



# The Handbook

Version 1.5.7.1 (beta)

**David Tschumperlé**

September 16, 2013



# Contents

<b>1 Usage</b>	<b>7</b>
1.1 Overall context . . . . .	7
1.2 Image definition and terminology . . . . .	7
1.3 Items of a processing pipeline . . . . .	8
1.4 Input data items . . . . .	8
1.5 Command items and selections . . . . .	9
1.6 Inputs / outputs properties . . . . .	10
1.7 Substitution rules . . . . .	11
1.8 Mathematical expressions . . . . .	13
1.9 Image and data viewers . . . . .	15
1.10 Adding custom commands . . . . .	15
1.11 List of commands . . . . .	16
<b>2 List of commands</b>	<b>17</b>
2.1 Global options . . . . .	17
2.2 Inputs / outputs . . . . .	17
2.3 List manipulation . . . . .	36
2.4 Mathematical operators . . . . .	41
2.5 Values manipulation . . . . .	81
2.6 Colors manipulation . . . . .	103
2.7 Geometry manipulation . . . . .	125
2.8 Filtering . . . . .	153
2.9 Features extraction . . . . .	197
2.10 Image drawing . . . . .	214
2.11 Matrix computation . . . . .	239
2.12 3d rendering . . . . .	243
2.13 Program controls . . . . .	289
2.14 Arrays, tiles and frames . . . . .	299
2.15 Artistic . . . . .	311
2.16 Warpings . . . . .	331
2.17 Degradations . . . . .	342
2.18 Blending and fading . . . . .	349
2.19 Image sequences . . . . .	356

2.20 Interactive demos . . . . .	360
2.21 PINK-library operators . . . . .	362
2.22 Convenience functions . . . . .	370
2.23 Others . . . . .	379
2.24 Commands shortcuts . . . . .	380
2.25 Examples of use . . . . .	382

# Preamble

## License

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

An online version of this documentation is available at:  
<http://gmic.sourceforge.net/reference.shtml>.

## Motivations

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

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

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

datasets.

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

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

## Version

gmic: GREYC's Magic for Image Computing.

Version 1.5.7.1 (beta), Copyright (C) 2008-2013, David Tschumperlé  
(<http://gmic.sourceforge.net>)

# Chapter 1

## Usage

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

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

This documentation proposes a complete description of the G'MIC language basics and rules.

### 1.1 Overall context

- At any time, G'MIC manages one list of numbered (and optionally named) pixel-based images, entirely stored in computer memory.
- The first image of the list has indice '0' and is denoted by '[0]'. The second image of the list is denoted by '[1]', the third by '[2]' and so on.
- Negative indices are treated in a cyclic 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 denoted by '[name]' if 'name' uses characters set [a-zA-Z0-9\_] and does not start with a number. Image names can be set or reassigned at any moment during the processing pipeline (see commands '--name' and '--input').
- G'MIC defines a set of various commands and substitution mechanisms to allow the design of complex pipelines managing this list of images, in a very flexible way:  
You can insert or remove images in the list, rearrange image indices, process images (individually or as a group), merge image data together and output image files.
- Such a pipeline can be written itself as a custom G'MIC command storable in a custom commands file, which can be re-used afterwards in another bigger pipeline if necessary.

### 1.2 Image definition and terminology

- In G'MIC, an 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 these arrays 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 2d color or grayscale 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 size limitations on each image dimensions. Particularly, the number of image slices or channels can be of arbitrary size within the limits of available memory.
- The width, height and depth of an image are considered as 'spatial' dimensions, while the spectrum has a 'multi-spectral' meaning. Thus, a 4d image in G'MIC should be most often regarded as a 3d dataset of multi-spectral voxels. Most of the G'MIC commands will stick with this idea (e.g. command '--blur' will blur images only along the 'xyz' axes).
- All pixel values of all images of the list have the same datatype. It can be one among:
  - . 'bool': Stands for 'boolean'. Value range is { 0=false | 1=true }.
  - . 'uchar': Stands for 'unsigned char'. Value range is [0,255] (8bits).
    - This type of pixel coding is commonly used to store 8bits/channels RGB[A] images.
  - . 'char': Value range is [-128,127] (8bits).
  - . 'ushort': Stands for 'unsigned short'. Value range is [0,65535] (16bits).
    - This type of pixel coding is commonly used to store 16bits/channels RGB[A] images.
  - . 'short': Value range is [-32768,32767] (16bits).
  - . 'uint': Stands for 'unsigned int'. Value range is [0,2^32-1] (32bits).
  - . 'int': Value range is [-2^31,2^31-1] (32 bits).
  - . 'float': Value range is [-3.4E38,+3.4E38] (32bits).
    - This type of coding is able to store pixels as 32 bits float-valued numbers. This is the default datatype used by G'MIC image processing operations.
  - . 'double': Value range is [-1.7E308,1.7E308] (64bits).
    - This type of coding is able to store pixels as 64 bits float-valued numbers.
- Considering pixel datatypes different than 'float' is generally useless, except to force the input/output of image data to a prescribed binary format. Hence, most G'MIC image processing commands are available only for the default 'float' pixel datatype (see command '--type' if you need to switch to another pixel datatype).

### 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:  
`'input.jpg -blur 3,0 -sharpen 10 -resize 200%,200% -output output.jpg'`  
defines a valid pipeline composed of nine G'MIC items.
- A G'MIC item is a string which represents either a command, a set of command arguments, a filename, or a special input string.
- Escape characters '\ ' and double quotes "" can be used (as usual) to define items containing spaces, or any other character sequences. For instance, the strings  
`'single\ item'` and  
`""single item""` define the same string item, with a space in it.

### 1.4 Input data items

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

- Special filenames ‘–’ and ‘–.ext’ stand for the standard input/output streams, optionally forced to be in a specific ‘ext’ file format (e.g. ‘–.jpg’ or ‘–.png’).
- The following special input strings may be used as G’MIC items to create and insert new images with prescribed values, at the end of the image list:
  - . ‘[selection]’ or ‘[selection]xN’: Insert 1 or N copies of selected existing images. ‘selection’ may contain one or several images (see next section for details).
  - . ‘width[%],\_height[%],\_depth[%],\_spectrum[%],\_values’: Insert a new image with specified size and values (adding ‘%’ to a dimension means ‘percentage of the size along the same axis, taken from the last image ’[−1]’’). Any specified dimension can be also written as ‘[image]’, and is then set to the size (along the same axis) of the existing specified image [image]. ‘values’ can be either a sequence of numbers separated by commas ‘,’, or a mathematical expression, as e.g. in input item ‘256,256,1,3,if(c==0,x,if(c==1,y,0))’ which creates a 256x256 RGB color image with a spatial shading on the red and green channels.
  - . ‘(v1,v2,...)’: Insert a new image from specified prescribed values. Value separator inside parentheses can be ‘,’ (column separator), ‘;’ (row sep.), ‘/’ (slice sep.) or ‘^’ (channel sep.). For instance, expression ‘(1,2,3;4,5,6;7,8,9)’ creates a 3x3 matrix (scalar image), with values from 1 to 9.
  - . ‘0’: Insert a new ‘empty’ image, containing no pixel data. Empty images are used only in rare occasions.
- Input item ‘name=value’ declares a new local or global variable ‘name’, or assign a new value to an existing variable. Variable names use characters set [a–zA–Z0–9\_] and cannot start with a number. A variable definition is always local to the current command except when it starts by the underscore character ‘\_’. In that case, it becomes also accessible by any command invoked outside the current command scope.

## 1.5 Command items and selections

- A G’MIC item starting by ‘–’ designates a command, most of the time. Generally, commands perform image processing operations on one or several available images of the list.
- Common 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, if any required.
- The execution of a G’MIC command may be restricted only to a subset of the image list, by appending ‘[subset]’ to the command name. Examples of valid syntaxes for ‘subset’ are:
  - . ‘–com[0,1,3]’: Apply command only on images [0],[1] and [3].
  - . ‘–com[3–5]’: Apply command only on images [3] to [5] (i.e, [3],[4] and [5]).
  - . ‘–com[50%–100%]’: Apply command only on the second half of the image list.
  - . ‘–com[0,–4––1]’: Apply command only on the first and the four latest images.
  - . ‘–com[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]).
  - . ‘–com[0––1:2]’: Apply command only on images of the list with even indices.
  - . ‘–com[0,2–4,50%––1]’: Apply command on images [0],[2],[3],[4] and on the second half of the image list.
  - . ‘–com[^0,1]’: Apply command on all images except the first two.
  - . ‘–com[name1,name2]’: Apply command on named images ‘name1’ and ‘name2’.
- Indices in selections are always sorted in increasing order, and duplicate indices are

discarded. For instance, selections '[3–1,1–3]' and '[1,1,1,3,2]' are both equivalent to '[1–3]'. If you want to repeat a single command multiple times on an image, use a '--repeat..--done' loop. Inverting the order of images in a selection can be achieved by inverting first the order of the images in the list, with command '--reverse[selection]'.

- G'MIC commands invoked without '[subset]' are applied on all images of the list.
- A G'MIC command starting with '--' instead of '-' does not act 'in-place' but inserts its result as one or several new images at the end of the image list.
- There are two different types of commands that can be run by the G'MIC interpreter:
  - . Native commands, are hard-coded functionalities in the interpreter core.  
They are thus compiled as machine code and run quickly, most of the time.  
Omitting an argument when invoking a native command is not permitted, except if all following arguments are also omitted. For instance, call to '--plasma 10,,5' is invalid but '--plasma 10' is correct.
  - . Custom commands, are defined as G'MIC pipelines of native or custom commands.  
They are interpreted by the G'MIC interpreter, and run slower than native commands.  
But omitting arguments when invoking a custom command is permitted. For instance, expressions '--flower ,,,100,,2' or '--flower ,' are correct.
- A user may easily add its own custom commands to the G'MIC interpreter (see section 'Adding custom commands'). Native commands cannot be added unless you modify the G'MIC interpreter source code.

## 1.6 Inputs / outputs 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, ..
  - . Image sequences: .mpeg, .avi, .mov, .ogg, .flv, ..
  - . Generic ascii or binary data files: .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 RGB color space.
- G'MIC is able to manage 3d objects that may be read from files or generated by G'MIC commands. They are stored as one-column scalar images containing the object data, in the following order: { magic\_number; sizes; vertices; primitives; colors; opacities }. These 3d representations can be processed as regular float-valued 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 coding of image pixels. For instance, saving an image that was initially loaded as a 16bits/channel image, as a .jpg file will result in loss of informations. Use the .cimg file extension (or .cimgz, its compressed version) to ensure that all data precision will be preserved when saving images.
- File options can/must be set for these specific file formats:
  - . Video files: Only sub-frames of an image sequence may be loaded, using the input expression 'filename.ext,[first\_frame[%][,last\_frame[%][,step]]]'. Output framerate and bitrate (in Kb/s) can be also set by using the output expression 'file.mpg,\_fps,\_bitrate'.
  - . raw binary files: Image dimensions and input pixel type may be specified when loading .raw files with input expression 'filename.raw[,type][,width][,height[,depth[,dim]]]'. If no dimensions are specified, the resulting image is a one-column vector with

- maximum possible height. Pixel type can also be specified with the output expression 'file.raw[,type]'.  
 'type' can be { bool | uchar | char | ushort | short | uint | int | float | double }.
- . .yuv files: Image dimensions must be specified, and only sub-frames of an image sequence may be loaded, using the input expression 'filename.yuv,width,height[,first\_frame[,last\_frame[,step]]]'.
  - . .tiff files: Only sub-images of multi-pages tiff files can be loaded, using the input expression 'filename.tif,[first\_frame,[last\_frame,[step]]]'.
  - . .gif files: Animated gif files can be saved, using the input expression 'filename.gif,fps,nb\_loops'. Specify 'nb\_loops=0' to get an infinite number of animation loops.
  - . .jpeg files: The output quality may be specified (in %), using the output expression 'filename.jpg,30' (here, to get a 30% quality output).
  - . .mnc files: The output header can set from another file, using the output expression 'filename.mnc,header\_template.mnc'.
  - . Filenames with extension '.gmic' are assumed to be G'MIC custom commands files. Loading such a file will add the commands it defines to the interpreter.
  - . Inserting 'ext:' on the beginning of a filename (e.g. 'jpg:filename') forces G'MIC to read/write the file as it would have been done if it had the specified extension.
  - Some input/output formats and options may not be supported by your current version of 'gmic', depending on the configuration flags set for the build of the 'gmic' binaries.

## 1.7 Substitution rules

- G'MIC items containing '@', '\$' or '{}' may be substituted before being interpreted. Use the substituting expressions below to access data from the interpreter environment:
- . '@#' is substituted by the current number of images in the list.
- . '@\*' is substituted by the number of available cpus.
- . '@.' is substituted by the current version number of the G'MIC interpreter
- . '@^' is substituted by the current verbosity level.
- . '@%' is substituted by the pid of the current process.
- . '@|' is substituted by the current value (expressed in seconds) of a millisecond precision timer.
- . '@?' is substituted by the current data type of image pixels.
- . '@/' is substituted by the current number of levels in the command scope.
- . '@{/}' or '@{/subset}' are substituted by the content of the global scope, or a subset of it. If specified subset refers to multiple scope items, they are separated by slashes '/'.
- . '@>' and '@<' are equivalent. They are both substituted by the number of nested 'repeat-done' loops that are currently running.
- . '@{>}' or '@{>,subset}' are substituted by the indice values (or a subset of them) of the running 'repeat-done' loops, expressed in the ascending order, starting from 0. If specified subset refers to multiple indices, they are separated by commas ','.
- . '@{<}' or '@{<,subset}' do the same but in descending order.
- . '@indice' or '@{indice,feature}' are substituted by the list of pixel values of the image [indice] (separated by commas), or by a specific feature (or subset) of it. 'indice' can be an indice or an image name. Requested 'featured' can be one of:
  - . 'w': image width (number of image columns).
  - . 'h': image height (number of image rows).

- . 'd': image depth (number of image slices).
- . 's': image spectrum (number of image channels).
- . 'wh': image width x image height.
- . 'whd': image width x image height x image depth.
- . 'whds': image width x image height x image depth x image spectrum.  
(i.e. number of values in the specified image, eq. to '#').
- . 'r': image shared state (1, if the pixel buffer is shared, 0 otherwise).
- . 'n': image name or filename (if the image has been read from a file).
- . 'b': image basename (i.e. filename without the folder path nor extension).
- . 'x': image extension (i.e last characters after the last '.' in the filename).
- . 'f': image folder name.
- . '#': number of image values (i.e. width x height x depth x spectrum).
- . '+': sum of all pixel values.
- . '-': difference of all pixel values.
- . '\*': product of all pixel values.
- . '/': quotient of all pixel values.
- . 'm': minimum pixel value.
- . 'M': maximum pixel value.
- . 'a': average pixel value.
- . 'v': variance of pixel values.
- . 't': text string built from the image values, regarded as ascii codes.
- . 'c': (x,y,z,c) coordinates of the minimum value, separated by commas ','.
- . 'C': (x,y,z,c) coordinates of the maximum value, separated by commas ','.
- . '(x[%],\_y[%],\_z[%],\_c[%],\_boundary)': pixel value at (x[%],y[%],z[%],c[%]), with  
specified boundary conditions { 0=dirichlet | 1=neumann | 2=cyclic }.
- . Any other 'feature' is considered either as a specified subset of image values, or  
as a mathematical expression to evaluate (associated to selected image).  
For instance, '@{-1,0-50%}' is substituted by the sequence of numerical values  
coming from the first half data of the last image, separated by commas ','.  
Expression '@{0,w+h}' is substituted by the sum of the width and height of the  
first image.
- . '@!' is substituted by the visibility state of the instant display window [0]  
(can be { 0=closed | 1=visible }).
- . '@{!,feature}' or '@{!indice,feature}' is substituted by a specific feature of the  
instant display window [0] (or [indice], if specified). Requested 'feature' can be:
  - . 'w': display width (i.e. width of the display area managed by the window).
  - . 'h': display height (i.e. height of the display area managed by the window).
  - . 'wh': display width x display height.
  - . 'd': window width (i.e. width of the window widget).
  - . 'e': window height (i.e. height of the window widget).
  - . 'de': window width x window height.
  - . 'u': screen width (actually independent on the window size).
  - . 'v': screen height (actually independent on the window size).
  - . 'uv': screen width x screen height.
  - . '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.

- . 'n': current normalization type of the instant display.
- . '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 }.
- . '@{"command line"}' is substituted by the status value set by the execution of the specified command line (see command '--status').
- . Expression '@{}' stands thus for the current status value.
- '\$name' and '\${name}' are both substituted by the value of the specified named variable (set previously by item 'name=value'), or by the current positive indice of the named image '[name]', or by the value of the named OS environment variable (in this order).
- '\$>' and '\$<' (resp. '\${>}' and '\${<}') are shortcuts respectively for '@{>,-1}' and '@{<,-1}'. They refer to the increasing/decreasing indice of the latest (currently running) 'repeat..done' loop.
- Any other expression inside braces (as in '{expression}') is considered as a mathematical expression, and is evaluated, except for the three following cases:
  - . If expression starts and ends by single quotes, it is substituted by the sequence of ascii codes that composes the specified string, separated by commas ','. For instance, item '{'foo'}' is substituted by '102,111,111'.
  - . If expression starts and ends with backquotes ``', it is substituted by the string whose ascii codes are given by the list of values in between the backquotes. For instance, item '{`102,111,111`} is substituted by 'foo'.
  - . If expression contains operator "==" or "!=", it is substituted by 0 or 1, whether the strings beside the operator are the same or not (case-sensitive). For instance, both items '{foo==foo}' and '{foo!=FOO}' are substituted by '1'.
  - . If expression starts with an underscore '\_', it is substituted by the mathematical evaluation of the expression, truncated to a readable format.
- Item substitution is never done in items between double quotes. One must break the quotes to enable substitution if needed, as in "3+8 kg =" "{3+8}" kg". Using double quotes is then a convenient way to disable the substitutions mechanism in items, when necessary.
- One can also disable the substitution mechanism on items outside double quotes, by escaping the '@','{,'}' or '\$' characters, as in '\{3+4\}\ doesn't\ evaluate'.

## 1.8 Mathematical expressions

- G'MIC has an embedded mathematical parser. It is used to evaluate expressions inside braces '{}', or formulas in commands that may take one as an argument (e.g. '--fill').
- When used in commands, a formula is evaluated for each pixel of the selected images.
- The math parser understands the following set of functions, operators and variables:
  - \_ Usual operators: || (logical or), && (logical and), | (bitwise or), & (bitwise and), !=, ==, <=, >=, <, >, << (left bitwise shift), >> (right bitwise shift), -, +, \*, /, % (modulo), ^ (power), ! (logical not), ~ (bitwise not).
  - \_ Usual functions: sin(), cos(), tan(), asin(), acos(), atan(), sinh(), cosh(), tanh(), log(), log2(), log10(), exp(), sign(), abs(), atan2(), round(), narg(), arg(), isval(), isnan(), isinf(), isint(), isbool(), rol() (left bit rotation), ror() (right bit rotation), min(), max(), sinc(), int().
- Function 'atan2()' is the version of atan() with two arguments 'y,x' (as in C/C++).
- Function 'narg()' returns the number of specified arguments.

Function 'arg(i,a\_1,...,a\_n)' returns the ith argument a\_i.

Functions 'min()' and 'max()' can be called with an arbitrary number of arguments.

Functions 'isval()', 'isnan()', 'isinf()', 'isbool()' can be used to test the type of a given number or expression.

- The variable names below are pre-defined. They cannot be overloaded:

- . 'w': width of the associated image, if any (0 otherwise).
- . 'h': height of the associated image, if any (0 otherwise).
- . 'd': depth of the associated image, if any (0 otherwise).
- . 's': spectrum of the associated image, if any (0 otherwise).
- . '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).
- . 'i': current processed pixel value (i.e. value located at (x,y,z,c)) of the associated image, if any (0 otherwise).
- . 'im','iM','ia','iv': Respectively the minimum, maximum, average values and variance 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).
- . 'pi': value of pi, i.e. 3.1415926..
- . 'e': value of e, i.e. 2.71828..
- . '?' or 'u': a random value between [0,1], following a uniform distribution.
- . 'g': a random value, following a gaussian distribution of variance 1 (roughly in [-5,5]).

- These special operators can be used:

- . ';': expression separator. The returned value is always the last encountered expression. For instance expression '1;2;pi' is evaluated as 'pi'.
- . '=': variable assignment. Variables in mathematical parser can only refer to numerical values. Variable names are case-sensitive. Use this operator in conjunction with ';' to define complex evaluable expressions, such as  
't=cos(x);3\*t^2+2\*t+1'.

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

- The following specific functions are also defined:

- . 'if(expr\_cond,expr\_then,expr\_else)': return value of 'expr\_then' or 'expr\_else', depending on the value of 'expr\_cond' (0=false, other=true). For instance, G'MIC command '-fill if(x%10==0,255,i)' will draw blank vertical lines on every 10th column of an image.
- . '?(max)' or '?(min,max)': return a random value between [0,max] or [min,max], following a uniform distribution. 'u(max)' and 'u(0,max)' mean the same.
- . 'i(\_a,\_b,\_c,\_d,.interpolation,.boundary)': return the value of the pixel located at position (a,b,c,d) in the associated image, if any (0 otherwise). Interpolation parameter can be { 0=nearest neighbor | other=linear }. Boundary conditions can be { 0=dirichlet | 1=neumann | 2=cyclic }. Omitted coordinates are replaced by their default values which are respectively x, y, z, c and 0.
- . 'j(\_dx,\_dy,\_dz,\_dc,.interpolation,.boundary)': does the same for the pixel located

at position  $(x+dx, y+dy, z+dz, c+dc)$ .

For instance command '`-fill 0.5*(i(x+1)-i(x-1))`' will estimate the X-derivative of an image with a classical finite difference scheme.

- . If specified formula starts with '`>`' or '`<`', the operators '`i(..)`' and '`j(..)`' will return values of the image currently being modified, in forward ('`>`') or backward ('`<`') order.
- The last image of the list is always associated to the evaluations of '`{expressions}`', e.g. G'MIC sequence '`256,128 -f {w}`' will create a 256x128 image filled with value 256.

## 1.9 Image and data viewers

- G'MIC has some very handy embedded visualization modules, for 1d signals (command '`-plot`'), 1d/2d/3d images (command '`-display`') and 3d objects (command '`-display3d`'). It enables an interactive view of the selected image data.
- The following keyboard shortcuts are available in the interactive viewers:
  - . `CTRL+D`: Increase window size.
  - . `CTRL+C`: Decrease window size.
  - . `CTRL+R`: Reset window size.
  - . `CTRL+F`: Toggle fullscreen mode.
  - . `CTRL+S`: Save current window snapshot as numbered file '`gmic_xxxx.bmp`'.
  - . `CTRL+O`: Save current instance of the viewed data, as numbered file '`gmic_xxxx.cimgz`'.
- Shortcuts specific to the 1d/2d/3d image viewer are:
  - . `CTRL+P`: Play z-stack of frames as a movie (for volumetric 3d images).
  - . `CTRL+V`: Enable/disable 3D view (for volumetric 3d images).
  - . `CTRL+(mousewheel)`: Zoom in/out.
  - . `SHIFT+(mousewheel)`: Go left/right.
  - . `ALT+(mousewheel)`: Go up/down.
  - . Numeric PAD: Zoom in/out (+/-) and move through zoomed image (digits).
  - . `BACKSPACE`: Reset zoom scale.
- Shortcuts specific to the 3d object viewer are:
  - . `(mouse)+(left mouse button)`: Rotate 3d object.
  - . `(mouse)+(right mouse button)`: Zoom 3d object.
  - . `(mouse)+(middle mouse button)`: Shift 3d object.
  - . `(mousewheel)`: Zoom in/out.
  - . `CTRL+F1 .. CTRL+F6`: Switch between different 3d rendering modes.
  - . `CTRL+Z`: Enable/disable z-buffered rendering.
  - . `CTRL+A`: Show/hide 3d axes.
  - . `CTRL+G`: Save 3d object, as numbered file '`gmic_xxxx.off`'.
  - . `CTRL+T`: Switch between single/double-sided 3d modes.

## 1.10 Adding custom commands

- Custom commands can be defined by a user, through the use of G'MIC custom commands files.
- A command file is a simple ascii text file, where each line starts either by '`command.name: command.definition`' or '`command.definition (continuation)`'.
- Custom command names must use characters [a-zA-Z0-9.] and cannot start with a number.
- Any '`# comment`' expression found in a custom commands file is discarded by the G'MIC interpreter, wherever it is located in a line.
- In custom commands, the following \$-expressions are substituted:
  - . '`$*`' is substituted by a verbatim copy of the specified arguments string.

- . '\$#' is substituted by the maximum indice of known arguments (either specified by the user or set to a default value in the custom command).
- . '\$?' is substituted by a string telling about the command subset restriction (only useful when custom commands need to output descriptive messages).
- . '\$i' and '\${i}' are both substituted by the i-th specified argument. Negative indices such as '\${-j}' are allowed and refer to the j'th latest argument. '\$0' is substituted by the custom command name.
- . '\${i=default}' is substituted by the value of \$i (if defined) or by its new value set to 'default' otherwise ('default' may be a \$-expression as well).
- . '\${subset}' is substituted by the arguments values (separated by commas ',') of a specified argument subset. For instance expression '\${2--2}' is substituted by all specified arguments except the first and the last one. Expression '\${^0}' is then substituted by all arguments of the invoked command (eq. to '\$\*' if all specified arguments have indeed a value).
- . '\$=var' is substituted by the set of instructions that will assign each argument \$i to the named variable 'var\$*i*' (for i in [0..\$#]). This is particularly useful when a custom command want to manage variable numbers of arguments. Variables names must use characters [a-zA-Z0-9\_] and cannot start with a number.
- These particular \$-expressions are always substituted, even in double quoted items or when the dollar sign '\$' is escaped with a backslash '\\'. To avoid substitution, place an empty double quoted string just after the '\$' (as in '\$'''1').
- Specifying arguments may be skipped when invoking a custom command, by replacing them by commas ',', as in expression '--flower „3''. Omitted arguments are set to their default values, which must be thus explicitly defined in the code of the corresponding custom command (using default argument expressions as '\${1=default}').
- If one numbered argument requested in a custom command has no value, an error is thrown by the interpreter.

## 1.11 List of commands

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

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

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

In this case, the '[]' characters are mandatory when writing the item. A command marked with '(\*)' or '(+)' is a native command. '(\*)' means the command is available for all pixel types, otherwise only for the default 'float' pixel type.

Remember that native commands run faster than custom commands, so use them when possible.

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

# 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 additionnal log messages about its internal state on the standard output (stdout).

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

#### 2.1.2 *-help* (\*)

**Arguments:** `_command |  
(no args)`

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

#### 2.1.3 *-version*

Display current version number and exit.

### 2.2 Inputs / outputs

#### 2.2.1 *-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 'filename=""'.

### 2.2.2 -camera (\*)

**Arguments:** \_camera\_index>=0, \_nb\_frames>0, \_skip\_frames>=0, release\_camera={0 | 1}, \_capture\_width>=0, \_capture\_height>=0

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

When 'release\_camera==1', the camera stream is released instead of capturing new images.

**Default values:** 'camera\_index=0' (default camera), 'nb\_frames=1', 'skip\_frames=0', 'release\_camera=0' and 'capture\_width=capture\_height=0' (default size).

### 2.2.3 -command (\*)

**Arguments:** filename |  
http[s]://URL |  
"string"

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

(eq. to '-m').

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



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

### 2.2.4 -display (+)

**Arguments:** \_X, \_Y, \_Z

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

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

### **2.2.5 -display0**

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

### **2.2.6 -display3d (+)**

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

### **2.2.7 -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`' .

### **2.2.8 -display\_fft**

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



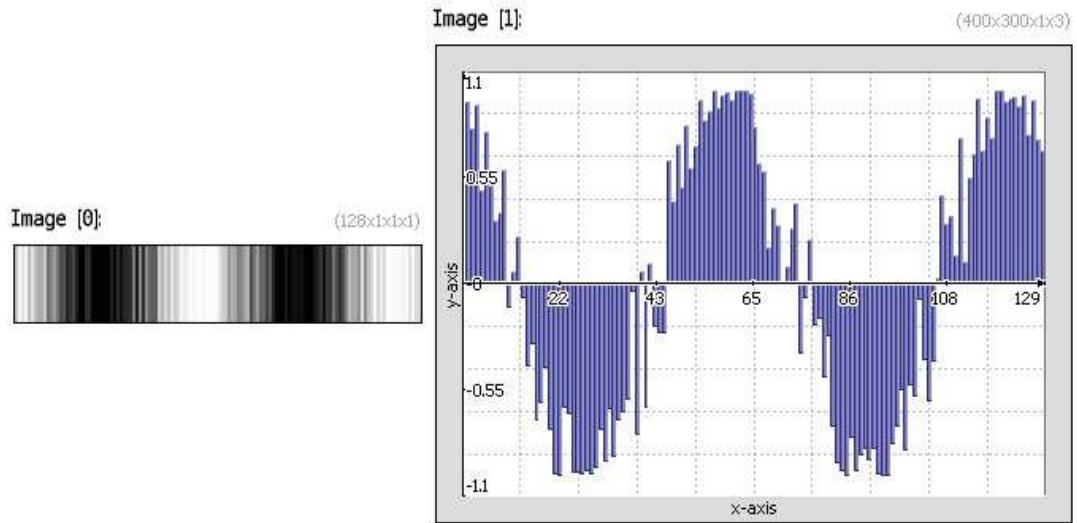
**Example 2 :** `image.jpg --display_fft`

### **2.2.9 -display\_graph**

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

Render graph plot from selected image data.

**Default values:** '`width=640`', '`height=480`', '`plot_type=1`', '`vertex_type=1`', '`xmin=xmax=ymin=ymax=0`', '`xlabel="x-axis"`' and '`ylabel="y-axis"`' .



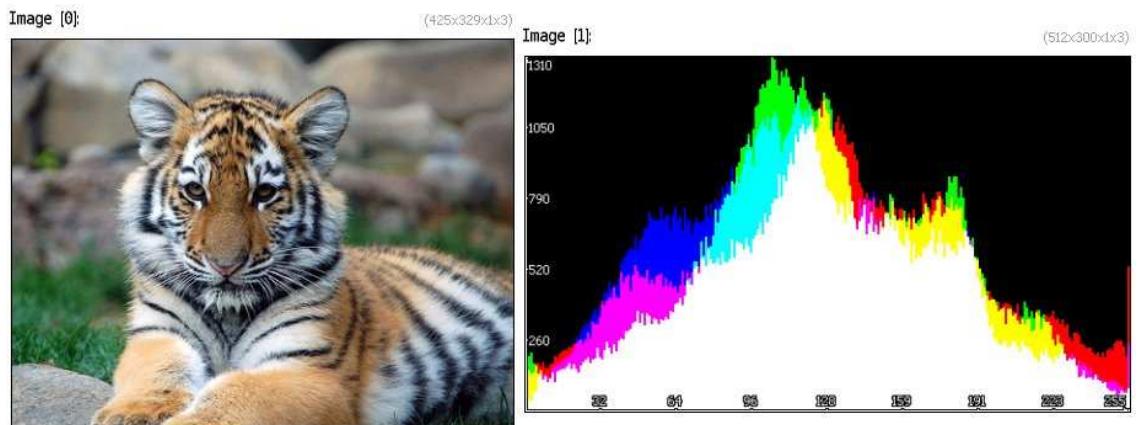
**Example 3:** `128,1,1,1,'cos(x/10+?)' --display_graph 400,300,3`

### 2.2.10 *-display\_histogram*

**Arguments:** `_width>0, _height>0, _clusters>0, _min_value[%], _max_value[%], _show_axes={0 | 1}`

Render a channel-by-channel histogram.  
(*eq. to '-dh'*).

**Default values:** '`width=512', 'height=300', 'clusters=256', 'min_value=0%', 'max_value=100%`' and '`show_axes=1`'.



**Example 4:** `image.jpg --display_histogram 512,300`

### 2.2.11 -display\_polar

**Arguments:** `_width>32, _height>32, _outline_type, _fill_R, _fill_G, _fill_B, _theta_start, _theta_end`

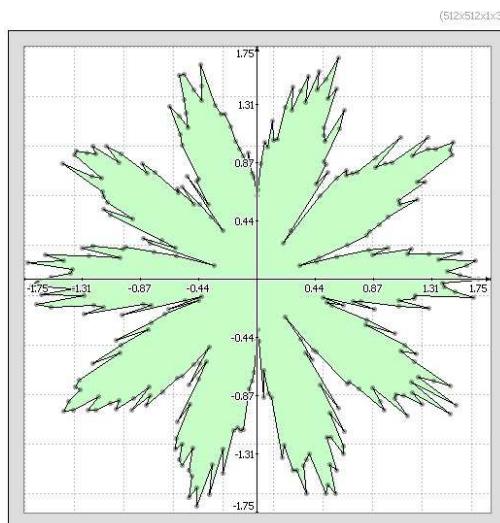
Render polar curve from selected image data.

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

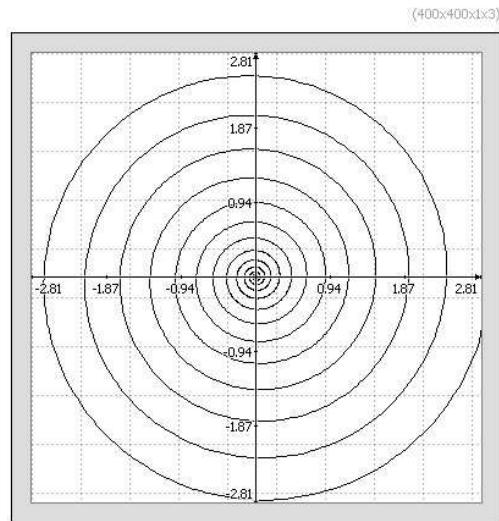
'outline\_type' can be { `r<0=dots with radius -r` | `0=no outline` | `r>0=lines+dots with radius r` }.

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

**Default values:** `'width=500', 'height=width', 'outline_type=1', 'fill_R=fill_G=fill_B=200', 'theta_start=0' and 'theta_end=360'`.



**Example 5 :** `300,1,1,1,'0.3+abs(cos(10*pi*x/w))+?(0.4)' -display_polar  
512,512,4,200,255,200`



**Example 6 :** `3000,1,1,1,'x^3/1e10' -display_polar 400,400,1,-1,,,0,{15*360}`

### 2.2.12 *-display\_rgba*

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



**Example 7 :** `image.jpg --norm -threshold[-1] 40% -blur[-1] 3 -normalize[-1] 0,255 -append c -display_rgba`

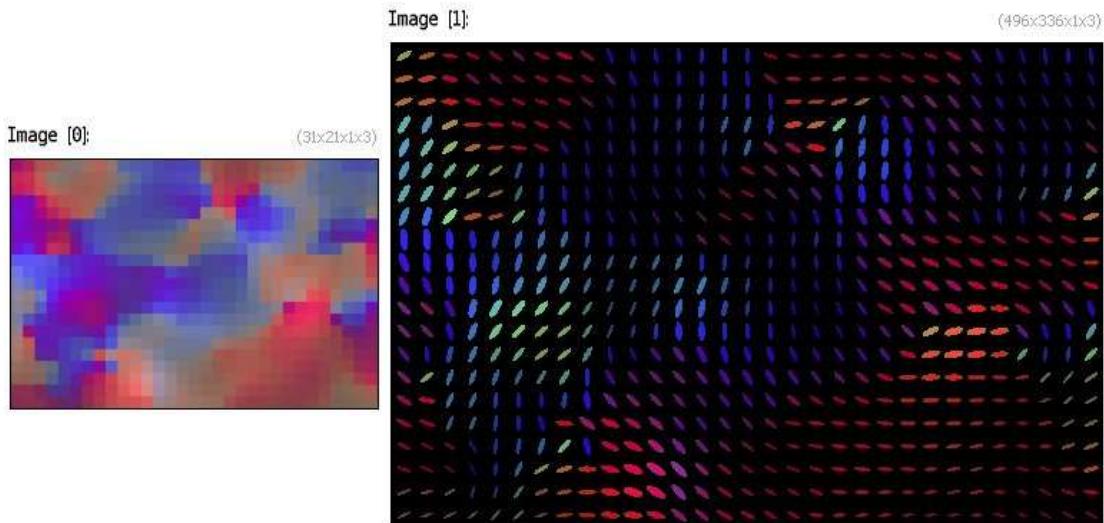
### 2.2.13 *-display\_tensors*

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

Render selected mask field of 2x2 tensors with ellipses.

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

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



**Example 8 :** `image.jpg -diffusiontensors 0.7,0.6 -crop 60,10,90,30 --display_tensors ,`

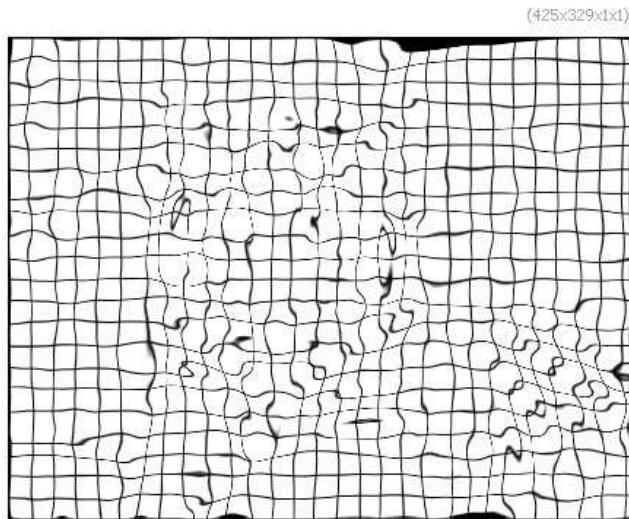
### 2.2.14 `-display_warp`

**Arguments:** `_cell_size>0`

Render selected 2d warping fields.

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

**Default value:** '`cell_size=15`'.



**Example 9 :** `image.jpg -luminance -blur 5 -gradient -append c -display warp ,`

### 2.2.15 *-document\_gmic*

**Arguments:** `-format={ ascii | html | latex | xml | bash | images }`, `-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) :** `raw:file.gmic,uchar -document_gmic html,img`

### 2.2.16 *-echo (\*)*

**Arguments:** `message`

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

Command subset (if any) stands for displayed scope indices instead of image indices.

