 1=neumann | 2=periodic | 3=mirror}. . When `interpolation={3 | 5 | 6}`, `boundary_conditions` can be {0=none | 1=neumann}.
- `ax,ay,az,ac` set the centering along each axis when `interpolation=0` (set to ‘0’ by default, must be defined in range [0,1]).

Default values:

- `interpolation=3`, `boundary_conditions=0` and `ax=ay=az=ac=0`.
2.7.29  resize3dx

Arguments:

- width[%]$>0$, interpolation, boundary_conditions, ax, ay, az, ac

Resize selected images along the x-axis, preserving 3D ratio. (eq. to ‘r3dx’).

'interpolation' can be \{ -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos \}. ‘boundary_conditions’ has different meanings, according to the chosen ‘interpolation’ mode : . When ‘interpolation==\{-1 | 1 | 2 | 4 \}’, ‘boundary_conditions’ is meaningless. . When ‘interpolation==0’, ‘boundary_conditions’ can be \{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror \}. . When ‘interpolation==\{ 3 | 5 | 6 \}’, ‘boundary_conditions’ can be \{ 0=none | 1=neumann \}.

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

Default values:

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

2.7.30  resize3dy

Arguments:

- height[%]$>0$, interpolation, boundary_conditions, ax, ay, az, ac
2.7. GEOMETRY MANIPULATION

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

(eq. to ‘r3dy’).

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

‘boundary_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 }.

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

Default values:

• ‘interpolation=3’, ‘boundary_conditions=0’ and ‘ax=ay=az=ac=0’.

2.7.31 resize3dz

Arguments:

• depth[%]>0, interpolation, boundary_conditions, ax, ay, az, ac

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

(eq. to ‘r3dz’).

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

‘boundary_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 }.

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

Default values:

• ‘interpolation=3’, ‘boundary_conditions=0’ and ‘ax=ay=az=ac=0’.

2.7.32 rotate (+)

Arguments:

• angle, interpolation, boundary_conditions, center_x[%], center_y[%]
• u,v,w,angle, interpolation, boundary_conditions, center_x[%], center_y[%], center_z[%]

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

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

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

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

Default values:

• ‘interpolation=1’, ‘boundary_conditions=0’ and ‘center.x=center.y=(undefined)’.
2.7.33  **rotate tileable**

**Arguments:**

- \texttt{angle,max_size_factor}>0

Rotate selected images by specified angle and make them tileable. If resulting size of an image is too big, the image is replaced by a 1x1 image.

**Default values:**

- \texttt{max_size_factor=8}.

2.7.34  **rows (+)**

**Arguments:**

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

Keep only specified rows of selected images. Dirichlet boundary conditions are used when specified rows are out of range.
2.7.35 *scale2x*

Resize selected images using the Scale2x algorithm.

Example 272: `image.jpg threshold 50% resize 50%,50% +scale2x`
2.7.36 **scale3x**

Resize selected images using the Scale3x algorithm.

**Example 273:** image.jpg threshold 50% resize 33%,33% +scale3x

2.7.37 **scale_dcci2x**

**Arguments:**

- **edge_threshold**>=0, **exponent**>0, **extend_lpx**={ 0=false | 1=true }

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

**Default values:**

- ‘edge_threshold=1.15’, ‘exponent=5’ and ‘extend_lpx=0’.
2.7. GEOMETRY MANIPULATION

2.7.38 seamcarve

Arguments:

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

Resize selected images with specified 2D geometry, using the seam-carving algorithm.

Default values:

- 'height=100%', 'is_priority_channel=0', 'is_antialiasing=1' and 'maximum_seams=25%'.

Example 274: `image.jpg +scale_dcci2x`,
2.7.39  shift (+)

Arguments:

- \( vx\% \), \( vy\% \), \( vz\% \), \( vc\% \), boundary_conditions, interpolation=
  
  \( 0=\text{nearest neighbor} \mid 1=\text{linear} \)

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 \( \{ 0=\text{dirichlet} \mid 1=\text{neumann} \mid 2=\text{periodic} \mid 3=\text{mirror} \} \).

Default value:

- 'boundary_conditions=0' and 'interpolation=0'.
2.7. GEOMETRY MANIPULATION

Example 276: image.jpg +shift[0] 50%,50%,0,0,0 +shift[0] 50%,50%,0,0,1 +shift[0] 50%,50%,0,0,2

2.7.40 shrink\_x

Arguments:

- size\_x > 0

Shrink selected images along the x-axis.
2.7.41  \textit{shrink\_xy}

Arguments:

\begin{itemize}
  \item \texttt{size} \geq 0
\end{itemize}

Shrink selected images along the xy-axes.
2.7. GEOMETRY MANIPULATION

2.7.42 shrink\_xyz

**Arguments:**

- \( \text{size}\geq0 \)

Shrink selected images along the xyz-axes.

2.7.43 shrink\_y

**Arguments:**

- \( \text{size}_y\geq0 \)

Shrink selected images along the y-axis.

Example 278: `image.jpg shrink\_xy 30`
2.7.44  **shrink** \(_z\)

**Arguments:**

- \(\text{size}_{z}>0\)

Shrink selected images along the \(z\)-axis.

2.7.45  **slices** (+)

**Arguments:**

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

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

2.7.46  **sort** (+)

**Arguments:**

- \(\text{ordering} = \{ + \, | \, - \}, \text{axis} = \{ x \, | \, y \, | \, z \, | \, c \}\)

Sort pixel values of selected images. If ‘axis’ is specified, the sorting is done according to the data of the first column/row/slice/channel of selected images.
Default values:

- 'ordering=+' and 'axis=(undefined)'.

Example 280: 64 rand 0,100 +sort display_graph 400,300,3

2.7.47 split (+)

Arguments:

- \{x | y | z | c\}...{x | y | z | c}, _split_mode
- keep_splitting_values={+ | -},{x | y | z | c}...{x | y | z | c}, value1, value2,...
- (no arg)

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

(eq. to 's').

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

Default value:

- 'split_mode=-1'.
Example 281: `image.jpg split c`

Example 282: `image.jpg split y,3`
Example 283: `image.jpg split x,-128`

Example 284: `1,20,1,"1,2,3,4" +split -,2,3 append[1--1] y`
Example 285: \((1,2,2,3,3,3,4,4,4,4) + \text{split } x,0 \text{ append[1--1] y}\)

2.7.48  \textit{split} \textit{tiles}

Arguments:

- \(M\neq 0, N\neq 0, \text{is homogeneous} = \{0 \mid 1\}\)

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 \(\text{is homogeneous}=0\).
Example 286: image.jpg +local split_tiles 5,4 blur 3,0 sharpen 700 append_tiles 4,5 endlocal

2.7.49 **undistort**

**Arguments:**

- $-1\leq \text{amplitude}\leq 1$, $\text{aspect\_ratio}$, $\text{zoom}$, $\text{center\_x}\%$, $\text{center\_y}\%$

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


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

**Default values:**

- `'amplitude=0.25', 'aspect\_ratio=0', 'zoom=0', 'center\_x=\text{center\_y}=50\%'` and `'boundary_conditions=0'`.

2.7.50 **unroll (+)**

**Arguments:**

- $\text{axis}=\{ \text{x} \mid \text{y} \mid \text{z} \mid \text{c} \}$

Unroll selected images along specified axis. 

*(eq. to 'y').*
Default value:

- ‘axis=y’.

Example 287: \((1,2,3;4,5,6;7,8,9) +\text{unroll y}\)

2.7.51 `upscale\_smart`

Arguments:

- `width[\%], height[\%], depth, smoothness>=0, anisotropy=[0,1], sharpening>=0`

Upscale selected images with an edge-preserving algorithm.

Default values:

- ‘height=100\%', ‘depth=100\%', ‘smoothness=2’, ‘anisotropy=0.4' and ‘sharpening=10'.
2.7. GEOMETRY MANIPULATION

Example 288: `image.jpg resize2dy 100 +upscale_smart 500%,500% append x`

2.7.52 warp (+)

Arguments:

- `[warping_field], mode, interpolation, boundary_conditions, nb_frames>0`

Warp selected images with specified displacement field.

- `mode` can be `{ 0=backward-absolute | 1=backward-relative | 2=forward-absolute | 3=forward-relative }`
- `interpolation` can be `{ 0=nearest-neighbor | 1=linear | 2=cubic }`
- `boundary_conditions` can be `{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }`

Default values:

- `mode=0', 'interpolation=1', 'boundary_conditions=1' and 'nb_frames=1'`. 
2.7.53  **warp_patch**

**Arguments:**

- `[warping_field],patch_width>=1,patch_height>=1,patch_depth>=1,_std_factor>=0,boundary_conditions`.

Patch-warp selected images, with specified 2D or 3D displacement field (in backward-absolute mode). Argument 'std_factor' sets the std of the gaussian weights for the patch overlap, equal to 'std = std_factor*patch_size'.

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

**Default values:**

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

2.7.54  **warp_rbf**

**Arguments:**

- `xs0[%,]ys0[%,]xt0[%,]yt0[%,]...,xsN[%,]ysN[%,]xtN[%,]ytN[%]`
Warp selected images using RBF-based interpolation. Each argument \((xsk, ysk)-(xtk, ytk)\) corresponds to the coordinates of a keypoint respectively on the source and target images. The set of all keypoints define the overall image deformation.

Example 290: image.jpg +warp_rbf  
0,0,0,0,0,100%,0,100%,0,100%,0,100%,100%,100%,0,100%,0,100%,50%,50%,70%,50%,25%,25%,25%,75%

### 2.8 Filtering

#### 2.8.1 bandpass

**Arguments:**

- \(_\text{min}\_freq\)%, \(_\text{max}\_freq\)%

Apply bandpass filter to selected images.

**Default values:**

- ‘\(_\text{min}\_freq=0\)’ and ‘\(_\text{max}\_freq=20\)%’. 
Example 291: image.jpg bandpass 1%, 3%

Tutorial page:
https://gmic.eu/tutorial/_bandpass.shtml

2.8.2 bilateral (+)

Arguments:

- [guide], std_deviation_s[%] > 0, std_deviation_r[%] > 0, sampling_s = 0, sampling_r = 0
- std_deviation_s[%] > 0, std_deviation_r[%] > 0, sampling_s = 0, sampling_r = 0

Blur selected images by anisotropic (eventually joint/cross) bilateral filtering.
If a guide image is provided, it is used 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.
2.8.3  blur (+)

Arguments:

- `std_deviation>=0\%`, `boundary_conditions`, `kernel`
- `axes, std_deviation>=0\%`, `boundary_conditions`, `kernel`

Blur selected images by a quasi-gaussian or gaussian filter (recursive implementation). 
(\textit{eq. to ‘b’}).
`boundary_conditions` can be \{ 0=dirichlet \mid 1=neumann \}.
`kernel` can be \{ 0=quasi-gaussian (faster) \mid 1=gaussian \}.
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:

- ‘boundary_conditions=1’ and ‘kernel=0’.
Example 293: `image.jpg +blur 5,0 +blur[0] 5,1`

Example 294: `image.jpg +blur y,10%`

Tutorial page:
https://gmic.eu/tutorial/_blur.shtml
2.8. FILTERING

2.8.4 *blur_angular*

**Arguments:**
- amplitude[\%], center_x[\%], center_y[\%]

Apply angular blur on selected images.

**Default values:**
- ‘center_x=center_y=50\%’.

---

Example:

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

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

2.8.5 *blur_bloom*

**Arguments:**
- _amplitude>_0, _ratio>_0, _nb_iter>_0, _blend_operator_={ + | max | min }, _kernel_={ 0=quasi-gaussian (faster) | 1=gaussian | 2=box | 3=triangle | 4=quadratic }, _normalize_scales_={ 0 | 1 }, _axes_

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

When specified, argument ‘axes’ is a sequence of { x | y | z | c }.
Specifying one axis multiple times apply also the blur multiple times.
Default values:

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

Example 296: `image.jpg blur_bloom`

2.8.6 **blur_linear**

Arguments:

- `amplitude1[%], amplitude2[%], angle, boundary_conditions={ 0=dirichlet | 1=neumann }

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

Default values:

- ‘amplitude2=0’, ‘angle=0’ and ‘boundary_conditions=1’.
2.8.7 blur_radial

Arguments:

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

Apply radial blur on selected images.

Default values:

- 'center_x=center_y=50\%'.
Tutorial page:
https://gmic.eu/tutorial/_blur_radial.shtml

2.8.8  *blur_selective*

Arguments:

- `sigma>0, edges>0, nb_scales>0`

Blur selected images using selective gaussian scales.

Default values:

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

Example 298: image.jpg blur_radial 2%
2.8. FILTERING

Example 299: `image.jpg noise 20 cut 0,255 +local[-1] repeat 4 blur_selective , done endlocal`

Tutorial page:
https://gmic.eu/tutorial/_blur_selective.shtml

2.8.9  \textit{blur}_x

Arguments:

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

Blur selected images along the x-axis.

Default value:

- `\text{boundary_conditions}=1`.
2.8.10  \textit{blur\_xy}

\textbf{Arguments:}

- \texttt{amplitude\_x\%}, \texttt{amplitude\_y\%}, \texttt{boundary\_conditions\{} 0=\texttt{dirichlet}  |  1=\texttt{neumann} \texttt{\}}

Blur selected images along the X and Y axes.

\textbf{Default value:}

- \texttt{‘boundary\_conditions=1’}
2.8. FILTERING

Example 301: image.jpg +blur xy 6

Tutorial page:
https://gmic.eu/tutorial/_blur_xy.shtml

2.8.11  \textit{blur} \_\textit{xyz}

Arguments:

\begin{itemize}
  \item \texttt{amplitude\_x[\%],amplitude\_y[\%],amplitude\_z,boundary\_conditions=\{ 0=dirichlet | 1=neumann \}}
\end{itemize}

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

Default value:

\begin{itemize}
  \item \texttt{boundary\_conditions=1}'.
\end{itemize}

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

2.8.12  \textit{blur} \_\textit{y}

Arguments:

\begin{itemize}
  \item \texttt{amplitude[\%]\geq 0,boundary\_conditions=\{ 0=dirichlet | 1=neumann \}}
\end{itemize}

Blur selected images along the y-axis.
Default value:

- `'boundary_conditions=1'`.

Example 302: `image.jpg +blur_y 6`

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

2.8.13  **blur** \_z

Arguments:

- `amplitude[%] \geq 0`, `boundary_conditions={ 0=dirichlet \mid 1=neumann }`

Blur selected images along the z-axis.

Default value:

- `'boundary_conditions=1'`.

Tutorial page:
https://gmic.eu/tutorial/_blur_z.shtml

2.8.14  **boxfilter** (+)

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

- 'order' can be \{ 0=smooth | 1=1st-derivative | 2=2nd-derivative \}.
- 'boundary_conditions' can be \{ 0=dirichlet | 1=neumann \}.
When specified, argument 'axes' is a sequence of \{ x | y | z | c \}.
Specifying one axis multiple times applies also the blur multiple times.

Default values:

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

Example 303: image.jpg +boxfilter 5%
2.8.15 **bump2normal**

Convert selected bumpmaps to normalmaps.

**Example 305**: 300,300 circle 50%,50%,128,1,1 blur 5% bump2normal
2.8.16 **compose_freq**

Compose selected low and high frequency parts into new images.

Example 306: `image.jpg split_freq 2\% mirror[-1] \times compose_freq`

2.8.17 **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`

Convolve selected images by specified mask.

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

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

Default values:

- `{boundary_conditions=1’, ‘is_normalized=0’, ‘channel_mode=1’, ‘xcenter=ycenter=zcenter=-1’ (-1=centered), ‘xstart=ystart=zstart=0’, ‘xend=yend=zend=-1’ (-1=max coordinates), ‘xstride=ystride=zstride=1’ and ‘xdilation=ydilation zdilation=1’}.`
Example 307: image.jpg (0,1,0;1,-4,1;0,1,0) convolve[-2] [-1] keep[-2]

Example 308: image.jpg (0,1,0) resize[-1] 130,1,1,1,3 +convolve[0] [1]

Tutorial page:
https://gmic.eu/tutorial/_convolve.shtml
2.8. FILTERING

2.8.18  **convolve**\_\_\_\_fft

**Arguments:**

- \{\text{mask}, \text{boundary\_conditions}\}

Convolve selected images with specified mask, in the fourier domain. 'boundary\_conditions' can be \{0=dirichlet | 1=neumann | 2=periodic | 3=mirror\}.

Example 309: image.jpg 100%,100% gaussian[-1] 20,1,45 +convolve\_\_\_\_fft[0] [1]

2.8.19  **correlate** (+)

**Arguments:**

- \{\text{mask}, \text{boundary\_conditions}, \text{is\_normalized}=\{0 | 1\}, \text{channel\_mode}, \text{xcenter}, \text{ycenter}, \text{zcenter}, \text{xstart}, \text{ystart}, \text{zstart}, \text{xend}, \text{yend}, \text{zend}, \text{xstride}, \text{ystride}, \text{zstride}, \text{xdilation}, \text{ydilation}, \text{zdilation}\}

Correlate selected images by specified mask. 'boundary\_conditions' can be \{0=dirichlet | 1=neumann | 2=periodic | 3=mirror\}. 'channel\_mode' can be \{0=sum input channels | 1=one-for-one | 2=expand\}.

**Default values:**

- \{'boundary\_conditions=1', 'is\_normalized=0', 'channel\_mode=1', 'xcenter=ycenter=zcenter=-1' (-1=centered), 'xstart=ystart=zstart=0', 'xend=yend=zend=-1' (-1=max coordinates), 'xstride=ystride=zstride=1' and 'xdilation=ydilation=zdilation=1'\}. 
Example 310: image.jpg (0,1,0;1,-4,1;0,1,0) correlate[-2] [-1] keep[-2]

Example 311: image.jpg +crop 40%,40%,60%,60% correlate[0] [-1],0,1

2.8.20  cross correlation

Arguments:
2.8. FILTERING

- [mask]

Compute cross-correlation of selected images with specified mask.

Example 312: image.jpg +shift -30,-20 +cross_correlation[0] [1]

2.8.21 curvature

Compute isophote curvatures on selected images.
2.8.22 \textit{dct}

Arguments:

\begin{itemize}
  \item \( \_\{ x \mid y \mid z \}_\ldots\{ x \mid y \mid z \} \)
  \item (no arg)
\end{itemize}

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

Default values:

\begin{itemize}
  \item (no arg)
\end{itemize}
2.8. FILTERING


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

2.8.23 deblur

Arguments:

- amplitude[%]>0, nb_iter>=0, dt>0, regul>0, regul.type={ 0=Tikhonov 1=meancurv. 2=TV }

Deblur image using a regularized Jansson-Van Cittert algorithm.

Default values:

- ‘nb_iter=10’, ‘dt=20’, ‘regul=0.7’ and ‘regul.type=1’.
Example 315: `image.jpg blur 3 +deblur 3,40,20,0.01`

2.8.24  `deblur.goldmeinel`

Arguments:

- `sigma>0`, `nb_iter>0`, `acceleration>0`, `kernel_type={ 0=quasi-gaussian (faster) | 1=gaussian }`.

Deblur selected images using Gold-Meinel algorithm

Default values:

- `‘nb_iter=8’, ‘acceleration=1’ and ‘kernel_type=1’`. 
2.8. FILTERING

Example 316: image.jpg +blur 1 +deblur.goldmeinel[-1] 1

2.8.25 deblur.richardsonlucy

Arguments:

- sigma>0, nb_iter>0, kernel_type={ 0=quasi-gaussian (faster) | 1=gaussian }

Deblur selected images using Richardson-Lucy algorithm.

Default values:

- ‘nb_iter=50’ and ‘kernel_type=1’.
2.8.26  \textit{deconvolve.fft}

\textbf{Arguments:}

- \texttt{[kernel],.regularization=0}

Deconvolve selected images by specified mask in the fourier space.

\textbf{Default value:}

- \texttt{'regularization=0'}. 

Example 317: image.jpg +blur 1 +deblur.richardsonlucy[-1] 1
2.8. FILTERING

2.8.27 deinterlace

Arguments:

* _method={ 0 | 1 }

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

Default value:

* ‘method=0’.
Example 319: `image.jpg +rotate 3,1,1,50%,50% resize 100%,50% resize 100%,200%,1,3,4 shift[-1] 0,1 add +deinterlace 1`

2.8.28  *denoise (+)*

**Arguments:**

- [{guide}, std_deviation_s[\%] \geq 0, std_deviation_r[\%] \geq 0, patch_size > 0, lookup_size > 0, smoothness, fast_approx={ 0 | 1 }]
- std_deviation_s[\%] \geq 0, std_deviation_r[\%] \geq 0, patch_size > 0, lookup_size > 0, smoothness, fast_approx={ 0 | 1 }

Denoise selected images by non-local patch averaging.

**Default values:**

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

---

**CHAPTER 2. LIST OF COMMANDS**

---

**Example 319:** `image.jpg +rotate 3,1,1,50%,50% resize 100%,50% resize 100%,200%,1,3,4 shift[-1] 0,1 add +deinterlace 1`
2.8. FILTERING

2.8.29  denoise_haar

Arguments:

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

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

Default values:

- 'threshold=1.4', 'nb_scale=0' and 'cycle_spinning=10'.
2.8.30  \textit{denoise\_patchpca}

Arguments:

- \texttt{\_strength>0, \_patch\_size>0, \_lookup\_size>0, \_spatial\_sampling>0}

Denoise selected images using the patch-pca algorithm.

Default values:

- \texttt{\'patch\_size=7', \'lookup\_size=11', \'details=1.8' and \'spatial\_sampling=5\'.}
Example 322: image.jpg +noise 20 cut[-1] 0,255 +denoise_patchpca[-1] ,

2.8.31 deriche (+)

Arguments:

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

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

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

Default value:

- 'boundary_conditions=1'.
CHAPTER 2. LIST OF COMMANDS

Example 323: image.jpg deriche 3,1,x

Example 324: image.jpg +deriche 30,0,x deriche[-2] 30,0,y add

Tutorial page:
https://gmic.eu/tutorial/_deriche.shtml
2.8.32  \textit{dilate (+)}

Arguments:

- \textit{size} \geq 0
- \textit{size_{x,y,z}} \geq 0
- \textit{[kernel], boundary_conditions, is_real} = \{0=binary-mode, 1=real-mode\}

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

\textbf{Default values:}

- \textit{size_{z}=1, boundary_conditions=1 and is_real=0}.

\begin{figure}
\includegraphics[width=0.5\textwidth]{image.jpg}
\caption{Example 325: image.jpg +dilate 10}
\end{figure}

2.8.33  \textit{dilate\_circ}

Arguments:

- \textit{size} \geq 0, \textit{boundary_conditions, is_normalized} = \{0 | 1\}

Apply circular dilation of selected images by specified size.

\textbf{Default values:}

- \textit{boundary_conditions=1 and is_normalized=0}. 

\begin{figure}
\includegraphics[width=0.5\textwidth]{image_cl.jpg}
\caption{Example 325: image_cl.jpg +dilate 10}
\end{figure}
2.8.34  \textit{dilate\_oct}

Arguments:

- \texttt{size>0, boundary\_conditions, is\_normalized=\{ 0 \mid 1 \}}

Apply octagonal dilation of selected images by specified size.

Default values:

- \texttt{boundary\_conditions=1} and \texttt{is\_normalized=0}.
2.8. FILTERING

2.8.35 dilate_threshold

Arguments:

- \texttt{size_x} \geq 1, \texttt{size_y} \geq 1, \texttt{size_z} \geq 1, \texttt{threshold} \geq 0, 'boundary_conditions'

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

Default values:

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

2.8.36 divergence

Compute divergence of selected vector fields.
2.8.37  

**dog**

**Arguments:**

- \( \sigma_1 > 0 \)\% and \( \sigma_2 > 0 \)\%

Compute difference of gaussian on selected images.

**Default values:**

- \('\sigma_1 \approx 2\%' and \('\sigma_2 \approx 3\%'.\)
2.8.38 \textit{diffusion\textsc{tensors}}

Arguments:

- \texttt{\_sharpness}>0, 0<=\texttt{\_anisotropy}<=1, \texttt{\_alpha}[%], \texttt{\_sigma}[%], \texttt{\_is\_sqrt}\{ 0 | 1 \}

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

Default values:

- '\texttt{\_sharpness}=0.7', '\texttt{\_anisotropy}=0.3', '\texttt{\_alpha}=0.6', '\texttt{\_sigma}=1.1' and '\texttt{\_is\_sqrt}=0'. 
Tutorial page:
https://gmic.eu/tutorial/_diffusontensors.shtml

2.8.39  edges

Arguments:

- _threshold[%] >= 0

Estimate contours of selected images.

Default value:

- ‘edges=15%’
2.8. FILTERING

Example 331: image.jpg $+$edges 15%

2.8.40  erode (+)

Arguments:

- $size \geq 0$
- $size_x \geq 0, size_y \geq 0, size_z \geq 0$
- $[kernel], boundary\_conditions, is\_real=\{ 0=binary\_mode | 1=real\_mode \}$

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

Default values:

- 'size_z=1', 'boundary_conditions=1' and 'is_real=0'.
2.8.41  

**erode_circ**

**Arguments:**

- \texttt{size} > 0, \texttt{boundary_conditions}, \texttt{is_normalized} = \{ 0 | 1 \}

Apply circular erosion of selected images by specified size.

**Default values:**

- \texttt{boundary_conditions=1} \texttt{and \texttt{is_normalized=0}}.
2.8.42 erode\_oct

Arguments:

- \_size>0, \_boundary\_conditions, \_is\_normalized\{ 0 | 1 \}

Apply octagonal erosion of selected images by specified size.

Default values:

- ‘\_boundary\_conditions=1’ and ‘\_is\_normalized=0’.
2.8.43  \texttt{erode\_threshold}

\textbf{Arguments:}

- `size_x>=1, size_y>=1, size_z>=1, threshold>=0, boundary\_conditions`

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

\textbf{Default values:}

- 'size.y=size.x', 'size.z=1', 'threshold=255' and 'boundary\_conditions=1'.

2.8.44  \texttt{fft (+)}

\textbf{Arguments:}

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

Compute the direct fourier transform (real and imaginary parts) of selected images, optionally along the specified axes only.
Example 335: `image.jpg` luminance +fft append[-2,-1] c norm[-1] log[-1] shift[-1] 50%,50%,0,0,2

Example 336: `image.jpg` $w2=\{\text{int}(w/2)\}$ $h2=\{\text{int}(h/2)\}$ fft shift $w2,h2,0,0,2$ ellipse $w2,h2,30,30,0,1,0$ shift $-w2,-h2,0,0,2$ ifft remove[-1]

Tutorial page:
2.8.45 gradient (+)

Arguments:

- \{ x | y | z \}...\{ x | y | z \}, scheme
- (no arg)

Compute the gradient components (first derivatives) of selected images. (eq. to ‘g’).

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

(no arg) compute all significant components.

Default value:

- ‘scheme=0’.

Example 337: image.jpg gradient

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

2.8.46 gradient norm

Compute gradient norm of selected images.
2.8.47 gradient_orientation

Arguments:

- \texttt{\_dimension=\{1,2,3\}}

Compute N-d gradient orientation of selected images.

Default value:

- \texttt{\‘dimension=3\’}.
2.8.48  **guided (+)**

**Arguments:**

-  $[\text{guide}], \text{radius}[%] > 0, \text{regularization}[%] > 0$
-  $\text{radius}[%] > 0, \text{regularization}[%] > 0$

Blur selected images by guided image filtering.
If a guide image is provided, it is used to drive the smoothing process.
A guide image must be of the same xyz-size as the selected images.
This command implements the filtering algorithm described in:
He, Kaiming; Sun, Jian; Tang, Xiaoou. "Guided Image Filtering",
IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.35, no.6, pp.1397,1409, June 2013
2.8.49  *haar*

**Arguments:**

- `scale > 0`

Compute the direct haar multiscale wavelet transform of selected images.

**Tutorial page:**
https://gmic.eu/tutorial/_haar.shtml

2.8.50  *heat_flow*

**Arguments:**

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

Apply iterations of the heat flow on selected images.

**Default values:**

- ‘`nb_iter=10’, ‘dt=30’ and ‘keep_sequence=0’.’
2.8.51  \textit{hessian} (+)

Arguments:

- \{ xx | xy | xz | yy | yz | zz \}...
- \{ xx | xy | xz | yy | yz | zz \}
- (no arg)

Compute the hessian components (second derivatives) of selected images.
(no arg) compute all significant components.
2.8.52 \textit{idct}

\textbf{Arguments:}

\begin{itemize}
  \item \_\{ x | y | z \}_\ldots \{ x | y | z \}
  \item (no arg)
\end{itemize}

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

\textbf{Default values:}

\begin{itemize}
  \item (no arg)
\end{itemize}

\textbf{Tutorial page:}

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

2.8.53 \textit{iee}

Compute gradient-orthogonal-directed 2nd derivative of image(s).
2.8.54 **ifft (+)**

**Arguments:**

- \{ x \mid y \mid z \} \ldots \{ x \mid y \mid z \}

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

**Tutorial page:**

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

2.8.55 **ihaar**

**Arguments:**

- scale > 0

Compute the inverse haar multiscale wavelet transform of selected images.

2.8.56 **ilaplacian**

**Arguments:**

- \{ nb\_iterations > 0 \mid 0 \}, time\_step > 0, \{ initial\_estimate \}
Invert selected Laplacian images. If given 'nb_iterations' is '0', inversion is done in Fourier space (single iteration), otherwise, by applying 'nb_iterations' of a Laplacian-inversion PDE flow (with specified 'time_step'). Note that the resulting inversions are just estimation of possible/approximated solutions.