### 2.2.17 *-echo\_file*

**Arguments:** `filename,message`

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

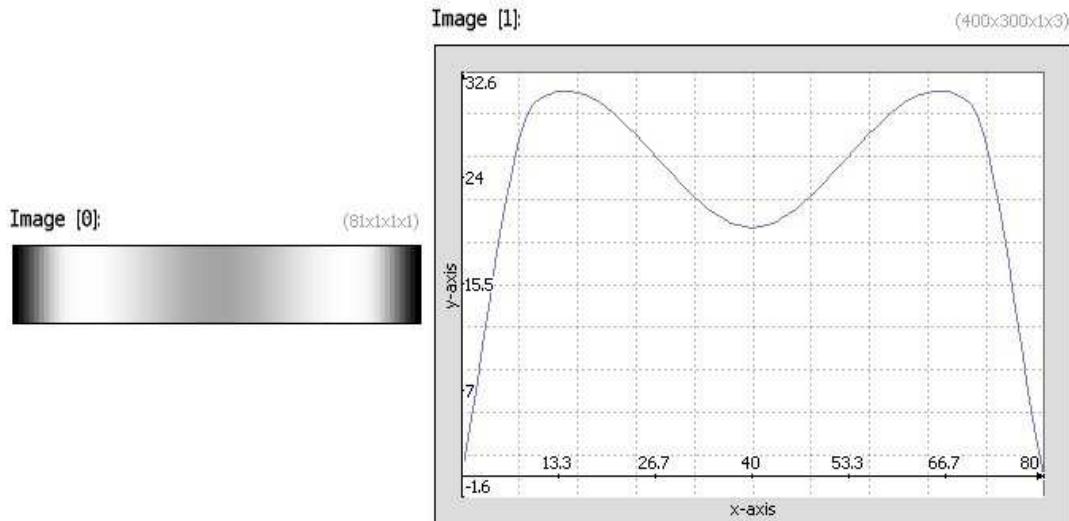
**2.2.18 -echo\_stdout****Arguments:** message

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

**2.2.19 -function1d****Arguments:** 0<=smoothness<=1, x0>=0, y0, x1>=0, y1, ..., xn>=0, yn

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

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



**Example 10 :** -function1d 1,0,0,10,30,40,20,70,30,80,0 --display\_graph 400,300

**2.2.20 -gmicky**

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

**Example 11 :** -gmicky

### 2.2.21 -gmicky\_wilber

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

**Example 12 :** -gmicky\_wilber

### 2.2.22 -input (\*)

**Arguments:** [type:]filename |  
 [type:]http://URL |  
 [selection]x\_nb\_copies>0 |  
 { width>0[%] | [image\_w] }, { \_height>0[%] | [image\_h] } , { \_depth>0[%] | [image\_d] }, { \_spectrum>0[%] | [image\_s] }, {

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

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

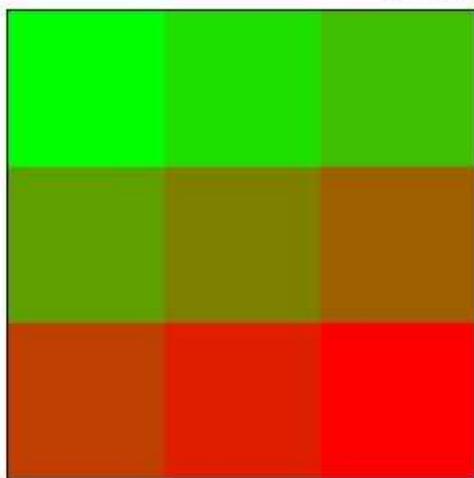
(*e.g. to '-i'* | (*no args*).

**Default values:** 'nb\_copies=1', 'height=depth=spectrum=1' and 'value1=0'.

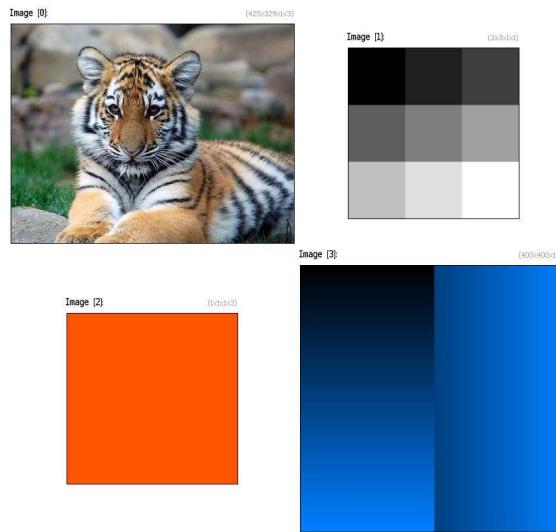


**Example 13 :** -input image.jpg

(3x3x1x2)



**Example 14 :** -i (1,2,3;4,5,6;7,8,9^9,8,7;6,5,4;3,2,1)



**Example 15 :** `image.jpg (1,2,3;4,5,6;7,8,9) (255^128^64)  
400,400,1,3,'if(x>w/2,x,y)*c'`

### 2.2.23 `-output` (\*)

**Arguments:** [type:]filename,\_format\_options

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

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

### 2.2.24 `-outputn`

**Arguments:** filename

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

### 2.2.25 `-outputp`

**Arguments:** prefix

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

**Default value:** '`prefix=_`' .

**2.2.26 -outputw**

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

**2.2.27 -plot (+)**

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

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

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

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

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

**Default values:** 'plot\_type=1', 'vertex\_type=1' and 'xmin=xmax=ymin=ymax=0 (auto)' .

**2.2.28 -print (\*)**

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

**2.2.29 -rainbow\_lut**

Input a 256-entries RGB colormap of rainbow colors.



**Example 16 :** `image.jpg -rainbow_lut --luminance[-2] -map[-1] [-2]`

**2.2.30 -roddy**

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



**Example 17 :** -roddy

### 2.2.31 *-remove\_duplicates*

Remove duplicates images in the selected images list.



**Example 18 :** (1, 2, 3, 4, 2, 4, 3, 1, 3, 4, 2, 1) -split x -remove\_duplicates -append x

### 2.2.32 *-remove\_empty*

Remove empty images in the selected image list.

### 2.2.33 *-select (+)*

**Arguments:** feature\_type, -X, -Y, -Z

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

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

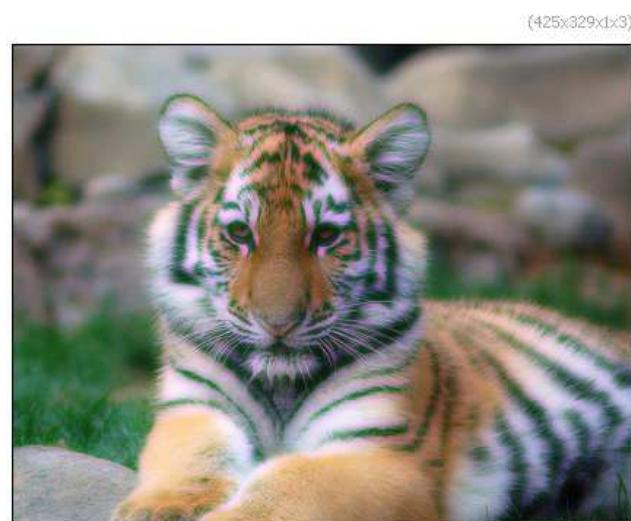
Arguments 'X','Y','Z' determine the initial selection view, for 3d volumetric images.

The retrieved feature is returned as a 3d or 6d vector containing the feature coordinates.

### 2.2.34 -shared (\*)

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

Insert shared buffers from (opt. points/rows/planes/channels of) selected images.  
(eq. to '-sh').



**Example 19 :** image.jpg --shared 1,1 -blur[-1] 3 -remove[-1]



**Example 20 :** `image.jpg -repeat {s} --shared 25%,75%,0,$> -mirror[-1] x -remove[-1] -done`

### 2.2.35 `-strand` (\*)

**Arguments:** value |  
(no args)

Set random generator seed.

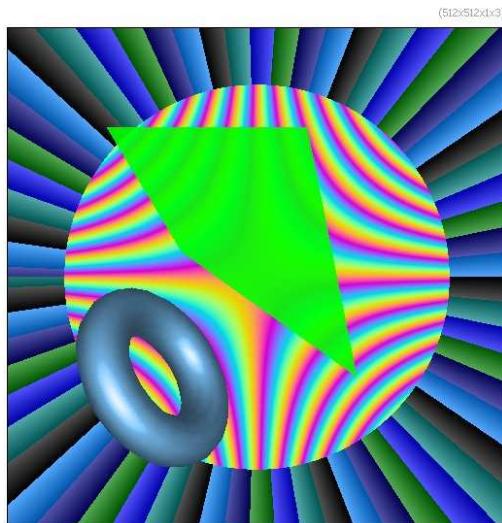
If no argument is specified, a random value is used as the random generator seed.

### 2.2.36 `-testimage2d`

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

Input a 2d synthetic image.

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



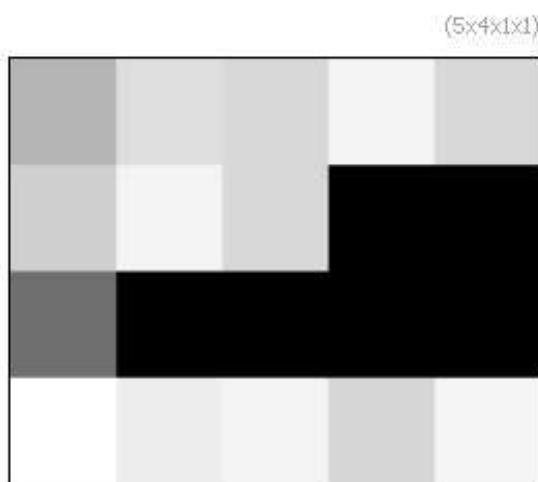
**Example 21 :** -testimage2d 512

### 2.2.37 -text2img

**Arguments:** text, line\_separator

Input a 2d image whose values are ASCII characters of specified input text.

**Default value:** 'line\_separator= '.



**Example 22 :** -text2img "There are 4 words"

**2.2.38 -type (\*)****Arguments:** datatype

Set pixel datatype for all images of the list.

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

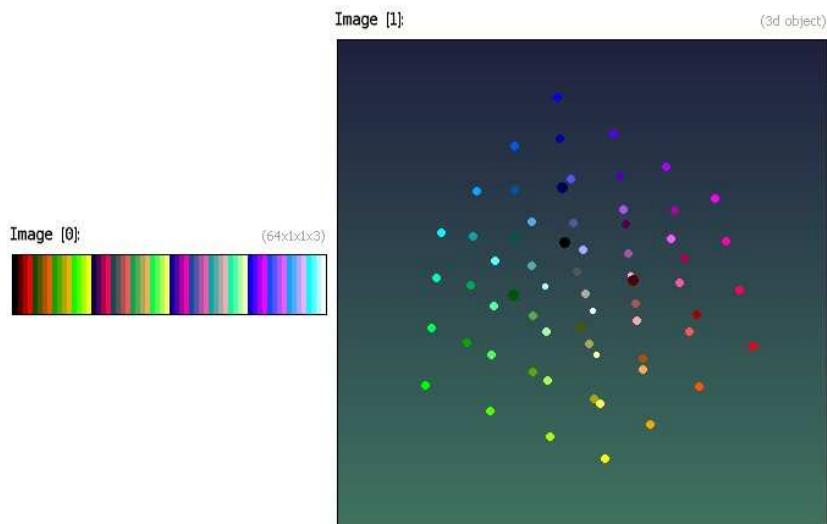
**2.2.39 -uncommand (\*)****Arguments:** command\_name | \*

Discard last definition of specified custom command.

Set argument to '\*' for discarding all existing custom commands.

**2.2.40 -uniform\_distribution****Arguments:** nb\_levels>=1, spectrum>=1

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

**Example 23 :** -uniform\_distribution 64,3 -\* 255 --distribution3d -circles3d[-1] 10**2.2.41 -update**

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

This requires an active Internet connection and an access to the external tools 'curl' or 'wget'.  
(eq. to '-up').

**2.2.42 -verbose (\*)**

**Arguments:** level |  
   { + | - }

Set or increment/decrement the verbosity level.

(eq. to '-v').

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

Default value for the verbosity level is 0.

**2.2.43 -wait (+)**

**Arguments:** delay |  
   (no args)

Wait for a given delay (in ms) or for a user event occurring on the selected instant window.

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

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

**2.2.44 -warn (\*)**

**Arguments:** message

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

Command subset (if any) stands for displayed scope indices instead of image indices.

**2.2.45 -window (+)**

**Arguments:** \_width[%]>=-1,\_height[%]>=-1,\_normalization,\_fullscreen,\_title

Display selected images into an instant window with specified size, normalization type, fullscreen mode and title.

(eq. to '-w').

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

'width'=0 or 'height'=0 closes the instant window.

'normalization' can be { -1=keep same | 0=none | 1=always | 2=1st-time | 3=auto }.

'fullscreen' can be { -1=keep same | 0=no | 1=yes }.

You can manage up to 10 different instant windows by using the numbered variants

'-w0' (default, eq. to '-w'), '-w1',...,'-w9' of the command '-w'.

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

## 2.3 List manipulation

### 2.3.1 *-keep* (\*)

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

(107x329x1x3)



**Example 24 :** `image.jpg -split x -keep[0-50%:2] -append x`  
(254x329x1x3)

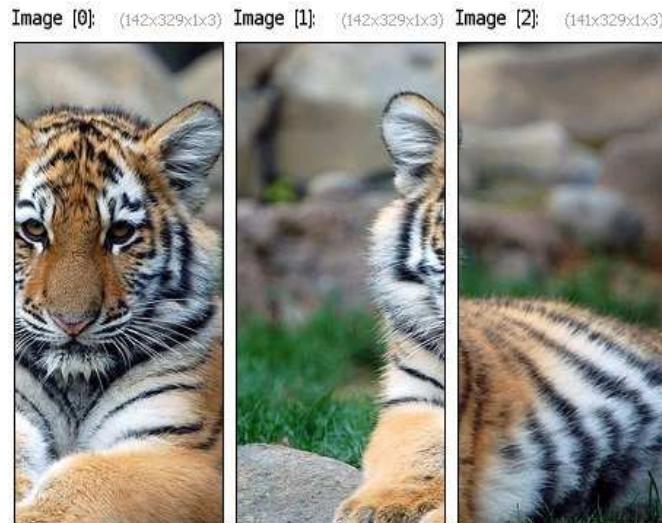


**Example 25 :** `image.jpg -split x -keep[^30%-70%] -append x`

### 2.3.2 -move (\*)

**Arguments:** position[%]

Move selected images at specified position.  
(*e.g.* to '-mv').



**Example 26 :** image.jpg -split x,3 -move[1] 0

(425x329x1x3)



**Example 27 :** image.jpg -split x -move[50%--1:2] 0 -append x

### 2.3.3 -name (\*)

**Arguments:** name, \_is\_modified={ 0 | 1 }

Set name of selected images.

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

Argument 'is\_modified' tells about the modified state of selected images.

**Default value:** 'is\_modified=0' .



**Example 28 :** image.jpg -name image -blur[image] 2

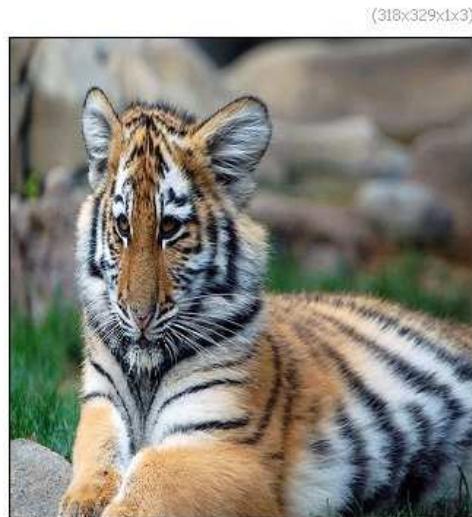
#### 2.3.4 **-remove** (\*)

Remove selected images.

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



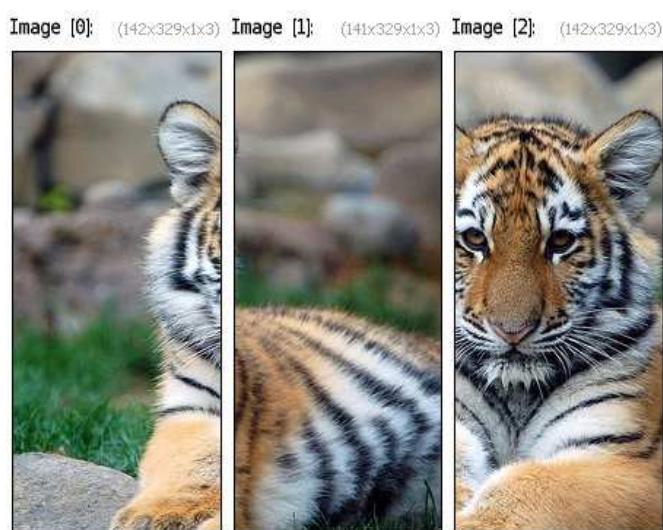
**Example 29 :** image.jpg -split x -remove[30%-70%] -append x



**Example 30 :** `image.jpg -split x -remove[0-50%:2] -append x`

### 2.3.5 *-reverse* (\*)

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



**Example 31 :** `image.jpg -split x,3 -reverse[-2,-1]`



**Example 32:** `image.jpg -split x,-16 -reverse[50%-100%] -append x`

### 2.3.6 *-sort\_list*

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

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

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



**Example 33:** `(1;4;7;3;9;2;4;7;6;3;9;1;0;3;3;2) -split y -sort_list + -append y`

### 2.3.7 *-sort\_str*

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

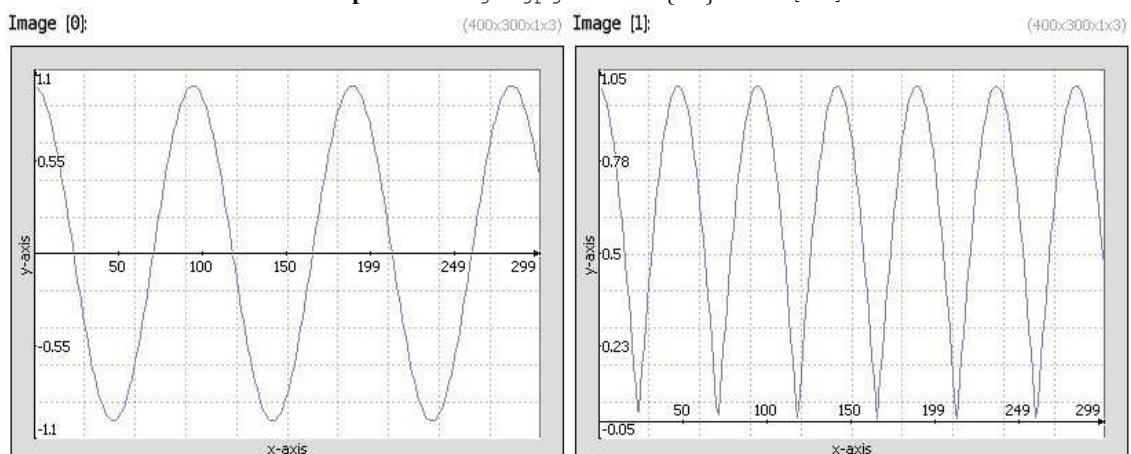
## 2.4 Mathematical operators

### 2.4.1 -abs (+)

Compute the pointwise absolute values of selected images.



**Example 34 :** `image.jpg --sub {ia} -abs [-1]`



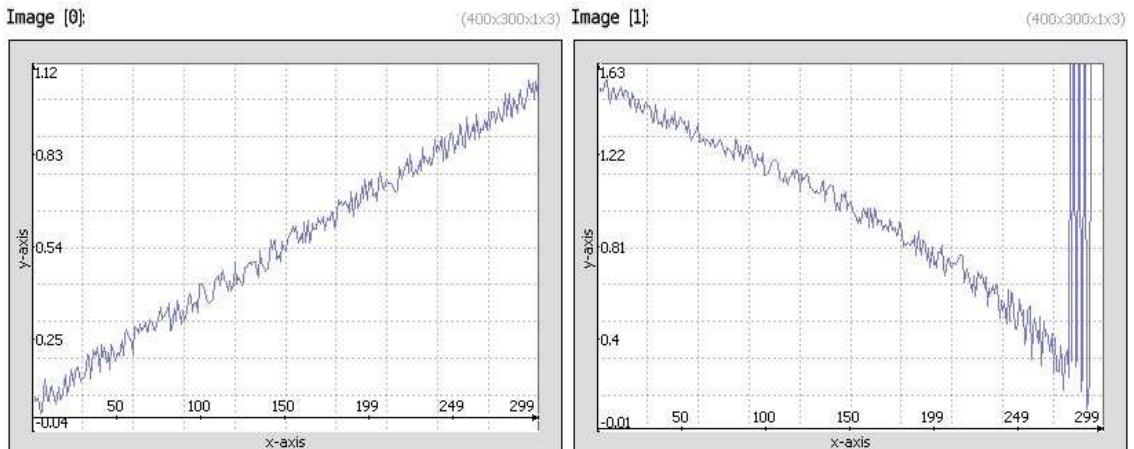
**Example 35 :** `300,1,1,1,'cos(20*x/w)' --abs -display_graph 400,300`

### 2.4.2 -acos (+)

Compute the pointwise arc-cosine of selected images.



**Example 36 :** `image.jpg --normalize -1,1 -acos[-1]`



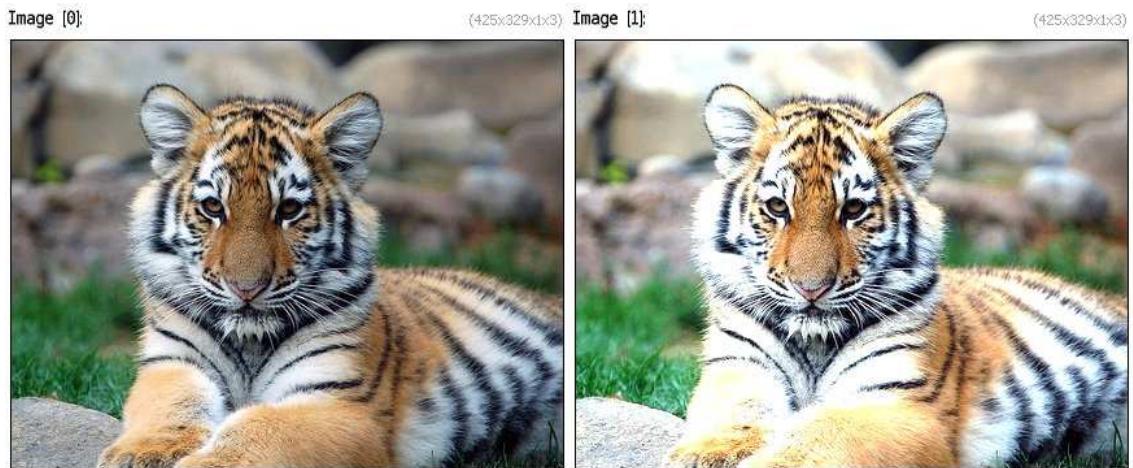
**Example 37 :** `300,1,1,1,'x/w+0.1*u' --acos -display_graph 400,300`

### 2.4.3 `-add (+)`

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

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

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



**Example 38 :** image.jpg --add 30% -cut 0,255



**Example 39 :** image.jpg --blur 5 -normalize 0,255 -add[1] [0]



**Example 40 :** `image.jpg -add '80*cos(80*(x/w-0.5)*(y/w-0.5)+c)' -cut 0,255  
(425x329x1x3)`



**Example 41 :** `image.jpg -repeat 9 --rotate[0] {$>*36},1,0,50%,50% -done -add -div  
10`

#### 2.4.4 *-and (+)*

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

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



**Example 42 :** image.jpg -and {128+64}

(425x329x1x3)



**Example 43 :** image.jpg --mirror x -and

#### 2.4.5 -asin (+)

Compute the pointwise arc-sine of selected images.

Image [0]:

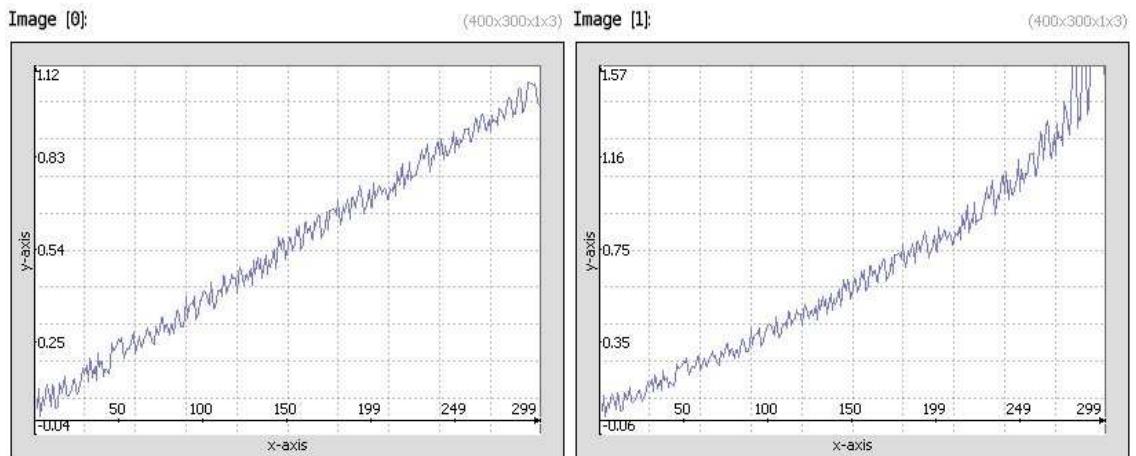
(425x329x1x3)

Image [1]:

(425x329x1x3)



**Example 44 :** image.jpg --normalize -1,1 -asin[-1]



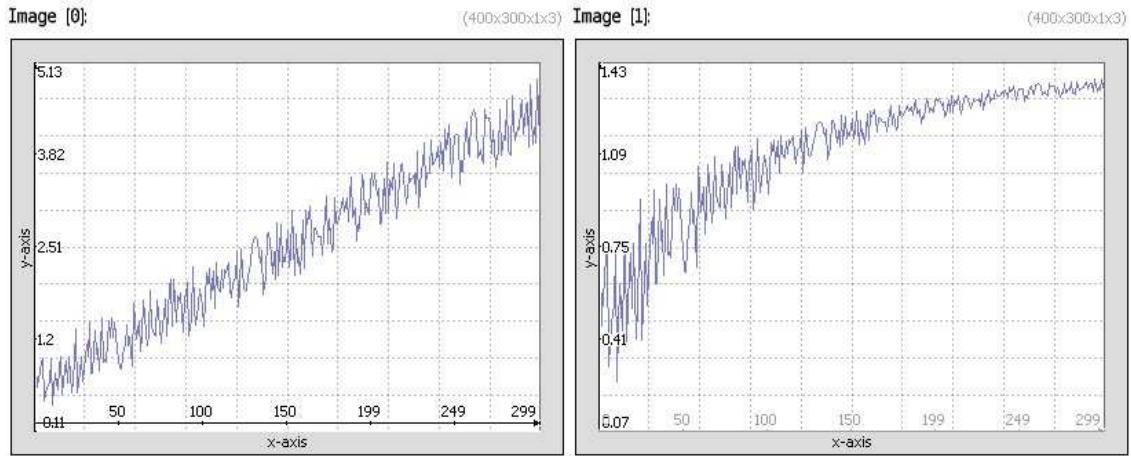
**Example 45 :** 300,1,1,1,'x/w+0.1\*u' --asin -display\_graph 400,300

#### 2.4.6 -atan (+)

Compute the pointwise arc-tangent of selected images.



**Example 46 :** image.jpg --normalize 0,8 -atan[-1]



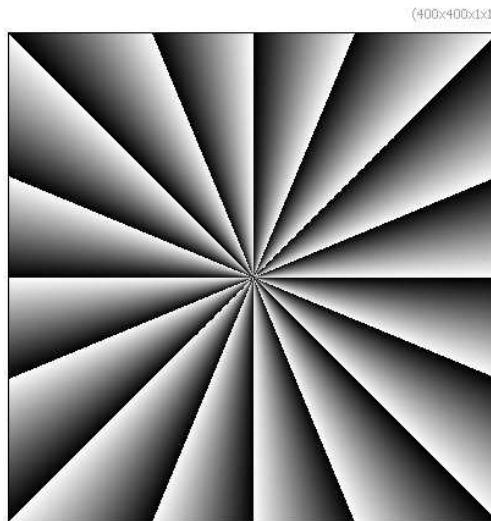
**Example 47 :** 300,1,1,1,'4\*x/w+u' --atan -display\_graph 400,300

#### 2.4.7 -atan2 (+)

**Arguments:** [x\_argument]

Compute the pointwise oriented arc-tangent of selected images.

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



**Example 48 :** (-1,1) (-1;1) -resize 400,400,1,1,3 -atan2[1] [0] -keep[1] -mod {pi/8}

#### 2.4.8 -bsl (+)

**Arguments:** value[%] |

```
[image] |
'formula' |
(no args)
```

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



**Example 49 :** `image.jpg -bsl 'round(3*x/w, 0)' -cut 0,255`

#### 2.4.9 **-bsr (+)**

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

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 '->>'*).



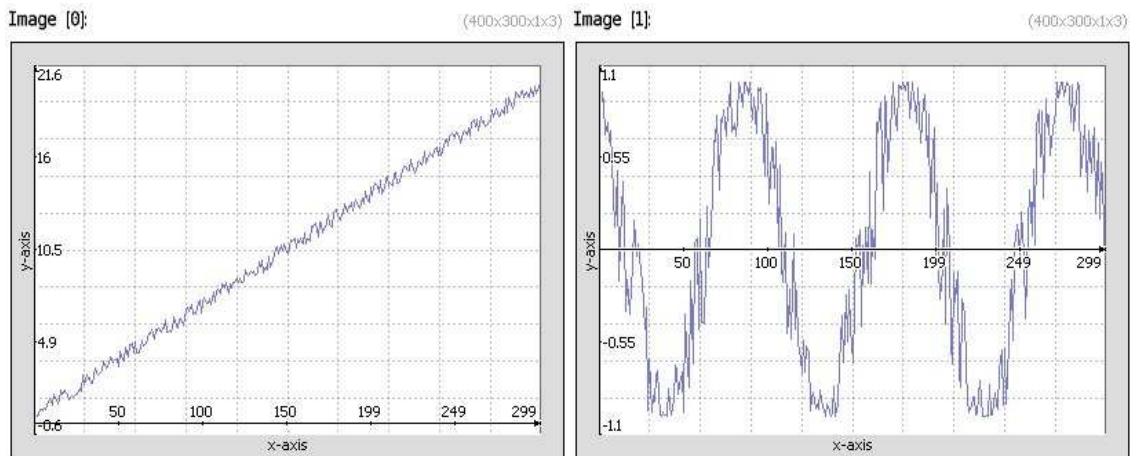
(425x329x1x3)

**Example 50 :** `image.jpg -bsr 'round(3*x/w, 0)' -cut 0,255`

#### 2.4.10 $-\cos(+)$

Compute the pointwise cosine of selected images.

**Example 51 :** `image.jpg --normalize 0,{2*pi} -cos[-1]`



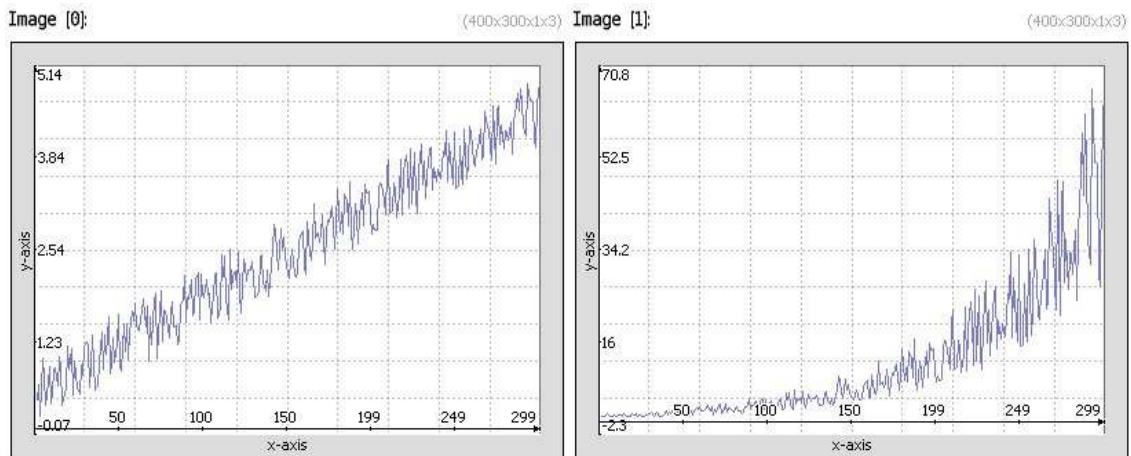
```
Example 52: 300,1,1,1,'20*x/w+u' --cos -display_graph 400,300
```

### 2.4.11 **-cosh (+)**

Compute the pointwise hyperbolic cosine of selected images.



```
Example 53: image.jpg --normalize -3,3 -cosh[-1]
```



**Example 54:** 300,1,1,1,'4\*x/w+u' --cosh -display\_graph 400,300

### 2.4.12 -div (+)

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

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



**Example 55:** image.jpg -div '1+abs(cos(x/10)\*sin(y/10))'



**Example 56 :** `image.jpg --luminance --div`

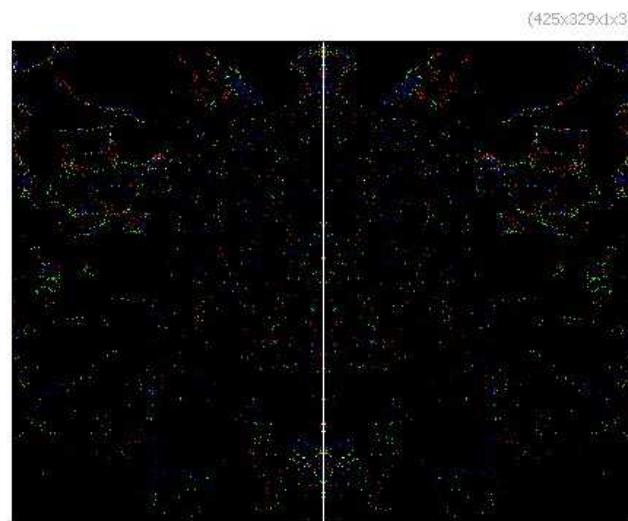
#### 2.4.13 -eq (+)

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

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



**Example 57 :** `image.jpg -round 40 -eq {round(ia,40)}`



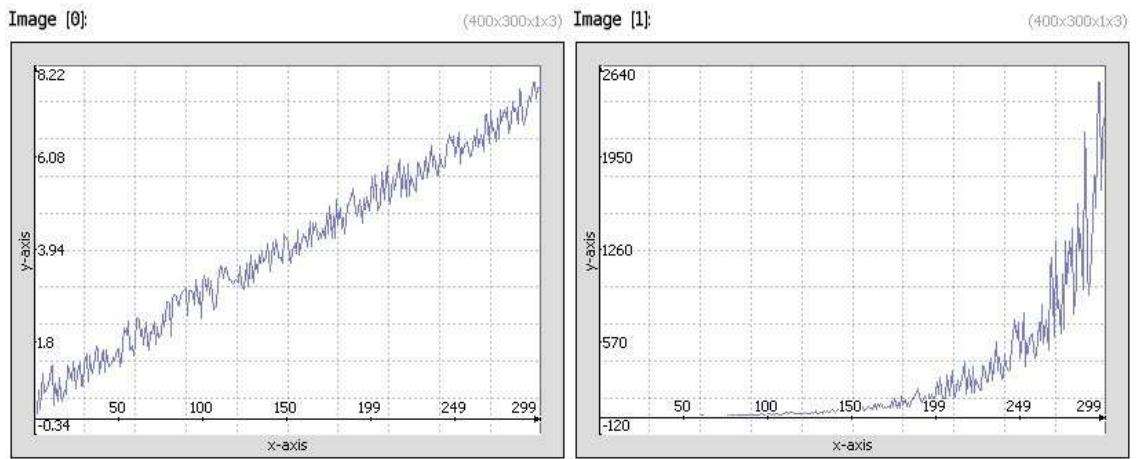
**Example 58 :** image.jpg --mirror x -eq

#### 2.4.14 -exp (+)

Compute the pointwise exponential of selected images.



**Example 59 :** image.jpg --normalize 0,2 -exp [-1]



**Example 60 :** `300,1,1,1,'7*x/w+u' --exp -display_graph 400,300`

#### 2.4.15 -ge (+)

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

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 61 :** `image.jpg -ge {ia}`



**Example 62 :** image.jpg --mirror x -ge

#### 2.4.16 -gt (+)

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

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



**Example 63 :** image.jpg -gt {ia}



**Example 64 :** image.jpg --mirror x -gt

#### 2.4.17 -le (+)

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

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 65 :** image.jpg -le {ia}



**Example 66 :** image.jpg --mirror x -le

#### 2.4.18 -lt (+)

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

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



**Example 67 :** image.jpg -lt {ia}



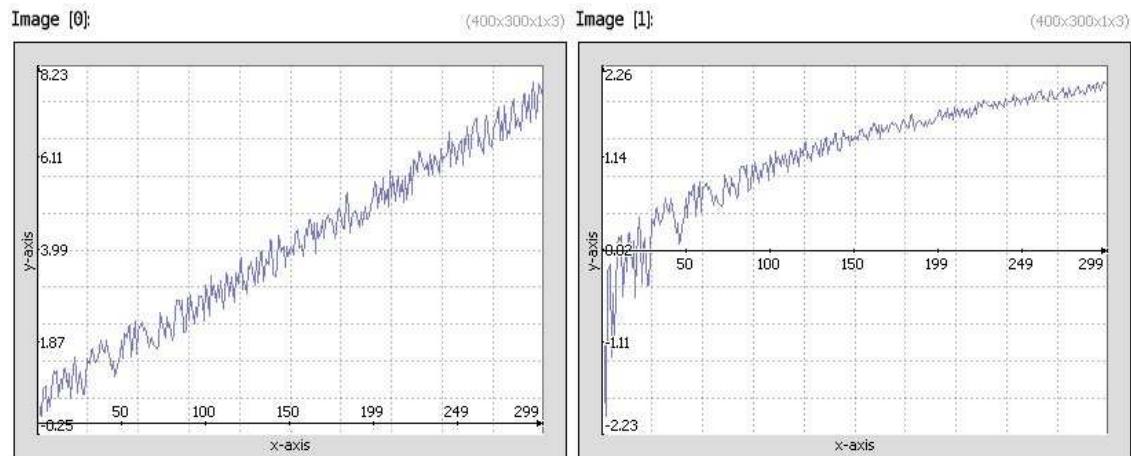
**Example 68 :** `image.jpg --mirror x -lt`

#### 2.4.19 $-\log (+)$

Compute the pointwise base-e logarithm of selected images.