**Default values:**

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

Example 344: image.jpg +laplacian +ilaplacian[-1] 0

### 2.8.57 inn

Compute gradient-directed 2nd derivative of image(s).
2.8.58 \textit{inpaint (+)}

\textbf{Arguments:}

- \{\texttt{[mask]}\}
- \{\texttt{[mask]},0,\texttt{fast\_method}\}
- \{\texttt{[mask],patch\_size}=1,\texttt{lookup\_size}=1,\texttt{lookup\_factor}=0,\texttt{lookup\_increment}=0,\texttt{blend\_size}=0,0<\texttt{blend\_threshold}<1,\texttt{blend\_decay}=0,\texttt{blend\_scales}=1,\texttt{is\_blend\_outer}=\{0\mid1\}\}

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.
't\texttt{fast\_method}' can be \{0=\texttt{low\_connectivity average} | 1=\texttt{high\_connectivity average} | 2=\texttt{low\_connectivity median} | 3=\texttt{high\_connectivity median}\}.

\textbf{Default values:}

- '\texttt{patch\_size}=0', '\texttt{fast\_method}=1', '\texttt{lookup\_size}=22', '\texttt{lookup\_factor}=0.5', '\texttt{lookup\_increment}=1', '\texttt{blend\_size}=0', '\texttt{blend\_threshold}=0', '\texttt{blend\_decay}=0.05', '\texttt{blend\_scales}=10' and '\texttt{is\_blend\_outer}=1'.

Example 346: 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 347: image.jpg 100%,100% circle 30%,30%,30,1,255,0,255 circle 70%,70%,50,1,255,0,255
  +inpaint[0] [1],5,15,0.5,1,9,0 remove[1]
2.8.59  

**inpaint_pde**

Arguments:

- \( [\text{mask}], \text{nb\_scales\%}>0, \text{diffusion\_type}=\{0=\text{isotropic} | 1=\text{delaunay-guided} | 2=\text{edge-guided} | 3=\text{mask-guided}\}, \text{diffusion\_iter}>0 \)

Inpaint selected images by specified mask using a multiscale transport-diffusion algorithm. If `diffusion_type==3`, non-zero values of the mask (e.g. a distance function) are used to guide the diffusion process.

Default values:

- `‘nb\_scales=75%’, ‘diffusion\_type=1’ and ‘diffusion\_iter=20’.`

Example 348: `image.jpg 100%,100% ellipse[-1] 30%,30%,40,30,0,1,255 +inpaint\_pde[0] [1]`

2.8.60  

**inpaint_flow**

Arguments:

- \( [\text{mask}], \text{nb\_global\_iter}>0, \text{nb\_local\_iter}>0, \text{dt}>0, \text{alpha}>0, \text{sigma}>0 \)

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’.`
2.8.61  *inpaint holes*

Arguments:

- `maximal_area[%]=0, tolerance=0, is_high_connectivity={ 0 | 1 }`

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

Default values:

- `maximal_area=4`, `tolerance=0` and `is_high_connectivity=0`. 
2.8.62  \textit{inpaint\_morpho}

Arguments:

\begin{itemize}
  \item [mask]
\end{itemize}

Inpaint selected images by specified mask using morphological operators.
2.8. FILTERING

Example 351: image.jpg 100%,100% ellipse[-1] 30%,30%,40,30,0,1,255 inpaint_morpho|0|1

2.8.63 inpaint_matchpatch

Arguments:

- [mask], nb_scales={ 0=auto | >0 }, patch_size>0, nb_iterations_per_scale>0, blend_size>=0, allow_outer_blending={ 0 | 1 }, is Already Initialized={ 0 | 1 }

Inpaint selected images by specified binary mask, using a multi-scale matchpatch algorithm.

Default values:

- 'nb_scales=0', 'patch_size=9', 'nb_iterations_per_scale=10', 'blend_size=5', 'allow_outer_blending=1' and 'is Already Initialized=0'.
Example 352: image.jpg 100%,100% ellipse[-1] 30%,30%,40,30,0,1,255 +inpaint_matchpatch[0] [1]

2.8.64  **kuwahara**

Arguments:

- *size*>0

Apply Kuwahara filter of specified size on selected images.
2.8. FILTERING

Example 353: `image.jpg +kuwahara 5`

2.8.65 laplacian
Compute Laplacian of selected images.

Example 354: `image.jpg laplacian`
2.8.66 lic

Arguments:

- $\text{amplitude} > 0$, $\text{channels} > 0$

Render LIC representation of selected vector fields.

Default values:

- ‘amplitude=30’ and ‘channels=1’.

Example 355: 400,400,1,2,’if(c==0,x-w/2,y-h/2)’ +lic 200,3 quiver[-2] [-2],10,1,1,1,255

2.8.67 map tones

Arguments:

- $\text{threshold} > 0$, $\text{gamma} > 0$, $\text{smoothness} > 0$, $\text{nb_iter} > 0$

Apply tone mapping operator on selected images, based on Poisson equation.

Default values:

- ‘threshold=0.1’, ‘gamma=0.8’, ‘smoothness=0.5’ and ‘nb_iter=30’.
2.8.68 *map_tones_fast*

**Arguments:**

- \(_{radius} [%] \geq 0, \_{power} = 0\)

Apply fast tone mapping operator on selected images.

**Default values:**

- ‘radius=3%’ and ‘power=0.3’. 
2.8.69 meancurvature_flow

Arguments:

- \_nb\_iter\_\geq 0, \_dt, \_keep\_sequence=\{ 0 \mid 1 \}

Apply iterations of the mean curvature flow on selected images.

Default values:

- ‘\_nb\_iter=10’, ‘\_dt=30’ and ‘\_keep\_sequence=0’.
2.8.70  median (+)

Arguments:

- \textit{size} \geq 0, \textit{threshold} > 0

Apply (opt. thresholded) median filter on selected images with structuring element size \(x\) x \(x\).
2.8.71  \textit{nlmeans}

Arguments:

- \{guide\}, \texttt{patch\_radius}>0, \texttt{spatial\_bandwidth}>0, \texttt{tonal\_bandwidth}>0, \texttt{patch\_measure\_command}
- \texttt{patch\_radius}>0, \texttt{spatial\_bandwidth}>0, \texttt{tonal\_bandwidth}>0, \texttt{patch\_measure\_command}

Apply non local means denoising of Buades et al, 2005. on selected images. The patch is a gaussian function of \texttt{std\_patch\_radius}'. The spatial kernel is a rectangle of radius \texttt{spatial\_bandwidth}'. The tonal kernel is exponential (exp(-d^2/\texttt{tonal\_bandwidth}^2)) with d the euclidian distance between image patches.

Default values:

- \texttt{patch\_radius=4}', \texttt{spatial\_bandwidth=4}', \texttt{tonal\_bandwidth=10}' \texttt{and} \texttt{patch\_measure\_command=--norm}'.

Example 359: image.jpg +median 5
2.8.72  *nlmeans.core*

**Arguments:**

- *reference_image, scaling_map, patch_radius>*0, *spatial_bandwidth>*0

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

2.8.73  *normalize_local*

**Arguments:**

- *amplitude>*0, *radius>*0, *n_smooth>*0[, ]*, *a_smooth>*0[, ]*, *is_cut>*0 | 1*, *min=0*, *max=255

Normalize selected images locally.

**Default values:**

- *amplitude=3*, *radius=16*, *n_smooth=4%,* *a_smooth=2%,* *is_cut=1*, *min=0* and *max=255*. 
2.8.74  normalized_cross_correlation

Arguments:

- \([\text{mask}]\)

Compute normalized cross-correlation of selected images with specified mask.
2.8. FILTERING

Example 362: image.jpg +shift -30,-20 +normalized_cross_correlation[0] [1]

2.8.75 percentile

Arguments:

- \([\text{mask}, 0 \leq \text{min}_\text{percentile} \% \leq 100, 0 \leq \text{max}_\text{percentile} \% \leq 100]\).

Apply percentile averaging filter to selected images.

Default values:

- 'min_percentile=0' and 'max_percentile=100'.
Example 363: `image.jpg shape circle 11,11 +percentile[0] [1],25,75`

2.8.76 `peronamalik_flow`

Arguments:

- `K_factor>0`, `nb_iter>=0`, `dt`, `keep_sequence={ 0 | 1 }`

Apply iterations of the Perona-Malik flow on selected images.

Default values:

- ‘`K_factor=20’`, ‘`nb_iter=5’’, ‘`dt=5’' and ‘`keep_sequence=0’'.

2.8.77 *phase_correlation*

**Arguments:**

- `[destination]`

Estimate translation vector between selected source images and specified destination.
2.8.78  \textit{pde\_flow}

Arguments:

- \_\text{nb\_iter}\geq0, \text{dt}, \text{velocity\_command}, \text{keep\_sequence} = \{0 \mid 1\}

Apply iterations of a generic PDE flow on selected images.

Default values:

- ‘\text{nb\_iter}=10’, ‘\text{dt}=30’, ‘\text{velocity\_command}=\text{laplacian}’ and ‘\text{keep\_sequence}=0’. 
2.8.79 *periodize-poission*

Periodize selected images using a Poisson solver in Fourier space.

Example 367: `image.jpg +periodize-poission array 2,2,2`
2.8.80 rbf

Arguments:

- \( dx, x_0, x_1, \phi(r) \)
- \( dx, dy, x_0, y_0, x_1, y_1, \phi(r) \)
- \( dx, dy, dz, x_0, y_0, z_0, x_1, y_1, z_1, \phi(r) \)

Reconstruct 1D/2D or 3D image from selected sets of keypoints, by RBF-interpolation. A set of keypoints is represented by a vector-valued image, where each pixel represents a single keypoint. Vector components of a keypoint have the following meaning: - For 1D reconstruction: \([ x_k, f_1(k), ..., f_N(k) ]\). - For 2D reconstruction: \([ x_k, y_k, f_1(k), ..., f_N(k) ]\). - For 3D reconstruction: \([ x_k, y_k, z_k, f_1(k), ..., f_N(k) ]\). Values \( x_k, y_k, z_k \) are the spatial coordinates of keypoint \( k \). Values \( f_1(k), ..., f_N(k) \) are the \( N \) components of the vector value of keypoint \( k \).

The command reconstructs an image with specified size \( dx \times dy \times dz \), with \( N \) channels.

Default values:

- \( x_0 = y_0 = z_0 = 0 \), \( x_1 = dx - 1 \), \( y_1 = dy - 1 \), \( z_1 = dz - 1 \), \( \phi(r) = r^2 \log(1e-5 + r) \).

Example 368: sp colorful r2dx 400 100%,100% noise_poissondisk. 10 l,[is],l,5 eval[-2] "begin(p=0);i?l[t-1,p++]=[x,y,I(0)]" to_rgb[l] mul[0,1] dilate_circ[0] 5 +rbf[-1] {0,[w,h]} c[-1] 0,255
2.8. Filtering

Example 369: 32, 1, 1, 5, u([400, 400, 255, 255]) rbf 400, 400 c 0, 255

2.8.81 red_eye

Arguments:

- $0 < \text{threshold} \leq 100$, $\text{smoothness} = 0$, $0 \leq \text{attenuation} \leq 1$

Attenuate red-eye effect in selected images.

Default values:

- ‘threshold=75’, ‘smoothness=3.5’ and ‘attenuation=0.1’.
2.8.82  *remove_hotpixels*

**Arguments:**

- \( \text{mask\_size} > 0, \text{threshold}[] > 0 \)

Remove hot pixels in selected images.

**Default values:**

- \('\text{mask\_size}=3\)' and \('\text{threshold}=10\%\).
2.8.83  *remove_pixels*

Arguments:

- \texttt{number\_of\_pixels} \geq 0

Remove specified number of pixels (i.e. set them to 0) from the set of non-zero pixels in selected images.


2.8.84 **rolling guidance**

**Arguments:**

- $std\_deviation\_s(%)>0$, $std\_deviation\_r(%)>0$, $precision>0$

Apply the rolling guidance filter on selected image.
Rolling guidance filter is a fast image abstraction filter, described in: "Rolling Guidance Filter”, Qi Zhang Xiaoyong, Shen Li, Xu Jiaya Jia, ECCV’2014.

**Default values:**

- ‘$std\_deviation\_s=4$’, ‘$std\_deviation\_r=10$’ and ‘$precision=0.5$’.
2.8. FILTERING

Example 373: image.jpg +rolling, guidance, +-

2.8.85 sharpen (+)

Arguments:

- \texttt{amplitude}>0
- \texttt{amplitude}=0, \texttt{edge}=0, \texttt{alpha}, \texttt{sigma}

Sharpen selected images by inverse diffusion or shock filters methods. \texttt{'edge'} must be specified to enable shock-filter method.

Default values:

- \texttt{'alpha=0'} and \texttt{'sigma=0'}. 
2.8.86 \textit{smooth} (+)

\textbf{Arguments:}
2.8. FILTERING

- amplitude[%]>=0, _sharpness>=0, _anisotropy<=1, _alpha[%, _sigma[%, _dl]>0, _da>0, _precision>0, _interpolation, _fast_approx={ 0 | 1 }

- nb_iterations>0, _sharpness>=0, _anisotropy, _alpha, _sigma, _dt>0, 0

- [tensor_field], _amplitude>=0, _dl>0, _da>0, _precision>0, _interpolation, _fast_approx={ 0 | 1 }

- [tensor_field], _nb_iterations>0, _dt>0, 0

Smooth selected images anisotropically using diffusion PDE’s, with specified field of diffusion tensors. ’interpolation’ can be { 0=nearest | 1=linear | 2=runge-kutta }.

**Default values:**

- ‘sharpness=0.7’, ‘anisotropy=0.3’, ‘alpha=0.6’, ‘sigma=1.1’, ‘dl=0.8’, ‘da=30’, ‘precision=2’, ‘interpolation=0’ and ‘fast_approx=1’.

Example 376: image.jpg repeat 3 smooth 40,0,1,1,2 done
Example 377: `image.jpg 100%,100%,1,2 rand[-1] -100,100 repeat 2 smooth[-1] 100,0.2,1,4,4
done warp[0] [-1],1,1`

Tutorial page:
https://gmic.eu/tutorial/_smooth.shtml

2.8.87 split_freq

Arguments:

- `smoothness>0[%]`

Split selected images into low and high frequency parts.
2.8. Filtering

Example 378: `image.jpg` split_freq 2%

2.8.88 `solve_poisson`

Arguments:

- "laplacian_command", `nb_iterations>`0, `time_step>`0, `nb_scales>`0

Solve Poisson equation so that applying `laplacian[n]` is close to the result of `laplacian_command[n]`. Solving is performed using a multi-scale gradient descent algorithm. If `nb_scales=0`, the number of scales is automatically determined.

Default values:

- `nb_iterations=60`, `dt=5` and `nb_scales=0`. 
Example 379: image.jpg command "foo : gradient x" +solve_poisson foo +foo[0] +laplacian[1]

2.8.89 \textit{split\_details}

Arguments:

- \_nb\_scales\_>0, base\_scale[\%]\_>0, detail\_scale[\%]\_>0

Split selected images into \textquoteleft nb\_scales\textquoteright detail scales. If \textquoteleft base\_scale\textquoteright =\textquoteleft detail\_scale\textquoteright =0, the image decomposition is done with \textquoteleft a trous\textquoteright wavelets. Otherwise, it uses laplacian pyramids with linear standard deviations.

Default values:

- \textquoteleft nb\_scales=4\textquoteright, \textquoteleft base\_scale=0\textquoteright and \textquoteleft detail\_scale=0\textquoteright.
2.8.90 \textbf{structuretensors (+)}

Arguments:

- \_scheme\{ 0=\text{centered} | 1=\text{forward/backward} \}

Compute the structure tensor field of selected images.

Default value:

- ‘scheme=1’. 

Example 381: image.jpg structuretensors abs pow 0.2

Tutorial page:
https://gmic.eu/tutorial/_structuretensors.shtml

2.8.91 solidify

Arguments:

- \_smoothness\[%\]>=0, \_diffusion\_type={0=\textit{isotropic} | 1=\textit{delaunay-oriented} | 2=\textit{edge-oriented}}, \_diffusion\_iter>=0

Solidify selected transparent images.

Default values:

- ‘smoothness=75%’, ‘diffusion\_type=1’ and ‘diffusion\_iter=20’.
2.8. FILTERING

Example 382: image.jpg 100%,100% circle[-1] 50%,50%,25%,1,255 append c +solidify, display rgba

2.8.92 syntexturize

Arguments:

- width[%] > 0, height[%] > 0

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: http://www.ipol.im/pub/art/2011/ggm_rpn/

Default values:

- 'width=height=100%'.

Example 383: `image.jpg` crop 2,282,50,328 +syntextrizer 320,320

2.8.93 syntextrizer matchpatch

**Arguments:**

- `width[%]>0`, `height[%]>0`, `nb_scales>=0`, `patch_size>0`, `blending_size>0`, `precision>=0`

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 384: image.jpg crop 25%,25%,75%,75% syntexturize matchpatch 512,512

2.8.94 *tv flow*

Arguments:

- \( \text{nb_iter} \geq 0, \ dt, \ \text{keep_sequence} = \{ 0 \ | \ 1 \} \)

Apply iterations of the total variation flow on selected images.

Default values:

- ‘\( \text{nb_iter}=10 \)’, ‘\( \text{dt}=30 \)’ and ‘\( \text{keep_sequence}=0 \)’.
2.8.95  **unsharp**

Arguments:

- \( \text{radius[\%]} > 0, \text{amount} > 0, \text{threshold[\%]} > 0 \)

Apply unsharp mask on selected images.

**Default values:**

- ‘\text{amount}=2’ and ‘\text{threshold}=0’.
2.8. FILTERING

Example 386: `image.jpg blur 3 +unsharp 1.5,15 cut 0,255`

### 2.8.96 unsharp_octave

**Arguments:**

- \( _{nb \_ scales}>0, _{radius[\%]}>=0, _{amount}>0, _{threshold[\%]}>=0 \)

Apply octave sharpening on selected images.

**Default values:**

- `nb_scales=4`, `radius=1`, `amount=2` and `threshold=0`. 
Example 387: `image.jpg blur 3 +unsharp_octave 4,5,15 cut 0,255`

2.8.97 *vanvliet* (+)

**Arguments:**

- `std_deviation>=0[%], order={0 | 1 | 2 | 3}, axis={x | y | z | c}, boundary_conditions`

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

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

**Default value:**

- `'boundary_conditions=1'`.
2.8. FILTERING

Example 388: image.jpg +vanvliet 3,1,x

Example 389: image.jpg +vanvliet 30,0,x vanvliet[-2] 30,0,y add

2.8.98 voronoi

Compute the discrete Voronoi diagram of non-zero pixels in selected images.
Example 390: 400,400 noise 0.2,2 eq 1 +label_fg 0 voronoi[-1] +gradient[-1] xy,1 append[-2,-1] c norm[-1] ==[-1] 0 map[-2] 2,2 mul[-2,-1] normalize[-2] 0,255 dilate_circ[-2] 4 reverse max

2.8.99 watermark fourier

Arguments:

- text, size>0

Add a textual watermark in the frequency domain of selected images.

Default value:

- ‘size=33’. 
2.8. FILTERING

Example 391: image.jpg +watermark_fourier "Watermarked!" +display.fft remove[-3,-1] normalize 0,255 append[-4,-2] y append[-2,-1] y

2.8.100  watershed (+)

Arguments:

- \([\text{priority}_\text{image}], \text{is\_high\_connectivity} = \{ 0 \mid 1 \}\)

Compute the watershed transform of selected images.

Default value:

- ‘\text{is\_high\_connectivity}=1’.
Example 392: 400,400 noise 0.2,2 eq 1 +distance 1 mul[-1] -1 label[-2] watershed[-2] [-1] mod[-2] 256 map[-2] 0 reverse

2.9 Features Extraction

2.9.1 area

Arguments:

- tolerance\geq0, is\_high\_connectivity=\{0 \mid 1\}

Compute area of connected components in selected images.

Default values:

- ‘is\_high\_connectivity=0’.
2.9. FEATURES EXTRACTION

Example 393: image.jpg luminance stencil[-1] 1 +area 0

Tutorial page: https://gmic.eu/tutorial/_area.shtml

2.9.2 area_fg

Arguments:

- \texttt{tolerance} \geq 0, is\_high\_connectivity = \{0 \mid 1\}

Compute area of connected components for non-zero values in selected images. Similar to 'area' except that 0-valued pixels are not considered.

Default values:

- 'is\_high\_connectivity=0'.
 Example 394: \texttt{image.jpg} luminance stencil[-1] 1 +area fg 0

2.9.3 \textit{at line}

Arguments:

- \(x0\%,y0\%,z0\%,x1\%,y1\%,z1\%\)

Retrieve pixels of the selected images belonging to the specified line \((x0,y0,z0)-(x1,y1,z1)\).
2.9.4 at_quadrangle

Arguments:

- \( x_0[\%], y_0[\%], x_1[\%], y_1[\%], x_2[\%], y_2[\%], x_3[\%], y_3[\%], \_interpolation, \_boundary\_conditions \)
- \( x_0[\%], y_0[\%], z_0[\%], x_1[\%], y_1[\%], z_1[\%], x_2[\%], y_2[\%], z_2[\%], x_3[\%], y_3[\%], z_3[\%], \_interpolation, \_boundary\_conditions \)

Retrieve pixels of the selected images belonging to the specified 2D or 3D quadrangle. 'interpolation' can be \{ 0=nearest-neighbor | 1=linear | 2=cubic \}. 'boundary_conditions' can be \{ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror \}.

Example 396: image.jpg params=5\%,5\%,95\%,5\%,60\%,95\%,40\%,95\% +at_quadrangle $params$ polygon.. 4,$params,0.5,255

2.9.5 barycenter

Compute the barycenter vector of pixel values.
2.9.6 **delaunay**

Generate discrete 2D Delaunay triangulation of non-zero pixels in selected images. Input images must be scalar.

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

Example 397: 256,256 ellipse 50%,50%,20%,20%,0,1,1 deform 20 +barycenter +ellipse[-2] 
{(80,1),5,5,0,10}
2.9. FEATURES EXTRACTION

Example 398: 400,400 rand 32,255 100%,100% noise. 0.4,2 eq. 1 mul +delaunay

Example 399: image.jpg b 1% 100%,100% noise. 0.8,2 eq. 1 mul +delaunay channels 0,2

2.9.7 *detect_skin*

Arguments:

- \(0 \leq \text{tolerance} \leq 1, \text{skin}_x, \text{skin}_y, \text{skin}_\text{radius} \geq 0\)

Detect skin in selected color images and output an appartenance probability map. Detection is performed using CbCr chromaticity data of skin pixels. If arguments 'skin_x', 'skin_y' and 'skin_radius' are provided, skin pixels are learnt from the sample pixels inside the circle located at ('skin_x','skin_y') with radius 'skin_radius'.

Default value:

- 'tolerance=0.5' and 'skin_x=skin_y=skin_radius=-1'.

2.9.8 *displacement (+)*

Arguments:

- \{\text{source}_\text{image}, \text{smoothness}, \text{precision}\geq 0, \text{nb}_\text{scales}\geq 0, \text{iteration}_\text{max}\geq 0, \text{is}_\text{backward}\{ 0 | 1 \}\}, \{\text{guide}\}

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 400: image.jpg +rotate 3,1,0,50%,50% +displacement[-1] [-2] quiver[-1] [-1],15,1,1,1,[1.5+1M]

2.9.9 distance (+)

Arguments:

- iso_value[%,metric]
- iso_value[%,metric],method

Compute the unsigned distance function to specified isovalues, opt. according to a custom metric. ‘metric’ can be {0=chebyshev | 1=manhattan | 2=euclidean | 3=squared-euclidean}. ‘method’ can be {0=fast-marching | 1=low-connectivity dijkstra | 2=high-connectivity dijkstra | 3=1+return path | 4=2+return path}.

Default value:

- ‘metric=2’ and ‘method=0’.
2.9. FEATURES EXTRACTION

Example 401: `image.jpg` threshold 20% distance 0 pow 0.3

Example 402: 400,400 set 1,50%,50% +distance[0] 1,2 +distance[0] 1,1 distance[0] 1,0 mod 32 threshold 16 append c

Tutorial page:
https://gmic.eu/tutorial/_distance.shtml
2.9.10  \textit{fftpolar}

Compute fourier transform of selected images, as centered magnitude/phase images.

\textbf{Example 403:} \texttt{image.jpg fftpolar ellipse 50\%,50\%,10,10,0,1,0} \texttt{ifftpolar}

2.9.11  \textit{histogram} (+)

**Arguments:**

- \_\_\_nb\_levels\_\_\_>0\%\,,\_\_\_value0\_\_\_\%,\_\_\_value1\_\_\_\%

Compute the histogram of selected images.

If value range is set, the histogram is estimated only for pixels in the specified value range. Argument `value1` must be specified if `value0` is set.

**Default values:**

- `\_\_\_nb\_levels\_\_\_=256\,\,` `\_\_\_value0\_\_\_=0\%\,` \texttt{and} `\_\_\_value1\_\_\_=100\%`
2.9. FEATURES EXTRACTION

Example 404: image.jpg +histogram 64 display_graph[-1] 400,300,3

2.9.12  histogram Nd

Arguments:

• nb_levels>0[%,value0[%,value1[%]

Compute the 1D,2D or 3D histogram of selected multi-channels images (having 1, 2 or 3 channels). If value range is set, the histogram is estimated only for pixels in the specified value range.

Default values:

• ‘value0=0%’ and ‘value1=100%’.
2.9.13  histogram\_cumul

Arguments:

- _nb\_levels>0, _is\_normalized={ 0 | 1 }, _val0[\%], _val1[\%]

Compute cumulative histogram of selected images.

Default values:

- ‘nb\_levels=256’, ‘is\_normalized=0’, ‘val0=0\%’ and ‘val1=100\%’.
2.9. FEATURES EXTRACTION

Example 406: image.jpg +histogram_cumul 256 histogram[0] 256 display_graph 400,300,3

2.9.14  **histogram_pointwise**

Arguments:

- nb_levels>0, value0, value1

Compute the histogram of each vector-valued point of selected images. If value range is set, the histogram is estimated only for values in the specified value range.

Default values:

- ‘value0=0%’ and ‘value1=100%’.

2.9.15  **hough**

Arguments:

- width>0, height>0, gradient_norm_voting={ 0 | 1 }

Compute hough transform (theta,rho) of selected images.

Default values:

- ‘width=512’, ‘height=width’ and ‘gradient_norm_voting=1’.
2.9.16  *ifftpolar*

Compute inverse fourier transform of selected images, from centered magnitude/phase images.

2.9.17  *isophotes*

Arguments:

- `_nb_levels>0`

Render isophotes of selected images on a transparent background.

Default value:

- `‘nb_levels=64’`
2.9. FEATURES EXTRACTION

Example 408: image.jpg blur 2 isophotes 6 dilate_circ 5 display_rgba

2.9.18 label (+)

Arguments:

- _tolerance>_0, _is_high_connectivity_={ 0 | 1 }, _is_L2_norm_={ 0 | 1 }

Label connected components in selected images.

Default values:

- ‘tolerance=0’, ‘is_high_connectivity=0’ and ‘is_L2_norm=1’.
Example 409: `image.jpg` luminance threshold 60% label normalize 0,255 map 0

Example 410: `400,400` set 1,50%,50% distance 1 mod 16 threshold 8 label mod 255 map 2

Tutorial page:
https://gmic.eu/tutorial/_label.shtml
2.9. FEATURES EXTRACTION

2.9.19  \textit{label\_fg}

\textbf{Arguments:}

- \textit{tolerance}>0, \textit{is\_high\_connectivity}=\{ 0 | 1 \}

Label connected components for non-zero values (foreground) in selected images. Similar to ‘label’ except that 0-valued pixels are not labeled.

\textbf{Default value:}

- ‘\textit{is\_high\_connectivity}=0’.

2.9.20  \textit{laar}

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.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{image}
\caption{Example 411: shape cupid 256 coords=$\{-\text{laar}\}$ normalize 0,255 to rgb rectangle $\text{coords},0.5,0,128,0$}
\end{figure}

2.9.21  \textit{max\_patch}

\textbf{Arguments:}

- \textit{patch\_size}>1

Return locations of maximal values in local patch-based neighborhood of given size for selected images.

\textbf{Default value:}

- ‘\textit{patch\_size}=16’.
2.9.22  

**min patch**

Arguments:

- \( \text{patch.size} \geq 1 \)

Return locations of minimal values in local patch-based neighborhood of given size for selected images.

Default value:

- ‘patch.size=16’.
2.9.23 minimal_path

Arguments:

- \( x_0 \), \( y_0 \), \( z_0 \), \( x_1 \), \( y_1 \), \( z_1 \), \( \text{is.high.connectivity} \) = \{ 0 \mid 1 \}

Compute minimal path between two points on selected potential maps.

Default value:

- ‘is.high.connectivity=0’.
CHAPTER 2. LIST OF COMMANDS

Example 414: 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

2.9.24 mse (+)
Compute MSE (Mean-Squared Error) matrix between selected images.

Example 415: image.jpg +noise 30 +noise[0] 35 +noise[0] 38 cut. 0,255 mse
2.9. FEATURES EXTRACTION

2.9.25 patches

Arguments:

- `patch_width>0, patch_height>0, patch_depth>0, x0, y0, z0, x1, y1, z1, ..., xN, yN, zN`

Extract N+1 patches from selected images, centered at specified locations.