**Example 69 :** `image.jpg --add 1 -log[-1]`



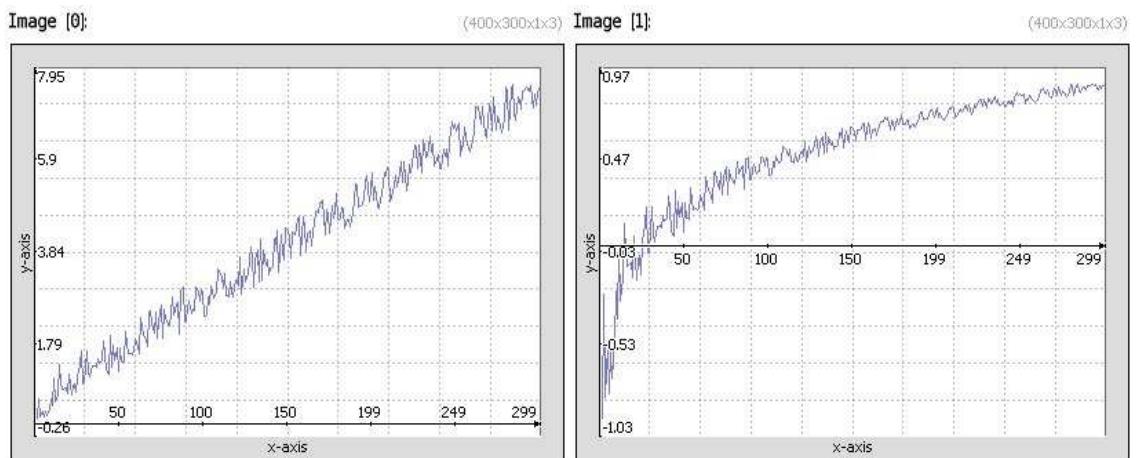
**Example 70 :** 300,1,1,1,'7\*x/w+u' --log -display\_graph 400,300

#### 2.4.20 $\text{-log10}(+)$

Compute the pointwise base-10 logarithm of selected images.



**Example 71 :** image.jpg --add 1 -log10[-1]



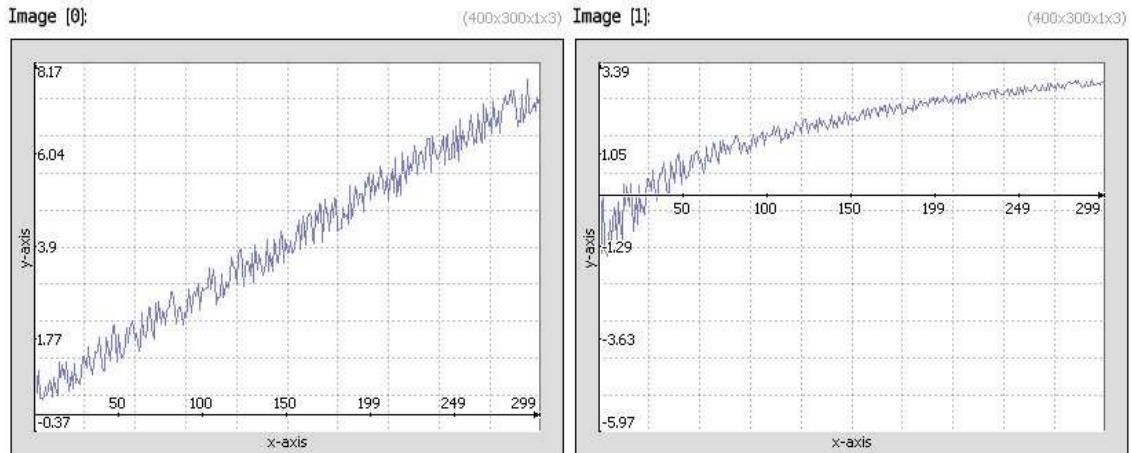
**Example 72 :** `300,1,1,1,'7*x/w+u' --log10 -display_graph 400,300`

#### 2.4.21 $-\log_2 (+)$

Compute the pointwise base-2 logarithm of selected images



**Example 73 :** `image.jpg --add 1 -log2[-1]`



**Example 74:** 300,1,1,1,'7\*x/w+u' --log2 -display\_graph 400,300

#### 2.4.22 -max (+)

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

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



**Example 75:** image.jpg --mirror x -max



**Example 76 :** `image.jpg -max 'R=( (x/w-0.5)^2+(y/h-0.5)^2)^0.5;255*R'`

#### 2.4.23 **-mdiv (+)**

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

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

#### 2.4.24 **-min (+)**

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

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



**Example 77 :** image.jpg --mirror x -min



**Example 78 :** image.jpg -min 'R=((x/w-0.5)^2+(y/h-0.5)^2)^0.5;255\*R'

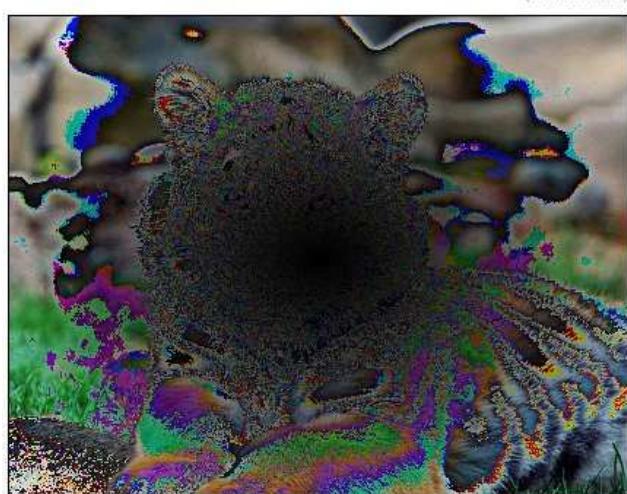
#### 2.4.25 -mod (+)

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

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



**Example 79 :** `image.jpg --mirror x -mod`



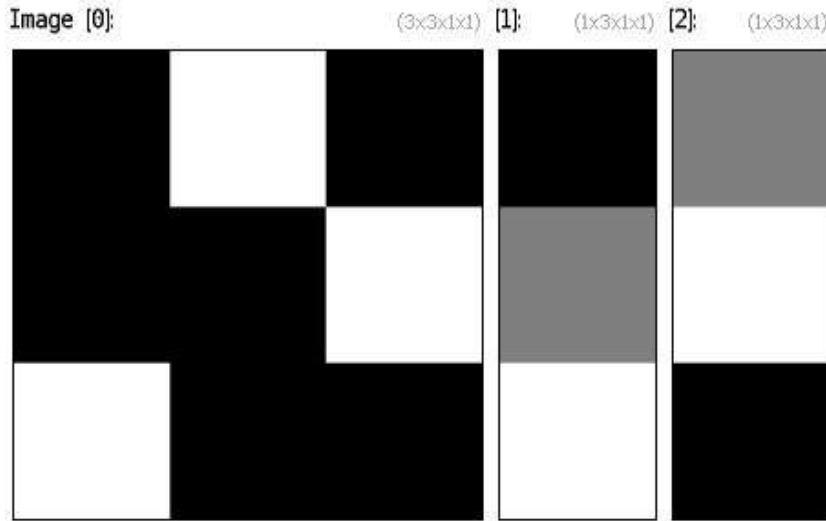
**Example 80 :** `image.jpg -mod 'R=((x/w-0.5)^2+(y/h-0.5)^2)^0.5;255*R'`

#### 2.4.26 `-mmul (+)`

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

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

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



**Example 81:** (0,1,0;0,0,1;1,0,0) (1;2;3) --mmul

#### 2.4.27 -mul (+)

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

Multiply selected images by specified value, image or mathematical expression, or compute the pointwise product of selected images.  
(eq. to ' $\text{--} \times$ ').



**Example 82:** image.jpg --mul 2 -cut 0,255



**Example 83 :** `image.jpg (1,2,3,4,5,6,7,8) -resize[-1] [0] -mul[0] [-1] (425x329x1x3)`



**Example 84 :** `image.jpg -mul '1-3*abs(x/w-0.5)' -cut 0,255 (425x329x1x3)`



**Example 85 :** `image.jpg --luminance -negative[-1] --mul`

**2.4.28 -neq (+)**

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

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 86 :** image.jpg -round 40 -neq {round(ia,40)}

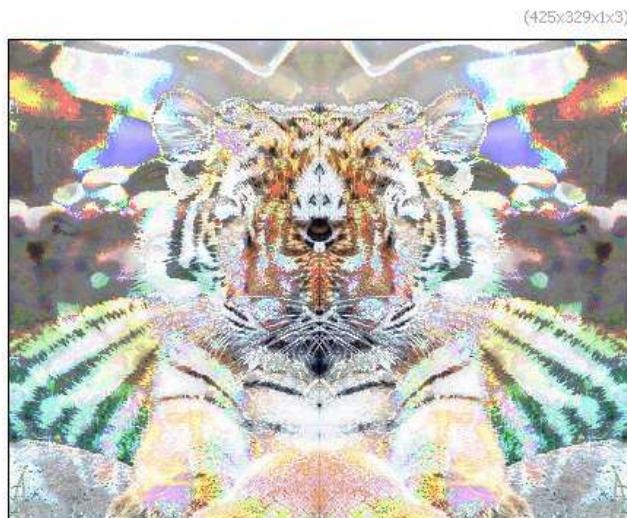
**2.4.29 -or (+)**

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

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



**Example 87 :** image.jpg -or 128



**Example 88 :** image.jpg --mirror x -or

#### 2.4.30 -pow (+)

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

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



**Example 89 :** image.jpg -div 255 --pow 0.5 -mul 255

(425x329x1x3)



**Example 90 :** image.jpg -gradient -pow 2 -add -pow 0.2

### 2.4.31 *-rol (+)*

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

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

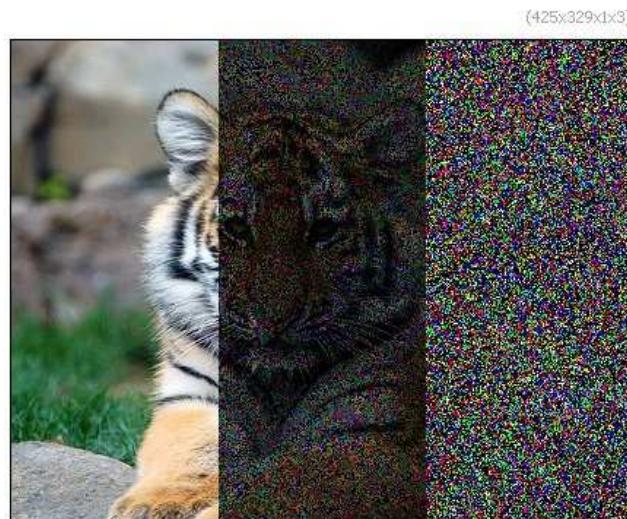


**Example 91 :** `image.jpg -rol 'round(3*x/w, 0)' -cut 0, 255`

#### 2.4.32 `-ror (+)`

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

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



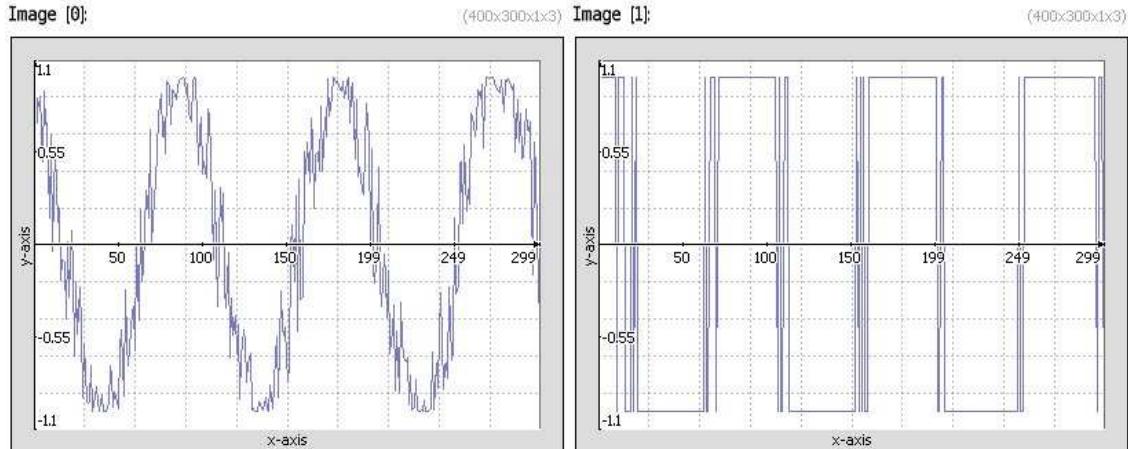
**Example 92 :** `image.jpg -ror 'round(3*x/w, 0)' -cut 0, 255`

**2.4.33 -sign (+)**

Compute the pointwise sign of selected images.



**Example 93 :** image.jpg --sub {ia} -sign[-1]



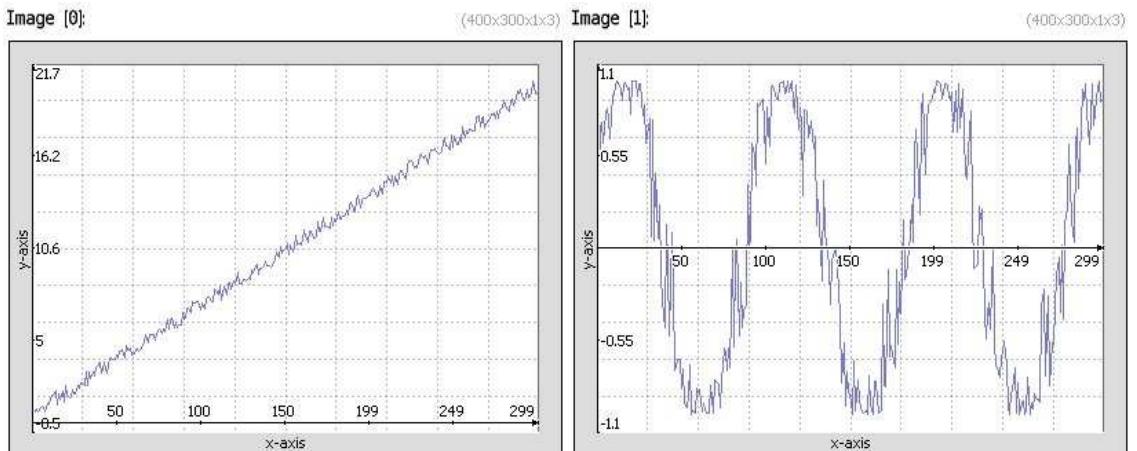
**Example 94 :** 300,1,1,1,'cos(20\*x/w+u)' --sign -display\_graph 400,300

**2.4.34 -sin (+)**

Compute the pointwise sine of selected images.



**Example 95 :** `image.jpg --normalize 0,{2*pi} -sin[-1]`



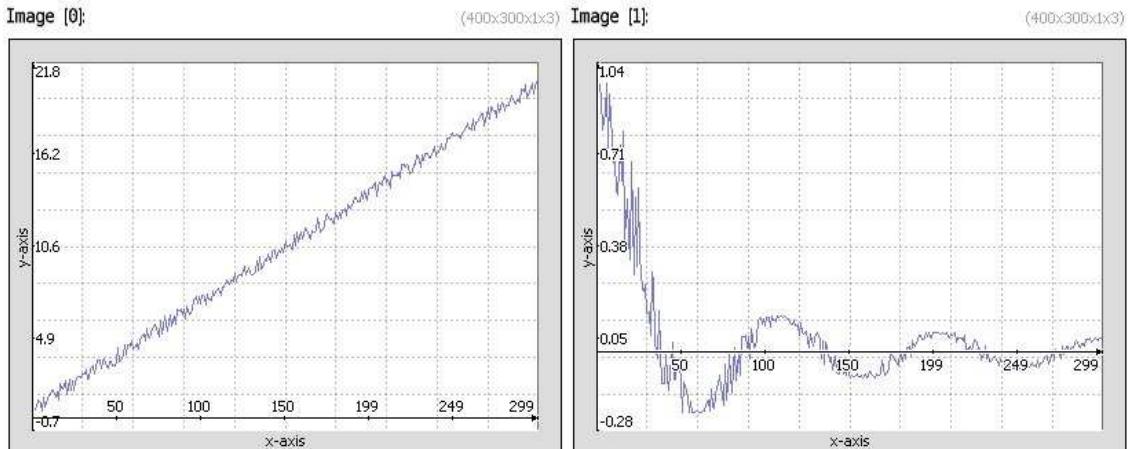
**Example 96 :** `300,1,1,1,'20*x/w+u' --sin -display_graph 400,300`

#### 2.4.35 `-sinc (+)`

Compute the pointwise sinc function of selected images.



**Example 97 :** image.jpg --normalize {-2\*pi},{2\*pi} -sinc[-1]



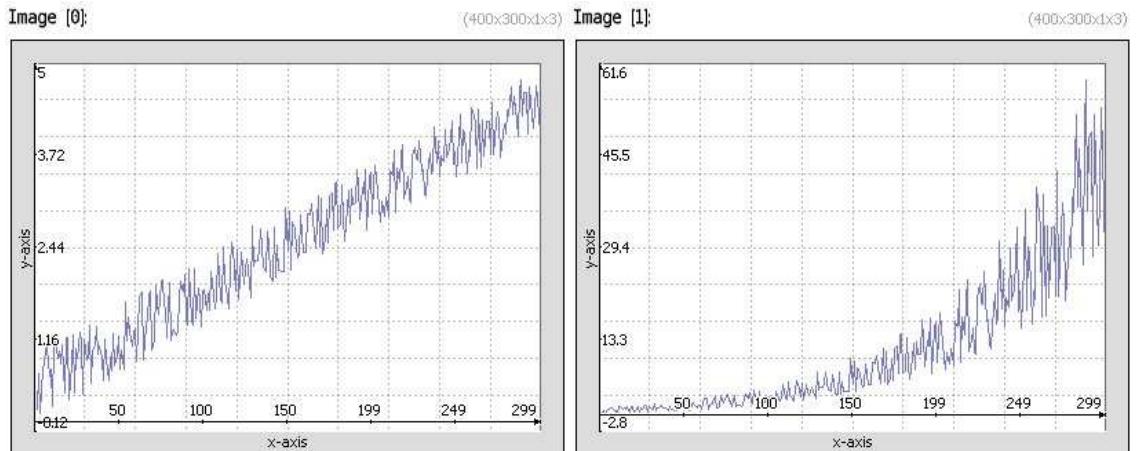
**Example 98 :** 300,1,1,1,'20\*x/w+u' --sinc -display\_graph 400,300

#### 2.4.36 -sinh (+)

Compute the pointwise hyperbolic sine of selected images.



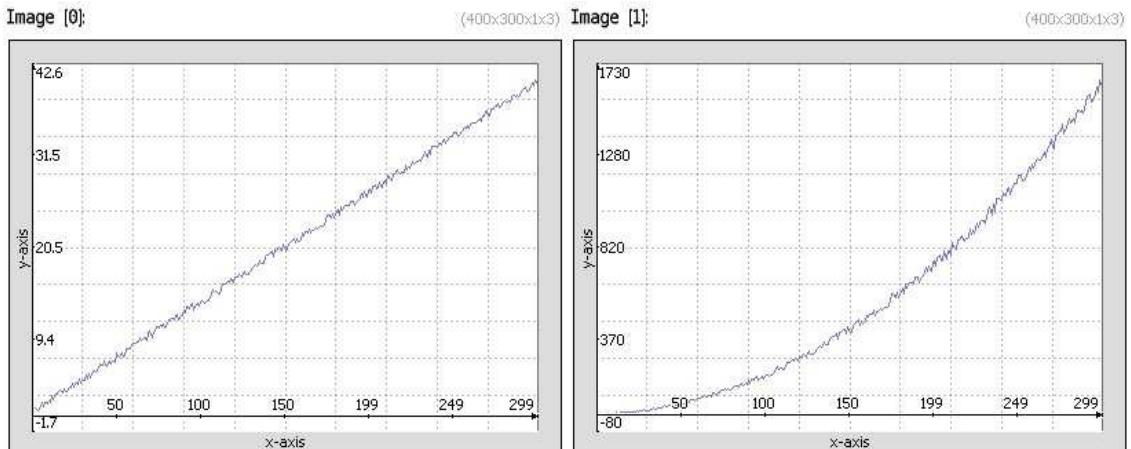
**Example 99 :** `image.jpg --normalize -3,3 -sinh[-1]`



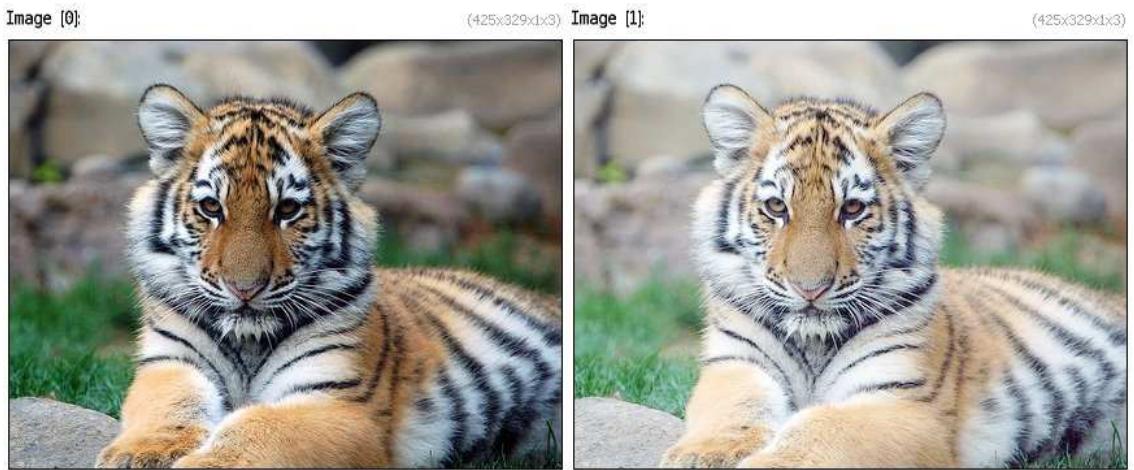
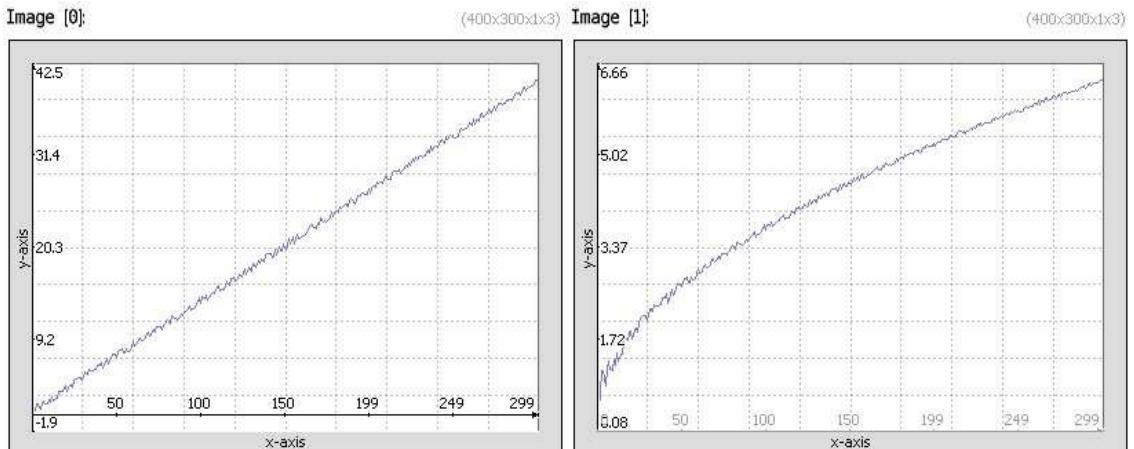
**Example 100 :** `300,1,1,1,'4*x/w+u' --sinh -display_graph 400,300`

#### 2.4.37 `-sqr (+)`

Compute the pointwise square function of selected images.

**Example 101 :** image.jpg --sqr**Example 102 :** 300,1,1,1,'40\*x/w+u' --sqr -display\_graph 400,300**2.4.38 -sqrt (+)**

Compute the pointwise square root of selected images.

**Example 103 :** image.jpg --sqrt**Example 104 :** 300,1,1,1,'40\*x/w+u' --sqrt -display\_graph 400,300

### 2.4.39 -sub (+)

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

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



**Example 105 :** image.jpg --sub 30% -cut 0,255



**Example 106 :** image.jpg --mirror x -sub[-1] [0]



**Example 107 :** `image.jpg -sub 'i(w/2+0.9*(x-w/2),y)'`  
(425x329x1x3)



**Example 108 :** `image.jpg --mirror x -sub`

#### 2.4.40 *-tan (+)*

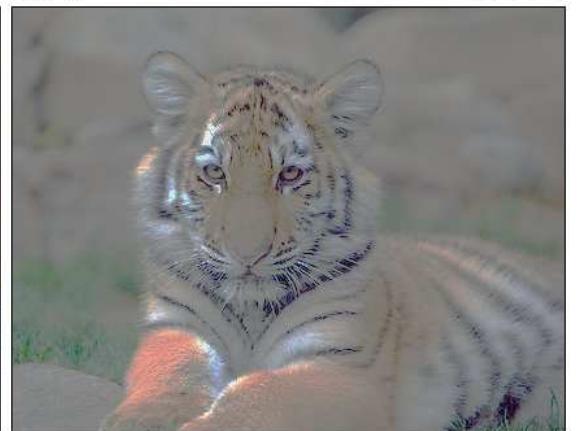
Compute the pointwise tangent of selected images.

Image [0]:

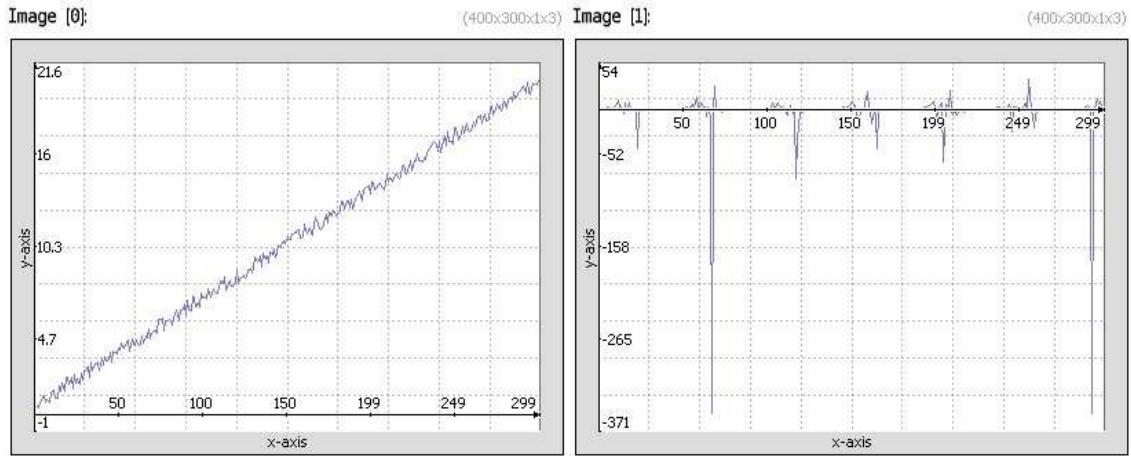
(425x329x1x3)

Image [1]:

(425x329x1x3)



**Example 109 :** `image.jpg --normalize {-0.47*pi},{0.47*pi} -tan[-1]`



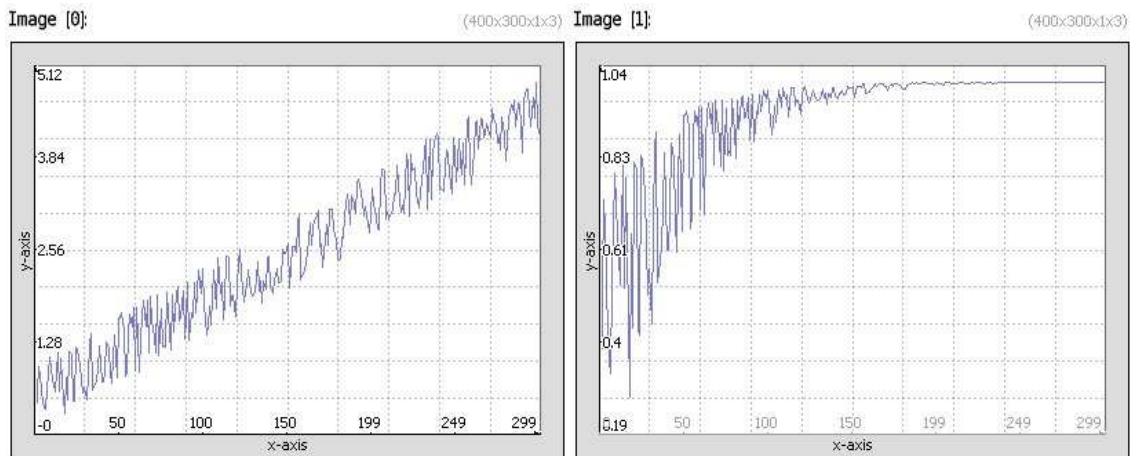
**Example 110 :** `300,1,1,1,'20*x/w+u' --tan -display_graph 400,300`

#### 2.4.41 $-\tanh(+)$

Compute the pointwise hyperbolic tangent of selected images.



**Example 111 :** `image.jpg --normalize -3,3 -tanh[-1]`

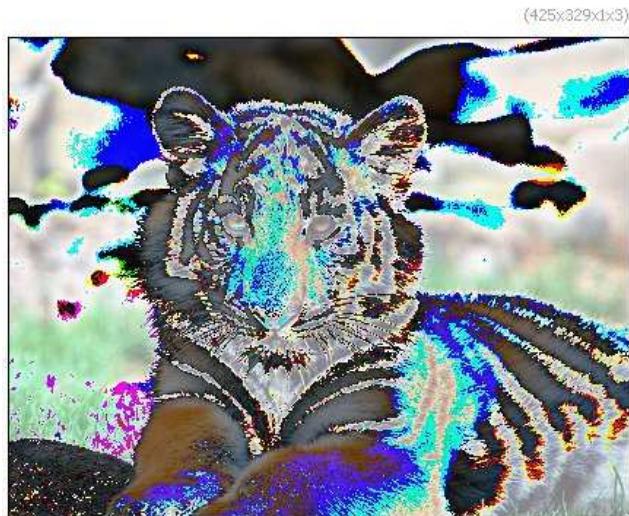


**Example 112 :** `300,1,1,1,'4*x/w+u' --tanh -display_graph 400,300`

#### 2.4.42 `-xor (+)`

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

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 113 :** `image.jpg -xor 128`



**Example 114 :** image.jpg --mirror x -xor

## 2.5 Values manipulation

### 2.5.1 -apply\_curve

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

Apply curve transformation to image values.

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



**Example 115 :** image.jpg --apply\_curve 1,0,0,128,255,255,0

### 2.5.2 *-apply\_gamma*

**Arguments:** `gamma>=0`

Apply gamma correction to selected images.



**Example 116 :** `image.jpg --apply_gamma 2`

### 2.5.3 *-balance\_gamma*

**Arguments:** `_ref_color1, ...`

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

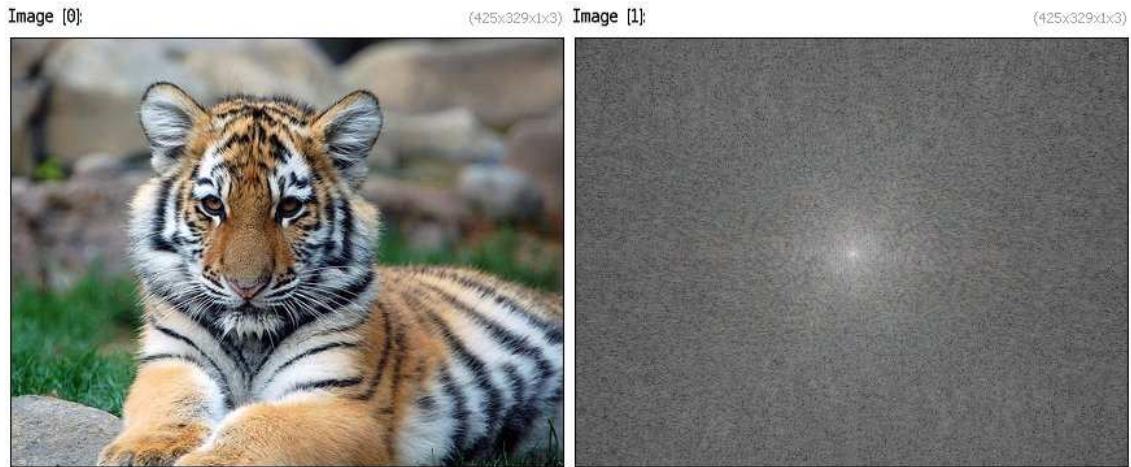
**Default values:** '`ref_color1=128`'.



**Example 117 :** `image.jpg --balance_gamma 128, 64, 64`

### 2.5.4 -complex2polar

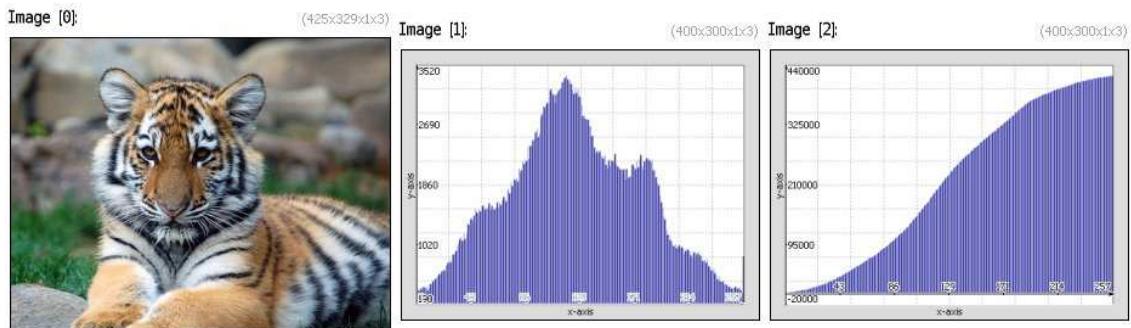
Compute complex to polar transforms of selected images.



```
Example 118: image.jpg --fft -complex2polar[-2,-1] -log[-2] -shift[-2]
50%,50%,0,0,2 -remove[-1]
```

### 2.5.5 -cumul

Compute the cumulative function of specified image data.



```
Example 119: image.jpg --histogram 256 --cumul[-1] -display_graph[-2,-1]
400,300,3
```

### 2.5.6 -cut (+)

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

Cut values of selected images in specified range.

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

(noargs) runs interactive mode (uses the instant window [0] if opened).



**Example 120 :** `image.jpg --add 30% -cut [-1] 0,255`



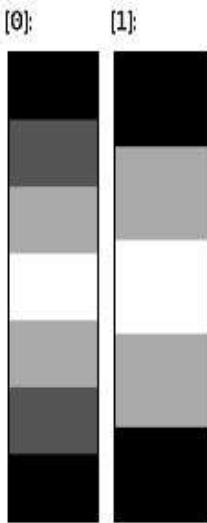
**Example 121 :** `image.jpg --cut 25%,75%`

### 2.5.7 *-discard*

**Arguments:** `value, _remove_if_not_found = { 0 | 1 }.`

Remove specified value in selected images and return results as single-column vector.

**Default value:** '`remove_if_not_found=0`'.



**Example 122 :** `(1;2;3;4;3;2;1) --discard 2`

### 2.5.8 *-eigen2tensor*

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

### 2.5.9 *-endian* (\*)

Reverse data endianness of selected images.

### 2.5.10 *-equalize*

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

Equalize histograms of selected images.

If value range is specified, the equalization is done only for pixels in the specified value range.

**Default values:** '`value_min=0%`' and '`value_max=100%`'.

Image [0]:

(425x329x1x3)



Image [1]:

(425x329x1x3)

Example 123 : `image.jpg --equalize 256`

Image [0]:

(425x329x1x3)



Image [1]:

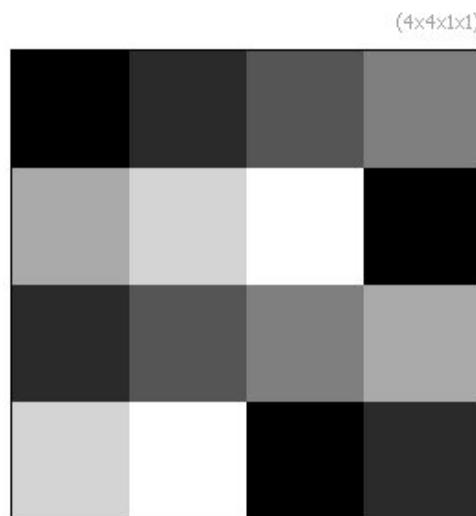
(425x329x1x3)

Example 124 : `image.jpg --equalize 4,0,128`

### 2.5.11 `-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 125 :** 4, 4 -fill 1, 2, 3, 4, 5, 6, 7

Image [0]:

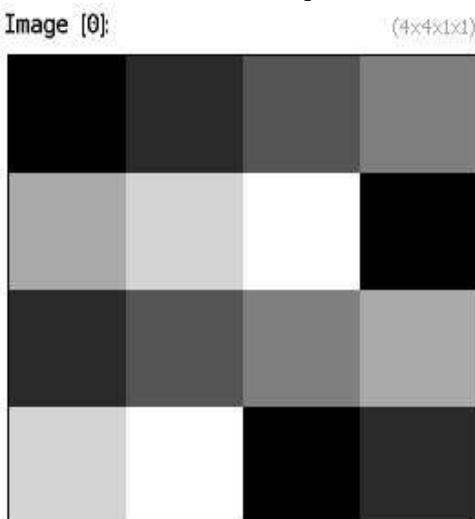
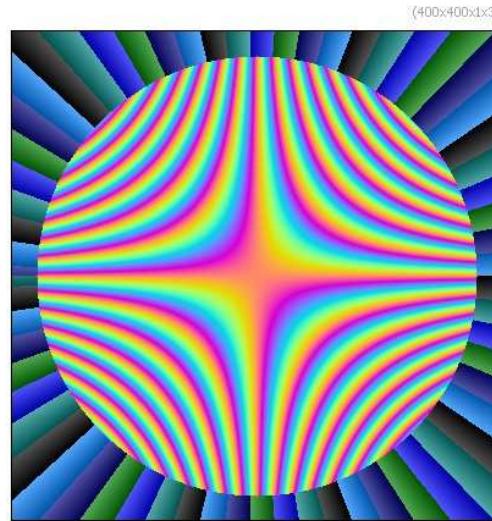


Image [1]:



**Example 126 :** 4, 4 (1, 2, 3, 4, 5, 6, 7) -fill [-2] [-1]



**Example 127:** `400,400,1,3 -fill "X=x-w/2; Y=y-h/2; R=sqrt(X^2+Y^2); a=atan2(Y,X); if (R<=180,255*abs(cos(c+200*(x/w-0.5)*(y/h-0.5))),850*(a%(0.1*(c+1))))"`

### 2.5.12 *-float2int8*

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

### 2.5.13 *-int82float*

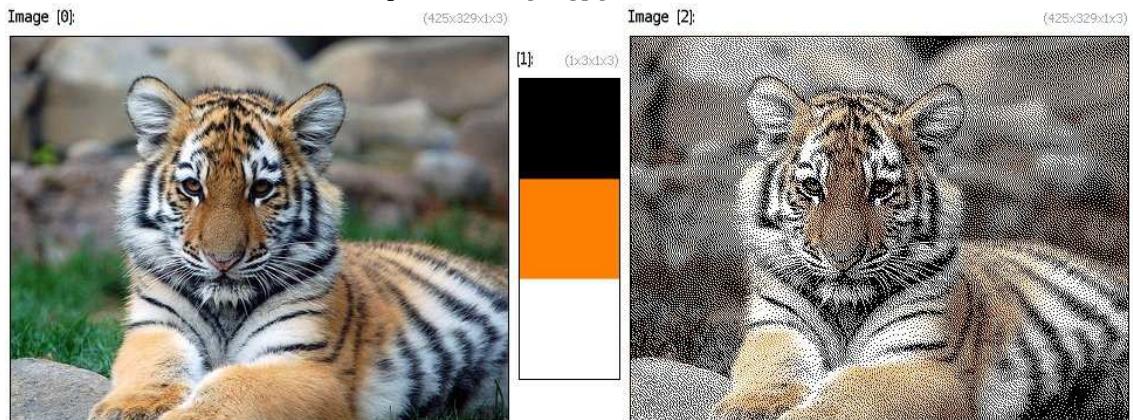
Convert selected 8bits integer representations to float-valued images.

### 2.5.14 *-index (+)*

**Arguments:** { [palette] | predefined\_palette }, 0<=\_dithering<=1, \_map\_palette={ 0 | 1 }

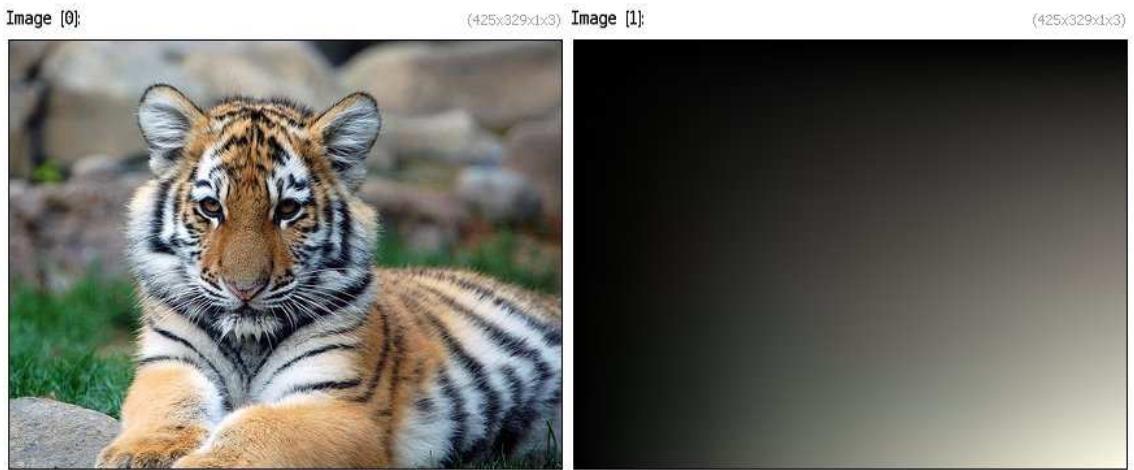
Index selected vector-valued images by specified vector-valued palette.  
'predefined\_palette' can be { 0=default | 1=HSV | 2=lines | 3=hot | 4=cool | 5=jet | 6=flag | 7=cube }.

**Default values:** 'dithering=0' and 'map\_palette=0'.

**Example 128 :** `image.jpg --index 1,1,1`**Example 129 :** `image.jpg (0;255;255^0;128;255^0;0;255) --index[-2] [-1],1,1`

### 2.5.15 -*image\_integral*

Compute the image integral (summed area table) of selected images.



**Example 130 :** `image.jpg --image_integral`

### 2.5.16 `-map (+)`

**Arguments:** [palette] |  
predefined\_palette

Map specified vector-valued palette to selected indexed scalar images.  
'predefined\_palette' can be { 0=default | 1=HSV | 2=lines | 3=hot | 4=cool | 5=jet | 6=flag  
| 7=cube }.



**Example 131 :** `image.jpg --luminance -map [-1] 3`



```
Example 132 : image.jpg --rgb2ycbcr -split[-1] c (0,255,0) -resize[-1] 256,1,1,1,3 -map[-4] [-1] -remove[-1] -append[-3--1] c -ycbcr2rgb[-1]
```

### 2.5.17 *-map\_clut*

Map RGB color LUT image (regarded as the last image) to all other selected images.



```
Example 133 : image.jpg -uniform_distribution {2^5},3 -mirror[-1] x --map_clut
```

### 2.5.18 *-mix\_channels*

**Arguments:** (a<sub>00</sub>, ..., a<sub>MN</sub>)

Apply specified matrix to channels of selected images.



**Example 134 :** `image.jpg --mix_channels (0,1,0;1,0,0;0,0,1)`

### 2.5.19 *-negative*

Compute negative of selected images.



**Example 135 :** `image.jpg --negative`

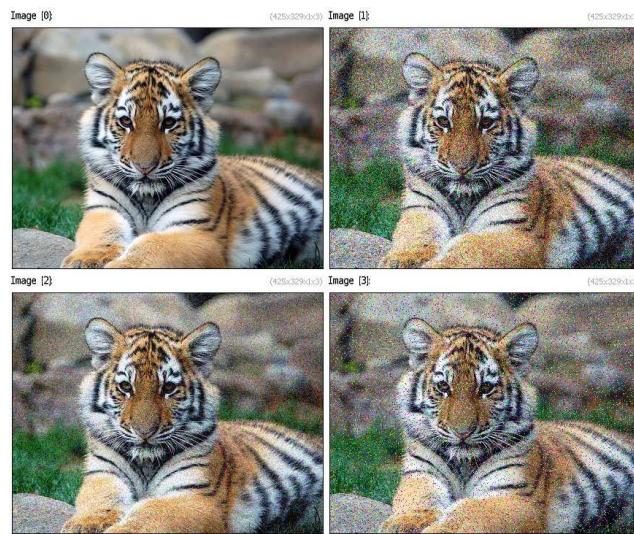
### 2.5.20 *-noise (+)*

**Arguments:** `std_variation>=0 [%], noise_type`

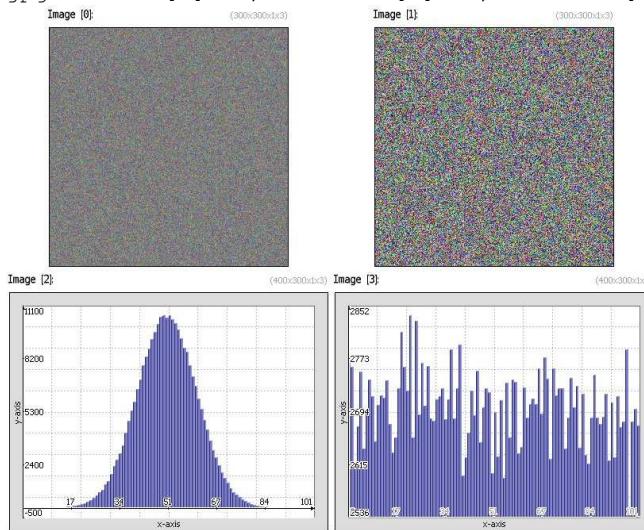
Add random noise to selected images.

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

**Default value:** 'noise\_type=0' .



**Example 136 :** image.jpg --noise[0] 50,0 --noise[0] 50,1 --noise[0] 10,2 -cut 0,255



**Example 137 :** 300,300,1,3 [0] -noise[0] 20,0 -noise[1] 20,1 --histogram 100  
-display\_graph[-2,-1] 400,300,3

### 2.5.21 *-norm*

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



**Example 138 :** `image.jpg --norm`

### 2.5.22 `-normalize (+)`

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

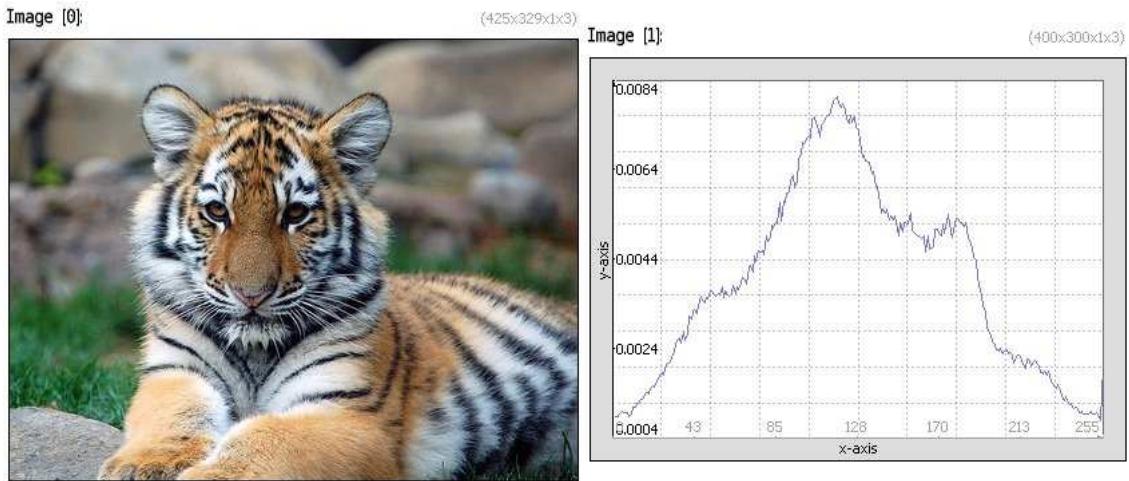
Linearly normalize values of selected images in specified range.  
(*eq. to '-n'*).



**Example 139 :** `image.jpg -split x,2 -normalize[-1] 64,196 -append x`

### 2.5.23 `-normalize sum`

Normalize selected images with a unitary sum.



```
Example 140: image.jpg --histogram[-1] 256 -normalize_sum[-1] -display_graph[-1]
400,300
```

### 2.5.24 *-orientation*

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



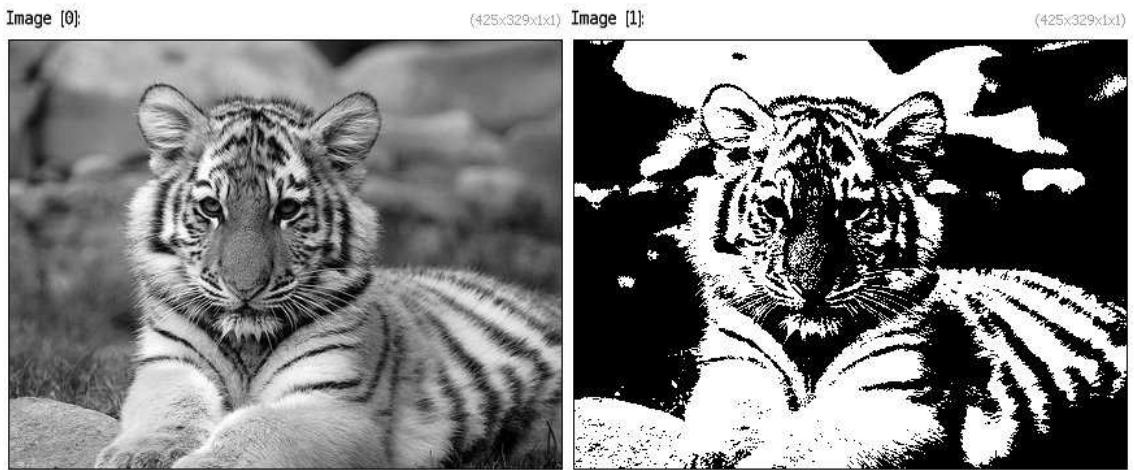
```
Example 141: image.jpg --orientation --norm[-2] -negative[-1] -mul[-2] [-1]
-reverse[-2,-1]
```

### 2.5.25 *-otsu*

**Arguments:** `_nb_levels>0`

Hard-threshold image using Otsu's method.

**Default value:** '`nb_levels=256`'.



**Example 142 :** `image.jpg -luminance --otsu ,`

### 2.5.26 *-polar2complex*

Compute polar to complex transforms of selected images.

### 2.5.27 *-quantize*

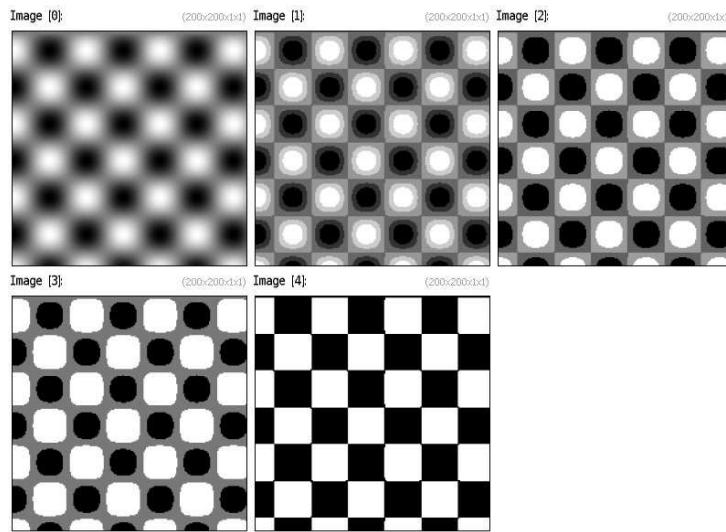
**Arguments:** `nb_levels>=1, keep_values={ 0 | 1 }, is_uniform={ 0 | 1 }`

Quantize selected images.

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



**Example 143 :** `image.jpg -luminance --quantize 3`

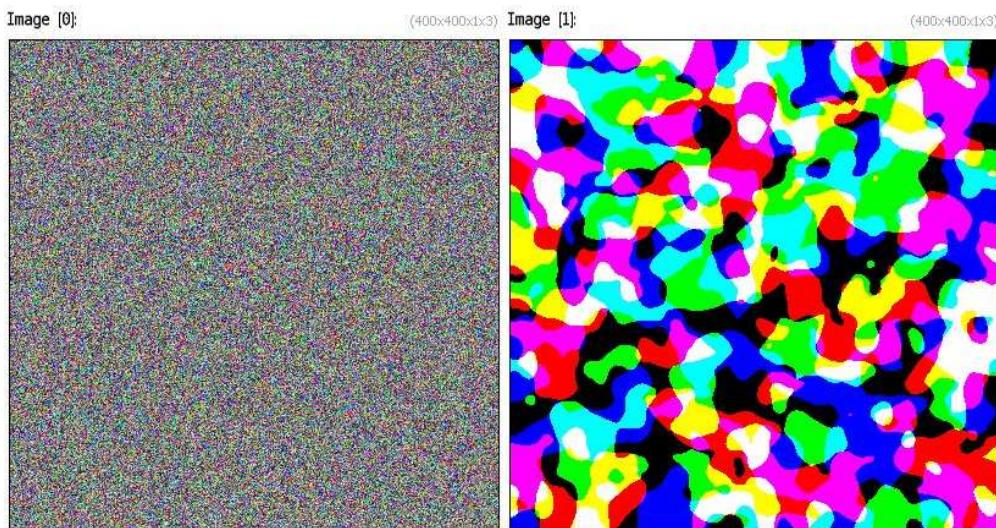


```
Example 144 : 200,200,1,1,'cos(x/10)*sin(y/10)' --quantize[0] 6 --quantize[0] 4  
--quantize[0] 3 --quantize[0] 2
```

### 2.5.28 -rand (+)

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

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

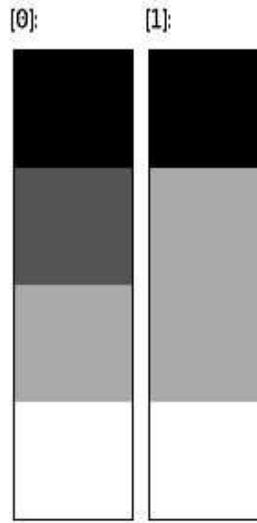


```
Example 145 : 400,400,1,3 -rand -10,10 --blur 10 -sign[-1]
```

### 2.5.29 *-replace*

**Arguments:** `value_src, value_dest`

Replace pixel values in selected images.



**Example 146 :** `(1;2;3;4) --replace 2,3`

### 2.5.30 *-replace\_inf*

**Arguments:** `_expression`

Replace all infinite values in selected images by specified expression.



**Example 147 :** `(0;1;2) -log --replace_inf 2`

### 2.5.31 -replace\_nan

**Arguments:** \_expression

Replace all NaN values in selected images by specified expression.

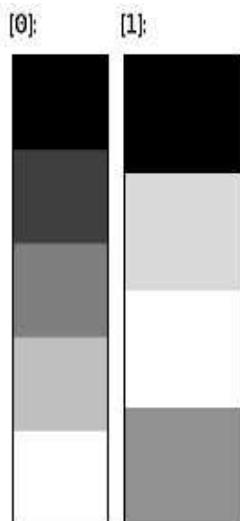


**Example 148 :** (-1; 0; 2) -sqrt --replace\_nan 2

### 2.5.32 -replace\_seq

**Arguments:** "search\_seq", "replace\_seq"

Search and replace a sequence of values in selected images.



**Example 149 :** (1; 2; 3; 4; 5) --replace\_seq "2, 3, 4", "7, 8"

### 2.5.33 *-round (+)*

**Arguments:** rounding\_value $\geq 0$ , rounding\_type |  
(no args)

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

**Default value:** 'rounding\_type=0' .



**Example 150 :** image.jpg --round 100



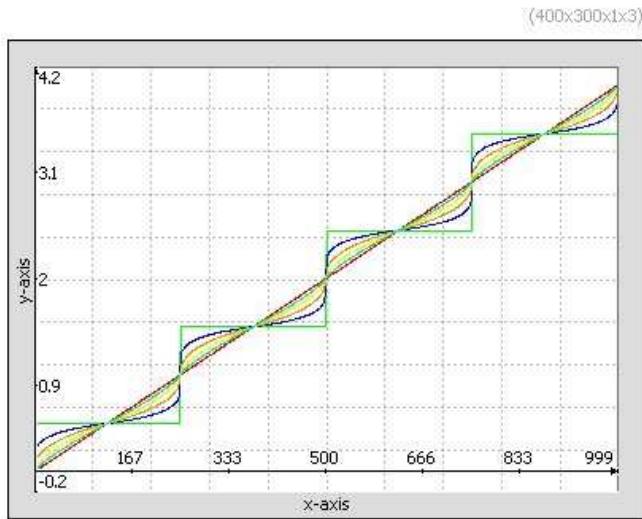
**Example 151 :** image.jpg -mul {pi/180} -sin --round

### 2.5.34 *-roundify*

**Arguments:** gamma $\geq 0$

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

**Default value:** 'gamma=0' .



**Example 152 :** 1000 -fill '4\*x/w' -repeat 5 --roundify[0] {\$>\*0.2} -done -append c  
-display-graph 400,300

### 2.5.35 -set (\*)

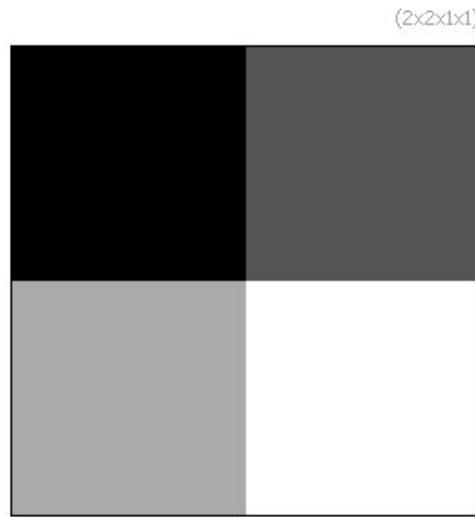
**Arguments:** value, -x [%], -y [%], -z [%], -c [%]

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 153 :** `2,2 -set 1,0,0 -set 2,1,0 -set 3,0,1 -set 4,1,1`

`(425x329x1x3)`



**Example 154 :** `image.jpg -repeat 10000 -set 255,{?(100)}%,{?(100)}%,0,{?(100)}%`  
`-done`

### 2.5.36 *-threshold (+)*

**Arguments:** `value[%], -is_soft |`  
`(no args)`

Threshold values of selected images.

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

(noargs) runs interactive mode (uses the instant window [0] if opened).

**Default value:** '`is_soft=0`' .



**Example 155 :** `image.jpg --threshold[0] 50% --threshold[0] 50%,1`

### 2.5.37 -threshold2

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

Threshold selected images between the two given values.  
(*eq. to '-t2'*).



**Example 156 :** `image.jpg --threshold2 25%,75%`

### 2.5.38 -vector2tensor

Convert selected vector fields to corresponding diffusion tensor fields.

## 2.6 Colors manipulation

### 2.6.1 -apply\_channels

**Arguments:** `"command", _channels={ all=0 | rgba=1 | rgb=2 | y=3 | cbcr=4 | cb=5 | cr=6 | l=7 | ab=8 | a=9 | b=10 | h=11 }`

```
| s=12 | v=13 | k=14 | cg=15 | ch=16 | c=17 | H=18 | r=19
| g=20 | b=21 | alpha==22 },_normalize={ 0=cut | 1=normalize }
```

Apply specified command on chosen normalized channels of each selected images.

**Default value:** 'normalize=0' .



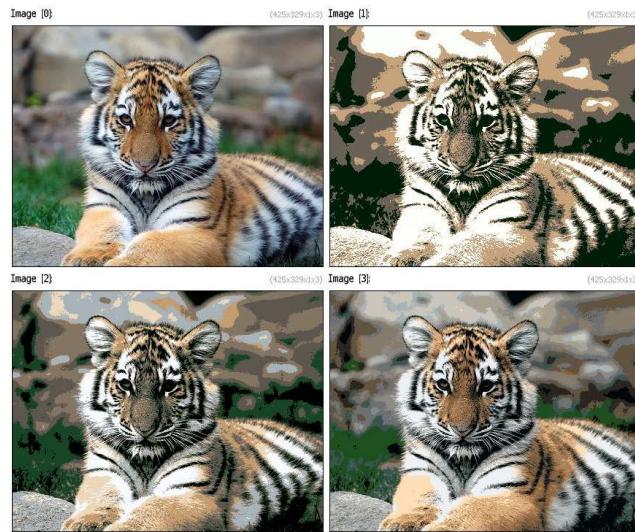
**Example 157 :** image.jpg --apply\_channels "-equalize 256 -blur 2",5

## 2.6.2 *-autoindex*

**Arguments:** nb\_colors>0,0<=\_dithering<=1,\_method={ 0=median-cut  
| 1=k-means }

Index selected vector-valued images by adapted colormaps.

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



**Example 158 :** `image.jpg --autoindex[0] 4 --autoindex[0] 8 --autoindex[0] 16`

### 2.6.3 -bayer2rgb

**Arguments:** `_GM_smoothness, _RB_smoothness1, _RB_smoothness2`

Transform selected RGB-Bayer sampled images to color images.

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



**Example 159 :** `image.jpg -rgb2bayer 0 --bayer2rgb 1,1,0.5`

### 2.6.4 -cmy2rgb

Convert selected images from CMY to RGB colorbases.

### **2.6.5 -cmyk2rgb**

Convert selected images from CMYK to RGB colorbases.

### **2.6.6 -colormap**

**Arguments:** nb\_levels>=1, \_method={ 0=median-cut | 1=k-means }, \_sort\_vectors={ 0 | 1 }

Estimate best-fitting colormap with 'nb\_colors' entries, to index selected images.

**Default value:** 'method=0' and 'sort\_vectors=1' .



**Example 160 :** image.jpg --colormap[0] 4 --colormap[0] 8 --colormap[0] 16

### **2.6.7 -compose\_channels**

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

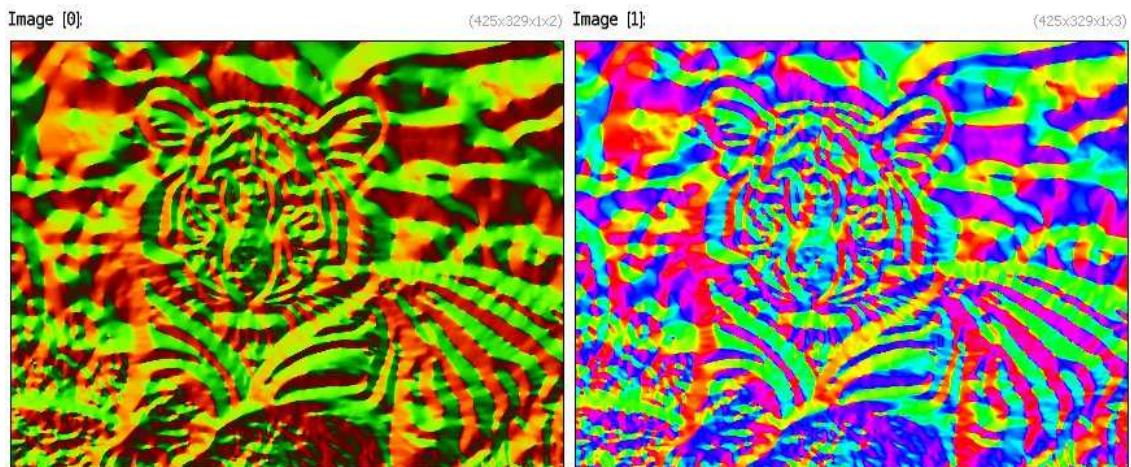
**Default value:** '1=+' .



**Example 161 :** `image.jpg --compose_channels and`

### 2.6.8 *-direction2rgb*

Compute RGB representation of selected 2d direction fields.



**Example 162 :** `image.jpg -luminance -gradient -append c -blur 2 -orientation  
--direction2rgb`

### 2.6.9 *-ditheredbw*

Create dithered B&W version of selected images.

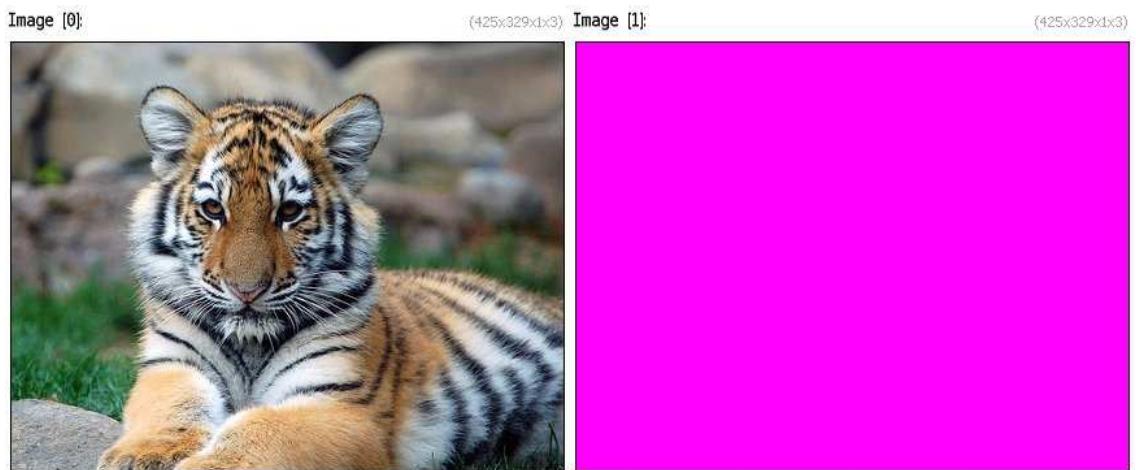


**Example 163 :** `image.jpg --equalize 256 -ditheredbw[-1]`

### 2.6.10 *-fill\_color*

**Arguments:** `col1, ..., colN`

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



**Example 164 :** `image.jpg --fill_color 255,0,255`

### 2.6.11 *-gradient2rgb*

**Arguments:** `_is_orientation={ 0 | 1 }`

Compute RGB representation of 2d gradient of selected images.

**Default value:** 'is\_orientation=0'.



**Example 165 :** image.jpg --gradient2rgb 0 -equalize[-1] 256

### 2.6.12 **-hsi2rgb (+)**

Convert selected images from HSI to RGB colorbases.

### 2.6.13 **-hsi82rgb**

Convert selected images from HSI8 to RGB color bases.

### 2.6.14 **-hsl2rgb (+)**

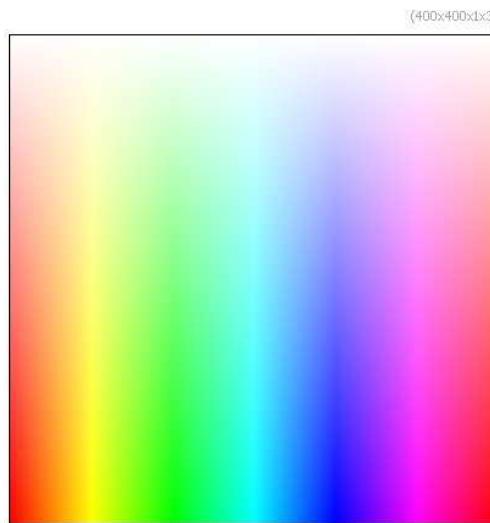
Convert selected images from HSL to RGB colorbases.

### 2.6.15 **-hsl82rgb**

Convert selected images from HSL8 to RGB color bases.

### 2.6.16 **-hsv2rgb (+)**

Convert selected images from HSV to RGB colorbases.



**Example 166 :** `(0,360;0,360^0,0;1,1^1,1;1,1) -resize 400,400,1,3,3 -hsv2rgb`

### 2.6.17 *-hsv2rgb*

Convert selected images from HSV8 to RGB color bases.

### 2.6.18 *-lab2lch*

Convert selected images from Lab to Lch color bases.

### 2.6.19 *-lab2rgb (+)*

Convert selected images from Lab to RGB colorbases.



**Example 167 :** `(50,50;50,50^-3,3;-3,3^-3,-3;3,3) -resize 400,400,1,3,3 -lab2rgb`

**2.6.20 -lab8rgb**

Convert selected images from Lab8 to RGB color bases.

**2.6.21 -lch2lab**

Convert selected images from Lch to Lab color bases.

**2.6.22 -lch2rgb**

Convert selected images from Lch to RGB color bases.

**2.6.23 -lch8rgb**

Convert selected images from Lch8 to RGB color bases.

**2.6.24 -luminance**

Compute luminance of selected sRGB images.



**Example 168 :** image.jpg --luminance

**2.6.25 -mix\_rgb**

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

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

**Default values:** 'a11=1', 'a12=a13=a21=0', 'a22=1', 'a23=a31=a32=0' and 'a33=1'.



**Example 169 :** `image.jpg --mix_rgb 0,1,0,1,0,0,0,0,1`

### 2.6.26 *-pseudogray*

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

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

**Default values:** '`max_increment=5`', '`JND_threshold=2.3`' and '`bits_depth=8`'.



**Example 170 :** `-pseudogray 5`

### 2.6.27 *-replace\_color*

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

Replace pixels from/to specified colors in selected images.



**Example 171 :** image.jpg --replace\_color 40,3,204,153,110,255,0,0

### 2.6.28 -rgb2bayer

**Arguments:** `_start_pattern=0`, `_color_grid=0`

Transform selected color images to RGB-Bayer sampled images.

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



**Example 172 :** image.jpg --rgb2bayer 0

### 2.6.29 -rgb2cmy

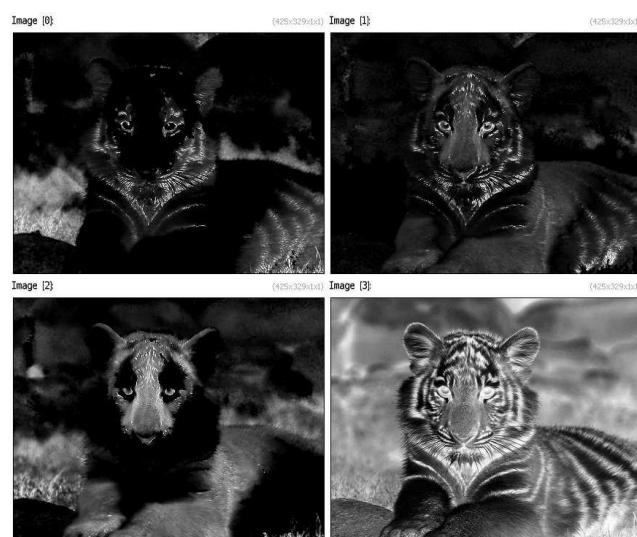
Convert selected images from RGB to CMY colorbases.



**Example 173 :** image.jpg -rgb2cmy -split c

### 2.6.30 -rgb2cmyk

Convert selected images from RGB to CMYK colorbases.



**Example 174 :** image.jpg -rgb2cmyk -split c



**Example 175 :** `image.jpg -rgb2cmyk -split c -fill[3] 0 -append c -cmyk2rgb`

### 2.6.31 `-rgb2hsi (+)`

Convert selected images from RGB to HSI colorbases.



**Example 176 :** `image.jpg -rgb2hsi -split c`

### 2.6.32 `-rgb2hsi8`

Convert selected images from RGB to HSI8 color bases.



**Example 177 :** `image.jpg -rgb2hsi8 -split c`

### 2.6.33 ***-rgb2hsl (+)***

Convert selected images from RGB to HSL colorbases.



**Example 178 :** `image.jpg -rgb2hsl -split c`



**Example 179 :** `image.jpg -rgb2hsl --split c -add[-3] 100 -mod[-3] 360 -append[-3--1] c -hsl2rgb`

### 2.6.34 ***-rgb2hsl8***

Convert selected images from RGB to HSL8 color bases.



**Example 180 :** `image.jpg -rgb2hsl8 -split c`

**2.6.35 -rgb2hsv (+)**

Convert selected images from RGB to HSV colorbases.



**Example 181 :** image.jpg -rgb2hsv -split c



**Example 182 :** image.jpg -rgb2hsv --split c -add[-2] 0.3 -cut[-2] 0,1  
-append[-3--1] c -hsv2rgb

**2.6.36 -rgb2hsv8**

Convert selected images from RGB to HSV8 color bases.



**Example 183 :** image.jpg -rgb2hsv8 -split c

**2.6.37 -rgb2lab (+)**

Convert selected images from RGB to Lab colorbases.



**Example 184 :** `image.jpg -rgb2lab -split c`



**Example 185 :** `image.jpg -rgb2lab --split c -mul[-2,-1] 2.5 -append[-3--1] c -lab2rgb`

**2.6.38 -rgb2lab8**

Convert selected images from RGB to Lab8 color bases.



**Example 186 :** `image.jpg -rgb2lab8 -split c`

**2.6.39 -rgb2lch**

Convert selected images from RGB to Lch color bases.



**Example 187 :** image.jpg -rgb2lch -split c

**2.6.40 -rgb2lch8**

Convert selected images from RGB to Lch8 color bases.



**Example 188 :** image.jpg -rgb2lch8 -split c

**2.6.41 -rgb2luv**

Convert selected images from RGB to LUV color bases.



**Example 189 :** image.jpg -rgb2luv -split c

**2.6.42 -rgb2srgb (+)**

Convert selected images from RGB to sRGB colorbases.

### 2.6.43 ***-rgb2xyz***

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



**Example 190 :** image.jpg -rgb2xyz -split c

### 2.6.44 ***-rgb2xyz8***

Convert selected images from RGB to XYZ8 color bases.



**Example 191 :** image.jpg -rgb2xyz8 -split c

### 2.6.45 ***-rgb2ycbcr***

Convert selected images from RGB to YCbCr colorbases.



**Example 192 :** image.jpg -rgb2ycbcr -split c

**2.6.46 -rgb2yuv**

Convert selected images from RGB to YUV colorbases.



**Example 193 :** `image.jpg -rgb2yuv -split c`

**2.6.47 -rgb2yuv8**

Convert selected images from RGB to YUV8 color bases.



**Example 194 :** `image.jpg -rgb2yuv8 -split c`

**2.6.48 -remove\_opacity**

Remove opacity channel of selected images.

**2.6.49 -select\_color**

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

Select pixels with specified color in selected images.

Image [0]:

(425x329x1x3)



Image [1]:

(425x329x1x1)



**Example 195 :** image.jpg --select\_color 40,204,153,110

### 2.6.50 -sepia

Apply sepia tones effect on selected images.

Image [0]:

(425x329x1x3)



Image [1]:

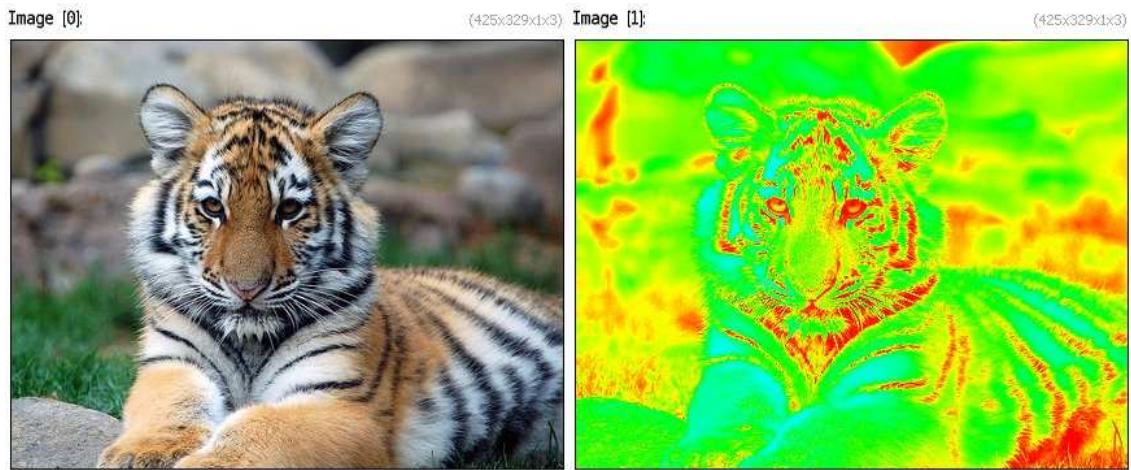
(425x329x1x3)



**Example 196 :** image.jpg --sepia

### 2.6.51 -solarize

Solarize selected images.



**Example 197:** `image.jpg --solarize`

### 2.6.52 `-split_opacity`

Split color and opacity parts of selected images.

### 2.6.53 `-srgb2rgb (+)`

Convert selected images from sRGB to RGB colorbases.

### 2.6.54 `-to_a`

Force selected images to have an alpha channel.

### 2.6.55 `-to_color`

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

### 2.6.56 `-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`' .

### 2.6.57 `-to_gray`

Force selected images to be in GRAY mode.



**Example 198 :** `image.jpg --to-gray`

### 2.6.58 `-to_graya`

Force selected images to be in GRAYA mode.