Example:

```
image.jpg +patches 64,64,1,153,124,0,184,240,0,217,126,0,275,38,0
```

2.9.26 matchpatch (+)

Arguments:

- `[patch_image], patch_width=1, patch_height=1, patch_depth=1, nb_iterations=0, nb_randoms=0, patch_penalization, output_score={0 | 1}`

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 417: image.jpg sample ? to_rgb +matchpatch[0] [1],3 +warp[-2] [-1],0

2.9.27 plot2value

Retrieve values from selected 2D graph plots.
Example 418: 400,300,1,1,’if(y>300*abs(cos(x/10+2*u)),1,0)’ +plot2value +display_graph([-1] 400,300

2.9.28 pointcloud

Arguments:

• `type` = { 0=X=X-opacity | 0=binary | 1=cumulative | 2=label |
  3=retrieve coordinates }, width, height>0, depth>0

Render a set of point coordinates, as a point cloud in a 1D/2D or 3D binary image
(or do the reverse, i.e. retrieve coordinates of non-zero points from a rendered point cloud).
Input point coordinates can be a 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’.
Example 419: 3000,2 rand 0,400 +pointcloud 0 dilate[-1] 3

Example 420: 3000,2 rand 0,400 {w} {w},3 rand[-1] 0,255 append y +pointcloud 0 dilate[-1] 3

2.9.29 psnr

Arguments:

- _max_value

Compute PSNR (Peak Signal-to-Noise Ratio) matrix between selected images.

Default value:
2.9. FEATURES EXTRACTION

- ‘max_value=255’.

Example 421: image.jpg +noise 30 +noise[0] 35 +noise[0] 38 cut[-1] 0,255 psnr 255 replace inf

2.9.30 segment_watershed

Arguments:

- _threshold>=0

Apply watershed segmentation on selected images.

Default values:

- ‘threshold=2’.
2.9.31 \textit{shape2bump}

Arguments:

\begin{itemize}
\item \texttt{resolution}=0, 0<=\texttt{weight\_avg}\texttt{max\_avg}\leq1, \texttt{dilation}, \texttt{smoothness}=0
\end{itemize}

Estimate bumpmap from binary shape in selected images.

Default value:

\begin{itemize}
\item ‘\texttt{resolution}=256’, ‘\texttt{weight\_avg\_max}=0.75’, ‘\texttt{dilation}=0’ and ‘\texttt{smoothness}=100’.
\end{itemize}

2.9.32 \textit{skeleton}

Arguments:

\begin{itemize}
\item \texttt{boundary\_conditions}={ 0=dirichlet | 1=neumann }
\end{itemize}

Compute skeleton of binary shapes using distance transform and constrained thinning.

Default value:

\begin{itemize}
\item ‘\texttt{boundary\_conditions}=1’.
\end{itemize}
2.9. FEATURES EXTRACTION

Example 423: shape_cupid 320 +skeleton 0

2.9.33 slic

Arguments:

- size>0, regularity>=0, nb_iterations>0

Segment selected 2D images with superpixels, using the SLIC algorithm (Simple Linear Iterative Clustering). Scalar images of increasingly labeled pixels are returned.

Default values:

- ‘size=16’, ‘regularity=10’ and ‘nb_iterations=10’.
Example 424: image.jpg +srq2lab slic[-1] 16 +blend shapeaverage f[-2] *j(1,0)==i && j(0,1)==i* *[-1] [-2]

2.9.34  **ssd_patch**

**Arguments:**

* [patch], use_fourier={ 0 | 1 }, boundary_conditions={ 0=dirichlet | 1=neumann }

Compute fields of SSD between selected images and specified patch. Argument 'boundary_conditions' is valid only when 'use_fourier=0'.

**Default value:**

* 'use_fourier=0' and 'boundary_conditions=0'.
2.9. FEATURES EXTRACTION

2.9.35  thinning

Arguments:

- _boundary_conditions={ 0=dirichlet | 1=neumann } 

Compute skeleton of binary shapes using morphological thinning
(beware, this is a quite slow iterative process)

Default value:

- ‘boundary_conditions=1’. 
2.9.36 tones

Arguments:

- \( N > 0 \)

Get \( N \) tones masks from selected images.
2.9.37  topographic map

Arguments:

- \( _{\text{nb\_levels}} > 0, _{\text{smoothness}} \)

Render selected images as topographic maps.

Default values:

- ‘\( _{\text{nb\_levels}} = 16 \)’ and ‘\( _{\text{smoothness}} = 2 \)’.

Example 428: `image.jpg topographic_map 10`

2.9.38  tsp

Arguments:

- \( _{\text{precision}} \geq 0 \)

Try to solve the 'travelling salesman' problem, using a combination of greedy search and 2-opt algorithms. Selected images must have dimensions \( N \times 1 \times 1 \times C \) to represent \( N \) cities each with \( C \)-dimensional coordinates. This command re-order the selected data along the x-axis so that the point sequence becomes a shortest path.

Default values:

- ‘\( _{\text{precision}} = 256 \)’.
Example 429: 256,1,1,2 \ random 0,512 tsp \ 512,512,1,3 \ repeat w0 \ circle[-1]
{0,1[$>\},2,1,255,255,255 \ line[-1] \ {0,boundary=2;[I[$>\,$I[$>]+1]\},1,255,128,0 \ done
keep[-1]

2.9.39 \ variance_patch

Arguments:

- \_patch_size>=1

Compute variance of each images patch centered at (x,y), in selected images.

Default value:

- \‘patch_size=16’
2.10 Image Drawing

2.10.1 arrow

Arguments:

- \( x_0 \%, y_0 \%, x_1 \%, y_1 \%, \text{thickness} \% \geq 0, \text{head length} \% \geq 0, \text{-head thickness} \% \geq 0, \text{opacity}, \text{pattern}, \text{color1}, \ldots \)

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 431: 400,400,1,3 repeat 100 arrow 50%,50%,{u(100)}%,{u(100)}%,3,20,10,0.3,0{-RGB}
done

2.10.2 axes

Arguments:

- \(x_0, x_1, y_0, y_1, \text{font height}=0, \text{opacity}, \text{pattern}, \text{color}_1, \ldots\)

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'.

2.10.3 \textit{ball}

**Arguments:**

- \texttt{size} $> 0$, \texttt{R}, \texttt{G}, \texttt{B}, \texttt{specular\_light} $\leq 8$, \texttt{specular\_size} $\leq 8$, \texttt{shadow} $= 0$

Input a 2D RGBA colored ball sprite.

**Default values:**

- \texttt{\textasciitilde size=64}, \texttt{\textasciitilde R=255}, \texttt{\textasciitilde G=R}, \texttt{\textasciitilde B=R}, \texttt{\textasciitilde specular\_light=0.8}, \texttt{\textasciitilde specular\_size=1} and \texttt{\textasciitilde shading=1.5}.
2.10.4 chessboard

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

Draw chessboard on selected images.

Default values:
- 'size2=size1', 'offset1=offset2=0', 'angle=0', 'opacity=1', 'color1=0' and 'color2=255'.
Example 434: image.jpg chessboard 32,32,0,0,25,0.3,255,128,0,0,128,255

2.10.5  **cie1931**

Draw CIE-1931 chromaticity diagram on selected images.

![CIE-1931 Chromaticity Diagram](image.jpg)

Example 435: 500,400,1,3 cie1931

2.10.6  **circle**

**Arguments:**

- \(x\%\), \(y\%\), \(R\%\), \(\text{opacity}\), \(\text{pattern}\), \(\text{color1}\), ...

Draw specified colored circle on selected images.

A radius of ’100%’ stands for ’\(\sqrt{\text{width}^2+\text{height}^2}\)’.

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

**Default values:**

- ’\text{opacity}=1\’, ’\text{pattern}=(\text{undefined})\’ and ’\text{color1}=0\’. 
Example 436: image.jpg repeat 300 circle \{u(100)\}%\{u(100)\}%\{u(30)\},0.3,0\{-RGB\} done circle 50\%,50\%,100,0.7,255

2.10.7 \textit{close\_binary}

Arguments:

- \(0\leq endpoint\_rate\leq 100\), \(endpoint\_connectivity\geq 0\), \(spline\_distmax\geq 0\), \(segment\_distmax\geq 0\), \(0\leq spline\_anglemax\leq 180\), \(spline\_roundness\geq 0\), \(area\_min\geq 0\), \(allow\_self\_intersection\in\{0,1\}\)

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

Default values:


2.10.8 \textit{ellipse (+)}

Arguments:

- \(x\%\),\(y\%\),\(R\%\),\(r\%\),\(angle\),\(opacity\),\(pattern\),\(color1\),...

Draw specified colored ellipse on selected images. A radius of ‘100\%’ stands for \(\sqrt{\text{width}^2+\text{height}^2}\). ‘pattern’ is an hexadecimal number starting with ‘0x’ which can be omitted even if a color is specified. If a pattern is specified, the ellipse is drawn outlined instead of filled.
2.10. IMAGE DRAWING

Default values:

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

Example 437: image.jpg repeat 300 ellipse 
{u(100)}%,{u(100)}%,{u(30)}%,{u(30)}%,{u(180)}%,0.3,${-RGB}$ done ellipse 50%,50%,100,100,0,0.7,255

2.10.9 flood (+)

Arguments:

- x[%], y[%], z[%], tolerance>=0, is_high_connectivity={ 0 | 1 }, opacity, color1,...

Flood-fill selected images using specified value and tolerance.

Default values:

- ‘y=z=0’, ‘tolerance=0’, ‘is_high_connectivity=0’, ‘opacity=1’ and ‘color1=0’.
2.10.10  \textit{gaussian}

\textbf{Arguments:}

\begin{itemize}
  \item \texttt{\_sigma1[\%], \_sigma2[\%], \_angle}
\end{itemize}

Draw a centered gaussian on selected images, with specified standard deviations and orientation.

\textbf{Default values:}

\begin{itemize}
  \item ‘\texttt{\_sigma1=3’}, ‘\texttt{\_sigma2=\_sigma1’ and ‘\_angle=0’}.
\end{itemize}
2.10. IMAGE DRAWING

Example 439: 400,400 gaussian 100,30,45

Tutorial page:
https://gmic.eu/tutorial/_gaussian.shtml

2.10.11 graph (+)

Arguments:

- \([\text{function}\_\text{image}], \text{plot}\_\text{type}, \text{vertex}\_\text{type}, \text{ymin}, \text{ymax}, \text{opacity}, \text{pattern}, \text{color1}, \ldots\)
- `'formula', \text{resolution}>0, \text{plot}\_\text{type}, \text{vertex}\_\text{type}, \text{xmin}, \text{xmax}, \text{ymin}, \text{ymax}, \text{opacity}, \text{pattern}, \text{color1}, \ldots`

Draw specified function graph on selected images.

- `'plot\_type'` can be {0=none | 1=lines | 2=splines | 3=bar}.
- `'vertex\_type'` can be {0=none | 1=points | 2,3=crosses | 4,5=circles | 6,7=squares}.
- `'pattern'` is an hexadecimal number starting with '0x' which can be omitted even if a color is specified.

Default values:

- `'plot\_type=1', 'vertex\_type=1', 'ymin=ymax=0 (auto)', 'opacity=1', 'pattern=(undefined)'`

and `'color1=0'`.
Example 440: 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],2,0,0,1,0,255,0 graph[0] [3],2,0,0,1,0,0,255 keep[0]

2.10.12 grid

Arguments:

- `size_x[%]>=0, size_y[%]>=0, offset_x[%], offset_y[%], opacity, pattern, color1, ...

Draw xy-grid on selected images.
'pattern' is an hexadecimal number starting with '0x' which can be omitted even if a color is specified.

Default values:

- ‘offset_x=offset_y=0’, ‘opacity=1’, ‘pattern=(undefined)’ and ‘color1=0’.
2.10. IMAGE DRAWING

Example 441: `image.jpg` grid 10%,10%,0,0,0,0.5,255

Example 442: 400,400,1,3,255 grid 10%,10%,0,0,0,0.3,0xCCCCCCCC,128,32,16

2.10.13  `image (+)`

Arguments:
CHAPTER 2. LIST OF COMMANDS

- \{\text{sprite}, x\% | \}, y\% | \}, z\% | \}, c\% | \}, \text{opacity}, \text{opacity\_mask}, \text{max\_opacity\_mask}\}

Draw specified sprite image on selected images.
(eq. to 'j').
If one of the x,y,z or c argument ends with a \', its value is expected to be a centering ratio (in \[0,1\]) rather than a position.
Usual centering ratio are \{ 0=left-justified | 0.5=centered | 1=right-justified \}.

Default values:

- \{x=y=z=c=0', 'opacity=1', 'opacity\_mask=(undefined)' \} and 'max\_opacity\_mask=1'.

Example 443: image.jpg +crop 40\%,40\%,60\%,60\% resize[-1] 200\%,200\%,1,3,5 frame[-1] 2,2,0
image[0] [-1],30\%,30\% keep[0]

2.10.14 \textit{line} (+)

Arguments:

- \{x0\%,y0\%,x1\%,y1\%}, \text{opacity}, \text{pattern}, \text{color1},...

Draw specified colored line on selected images.
'pattern' is a hexadecimal number starting with '0x' which can be omitted even if a color is specified.

Default values:

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

Example 444: image.jpg repeat 500 line 50%,50%,\{u(w)\},\{u(h)\},0.5,\{\text{-RGB}\} done line
0,0,100%,1,0xCCCCCCCC,255 line 100%,0,0,100%,1,0xCCCCCCCC,255

2.10.15 \textit{linethick}

Arguments:

- \textit{x0[\%], y0[\%], x1[\%], y1[\%], thickness, opacity, color1}

Draw specified colored thick line on selected images.

Default values:

- '\textit{thickness=2}', '\textit{opacity=1}' and '\textit{color1=0}'.

[0]: 'image.jpg' (480x320x1x3)
Example 445: 400, 400, 1, 3 repeat 100 linethick \{u([w,h,w,h,5])\}, 0.5, \$\{-RGB\} done

2.10.16 mandelbrot (+)

Arguments:

-\ z0r, z0i, z1r, z1i, iteration\_max>=0, is\_julia=\{0 | 1\}, c0r, c0i, opacity

Draw mandelbrot/julia fractal on selected images.

Default values:

- 'iteration\_max=100', 'is\_julia=0', 'c0r=c0i=0' and 'opacity=1'.

Example 446: 400,400 mandelbrot -2.5,-2,2,2,1024 map 0 +blur 2 elevation3d[-1] -0.2

2.10.17 marble

Arguments:

- image_weight, pattern_weight, angle, amplitude, sharpness=0, anisotropy=0, alpha, sigma, cut_low=0, cut_high=0

Render marble like pattern on selected images.

Default values:

- 'image_weight=0.2', 'pattern_weight=0.1', 'angle=45', 'amplitude=0', 'sharpness=0.4' and 'anisotropy=0.8',

- 'alpha=0.6', 'sigma=1.1' and 'cut_low=cut_high=0'.

Example 446: 400,400 mandelbrot -2.5,-2,2,2,1024 map 0 +blur 2 elevation3d[-1] -0.2
2.10.18 maze

Arguments:

- width>0, height>0, cell_size>0

Input maze with specified size.
2.10.19  *maze_mask*

**Arguments:**

- `_cellsize>`0

Input maze according to size and shape of selected mask images.
Mask may contain disconnected shapes.
2.10.20 \textit{newton\_fractal}

\textbf{Arguments:}

\begin{itemize}
  \item \texttt{z0r,z0i,z1r,z1i,\_angle,0<=\_descent\_method<=2,\_iteration\_max>=0,}
  \_convergence\_precision>0,\_expr\_p(z),\_expr\_dp(z),\_expr\_d2p(z)
\end{itemize}

Draw newton fractal on selected images, for complex numbers in range \((z0r,z0i) - (z1r,z1i)\).
Resulting images have 3 channels whose meaning is \([\text{last}\_zr, \text{last}\_zi, \text{nb\_iter\_used\_for\_convergence}]\).

\textit{descent\_method} can be \{0=secant | 1=newton | 2=householder\}.

\textbf{Default values:}

\begin{itemize}
  \item \texttt{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\n  \_expr\_d2p(z)=6*z'.
\end{itemize}

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{fractal_example}
\caption{Example output of \texttt{newton\_fractal} command.}
\end{figure}

\textbf{Example 450} : 400,400 \texttt{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\" \& \"[ atan2(i1,i0)*90+20,1,cut(i2/30,0.2,0.7) ]\" hsl2rgb

2.10.21 \textit{object3d (\texttt{+})}

\textbf{Arguments:}

\begin{itemize}
  \item \texttt{[object3d],x[%],y[%],z,\_opacity,\_rendering\_mode,\_is\_double\_sided={ 0 | 1 }
  ,\_is\_zbuffer={ 0 | 1 }
  ,\_focale,\_light\_x,\_light\_y,\_light\_z,\_specular\_lightness,\_specular\_shininess}
\end{itemize}

Draw specified 3D object on selected images.
\textit{(eq. to 'j3d').\texttt{\textbackslash n)}}.
'rendering_mode' can be \{ 0=dots | 1=wireframe | 2=flat | 3=flat-shaded | 4=gouraud-shaded | 5=phong-shaded \}.

**Default values:**

- 'x=y=z=0', 'opacity=1' and 'is_zbuffer=1'. All other arguments take their default values from the 3D environment variables.

Example 451: `image.jpg torus3d 100,10 cone3d 30,-120 add3d[-2,-1] rotate3d. 1,1,0,60 object3d[0] [-1],50%,50% keep[0]`

### 2.10.22 pack sprites

**Arguments:**

- \_nb_scales\_=0,0<\_min\_scale\_<100,\_allow\_rotation\_={ 0=0 deg. | 1=180 deg. | 2=90 deg. | 3=any },\_spacing,\_precision\_=0,\_max\_iterations\_=0

Try to randomly pack as many sprites as possible onto the 'empty' areas of an image. Sprites can be eventually rotated and scaled during the packing process. First selected image is the canvas that will be filled with the sprites. Its last channel must be a binary mask whose zero values represent potential locations for drawing the sprites. All other selected images represent the sprites considered for packing. Their last channel must be a binary mask that represents the sprite shape (i.e. a 8-connected component). The order of sprite packing follows the order of specified sprites in the image list. Sprite packing is done on random locations and iteratively with decreasing scales. 'nb_scales' sets the number of decreasing scales considered for all specified sprites to be packed. 'min_scale' (in %) sets the minimal size considered for packing (specified as a percentage of the original size).
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 452:

```
512,512,1,3,"min(255,y*c/2)" 100%,100% circle 50%,50%,100,1,255 append c
image.jpg resize2dy[-1] 24 to_rgba pack_sprites 3,25
```

### 2.10.23 piechart

**Arguments:**

- `label_height>=0,label_R,label_G,label_B,"label1",value1,R1,G1,B1,...,
  "labelN",valueN,RN,GN,BN`

Draw pie chart on selected (RGB) images.
2.10.24  plasma (+)

Arguments:

- α, β, scale > 0

Draw a random colored plasma fractal on selected images. This command implements the so-called 'Diamond-Square' algorithm.

Default values:

- ‘α=1’, ‘β=1’ and ‘scale=8’.
408
CHAPTER 2. LIST OF COMMANDS

**Example 454**: 400,400,1,3 plasma

Tutorial page:
https://gmic.eu/tutorial/_plasma.shtml

### 2.10.25 point (+)

**Arguments:**

- \(x\%\), \(y\%\), \(z\%\), \(opacity\), \(color1\), ...

Set specified colored pixel on selected images.

**Default values:**

- \('z=0'\), \('opacity=1'\) and \('color1=0'\).
2.10.26 **polka_dots**

Arguments:

- **diameter>=0**, **density**, **offset1**, **offset2**, **angle**, **aliasing**, **shading**, **opacity**, **color**, ...

Draw dots pattern on selected images.

Default values:

- `'density=20'`, `'offset1=offset2=50'`, `'angle=0'`, `'aliasing=10'`, `'shading=1'`, `'opacity=1'` and `'color=255'`.

Example 455: `image.jpg` repeat 10000 point `{u(100)}, {u(100)}, 0, 1, $-RGB$ done`
2.10.27  polygon (+)

Arguments:

- $N\geq 1, x_1[^\%], y_1[^\%], \ldots, x_N[^\%], y_N[^\%], \text{opacity}, \text{pattern}, \text{color}_1, \ldots$

Draw specified colored $N$-vertices polygon on selected images. 'pattern' is an hexadecimal number starting with '0x' which can be omitted even if a color is specified. If a pattern is specified, the polygon is drawn outlined instead of filled.

Default values:

- 'opacity=1', 'pattern=(undefined)' and 'color1=0'.
Example 457: `image.jpg polygon 4,20%,20%,80%,80%,70%,80%,0.3,0,255,0 polygon 4,20%,20%,80%,80%,70%,80%,1,0xCCCCCCCC,255`

Example 458: `image.jpg 2,16,1,'u(if(x,{h},{w}))' polygon[-2] {h},{'},0.6,255,0,255 remove[-1]`
2.10.28  \textit{quiver}

\textbf{Arguments:}

- \{\text{function}, \text{image}\}, \text{sampling} \% > 0, \text{factor} > 0, \text{is\_arrow} = \{0 \ | \ 1\}, \text{opacity}, \text{color1},\ldots

Draw specified 2D vector/orientation field on selected images.

\textbf{Default values:}

- 'sampling=5\%', 'factor=1', 'is\_arrow=1', 'opacity=1', 'pattern=(undefined)'

and 'color1=0'.

\begin{example}
100, 100, 1, 2, 'if(c==0, x-w/2, y-h/2)' 500, 500, 1, 3, 255 quiver[-1] [-2], 10
\end{example}
2.10.29 rectangle

Arguments:

- \( x_0, y_0, x_1, y_1, \) \( \text{opacity}, \) \( \text{pattern}, \) \( \text{color1}, \ldots \)

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

Default values:

- 'opacity=1', 'pattern=(undefined)' and 'color1=0'.
414 

CHAPTER 2. LIST OF COMMANDS

Example 461: image.jpg repeat 30 rectangle
\{u(100)\},\{u(100)\},\{u(100)\},\{u(100)\},0.3,8{-RGB} done

2.10.30 rorschach

Arguments:

- ‘smoothness[\%] &gt;= 0’, ‘mirroring=\{ 0=none | 1=x | 2=y | 3=xy \}"

Render rorschach-like inkblots on selected images.

Default values:

- ‘smoothness=5\%’ and ‘mirroring=1’.
2.10.31 *sierpinski*

**Arguments:**

- `recursion_level >= 0`

Draw Sierpinski triangle on selected images.

**Default value:**

- `recursion_level=7`.
2.10.32  *spiralbw*

**Arguments:**

- width > 0, height > 0, is_2dcoords = { 0 | 1 }

Input a 2D rectangular spiral image with specified size.

**Default values:**

- ‘height = width’ and ‘is_2dcoords = 0’.
Example 464: `spiralbw 16`

Example 465: `image.jpg spiralbw \{w,h\},1 +warp\{0\} [1],0 +warp\{2\} [1],2`

### 2.10.33 `spline`

**Arguments:**
• \(x0, y0, u0, v0, x1, y1, u1, v1, \text{opacity}, \text{color1}, \ldots\)

Draw specified colored spline curve on selected images (cubic hermite spline).

**Default values:**

• ‘\text{opacity}=1’ and ‘\text{color1}=0’.

**Example 466:**

\begin{verbatim}
image.jpg repeat 30 spline \{u(100)\},\{u(100)\},\{u(-600,600)\},\{u(-600,600)\},
\{u(100)\},\{u(100)\},\{u(-600,600)\},\{u(-600,600)\},0.6,255
done
\end{verbatim}

2.10.34 **tetraedron shade**

**Arguments:**

• \(x0, y0, z0, x1, y1, z1, x2, y2, z2, x3, y3, z3, R0, G0, B0, \ldots, R1, G1, B1, \ldots, R2, G2, B2, \ldots, R3, G3, B3, \ldots\)

Draw tetraedron with interpolated colors on selected (volumetric) images.

2.10.35 **text (+)**

**Arguments:**

• \(\text{text}, x\% | y\% | \ldots, \text{font_height}\% | \text{opacity}, \text{color1}, \ldots\)

Draw specified colored text string on selected images.

*(eq. to ‘\text{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
2.10. IMAGE DRAWING

a position.
Usual centering ratio are \{ 0=left-justified | 0.5=centered | 1=right-justified \}.
Sizes '13' and '128' are special and correspond to binary fonts (no-antialiasing).
Any other font size is rendered with anti-aliasing.
Specifying an empty target image resizes it to new dimensions such that the image contains the entire text string.

Default values:

• 'x=y=0.01', 'font_height=16', 'opacity=1' and 'color1=0'.

Example 467: image.jpg resized dy 600 y=0 repeat 30 text {2+}$*: This is a nice text, isn't it?^{*,10,5y,\{2+\}},0.9,255 y+=\{2+\} done
2.10.36 **text** _outline_

**Arguments:**

- `text, x[%] | y[%] | font_height[%]>0, outline=0, opacity, color1,...`

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 469: `image.jpg` text outline "Hi there!", 10, 10, 63, 3

2.10.37 \textit{triangleshade}

Arguments:

- $x_0, y_0, x_1, y_1, x_2, y_2, R_0, G_0, B_0, \ldots, R_1, G_1, B_1, \ldots, R_2, G_2, B_2, \ldots$

Draw triangle with interpolated colors on selected images.
Example 470: `image.jpg triangle_shade 20,20,400,100,120,200,255,0,0,0,255,0,0,0,255`

### 2.10.38 truchet

**Arguments:**

- `scale>0, radius>0, pattern_type={ 0=straight | 1=curved }`

Fill selected images with random truchet patterns.

**Default values:**

- `'scale=32', 'radius=5' and 'pattern_type=1'.
2.10.39 *turbulence*

**Arguments:**

- \( \text{radius} > 0 \), \( \text{octaves} = \{1,2,3...12\} \), \( \text{alpha} > 0 \), \( \text{difference} = \{-10,10\} \), \( \text{mode} = \{0,1,2,3\} \)

Render fractal noise or turbulence on selected images.

**Default values:**

- ‘radius=32’, ‘octaves=6’, ‘alpha=3’, ‘difference=0’ and ‘mode=0’.
Example 472: $[400,400,1,3 \ \text{turbulence\ } 16$

Tutorial page:
https://gmic.eu/tutorial/_turbulence.shtml

2.10.40 yinyang

Draw a yin-yang symbol on selected images.
2.11 Matrix Computation

2.11.1 dijkstra (+)

Arguments:

- starting_node>=0, ending_node>=0

Compute minimal distances and paths from specified adjacency matrices by the Dijkstra algorithm.

2.11.2 eigen (+)

Compute the eigenvalues and eigenvectors of selected symmetric matrices or matrix fields. If one selected image has 3 or 6 channels, it is regarded as a field of 2x2 or 3x3 symmetric matrices, whose eigen elements are computed at each point of the field.
Example 474: \((1,0,0;0,2,0;0,0,3) +\text{eigen}\)

Example 475: \(\text{image.jpg structuretensors blur 2 eigen split[0] c}\)

Tutorial page:
https://gmic.eu/tutorial/_eigen.shtml
2.11.3  *invert* (+)

**Arguments:**

- **solver** = \{0=SVD | 1=LU\}

Compute the inverse of the selected matrices.
SVD solver is slower but less numerically unstable than LU.

**Default value:**

- `solver=1`

---

2.11.4  *orthogonalize*

**Arguments:**

- **mode** = \{0=orthogonalize | 1=orthonormalize\}

Orthogonalize or orthonormalize selected matrices, using Modified Gram-Schmidt process.

**Default value:**

- `mode=0`

---

2.11.5  *mproj* (+)

**Arguments:**

- `{dictionary}, method, max_iter` = \{0=auto | >0\}, max_residual > 0
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.
- $\geq 3 =$ 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'.

### 2.11.6 solve (+)

**Arguments:**

- [image]

Solve linear system $AX = B$ for selected $B$-matrices and specified $A$-matrix. If the system is under- or over-determined, the least squares solution is returned (using SVD-based solver).

![Image of matrices](image)
2.11.7 \textit{svd} (+)

Compute SVD decomposition of selected matrices.

Example 477: \((0,1,0;1,0,0;0,0,1) \ (1;2;3) +\text{solve}[-1] \ [-2]

Example 478: 10,10,1,1,’if(x==y,x+u(-0.2,0.2),0)’ +\text{svd}

2.11.8 \textit{transpose}

Transpose selected matrices.
2.11.9  trisolve (+)

Arguments:

- [image]

Solve tridiagonal system $AX = B$ for selected $B$-vectors and specified tridiagonal $A$-matrix. Tridiagonal matrix must be stored as a 3 column vector, where 2nd column contains the diagonal coefficients, while 1st and 3rd columns contain the left and right coefficients.
2.12 3D Meshes

2.12.1 add3d (+)

Arguments:

• tx, ty, tz
• [object3d]
• (no arg)

Shift selected 3D objects with specified displacement vector, or merge them with specified 3D object, or merge all selected 3D objects together.
(eq. to ’+3d’).

Default values:

• ’ty=tz=0’. 
Example 481: `sphere3d 10 repeat 5 +add3d[-1] 10, {u(-10,10)}, 0 color3d[-1] $\{-RGB\}$ done add3d

Example 482: `repeat 20 torus3d 15, 2 color3d[-1] $\{-RGB\}$ mul3d[-1] 0.5, 1 if $>$%2 rotate3d[-1] 0, 1, 0, 90 fi add3d[-1] 70 add3d rotate3d[-1] 0, 0, 1, 18 done double3d 0
2.12.3D MESHES

2.12.2 animate3d
Arguments:
- width>0, height>0, angle_dx, angle_dy, angle_dz, zoom_factor>0, filename

Animate selected 3D objects in a window.
If argument 'filename' is provided, each frame of the animation is saved as a numbered filename.
Default values:
- 'width=640', 'height=480', 'angle_dx=0', 'angle_dy=1', 'angle_dz=0', 'zoom_factor=1' and 'filename=(undefined)'.