### 2.6.59 `-to_pseudogray`

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

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

Default parameters : '`max_step=5`', '`is_perceptual_constraint=1`' and '`bits_depth=8`'.

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

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

### 2.6.60 `-to_rgb`

Force selected images to be in RGB mode.

### 2.6.61 `-to_rgba`

Force selected images to be in RGBA mode.

### 2.6.62 `-transfer_colors`

**Arguments:** `_transfer_brightness={ 0 | 1 }`

Transfer colors of the first selected image to the other ones.

**Default value:** 'transfer\_brightness=0' .



**Example 199 :** image.jpg --rand 0,255 -reverse --transfer-colors 1

### 2.6.63 -xyz2rgb

Convert selected images from XYZ to RGB colorbases.

### 2.6.64 -xyz82rgb

Convert selected images from XYZ8 to RGB color bases.

### 2.6.65 -ycbcr2rgb

Convert selected images from YCbCr to RGB colorbases.

### 2.6.66 -yuv2rgb

Convert selected images from YUV to RGB colorbases.

### 2.6.67 -yuv82rgb

Convert selected images from YUV8 to RGB color bases.

## 2.7 Geometry manipulation

### 2.7.1 -append (\*)

**Arguments:** [image],axis,\_alignment |  
axis,\_alignment

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

(eq. to '-a').

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

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

**Default value:** 'alignment=0'.



**Example 200 :** image.jpg -split y,10 -reverse -append y



**Example 201 :** image.jpg -repeat 5 --rows[0] 0,{10+18\*\$>}% -done -rm[0] -append x, 0.5



**Example 202 :** `image.jpg -append[0] [0],y`

## 2.7.2 *-append\_tiles*

**Arguments:** `_M>=0, _N>=0, 0<=_x_alignment<=1, 0<=_y_alignment<=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.

'x\_alignment' and 'y\_alignment' tells about the alignment of tiles when they have different sizes.

**Default values:** `'M=0', 'N=0', 'x_alignment=y_alignment=0.5'`.

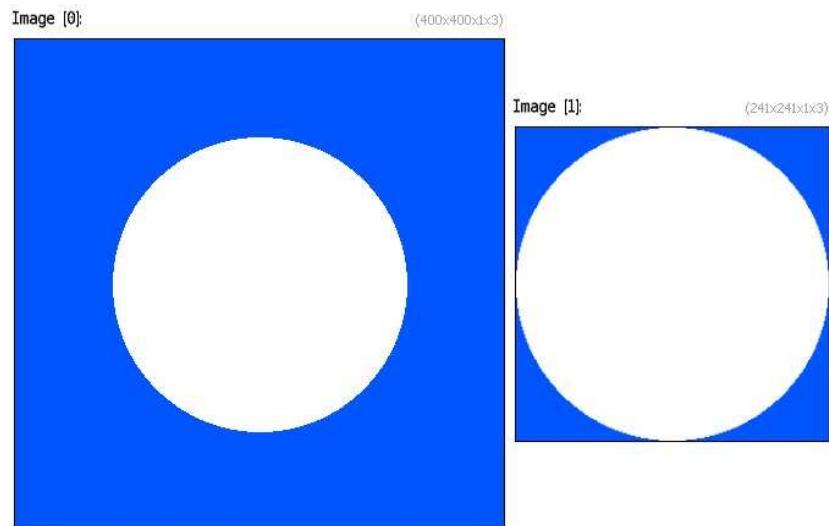


**Example 203:** `image.jpg -split xy,4 -append_tiles ,`

### 2.7.3 *-autocrop* (\*)

**Arguments:** `value1,value2,... |  
(no args)`

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



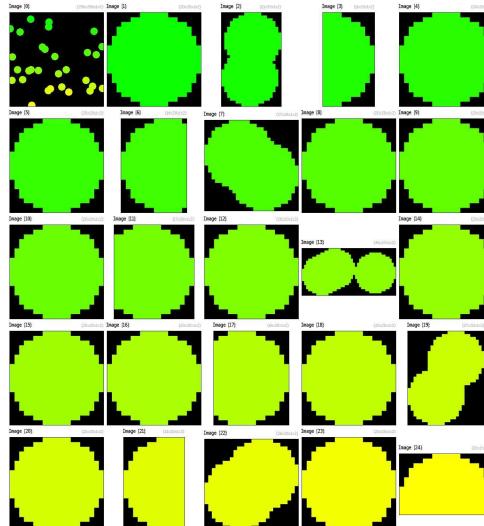
**Example 204:** `400,400,1,3 -fill_color 64,128,255 -ellipse 50%,50%,120,120,0,1,255  
--autocrop`

### 2.7.4 -autocrop\_components

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

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

**Default values:** '`threshold=0%`', '`min_area=0.1%`', '`is_high_connectivity=0`' and '`output_type=1`'.



**Example 205:** `256,256 -noise 0.1,2 -dilate_circ 20 -label_fg 0,1 -n 0,255 --neq 0 -*[ -1 ] 255 -a c --autocrop_components ,`

### 2.7.5 -autocrop\_seq

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

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

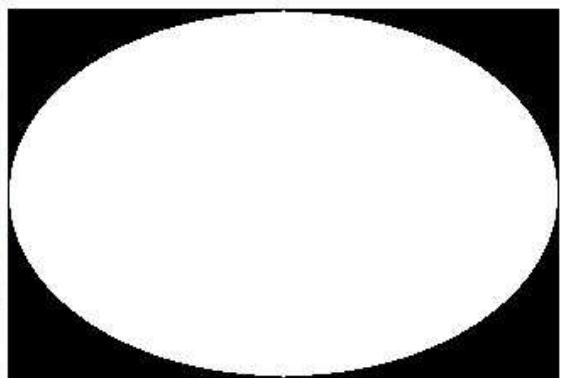
**Default value:** `auto` mode.

**Image [0]:**

(324x217x1x3)

**Image [1]:**

(324x217x1x3)



**Example 206:** `image.jpg --f[-1] 0 -ellipse[-1] 50%,50%,30%,20%,0,1,1 -autocrop-seq 0`

### 2.7.6 *-channels* (\*)

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

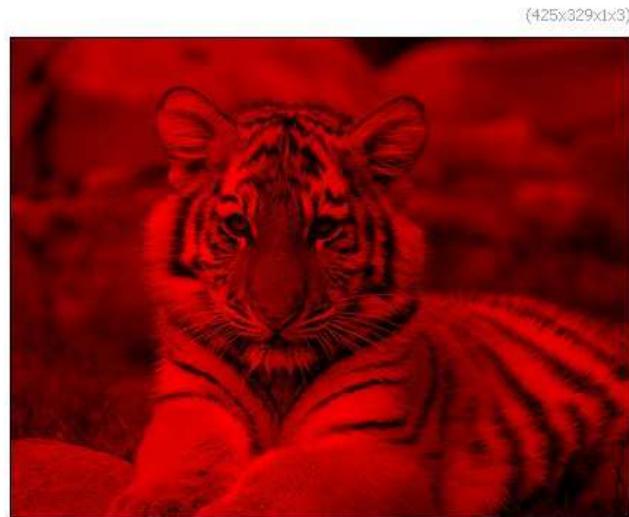
Keep only specified channels of selected images.

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

(425x329x1x2)



**Example 207:** `image.jpg -channels 0,1`



**Example 208 :** image.jpg -luminance -channels 0,2

### 2.7.7 *-columns* (\*)

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

Keep only specified columns of selected images.

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



**Example 209 :** image.jpg -columns -25%, 50%

### 2.7.8 *-crop* (\*)

**Arguments:** x0[%], x1[%], boundary |

```
x0[%],y0[%],x1[%],y1[%],_boundary |
x0[%],y0[%],z0[%],x1[%],y1[%],z1[%],_boundary |
x0[%],y0[%],z0[%],c0[%],x1[%],y1[%],z1[%],c1[%],_boundary |
(noargs)
```

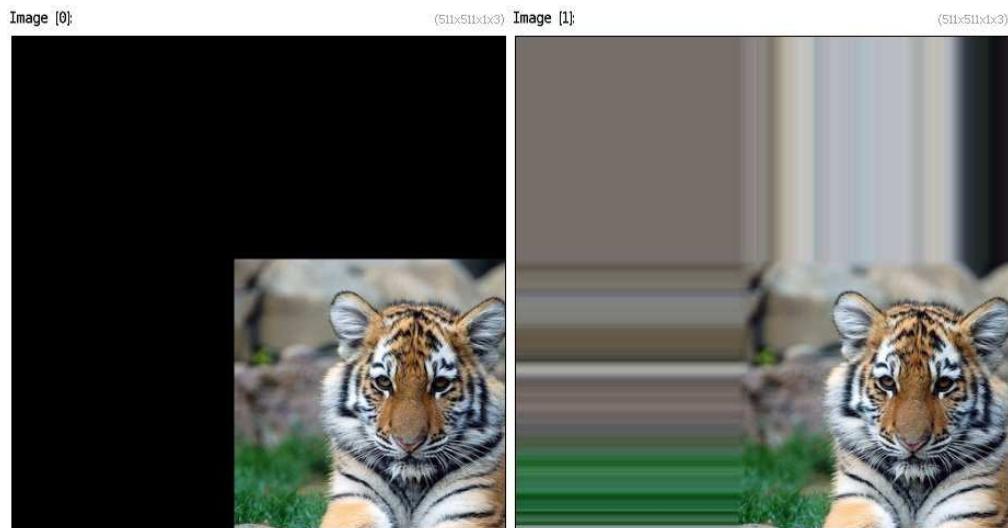
Crop selected images with specified region coordinates.

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

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

(noargs) runs interactive mode (uses the instant window [0] if opened).

**Default value:** 'boundary=0' .



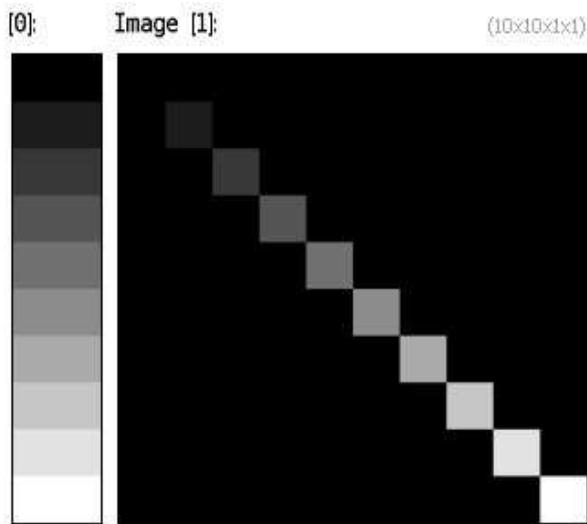
**Example 210 :** image.jpg --crop -230,-230,280,280,1 -crop[0] -230,-230,280,280,0



**Example 211 :** image.jpg -crop 25%,25%,75%,75%

### 2.7.9 *-diagonal*

Transform selected vectors as diagonal matrices.



**Example 212 :** `1,10,1,1,'y' --diagonal`

### 2.7.10 *-elevate*

**Arguments:** `_depth, _is_plain, _is_colored`

Elevate selected 2d images into 3d volumes.

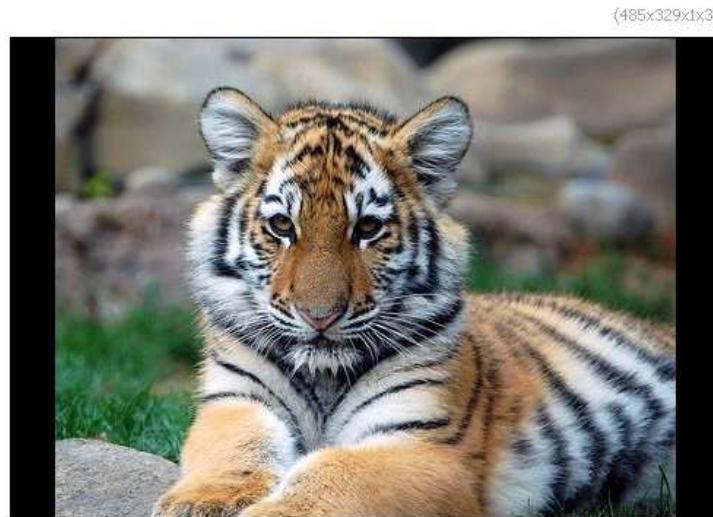
**Default values:** '`depth=64`', '`is_plain=1`' and '`is_colored=1`'.

### 2.7.11 *-expand\_x*

**Arguments:** `size_x>=0, _boundary={ 0=dirichlet | 1=neumann | 2=cyclic }`

Expand selected images along the x-axis.

**Default value:** '`border=1`'.



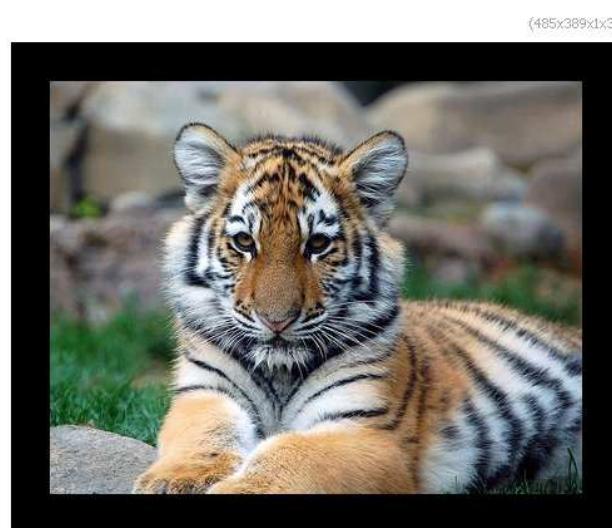
**Example 213 :** image.jpg -expand\_x 30,0

### 2.7.12 *-expand\_xy*

**Arguments:** size $\geq 0$ , boundary={ 0=dirichlet | 1=neumann | 2=cyclic }

Expand selected images along the xy-axes.

**Default value:** 'border=1' .



**Example 214 :** image.jpg -expand\_xy 30,0

### 2.7.13 -expand\_xyz

**Arguments:** size $\geq 0$ , boundary={ 0=dirichlet | 1=neumann | 2=cyclic }

Expand selected images along the xyz-axes.

**Default value:** 'border=1'.

### 2.7.14 -expand\_y

**Arguments:** size\_y $\geq 0$ , boundary={ 0=dirichlet | 1=neumann | 2=cyclic }

Expand selected images along the y-axis.

**Default value:** 'border=1'.



**Example 215 :** image.jpg -expand\_y 30,0

### 2.7.15 -expand\_z

**Arguments:** size\_z $\geq 0$ , boundary={ 0=dirichlet | 1=neumann | 2=cyclic }

Expand selected images along the z-axis.

**Default value:** 'border=1'.

### 2.7.16 -mirror (\*)

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

Mirror selected images along specified axes.



**Example 216 :** `image.jpg --mirror y --mirror[0] c`



**Example 217 :** `image.jpg --mirror x --mirror y -append_tiles 2,2`

### 2.7.17 *-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 218 :** image.jpg -permute yxzc

### 2.7.18 **-resize** (\*)

**Arguments:** [image], -interpolation, -boundary, -ax, -ay, -az, -ac | {[image\_w] | width>0[%]}, {[image\_h] | height>0[%]}, {[image\_d] | depth>0[%]}, {[image\_s] | spectrum>0[%]}, -interpolation, -boundary, -ax, -ay, -az, -ac | (noargs)

Resize selected images with specified geometry.

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

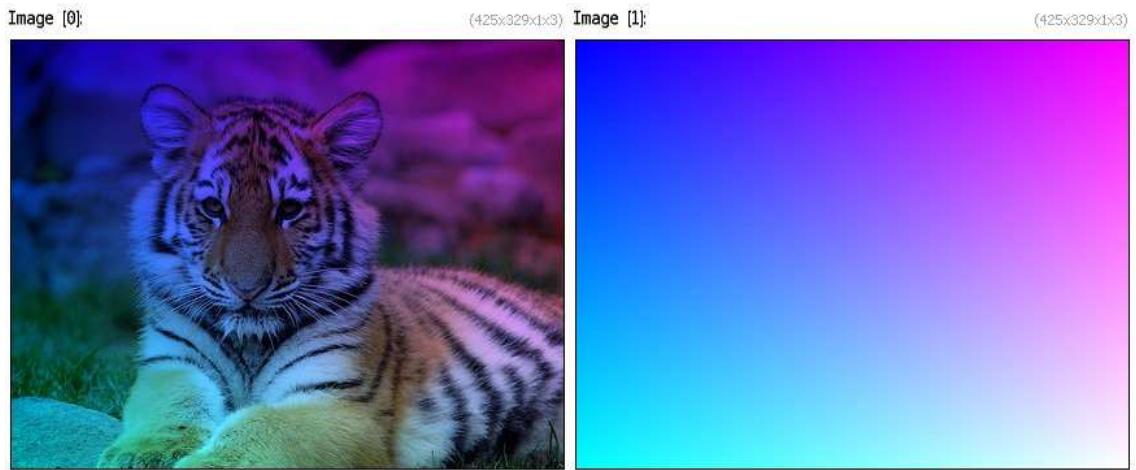
'interpolation' can be { -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }.

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

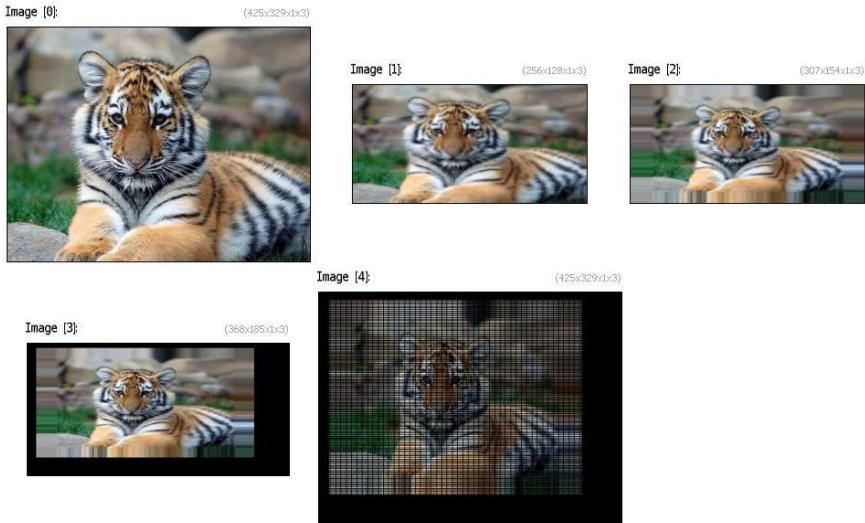
'ax,ay,az,ac' set the alignment mode along each axis when 'interpolation=0 or 4' (set to '0' by default, must be defined in range [0,1]).

(noargs) runs interactive mode (uses the instant window [0] if opened).

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



**Example 219:** `image.jpg (0,1;0,1^0,0;1,1^1,1;1,1) -resize[-1] [-2],3 -mul[-2] [-1]`



**Example 220:** `image.jpg --resize[-1] 256,128,1,3,2 --resize[-1] 120%,120%,1,3,0,1,0.5,0.5 --resize[-1] 120%,120%,1,3,0,0,0.2,0.2 --resize[-1] [0],[0],1,3,4`

### 2.7.19 *-pow2*

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

Resize selected images so that each dimension is a power of 2.

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

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

| 6 }}, 'boundary' can be { 0=none | 1=neumann }.  
 'ax,ay,az,ac' set the alignment mode along each axis when 'interpolation=0'  
 (set to '0' by default, must be defined in range [0,1]).

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



**Example 221 :** image.jpg --resize\_pow2 [-1] 0

### 2.7.20 -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' .

### 2.7.21 -resize2dx

**Arguments:** width>0, \_interpolation, \_boundary, \_ax, \_ay, \_az, \_ac

Resize selected images along the x-axis, preserving 2d ratio.

(eq. to '-r2dx').

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

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

| 6 }', 'boundary' can be { 0=none | 1=neumann }.

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

**Default values:** 'interpolation=3', 'boundary=0' and 'ax=ay=az=ac=0'.



**Example 222:** image.jpg --resize2dx 100,2 -append x

### 2.7.22 -resize2dy

**Arguments:** height>0,\_interpolation,\_boundary,\_ax,\_ay,\_az,\_ac

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

(*eq. to '-r2dy'*).

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

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

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

**Default values:** 'interpolation=3', 'boundary=0' and 'ax=ay=az=ac=0'.



**Example 223 :** image.jpg --resize2dy 100,2 -append x

### 2.7.23 -resize3dx

**Arguments:** width>0, interpolation, boundary, ax, ay, az, ac

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

(eq. to '-r3dx').

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

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

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

**Default values:** 'interpolation=3', 'boundary=0' and 'ax=ay=az=ac=0' .

### 2.7.24 -resize3dy

**Arguments:** height>0, interpolation, boundary, ax, ay, az, ac

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

(eq. to '-r3dy').

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

'boundary' has different meanings, according to the chosen 'interpolation' mode : . When

'interpolation=={ -1 | 1 | 2 | 4 }', 'boundary' is meaningless. . When 'interpolation==0', 'boundary' can be { 0=dirichlet | 1=neumann | 2=cyclic }. . When 'interpolation=={ 3 | 5 | 6 }', 'boundary' can be { 0=none | 1=neumann }.  
 'ax,ay,az,ac' set the alignment mode along each axis when 'interpolation=0' (set to '0' by default, must be defined in range [0,1]).

**Default values:** 'interpolation=3', 'boundary=0' and 'ax=ay=az=ac=0'.

### 2.7.25 *-resize3dz*

**Arguments:** depth>0, \_interpolation, \_boundary, \_ax, \_ay, \_az, \_ac

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

(eq. to '-r3dz').

'interpolation' can be { -1=none (memory content) | 0=none | 1=nearest | 2=average | 3=linear | 4=grid | 5=bicubic | 6=lanczos }.  
 'boundary' has different meanings, according to the chosen 'interpolation' mode : . When 'interpolation=={ -1 | 1 | 2 | 4 }', 'boundary' is meaningless. . When 'interpolation==0', 'boundary' can be { 0=dirichlet | 1=neumann | 2=cyclic }. . When 'interpolation=={ 3 | 5 | 6 }', 'boundary' can be { 0=none | 1=neumann }.  
 'ax,ay,az,ac' set the alignment mode along each axis when 'interpolation=0' (set to '0' by default, must be defined in range [0,1]).

**Default values:** 'interpolation=3', 'boundary=0' and 'ax=ay=az=ac=0'.

### 2.7.26 *-rotate* (\*)

**Arguments:** angle, \_interpolation, \_boundary, \_cx[%], \_cy[%], \_zoom

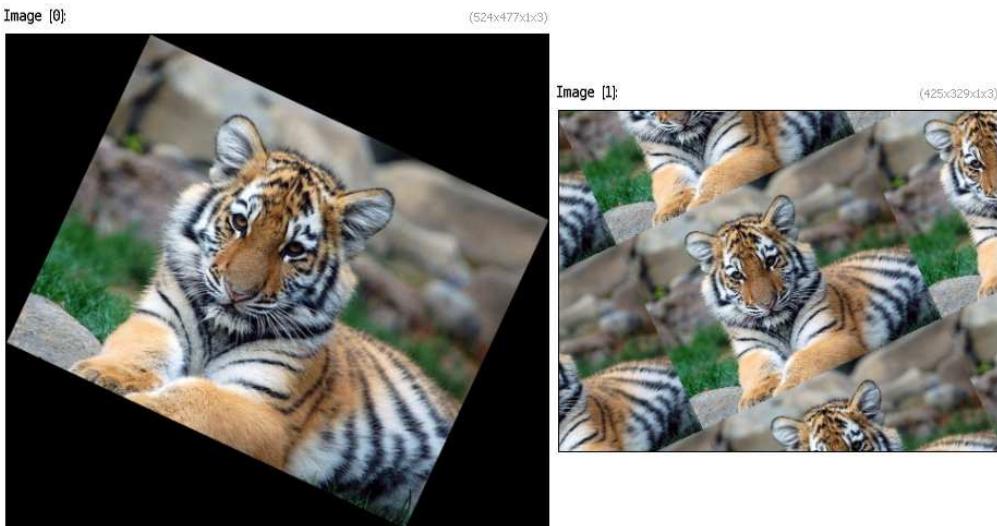
Rotate selected images with specified angle (in deg.).

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

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

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

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



**Example 224 :** `image.jpg --rotate -25,1,2,50%,50%,0.6 -rotate[0] 25`

### 2.7.27 *-rotate tileable*

**Arguments:** `angle, max_size_factor>=0`

Rotate selected images by specified angle and make them tileable.  
If resulting size of an image is too big, the image is replaced by a 1x1 image.

**Default values:** '`max_size_factor=8`'.

### 2.7.28 *-rows (\*)*

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

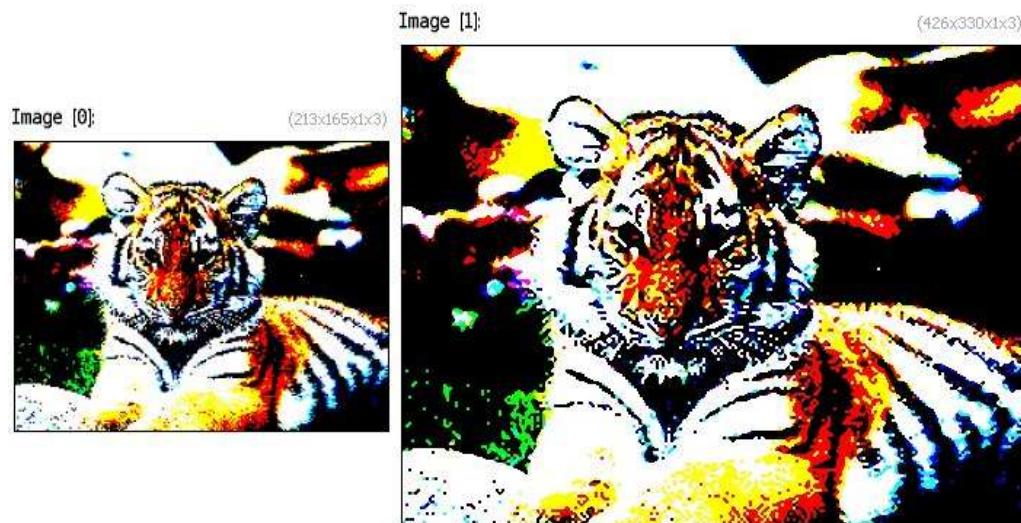
Keep only specified rows of selected images.  
Dirichlet boundary is used when specified rows are out of range.



**Example 225 :** image.jpg -rows -25%, 50%

### 2.7.29 *-scale2x*

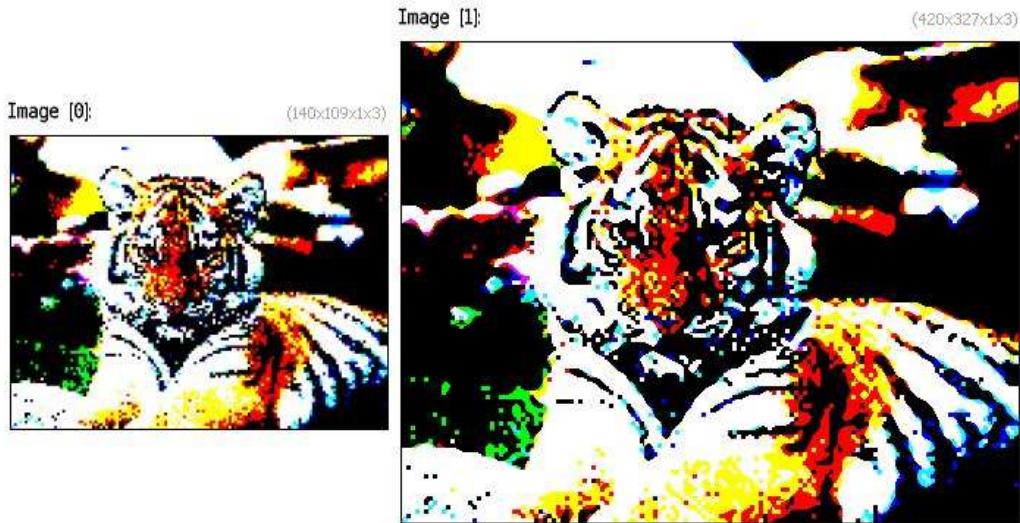
Resize selected images using the Scale2x algorithm.



**Example 226 :** image.jpg -threshold 50% -resize 50%, 50% --scale2x

### 2.7.30 *-scale3x*

Resize selected images using the Scale3x algorithm.



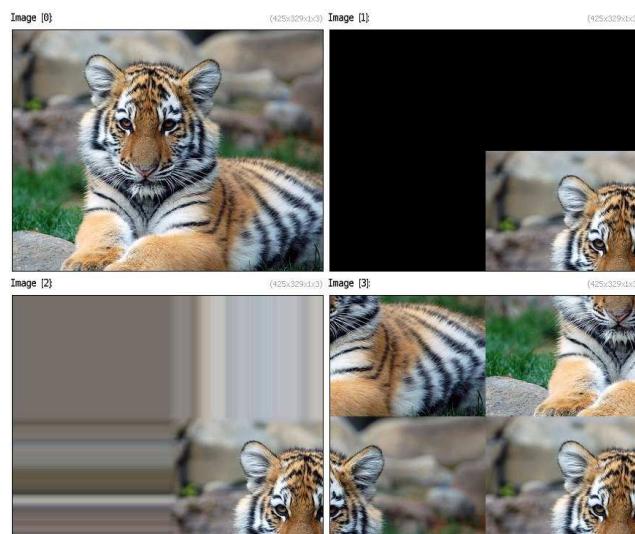
**Example 227 :** image.jpg -threshold 50% -resize 33%,33% --scale3x

### 2.7.31 **-shift (\*)**

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

Shift selected images by specified displacement vector.  
'boundary' can be { 0=dirichlet | 1=neumann | 2=cyclic }.

**Default value:** 'boundary=0' .



**Example 228 :** image.jpg --shift[0] 50%,50%,0,0,0 --shift[0] 50%,50%,0,0,1  
--shift[0] 50%,50%,0,0,2

### 2.7.32 *-shrink\_x*

**Arguments:** size\_x>=0

Shrink selected images along the x-axis.

(365x329x1x3)



**Example 229:** image.jpg -shrink\_x 30

### 2.7.33 *-shrink\_xy*

**Arguments:** size>=0

Shrink selected images along the xy-axes.

(365x269x1x3)



**Example 230:** image.jpg -shrink\_xy 30

**2.7.34 -shrink\_xyz****Arguments:** size>=0

Shrink selected images along the xyz-axes.

**2.7.35 -shrink\_y****Arguments:** size\_y>=0

Shrink selected images along the y-axis.



**Example 231 :** image.jpg -shrink\_y 30

**2.7.36 -shrink\_z****Arguments:** size\_z>=0

Shrink selected images along the z-axis.

**2.7.37 -slices (\*)****Arguments:** { [image0] | z0[%] }, -{ [image1] | z1[%] }

Keep only specified slices of selected images.

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

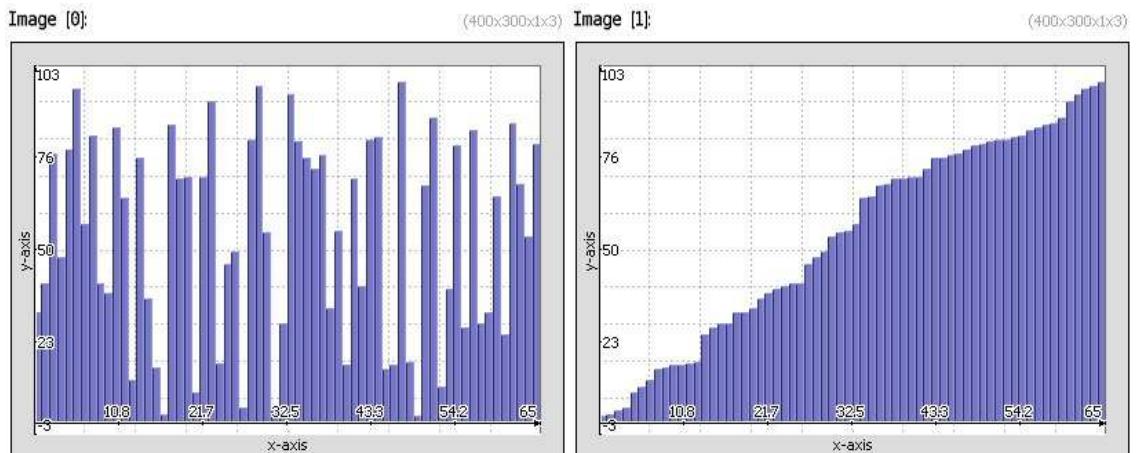
### 2.7.38 -sort (+)

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

Sort pixel values of selected images.

If 'axis' is specified, the sorting is done according to the data of the first column/row/slice/channel of selected images.

**Default values:** '`ordering=+'` and '`axis=(undefined)'`.



**Example 232 :** `64 -rand 0,100 --sort -display_graph 400,300,3`

### 2.7.39 -split (\*)

**Arguments:** `{ x | y | z | c }..{ x | y | z | c },_nb_parts | keep_splitting_values={ + | - },value1,value2,...`

Split selected images along specified axis, or sequence of scalar values.

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

'`nb_parts`' can be { 0=maximum split | >0=split in N parts | <0=split in parts of size -N }.

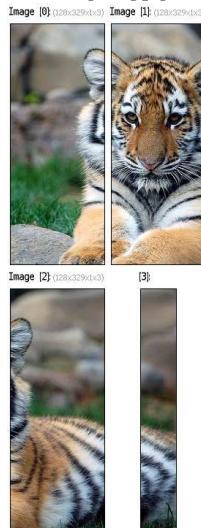
**Default value:** '`nb_parts=0`'.



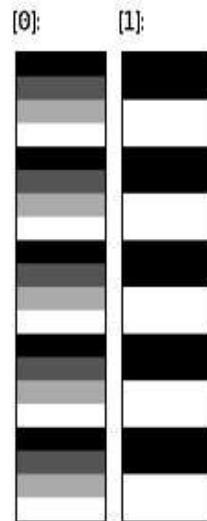
**Example 233 :** image.jpg -split c



**Example 234 :** image.jpg -split y, 3



**Example 235 :** image.jpg -split x, -128



**Example 236 :** `1,20,1,1,"1,2,3,4" --split -,2,3 -append[1--1] y`

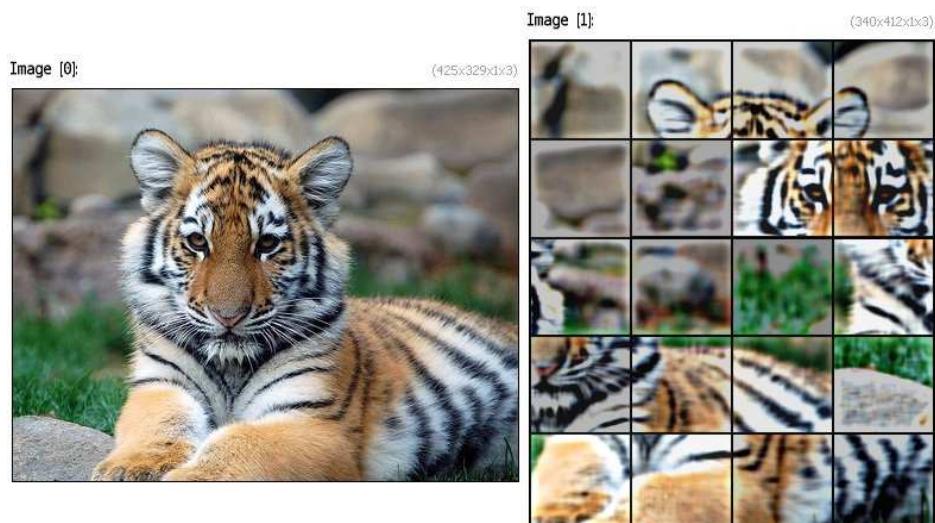
### 2.7.40 *-split\_tiles*

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

Split selected images as a MxN array of tiles.

If M or N is negative, it stands for the tile size instead.

**Default values:** '`N=M`' and '`_is_homogeneous=0`'.

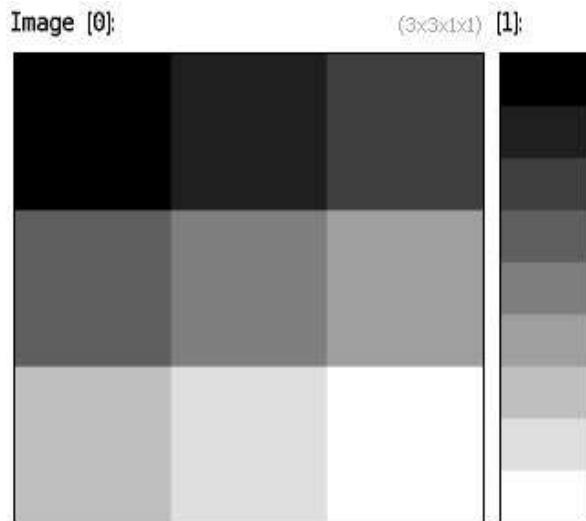


**Example 237 :** `image.jpg --local -split_tiles 5,4 -blur 3,0 -sharpen 700 -append_tiles 4,5 -endlocal`

**2.7.41 -unroll (\*)**

**Arguments:** axis={ x | y | z | c }

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



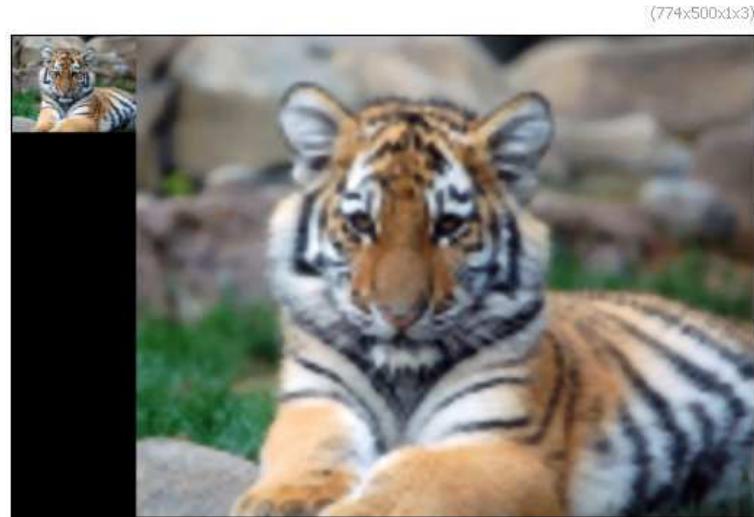
**Example 238 :** (1, 2, 3; 4, 5, 6; 7, 8, 9) --unroll y

**2.7.42 -upscale\_smart**

**Arguments:** width,\_height,\_depth,\_smoothness>=0,\_anisotropy=[0,1],sharpening>=0

Upscale selected images with an edge-preserving algorithm.