2.12.3 apply_camera3d
Arguments:
- pos_x, pos_y, pos_z, target_x, target_y, target_z, up_x, up_y, up_z

Apply 3D camera matrix to selected 3D objects.
Default values:
- 'target_x=0', 'target_y=0', 'target_z=0', 'up_x=0', 'up_y=1' and 'up_z=0'.

2.12.4 apply_matrix3d
Arguments:
- a11, a12, a13, ..., a31, a32, a33

Apply specified 3D rotation matrix to selected 3D objects.
Example 483: \texttt{torus3d 10,1 +apply\_matrix3d \{mul(rot(1,0,1,-15),[1,0,0,0,2,0,0,0,8],3)\}}
\texttt{double3d 0}

\subsection*{2.12.5 \texttt{array3d}}

\textbf{Arguments:}

- \texttt{size\_x>1, size\_y>1, size\_z>1, offset\_x\%, offset\_y\%, offset\_z\%}

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

\textbf{Default values:}

- ‘\texttt{size\_y=1, size\_z=1}’ and ‘\texttt{offset\_x=offset\_y=offset\_z=100\%}’.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{example483.png}
\caption{Example 484: \texttt{torus3d 10,1 +array3d 5,5,5,110\%,110\%,300\%}}
\end{figure}

\subsection*{2.12.6 \texttt{arrow3d}}

\textbf{Arguments:}

- \texttt{x0,y0,z0,x1,y1,z1, radius\%>0, head\_length\%>0, head\_radius\%>0}

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

\textbf{Default values:}

- ‘\texttt{radius=5\%}’, ‘\texttt{head\_length=25\%}’ and ‘\texttt{head\_radius=15\%}’. 
Example 485: \[ \text{repeat 10 } a=\{\$>\cdot2\cdot\pi/10\} \text{ arrow3d } 0,0,0,\{\cos(a)\},\{\sin(a)\},-0.5 \text{ done } +3d \]

### 2.12.7 axes3d

**Arguments:**

- \( \text{size}_x, \text{size}_y, \text{size}_z, \text{font.size}>0, \text{label}_x, \text{label}_y, \text{label}_z, \text{is.origin}=\{0=\text{no} \mid 1=\text{yes}\} \)

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

**Default values:**

- \(\text{‘size}_x=\text{size}_y=\text{size}_z=1’, \text{‘font.size}=23’, \text{‘label}_x=X’, \text{‘label}_y=Y’, \text{‘label}_z=Z’ \text{ and ‘is.origin}=1’\)
2.12.8 boundingbox3d

Replace selected 3D objects by their 3D bounding boxes.
2.12.9  box3d

Arguments:

- \texttt{\_size}_x, \texttt{\_size}_y, \texttt{\_size}_z

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

Default values:

- \texttt{\_size}_x=1 and \texttt{\_size}_z=\texttt{\_size}_y=\texttt{\_size}_x.

\begin{center}
\includegraphics[width=0.5\textwidth]{box3d.png}
\end{center}

Example 488: box3d 100,40,30 +primitives3d 1 color3d[-2] $\$\{-RGB\}$

2.12.10  center3d

Center selected 3D objects at (0,0,0).
\textit{(eq. to \texttt{c3d}).}
Example 489: repeat 100 circle3d {u(100)}, {u(100)}, {u(100)}, 2 done add3d color3d[-1] 255,0,0 +center3d color3d[-1] 0,255,0 add3d

2.12.11 circle3d

Arguments:

- \(x_0, y_0, z_0, \text{radius}\) \(\geq 0\)

Input 3D circle at specified coordinates.

Default values:

- \('x_0 = y_0 = z_0 = 0'\) and \('\text{radius}=1'\).
Example 490: repeat 500 $a=\{a>\pi/250\}$ circle3d \{cos(3*$a$),sin(2*$a$),$a/50\} color3d[-1] $\{\text{-RGB}\},0.4$ done add3d

2.12.12 circles3d

Arguments:

- \_radius\_>=0, is\_wireframe=\{ 0 | 1 \}

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

Default values:

- ‘radius=1’ and ‘is\_wireframe=1’.
2.12.13  \textit{color3d} (+)

Arguments:

- \textit{R,G,B,opacity}

Set color and opacity of selected 3D objects.
\textit{(eq. to `col3d`).}

Default value:

- `B=G=R` and `opacity=(undefined)`. 
2.12.3D MESHES

Example 492: torus3d 100,10 double3d 0 repeat 7 +rotate3d[-1] 1,0,0,20 color3d[-1] $\{-$RGB$\}$
done add3d

2.12.14 colorcube3d
Input 3D color cube.

Example 493: colorcube3d mode3d 2 +primitives3d 1
2.12.15 \textit{cone3d}

\textbf{Arguments:}

- \_radius, \_height, \_nb\_subdivisions > 0

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

\textbf{Default value:}

- \textquote{radius=1}, \textquote{height=1} \text{ and } \textquote{nb\_subdivisions=24}.

\begin{itemize}
  \item \text{Example 494:} \texttt{cone3d 10,40 +primitives3d 1 color3d[-2] 5{-RGB}}
\end{itemize}

2.12.16 \textit{cubes3d}

\textbf{Arguments:}

- \_size \geq 0

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

\textbf{Default value:}

- \textquote{size=1}. 
2.12.3D MESHES

Example 495: `image.jpg luminance resize2dy 40 threshold 50% * 255 pointcloud3d color3d[-1] 255,255,255 cubes3d 1`

2.12.17  *cup3d*

Arguments:

- `_resolution>0`

Input 3D cup object.

Default value:

- `'resolution=128'`. 
2.12.18  \textit{cylinder3d}

Arguments:

- \texttt{\_radius,\_height,\_nb_subdivisions}>0

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

Default value:

- ‘radius=1’, ‘height=1’ and ‘nb_subdivisions=24’.
2.12. 3D MESHES

Example 497: cylinder3d 10,40 +primitives3d 1 color3d[-2] $\{\{-RGB\}\}$

2.12.19 delaunay3d

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

Example 498: 500,500 noise 0.05,2 eq 1 * 255 +delaunay3d color3d[1] 255,128,0 dilate circ[0] 5 to_rgb[0] +object3d[0] [1],0,0,0,1,1 max[-1] [0]

2.12.20 distribution3d

Get 3D color distribution of selected images.
2.12.21  \textit{div3d} (+)

\textbf{Arguments:}

- \texttt{factor}
- \texttt{factor}_x, \texttt{factor}_y, \texttt{factor}_z

Scale selected 3D objects isotropically or anisotropically, with the inverse of specified factors. \textit{(eq. to '/3d').}

\textbf{Default value:}

- \texttt{‘factor}_z=0’.
2.12. 3D MESHES

Example 500: torus3d 5,2 repeat 5 +add3d[-1] 12,0,0 div3d[-1] 1.2 color3d[-1] &{-RGB} done
   add3d

2.12.22 double3d (+)

Arguments:

- \_is\_double\_sided={ 0 | 1 }

Enable/disable double-sided mode for 3D rendering.
(eq. to ‘db3d’).

Default value:

- ‘is\_double\_sided=1’.
2.12.23  elevation3d (+)

Arguments:

- z-factor
- [elevation_map]
- 'formula'
- (no arg)

Build 3D elevation of selected images, with a specified elevation map. When invoked with (no arg) or 'z-factor', the elevation map is computed as the pointwise L2 norm of the pixel values. Otherwise, the elevation map is taken from the specified image or formula.
2.12. 3D MESHES

Example 502: `image.jpg blur 5 elevation3d 0.5`

Example 503: `128,128,1,3,u(255) plasma 10,3 blur 4 sharpen 10000 elevation3d[-1]

\[ X=(x-64)/6; Y=(y-64)/6; -100*exp(-(X^2+Y^2)/30)*abs(cos(X)*sin(Y)) \]`

2.12.24 empty3d

Input empty 3D object.
2.12.25 extrude3d

Arguments:

- \_depth>0, \_resolution>0, \_smoothness[%]>0

Generate extruded 3D object from selected binary XY-profiles.

Default values:

- ‘depth=16’, ‘resolution=1024’ and ‘smoothness=0.5%’.
2.12. 3D MESHES

Example 505: image.jpg threshold 50% extrude3d 16

2.12.26 focale3d (+)

Arguments:

- focale

Set 3D focale.
(eq. to ‘f3d’).
Set ’focale’ to 0 to enable parallel projection (instead of perspective).
Set negative ’focale’ will disable 3D sprite zooming.

Default value:

- ‘focale=700’.
Example 506: repeat 5 torus3d 100,30 rotate3d[-1] 1,1,0,60 focale3d {5<90} snapshot3d[-1] 400 done remove[0]

2.12.27  *gaussians3d*

Arguments:

- *size* > 0, *opacity*

Convert selected 3D objects into set of 3D gaussian-shaped sprites.
2.12. 3D MESHES

Example 507: image.jpg r2dy 32 distribution3d gaussians3d 20 colorcube3d primitives3d[-1] 1 +3d

2.12.28  gmic3d

Input a 3D G’MIC logo.

Example 508: gmic3d +primitives3d 1
2.12.29  

**gyroid3d**

**Arguments:**

- \_resolution\_>0,\_zoom

Input 3D gyroid at (0,0,0), with specified resolution.

**Default values:**

- 'resolution=32' and 'zoom=5'.

---

2.12.30  

**histogram3d**

Get 3D color histogram of selected images.
Example 510: image.jpg resize2dx 64 histogram3d circles3d 3 opacity3d 0.75 colorcube3d
primitives3d[-1] 1 add3d

2.12.31 image6cube3d
Generate 3D mapped cubes from 6-sets of selected images.

Example 511: image.jpg animate flower,"30,0","30,5",6 image6cube3d
2.12.32 \textit{imageblocks3d}

\textbf{Arguments:}

- \texttt{maximum\_elevation, smoothness[\%]}\geq0

Generate 3D blocks from selected images.
Transparency of selected images is taken into account.

\textbf{Default values:}

- \texttt{\textquotesingle maximum\_elevation=10\textquotesingle \ and \textquotesingle smoothness=0	extquotesingle}.

\textbf{Example 512}: \texttt{image.jpg\ resize2dy\ 32\ imageblocks3d\ -20\ mode3d\ 3}

2.12.33 \textit{imagecube3d}

Generate 3D mapped cubes from selected images.
2.12.34  \textit{imageplane3d}

Generate 3D mapped planes from selected images.
2.12.35  *imagepyramid3d*

Generate 3D mapped pyramids from selected images.

Example 515: `image.jpg imagepyramid3d`

2.12.36  *imagerubik3d*

**Arguments:**

- `xy_tiles>=1`, `0<=xy_shift<=100`, `0<=z_shift<=100`

Generate 3D mapped rubik’s cubes from selected images.

**Default values:**

- `'xy_tiles=3'`, `'xy_shift=5'` and `'z_shift=5'`. 
2.12.37 imagesphere3d

Arguments:

- \(_{resolution1}>3\), \(_{resolution2}>3\)

Generate 3D mapped sphere from selected images.

Default values:

- \(\text{'resolution1}=32\) and \(\text{'resolutions2}=16\).
2.12.38 *isoline3d* (+)

**Arguments:**

- `isovalue[%]`
- `‘formula’,value,x0,y0,x1,y1,size_x>0[%],size_y>0[%]`

Extract 3D isolines with specified value from selected images or from specified formula.

**Default values:**

- `‘x0=y0=-3’, ‘x1=y1=3’ and ‘size_x=size_y=256’.`
2.12.39 \textit{isosurface3d} (+)

\textbf{Arguments:}

Example 518: image.jpg blur 1 isoline3d 50%

Example 519: isoline3d \textbf{"}X=x-w/2;Y=y-h/2;(X^2+Y^2)/20\textbf{"},10,-10,-10,10,10
• `isovalue[%]`
• `'formula',value,x0,y0,z0,x1,y1,z1,size_x>0[%],size_y>0[%],size_z>0[%]`

Extract 3D isosurfaces with specified value from selected images or from specified formula.

**Default values:**

• `'x0=y0=z0=-3', 'x1=y1=z1=3' and 'size_x=size_y=size_z=32'`.

**Example 520:**

`image.jpg resize2dy 128 luminance threshold 50% expand_z 2,0 blur 1 isosurface3d 50% mul3d 1,1,30`
Example 521: \( \text{isosurface3d } (x^2+y^2+\text{abs}(z)^{\text{abs}(4\cos(x+y+z^3))})') \)

2.12.40 \textit{label3d}

Arguments:

- "text", font_height \textgreater{} 0, opacity, color1, ...

Generate 3D text label.

Default values:

- \( \text{'font_height=13', 'opacity=1' and 'color=255,255,255'.} \)

2.12.41 \textit{label_points3d}

Arguments:

- \_label_size > 0, \_opacity

Add a numbered label to all vertices of selected 3D objects.

Default values:

- \( \text{'label_size=13' and 'opacity=0.8'.} \)
2.12.42  \textit{lathe3d}

Arguments:

- \_resolution > 0, \_smoothness[\%] > 0, \_max_angle > 0

Generate 3D object from selected binary XY-profiles.

Default values:

- \textquote{resolution=128}, \textquote{smoothness=0.5\%} \textit{and} \textquote{max_angle=361}.
2.12.43  light3d (+)

Arguments:

- `position_x`, `position_y`, `position_z`
- `[texture]`
- `(no arg)`

Set the light coordinates or the light texture for 3D rendering. 
(`eq. to 'l3d').` 

(no arg) resets the 3D light to default.
Example 524: `torus3d 100,30 double3d 0 specs3d 1.2 repeat 5 light3d \{s>100\},0,-300 +snapshot3d[0] 400 done remove[0]`

2.12.44  **line3d**

**Arguments:**

- \( x_0, y_0, z_0, x_1, y_1, z_1 \)

Input 3D line at specified coordinates.
Example 525: repeat 100 a={$>$*pi/50} line3d 0,0,0,{$cos(3*$a)$},{$sin(2*$a)$},0 color3d. ${-}RGB$

done add3d

2.12.45  lissajous3d

Arguments:

- resolution>1, a, A, b, B, c, C

Input 3D lissajous curves (x(t)=sin(a*t+A*2*pi), y(t)=sin(b*t+B*2*pi), z(t)=sin(c*t+C*2*pi)).

Default values:

- ‘resolution=1024’, ‘a=2’, ‘A=0’, ‘b=1’, ‘B=0’, ‘c=0’ and ‘C=0’.
2.12.46 \textit{mode3d} (+)

**Arguments:**

- \_mode

Set static 3D rendering mode. (eq. to \textit{m3d}).

`mode` can be \{-1=bounding-box | 0=dots | 1=wireframe | 2=flat | 3=flat-shaded | 4=gouraud-shaded | 5=phong-shaded \}.

Bounding-box mode (`mode=-1`) is active only for the interactive 3D viewer.

**Default value:**

- `mode=4`.
2.12. 3D MESHES

Example 527: (0,1,2,3,4,5) double3d 0 repeat w torus3d 100,30 rotate3d[-1] 1,1,0,60 mode3d
{0,0,0} snapshot3d[-1] 300 done remove[0]

2.12.47  mod3d3d (+)

Arguments:
  * _mode_

Set dynamic 3D rendering mode for interactive 3D viewer.
(eq. to 'md3d').
'mode' can be {-1=bounding-box | 0=dots | 1=wireframe | 2=flat | 3=flat-shaded | 4=gouraud-shaded
| 5=phong-shaded}.

Default value:
  * 'mode=-1'.

2.12.48  mul3d (+)

Arguments:
  * factor
  * factor_x,factor_y,factor_z

Scale selected 3D objects isotropically or anisotropically, with specified factors.
(eq. to 'x3d').

Default value:
  * 'factor_z=0'.

[Images of 3D meshes shown]
Example 528: torus3d 5,2 repeat 5 +add3d[-1] 10,0,0 mul3d[-1] 1.2 color3d[-1] $\{-\text{RGB}\}$ done
add3d

2.12.49 normalize3d
Normalize selected 3D objects to unit size.
(eq. to 'n3d').
Example 529: repeat 100 circle3d \{u(3)\},\{u(3)\},\{u(3)\},0.1 done add3d color3d[-1] 255,0,0 +normalize3d[-1] color3d[-1] 0,255,0 add3d

2.12.50 opacity3d (+)

Arguments:

- \_opacity

Set opacity of selected 3D objects.  
\textit{(eq. to \textquote{\textasciitilde o3d}).}

Default value:

- \textquote{\textasciitilde opacity=1}.

Example 530: torus3d 100,10 double3d 0 repeat 7 +rotate3d[-1] 1,0,0,20 opacity3d[-1] \{u\} done add3d

2.12.51 parametric3d

Arguments:

- \_x(a,b)\_y(a,b)\_z(a,b)\_amin\_amax\_bmin\_bmax\_res\_a>0\_res\_b>0\_res\_x>0\_res\_y>0\_res\_z>0\_smoothness>0\_isovalue>0

Input 3D object from specified parametric surface (x(a,b),y(a,b),z(a,b)).

Default values:
• \('x=(2+\cos(b))\sin(a)', \('y=(2+\cos(b))\cos(a)', \('c=\sin(b)', \('\amin=\pi', \('\amax=\pi', \('\bmin=\pi' \text{ and } \bmax=\pi', \('\res_a=512', \('\res_b=\res_a', \('\res_x=64', \('\res_y=\res_x', \('\res_z=\res_y', \('\smoothness=2\%', \text{ and } \'\text{isoval}=10\%'.

Example 531: \texttt{parametric3d}

\textbf{2.12.52 pca\_patch3d}

\textbf{Arguments:}

- \_\texttt{patch\_size}>0, \_\texttt{M}>0, \_\texttt{N}>0, \_\texttt{normalize\_input}\{} 0 | 1 \}, \_\texttt{normalize\_output}\{} 0 | 1 \}, \_\texttt{lambda\_xy}

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.

\textbf{Default values:}

- \'\texttt{patch\_size}=7', \'\texttt{M}=1000', \'\texttt{N}=3000', \'\texttt{normalize\_input}=1', \'\texttt{normalize\_output}=0', \text{ and } \'\texttt{lambda\_xy}=0'.

Example: \texttt{parametric3d}
2.12.53  plane3d

Arguments:

- 

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

Default values:

- ‘size_x=1’, ‘size_y=size_x’ and ‘nb_subdivisions_x=nb_subdivisions_y=24’.
2.12.54 \textit{point3d}

\textbf{Arguments:}

- $x_0, y_0, z_0$

Input 3D point at specified coordinates.
Example 534: repeat 1000 a=($>\pi/500$) point3d {cos(3*$a$),{sin(2*$a$)},0 color3d[-1] $\{-$RGB$}\}
done add3d

2.12.55 pointcloud3d

Convert selected planar or volumetric images to 3D point clouds.

Example 535: image.jpg luminance resize2dy 100 threshold 50% mul 255 pointcloud3d
color3d[-1] 255,255,255

2.12.56 pose3d

Arguments:

- p1,\ldots,p12

Apply 3D pose matrix to selected 3D objects.
Example 536: torus3d 100,20 pose3d 0.152437,1.20666,-0.546366,0,-0.535962,0.559129,1.08531,0,1.21132,0.0955431,0.548966,0,0,0,-206,1 snapshot3d 400

2.12.57 primitives3d

Arguments:

- mode

Convert primitives of selected 3D objects. (eq. to 'p3d'). ‘mode’ can be { 0=points | 1=outlines | 2=non-textured }. 
2.12.58 projections3d

Arguments:

- \(x, y, z\), \(is\ bounding\ box = \{0 \mid 1\}\)

Generate 3D xy, xz, yz projection planes from specified volumetric images.

2.12.59 pyramid3d

Arguments:

- \(width, height\)

Input 3D pyramid at (0,0,0), with specified geometry.
Example 538: \texttt{pyramid3d 100,-100 +primitives3d 1 color3d[-2] $\{-\text{RGB}\}$}

\subsection*{2.12.60 \texttt{quadrangle3d}}

\textbf{Arguments:}

- \texttt{$x_0,y_0,z_0,x_1,y_1,z_1,x_2,y_2,z_2,x_3,y_3,z_3$}

Input 3D quadrangle at specified coordinates.
Example 539:  
quadrangle3d -10,-10,10,10,-10,10,10,-10,10,10 repeat 10 +rotate3d[-1]  
0,1,0,30 color3d[-1] $\{\text{RGB}\},0.6$ done add3d mode3d 2

2.12.61  \textit{random3d}

\textbf{Arguments:}

- \textit{nb\_points\textgreater{}=0}

Input random 3D point cloud in $[0,1]^3$.

Example 540:  
random3d 100 circles3d 0.1 opacity3d 0.5

2.12.62  \textit{reverse3d} (+)

Reverse primitive orientations of selected 3D objects.  
\textit{(eq. to 'rv3d').}
2.12.63  \textit{rotate3d} (+)

\textbf{Arguments:}

\begin{itemize}
  \item \texttt{u,v,w,angle}
\end{itemize}

Rotate selected 3D objects around specified axis with specified angle (in deg.).
\textit{(eq. to \textit{r3d}).}
2.12. 3D MESHES

Example 542: torus3d 100,10 double3d 0 repeat 7 +rotate3d[-1] 1,0,0,20 done add3d

### 2.12.64 \textit{rotation3d}

**Arguments:**

- \(u,v,w,\text{angle}\)

Input 3x3 rotation matrix with specified axis and angle (in deg).

Example 543: rotation3d 1,0,0,0 rotation3d 1,0,0,90 rotation3d 1,0,0,180

### 2.12.65 \textit{sierpinski3d}

**Arguments:**
• \_\texttt{recursion\_level}\geq0, width, height

Input 3d Sierpinski pyramid.

Example 544: \texttt{sierpinski3d 3,100,-100 +primitives3d 1 color3d[-2] \$\{-RGB\}}

\textbf{2.12.66 size3d}

Return bounding box size of the last selected 3D object.

\textbf{2.12.67 skeleton3d}

\textbf{Arguments:}

• \_\texttt{metric, frame.type=\{ 0=squares | 1=diamonds | 2=circles | 3=auto \}}, \texttt{skeleton\_opacity, frame\_opacity, is\_frame\_wireframe=\{ 0 | 1 \}}

Build 3D skeletal structure object from 2d binary shapes located in selected images. 'metric' can be \{ 0=chebyshev | 1=manhattan | 2=euclidean \}.

\textbf{Default values:}

• 'metric=2', 'bones\_type=3', 'skeleton\_opacity=1' and 'frame\_opacity=0.1'. 
2.12.68  snapshot3d

Arguments:

- \texttt{size}>0, \texttt{zoom}>0, \texttt{backgroundR}, \texttt{backgroundG}, \texttt{backgroundB}, \texttt{backgroundA}
- \texttt{[background_image]}, \texttt{zoom}>0

Take 2d snapshots of selected 3D objects.
Set ‘zoom’ to 0 to disable object auto-scaling.

Default values:

- ‘size=512’, ‘zoom=1’ and ‘[background_image]=(default)’.
Example 546: \texttt{torus3d 100,20 rotate3d 1,1,0,60 snapshot3d 400,1.2,128,64,32}

Example 547: \texttt{torus3d 100,20 rotate3d 1,1,0,60 sample ? +snapshot3d[0] [1],1.2}

2.12.69 \hspace{1em} \textit{spec13d (\texttt{+})}

\textbf{Arguments:}

- \texttt{value>0}

Set lightness of 3D specular light. \textit{(eq. to \texttt{sl3d}).}

\textbf{Default value:}

- \texttt{`value=0.15'}. 

2.12. 3D MESHES

Example 548: 
(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 specl3d {-1} snapshot3d[-1] 400 done remove[0]

2.12.70 specs3d (+)

Arguments:

• value>=0

Set shininess of 3D specular light. 
(eg. to ’ss3d’).

Default value:

• ’value=0.8’.
Example 549: \((0,0.3,0.6,0.9,1.2)\) repeat w torus3d 100,30 rotate3d[-1] 1,1,0,60 color3d[-1] 255,0,0 specs3d {0,0} snapshot3d[-1] 400 done remove[0]

2.12.71 \textit{sphere3d}\ (+++

\textbf{Arguments:}

- \texttt{radius, nb\_recursions} \(\geq 0\)

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

\textbf{Default value:}

- \texttt{`nb\_recursions=3'}. 

2.12. 3D MESHES

Example 550: sphere3d 100 +primitives3d 1 color3d[{-2} $\{$RGB$}

2.12.72 spherical3d

Arguments:

- $\_nb\_azimuth\geq3, \_nb\_zenith\geq3, \_radius\_function(\phi, \theta)$

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

Default values:

- ‘\_nb\_zenith=\_nb\_azimuth=64’ and
  ‘\_radius\_function="abs(1+0.5*cos(3*\phi)*sin(4*\theta))"’.
Chapter 2. List of Commands

Example 551: spherical3d 64 +primitives3d 1

2.12.73  **spline3d**

Arguments:

- \(x0[], y0[], z0[], u0[], v0[], w0[], x1[], y1[], z1[], u1[], v1[], w1[], \_nb\_vertices\) > 2

Input 3D spline with specified geometry.

Default values:

- ‘\_nb\_vertices=128’.
Example 552: repeat 100 spline3d \{u\},\{u\},\{u\},\{u\},\{u\},\{u\},\{u\},\{u\},\{u\},128
color3d[-1] ${-\text{RGB}}$ done box3d 1 primitives3d[-1] 1 add3d

2.12.74 \textit{split3d} (+)

Arguments:

- \_keep\_shared\_data={ 0 \| 1 }\n
Split selected 3D objects into 6 feature vectors: \{ header, sizes, vertices, primitives, colors, opacities \}. (\textit{eq. to } s3d').\nTo recreate the 3D object, append these 6 images along the y-axis.

Default value:

- 'keep\_shared\_data=1'.
2.12.75 **sprite3d**

Convert selected images as 3D sprites. Selected images with alpha channels are managed.
2.12.76  *sprites3d*

**Arguments:**

- `{sprite}, .sprite_has_alpha_channel={ 0 | 1 }

Convert selected 3D objects as a sprite cloud. Set 'sprite_has_alpha_channel' to 1 to make the last channel of the selected sprite be a transparency mask.

**Default value:**

- 'mask_has_alpha_channel=0'.

Example 555:
```
torus3d 100,20 image.jpg resize2dy[-1] 64 100%,100% gaussian[-1] 30%,30% *[-1] 255 append[-2,-1] c +sprites3d[0] [1],1 display_rgba[-2]
```

2.12.77  *star3d*

**Arguments:**

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

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

**Default values:**

- 'nb_branches=5' and 'thickness=0.38'.
2.12.78  streamline3d (+)

Arguments:

\[
\begin{align*}
\text{x}, \text{y}, \text{z}, \text{L} > 0, \text{dl} > 0, \text{interpolation}, \text{is\_backward} = \{ 0 | 1 \}, \text{is\_oriented} = \{ 0 | 1 \} \\
\text{formula}, \text{x}, \text{y}, \text{z}, \text{L} > 0, \text{dl} > 0, \text{interpolation}, \text{is\_backward} = \{ 0 | 1 \}, \text{is\_oriented} = \{ 0 | 1 \}
\end{align*}
\]

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:

\[
\text{\textquote{dl}=0.1'}, \text{\textquote{interpolation}=2'}, \text{\textquote{is\_backward}=0} \text{ and } \text{\textquote{is\_oriented}=0}'.
\]
Example 557: 100,100,100,3 rand -10,10 blur 3 repeat 300 +streamline3d[0] \{u(100),u(100),u(100)\},1000,1,1 color3d[-1] $[-RGB]$ done remove[0] box3d 100 primitives3d[-1] 1 add3d

2.12.79  \textit{sub3d} (+)

Arguments:

- \(tx, ty, tz\)

Shift selected 3D objects with the opposite of specified displacement vector. \textit{(eq. to '3d')}.

Default values:

- \('ty=tz=0'\).
**Example 558**: `sphere3d 10 repeat 5 +sub3d([-1] 10, {u(-10,10)},0 color3d([-1] $-{RGB}$) done add3d`

### 2.12.80 `superformula3d`

**Arguments:**

- `resolution>1,m>=1,n1,n2,n3`

Input 2D superformula curve as a 3D object.

**Default values:**

- ‘`resolution=1024’, `m=8’, `n1=1’, `n2=5’ and `n3=8’.’
2.12.81  tensors3d

Arguments:

- `_radius_factor>_0, shape={ 0=box | >N=ellipsoid }, _radius_min>_0`

Generate 3D tensor fields from selected images. when `shape>_0`, it gives the ellipsoid shape precision.

Default values:

- `'radius_factor=1', 'shape=2' and 'radius_min=0.05'`. 
2.12.82  \texttt{text\_pointcloud3d}

\textbf{Arguments:}

- \texttt{\_\_text1}, \texttt{\_\_text2}, \texttt{\_smoothness}

Input 3D text pointcloud from the two specified strings.

\textbf{Default values:}

- \texttt{'text1=\"text1\"'}, \texttt{'text2=\"text2\"'} and \texttt{\_smoothness=1'}. 
2.12.3D MESHES

Example 561: `text_pointcloud3d "G'MIC","Rocks!"

2.12.83 text3d

Arguments:

- `text`, `font_height>0`, `depth>0`, `smoothness`

Input a 3D text object from specified text.

Default values:

- `font_height=53`, `depth=10` and `smoothness=1.5`. 
2.12.84  `texturize3d`

Arguments:

- `[ind_texture], [ind_coords]`

Texturize selected 3D objects with specified texture and coordinates. (eq. to `t3d`). When `[ind_coords]` is omitted, default XY texture projection is performed.

Default value:

- `ind_coords=(undefined)`.
Example 563: image.jpg torus3d 100,30 texturize3d[-1] [-2] keep[-1]

2.12.85 torus3d

Arguments:

- \_radius1, \_radius2, \_nb_subdivisions1>2, \_nb_subdivisions2>2

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

Default values:

- ‘radius1=1’, ‘radius2=0.3’, ‘nb_subdivisions1=24’ and ‘nb_subdivisions2=12’.
Example 564: \( \text{torus3d 10,3 +primitives3d 1 color3d[-2] $\{$RGB\$} } \)

### 2.12.86 \textit{triangle3d}

**Arguments:**

- \( x_0, y_0, z_0, x_1, y_1, z_1, x_2, y_2, z_2 \)

Input 3D triangle at specified coordinates.
2.12. 3D MESHES

Example 565: repeat 100 a={$>pi/50} triangle3d 0,0,0,0,0,3,\{\cos(3*a)\},\{\sin(2*a)\},0
  color3d[-1] ${-RGB} done add3d

2.12.87 volume3d

Transform selected 3D volumetric images as 3D parallelepipedic objects.

Example 566: image.jpg animate blur,0,5,30 append z volume3d

2.12.88 weird3d

Arguments:

- _resolution_>0

Input 3D weird object at (0,0,0), with specified resolution.

Default value:

- ‘resolution=32’.
2.13 Control Flow

2.13.1 apply_parallel

Arguments:

- "command"

Apply specified command on each of the selected images, by parallelizing it for all images of the list. (*eq. to 'ap*).
Example 568: `image.jpg +mirror x +mirror y apply_parallel "blur 3"

### 2.13.2 *apply_parallel_channels*

**Arguments:**

- "command"

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

*(eq. to `apc`)*
Example 569: `image.jpg apply_parallel_channels "blur 3"`

2.13.3 `apply_parallel_overlap`

Arguments:

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

Apply specified command on each of the selected images, by parallelizing it on `nb_threads` overlapped sub-images. (eq. to `apo`). `nb_threads` must be a power of 2.

Default values:

- `overlap=0`, `nb_threads=0`.
2.13. CONTROL FLOW

Example 570: image.jpg +apply_parallel_overlap *smooth 500,0,1", 1

2.13.4 apply_tiles

Arguments:

- "command", tile_width[%]>0, tile_height[%]>0, tile_depth[%]>0,
  overlap_width[%]>=0, overlap_height[%]>=0, overlap_depth[%]>=0,
  boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }

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

Default values:

- 'tile_width=tile_height=tile_depth=10%',
  'overlap_width=overlap_height=overlap_depth=0' and
  'boundary_conditions=1'.
Example 571: image.jpg +equalize[0] 256 +apply.tiles[0] "equalize 256",16,16,1,50%,50%

2.13.5 apply timeout

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

Apply a command with a timeout.
Set variable "$is_timeout" to '1' if timeout occurred, '0' otherwise.

Default value:
- 'timeout=20'.

2.13.6 check (+)

Arguments:
- condition

Evaluate specified condition and display an error message if evaluated to false.
If `expression` is not a math expression, it is regarded as a filename and checked if it exists.

2.13.7 check3d (+)

Arguments:
- _is_full_check={ 0 | 1 }
2.13. CONTROL FLOW

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

Default value:

- 'is_full_check=1'.

2.13.8 check_display

Check if a display is available, and throw an error otherwise.

2.13.9 continue (+)

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

2.13.10 break (+)

Break current ‘repeat...done’, ‘do...while’ or ‘local...endlocal’ block.
Example 573: `image.jpg` repeat 10 blur 1 if 1==1 break fi deform 10 done

### 2.13.11 do (+)

Start a `do...while` block.

Example 574: `image.jpg` luminance i=ia+2 do set 255,u(100),u(100) while ia<i
2.13.12  **done (+)**
End a `repeat/for...done` block, and go to associated `repeat/for` position, if iterations remain.

2.13.13  **elif (+)**
**Arguments:**
- **condition**

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 \( \{ 0=\text{false} \mid \text{other}=\text{true} \} \).

2.13.14  **else (+)**
Execute following commands if previous `if` or `elif` conditions failed.

2.13.15  **fi (+)**
End a `if...[elif]...[else]...fi` block.
(\textit{eq. to } `fi`).\textsl{n}.

2.13.16  **endlocal (+)**
End a `local...endlocal` block.
(\textit{eq. to } `endl`).\textsl{n}.

2.13.17  **error (+)**
**Arguments:**
- **message**

Print specified error message on the standard error (stderr) and exit interpreter, except if error is caught by a
`onfail` command.
Command selection (if any) stands for displayed call stack subset instead of image indices.

2.13.18  **eval (+)**
**Arguments:**
- **expression**

Evaluate specified math expression. - If no command selection is specified, the expression is evaluated once
and its result is set to status. - If command selection is specified, the evaluation is looped over selected images.
Status is not modified.
(in this latter case, `eval` is similar to `fill` without assigning the image values).
2.13.19  *exec* (+)

**Arguments:**

- `_is_verbose= { 0 | 1 }, "command"`

Execute external command using a system call.
The status value is then set to the error code returned by the system call.
If `is_verbose=1`, the executed command is allowed to output on stdout/stderr.
(`eq. to 'x').

**Default value:**

- `'is_verbose=1'`.

2.13.20  *for* (+)

**Arguments:**

- `condition`

Start a 'for...done' block.

```plaintext
Example 575: image.jpg resize2dy 32 400,400,1,3 x=0 for $x<400 image[1] [0],$x,$x x+=40 done
```

2.13.21  *if* (+)

**Arguments:**

- `condition`

Start a 'if...[elif]...[else]...fi' block and test if specified condition holds.
`'condition'` is a mathematical expression, whose evaluation is interpreted as `{ 0=false | other=true }`. 
Example 576: image.jpg if ia<64 add 50% elif ia<128 add 25% elif ia<192 sub 25% else sub 50% fi cut 0,255

2.13.22  local (+)
Start a 'local...[onfail]...endlocal' block, with selected images.
(eq. to '1').
Example 577: `image.jpg local[] 300,300,1,3 rand[0] 0,255 blur 4 sharpen 1000 endlocal`

Example 578: `image.jpg +local repeat 3 deform 20 done endlocal`

Tutorial page: https://gmic.eu/tutorial/_local.shtml

### 2.13.23 mutex (+)

**Arguments:**

- `index, action={0=unlock | 1=lock}`

Lock or unlock specified mutex for multi-threaded programming. A locked mutex can be unlocked only by the same thread. All mutexes are unlocked by default. `index` designates the mutex index, in [0,255].

**Default value:**

- `action=1`.

### 2.13.24 noarg (+)

Used in a custom command, `noarg` tells the command that its argument list have not been used finally, and so they must be evaluated next in the G’MIC pipeline, just as if the custom command takes no arguments at all.

Use this command to write a custom command which can decide if it takes arguments or not.
2.13. CONTROL FLOW

2.13.25  onfail (+)

Execute following commands when an error is encountered in the body of the ‘local...endlocal’ block. The status value is set with the corresponding error message.

Example 579: image.jpg +local blur -3 onfail mirror x endlocal

2.13.26  parallel (+)

Arguments:

- `_wait_threads","command1","command2",...

Execute specified commands in parallel, each in a different thread. Parallel threads share the list of images. 
’wait_threads’ can be { 0=when current environment ends | 1=immediately }.

Default value:

- ‘_wait_threads=1’. 
2.13.27  progress (+)

Arguments:

- \( 0 \leq \text{value} \leq 100 \)
- \(-1\)

Set the progress index of the current processing pipeline. This command is useful only when G'MIC is used by an embedding application.

2.13.28  quit (+)

Quit G'MIC interpreter. (eq. to 'q').

2.13.29  repeat (+)

Arguments:

- \( \text{nb\_iterations,\_variable\_name} \)

Start 'nb\_iterations' iterations of a 'repeat...done' block. 'nb\_iterations' is a mathematical expression that will be evaluated.
Example 581: `image.jpg split y repeat $!,n shift[$n] $<,0,0,0,2 done append y`

Example 582: `image.jpg model3d 2 repeat 4 imagecube3d rotate3d 1,1,0,40 snapshot3d 400,1.4 done`

Tutorial page:
https://gmic.eu/tutorial/_repeat.shtml
2.13.30  return (+)
Return from current custom command.

2.13.31  rprogress

Arguments:

- $0\leq\text{value}\leq100$ \mid -1 \mid "command", $0\leq\text{value}_{\text{min}}\leq100, 0\leq\text{value}_{\text{max}}\leq100$

Set the progress index of the current processing pipeline (relatively to previously defined progress bounds),
or call the specified command with specified progress bounds.

2.13.32  run

Arguments:

- "G'MIC pipeline"

Run specified G'MIC pipeline.
This is only useful when used from a shell, e.g. to avoid shell substitutions to happen in argument.

2.13.33  skip (+)

Arguments:

- item

Do nothing but skip specified item.

2.13.34  status (+)

Arguments:

- status_string

Set the current status. Used to define a returning value from a function.
(eq. to 'u').
2.14. ARRAYS, TILES AND FRAMES

Example 583: image.jpg command "foo: u0=Dark u1=Bright status $\{u\{ia>=128\}\}" text_outline 
$\{-foo\},2,23,2,1,255

2.13.35 while (+)

Arguments:

- condition

End a 'do...while' block and go back to associated 'do' if specified condition holds. 'condition' is a mathematical expression, whose evaluation is interpreted as \{ 0=false | other=true \}.

2.14 Arrays, Tiles and Frames

2.14.1 array

Arguments:

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

Create MxN array from selected images.

Default values:

- 'N=M' and 'expand_type=0'.

[Image of colored pencils]
CHAPTER 2. LIST OF COMMANDS

Example 584: image.jpg array 3,2,2

2.14.2  array_fade

Arguments:

- $M>0, N>0, 0 <= \text{fade}_\text{start} <= 100, 0 <= \text{fade}_\text{end} <= 100, \text{expand}_\text{type} = \{0=\text{min} \mid 1=\text{max} \mid 2=\text{all}\}$

Create $MxN$ array from selected images.

Default values:

- $'N=M', '\text{fade}_\text{start}=60', '\text{fade}_\text{end}=90$ and $'\text{expand}_\text{type}=1'$. 
2.14. **ARRAYS, TILES AND FRAMES**

2.14.3  

*array_mirror*

Arguments:

- \( N > 0, \text{dir}\{0\cdot x \mid 1\cdot y \mid 2\cdot xy \mid 3\cdot tri-xy\}, \text{expand_type}\{0 \mid 1\} \)

Create \(2^\text{N}\cdot x\cdot 2^\text{N}\) array from selected images.

Default values:

- ‘dir=2’ and ‘expand_type=0’.
2.14.4 *array_random*

**Arguments:**

- \( M_s > 0, N_s > 0, M_d > 0, N_d > 0 \)

Create \( M_d \times N_d \) array of tiles from selected \( M_s \times N_s \) source arrays.

**Default values:**

- ‘\( N_s = M_s \)’, ‘\( M_d = M_s \)’ and ‘\( N_d = N_s \)’.
2.14. ARRA YS, TILES AND FRAMES

Example 587: `image.jpg +array_random 8,8,15,10`

2.14.5 frame blur

Arguments:

- `_sharpness>0, _size>=0, _smoothness, _shading, _blur`

Draw RGBA-colored round frame in selected images.

Default values:

- ‘sharpness=10’, ‘size=30’, ‘smoothness=0’, ‘shading=1’ and ‘blur=3%’.
2.14.6  *frame_cube*

**Arguments:**

- \_depth\geq 0, \_centering\_x, \_centering\_y, \_left\_side=\{0=normal \mid 1=mirror-x \\
  2=mirror-\_y \mid 3=mirror-\_xy\}, \_right\_side, \_lower\_side, \_upper\_side

Insert 3D frames in selected images.

**Default values:**

- ‘\_depth=1’, ‘\_centering\_x=\_centering\_y=0’ and
  ‘\_left\_side=\_right\_side, \_lower\_side=\_upper\_side=0’.
2.14.7  frame_fuzzy

Arguments:

- \texttt{size\_x[^%]} >0, \texttt{size\_y[^%]} >0, \texttt{fuzzyness} >0, \texttt{smoothness[^%]} >0, \texttt{R, G, B, A}

Draw RGBA-colored fuzzy frame in selected images.

Default values:

- \texttt{\`size\_y=size\_x\', 'fuzzyness=5', 'smoothness=1' and 'R=G=B=A=255'}.  

2.14.8  frame_painting

Arguments:

- \texttt{size}=[0,1], \texttt{contrast}=0, \texttt{profile.smoothness}=0, \texttt{R}, \texttt{G}, \texttt{B},
- \texttt{vignette.size}=[0,1], \texttt{vignette.contrast}=0, \texttt{defects.contrast}=0,
- \texttt{defects.density}=[0,100], \texttt{defects.size}=0, \texttt{defects.smoothness}=0, \texttt{serial.number}=0

Add a painting frame to selected images.

Default values:

- \texttt{‘size=10’}, \texttt{‘contrast=0.4’}, \texttt{‘profile.smoothness=6’}, \texttt{‘R=225’}, \texttt{‘G=200’},
- \texttt{‘B=120’}, \texttt{‘vignette.size=2’}, \texttt{‘vignette.contrast=400’}, \texttt{‘defects.contrast=50’},
- \texttt{‘defects.density=10’}, \texttt{‘defects.size=1’}, \texttt{‘defects.smoothness=0.5’} and
- \texttt{‘serial.number=123456789’}.
2.14.9  *frame_pattern*

**Arguments:**

- \( M \geq 3, \_\text{constrain\_size}=\{0 \mid 1\} \)
- \( M \geq 3, \_\text{[frame\_image]}, \_\text{constrain\_size}=\{0 \mid 1\} \)

Insert selected pattern frame in selected images.

**Default values:**

- ‘pattern=0’ and ‘constrain\_size=0’.
2.14.10  frame round

Arguments:

- sharpness > 0, size > 0, smoothness, shading, R, G, B, A

Draw RGBA-colored round frame in selected images.

Default values:

- 'sharpness=10', 'size=10', 'smoothness=0', 'shading=0' and R=G=B=A=255.
2.14. ARRS, TILES AND FRAMES

2.14.11  \textit{frame} \textit{seamless}

**Arguments:**

- \texttt{frame} \texttt{size} > 0, \texttt{patch} \texttt{size} > 0, \texttt{blend} \texttt{size} > 0, \texttt{frame} \texttt{direction} = \{ 0 = \text{inner (preserve image size)} \mid 1 = \text{outer} \}

Insert frame in selected images, so that tiling the resulting image makes less visible seams.

**Default values:**

- \texttt{‘patch\_size=7’, ‘blend\_size=5’ and ‘frame\_direction=1’}.
Example 594: image.jpg +frame_seamless 30 array 2,2

2.14.12  \textit{frame}_x

Arguments:

- \textit{size}_x[/%], \textit{col1},...,\textit{colN}

Insert colored frame along the x-axis in selected images.

Default values:

- \textquote{\textit{col1}=col2=col3=255} and \textquote{\textit{col4}=255}.
2.14.13  \textit{frame}_xy

Arguments:

- \texttt{size\_x[\%],size\_y[\%],\texttt{col1,...,colN}}

Insert colored frame along the x-axis in selected images.

Default values:

- \texttt{’size\_y=size\_x’, ’\texttt{col1=col2=col3=255}’ and \texttt{col4=255’}.}

\textit{(eq. to ’frame’).}
2.14.14  \textit{frame} \_xyz

Arguments:

- \texttt{size} \_x[\%], \texttt{size} \_y[\%], \texttt{size} \_z[\%], \texttt{col}1,...,\texttt{col}N

Insert colored frame along the x-axis in selected images.

Default values:

- \texttt{‘size} \_y=\texttt{size} \_x=\texttt{size} \_z’, \texttt{‘col}1=\texttt{col}2=\texttt{col}3=255’ and \texttt{‘col}4=255’.

2.14.15  \textit{frame} \_y

Arguments:

- \texttt{size} \_y[\%], \texttt{col}1,...,\texttt{col}N

Insert colored frame along the y-axis in selected images.

Default values:

- \texttt{‘col}1=\texttt{col}2=\texttt{col}3=255’ and \texttt{‘col}4=255’.

Example 596: \texttt{image.jpg frame} \_xy 1,1,0 \texttt{frame} \_xy 20,10,255,0,255
2.14. ARRAYS, TILES AND FRAMES

2.14.16  \textit{img2ascii}

Arguments:

- \_charset, \_analysis\_scale>0, \_analysis\_smoothness[\%]>0, \_synthesis\_scale>0, \_output\_ascii\_filename

Render selected images as binary ascii art. This command returns the corresponding 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]’.
2.14.17  imagegrid

Arguments:

- \( M > 0, N > 0 \)

Create MxN image grid from selected images.

Default value:

- \(' N=M '\)
2.14.18  *imagegrid\_hexagonal*

**Arguments:**

- \_resolution\_0, 0 < \_outline\_0 < 1

Create hexagonal grids from selected images.

**Default values:**

- 'resolution=32', 'outline=0.1' and 'is\_antialiased=1'.
2.14.19  \texttt{imagegrid\_triangular}

Arguments:

- \texttt{pattern\_width>1, pattern\_height>1, pattern\_type, 0<=outline\_opacity<=1, outline\_color1, ...}

Create triangular grids from selected images.
`pattern type` can be \{ 0=horizontal | 1=vertical | 2=crossed | 3=cube | 4=decreasing | 5=increasing \}.

Default values:

- `pattern_width=24`, `pattern_height=pattern_width`, `pattern_type=0`, `outline_opacity=0.1` and `outline_color1=0`. 
2.14.20  *linearize files*

Arguments:

- \( M > 0, N > 0 \)

Linearize \( M \times N \) tiles on selected images.

**Default value:**

- \(' N=M' \)
2.14.21  \textit{map\_sprites}

Arguments:

- \texttt{\_nb\_sprites\>=1, allow\_rotation=\{} 0=none | 1=90 deg. | 2=180 deg. \}\}

Map set of sprites (defined as the \texttt{\_nb\_sprites}' latest images of the selection) to other selected images, according to the luminosity of their pixel values.
2.14. ARRAYS, TILES AND FRAMES

Example 603: image.jpg resize2dy 48 repeat 16 ball \{(8+2*x>)\},\{-RGB\} mul\{-1\} \{(1+x>)\}/16 \} done
map\_sprites 16

2.14.22 pack

Arguments:
- is\_ratio\_constraint\{ 0 | 1 \}, sort\_criterion

Pack selected images into a single image.
The returned status contains the list of new (x,y) offsets for each input image.
Parameter 'is\_ratio\_constraint' tells if the resulting image must tend to a square image.

Default values:
- 'is\_ratio\_constraint=0' and 'sort\_criterion=max(w,h)'.

Example 604: image.jpg repeat 10 +resize2dx\[-1\] 75% balance\_gamma\[-1\] \{-RGB\} done pack 0
2.14.23  puzzle

Arguments:

- \textit{width}>0, \textit{height}>0, M=1, N=1, \textit{curvature}, \textit{centering}, \textit{connectors\_variability}, \textit{resolution}>1

Input puzzle binary mask with specified size and geometry.

Default values:

- '\textit{width=height=512}', '\textit{M=5}', '\textit{curvature=0.5}', '\textit{centering=0.5}', '\textit{connectors\_variability=0.5}' and '\textit{resolution=64}'.

Example 605: \texttt{puzzle},

2.14.24  quadratize tiles

Arguments:

- \textit{M}>0, \textit{N}>0

Quadratize MxN tiles on selected images.

Default value:

- '\textit{N=M}'.

2.14.25 \textit{rotate tiles}

Arguments:

- \texttt{angle, M>0, N>0}

Apply MxN tiled-rotation effect on selected images.

Default values:

- \texttt{‘M=8’ and ‘N=M’}.
2.14.26 *shift_tiles*

**Arguments:**

- $M>0, N>0, \text{amplitude}$

Apply MxN tiled-shift effect on selected images.

**Default values:**

- ‘$N=M$’ and ‘amplitude=20’.
2.14.27 taquin

Arguments:

- $M>0$, $N>0$, remove_tile=\{ 0=none | 1=first | 2=last | 3=random \}, relief, border_thickness[\%], border_outline[\%], outline_color

Create MxN taquin puzzle from selected images.

Default value:

- ‘N=M’, ‘relief=50’, ‘border_thickness=5’, ‘border_outline=0’ and ‘remove_tile=0’.
CHAPTER 2. LIST OF COMMANDS

Example 609: image.jpg +taquin 8

2.14.28 \textit{tunnel}

Arguments:

\begin{itemize}
  \item \texttt{level}>0, \texttt{factor}>0, \texttt{centering}_x, \texttt{centering}_y, \texttt{opacity}, \texttt{angle}
\end{itemize}

Apply tunnel effect on selected images.

Default values:

\begin{itemize}
  \item \texttt{level=9}, \texttt{factor=80\%}, \texttt{centering}_x=\texttt{centering}_y=0.5, \texttt{opacity=1} \texttt{and} \texttt{angle=0}
\end{itemize}
2.15  Artistic

2.15.1  boxfitting

Arguments:

- \( \text{min\_box\_size} > 1 \), \( \text{max\_box\_size} = 0 \), \( \text{initial\_density} = 0 \), \( \text{nb\_attempts} = 1 \)

Apply box fitting effect on selected images, as displayed the web page: [http://www.complexification.net/gallery/machines/boxFittingImg/]

Default values:

- ‘\text{min\_box\_size}=1’, ‘\text{max\_box\_size}=0’, ‘\text{initial\_density}=0.1’ and ‘\text{nb\_attempts}=3’.
2.15.2  brushify

Arguments:

- \([\text{brush}], \text{brush_nb_sizes}=3, \text{brush_min_size_factor}=0.66, \text{brush_nb_orientations}=12, \text{brush_light_type}=0, \text{brush_light_strength}=0.25, \text{brush_opacity}=0.8, \text{painting_density}=20\%, \text{painting_contours_coherence}=0.9, \text{painting_coherence_alpha}=1, \text{painting_coherence_sigma}=1, \text{painting_primary_angle}=0, \text{painting_angle_dispersion}=0.2\]

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

2.15.3 cartoon

Arguments:

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

Apply cartoon effect on selected images.

Default values:

- ‘smoothness=3’, ‘sharpening=150’, ‘threshold=20’, ‘thickness=0.25’, ‘color=1.5’ and ‘quantization=8’.
2.15.4 \textit{color\_ellipses}

\textbf{Arguments:}

- \texttt{\_count>0, \_radius>=0, \_opacity>=0}

Add random color ellipses to selected images.

\textbf{Default values:}

- \texttt{‘count=400’, ‘radius=5’ and ‘opacity=0.1’}.
Example 614: image.jpg +color_ellipses , , 0.15

2.15.5 **cubism**

**Arguments:**

- \( \text{density} > 0, \text{thickness} < 50, \text{max\_angle}, \text{opacity}, \text{smoothness} > 0 \)

Apply cubism effect on selected images.

**Default values:**

- 'density=50', 'thickness=10', 'max\_angle=75', 'opacity=0.7' and 'smoothness=0'.

2.15.6 \textit{draw\_whirl}

Arguments:

- \texttt{amplitude}>0

Apply whirl drawing effect on selected images.

Default value:

- \texttt{‘amplitude=100’}.
2.15.7 \textit{drawing}

Arguments:

- \texttt{amplitude} \geq 0

Apply drawing effect on selected images.

Default value:

- \texttt{amplitude=200}.
2.15.8  drop shadow

Arguments:

- _offset_x[%], _offset_y[%, _smoothness[%]]>0, 0<-_curvature<_1, _expand_size={ 0 | 1 }

Drop shadow behind selected images.

Default values:

- 'offset_x=20', 'offset_y=offset_x', 'smoothness=5', 'curvature=0' and 'expand_size=1'.
2.15. ARTISTIC

Example 618: image.jpg drop_shadow 10,20,5,0.5 expand_xy 20,0 display_rgba

2.15.9 ellipsionism

Arguments:

- \( R > 0\%\), \( r > 0\%\), \( \text{smoothness} > 0\%\), \( \text{opacity} \), \( \text{outline} > 0 \), \( \text{density} > 0 \)

Apply ellipsionism filter to selected images.

Default values:

- ‘\( R=10\%\)’, ‘\( r=3\%\)’, ‘\( \text{smoothness}=1\%\)’, ‘\( \text{opacity}=0.7\)’, ‘\( \text{outline}=8\)’ and ‘\( \text{density}=0.6\)’.
2.15.10  fire_edges

Arguments:

- `edges>=0,0<=attenuation<1,smoothness>=0,threshold>=0,nb_frames>0,starting_frame>=0,frame_skip>=0`

Generate fire effect from edges of selected images.

Default values:

- `’edges=0.7’, ’attenuation=0.25’, ’smoothness=0.5’, ’threshold=25’, ’nb_frames=1’, ’starting_frame=20’ and ’frame_skip=0’.`
2.15.11 \textit{fractalize}

Arguments:

\begin{itemize}
  \item $0 \leq \text{detail\_level} \leq 1$
\end{itemize}

Randomly fractalize selected images.

Default value:

\begin{itemize}
  \item \texttt{‘detail\_level=0.8’}
\end{itemize}
Example 621: `image.jpg fractalize`

2.15.12  **glow**

**Arguments:**

- `amplitude>=0`

Add soft glow on selected images.

**Default value:**

- `'amplitude=1%'`
2.15.13 halftone

Arguments:

- \texttt{nb\_levels} > 2, \texttt{size\_dark} = 2, \texttt{size\_bright} > 2, \texttt{shape} = \{ 0 = \text{square} | 1 = \text{diamond} | 2 = \text{circle} | 3 = \text{inv-square} | 4 = \text{inv-diamond} | 5 = \text{inv-circle} \}, \texttt{smoothness} \% > 0

Apply halftone dithering to selected images.

Default values:

- \texttt{‘nb\_levels=5’, ‘size\_dark=8’, ‘size\_bright=8’, ‘shape=5’ and ‘smoothness=0’.}
2.15.14  **hardsketchbw**

Arguments:

- \_amplitude\_>0, \_density\_>0, \_opacity\_0<\_edge\_threshold\_<100, \_is\_fast\_=\{ 0 | 1 \}

Apply hard B&W sketch effect on selected images.

**Default values:**

- ‘amplitude=1000’, ‘sampling=3’, ‘opacity=0.1’, ‘edge_threshold=20’ and ‘is_fast=0’.
Example 624: `image.jpg +hardsketchbw 200,70,0.1,10 median[-1] 2 +local reverse blur[-1] 3 blend[-2,-1] overlay endlocal`

### 2.15.15 hearts

**Arguments:**

- `_density>=0`

Apply heart effect on selected images.

**Default value:**

- `'density=10'`. 

2.15.16 \textit{houghsketchbw}

Arguments:

- $\_density>0$, $\_radius>0$, $0<=\_threshold<=100$, $0<=\_opacity<=1$, $\_votesize[]>0$

Apply hough B\&W sketch effect on selected images.

Default values:

- ‘density=8’, ‘radius=5’, ‘threshold=80’, ‘opacity=0.1’ and ‘votesize=100%’. 
2.15.17  \textit{lightrays}

Arguments:

\begin{itemize}
  \item $100 \leq \text{density} \leq 0$, \texttt{center\_x[\%]}, \texttt{center\_y[\%]}, \texttt{ray\_length} \geq 0, \texttt{ray\_attenuation} \geq 0$
\end{itemize}

Generate ray lights from the edges of selected images.

Defaults values: ‘density=50\%’, ‘center\_x=50\%’, ‘center\_y=50\%’, ‘ray\_length=0.9’ and ‘ray\_attenuation=0.5’.
2.15.18 \textit{light\_relief}

\textbf{Arguments:}

- \texttt{ambient\_light}, \texttt{specular\_lightness}, \texttt{specular\_size}, \texttt{darkness}, \texttt{light\_smoothness}, \texttt{x1}, \texttt{y1}, \texttt{z1}, \texttt{zscale}, \texttt{opacity\_is\_heightmap}=\{0 \mid 1\}

Apply relief light to selected images.

\textbf{Default values(s):}

- \texttt{’ambient\_light=0.3’}, \texttt{’specular\_lightness=0.5’}, \texttt{’specular\_size=0.2’}, \texttt{’darkness=0’}, \texttt{’x1=0.2’}, \texttt{’y1=zl=0.5’},

- \texttt{’zscale=1’}, \texttt{’opacity=1’} and \texttt{’opacity\_is\_heightmap=0’}.
2.15.19 **linify**

**Arguments:**

- \(0 < \_\text{density} < 100, \_\text{spreading} > 0, \_\text{resolution}[] > 0, \_\text{line\_opacity} > 0, \_\text{line\_precision} > 0, \_\text{mode} = \{ 0 = \text{subtractive} ~ | ~ 1 = \text{additive} \} \)

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'.
2.15.20 *mosaic*

Arguments:

- $0 \leq \text{density} \leq 100$

Create random mosaic from selected images.

Default values:

- ‘density=30’.
2.15.21  *old_photo*

Apply old photo effect on selected images.
2.15.22  pencilbw

Arguments:

- \_size\_>=0, \_amplitude\_>=0

Apply B&W pencil effect on selected images.

Default values:

- ‘size=0.3’ and ‘amplitude=60’.

Example 632: `image.jpg pencilbw`

2.15.23  pixelsort

Arguments:

- \_ordering\_={+ | -}, \_axis\_={x | y | z | xy | yx}, \_sorting\_criterion\_={mask}

Apply a ‘pixel sorting’ algorithm on selected images, as described in the page: http://satyarth.me/articles/pixel-sorting/

Default values:

- ‘ordering=+’, ‘axis=x’ and ‘sorting\_criterion\_mask’=(undefined)’.
2.15. **ARTISTIC**

Example 633: `image.jpg +norm +ge[-1] 30% +pixelsort[0] +,y,[1],[2]`

2.15.24 **polaroid**

**Arguments:**

- `_size1>=0, _size2>=0`

Create polaroid effect in selected images.