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



**Example 239 :** `image.jpg -resize2dy 100 --upscale_smart 500%,500% -append x`

#### 2.7.43 `-warp (+)`

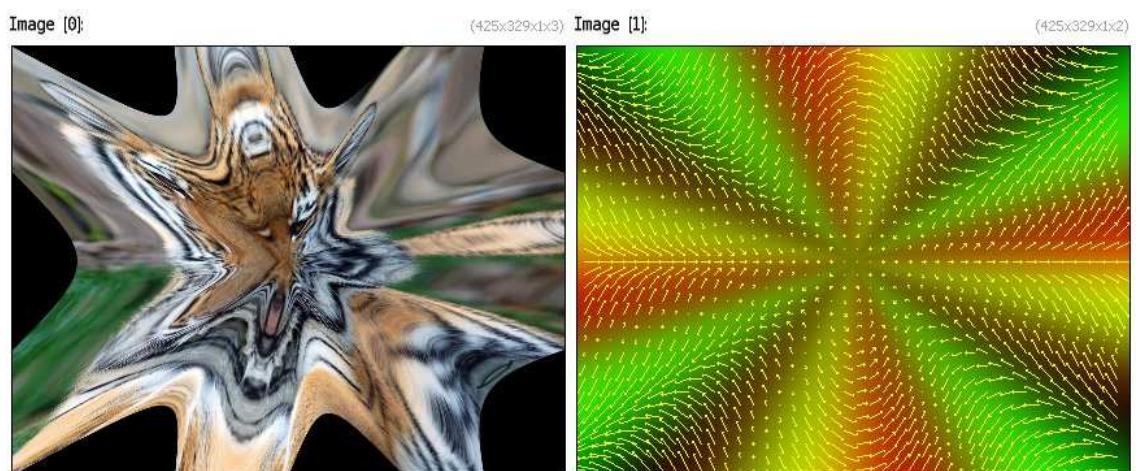
**Arguments:** [warping\_field], `_is_relative={ 0 | 1 }`  
`,_interpolation, _boundary, _nb_frames>0`

Warp selected image with specified displacement field.

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

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

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



**Example 240 :** `image.jpg 100%,100%,1,2,'X=x/w-0.5;Y=y/h-0.5;R=(X*X+Y*Y)^0.5;A=atan2(Y,X);130*R*if(c==0,cos(4*A),sin(4*A)) -warp (+)`

```
-warp[-2] [-1],1,1,0 -quiver[-1] [-1],10,0.2,1,1,100
```

## 2.8 Filtering

### 2.8.1 *-bandpass*

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

Apply bandpass filter to selected images.

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



**Example 241 :** `image.jpg -bandpass 1%,3%`

### 2.8.2 *-bilateral (+)*

**Arguments:** `[guide]`, `std_variation_s>0[%]`, `std_variation_r>0` |  
`std_variation_s>0[%]`, `std_variation_r>0`

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

If a guide image is provided, it is used for computing the smoothing geometry in the cross bilateral filter.

A guide image must be of the same xyz-size as the selected images.



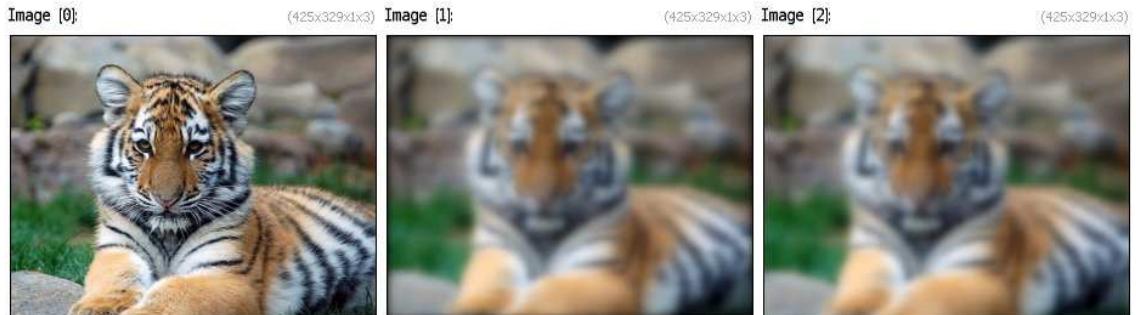
**Example 242 :** `image.jpg [0] -repeat 5 -bilateral[-1] 10,10 -done`

### 2.8.3 *-blur (+)*

**Arguments:** `std_variation>=0 [%], boundary={ 0=dirichlet | 1=neumann }, _kernel={ 0=quasi-gaussian (faster) | 1=gaussian }`

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

**Default value:** '`boundary=1`' and '`kernel=0`' .



**Example 243 :** `image.jpg --blur 5,0 --blur[0] 5,1`

### 2.8.4 *-blur\_angular*

**Arguments:** `amplitude[%], _cx, _cy`

Apply angular blur on selected images.

**Default values:** '`cx=cy=0.5`' .



**Example 244 :** image.jpg --blur\_angular 2%

### 2.8.5 -blur\_linear

**Arguments:** `amplitude1[%], -amplitude2[%], -angle, -boundary={ 0=dirichlet | 1=neumann }`

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

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



**Example 245 :** image.jpg --blur\_linear 10,0,45

### 2.8.6 -blur\_radial

**Arguments:** `amplitude[%], -cx, -cy`

Apply radial blur on selected images.

**Default values:** ' cx=cy=0.5' .



**Example 246 :** image.jpg --blur\_radial 2%

### 2.8.7 -blur\_selective

**Arguments:** sigma>=0, edges>0, nb\_scales>0

Blur selected images using selective gaussian scales.

**Default values:** ' sigma=5' , ' edges=0.5' and ' nb\_scales=5' .



**Example 247 :** image.jpg -noise 20 -cut 0,255 --l[-1] -repeat 4 -blur\_selective ,  
-done -endl

### 2.8.8 -blur\_x

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

Blur selected images along the x-axis.

**Default value:** 'boundary=1' .



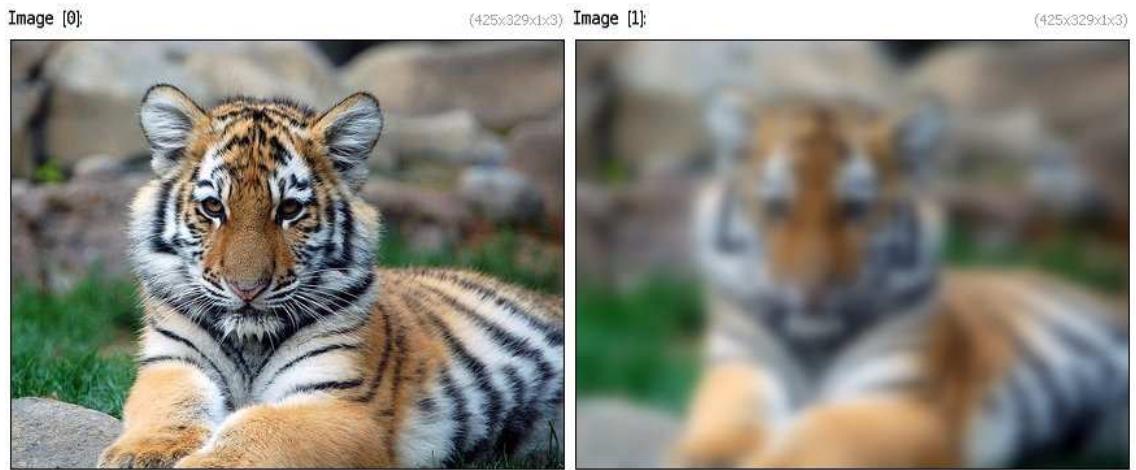
**Example 248:** image.jpg --blur\_x 6

### 2.8.9 -blur\_xy

**Arguments:** amplitude\_x[%], amplitude\_y[%], \_boundary={ 0=dirichlet | 1=neumann }

Blur selected images along the X and Y axes.

**Default value:** 'boundary=1' .



**Example 249 :** `image.jpg --blur_xy 6`

### 2.8.10 `-blur_xyz`

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

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

**Default value:** '`boundary=1`' .

### 2.8.11 `-blur_y`

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

Blur selected images along the y-axis.

**Default value:** '`boundary=1`' .



**Example 250:** `image.jpg --blur_y 6`

### 2.8.12 *-blur\_z*

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

Blur selected images along the z-axis.

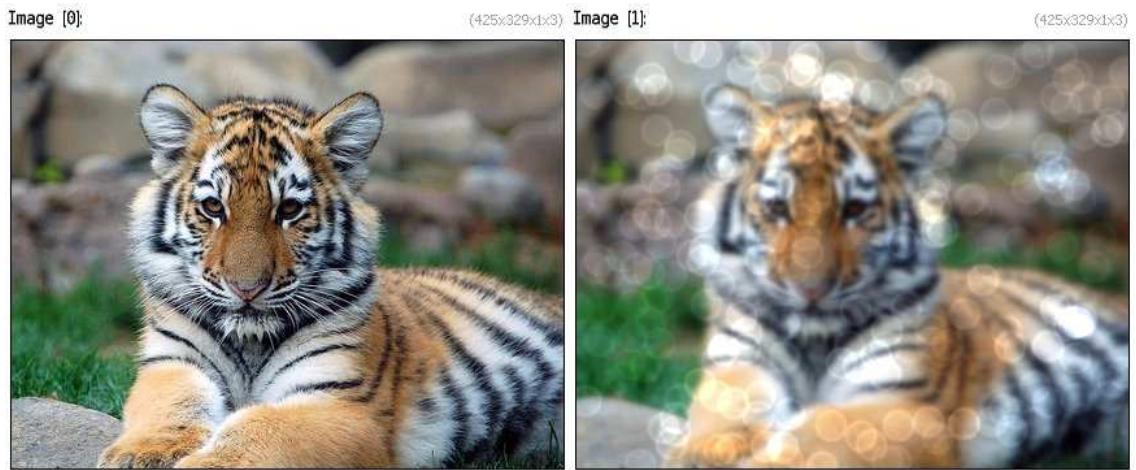
**Default value:** `'boundary=1'`.

### 2.8.13 *-bokeh*

**Arguments:** `_amplitude>=0, _smoothness>=0, 0<=_density<=100, _bokeh_size>0, 0<=_bokeh_outline_size<=100`

Create a Bokeh effect from selected images.

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



**Example 251 :** `image.jpg --bokeh ,`

#### 2.8.14 *-compose freq*

Compose selected low and high frequency parts into new images.



**Example 252 :** `image.jpg -split_freq 2% -mirror[-1] x -compose_freq`

#### 2.8.15 *-convolve (+)*

**Arguments:** `[mask], _boundary, _is_normalized={ 0 | 1 }`

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

**Default values:** 'boundary=1' and 'is\_normalized=0'.



**Example 253 :** `image.jpg (0,1,0;1,-4,1;0,1,0) -convolve[-2] [-1] -keep[-2]`



**Example 254 :** `image.jpg (0,1,0) -resize[-1] 130,1,1,1,3 --convolve[0] [1]`

### 2.8.16 *-convolve\_fft*

Convolve selected images two-by-two through fourier transforms.



**Example 255 :** `image.jpg 100%,100% -gaussian[-1] 20,1,45 --convolve_fft`

### 2.8.17 *-correlate (+)*

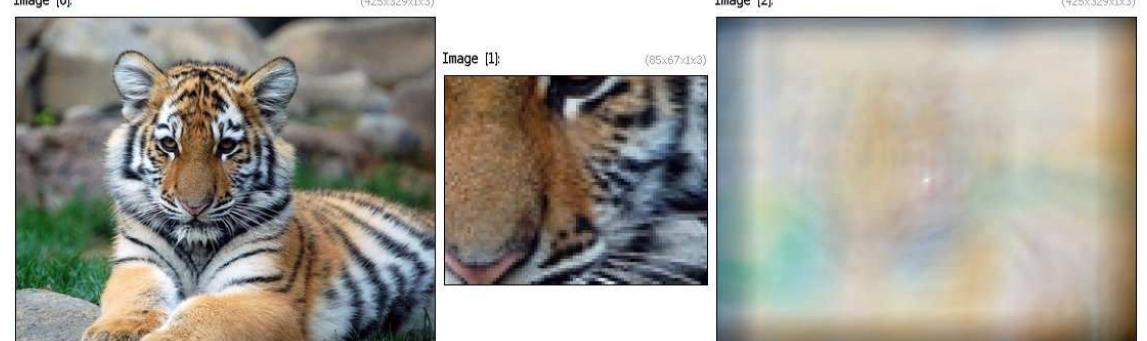
**Arguments:** [mask], \_boundary, \_is\_normalized={ 0 | 1 }

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

**Default values:** 'boundary=1' and 'is\_normalized=0'.



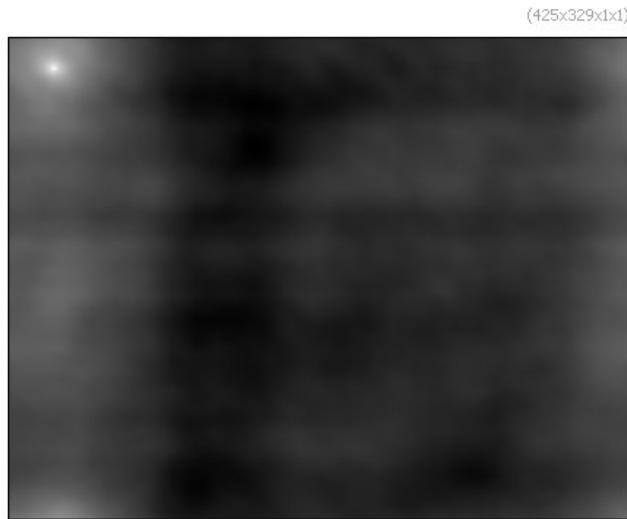
**Example 256 :** image.jpg (0,1,0;1,-4,1;0,1,0) -correlate[-2] [-1] -keep[-2]



**Example 257 :** image.jpg --crop 40%,40%,60%,60% --correlate[0] [-1],0,1

### 2.8.18 *-cross\_correlation*

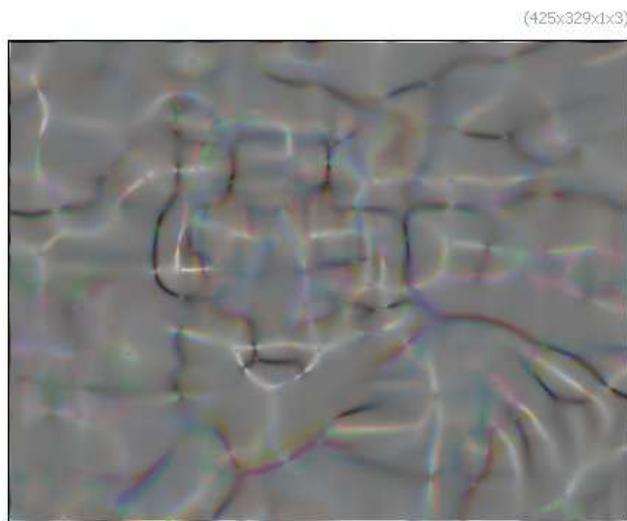
Compute cross-correlation using two-by-two selected images.



**Example 258 :** image.jpg --shift -30,-20 -cross\_correlation

### 2.8.19 -curvature

Compute isophote curvatures on selected images.



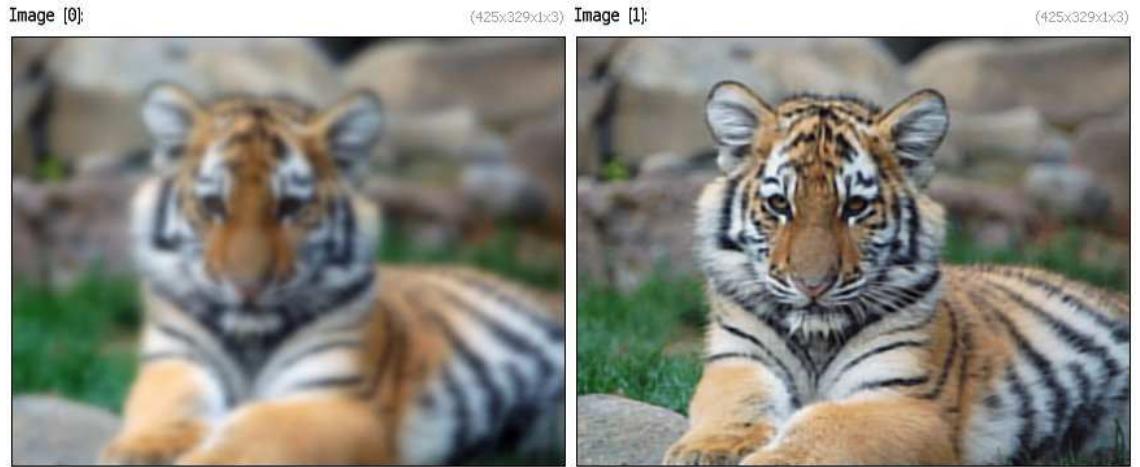
**Example 259 :** image.jpg -blur 10 -curvature

### 2.8.20 -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 260 :** image.jpg --blur 3 --deblur 3,40,20,0.01

### 2.8.21 -deblur goldmeinel

**Arguments:** sigma>=0, nb\_iter>=0, acceleration>=0, kernel\_type={ 0=quasi-gaussian (faster) | 1=gaussian }.

Deblur selected images using Gold-Meinel algorithm

**Default values:** 'nb\_iter=8', 'acceleration=1' and 'kernel\_type=1'.



**Example 261 :** image.jpg --blur 1 --deblur\_goldmeinel[-1] 1

### 2.8.22 -deblur richardsonlucy

**Arguments:** sigma>=0, nb\_iter>=0, kernel\_type={ 0=quasi-gaussian (faster) | 1=gaussian }.

Deblur selected images using Richardson-Lucy algorithm.

**Default values:** 'nb\_iter=50' and 'kernel\_type=1'.



**Example 262 :** image.jpg --blur 1 --deblur richardsonlucy[-1] 1

### 2.8.23 -deconvolve fft

Deconvolve selected images two-by-two through fourier transforms.

### 2.8.24 -deinterlace

**Arguments:** \_method={ 0 | 1 }

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

**Default value:** 'method=0'.



**Example 263 :** image.jpg --rotate 3,1,1,50%,50% -resize 100%,50% -resize 100%,200%,1,3,4 -shift[-1] 0,1 -add --deinterlace 1

### 2.8.25 *-denoise (+)*

**Arguments:** `std_variation_s>=0, std_variation_p>=0, patch_size>=0, lookup_size>=0, s`  
`0 | 1 }`

Denoise selected images by non-local patch averaging.

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



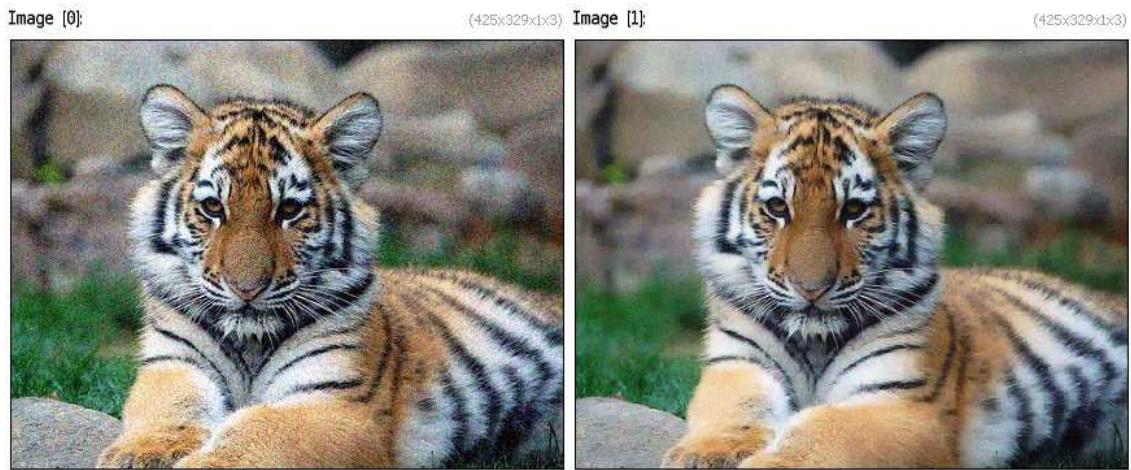
**Example 264 :** `image.jpg --denoise 5,5,8`

### 2.8.26 *-denoise haar*

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

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

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



**Example 265 :** `image.jpg -noise 20 -c 0,255 --denoise_haar[-1] 0.8`

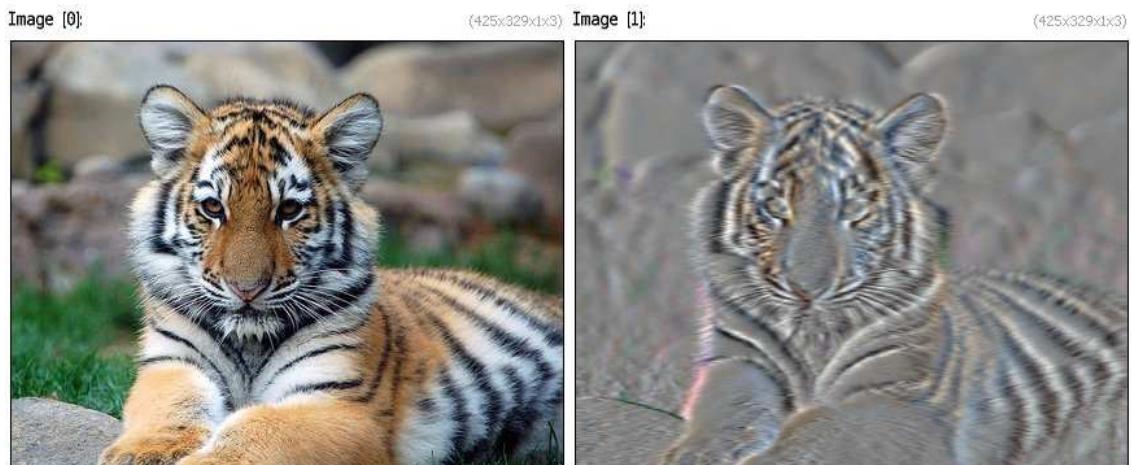
### 2.8.27 *-deriche (+)*

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

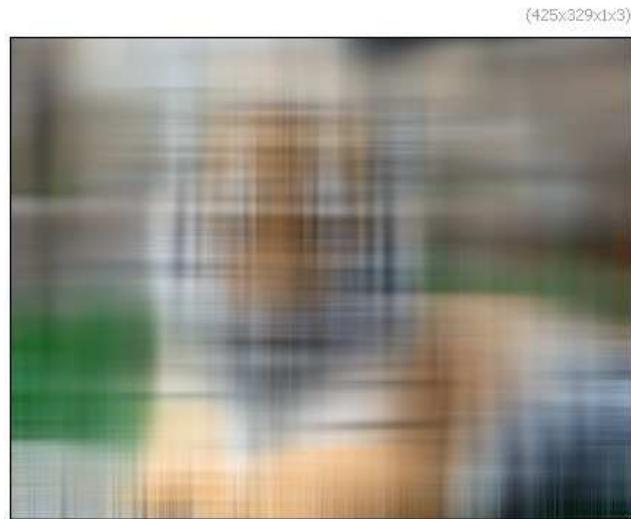
Apply Deriche recursive filter with specified standard deviation, order, axis and border conditions on selected images.

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

**Default value:** 'boundary=1'.



**Example 266 :** `image.jpg --deriche 3,1,x`



**Example 267 :** `image.jpg --deriche 30,0,x -deriche[-2] 30,0,y -add`

### 2.8.28 *-dilate (+)*

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

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

**Default values:** 'size\_z=1', 'boundary=1' and 'is\_normalized=0'.



**Example 268 :** `image.jpg --dilate 10`

### 2.8.29 *-dilate\_circ*

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

Apply circular dilation of selected image by specified size.

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



**Example 269 :** `image.jpg --dilate_circ 7`

### 2.8.30 *-dilate\_oct*

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

Apply octagonal dilation of selected image by specified size.

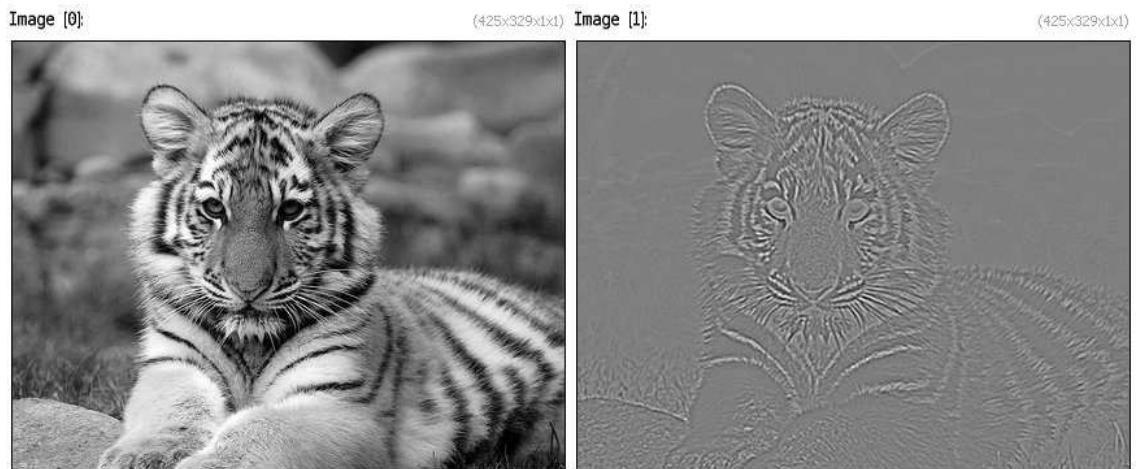
**Default values:** '`boundary=1`' and '`is_normalized=0`'.



**Example 270 :** `image.jpg --dilate_oct 7`

### 2.8.31 *-divergence*

Compute divergence of selected vector fields.



**Example 271 :** `image.jpg -luminance --gradient -a[-2,-1] c -divergence[-1]`

### 2.8.32 *-dog*

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

Compute difference of gaussian on selected images.

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



**Example 272 :** `image.jpg --dog 2,3`

**2.8.33 -diffusiontensors**

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

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

**Default values:** `'sharpness=0.7', 'anisotropy=0.3', 'alpha=0.6', 'sigma=1.1' and 'is_sqrt=0'.`



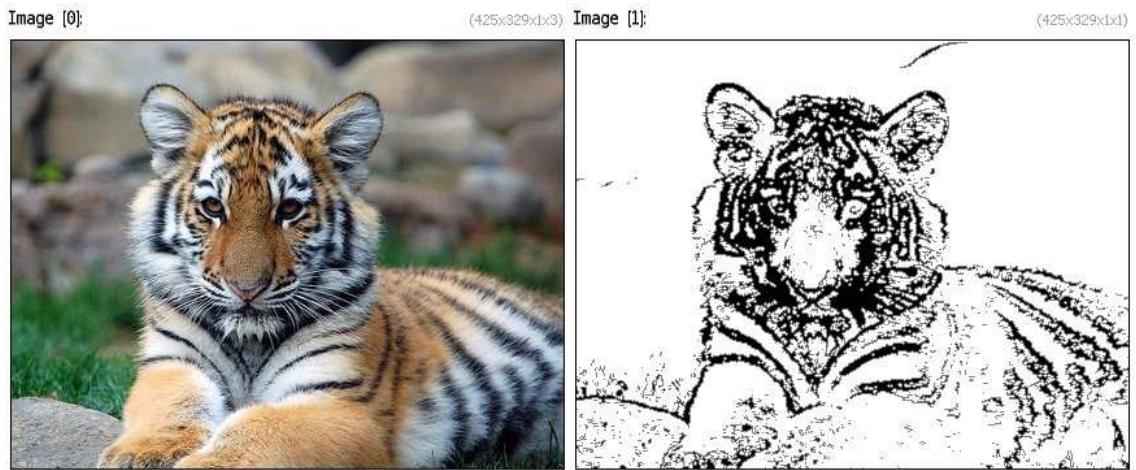
**Example 273 :** `image.jpg -diffusiontensors 0.8 -abs -pow 0.2`

**2.8.34 -edges**

**Arguments:** `_threshold[%]>=0`

Estimate contours of selected images.

**Default value:** `'edges=15%'`



**Example 274 :** `image.jpg --edges 15%`

### 2.8.35 *-eikonal* (+)

**Arguments:** `nb_iterations>=0, band_size>=0`

Compute iterations of the eikonal equation (signed distance function) on selected images.  
When 'band\_size==0', the algorithm performs on all image pixels.

**Default value:** '`band_size=0`' .



**Example 275 :** `image.jpg -blur 3 -threshold 50% -eikonal 40`

### 2.8.36 *-erode* (+)

**Arguments:** `size>=0 |`

```
size_x>=0, size_y>=0, size_z>=0 |
[mask], boundary, is_normalized={ 0 | 1 }
```

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

**Default values:** 'size\_z=1', 'boundary=1' and 'is\_normalized=0'.



**Example 276 :** image.jpg --erode 10

### 2.8.37 -erode\_circ

**Arguments:** \_size>=0, \_boundary, \_is\_normalized={ 0 | 1 }

Apply circular erosion of selected images by specified size.

**Default values:** 'boundary=1' and 'is\_normalized=0'.



**Example 277 :** image.jpg --erode\_circ 7

### 2.8.38 *-erode\_oct*

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

Apply octagonal erosion of selected images by specified size.

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



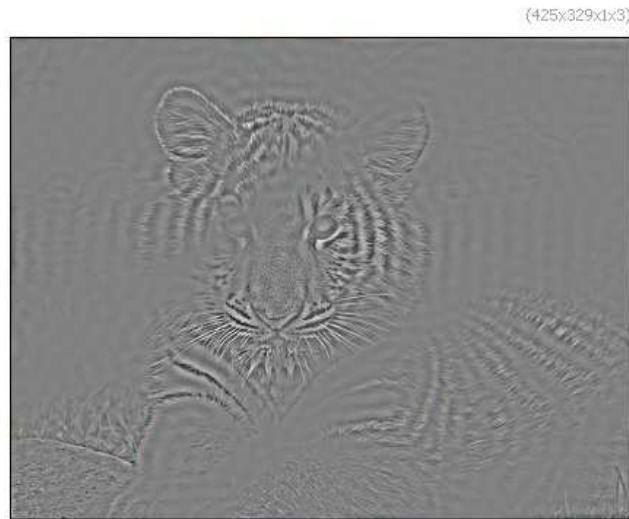
**Example 278 :** image.jpg --erode\_oct 7

### 2.8.39 *-fft (+)*

Compute the direct fourier transform (real and imaginary parts) of selected images.



**Example 279 :** image.jpg -luminance --fft -append[-2,-1] c -norm[-1] -log[-1] -shift[-1] 50%,50%,0,0,2



```
Example 280: image.jpg -fft -shift 50%,50%,0,0,2 -ellipse 50%,50%,30,30,0,1,0  
-shift -50%,-50%,0,0,2 -ifft -remove[-1]
```

#### 2.8.40 -gradient (+)

**Arguments:** { x | y | z }..{ x | y | z },\_scheme |  
(no args)

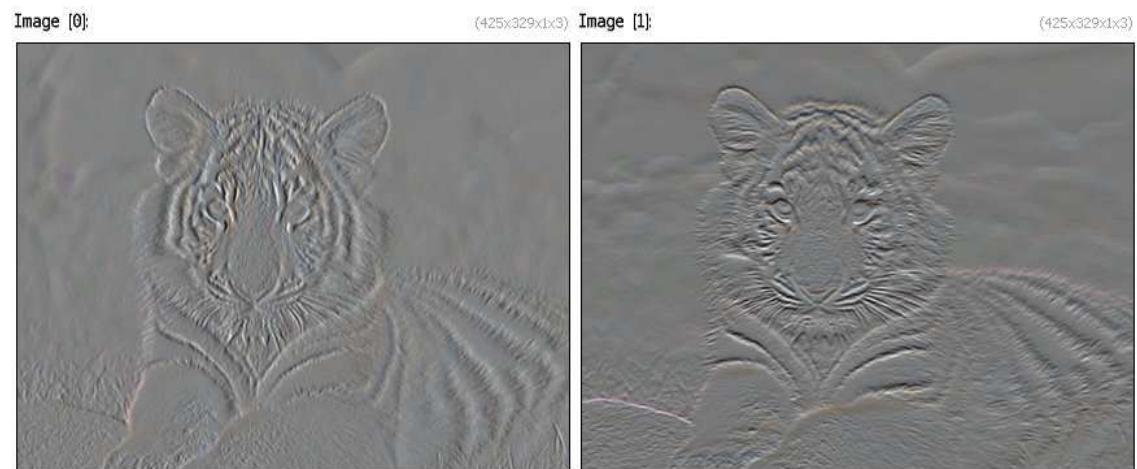
Compute the gradient components (first derivatives) of selected images.

(eq. to '-g').

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

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

**Default value:** ' scheme=3' .



**Example 281 :** `image.jpg -gradient`

### 2.8.41 *-gradient\_orientation*

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

Compute N-d gradient orientation of selected images.

**Default value:** '`dimension=3`' .



**Example 282 :** `image.jpg --gradient_orientation 2`

### 2.8.42 *-gradient\_norm*

Compute gradient norm of selected images.



**Example 283 :** `image.jpg --gradient_norm -equalize[-1] 256`

### 2.8.43 *-haar*

**Arguments:** `scale>0`

Compute the direct haar multiscale wavelet transform of selected images.

### 2.8.44 -heat flow

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

Apply iterations of the heat flow on selected images.

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



**Example 284 :** `image.jpg --heat_flow 20`

### 2.8.45 -hessian (+)

**Arguments:** { `xx` | `xy` | `xz` | `yy` | `yz` | `zz` }..{ `xx` | `xy` | `xz` | `yy` | `yz` | `zz` } | (no args)

Compute the hessian components (second derivatives) of selected images.  
(no args) compute all significant components.



**Example 285 :** `image.jpg -hessian`

**2.8.46 -iee**

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



**Example 286 :** image.jpg -iee

**2.8.47 -ifft (+)**

Compute the inverse fourier transform (real and imaginary parts) of selected images.

**2.8.48 -ihaar**

**Arguments:** scale>0

Compute the inverse haar multiscale wavelet transform of selected images.

**2.8.49 -inn**

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



**Example 287 :** image.jpg -inn

### 2.8.50 -inpaint (+)

**Arguments:** [mask]

Inpaint selected images by specified mask.



**Example 288 :** 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]

### 2.8.51 -inpaint flow

**Arguments:** \_nb\_iter1>=0,\_nb\_iter2>=0,\_dt>=0,\_alpha,\_sigma

Apply iteration of the inpainting flow on selected images.

**Default values:**    'nb\_iter1=4', 'nb\_iter2=15', 'dt=15', 'alpha=1'    and  
                           'sigma=3' .



**Example 289 :** image.jpg 100%,100% -ellipse[-1] 30%,30%,40,30,0,1,255 -reverse  
                           -inpaint\_flow ,

### 2.8.52 *-kuwahara*

**Arguments:** size>0

Apply Kuwahara filter of specified size on selected images.



**Example 290 :** image.jpg --kuwahara 5

### 2.8.53 *-laplacian*

Compute Laplacian of selected images.



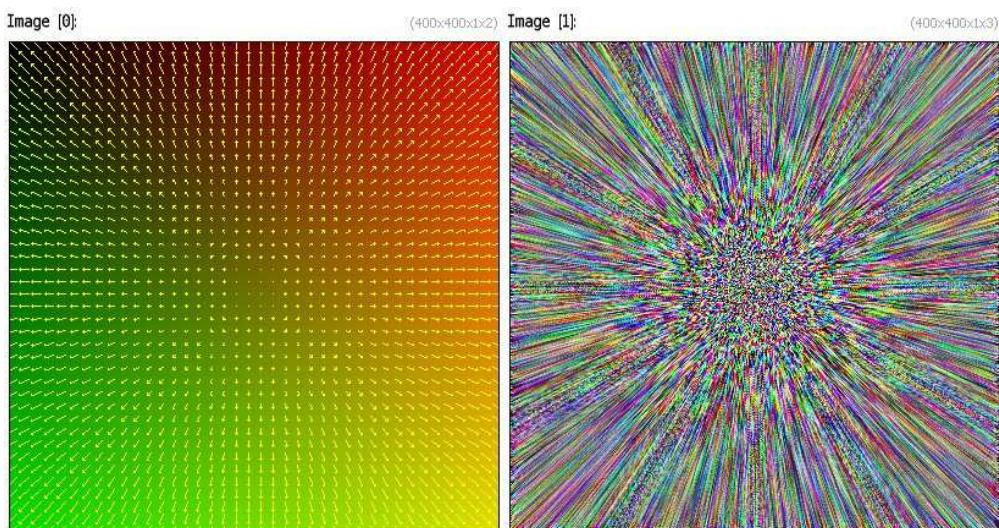
**Example 291 :** image.jpg -laplacian

### 2.8.54 -lic

**Arguments:** `-amplitude>0, -channels>0`

Render LIC representation of selected vector fields.

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



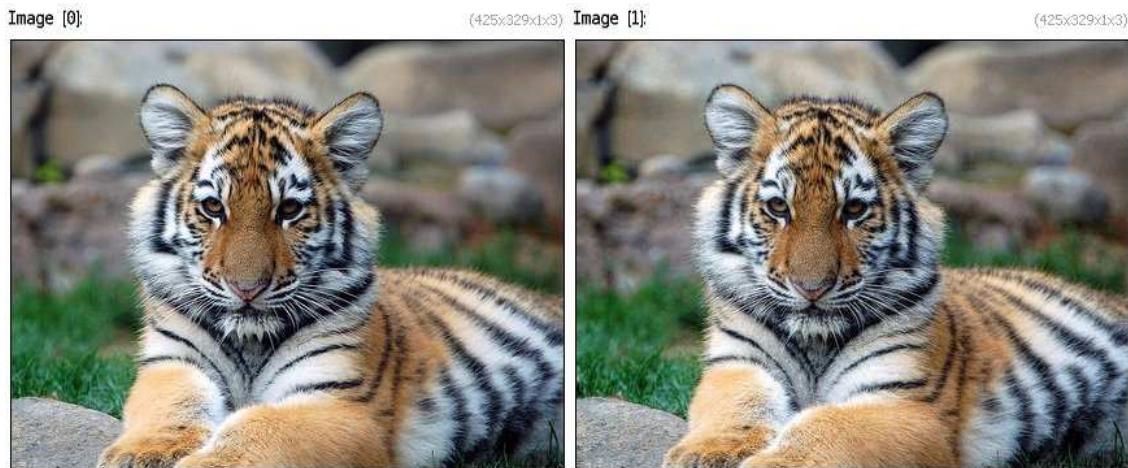
**Example 292 :** `400,400,1,2,'if(c==0,x-w/2,y-h/2)' --lic 200,3 -quiver[-2] [-2],10,-13,1,1,255`

### 2.8.55 *-map\_tones*

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

Apply tone mapping operator on selected images, based on Poisson equation.