**Default values:**

- `'size1=10' and 'size2=20'`.  

Example 634: `image.jpg` to_rgb polaroid 5,30 rotate 20 drop_shadow ,

2.15.25  *polygonize*

**Arguments:**

- `warp_amplitude>=0, smoothness[%]>=0, min_area[%]>0, resolution_x[%]>0, resolution_y[%]>0`

Apply polygon effect on selected images.

**Default values:**

- `warp_amplitude=300`, `smoothness=2%`, `min_area=0.1%`, `resolution_x=resolution_y=10%`. 
2.15.26  *poster_edges*

**Arguments:**

- $0 < \text{edge\_threshold} < 100$, $0 < \text{edge\_shade} < 100$, $\text{edge\_thickness} > 0$, $\text{edge\_antialiasing} > 0$, $0 < \text{posterization\_level} < 15$, $\text{posterization\_antialiasing} > 0$

Apply poster edges effect on selected images.

**Default values:**

- ‘edge\_threshold=40’, ‘edge\_shade=5’, ‘edge\_thickness=0.5’, ‘edge\_antialiasing=10’, ‘posterization\_level=12’ and ‘posterization\_antialiasing=0’.
2.15.27 \textit{poster\_hope}

\textbf{Arguments:}

\begin{itemize}
  \item \texttt{smoothness} \geq 0
\end{itemize}

Apply Hope stencil poster effect on selected images.

\textbf{Default value:}

\begin{itemize}
  \item \texttt{smoothness=3}.
\end{itemize}
2.15.28 rodilius

Arguments:

- $0 \leq \text{amplitude} \leq 100$, $0 \leq \text{thickness} \leq 100$, $\text{sharpness}>0$, $\text{nb_orientations}>0$, $\text{offset}$, $\text{color\_mode}=\{0=\text{darker} \mid 1=\text{brighter}\}$

Apply rodilius (fractalius-like) filter on selected images.

Default values:

- ‘amplitude=10’, ‘thickness=10’, ‘sharpness=400’, ‘nb_orientations=7’, ‘offset=0’ and ‘color\_mode=1’.
Example 638: `image.jpg rodilius 12,10,300,10 normalize_local 10,6`

Example 639: `image.jpg normalize_local 10,16 rodilius 10,4,400,16 smooth 60,0,1,1,4 normalize_local 10,16`
2.15.29 sketchbw

Arguments:

- \_nb\_angles > 0, \_start\_angle, \_angle\_range \geq 0, \_length \geq 0, \_threshold \geq 0, \_opacity, \_bg\_factor \geq 0, \_density \geq 0, \_sharpness \geq 0, \_anisotropy \geq 0, \_smoothness \geq 0, \_coherence \geq 0, \_is\_boost \in \{0 \mid 1\}, \_is\_curved \in \{0 \mid 1\}

Apply sketch effect to selected images.

**Default values:**

- ‘nb\_angles=2’, ‘start\_angle=45’, ‘angle\_range=180’, ‘length=30’, ‘threshold=3’, ‘opacity=0.03’, ‘bg\_factor=0’, ‘density=0.6’, ‘sharpness=0.1’, ‘anisotropy=0.6’, ‘smoothness=0.25’, ‘coherence=1’, ‘is\_boost=0’ and ‘is\_curved=1’.

Example 640: image.jpg +sketchbw 1 reverse blur[-1] 3 blend[-2,-1] overlay

2.15.30 sponge

Arguments:

- \_size > 0

Apply sponge effect on selected images.

**Default value:**

- ‘size=13’.
2.15.31  *stained_glass*

Arguments:

- \(\text{edges}\% > 0, \text{shading} > 0, \text{is\_thin\_separators} = \{0 \mid 1\}\)

Generate stained glass from selected images.

Default values:

- \('\text{edges}=40\% ', '\text{shading}=0.2' \text{ and } '\text{is\_precise}=0'\).
2.15. ARTISTIC

Example 642: image.jpg stained glass 20%, 0.1

2.15.32 stars

Arguments:

- \_density[%]\geq 0, \_depth\geq 0, \_size\geq 0, \_nb\_branches\geq 1, 0\leq \_thickness\leq 1, \\
  \_smoothness[%]\geq 0, R, G, B, \_opacity

Add random stars to selected images.

Default values:

- 'density=10%', 'depth=1', 'size=32', 'nb\_branches=5', 'thickness=0.38', \\
  'smoothness=0.5', 'R=G=B=200' and 'opacity=1'.

2.15.33  *stencil*

Arguments:

- \(_\text{radius}[%]\geq0\), \(_\text{smoothness}>=0\), \(_\text{iterations}>=0\)

Apply stencil filter on selected images.

Default values:

- \(\text{‘radius}=3\), ‘smoothness=1’ and ‘iterations=8’.\)
2.15.34 stencilbw

Arguments:

- \texttt{edges>0, smoothness>0}

Apply B&W stencil effect on selected images.

Default values:

- \texttt{edges=15} and \texttt{smoothness=10}.
Example 645: image.jpg +stencilbw 40,4

2.15.35 stylize

Arguments:

- [style_image], _fidelity_finest, _fidelity_coarsest, _fidelity_smoothness_finest>0, _fidelity_smoothness_coarsest>=0, _fidelity_chroma>1, _init_type, _init_resolution>0, init_max_gradient>0, patchsize_analysis>0, patchsize_synthesis>0, patchsize_synthesis_final>0, nb_matches_finest>0, nb_matches_coarsest>0, penalize_repetitions>0, matching_precision>0, scale_factor>1, skip_finest_scales>0, "image_matching_command"

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 }.

2.15.36 tetris

Arguments:

- _scale>0

Apply tetris effect on selected images.

Default value:

- ‘scale=10’.
2.15. ARTISTIC

Example 646: image.jpg +tetris 10

2.15.37 warhol

Arguments:

- $M > 0$, $N > 0$, smoothness $= 0$, color $= 0$

Create MxN Andy Warhol-like artwork from selected images.

Default values:

2.15.38  **weave**

**Arguments:**

- \(\_\text{density}\_\cdot\_\_\text{thickness}\_\cdot\_\_\text{shadow}\_\cdot\_\_\text{shading}\_\cdot\_\_\text{fibers\_amplitude}\_\cdot\_\_\text{fibers\_smoothness}\_\cdot\_\_\text{angle}\_\cdot\_\_\text{x\_curvature}\_\cdot\_\_\text{y\_curvature}\) = 0

Apply weave effect to the selected images.

'angle' can be {0=0 deg. | 1=22.5 deg. | 2=45 deg. | 3=67.5 deg.}.

**Default values:**

- \text{‘density=6’, ‘thickness=65’, ‘shadow=40’, ‘shading=0.5’, ‘fibers\_amplitude=0’, ‘fibers\_smoothness=0’, ‘angle=0’ and ‘curvature\_x=curvature\_y=0’}
2.15.39 *whirls*

Arguments:

- `_texture>=0, _smoothness>=0, _darkness>=0, _lightness>=0`

Add random whirl texture to selected images.

Default values:

- `'texture=3’, ‘smoothness=6’, ‘darkness=0.5’ and ‘lightness=1.8’.`
2.16 Warpings

2.16.1 deform

Arguments:

• _amplitude_>0, _interpolation_

Apply random smooth deformation on selected images. 'interpolation' can be { 0=none | 1=linear | 2=bicubic }.

Default value:

• 'amplitude=10'.
2.16. WARPINGS

Example 650 : image.jpg +deform[0] 10 +deform[0] 20

2.16.2 euclidean2polar

Arguments:

- \(_{center\_x}\), \(_{center\_y}\), \(_{stretch\_factor}\)>0, \(_{boundary\_conditions}\)=\{ 0=dirichlet
  | 1=neumann | 2=periodic | 3=mirror \}

Apply euclidean to polar transform on selected images.

Default values:

- ‘center\_x=center\_y=50\%’, ‘stretch\_factor=1’ and ‘boundary\_conditions=1’.
2.16.3  *equirectangular2nadirzenith*

Transform selected equirectangular images to nadir/zenith rectilinear projections.

2.16.4  *fisheye*

**Arguments:**

- \( _\text{center}_x, _\text{center}_y, 0 < _\text{radius} < 100, _\text{amplitude} > 0 \)

Apply fish-eye deformation on selected images.

**Default values:**

- 'x=y=50', 'radius=50' and 'amplitude=1.2'.
2.16. **WARPINGS**

Example 652: image.jpg +fisheye

2.16.5 **flower**

Arguments:

- \_amplitude, \_frequency, \_offset_r[%], \_angle, \_center_x[%], \_center_y[%], \_boundary_conditions={0=dirichlet | 1=neumann | 2=periodic | 3=mirror}

Apply flower deformation on selected images.

Default values:

- 'amplitude=30', 'frequency=6', 'offset_r=0', 'angle=0', 'center_x=center_y=50%' and 'boundary_conditions=3'.
2.16.6 *kaleidoscope*

**Arguments:**

- `_center_x[%]`, `_center_y[%]`, `_radius`, `_angle`, `_boundary_conditions`={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }

Create kaleidoscope effect from selected images.

**Default values:**

- `'center_x=center_y=50%`, `'radius=100`, `'angle=30` and `'boundary_conditions=3`. ```
2.16.7 \textit{map\_sphere}

Arguments:

\begin{itemize}
\item width\textgreater{}0, height\textgreater{}0, radius, dilation\textgreater{}0, fading\textgreater{}0, fading\_power\textgreater{}0
\end{itemize}

Map selected images on a sphere.

Default values:

\begin{itemize}
\item width=height=512', 'radius=100', 'dilation=0.5', 'fading=0' and 'fading\_power=0.5'.
\end{itemize}
2.16.8  *nadirzenith2equirectangular*

Transform selected nadir/zenith rectilinear projections to equirectangular images.

2.16.9  *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} \}

Apply euclidean to polar transform on selected images.

**Default values:**

- \('\text{center}_x=\text{center}_y=50\%\', \ '\text{stretch}\_factor=1\' \ and \ '\text{boundary}\_conditions=1'.\)
2.16. WARPINGS

Example 656: image.jpg +euclidean2polar,

2.16.10 raindrops

Arguments:

- amplitude, density ≥ 0, wavelength ≥ 0, merging steps ≥ 0

Apply raindrops deformation on selected images.

Default values:

- 'amplitude=80', 'density=0.1', 'wavelength=1' and 'merging steps=0'.
2.16.11  *ripple*

**Arguments:**

- \_amplitude, \_bandwidth, \_shape={ 0=bloc | 1=triangle | 2=sine | 3=sine+ | 4=random }, \_angle, \_offset

Apply ripple deformation on selected images.

**Default values:**

- ‘amplitude=10’, ‘bandwidth=10’, ‘shape=2’, ‘angle=0’ and ‘offset=0’.
2.16. WARPINGS

2.16.12 rotidoscope

Arguments:

- `center.x[%], center.y[%], tiles>0, smoothness[%]>=0, boundary_conditions={0=dirichlet | 1=neumann | 2=periodic | 3=mirror}

Create rotational kaleidoscope effect from selected images.

Default values:

- ‘center.x=center.y=50′, ‘tiles=10′, ‘smoothness=1’ and ‘boundary_conditions=3′.
2.16.13  *spherize*

**Arguments:**

- \_radius[\%]\geq0, \_strength, \_smoothness[\%]\geq0, \_center\_x[\%], \_center\_y[\%], \_ratio\_x/y\geq0, \_angle, \_interpolation

Apply spherize effect on selected images.

**Default values:**

- ‘radius=50\%', ‘strength=1', ‘smoothness=0', ‘center\_x=\_center\_y=50\%', ‘ratio\_x/y=1', ‘angle=0' and ‘interpolation=1'.
2.16. WARPINGS

Example 660: image.jpg grid 5%,5%,0,0,0.6,255 spherize,

2.16.14 symmetrize

Arguments:

- \(x\%\), \(y\%\), \(angle\), \(boundary\_conditions=\{0=dirichlet \mid 1=neumann \mid 2=periodic \mid 3=mirror\}\), \(is\_antisymmetry=\{0 \mid 1\}\), \(swap\_sides=\{0 \mid 1\}\)

Symmetrize selected images regarding specified axis.

Default values:

- \('x=y=50\%', 'angle=90', 'boundary\_conditions=3', 'is\_antisymmetry=0' and 'swap\_sides=0'.\)
Example 661: image.jpg +symmetrize 50%,50%,45 +symmetrize[-1] 50%,50%,-45

2.16.15 **transform.polar**

**Arguments:**

- "expr_radius", "expr_angle", center_x[%], center_y[%], boundary_conditions={0=dirichlet | 1=neumann}

Apply user-defined transform on polar representation of selected images.

**Default values:**

- 'expr_radius=R-r', 'expr_angle=a', 'center_x=center_y=50%' and 'boundary_conditions=1'.
2.16. WARPINGS

Example 662: image.jpg +transform_polar[0] R*(r/R)^2, a +transform_polar[0] r, 2*a

2.16.16 **twirl**

**Arguments:**

- **amplitude**, **center_x[%]**, **center_y[%]**, **boundary_conditions**={ 0=dirichlet, 1=neumann, 2=periodic, 3=mirror }

Apply twirl deformation on selected images.

**Default values:**

- ‘amplitude=1’, ‘center_x=center_y=50%’ and ‘boundary_conditions=3’.
2.16.17  \textit{warp\_perspective}

\textbf{Arguments:}

\begin{itemize}
  \item \texttt{x-angle, y-angle, zoom>0}, \texttt{x-center, y-center, boundary_conditions=}
    \{0=\texttt{dirichlet} | 1=\texttt{neumann} | 2=\texttt{periodic} | 3=\texttt{mirror}\}
\end{itemize}

Warp selected images with perspective deformation.

\textbf{Default values:}

\begin{itemize}
  \item \texttt{x-angle=1.5}, \texttt{y-angle=0}, \texttt{zoom=1}, \texttt{x-center=y-center=50} and
    \texttt{boundary_conditions=2}.
\end{itemize}
2.16. WARPINGS

Example 664: `image.jpg warp_perspective`

2.16.18 water

Arguments:

- `amplitude, smoothness>=0, angle`

Apply water deformation on selected images.

Default values:

- `amplitude=30`, `smoothness=1.5` and `angle=45`. 
2.16.19 \textit{wave}

Arguments:

- \texttt{amplitude} > 0, \texttt{frequency} > 0, \texttt{center.x}, \texttt{center.y}

Apply wave deformation on selected images.

Default values:

- \texttt{amplitude=4’, frequency=0.4’ and center.x=center.y=50’}. 
2.16.20  wind

Arguments:

- \texttt{amplitude}>0, \texttt{angle}=0<=\texttt{attenuation}=1, \texttt{threshold}

Apply wind effect on selected images.

Default values:

- \texttt{amplitude=20', 'angle=0', 'attenuation=0.7' and 'threshold=20'}. 
CHAPTER 2. LIST OF COMMANDS

Example 667: image.jpg +wind ,

[Image: image.jpg (480x320lx)]

2.16.21 **zoom**

Arguments:

- _factor, cx, cy, cz, boundary_conditions={ 0=dirichlet | 1=neumann | 2=periodic | 3=mirror }

Apply zoom factor to selected images.

Default values:

- ‘factor=1’, ‘cx=cy=cz=0.5’ and ‘boundary_conditions=0’.
2.17. Degradations

2.17.1. cracks

Arguments:

- \(0 \leq \text{density} \leq 100\), \text{is\_relief}\{0 \mid 1\}, \text{opacity}, \text{color1}, ...

Draw random cracks on selected images with specified color.

Default values:

- \(\text{density}=25\), \(\text{is\_relief}=0\), \(\text{opacity}=1\) and \(\text{color1}=0\).
2.17.2  \textit{light\_patch}

Arguments:

- \texttt{density>0, darkness>0, lightness>0}

Add light patches to selected images.

Default values:

- \texttt{‘density=10’, ‘darkness=0.9’ and ‘lightness=1.7’}. 
2.17.3  \textit{noise hurl}

\textbf{Arguments:}

\begin{itemize}
  \item $\_\text{amplitude} \geq 0$
\end{itemize}

Add hurl noise to selected images.

\textbf{Default value:}

\begin{itemize}
  \item `amplitude=10'.
\end{itemize}
2.17.4  **pixelize**

Arguments:

- \( \text{scale}_x > 0, \text{scale}_y > 0, \text{scale}_z > 0 \)

Pixelize selected images with specified scales.

Default values:

- ‘\( \text{scale}_x = 20 \)’ and ‘\( \text{scale}_y = \text{scale}_z = \text{scale}_x \)’. 
2.17.5  **scanlines**

**Arguments:**

- \_amplitude, \_bandwidth, \_shape={ 0=bloc | 1=triangle | 2=sine | 3=sine+ | 4=random }, \_angle, \_offset

Apply ripple deformation on selected images.

**Default values:**

- ‘amplitude=60’, ‘bandwidth=2’, ‘shape=0’, ‘angle=0’ and ‘offset=0’.
2.17.6 shade_stripe

Arguments:

- frequency>=0, direction={0=horizontal | 1=vertical}, darkness>=0, lightness>=0

Add shade stripes to selected images.

Default values:

- ‘frequency=5’, ‘direction=1’, ‘darkness=0.8’ and ‘lightness=2’.
2.17.7 *shadow_patch*

Arguments:

- `opacity>=0`

Add shadow patches to selected images.

**Default value:**

- `'opacity=0.7'`.
2.17.8 spread

Arguments:

- \( dx > 0, \ dy > 0, \ dz > 0 \)

Spread pixel values of selected images randomly along x, y and z.

Default values:

- ‘dx=3’, ‘dy=dx’ and ‘dz=0’.
2.17.9 **stripes y**

Arguments:

- \_frequency\_>=0

Add vertical stripes to selected images.

Default value:

- ‘frequency=10’. 
2.17.10  *texturize_canvas*

**Arguments:**

- \(_{\text{amplitude}}=0\), \(_{\text{fibrousness}}=0\), \(_{\text{emboss\ level}}=0\)

Add paint canvas texture to selected images.

**Default values:**

- ‘amplitude=20’, ‘fibrousness=3’ and ‘emboss\_level=0.6’.
2.17.11 \textit{texturize\_paper}

Add paper texture to selected images.
2.17.12 vignette

Arguments:

- \(\text{strength}>0,0<\text{radius\_min}<100,0<\text{radius\_max}<100\)

Add vignette effect to selected images.

Default values:

- \('\text{strength}=100', '\text{radius\_min}=70' \text{and} '\text{radius\_max}=90'\).

Example 680: `image.jpg vignette`

2.17.13 watermark_visible

Arguments:

- \(\text{text,0<\text{opacity}<1,\text{size}>0,\text{angle, mode}={0=remove | 1=add},\text{smoothness}>0}\)

Add or remove a visible watermark on selected images (value range must be [0,255]).

Default values:

- \('\text{text}=(c) \text{G'MIC}', '\text{opacity}=0.3', '\text{size}=53', '\text{angle}=25', '\text{mode}=1' \text{and} '\text{smoothness}=0'\).
2.18 Blending and Fading

2.18.1 blend

Arguments:

- {layer,blending_mode,opacity[%],selection_is={ 0-base-layers | 1-top-layers }
  - blending_mode,opacity[%]

Blend selected G,GA,RGB or RGBA images by specified layer or blend all selected images together, using specified blending mode. 
'blending_mode' can be { add | alpha | and | average | blue | burn | darken | difference | divide 
| dodge | edges | exclusion | freeze | grainextract | grainmerge | green | hardlight | hardmix | hue 
| interpolation | lighten | lightness | linearburn | linearlight | luminance | multiply | negation | or 
| overlay | pinlight | red | reflect | saturation | seamless | seamless_mixed | screen | shapeareamax 
| shapeareamx0 | shapeareamin | shapeareamin0 | shapeaverage | shapeaverage0 | shapemedian 
| shapemedian0 | shapemin | shapemin0 | shapemax | shapemax0 | softburn | softdodge | softlight 
| stamp | subtract | value | vividlight | xor }

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

Default values:

- ‘blending_mode=alpha’, ‘opacity=1’ and ‘selection_is=0’.
Example 682: `image.jpg +drop,shadow, resize2dy[-1] 200 rotate[-1] 20 +blend alpha display_rgba[-2]`

Example 683: `image.jpg testimage2d {w},{h} blend overlay`
2.18. BLENDING AND FADING

Example 684: 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 685: 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 686: 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 687: 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 688: 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 689: command "ex : $"*"arg repeat $"*"% +blend[0,1] $\{arg\{\$+1\}\} text_outline[-1] Mode:\" $\{arg\{\$+1\}\},2,2,23,2,1,255 done\" image.jpg testimage2d \{w\},\{h\} ex softdodge,softlight,stamp,subtract,value,vividlight,xor
2.18.2  \textit{blend\_edges}

\textbf{Arguments:}

- \textit{smoothness\%} $> 0$

Blend selected images together using ‘edges’ mode.

Example 690: image.jpg testimage2d \{w\},\{h\} +\textit{blend\_edges} 0.8

2.18.3  \textit{blend\_fade}

\textbf{Arguments:}

- \textit{[fading\_shape]}

Blend selected images together using specified fading shape.
2.18. BLENDING AND FADING

Example 691: `image.jpg testimage2d \{w\},\{h\} 100\%,100\%,1,1,'cos(y/10)' normalize[-1] 0,1 +blend,fade[0,1] [2]

2.18.4 blend_median

Blend selected images together using 'median' mode.

Example 692: `image.jpg testimage2d \{w\},\{h\} +mirror[0] y +blend_median`
2.18.5  **blend_seamless**

Arguments:

- \_is\_mixed\_mode=\{ 0 | 1 \}, \_inner\_fading(\%)\geq0, \_outer\_fading(\%)\geq0

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

Default values:

- ‘is\_mixed=0’, ‘inner\_fading=0’ and ‘outer\_fading=100\%’.

2.18.6  **fade\_diamond**

Arguments:

- 0<\_start\leq100, 0<\_end\leq100

Create diamond fading from selected images.

Default values:

- ‘start=80’ and ‘end=90’.

Example 693: image.jpg testimage2d \{w\},\{h\} + fade\_diamond 80,85

2.18.7  **fade\_linear**

Arguments:

- \_angle, 0<\_start\leq100, 0<\_end\leq100
Create linear fading from selected images.

**Default values:**

- ‘angle=45’, ‘start=30’ and ‘end=70’.

```
Example 694: image.jpg testimage2d {w},{h} +fade_linear 45,48,52
```

### 2.18.8 *fade_radial*

**Arguments:**

- $0<=\text{start}<=100, 0<=\text{end}<=100$

Create radial fading from selected images.

**Default values:**

- ‘start=30’ and ‘end=70’.
Example 695: `image.jpg testimage2d {w},{h} +fade_radial 30,70`

2.18.9  **fade x**

Arguments:

- \(0 \leq \text{start} \leq 100, 0 \leq \text{end} \leq 100\)

Create horizontal fading from selected images.

Default values:

- ‘\text{start}=30’ and ‘\text{end}=70’.
2.18. BLENDING AND FADING

Example 696: image.jpg testimage2d {w},{h} +fade_x 30,70

2.18.10 fade_y

Arguments:

- $0 \leq \text{start} \leq 100, 0 \leq \text{end} \leq 100$

Create vertical fading from selected images.

Default values:

- ‘start=30’ and ‘end=70’.
2.18.11  \textit{fade} \_z

\textbf{Arguments:}

- \(0 \leq \text{start} \leq 100, 0 \leq \text{end} \leq 100\)

Create transversal fading from selected images.

\textbf{Default values:}

- ‘\text{start}=30’ and ‘\text{end}=70’.

2.18.12  \textit{sub\_alpha}

\textbf{Arguments:}

- [base\_image], \texttt{opacity\_gain}=1

Compute the minimal alpha-channel difference (opposite of alpha blending) between the selected images and the specified base image.

The alpha difference A-B is defined as the image having minimal opacity, such that \text{alpha\_blend}(B,A-B) = A.

\textbf{Default value:}

- ‘\text{opacity\_gain}=1’.
2.19. IMAGE SEQUENCES AND VIDEOS

2.19.1 animate

Arguments:

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

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

Default value:

- ‘delay=30’.
2.19.2  apply_camera

Arguments:

- _"command"_, _camera_index>=0_, _skip_frames>=0_, _output_filename_

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

Default values:

- ‘command=’’’, ‘camera_index=0’ (default camera), ‘skip_frames=0’ and ‘output_filename=’’’.

2.19.3  apply_files

Arguments:

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

Apply a G’MIC command on specified input image files, in a streamed way.
If a display window is opened, rendered frames are displayed in it during processing.
The output filename may have extension ‘.avi’ (saved as a video), or any other usual image file extension (saved as a sequence of images).

Default values:

- ‘command=(undefined)’, ‘first_frame=0’, ‘last_frame=-1’, ‘frame_step=1’ and ‘output_filename=(undefined)’.
2.19.4 apply_video

Arguments:

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

Apply a G'MIC command on all frames of the specified input video file, in a streamed way. If a display window is opened, rendered frames are displayed in it during processing. The output filename may have extension '.avi' (saved as a video), or any other usual image file extension (saved as a sequence of images).

Default values:

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

2.19.5 average_files

Arguments:

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

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

Default values:

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

2.19.6 average_video

Arguments:

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

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

Default values:

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

2.19.7  **fade**  _files_

**Arguments:**

- "filename_pattern", \(nb\_inner\_frames>0, first\_frame>=0, last\_frame={ >=0 \mid -1=last }\), frame\_step>=1, output\_filename

Generate a temporal fading from specified input image files, in a streamed way.
If a display window is opened, rendered frames are displayed in it during processing.
The output filename may have extension ‘avi’ (saved as a video), or any other usual image file extension
(saved as a sequence of images).

**Default values:**

- ‘nb\_inner\_frames=10’, ‘first\_frame=0’, ‘last\_frame=-1’, ‘frame\_step=1’ and ‘output\_filename=(undefined)’.

2.19.8  **fade**  _video_

**Arguments:**

- video\_filename, \(nb\_inner\_frames>0, first\_frame>=0, last\_frame={ >=0 \mid -1=last }\), frame\_step>=1, output\_filename

Create a temporal fading sequence from specified input video file, in a streamed way.
If a display window is opened, rendered frames are displayed in it during processing.

**Default values:**

- ‘nb\_inner\_frames=10’, ‘first\_frame=0’, ‘last\_frame=-1’, ‘frame\_step=1’ and ‘output\_filename=(undefined)’.

2.19.9  **files2video**

**Arguments:**

- "filename_pattern", output\_filename, fps>0, codec

Convert several files into a single video file.

**Default values:**

- ‘output\_filename=output.avi’, ‘fps=25’ and ‘codec=mp4v’.

2.19.10  **median**  _files_

**Arguments:**

- "filename_pattern", first\_frame>=0, last\_frame={ >=0 \mid -1=last }, frame\_step>=1, frame\_rows[%]>=1, is\_fast\_approximation={ 0 \mid 1 }

Compute the median frame of specified input image files, in a streamed way.
If a display window is opened, rendered frame is displayed in it during processing.
2.19. IMAGE SEQUENCES AND VIDEOS

Default values:

- ‘first_frame=0’, ‘last_frame=-1’, ‘frame_step=1’, ‘frame_rows=20%’ and ‘is_fast_approximation=0’.

2.19.11 median_video

Arguments:

- video_filename, first_frame>0, last_frame={ >=0 | -1=last }, frame_step>=1, frame_rows[%]=1, is_fast_approximation={ 0 | 1 }

Compute the median of all frames of an input video file, in a streamed way.
If a display window is opened, rendered frame is displayed in it during processing.

Default values:

- ‘first_frame=0’, ‘last_frame=-1’, ‘frame_step=1’, ‘frame_rows=100%’ and ‘is_fast_approximation=1’.

2.19.12 morph

Arguments:

- nb_inner_frames>=1, smoothness>=0, precision>=0

Create morphing sequence between selected images.

Default values:

- ‘smoothness=0.1’ and ‘precision=4’.

[Images of pencil sketches]
2.19.13  \texttt{morph\_files}

Arguments:

- "filename_pattern", nb\_inner\_frames$> 0$, smoothness$> 0$, precision$> 0$, first\_frame$= 0$, last\_frame$= \{ > 0 \mid -1= last \}$, frame\_step$= 1$, output\_filename

Generate a temporal morphing from specified input image files, in a streamed way. If a display window is opened, rendered frames are displayed in it during processing. The output filename may have extension `.avi` (saved as a video), or any other usual image file extension (saved as a sequence of images).

Default values:

- `nb\_inner\_frames=10`, `smoothness=0.1`, `precision=4`, `first\_frame=0`, `last\_frame=-1`, `frame\_step=1` and `output\_filename=undefined`.

2.19.14  \texttt{morph\_rbf}

Arguments:

- nb\_inner\_frames$> 1$, xs\_0\%$, ys\_0\%$, xt\_0\%$, yt\_0\%$, \ldots$, xs\_N\%$, ys\_N\%$, xt\_N\%$, yt\_N\%

Create morphing sequence between 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.

2.19.15  \texttt{morph\_video}

Arguments:

- video\_filename, nb\_inner\_frames$> 0$, smoothness$> 0$, precision$> 0$, first\_frame$= 0$, last\_frame$= \{ > 0 \mid -1= last \}$, frame\_step$= 1$, output\_filename

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

Default values:

- `nb\_inner\_frames=10`, `smoothness=0.1`, `precision=4`, `first\_frame=0`, `last\_frame=-1`, `frame\_step=1` and `output\_filename=undefined`.
2.19. IMAGE SEQUENCES AND VIDEOS

2.19.16  \textit{register\_nonrigid}

Arguments:

- \{destination\}, \textit{smoothness}>0, \textit{precision}>0, \textit{nb\_scale}>0

Register selected source images with specified destination image, using non-rigid warp.

Default values:

- ‘\textit{smoothness}=0.2’, ‘\textit{precision}=6’ and ‘\textit{nb\_scale}=0\text{(auto)}’.

Example 701: \texttt{image.jpg +rotate 20,1,1,50\%,50\% +register\_nonrigid[0] [1]}

2.19.17  \textit{register\_rigid}

Arguments:

- \{destination\}, \textit{smoothness}>0, \textit{boundary\_conditions}={ 0=d\text{irichlet} | 1=n\text{eumann} | 2=p\text{eriodic} | 3=m\text{irror} }

Register selected source images with specified destination image, using rigid warp (shift).

Default values:

- ‘\textit{smoothness}=0.1\%’ and ‘\textit{boundary\_conditions}=0’.
2.19.18 *transition*

Arguments:

- \( [\text{transition}\_\text{shape}], nb\_\text{added}\_\text{frames}>=0,100, \text{shading}>=0, \text{single}\_\text{frame}\_\text{only}=-1\) = disabled | >=0

Generate a transition sequence between selected images.

Default values:

- ‘shading=0’ and ‘single_frame_only=-1’.
2.19. IMAGE SEQUENCES AND VIDEOS

Example 703: `image.jpg +mirror c 100%,100% plasma[-1] 1,1,6 transition[0,1] [2],5

2.19.19 transition3d

Arguments:

- _nb_frames>=2, _nb_xtiles>0, _nb_ytiles>0, _axis_x, _axis_y, _axis_z, _is_antialias=
  0 | 1

Create 3D transition sequence between selected consecutive images. 
'axis_x', 'axis_y' and 'axis_z' can be set as mathematical expressions, depending on 'x' and 'y'.

Default values:

- 'nb_frames=10', 'nb_xtiles=nb_ytiles=3', 'axis_x=1', 'axis_y=1', 'axis_z=0' and 
  'is_antialias=1'.

Example 704: `image.jpg +blur 5 transition3d 9 display rgba`  

2.19.20  `video2files`  

**Arguments:**  
- `input_filename, output_filename, first_frame>=0, last_frame={>=0 | -1=last}, frame_step>=1`  

Split specified input video file into image files, one for each frame. First and last frames as well as step between frames can be specified.  

**Default values:**  
- `'output_filename=frame.png', 'first_frame=0', 'last_frame=-1' and 'frame_step=1'`.  

2.20  **Neural Networks**  

2.20.1  `nn_new_input`  

**Arguments:**  
- `module_name, width, height, spectrum`  

Add an input module with specified size to the neural network.
2.20. NEURAL NETWORKS

2.20.2  **nn_new_output**

Arguments:

- **module_name, previous_module_name**

Add an output module to the neural network.

2.20.3  **nn_new_fullyconnected**

Arguments:

- **module_name, previous_module_name, nb_neurons, activation_function**

Add a fully-connected module to the neural network.

2.20.4  **nn_propagate_batch**

Arguments:

- **module_name, [inputs_zstacked]**

Batch propagate specified inputs through the neural network. Insert image of corresponding network outputs at the end of the list.

2.20.5  **nn_propagate**

Arguments:

- **module_name**

Propagate input through the neural network.

2.20.6  **nn_backpropagate_batch**

Arguments:

- **module_name, [inputs_zstacked], [expected_outputs_zstacked], insert_network_outputs={ 0 | 1 }, loss_function**

Batch propagate and backpropagate inputs and errors in neural network. Optionnally insert image of corresponding network outputs at the end of the list. Return averaged loss.

2.20.7  **nn_backpropagate**

Arguments:

- **module_name, [expected_output], loss_function**

Propagate input, then back-propagate output error, through the neural network. This command set the network output. Return average loss.
2.20.8  

\textit{nn\_update}

\textbf{Arguments:}

- \texttt{module\_name,epsilon}

Update neural network weights, after back-propagation of the error.

2.20.9  

\textit{nn\_output}

\textbf{Arguments:}

- \texttt{module\_name,\_filename}

Output specified network as a file.

2.20.10  

\textit{nn\_serialize}

\textbf{Arguments:}

- \texttt{module\_name,\_is\_compressed=\{0 | 1\}}

Serialize network into a single image, optionnally in a compressed form.

2.20.11  

\textit{nn\_unserialize}

Unserialize specified selection to retrieve a neural network.

2.20.12  

\textit{nn\_input}

\textbf{Arguments:}

- \texttt{"filename"}

Input neural network from file.

2.21  

\textbf{Convenience Functions}

2.21.1  

\textit{alert}

\textbf{Arguments:}

- \texttt{\_title,\_message,\_label\_button1,\_label\_button2,\ldots}

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.

\textbf{Default values:}

- ‘\texttt{title=[G'MIC Alert]}’ \textit{and} ‘\texttt{message=This is an alert box.’}.
2.21. CONVENIENCE FUNCTIONS

2.21.2 arg
Arguments:
- \( n > 1, \text{arg}_1, \ldots, \text{arg}_N \)

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

2.21.3 arg2var
Arguments:
- \( \text{variable}_\text{name}, \text{argument}_1, \ldots, \text{argument}_N \)

For each \( i \) in \([1..N]\), set \( \text{variable}_\text{name}_i=\text{argument}_i \). The variable name should be global to make this command useful (i.e. starts by an underscore).

2.21.4 autocrop_coords
Arguments:
- \( \text{value}_1, \text{value}_2, \ldots \) | auto

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

Default value:
- \( \text{auto} \)

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

2.21.6 base642img
Arguments:
- "base64_string"

Decode given base64-encoded string as a newly inserted image at the end of the list. The argument string must have been generated using command `img2base64`.

2.21.7 base642uchar
Arguments:
- "base64_string"

Decode given base64-encoded string as a newly inserted 1-column image at the end of the list. The argument string must have been generated using command `uchar2base64`.
2.21.8  basename

Arguments:

• file_path, variable_name_for_folder

Return the basename of a file path, and opt. its folder location. When specified 'variable_name_for_folder' must start by an underscore (global variable accessible from calling function).

2.21.9  bin

Arguments:

• binary_int1,...

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

2.21.10  bin2dec

Arguments:

• binary_int1,...

Convert specified binary integers into their decimal representations.

2.21.11  covariance_colors

Arguments:

• _avg_outvarname

Return the covariance matrix of the vector-valued colors in the latest of the selected images (for arbitrary number of channels). Parameter 'avg_outvarname' is used as a variable name that takes the value of the average vector-value.

2.21.12  dec

Arguments:

• decimal_int1,...

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

2.21.13  dec2str

Arguments:

• decimal_int1,...

Convert specified decimal integers into its string representation.
2.21. CONVENIENCE FUNCTIONS

2.21.14  \textit{dec2bin}

\textbf{Arguments:}
- decimal, int1,...

Convert specified decimal integers into their binary representations.

2.21.15  \textit{dec2hex}

\textbf{Arguments:}
- decimal, int1,...

Convert specified decimal integers into their hexadecimal representations.

2.21.16  \textit{dec2oct}

\textbf{Arguments:}
- decimal, int1,...

Convert specified decimal integers into their octal representations.

2.21.17  \textit{fact}

\textbf{Arguments:}
- value

Return the factorial of the specified value.

2.21.18  \textit{fibonacci}

\textbf{Arguments:}
- $N \geq 0$

Return the Nth number of the Fibonacci sequence.

2.21.19  \textit{file_mv}

\textbf{Arguments:}
- filename\_src, filename\_dest

Rename or move a file from a location $1$ to another location $2$.

2.21.20  \textit{file_rand}

Return a random filename for storing temporary data.
2.21.21  file rm

Arguments:

- filename

Delete a file.

2.21.22  filename

Arguments:

- filename, number1, number2, ..., numberN

Return a filename numbered with specified indices.

2.21.23  files (+)

Arguments:

- mode, path

Return the list of files and/or subfolders from specified path. 'path' can be eventually a matching pattern. 'mode' can be { 0=files only | 1=folders only | 2=files + folders }. Add '3' to 'mode' to return full paths instead of filenames only.

Default value:

- 'mode=5'.

2.21.24  fitratio_wh

Arguments:

- min_width, min_height, ratio_wh

Return a 2D size 'width,height' which is bigger than 'min_width,min_height' and has the specified w/h ratio.

2.21.25  fitscreen

Arguments:

- width, height, depth, minimal_size[%], maximal_size[%]
  - [image], minimal_size[%], maximal_size[%]

Return the 'ideal' size WxH for a window intended to display an image of specified size on screen.

Default values:

- 'depth=1', 'minimal_size=128' and 'maximal_size=85%'.
2.21.26  *fontchart*

Insert G’MIC font chart at the end of the image list.

Example 705 : *fontchart*

2.21.27  *fps*

Return the number of time this function is called per second, or -1 if this info is not yet available.
Useful to display the framerate when displaying animations.

2.21.28  *gcd*

**Arguments:**

- a, b

Return the GCD (greatest common divisor) between a and b.

2.21.29  *hex*

**Arguments:**

- hexadecimal_int1,...

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

2.21.30  *hex2dec*

**Arguments:**
• hexadecimal_int1,...

Convert specified hexadecimal integers into their decimal representations.

2.21.31  hex2img

Arguments:
  • "hexadecimal_string"

Insert new image 1xN at the end of the list with values specified by the given hexadecimal-encoded string.

2.21.32  hex2str

Arguments:
  • hexadecimal_string

Convert specified hexadecimal string into a string.

2.21.33  img2base64

Arguments:
  • encoding={ 0=base64 | 1=base64url }, store_names={ 0 | 1 }

Encode selected images as a base64-encoded string.
The images can be then decoded using command "base642img".

Default values:
  • 'encoding=0'.

2.21.34  img2hex

Return representation of last image as an hexadecimal-encoded string.
Input image must have values that are integers in [0,255].

2.21.35  img2str

Return the content of the latest of the selected images as a special G’MIC input string.

2.21.36  img2text

Arguments:
  • _line_separator

Return text contained in a multi-line image.

Default value:
  • 'line_separator= '.

2.21. CONVENIENCE FUNCTIONS

2.21.37  **img82hex**
Convert selected 8bits-valued vectors into their hexadecimal representations (ascii-encoded).

2.21.38  **hex2img8**
Convert selected hexadecimal representations (ascii-encoded) into 8bits-valued vectors.

2.21.39  **is 3d**
Return 1 if all of the selected images are 3D objects, 0 otherwise.

2.21.40  **is_change**

Arguments:

- _value={ 0=false | 1=true }

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 value:
- 'value=1'.

2.21.41  **is half**
Return 1 if the type of image pixels is limited to half-float.

2.21.42  **is ext**

Arguments:

- filename, extension

Return 1 if specified filename has a given extension.

2.21.43  **is image arg**

Arguments:

- string

Return 1 if specified string looks like '[ind]'.

2.21.44  **is pattern**

Arguments:

- string

Return 1 if specified string looks like a drawing pattern '0x.....'.
2.21.45  *is_percent*

**Arguments:**

- *string*

Return 1 if specified string ends with a ’%’, 0 otherwise.

2.21.46  *is_variable_name*

**Arguments:**

- "str"

Returns 1 if specified argument can be considered as a variable name, 0 otherwise.

2.21.47  *is_videofile*

Return 1 if extension of specified filename is typical from video files.

2.21.48  *is_macos*

Return 1 if current computer OS is Darwin (MacOS), 0 otherwise.

2.21.49  *is_windows*

Return 1 if current computer OS is Windows, 0 otherwise.

2.21.50  *math_lib*

Return string that defines a set of several useful macros for the embedded math evaluator.

2.21.51  *mad*

Return the MAD (Maximum Absolute Deviation) of the last selected image. 
The MAD is defined as $MAD = \text{med}_i \mid x_i - \text{med}_{j}(x_j) \mid$

2.21.52  *max_w*

Return the maximal width between selected images.

2.21.53  *max_h*

Return the maximal height between selected images.

2.21.54  *max_d*

Return the maximal depth between selected images.

2.21.55  *max_s*

Return the maximal spectrum between selected images.

2.21.56  *max_wh*

Return the maximal wxh size of selected images.
2.21.57  \texttt{max\_whd}  
Return the maximal wxhxd size of selected images.

2.21.58  \texttt{max\_whds}  
Return the maximal wxhxdxs size of selected images.

2.21.59  \texttt{median\_color}  
Return the median color value of the last selected image.

2.21.60  \texttt{min\_w}  
Return the minimal width between selected images.

2.21.61  \texttt{min\_h}  
Return the minimal height between selected images.

2.21.62  \texttt{min\_d}  
Return the minimal depth between selected images.

2.21.63  \texttt{min\_s}  
Return the minimal s size of selected images.

2.21.64  \texttt{min\_wh}  
Return the minimal wxh size of selected images.

2.21.65  \texttt{min\_whd}  
Return the minimal wxhxd size of selected images.

2.21.66  \texttt{min\_whds}  
Return the minimal wxhxdxs size of selected images.

2.21.67  \texttt{named (+)}  

\textbf{Arguments:}

- \texttt{\_mode,"name1","name2",...}

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).
(\textit{eq. to `nmd'}).
'mode' can be [0=all indices (default) \mid 1=lowest index \mid 2=highest index \mid 3 = all indices (case insensitive) \mid 4 = lowest index (case insensitive) \mid 5 = highest index (case insensitive)]
2.21.68  \textit{normalize\_filename}  

\textbf{Arguments:}  
\begin{itemize}
  \item \texttt{filename}
\end{itemize}  

Return a "normalized" version of the specified filename, without spaces and capital letters.

2.21.69  \textit{oct}  

\textbf{Arguments:}  
\begin{itemize}
  \item \texttt{octal\_int1,...}
\end{itemize}  

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

2.21.70  \textit{oct2dec}  

\textbf{Arguments:}  
\begin{itemize}
  \item \texttt{octal\_int1,...}
\end{itemize}  

Convert specified octal integers into their decimal representations.

2.21.71  \textit{padint}  

\textbf{Arguments:}  
\begin{itemize}
  \item \texttt{number,\_size>0}
\end{itemize}  

Return a integer with 'size' digits (eventually left-padded with '0').

2.21.72  \textit{path\_cache}  

Return a path to store G'MIC data files for one user (whose value is OS-dependent).

2.21.73  \textit{path\_gimp}  

Return a path to store GIMP configuration files for one user (whose value is OS-dependent).

2.21.74  \textit{path\_tmp}  

Return a path to store temporary files (whose value is OS-dependent).

2.21.75  \textit{remove\_copymark}  

\textbf{Arguments:}  
\begin{itemize}
  \item "image\_name"
\end{itemize}  

Remove copy mark from names of selected images.

2.21.76  \textit{reset}  

Reset global parameters of the interpreter environment.
2.21. **CONVENIENCE FUNCTIONS**

2.21.77 **RGB**
Return a random int-valued RGB color.

2.21.78 **RGBA**
Return a random int-valued RGBA color.

2.21.79 **std\_noise**
Return the estimated noise standard deviation of the last selected image.

2.21.80 **str**

 Arguments:

- *string*

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

2.21.81 **str2hex**

 Arguments:

- *string*

 Convert specified string into a sequence of hexadecimal values.

2.21.82 **strcapitalize**

 Arguments:

- *string*

 Capitalize specified string.

2.21.83 **strcontains**

 Arguments:

- *string1,string2*

 Return 1 if the first string contains the second one.

2.21.84 **strcut**

 Arguments:

- "string",\_max\_chars>0,"\_cut\_pattern"

 Return string that has been cutoff in several parts, separated by specified ‘cut_pattern’. Each part contains at most ‘max_chars’ characters.
2.21.85  \textit{strlen}  

\textbf{Arguments:}

\begin{itemize}
  \item \textit{string1}
\end{itemize}

Return the length of specified string argument.

2.21.86  \textit{strreplace}  

\textbf{Arguments:}

\begin{itemize}
  \item \textit{string, search, replace}
\end{itemize}

Search and replace substrings in an input string.

2.21.87  \textit{strlowercase}  

\textbf{Arguments:}

\begin{itemize}
  \item \textit{string}
\end{itemize}

Return a lower-case version of the specified string.

2.21.88  \textit{str uppercase}  

\textbf{Arguments:}

\begin{itemize}
  \item \textit{string}
\end{itemize}

Return an upper-case version of the specified string.

2.21.89  \textit{strvar}  

\textbf{Arguments:}

\begin{itemize}
  \item \textit{string}
\end{itemize}

Return a simplified version of the specified string, that can be used as a variable name.

2.21.90  \textit{strver}  

\textbf{Arguments:}

\begin{itemize}
  \item \texttt{\_version}
\end{itemize}

Return the specified version number of the G’MIC interpreter, as a string.

\textbf{Default value:}

\begin{itemize}
  \item \texttt{\‘version=\_version’}.
\end{itemize}
2.21.91 \textit{tic}  
Initialize tic-toc timer.  
Use it in conjunction with ‘toc’.

2.21.92 \textit{toc}  
Display elapsed time of the tic-toc timer since the last call to ‘tic’.  
This command returns the elapsed time in the status value.  
Use it in conjunction with ‘tic’.

2.21.93 \textit{to_clutname}  
Arguments:  
- "string"  
Return simplified name that can be used as a CLUT name, from specified input string.

2.21.94 \textit{uchar2base64}  
Arguments:  
- \texttt{encoding=0-base64 | 1-base64url}  
Encode the values of the latest of the selected images as a base64-encoded string.  
The string can be decoded using command 'base642uchar'.  
Selected images must have values that are integers in [0,255].  
Default values:  
- 'encoding=0'.

2.22 Other Interactive Commands

2.22.1 \textit{demos}  
Arguments:  
- \texttt{run_in_parallel=0-no | 1=yes | 2-auto}  
Show a menu to select and view all G’MIC interactive demos.

2.22.2 \textit{x.2048}  
Launch the 2048 game.

2.22.3 \textit{x.blobs}  
Launch the blobs editor.

2.22.4 \textit{x.bouncing}  
Launch the bouncing balls demo.
2.22.5 **x_color_curves**

**Arguments:**
- `colorspace`:
  - `rgb`, `cmy`, `cmyk`, `hsi`, `hsl`, `hsv`, `lab`, `lch`, `ycbcr`, `last`

Apply color curves on selected RGB[A] images, using an interactive window. Set `colorspace` to `last` to apply last defined color curves without opening interactive windows.

**Default value:**
- `'colorspace=rgb'`.

2.22.6 **x_colorize**

**Arguments:**
- `is_lineart`:
  - `0`, `1`
- `max_resolution`:
  - `0`, `>128`
- `multichannels_output`:
  - `0`, `1`
- `[palette1]`, `[palette2]`, `[grabber1]`

Colorize selected B&W images, using an interactive window. When `>0`, argument `max_resolution` defines the maximal image resolution used in the interactive window.

**Default values:**
- `'is_lineart=1'`, `'max_resolution=1024'` and `'multichannels_output=0'`.

2.22.7 **x_connect4**
Launch the Connect Four game.

2.22.8 **x_crop**
Crop selected images interactively. (eq. to `xz`).

2.22.9 **x_cut**
Cut selected images interactively.

2.22.10 **x_fire**
Launch the fire effect demo.

2.22.11 **x_fireworks**
Launch the fireworks demo.

2.22.12 **x_fisheye**
Launch the fish-eye effect demo.

2.22.13 **x_fourier**
Launch the fourier filtering demo.
2.22. OTHER INTERACTIVE COMMANDS

2.22.14 x_grab_color

Arguments:

- variable_name

Open a color grabber widget from the first selected image. Argument `variable_name` specifies the variable that contains the selected color values at any time. Assigning `-1` to it forces the interactive window to close.

Default values:

- `variable_name=xgc_variable`.

2.22.15 x_hanoi

Launch the Tower of Hanoi game.

2.22.16 x_histogram

Launch the histogram demo.

2.22.17 x_hough

Launch the hough transform demo.

2.22.18 x_jawbreaker

Arguments:

- `0<width<20,0<height<20,0<balls<=8`

Launch the Jawbreaker game.

2.22.19 x_landscape

Launch the virtual landscape demo.

2.22.20 x_life

Launch the game of life.

2.22.21 x_light

Launch the light effect demo.

2.22.22 x_mandelbrot

Arguments:

- `julia={ 0 | 1 },_c0r,_c0i`

Launch Mandelbrot/Julia explorer.
2.22.23  \texttt{x\_mask\_color}

Arguments:

\begin{itemize}
\item \texttt{\_colorspace=\{ all | rgb | lrgb | ycbcr | lab | lch | hsv | hsi | hsl | cmy | cmyk | yiq \}}, \texttt{\_spatial\_tolerance=0}, \texttt{\_color\_tolerance=0}
\end{itemize}

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 'channel' (e.g. 'hsv.h' for the hue).

Default values:

\begin{itemize}
\item 'colorspace=all', 'spatial\_tolerance=5' and 'color\_tolerance=5'.
\end{itemize}

2.22.24  \texttt{x\_metaballs3d}

Launch the 3D metaballs demo.

2.22.25  \texttt{x\_minesweeper}

Arguments:

\begin{itemize}
\item 8<=width<=20, 8<=height<=20
\end{itemize}

Launch the Minesweeper game.

2.22.26  \texttt{x\_minimal\_path}

Launch the minimal path demo.

2.22.27  \texttt{x\_morph}

Arguments:

\begin{itemize}
\item \texttt{\_nb\_frames>=2, \_preview\_fidelity=\{ 0=coarsest | 1=coarse | 2=normal | 3=fine | 4=finest \}}
\end{itemize}

Launch the interactive image morpher.

Default values:

\begin{itemize}
\item 'nb\_frames=16' and 'preview\_fidelity=3'.
\end{itemize}

2.22.28  \texttt{x\_pacman}

Launch pacman game.

2.22.29  \texttt{x\_paint}

Launch the interactive painter.
2.22. OTHER INTERACTIVE COMMANDS

2.22.30  *x plasma*
Launch the plasma effect demo.

2.22.31  *x quantize rgb*

Arguments:
- \( \_\text{nbcolors} \geq 2 \)
Launch the RGB color quantization demo.

2.22.32  *x reflection3d*
Launch the 3D reflection demo.

2.22.33  *x rubber3d*
Launch the 3D rubber object demo.

2.22.34  *x segment*

Arguments:
- \( \_\text{max\_resolution}=\{0 \mid >128\} \)
Segment foreground from background in selected opaque RGB images, interactively. Return RGBA images with binary alpha-channels.

Default value:
- ‘\text{max\_resolution}=1024’.

2.22.35  *x select\_color*

Arguments:
- \( \_\text{variable\_name} \)
Display a RGB or RGBA color selector. Argument ‘variable\_name’ specifies the variable that contains the selected color values (as R,G,B,[A]) at any time. Its value specifies the initial selected color. Assigning ‘-1’ to it forces the interactive window to close.

Default value:
- ‘\text{variable\_name}=\text{xsc\_variable}’.
2.22.36  \textit{x\_select\_function1d}

Arguments:

- \textit{variable\_name, background\_curve\_R, background\_curve\_G, background\_curve\_B}

Open an interactive window, where the user can defined its own 1D function.
If an image is selected, it is used to display additional information: - The first row defines the values of a
background curve displayed on the window (e.g. an histogram). - The 2nd, 3rd and 4th rows define the R,G,B
color components displayed beside the X and Y axes.
Argument 'variable\_name' specifies the variable that contains the selected function keypoints at any time.
Assigning '-1' to it forces the interactive window to close.

Default values:

- 'variable\_name=xsf\_variable', 'background\_curve\_R=220',
  'background\_curve\_G=background\_curve\_B=background\_curve\_T'.

2.22.37  \textit{x\_select\_palette}

Arguments:

- \textit{variable\_name, number\_of\_columns={ 0=auto | >0 }}

Open a RGB or RGBA color selector widget from a palette.
The palette is given as a selected image.
Argument 'variable\_name' specifies the variable that contains the selected color values (as R,G,B,[A]) at any
time.
Assigning '-1' to it forces the interactive window to close.

Default values:

- 'variable\_name=xsp\_variable' and 'number\_of\_columns=2'.

2.22.38  \textit{x\_shadebobs}

Launch the shade bobs demo.

2.22.39  \textit{x\_spline}

Launch spline curve editor.

2.22.40  \textit{x\_starfield3d}

Launch the 3D starfield demo.

2.22.41  \textit{x\_tetris}

Launch tetris game.

2.22.42  \textit{x\_threshold}

Threshold selected images interactively.
2.22.43  \texttt{x_fictactoe}
Launch tic-tac-toe game.

2.22.44  \texttt{x_warp}

\textbf{Arguments:}

- \_\_nb_keypoints\_\_xgrid\_\_ygrid\_\_contours\_\_preview_fidelity\_\_background\_image\_\_opacity\_\_.

Launch the interactive image warper.

\textbf{Default values:}

- ‘\_\_nb_keypoints\_\_xgrid=\_\_nb_keypoints\_\_ygrid=2’, ‘\_\_nb_keypoints\_\_contours=0’ and ‘\_\_preview_fidelity=1’.

2.22.45  \texttt{x_waves}
Launch the image waves demo.

2.22.46  \texttt{x_whirl}

\textbf{Arguments:}

- \_\_opacity\_\_\_.

Launch the fractal whirls demo.

\textbf{Default values:}

- ‘\_\_opacity=0.2’.

2.23  Funny One-Liners

2.23.1  \texttt{oneliner_sierpinski_carpet}
Draw a sierpinski carpet.
2.23.2 oneline\_sierpinski\_triangle

Draw a sierpinski triangle.

Example 707: 1024, 1024, 1, 1, "x>y?0:y<2?1:xor(j[0,-1],j[-1,-1])" f. "255*j(-w/2+y/2,0)"
2.23.3 oneliner lighten
A colorful lightspeed travel effect.

Example 708: 500,500 repeat 10 +noise poissondisk[0] {3+8} done rm[0] a z f
"%!z?(R=cut(norm(x-w/2,y-h/2)/20,0,d-1);i(x,y,R)):0" slices 0 to rgb f
"max(I)?u([255,255,255]):I" blur radial 0.6% equalize n 0,255

2.24 Command shortcuts

- '!=' (+) is equivalent to 'neq'.
- '<' (+) is equivalent to 'lt'.
- '<<' (+) is equivalent to 'bsl'.
- '<=' (+) is equivalent to 'le'.
- '>' (+) is equivalent to 'gt'.
- '>>' (+) is equivalent to 'bsr'.
- '=>' (+) is equivalent to 'ge'.
- '*' (+) is equivalent to 'mul'.
- '*3d' (+) is equivalent to 'mul3d'.
- '+' (+) is equivalent to 'add'.
- '+3d' (+) is equivalent to 'add3d'.
- '-' (+) is equivalent to 'sub'.
- '-3d' (+) is equivalent to 'sub3d'.
- '/' (+) is equivalent to 'div'.
- '/3d' (+) is equivalent to 'div3d'.
- '==' (+) is equivalent to 'set'.
- '%=' (+) is equivalent to 'mod'.
- '&=' (+) is equivalent to 'and'.
- '!' (+) is equivalent to 'neq'.
- '<=' (+) is equivalent to 'le'.
- '==' (+) is equivalent to 'eq'.
- '&&' (+) is equivalent to 'and'.
• "\"\" (+) is equivalent to 'pow'.
• 'a' (+) is equivalent to 'append'.
• 'ac' is equivalent to 'apply.channels'.
• 'ap' is equivalent to 'apply.parallel'.
• 'apc' is equivalent to 'apply.parallel.channels'.
• 'apo' is equivalent to 'apply.parallel.overlap'.
• 'at' is equivalent to 'apply.tiles'.
• 'b' (+) is equivalent to 'blur'.
• 'c' (+) is equivalent to 'cut'.
• 'c3d' is equivalent to 'center3d'.
• 'col3d' (+) is equivalent to 'color3d'.
• 'd' (+) is equivalent to 'display'.
• 'd0' is equivalent to 'display0'.
• 'd2d' is equivalent to 'display2d'.
• 'd3d' is equivalent to 'display3d'.
• 'da' is equivalent to 'display_array'.
• 'db3d' (+) is equivalent to 'double3d'.
• 'dfft' is equivalent to 'display_fft'.
• 'dg' is equivalent to 'display_graph'.
• 'dh' is equivalent to 'display_histogram'.
• 'dp' is equivalent to 'display_parallel'.
• 'dp0' is equivalent to 'display_parallel0'.
• 'dq' is equivalent to 'display_quiver'.
• 'drgba' is equivalent to 'display_rgba'.
• 'dt' is equivalent to 'display_tensors'.
• 'dw' is equivalent to 'display_warp'.
• 'e' (+) is equivalent to 'echo'.
• 'endl' (+) is equivalent to 'endlocal'.
• 'f' (+) is equivalent to 'fill'.
• 'f3d' (+) is equivalent to 'focale3d'.
• 'fc' is equivalent to 'fill_color'.
• 'frame' is equivalent to 'frame_xy'.
• 'g' (+) is equivalent to 'gradient'.
• 'h' is equivalent to 'help'.
• 'i' (+) is equivalent to 'input'.
• 'ig' is equivalent to 'input_glob'.
• 'ir' is equivalent to 'inrange'.
• 'it' is equivalent to 'input_text'.
• 'j' (+) is equivalent to 'image'.
• 'j3d' (+) is equivalent to 'object3d'.
• 'k' (+) is equivalent to 'keep'.
• 'l' (+) is equivalent to 'local'.
• 'l3d' (+) is equivalent to 'light3d'.
• 'm' (+) is equivalent to 'command'.
• 'm*' (+) is equivalent to 'mmul'.
• 'm/' (+) is equivalent to 'mdiv'.
• 'm3d' (+) is equivalent to 'mode3d'.
• 'md3d' (+) is equivalent to 'moded3d'.
• 'mv' (+) is equivalent to 'move'.
• 'n' (+) is equivalent to 'normalize'.