**Default values:** `'threshold=0.1', 'gamma=0.8', 'smoothness=0.5'` and `'nb_iter=30'`.



### 2.8.56 *-map\_tones fast*

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

Apply fast tone mapping operator on selected images.

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



**Example 294 :** `image.jpg --map_tones_fast ,`

### 2.8.57 -meancurvature flow

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

Apply iterations of the mean curvature flow on selected images.

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



**Example 295 :** `image.jpg --meancurvature_flow 20`

### 2.8.58 -median (+)

**Arguments:** `radius>=0`

Apply median filter of specified radius on selected images.



**Example 296 :** `image.jpg --median 5`

### 2.8.59 *-normalize\_local*

**Arguments:** `-amplitude>=0, -radius>0, -n_smooth>=0 [%], -a_smooth>=0 [%], -is_cut={0 | 1}, -min=0, -max=255`

Normalize selected images locally.

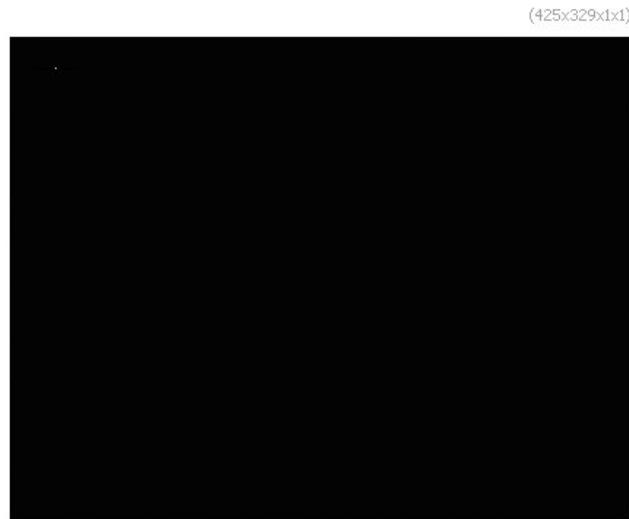
**Default values:** `'amplitude=3', 'radius=16', 'n_smooth=4%', 'a_smooth=2%', 'is_cut=1', 'min=0' and 'max=255'.`



**Example 297 :** `image.jpg --normalize_local 8,10`

### 2.8.60 *-normalized\_cross\_correlation*

Compute normalized cross-correlation using two-by-two selected images.



**Example 298 :** `image.jpg --shift -30,-20 -normalized_cross_correlation`

### 2.8.61 *-phase\_correlation*

Estimate translation vector using two-by-two selected images.



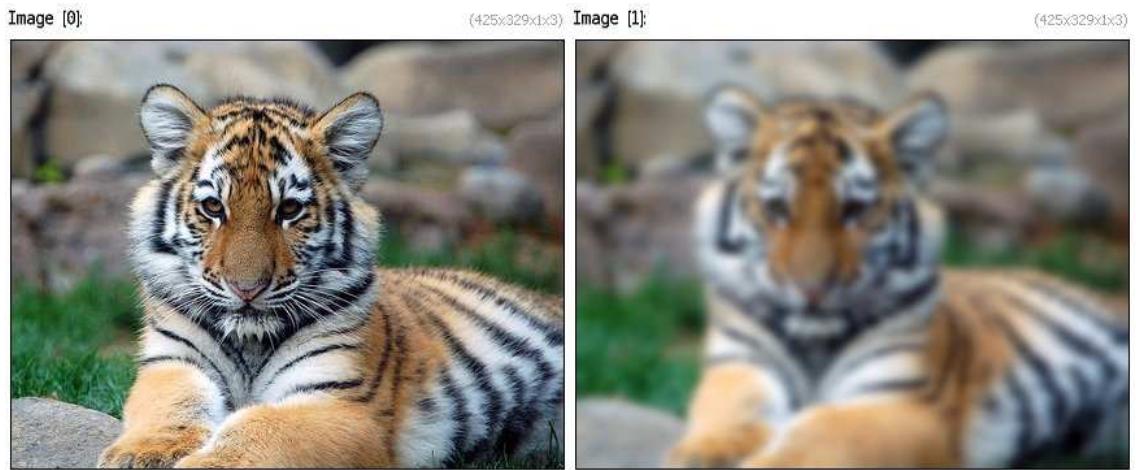
**Example 299 :** `image.jpg --shift -30,-20 --phase.correlation -unroll[-1] y`

### 2.8.62 *-pde\_flow*

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

Apply iterations of a generic PDE flow on selected images.

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



**Example 300:** `image.jpg --pde_flow 20`

### 2.8.63 *-red\_eye*

**Arguments:** `0 <= _threshold <= 100, _smoothness >= 0, 0 <= _attenuation <= 1`

Attenuate red-eye effect in selected images.

**Default values:** '`threshold=75`', '`smoothness=3.5`' and '`attenuation=0.1`'.



**Example 301:** `image.jpg --red_eye ,`

### 2.8.64 *-remove\_hotpixels*

**Arguments:** `_mask_size > 0, _threshold[%] > 0`

Remove hot pixels in selected images.

**Default values:** 'mask\_size=3' and 'threshold=10%'.



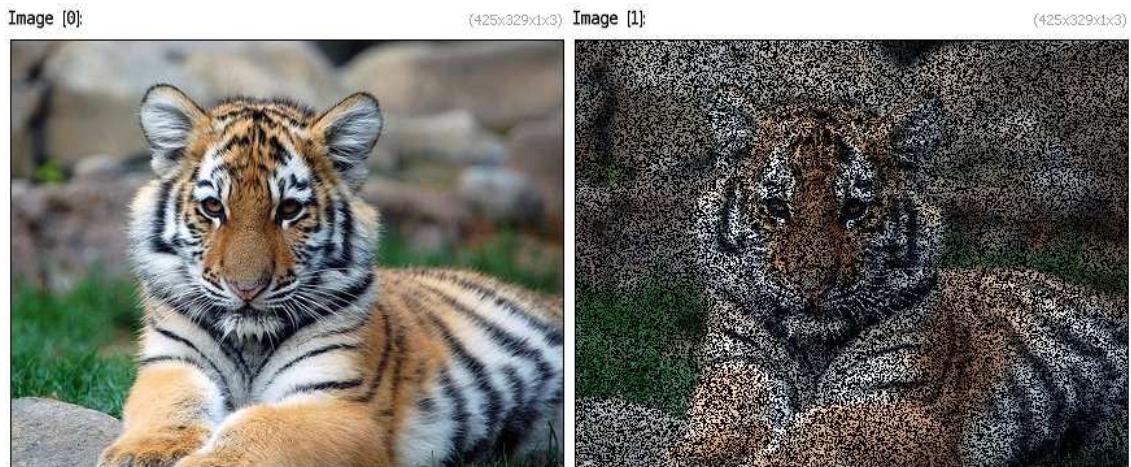
**Example 302 :** `image.jpg -noise 10,2 --remove_hotpixels ,`

### 2.8.65 *-remove\_pixels*

**Arguments:** `density>=0, pixel_sum>=0`

Remove (i.e. set to 0) specified density (in percent) of non-zero pixels to 0. Specified density is regarded against 'pixel\_sum' except if it is set to '0' (in this case, 'pixel\_sum' has default value 'width\*height').

**Default value:** 'density=10', 'pixel\_sum=0'.



**Example 303 :** `image.jpg --remove_pixels 50`

### 2.8.66 *-repair* (+)

**Arguments:** [mask], `_patch_size>=1`, `_lookup_size>=1`, `_lookup_increment>=1`, `_blend_size>=0` | `1` }

Repair masked image pixels in selected images, by patch-based inpainting.  
Beware, this is a very time consuming command.

**Default values:** `'patch_size=11'`, `'lookup_size=22'`, `'lookup_increment=1'`,  
`'blend_size=0'`, `'blend_threshold=0'`, `'blend_decay=0.05'`,  
`'blend_scales=10'` and `'is_blend_outer=0'`.



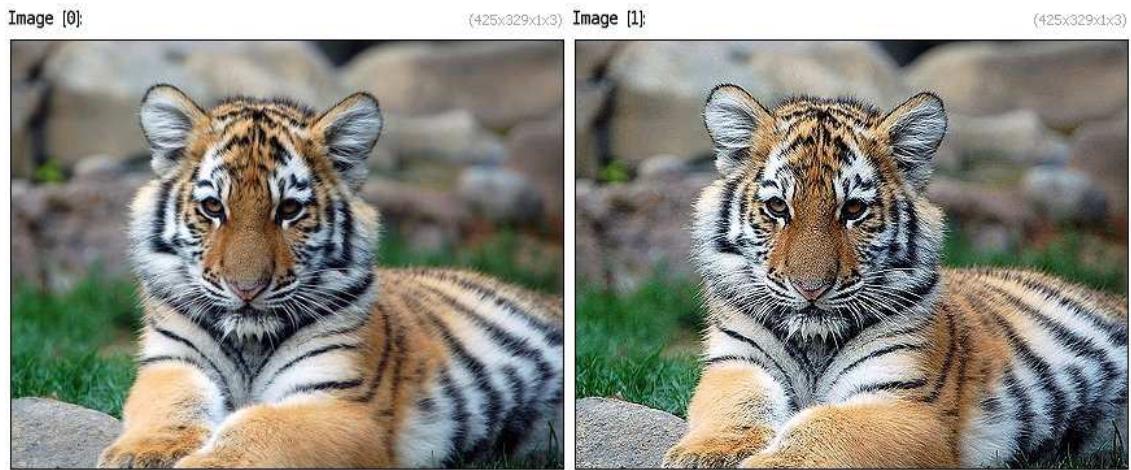
**Example 304 :** `image.jpg 100%,100% -circle 30%,30%,30,1,255,0,255 -circle 70%,70%,50,1,255,0,255 --repair[0] [1],17,24,2,30,0 -rm[1]`

### 2.8.67 *-sharpen* (+)

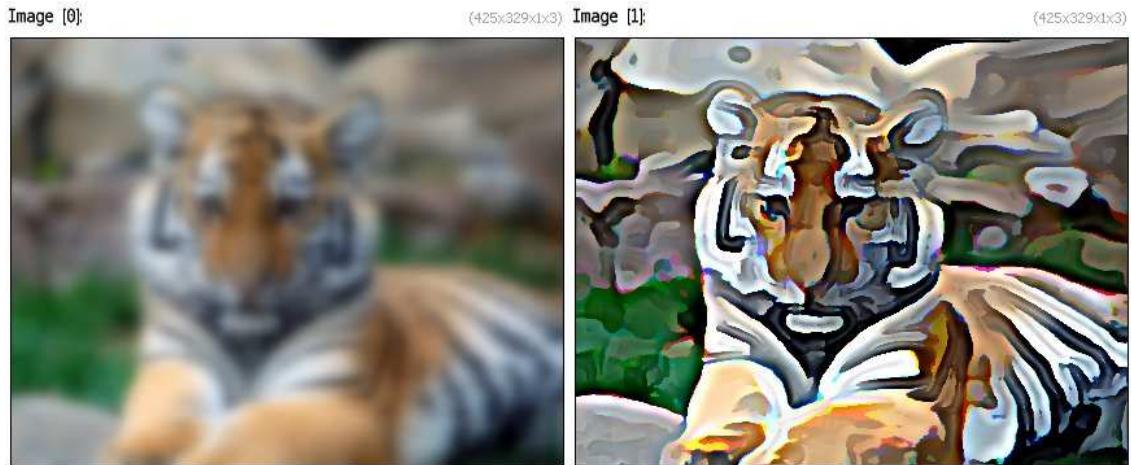
**Arguments:** `amplitude>=0` |  
`amplitude>=0,edge>=0,_alpha,_sigma`

Sharpen selected images by inverse diffusion or shock filters methods.  
'edge' must be specified to enable shock-filter method.

**Default values:** `'alpha=0'` and `'sigma=0'`.



**Example 305 :** image.jpg --sharpen 300



**Example 306 :** image.jpg -blur 5 --sharpen[-1] 300,1

### 2.8.68 -smooth (+)

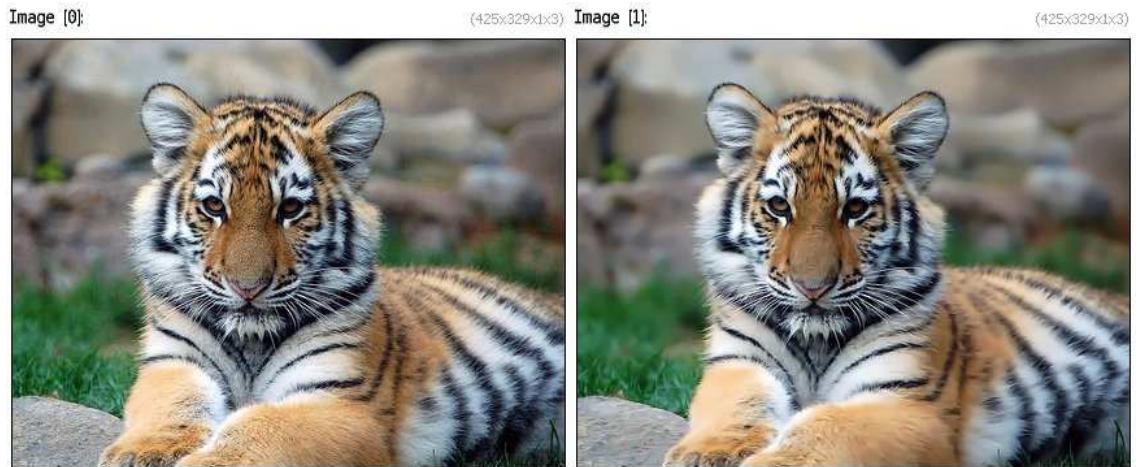
```
Arguments: amplitude>=0,_sharpness>=0,_anisotropy,_alpha,_sigma,_dl>0,_da>0,_precision>0,_i
0 | 1 } |
nb_iterations>=0,_sharpness>=0,_anisotropy,_alpha,_sigma,_dt>0,0 |
[tensor_field],_amplitude>=0,_dl>0,_da>0,_precision>0,_interpolation,_fast_appr
0 | 1 } |
[tensor_field],_nb_iters>=0,_dt>0,0
```

Smooth selected images anisotropically using diffusion PDE's, with specified field of diffusion tensors.

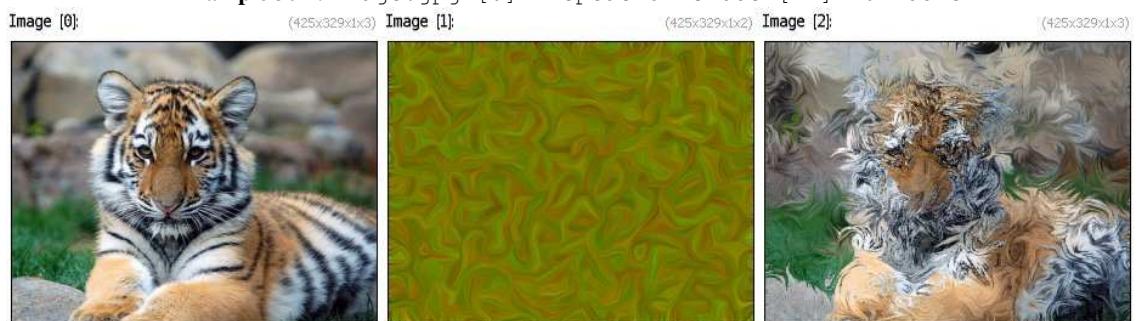
'anisotropy' must be in [0,1].

'interpolation' can be { 0=nearest | 1=linear | 2=runga-kutta }.

**Default values:**            'sharpness=0.7', 'anisotropy=0.3', 'alpha=0.6',  
 'sigma=1.1', 'dl=0.8', 'da=30', 'precision=2', 'interpolation=0'        and  
 'fast\_approx=1'.



**Example 307 :** image.jpg [0] -repeat 3 -smooth[-1] 20 -done

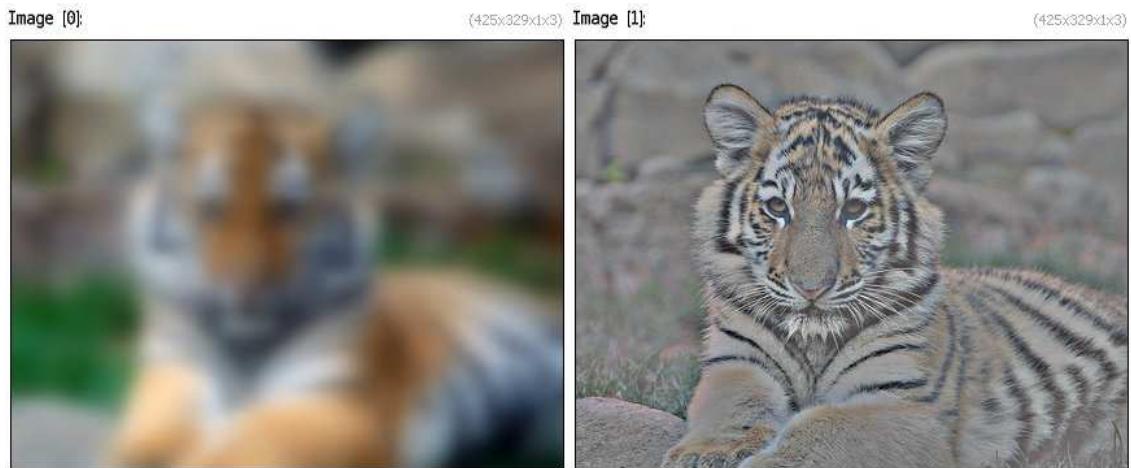


**Example 308 :** image.jpg 100%,100%,1,2 -rand[-1] -100,100 -repeat 2 -smooth[-1] 100,0.2,1,4,4 -done --warp[0] [-1],1,1

### 2.8.69 -split freq

**Arguments:** smoothness>0 [%]

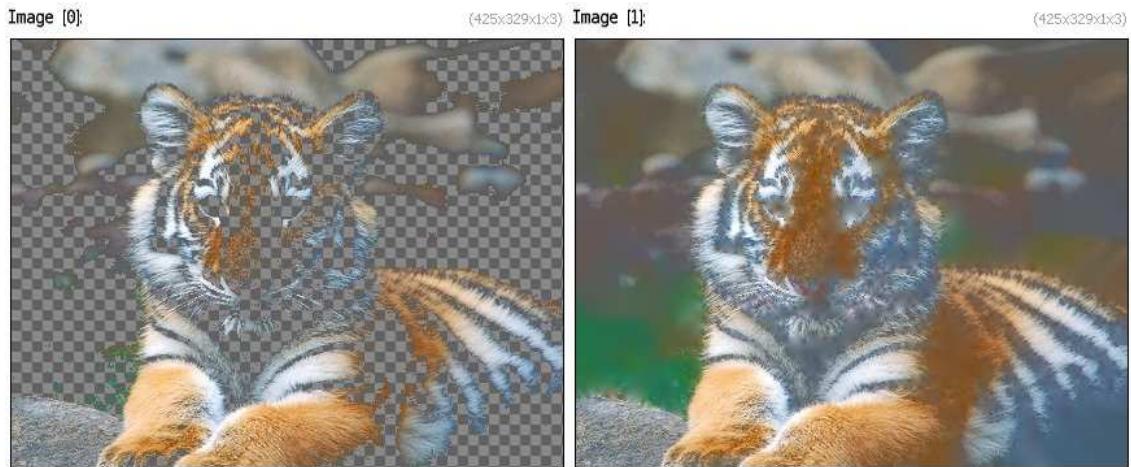
Split selected images into low and high frequency parts.



**Example 309 :** `image.jpg -split-freq 2%`

### 2.8.70 *-solidify*

Replace transparent regions of a RGBA image by morphologically interpolated color.



**Example 310 :** `image.jpg --luminance -ge[-1] 120 -*[-1] 255 -append c --solidify -display-rgba`

### 2.8.71 *-solidify\_linear*

**Arguments:** `_sigma>=1, _dsigma>=1, 0<=_precision<=1`

Replace transparent regions of a RGBA image by linearly interpolated color.

**Default values:** `'sigma=1.5', 'dsigma=1' and 'precision=0.5'`.



```
Example 311: image.jpg --luminance -ge[-1] 120 -*[-1] 255 -append c  
--solidify_linear , -display_rgba
```

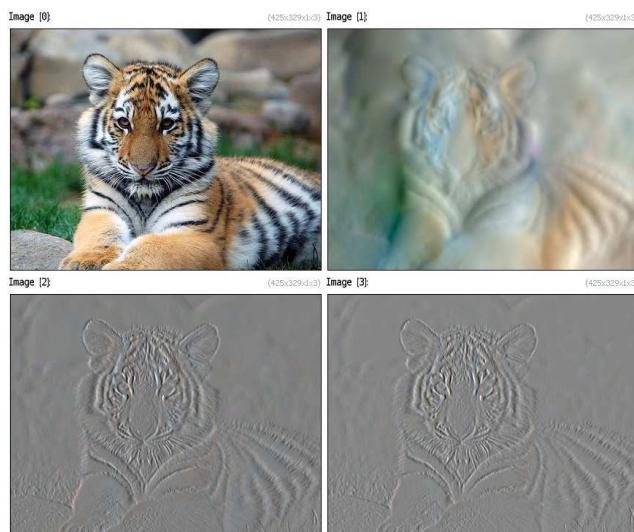
### 2.8.72 *-solve\_poisson*

**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 312: image.jpg -m "foo : -gradient x" --solve_poisson foo --foo[0]  
--laplacian[1]
```

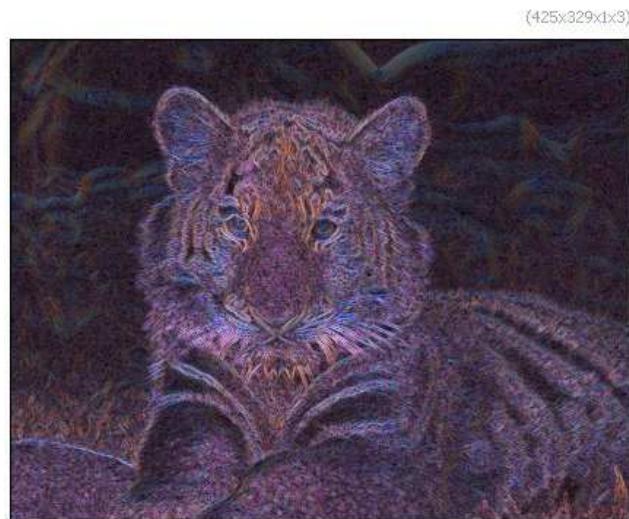
**2.8.73 -structuretensors (+)**

**Arguments:** `_scheme`

Compute the structure tensor field of selected images.

'scheme' can be { 0=centered | 1=forward-backward1 | 2=forward-backward2 }.

**Default value:** '`scheme=2`'.



**Example 313 :** `image.jpg -structuretensors -abs -pow 0.2`

**2.8.74 -tv\_flow**

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

Apply iterations of the total variation flow on selected images.

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



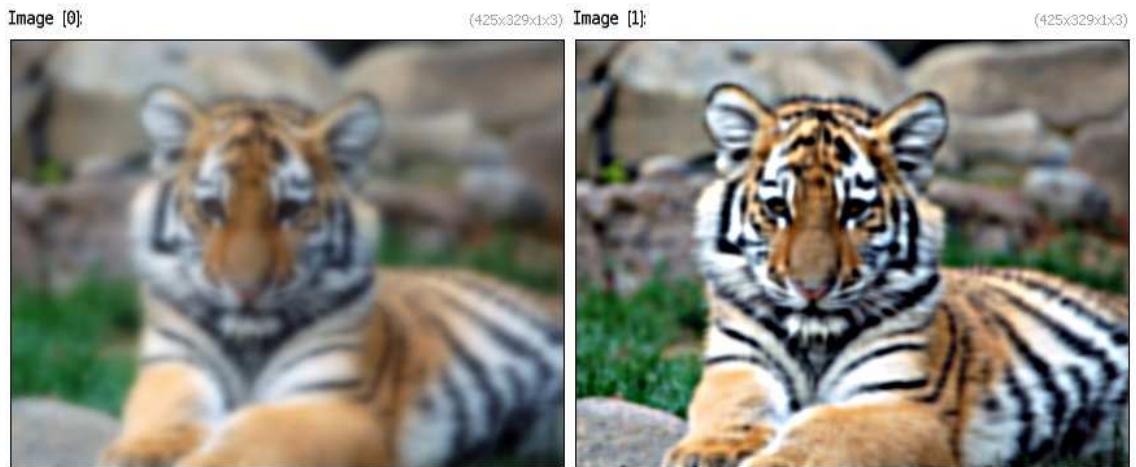
**Example 314:** `image.jpg --tv-flow 40`

### 2.8.75 *-unsharp*

**Arguments:** `radius [%] >=0, -amount >=0, -threshold [%] >=0`

Apply unsharp mask on selected images.

**Default values:** '`amount=2`' and '`threshold=0`'.



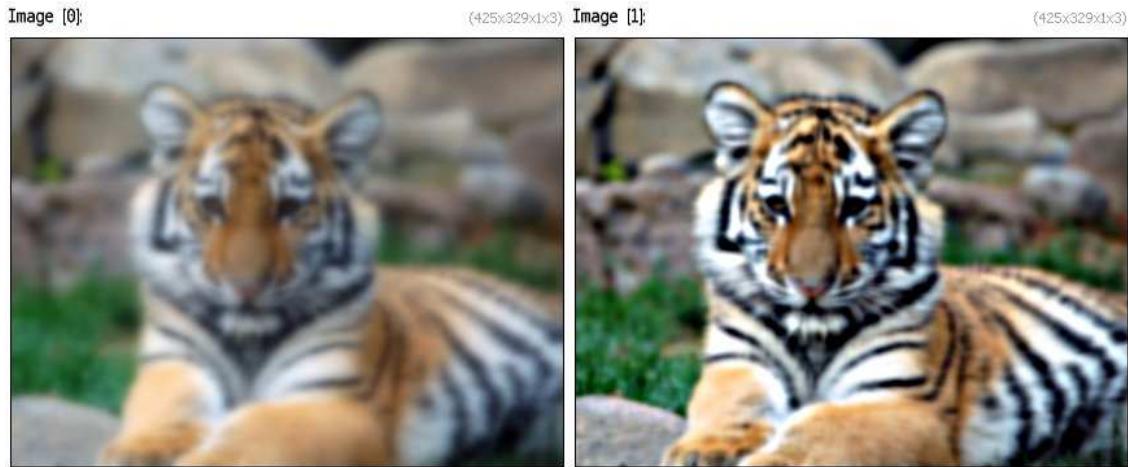
**Example 315:** `image.jpg -blur 3 --unsharp 1.5,15 -cut 0,255`

### 2.8.76 *-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 316:** `image.jpg -blur 3 --unsharp_octave 4,5,15 -cut 0,255`

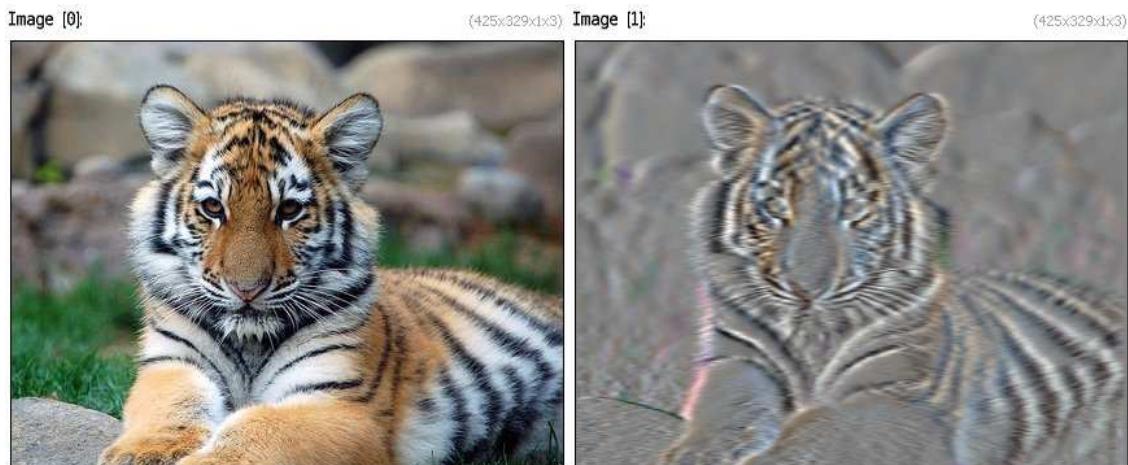
### 2.8.77 *-vanvliet (+)*

**Arguments:** std\_variation>=0[%], order={ 0 | 1 | 2 | 3 }, axis={ x | y | z | c }, boundary

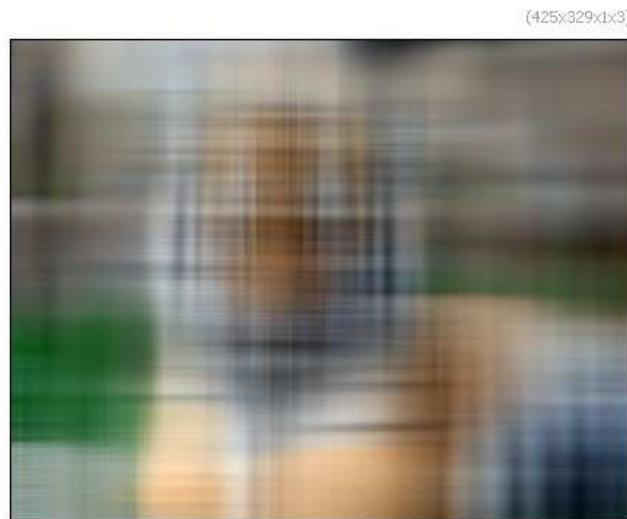
Apply Vanvliet recursive filter with specified standard deviation, order, axis and border conditions on selected images.

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

**Default value:** 'boundary=1'.



**Example 317:** `image.jpg --vanvliet 3,1,x`



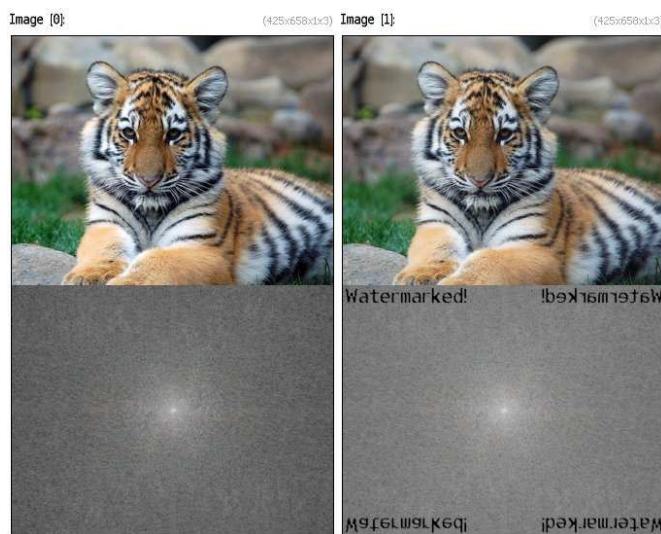
**Example 318 :** `image.jpg --vanvliet 30,0,x -vanvliet [-2] 30,0,y -add`

### 2.8.78 *-watermark\_fourier*

**Arguments:** `text, size>0`

Add a textual watermark in the frequency domain of selected images.

**Default value:** `'size=32'`.



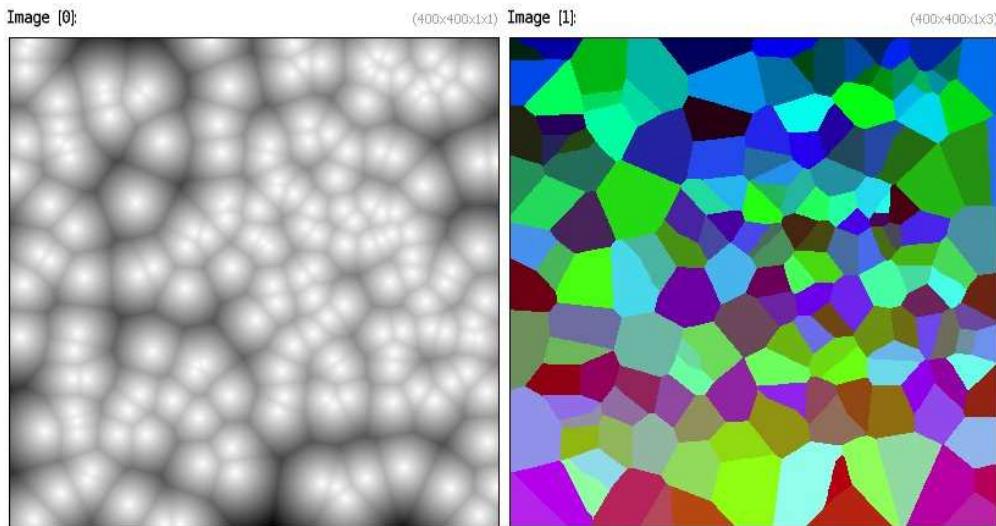
**Example 319 :** `image.jpg --watermark_fourier "Watermarked!" --display_fft -remove[-3,-1] -normalize 0,255 -append[-4,-2] y -append[-2,-1] y`

**2.8.79 -watershed (+)**

**Arguments:** [priority\_image], -fill\_lines={ 0 | 1 }

Compute the watershed transform of selected images.

**Default value:** 'fill\_lines=1'.



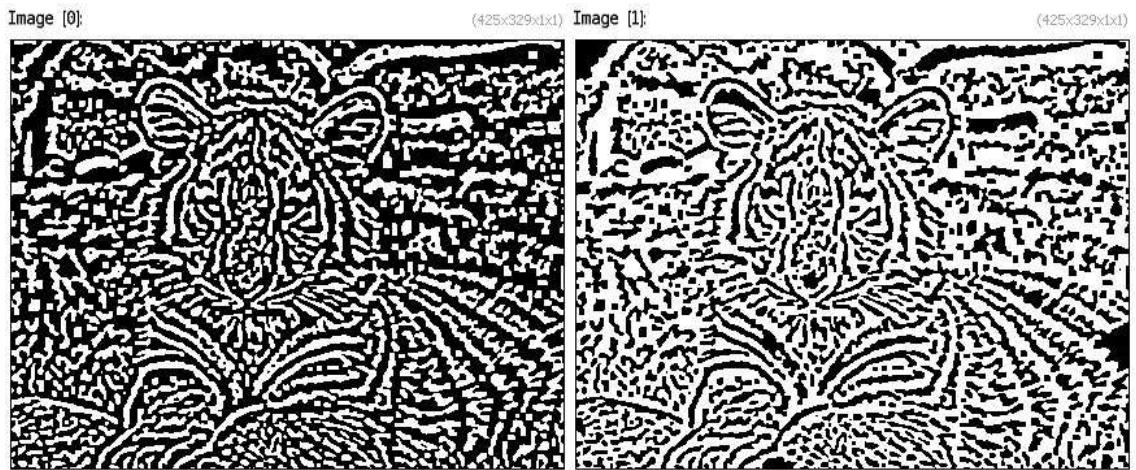
**Example 320 :** 400,400 -noise 0.2,2 --distance 1 -mul[-1] -1 -label[-2] 0  
-watershed[-2] [-1] -mod[-2] 256 -map[-2] 0 -reverse

**2.9 Features extraction****2.9.1 -area**

**Arguments:** tolerance $\geq 0$ , is\_high\_connectivity={ 0 | 1 }

Compute area of connected components in selected images.

**Default values:** 'is\_high\_connectivity=0'.



**Example 321 :** `image.jpg -luminance -stencil[-1] 1 --area 0`

### 2.9.2 *-area fg*

**Arguments:** `tolerance>=0, is_high_connectivity={ 0 | 1 }`

Compute area of connected components for non-zero values in selected images.  
Similar to '-area' except that 0-valued pixels are not considered.

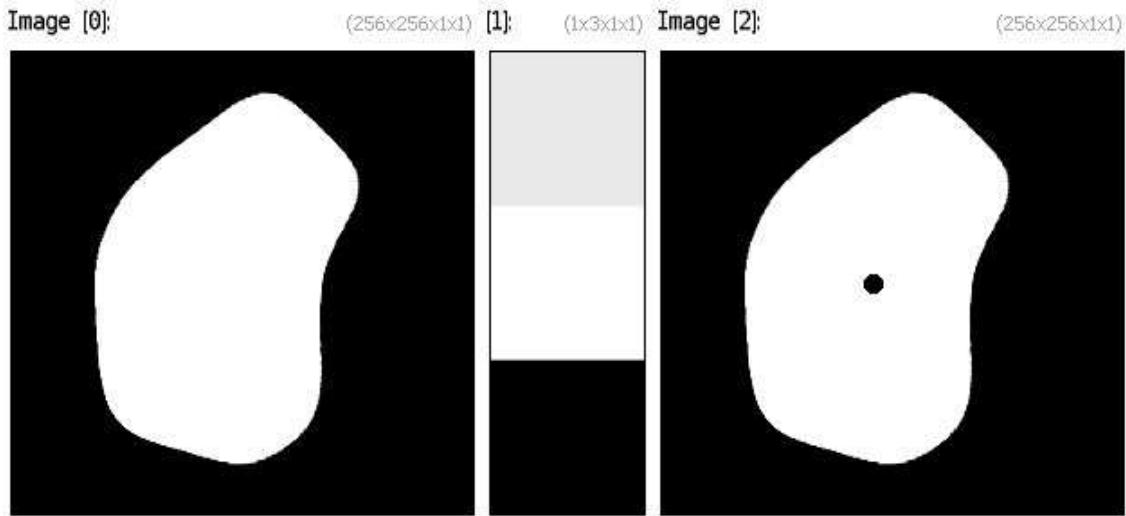
**Default values:** '`is_high_connectivity=0`'.



**Example 322 :** `image.jpg -luminance -stencil[-1] 1 --area_fg 0`

### 2.9.3 *-barycenter*

Compute the barycenter vector of pixel values.



```
Example 323 : 256,256 -ellipse 50%,50%,20%,20%,0,1,1 -deform 20 --barycenter  
--ellipse[-2] @{-1,0,1},5,5,0,10
```

#### 2.9.4 *-displacement (+)*

**Arguments:** [source\_image], \_smoothness, \_precision>=0, \_nb\_scales>=0, iteration\_max>=0, is\_backward 0 | 1 }

Estimate displacement field between specified source and selected images.

If 'smoothness>=0', regularization type is set to isotropic, else to anisotropic.

If 'nbscales==0', the number of needed scales is estimated from the image size.

**Default values:** 'smoothness=0.1', 'precision=5', 'nb\_scales=0', 'iteration\_max=10000' and 'is\_backward=1'.