656  |  CHAPTER 2. LIST OF COMMANDS
• 'n3d' is equivalent to 'normalize3d'.
• 'nm' (+) is equivalent to 'name'.
• 'nmd' (+) is equivalent to 'named'.
• 'o' (+) is equivalent to 'output'.
• 'o3d' (+) is equivalent to 'opacity3d'.
• 'on' is equivalent to 'outputn'.
• 'op' is equivalent to 'outputp'.
• 'ot' is equivalent to 'output_text'.
• 'ow' is equivalent to 'outputw'.
• 'ox' is equivalent to 'outputx'.
• 'p' (+) is equivalent to 'print'.
• 'p3d' is equivalent to 'primitives3d'.
• 'q' (+) is equivalent to 'quit'.
• 'r' (+) is equivalent to 'resize'.
• 'r2dx' is equivalent to 'resize2dx'.
• 'r2dy' is equivalent to 'resize2dy'.
• 'r3d' (+) is equivalent to 'rotate3d'.
• 'r3dx' is equivalent to 'resize3dx'.
• 'r3dy' is equivalent to 'resize3dy'.
• 'r3dz' is equivalent to 'resize3dz'.
• 'ri' (+) is equivalent to 'resize_as_image'.
• 'rm' (+) is equivalent to 'remove'.
• 'rmn' is equivalent to 'remove_named'.
• 'rr2d' is equivalent to 'resize_ratio2d'.
• 'rv' (+) is equivalent to 'reverse'.
• 'rv3d' (+) is equivalent to 'reverse3d'.
• 's' (+) is equivalent to 'split'.
• 's3d' (+) is equivalent to 'split3d'.
• 'sh' (+) is equivalent to 'shared'.
• 'sl3d' (+) is equivalent to 'spec13d'.
• 'sp' is equivalent to 'sample'.
• 'ss3d' (+) is equivalent to 'specs3d'.
• 't' (+) is equivalent to 'text'.
• 't3d' is equivalent to 'texturize3d'.
• 'to' is equivalent to 'text_outline'.
• 'u' (+) is equivalent to 'status'.
• 'um' is equivalent to 'uncommand'.
• 'up' is equivalent to 'update'.
• 'v' (+) is equivalent to 'verbose'.
• 'w' (+) is equivalent to 'window'.
• 'x' (+) is equivalent to 'exec'.
• 'xz' is equivalent to 'x_crop'.
• 'y' (+) is equivalent to 'unroll'.
• 'z' (+) is equivalent to 'crop'.
• '|' (+) is equivalent to 'or'.
2.25 Examples of Use

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

- View a list of images:
  ```gmic file1.bmp file2.jpeg```

- Convert an image file:
  ```gmic input.bmp output output.jpg```

- Create a volumetric image from a movie sequence:
  ```gmic input.mpg append z output output.hdr```

- Compute image gradient norm:
  ```gmic input.bmp gradient norm```

- Denoise a color image:
  ```gmic image.jpg denoise 30,10 output denoised.jpg```

- Compose two images using overlay layer blending:
  ```gmic image1.jpg image2.jpg blend overlay output blended.jpg```

- Evaluate a mathematical expression:
  ```gmic echo "\(\cos(\pi/4)/2+\sin(\pi/4)/2=\{\cos(\pi/4)/2+\sin(\pi/4)/2\}\)"```

- Plot a 2D function:
  ```gmic 1000,1,1,2 fill "X=3*(x−500)/500;X^2+\sin(3*X^2)+if(c==0,u(0,−1),\cos(X+10))" plot```

- Plot a 3D elevated function in random colors:
  ```gmic 128,128,1,3, u(0,255) plasma 10,3 blur sharpen 10000 \ elevation3d[−1] "X=(x−64)/6;Y=(y−64)/6;100+exp(-(X^2+Y^2)/30)*abs(cos(X)*sin(Y))""```

- Plot the isosurface of a 3D volume:
  ```gmic mode3d 5 moded3d 5 double3d 0 isosurface3d "X^2+y^2+abs(z)*abs(4*cos(x*y*z)+3)"",3```

- Render a G'MIC 3D logo:
  ```gmic 0 text G'MIC,0,0,53,1,1,1,1 expand_xy 10,0 blur 1 normalize 0,100 +plasma 0.4 add blur 1 elevation3d −0.1 moded3d 4```

- Generate a 3D ring of torii:
  ```gmic repeat 20 torus3d 15,2 color3d[−1] "\{u(60,255)\},\{u(60,255)\}\,\{u(60,255)\}\" \ +3d[−1] 0.5,1 if "\(5>\%2\)" rotate3d[−1] 0,1,0,90 fi add3d[−1] 70 add3d \ rotate3d 0,0,1,18 done moded3d 3 mode3d 5 double3d 0```

- Create a vase from a 3D isosurface:
  ```gmic moded3d 4 isosurface3d "X^2+2*y^2+abs(y/2)+\sin(2*y)^2+z^2−3",0" sphere3d 1.5 \ sub3d[−1] 0.5 plane3d 15,15 rotate3d[−1] 1,0,0,90 center3d[−1] 1 add3d[−1] 0.3,2 \ color3d[−1] 180,150,255 color3d[−2] 128,255,0 color3d[−3] 255,128,0 add3d```

- Display filtered webcam stream:
  ```gmic apply camera \"+mirror x +mirror y add div 4\"```

- Launch a set of G'MIC interactive demos:
gmic demos
Index of commands

abs, 77
acos, 78
acosh, 80
add, 80
add3d, 431
adjust_colors, 172
alert, 634
and, 82
animate, 623
animate3d, 433
append, 221
append_tiles, 223
apply_camera, 624
apply_camera3d, 433
apply_channels, 173
apply_curve, 134
apply_files, 624
apply_gamma, 135
apply_matrix3d, 433
apply_parallel, 502
apply_parallel_channels, 503
apply_parallel_overlap, 504
apply_scales, 224
apply_tiles, 505
apply_timeout, 506
apply_video, 625
area, 356
area_fg, 357
arg, 635
arg2var, 635
argmax, 83
argmaxabs, 84
argmin, 84
argminabs, 85
array, 517
array3d, 434
array_fade, 518
array_mirror, 519
array_random, 520
arrow, 387
arrow3d, 434
asin, 85
asinh, 87
at_line, 358
at_quadrangle, 359
atan, 87
atan2, 88
atanh, 88
autocrop, 225
autocrop_components, 226
autocrop_coords, 635
autocrop_seq, 227
autoindex, 174
average_colors, 635
average_files, 625
average_video, 625
axes, 388
axes3d, 435
balance_gamma, 136
ball, 389
bandpass, 265
barycenter, 359
base642img, 635
base642uchar, 635
basename, 636
bayer2rgb, 175
bilateral, 266
bin, 636
bin2dec, 636
blend, 611
blend_edges, 616
blend_fade, 616
blend_median, 617
blend_seamless, 618
blur, 267
blur_angular, 269
blur_bloom, 269
blur_linear, 270
blur_radial, 271
blurSelective, 272
blur_x, 273
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>blur_xy, 274</td>
<td></td>
</tr>
<tr>
<td>blur_xyz, 275</td>
<td></td>
</tr>
<tr>
<td>blur_y, 275</td>
<td></td>
</tr>
<tr>
<td>blur_z, 276</td>
<td></td>
</tr>
<tr>
<td>boundingbox3d, 436</td>
<td></td>
</tr>
<tr>
<td>box3d, 437</td>
<td></td>
</tr>
<tr>
<td>boxfilter, 276</td>
<td></td>
</tr>
<tr>
<td>boxfitting, 543</td>
<td></td>
</tr>
<tr>
<td>break, 507</td>
<td></td>
</tr>
<tr>
<td>brushify, 544</td>
<td></td>
</tr>
<tr>
<td>bsl, 89</td>
<td></td>
</tr>
<tr>
<td>bsr, 89</td>
<td></td>
</tr>
<tr>
<td>bump2normal, 278</td>
<td></td>
</tr>
<tr>
<td>camera, 27</td>
<td></td>
</tr>
<tr>
<td>cartoon, 545</td>
<td></td>
</tr>
<tr>
<td>cast, 137</td>
<td></td>
</tr>
<tr>
<td>center3d, 437</td>
<td></td>
</tr>
<tr>
<td>channels, 228</td>
<td></td>
</tr>
<tr>
<td>check, 506</td>
<td></td>
</tr>
<tr>
<td>check3d, 506</td>
<td></td>
</tr>
<tr>
<td>check_display, 507</td>
<td></td>
</tr>
<tr>
<td>chessboard, 390</td>
<td></td>
</tr>
<tr>
<td>cie1931, 391</td>
<td></td>
</tr>
<tr>
<td>circle, 391</td>
<td></td>
</tr>
<tr>
<td>circle3d, 438</td>
<td></td>
</tr>
<tr>
<td>circles3d, 439</td>
<td></td>
</tr>
<tr>
<td>close_binary, 392</td>
<td></td>
</tr>
<tr>
<td>clut, 28</td>
<td></td>
</tr>
<tr>
<td>cmv2rgb, 177</td>
<td></td>
</tr>
<tr>
<td>cmyk2rgb, 177</td>
<td></td>
</tr>
<tr>
<td>color3d, 440</td>
<td></td>
</tr>
<tr>
<td>color_ellipses, 546</td>
<td></td>
</tr>
<tr>
<td>colorblind, 177</td>
<td></td>
</tr>
<tr>
<td>colorcube3d, 441</td>
<td></td>
</tr>
<tr>
<td>colormap, 178</td>
<td></td>
</tr>
<tr>
<td>columns, 229</td>
<td></td>
</tr>
<tr>
<td>command, 31</td>
<td></td>
</tr>
<tr>
<td>complex2polar, 137</td>
<td></td>
</tr>
<tr>
<td>compose_channels, 179</td>
<td></td>
</tr>
<tr>
<td>compose_freq, 279</td>
<td></td>
</tr>
<tr>
<td>compress_clut, 138</td>
<td></td>
</tr>
<tr>
<td>compress_rle, 138</td>
<td></td>
</tr>
<tr>
<td>cone3d, 442</td>
<td></td>
</tr>
<tr>
<td>continue, 507</td>
<td></td>
</tr>
<tr>
<td>convolve, 279</td>
<td></td>
</tr>
<tr>
<td>convolve_fft, 281</td>
<td></td>
</tr>
<tr>
<td>correlate, 281</td>
<td></td>
</tr>
<tr>
<td>cos, 90</td>
<td></td>
</tr>
<tr>
<td>cosh, 91</td>
<td></td>
</tr>
<tr>
<td>covariance_colors, 636</td>
<td></td>
</tr>
<tr>
<td>cracks, 599</td>
<td></td>
</tr>
<tr>
<td>crop, 230</td>
<td></td>
</tr>
<tr>
<td>cross_correlation, 282</td>
<td></td>
</tr>
<tr>
<td>cubes3d, 442</td>
<td></td>
</tr>
<tr>
<td>cubism, 547</td>
<td></td>
</tr>
<tr>
<td>cumulative, 139</td>
<td></td>
</tr>
<tr>
<td>cup3d, 443</td>
<td></td>
</tr>
<tr>
<td>cursor, 32</td>
<td></td>
</tr>
<tr>
<td>curvature, 283</td>
<td></td>
</tr>
<tr>
<td>cut, 140</td>
<td></td>
</tr>
<tr>
<td>cylinder3d, 444</td>
<td></td>
</tr>
<tr>
<td>dct, 284</td>
<td></td>
</tr>
<tr>
<td>deblur, 285</td>
<td></td>
</tr>
<tr>
<td>deblur_goldmeinel, 286</td>
<td></td>
</tr>
<tr>
<td>deblur_richardsonlucy, 287</td>
<td></td>
</tr>
<tr>
<td>debug, 27</td>
<td></td>
</tr>
<tr>
<td>dec, 636</td>
<td></td>
</tr>
<tr>
<td>dec2bin, 637</td>
<td></td>
</tr>
<tr>
<td>dec2hex, 637</td>
<td></td>
</tr>
<tr>
<td>dec2oct, 637</td>
<td></td>
</tr>
<tr>
<td>dec2str, 636</td>
<td></td>
</tr>
<tr>
<td>decompress_clut, 141</td>
<td></td>
</tr>
<tr>
<td>decompress_clut_pde, 142</td>
<td></td>
</tr>
<tr>
<td>decompress_clut_rbf, 142</td>
<td></td>
</tr>
<tr>
<td>decompress_rle, 142</td>
<td></td>
</tr>
<tr>
<td>deconvolve_fft, 288</td>
<td></td>
</tr>
<tr>
<td>deform, 580</td>
<td></td>
</tr>
<tr>
<td>deinterlace, 289</td>
<td></td>
</tr>
<tr>
<td>delaunay, 360</td>
<td></td>
</tr>
<tr>
<td>delaunay3d, 445</td>
<td></td>
</tr>
<tr>
<td>deltaE, 176</td>
<td></td>
</tr>
<tr>
<td>demos, 647</td>
<td></td>
</tr>
<tr>
<td>denoise, 290</td>
<td></td>
</tr>
<tr>
<td>denoise_haar, 291</td>
<td></td>
</tr>
<tr>
<td>denoise_patchpca, 292</td>
<td></td>
</tr>
<tr>
<td>deriche, 293</td>
<td></td>
</tr>
<tr>
<td>detect_skin, 361</td>
<td></td>
</tr>
<tr>
<td>diagonal, 231</td>
<td></td>
</tr>
<tr>
<td>diffusiontensors, 299</td>
<td></td>
</tr>
<tr>
<td>dijkstra, 425</td>
<td></td>
</tr>
<tr>
<td>dilate, 295</td>
<td></td>
</tr>
<tr>
<td>dilate_circ, 295</td>
<td></td>
</tr>
<tr>
<td>dilate_oct, 296</td>
<td></td>
</tr>
<tr>
<td>dilate_threshold, 297</td>
<td></td>
</tr>
<tr>
<td>direction2rgb, 180</td>
<td></td>
</tr>
<tr>
<td>discard, 142</td>
<td></td>
</tr>
<tr>
<td>displacement, 361</td>
<td></td>
</tr>
<tr>
<td>display, 32</td>
<td></td>
</tr>
</tbody>
</table>
display0, 33
display2d, 33
display3d, 33
display_array, 33
display_fft, 34
display_graph, 34
display_histogram, 35
display_parallel, 38
display_parallel0, 38
display_parametric, 36
display_polar, 38
display_quiver, 39
display_rgba, 40
display_tensors, 41
display_warp, 42
distance, 362
distribution3d, 445
ditheredbw, 181
div, 92
div3d, 446
div_complex, 94
divergence, 297
do, 508
document_gmic, 43
dog, 298
done, 509
double3d, 447
draw_whirl, 548
drawing, 549
drop_shadow, 550
echo, 43
echo_file, 44
echo_stdout, 44
edges, 300
eigen, 425
eigen2tensor, 143
elevate, 232
elevation3d, 448
eif, 509
ellipse, 392
ellipsionism, 551
else, 509
empty3d, 449
endian, 144
endlocal, 509
eq, 94
equalize, 144
equirectangular2nadirzenith, 582
erode, 301
erode_circ, 302
erode_oct, 303
erode_threshold, 304
error, 509
euclidean2polar, 581
eval, 509
exec, 510
exp, 95
expand_x, 232
expand_xy, 233
expand_xyz, 234
expand_y, 234
expand_z, 235
extract, 235
extract_region, 236
extrude3d, 450
fact, 637
fade_diamond, 618
fade_files, 626
fade_linear, 618
fade_radial, 619
fade_video, 626
fade_x, 620
fade_y, 621
fade_z, 622
fft, 304
fftpolar, 364
fi, 509
fibonacci, 637
file_mv, 637
file_rand, 637
file_rm, 638
filename, 638
files, 638
files2video, 626
fill, 145
fill_color, 182
fire_edges, 552
fisheye, 582
fitratio_wh, 638
fitscreen, 638
flood, 393
flower, 583
locale3d, 451
fontchart, 639
for, 510
fps, 639
fractalize, 553
frame_blur, 521
INDEX OF COMMANDS 663

frame_cube, 522
frame_fuzzy, 523
frame_painting, 524
frame_pattern, 525
frame_round, 526
frame_seamless, 527
frame_x, 528
frame_xy, 529
frame_xyz, 530
frame_y, 530
function1d, 44

gaussian, 394
gaussians3d, 452
gcd, 639
go, 96
glow, 554
gmic3d, 453
gradient, 306
gradient2rgb, 182
gradient_norm, 306
gradient_orientation, 307
graph, 395
grid, 396
gt, 98
guided, 308
gyroid3d, 454

haar, 309
halftone, 555
hardsketchbw, 556
hcy2rgb, 183
hearts, 557
heat_flow, 309
help, 27
hessian, 310
hex, 639
hex2dec, 639
hex2img, 640
hex2img8, 641
hex2str, 640
histogram, 364
histogram3d, 454
histogram_cumul, 366
histogram_nd, 365
histogram_pointwise, 367
hough, 367
houghsketchbw, 558
hsi2rgb, 183
hsi82rgb, 183

idct, 311
iim, 311
if, 510
ifft, 312
ifftpolar, 368
ihaar, 312
ilaplacian, 312
image, 397
image6cube3d, 455
imageblocks3d, 456
imagecube3d, 456
imagegrid, 532
imagegrid_hexagonal, 533
imagegrid_triangular, 534
imageplane3d, 457
imagepyramid3d, 458
imerubik3d, 458
imagesphere3d, 459
img2ascii, 531
img2base64, 640
img2hex, 640
img2str, 640
img2text, 640
img82hex, 641
index, 147
inn, 313
inpaint, 314
inpaint_flow, 316
inpaint_holes, 317
inpaint_matchpatch, 319
inpaint_morpho, 318
inpaint_pde, 316
input, 45
input_565, 47
input_cube, 47
input_flo, 47
input_glob, 47
input_gpl, 47
input_text, 48
inrange, 149
int2rgb, 184
invert, 427
is_3d, 641
is_change, 641
is_ext, 641
INDEX OF COMMANDS

is_half, 641
is_image_arg, 641
is_macos, 642
is_pattern, 641
is_percent, 642
is_variable_name, 642
is_videofilename, 642
is_windows, 642
isoline3d, 460
isophotes, 368
isosurface3d, 461

jzazb2rgb, 184
jzazb2xyz, 184

kaleidoscope, 584
keep, 70
kuwahara, 320

laar, 371
lab2lch, 184
lab2rgb, 185
lab2srgb, 185
lab2xyz, 187
lab82rgb, 187
lab8srgb, 186
label, 369
label3d, 463
label_fg, 371
label_points3d, 463
laplacian, 321
lathe3d, 464
lch2lab, 187
lch2rgb, 188
lch82rgb, 188
le, 99
lic, 322
light3d, 465
light_patch, 600
light_relief, 560
lightness, 189
lightrays, 559
line, 398
line3d, 466
linearize_tiles, 535
linethick, 399
linify, 561
lissajous3d, 467
local, 511
log, 102

log10, 103
log2, 104
lt, 101
luminance, 188
lut_contrast, 189
mad, 642
mandelbrot, 400
map, 149
map_clut, 189
map_sphere, 585
map_sprites, 536
map_tones, 322
map_tones_fast, 323
marble, 401
matchpatch, 375
math_lib, 642
max, 105
max_d, 642
max_h, 642
max_patch, 371
max_s, 642
max_w, 642
max_wh, 642
max_whd, 643
max_whds, 643
maxabs, 107
maze, 402
maze_mask, 403
mdiv, 107
meancurvature_flow, 324
med, 108
median, 325
median_color, 643
median_files, 626
median_video, 627
min, 108
min_d, 643
min_h, 643
min_patch, 372
min_s, 643
min_w, 643
min_wh, 643
min_whd, 643
min_whds, 643
minabs, 109
minimal_path, 373
mirror, 238
mix_channels, 151
mix_rgb, 190
mmul, 111
mod, 110
mode3d, 468
moded3d, 469
montage, 237
morph, 627
morph_files, 628
morph_rbf, 628
morph_video, 628
mosaic, 562
move, 71
mproj, 427
mse, 374
mul, 112
mul3d, 469
mul_channels, 114
mul_complex, 115
mutex, 512

nadirzenith2equirectangular, 586
name, 73
named, 643
negate, 151
neq, 115
network, 48
newton_fractal, 404
nlmeans, 326
nlmeans_core, 327
nn_backpropagate, 633
nn_backpropagate_batch, 633
nn_input, 634
nn_new_fullyconnected, 633
nn_new_input, 632
nn_new_output, 633
nn_output, 634
nn_propagate, 633
nn_propagate_batch, 633
nn_serialize, 634
nn_unserialize, 634
nn_update, 634
noarg, 512
noise, 152
noise_hurl, 601
noise_perlin, 154
noise_poissondisk, 154
norm, 156
normalize, 157
normalize3d, 470
normalize_filename, 644
normalize_local, 327

normalize_sum, 157
normalized_cross_correlation, 328
normp, 155
not, 158

object3d, 404
oct, 644
oct2dec, 644
old_photo, 563
oneliner_lightspeed, 655
oneliner_sierpinski_carpet, 653
oneliner_sierpinski_triangle, 654
oneminus, 159
onfail, 513
opacity3d, 471
or, 116
orientation, 159
orthogonalize, 427
otsu, 160
output, 48
output_565, 48
output_cube, 48
output_flo, 49
output_ggr, 49
output_text, 49
outputn, 49
outputp, 49
outputw, 49
outputx, 50

pack, 537
pack_sprites, 405
padint, 644
palette, 191
parallel, 513
parametric3d, 471
parse_cli, 68
parse_gui, 68
pass, 50
patches, 375
path_cache, 644
path_gimp, 644
path_tmp, 644
pca_patch3d, 472
pde_flow, 332
pencilbw, 564
percentile, 329
periodize_poisson, 333
permute, 239
peronamalik_flow, 330
INDEX OF COMMANDS

phase_correlation, 331
piechart, 406
pixelize, 602
pixelsort, 564
plane3d, 473
plasma, 407
plot, 51
plot2value, 376
point, 408
point3d, 474
pointcloud, 377
pointcloud3d, 475
polar2complex, 161
polar2euclidean, 586
polaroid, 565
polka_dots, 409
polygon, 410
polygonize, 566
pose3d, 475
poster_edges, 567
poster_hope, 568
pow, 117
primitives3d, 476
print, 51
progress, 514
projections3d, 477
pseudogray, 192
psnr, 378
puzzle, 538
pyramid3d, 477
quadrangle3d, 478
quadratize_tiles, 538
quantize, 161
quantize_area, 163
quit, 514
quiver, 412
raindrops, 587
rand, 163
random3d, 479
random_pattern, 51
rbf, 334
rectangle, 413
red_eye, 335
register_nonrigid, 629
register_rigid, 629
remove, 73
remove_copymark, 644
remove_duplicates, 74
remove_empty, 75
remove_hotpixels, 336
remove_named, 75
remove_opacity, 211
remove_pixels, 337
repeat, 514
replace, 164
replace_color, 192
replace_inf, 165
replace_nan, 165
replace_naninf, 166
replace_seq, 166
replace_str, 167
reset, 644
resize, 240
resize2dx, 244
resize2dy, 245
resize3d, 246
resize3dy, 246
resize3dz, 247
resize_as_image, 241
resize_mm, 242
resize_pow2, 243
resize_ratio2d, 244
retinex, 193
return, 516
reverse, 75
reverse3d, 479
RGB, 645
rgb2bayer, 193
gb2cmy, 194
gb2cmyk, 195
gb2hcy, 196
gb2hsi, 196
gb2hsi8, 197
gb2hsl, 198
gb2hsl8, 199
gb2hsv, 199
gb2hsv8, 200
gb2int, 201
gb2jzazbz, 202
gb2lab, 202
gb2lab8, 202
gb2lch, 203
gb2lch8, 204
gb2luv, 205
gb2srgb, 206
gb2xyz, 206
gb2xyz8, 207
rgb2ycbcr, 209
rgb2yiq, 208
rgb2yiq8, 209
rgb2yuv, 210
rgb2yuv8, 211
RGBA, 645
ripple, 588
rodilius, 569
rol, 119
rolling_guidance, 338
ror, 120
rorschach, 414
rotate, 247
rotate3d, 480
rotate_tileable, 248
rotate_tiles, 539
rotation3d, 481
rotoidoscope, 589
round, 168
roundify, 169
rows, 248
rprogress, 516
run, 516
ryb2rgb, 211
sample, 65
scale2x, 249
scale3x, 250
scale_dcci2x, 250
scanlines, 603
screen, 52
seamcarve, 251
segment_watershed, 379
select, 52
select_color, 211
sepia, 212
serialize, 53
set, 170
shade_stripes, 604
shadow_patch, 605
shape2bump, 380
shape_circle, 53
shape_cupid, 54
shape_diamond, 55
shape_dragon, 56
shape_fern, 57
shape_gear, 58
shape_heart, 59
shape_polygon, 60
shape_snowflake, 61
shape_star, 62
shared, 63
sharpen, 339
shift, 252
shift_tiles, 540
shrink_x, 253
shrink_xy, 254
shrink_xyz, 255
shrink_y, 255
shrink_z, 256
sierpinski, 415
sierpinski3d, 481
sign, 121
sin, 122
sinc, 123
sinh, 124
size3d, 482
skeleton, 380
skeleton3d, 482
sketchbw, 571
skip, 516
slic, 381
slices, 256
smooth, 340
snapshot3d, 483
solarize, 213
solidify, 346
solve, 428
solve_poisson, 343
sort, 256
sort_list, 76
specl3d, 484
specs3d, 485
sphere3d, 486
spherical3d, 487
spherize, 590
spiralbw, 416
spline, 417
spline3d, 488
split, 257
split3d, 489
split_colors, 214
split_details, 344
split_freq, 342
split_opacity, 214
split_tiles, 260
sponge, 571
spread, 606
sprite3d, 490
sprites3d, 491
INDEX OF COMMANDS

sqr, 125
sqr, 126
srand, 65
srgb2lab, 214
srgb2lab8, 215
srgb2rgb, 216
ssd_patch, 382
stained_glass, 572
star3d, 491
stars, 573
status, 516
std_noise, 645
stencil, 574
stencilbw, 575
store, 66
str, 645
str2hex, 645
strcapitalize, 645
strcontains, 645
strcut, 645
streamline3d, 492
stripes_y, 607
strlen, 646
strlowercase, 646
strreplace, 646
structuretensors, 345
struppercase, 646
strvar, 646
strver, 646
stylize, 576
sub, 127
sub3d, 493
sub_alpha, 622
superformula3d, 494
svd, 429
symmetrize, 591
syntexturize, 347
syntexturize_matchpatch, 348
tan, 130
tanh, 131
taquin, 541
tensors3d, 495
testimage2d, 66
tetraedron_shade, 418
tetris, 576
text, 418
text3d, 497
text_outline, 420
text_pointcloud3d, 496
texturize3d, 498
texturize_canvas, 608
texturize_paper, 609
thinning, 383
threshold, 171
tic, 647
toc, 647
tones, 384
topographic_map, 385
torus3d, 499
transfer_histogram, 218
transfer_pca, 218
transfer_rgb, 219
transform_polar, 592
transition, 630
transition3d, 631
transpose, 429
triangle3d, 500
triangle_shade, 421
trisolve, 430
truchet, 422
tsp, 385
tunnel, 542
turbulence, 423
tv_flow, 349
twirl, 593
uchar2base64, 647
uncommand, 67
undistort, 261
uniform_distribution, 67
unroll, 261
unserialize, 68
unsharp, 350
unsharp_octave, 351
update, 68
upscale_smart, 262
vanvliet, 352
variance_patch, 386
vector2tensor, 172
INDEX OF COMMANDS

verbose, 69
version, 27
video2files, 632
vignette, 610
volume3d, 501
voronoi, 353
wait, 69
warhol, 577
warn, 70
warp, 263
warp_patch, 264
warp_perspective, 594
warp_rbf, 264
water, 595
watermark_fourier, 354
watermark_visible, 610
watershed, 355
wave, 596
weave, 578
weird3d, 501
while, 517
whirls, 579
wind, 597
window, 70

x_2048, 647
x_blobs, 647
x_bouncing, 647
x_color_curves, 648
x_colorize, 648
x_connect4, 648
x_crop, 648
x_cut, 648
x_fire, 648
x_fireworks, 648
x_fisheye, 648
x_fourier, 648
x_grab_color, 649
x_jawbreaker, 649
x_terrains, 649
x_jawbreaker, 649
x_landscape, 649
x_light, 649
x_mandelbrot, 649
x_mask_color, 650
x_metaballs3d, 650
x_minesweeper, 650
x_minimal_path, 650
x_morph, 650
x_pacman, 650
x_paint, 650
x_plasma, 651
x_quantize_rgb, 651
x_reflection3d, 651
x_rubber3d, 651
x_segment, 651
x_select_color, 651
x_select_function1d, 652
x_select_palette, 652
x_shadebobs, 652
x_slope, 652
x_starfield3d, 652
x_tetris, 652
x_threshold, 652
x_tictactoe, 653
x_warp, 653
x_waves, 653
x_whirl, 653
xor, 132
xyz2jzazbz, 220
xyz2lab, 220
xyz2rgb, 220
xyz2rgba, 221
ycbcr2rgb, 221
yinyang, 424
yiq2rgb, 221
yiq82rgb, 221
yuv2rgb, 221
yuv82rgb, 221
zoom, 598
End of document.