```
Example 324 : image.jpg --rotate 3,1,0,50%,50%,0.9 --displacement[-1] [-2]  
-quiver[-1] [-1],15,-20,1,1,{1.5*iM}
```

**2.9.5 -distance (+)**

**Arguments:** `isovalue[%],_metric |  
isovalue[%],[custom_metric] |  
x[%]>=0,y[%]>=0,z[%]>=0`

Compute the unsigned distance function to specified isovalue or coordinates.

'metric' can be { 0=chebyshev | 1=manhattan | 2=euclidean | 3=squared-euclidean }.

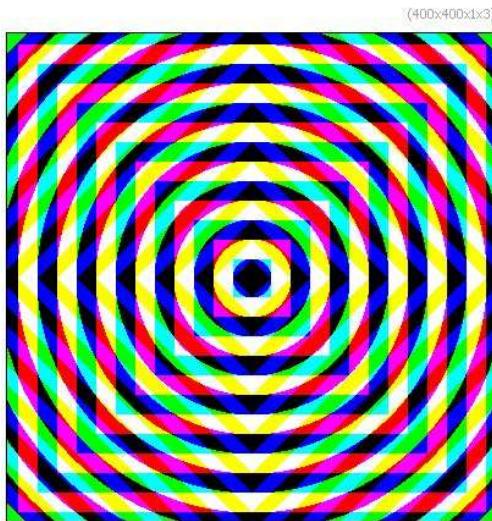
A custom metric for chamfer distances can be specified as a 2d or 3d image.

If arguments 'x','y','z' are provided, the image stands for a potential map used to determine the distance map to specified point (x,y,z).

**Default value:** 'metric=2' .



**Example 325 :** `image.jpg -threshold 20% -distance 0 -pow 0.3`



**Example 326 :** 400,400 -set 1,50%,50% --distance[0] 1,2 --distance[0] 1,1  
-distance[0] 1,0 -mod 32 -threshold 16 -append c



**Example 327 :** image.jpg -luminance -distance 50%,50%,50%

### 2.9.6 -float2fft8

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

### 2.9.7 -ffit82float

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

### 2.9.8 -fftpolar

Compute fourier transform of selected images, as centered magnitude/phase images.



**Example 328 :** `image.jpg -fftpolar -ellipse 50%,50%,10,10,0,1,0 -ifftpolar`

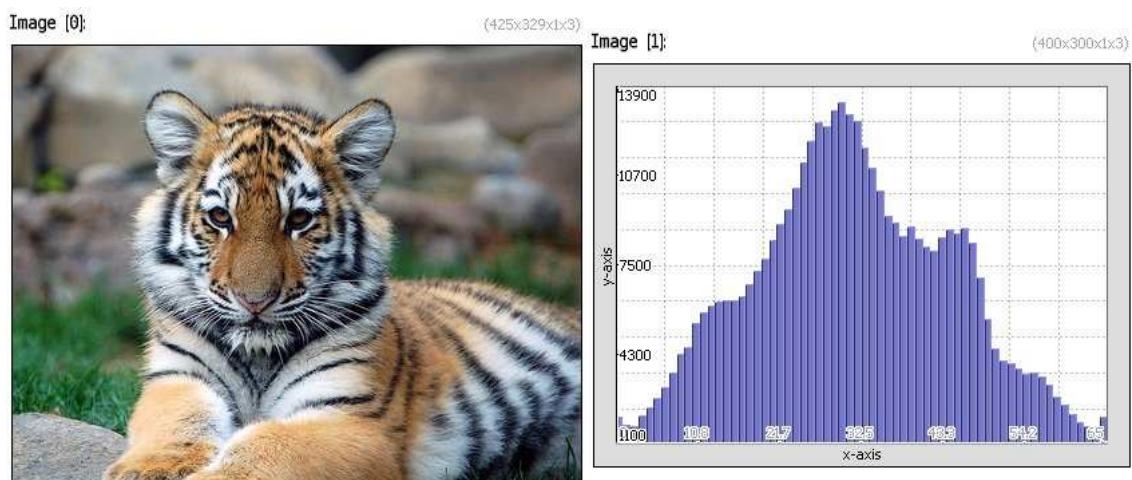
### 2.9.9 *-histogram (+)*

**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:** '`value0=0%`' and '`value1=100%`' .



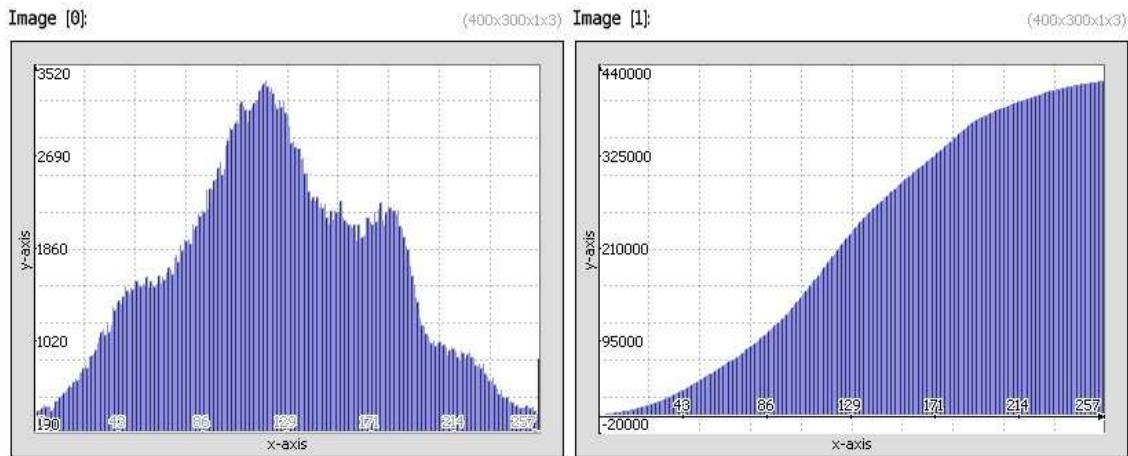
**Example 329 :** `image.jpg --histogram 64 -display_graph[-1] 400,300,3`

### 2.9.10 -histogram\_cumul

**Arguments:** `_nb_levels>0, _is_normalized={ 0 | 1 }, _val0[%], _val1[%]`

Compute cumulative histogram of selected images.

**Default values:** '`nb_levels=256`', '`is_normalized=0`' and '`val0=val1=0`'.



**Example 330 :** `image.jpg --histogram_cumul 256 -histogram[0] 256 -display_graph 400,300,3`

### 2.9.11 -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%`'.

### 2.9.12 -hough

**Arguments:** `_width>0, _height>0, gradient_norm_voting={ 0 | 1 }`

Compute hough transform (theta,rho) of selected images.

**Default values:** '`width=512`', '`height=width`' and '`gradient_norm_voting=1`'.



**Example 331 :** `image.jpg --blur[-1] 1.5 -hough[-1] 400,400 -blur[-1] 0.5 --[-1] 1 -log[-1]`

### 2.9.13 *-ifftpolar*

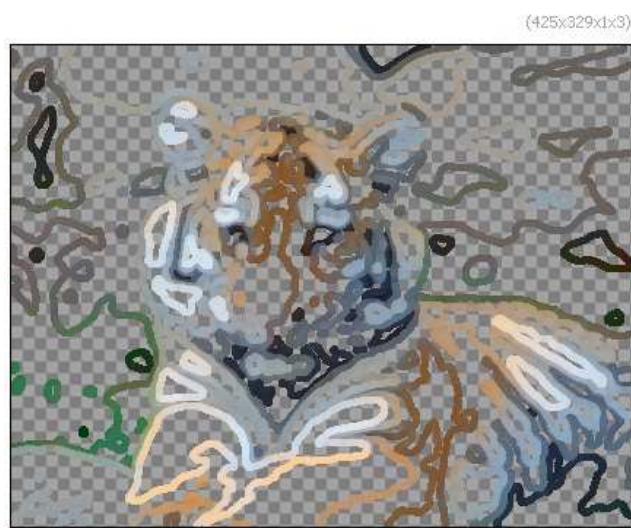
Compute inverse fourier transform of selected images, from centered magnitude/phase images.

### 2.9.14 *-isophotes*

**Arguments:** `_nb_levels>0`

Render isophotes of selected images on a transparent background.

**Default value:** `'nb_levels=64'`



**Example 332 :** `image.jpg -blur 2 -isophotes 6 -dilate_circ 5 -display_rgba`

### 2.9.15 *-label (+)*

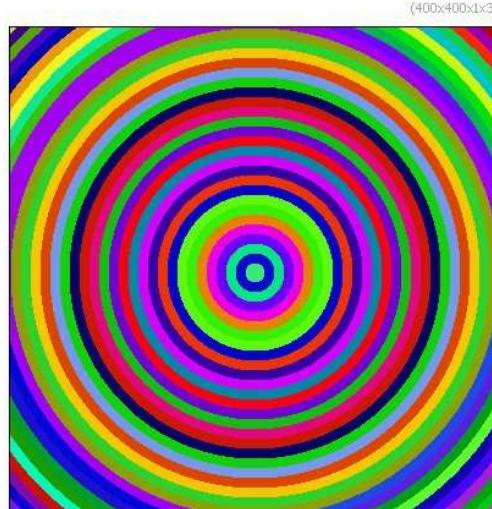
**Arguments:** `tolerance>=0, is_high_connectivity={ 0 | 1 }`

Label connected components in selected images.

**Default values:** '`tolerance=0`' and '`is_high_connectivity=0`'.



**Example 333 :** `image.jpg -luminance -threshold 60% -label 0 -normalize 0,255 -map 0`



**Example 334 :** `400,400 -set 1,50%,50% -distance 1 -mod 16 -threshold 8 -label 0  
-mod 255 -map 2`

**2.9.16 -label\_fg**

**Arguments:** tolerance $\geq 0$ , is\_high\_connectivity={ 0 | 1 }

Label connected components for non-zero values (foreground) in selected images.  
Similar to '-label' except that 0-valued pixels are not labeled.

**Default value:** 'is\_high\_connectivity=0'.

**2.9.17 -max\_patch**

**Arguments:** patch\_size $\geq 1$

Return locations of maximal values in local patch-based neighborhood of given size for selected images.

**Default value:** 'patch\_size=16'.



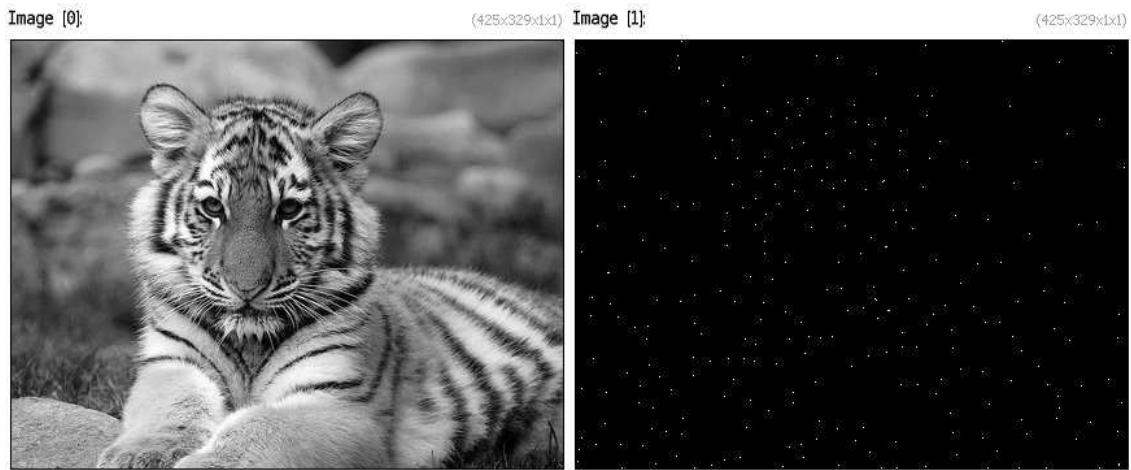
**Example 335 :** image.jpg --norm --max\_patch 16

**2.9.18 -min\_patch**

**Arguments:** 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'.

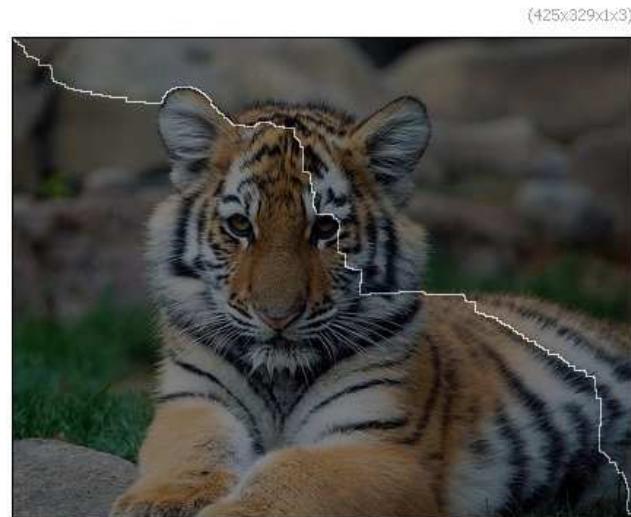


**Example 336:** `image.jpg -norm --min_patch 16`

### 2.9.19 *-minimal\_path*

**Arguments:** `x0 [%]>=0, y0 [%]>=0, z0 [%]>=0, x1 [%]>=0, y1 [%]>=0, z1 [%]>=0`

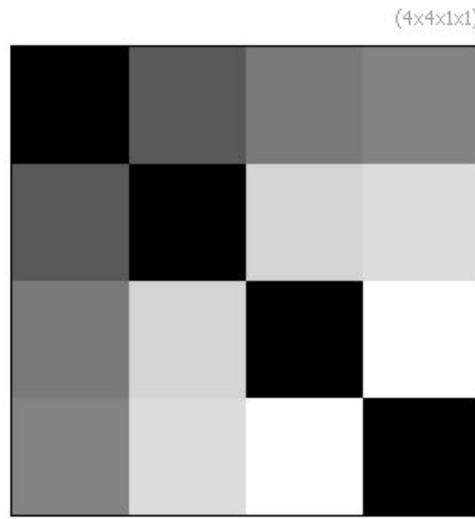
Compute minimal path between two points on selected potential maps.



**Example 337:** `image.jpg --gradient_norm -f[-1] 1/(1+i) -minimal_path[-1] 0,0,0,100%,100%,0 -pointcloud[-1] 0 -*[-1] 280 -to_rgb[-1] -resize[-1] [-2],0 -or`

### 2.9.20 *-mse (\*)*

Compute MSE (Mean-Squared Error) matrix between selected images.

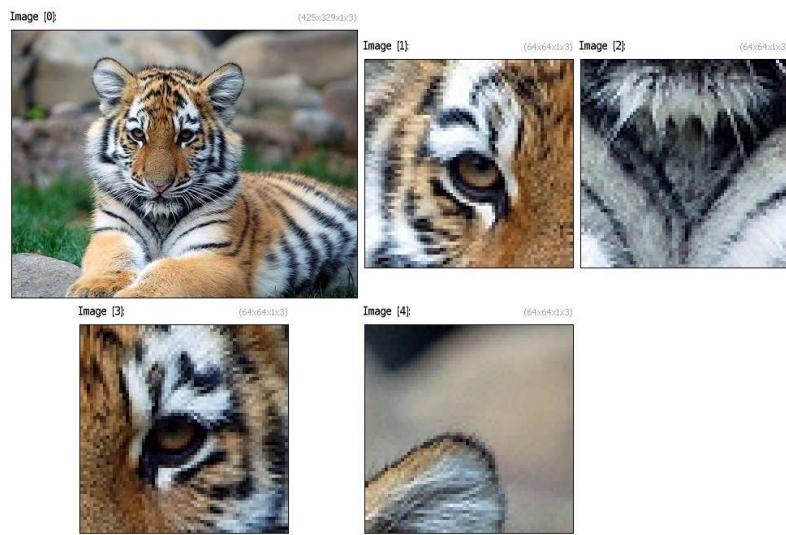


**Example 338:** `image.jpg --noise 30 --noise[0] 35 --noise[0] 38 -cut[-1] 0,255 -mse`

### 2.9.21 *-patches*

**Arguments:** `patch_width>0, patch_height>0, patch_depth>0, x0, y0, z0, -x1, -y1, -z1, ..., -xN`

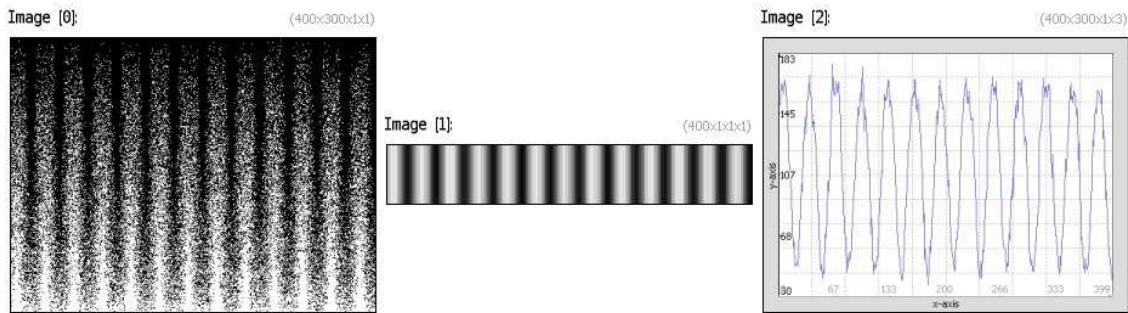
Extract N+1 patches from selected images, centered at specified locations.



**Example 339:** `image.jpg --patches 64,64,1,153,124,0,184,240,0,217,126,0,275,38,0`

### 2.9.22 *-plot2value*

Retrieve values from selected 2d graph plots.



```
Example 340 : 400,300,1,1,'if(y>300*abs(cos(x/10+2*x))),1,0)' --plot2value  
--display_graph[-1] 400,300
```

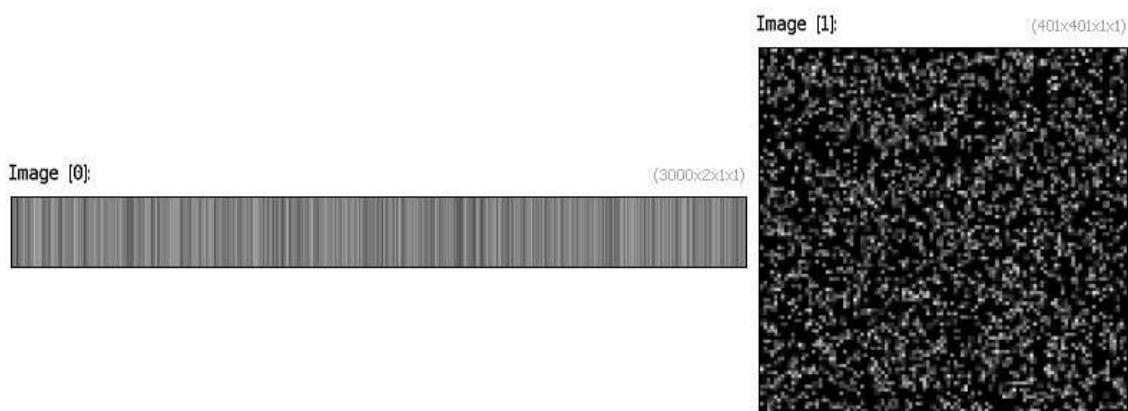
### 2.9.23 -pointcloud

**Arguments:** `-type = { -X=-X-opacity | 0=binary | 1=cumulative | 2=label }`

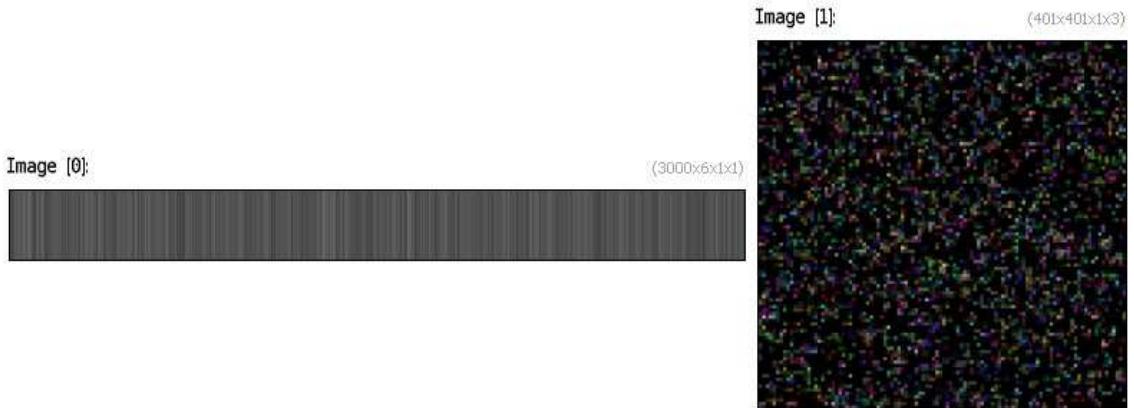
Convert a Nx1, Nx2, Nx3 or NxM image as a point cloud in a 1d/2d or 3d binary image.

If '`M`'>3, the 3-to-M lines sets the (`M`-3)-dimensional color at each point.

**Default value:** '`type=0`' .



```
Example 341 : 3000,2 -rand 0,400 --pointcloud 0 -dilate[-1] 3
```



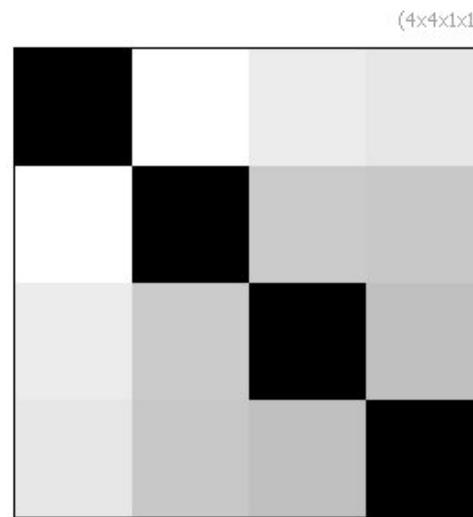
**Example 342 :** 3000,2 -rand 0,400 {w} {w},3 -rand[-1] 0,255 -append y --pointcloud  
0 -dilate[-1] 3

### 2.9.24 *-psnr* (+)

**Arguments:** `_max_value`

Compute PSNR (Peak Signal-to-Noise Ratio) matrix between selected images.

**Default value:** '`max_value=255`'.



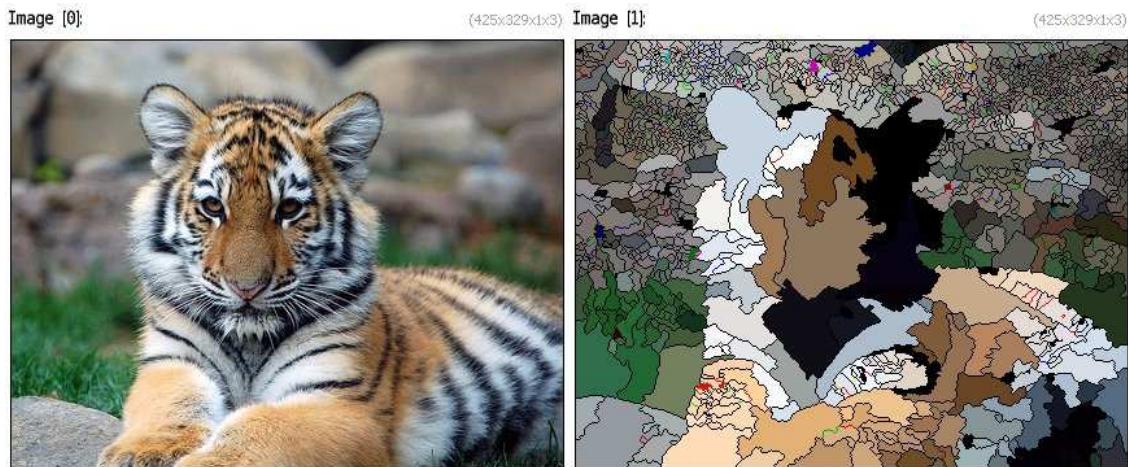
**Example 343 :** image.jpg --noise 30 --noise[0] 35 --noise[0] 38 -cut[-1] 0,255  
-psnr 255 -replace.inf 0

### 2.9.25 *-segment\_watershed*

**Arguments:** `_threshold>=0,_fill_lines={ 0 | 1 }`

Apply watershed segmentation on selected images.

**Default values:** 'threshold=2' and 'fill\_lines=1'.



**Example 344 :** image.jpg --segment\_watershed 2,0

### 2.9.26 -skeleton

**Arguments:** \_smoothness [%] >=0

Compute skeleton of binary shapes using distance transform.

**Default value:** 'smoothness=0'.



**Example 345 :** image.jpg -threshold 50% --skeleton 0

### 2.9.27 -ssd\_patch

**Arguments:** `-use_fourier={ 0 | 1 }, boundary_conditions={ 0=dirichlet | 1=neumann }`

Compute field of SSD between an image and a patch, taken as consecutive selected images.

Argument 'boundary\_conditions' is valid only when 'use\_fourier=0'.

**Default value:** 'use\_fourier=0' and 'boundary\_conditions=0'.



**Example 346 :** `image.jpg --crop 20%,20%,35%,35% --ssd_patch 0,0`

### 2.9.28 -thinning

Compute skeleton of binary shapes using morphological thinning  
(This is a quite slow iterative proces)

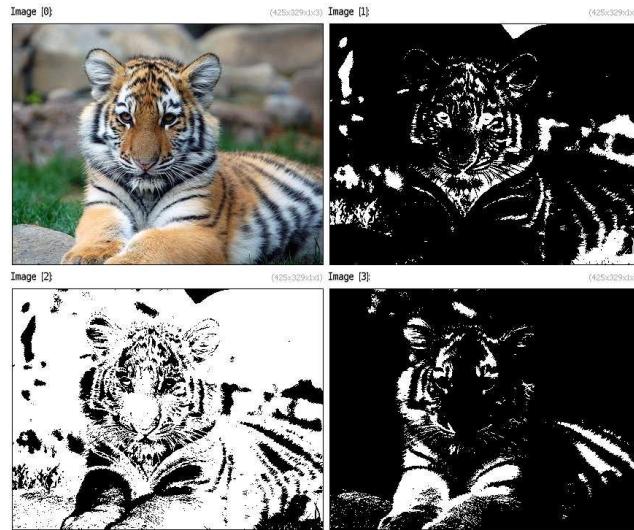


**Example 347 :** `image.jpg -threshold 50% --thinning`

### 2.9.29 *-tones*

**Arguments:**  $N > 0$

Get N tones masks from selected images.



**Example 348 :** image.jpg --tones 3

### 2.9.30 *-topographic\_map*

**Arguments:**  $\_nb\_levels > 0, \_smoothness$

Render selected images as topographic maps.

**Default values:** ' $nb\_levels=16$ ' and ' $smoothness=2$ '.



**Example 349 :** `image.jpg --topographic_map 10`

## 2.10 Image drawing

### 2.10.1 *-axes* (+)

**Arguments:** `x0, x1, y0, y1, font_height>=0, opacity, pattern, color1, ..`

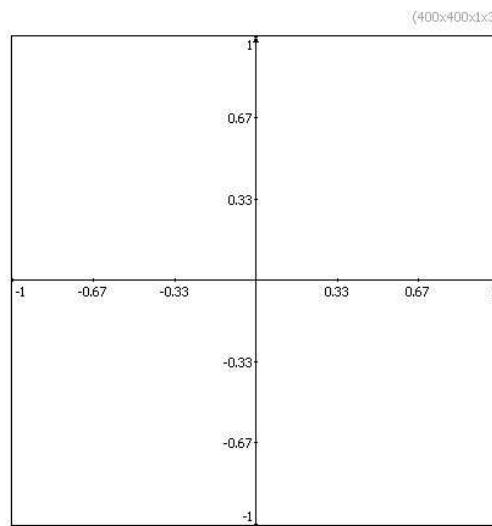
Draw xy-axes on selected images.

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

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

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

**Default values:** '`font_height=13'`, '`opacity=1'`, '`pattern=(undefined)'` and '`color1=0'`.



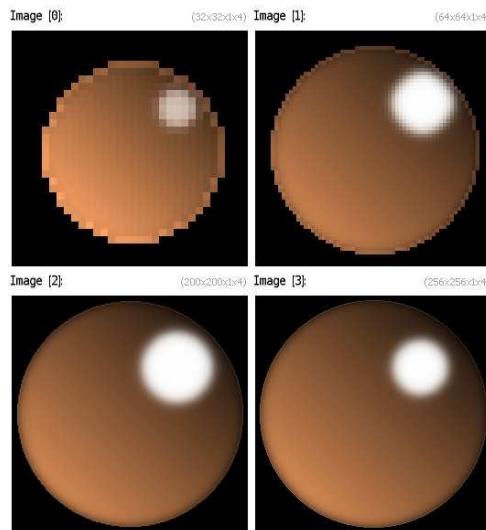
**Example 350 :** `400, 400, 1, 3, 255 -axes -1, 1, 1, -1`

### 2.10.2 *-ball*

**Arguments:** `_R, _G, _B`

Draw a colored RGBA ball sprite on selected images.

**Default values:** '`R=255'`, '`G=R'` and '`B=R'`.



**Example 351 :** 32,32 64,64 200,200 256,256 -ball 255,164,100

### 2.10.3 -chessboard

**Arguments:** `size1>0, -size2>0, -offset1, -offset2, -angle, -opacity, -color1, ..., -color2, ...`

Draw chessboard on selected images.

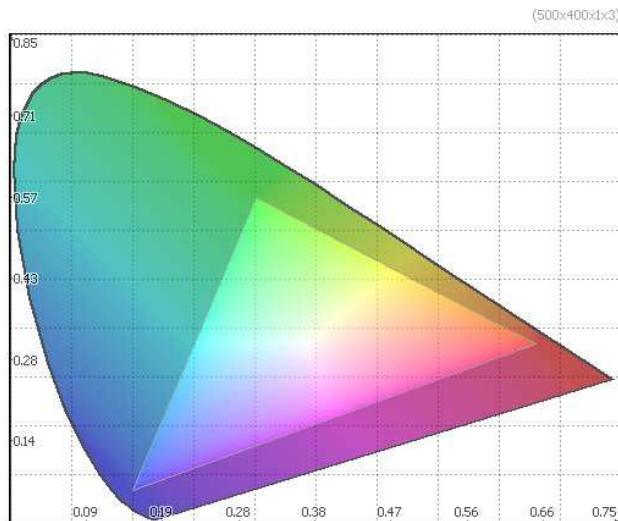
**Default values:** `'size2=size1', 'offset1=offset2=0', 'angle=0', 'opacity=1', 'color1=0' and 'color2=255'.`



**Example 352 :** image.jpg -chessboard 32,32,0,0,25,0.3,255,128,0,0,128,255

### 2.10.4 *-cie1931*

Draw CIE-1931 chromaticity diagram on selected images.



**Example 353 :** 500, 400, 1, 3 -cie1931

### 2.10.5 *-circle*

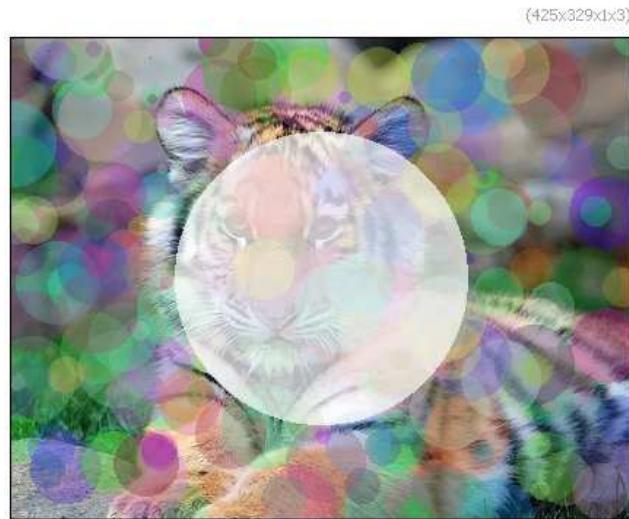
**Arguments:** x[%], y[%], R[%], -opacity, -pattern, -color1, ..

Draw specified colored circle on selected images.

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

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

**Default values:** ' opacity=1', ' pattern=(undefined)' and ' color1=0' .



```
Example 354 : image.jpg -repeat 300 -circle {?(100)}%,{?(100)}%,{?(30)},0.3,@{-RGB}
          -done -circle 50%,50%,100,0.7,255
```

### 2.10.6 -ellipse (+)

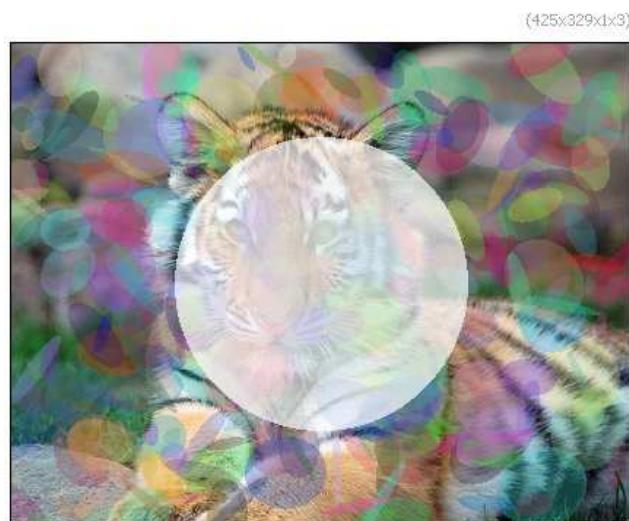
**Arguments:** `x[%],y[%],R[%],r[%],_angle,_opacity,_pattern,_color1,..`

Draw specified colored ellipse on selected images.

A radius of '100%' stands for ' $\sqrt{\text{width}^2 + \text{height}^2}$ '.

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

**Default values:** 'opacity=1', 'pattern=(undefined)' and 'color1=0'.



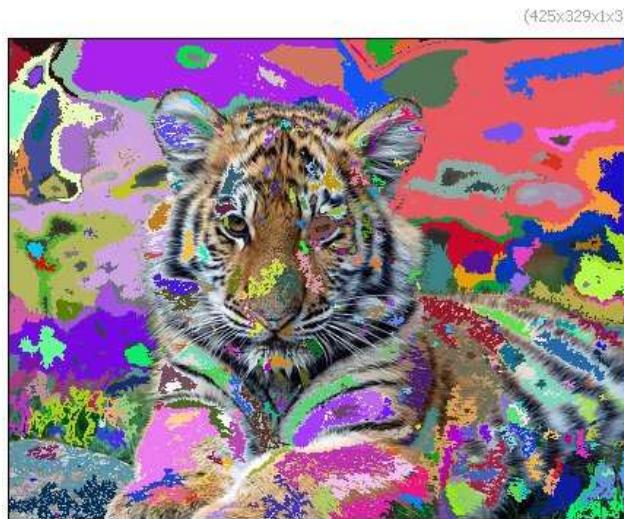
**Example 355 :** `image.jpg -repeat 300 -ellipse  
 {?(100)}%,{?(100)}%,{?(30)},{?(30)},{?(180)},0.3,@{-RGB} -done -ellipse  
 50%,50%,100,100,0,0.7,255`

### 2.10.7 **-flood (+)**

**Arguments:** `x[%], -y[%], -z[%], -tolerance>=0, -is_high_connectivity={ 0 | 1 }, -opacity, -color1, ..`

Flood-fill selected images using specified value and tolerance.

**Default values:** `'y=z=0', 'tolerance=0', 'is_high_connectivity=0', 'opacity=1' and 'color1=0'.`



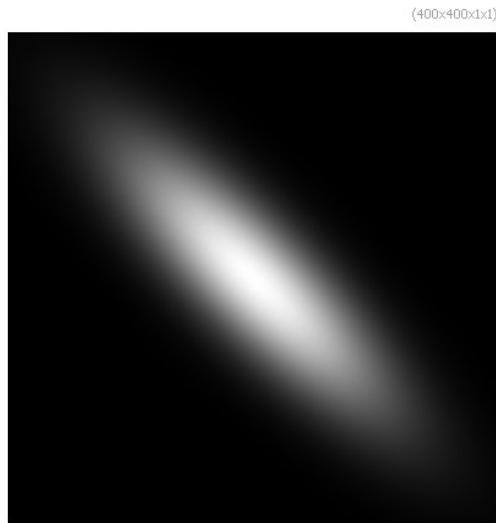
**Example 356 :** `image.jpg -repeat 1000 -flood {?(100)}%,{?(100)}%,0,20,0,1,@{-RGB}  
 -done`

### 2.10.8 **-gaussian**

**Arguments:** `-sigma1[%], -sigma2[%], -angle`

Draw a centered gaussian on selected images, with specified standard deviations and orientation.

**Default values:** `'sigma1=3', 'sigma2=sigma1' and 'angle=0'.`



**Example 357 :** 400,400 -gaussian 100,30,45

### 2.10.9 *-graph* (+)

**Arguments:** [function\_image],  
   'plot\_type',  
   'vertex\_type',  
   'ymin',  
   'ymax',  
   '\_opacity',  
   '\_pattern',  
   '\_color1',  
   '\_formula',  
   '\_resolution>=0',  
   'plot\_type',  
   'vertex\_type',  
   'xmin',  
   'xmax',  
   'ymin',  
   'ymax',  
   '\_opacity',  
   '\_pattern',  
   '\_color1',  
   '\_formula',  
   '\_resolution>=0'

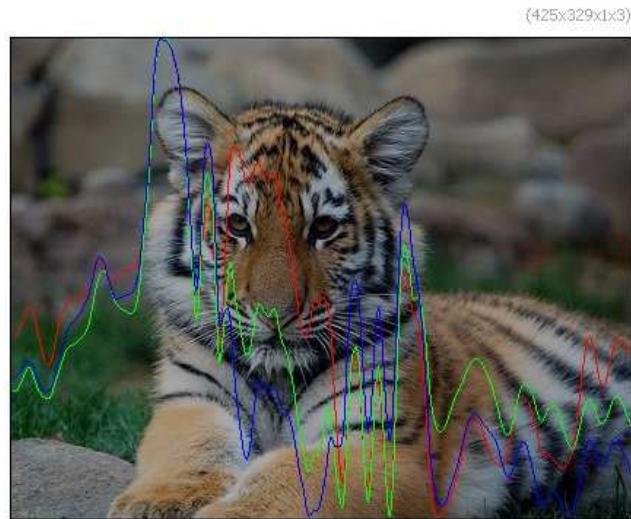
Draw specified function graph on selected images.

'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 358 : image.jpg --rows 50% -blur[-1] 3 -s[-1] c -div[0] 1.5 -graph[0]
[1],2,0,0,0,1,255,0,0 -graph[0] [2],2,0,0,0,1,0,255,0 -graph[0]
[3],2,0,0,0,1,0,0,255 -keep[0]
```

### 2.10.10 *-grid*

**Arguments:** `size_x[%]>=0, size_y[%]>=0, offset_x[%], offset_y[%], opacity, pattern, color1`

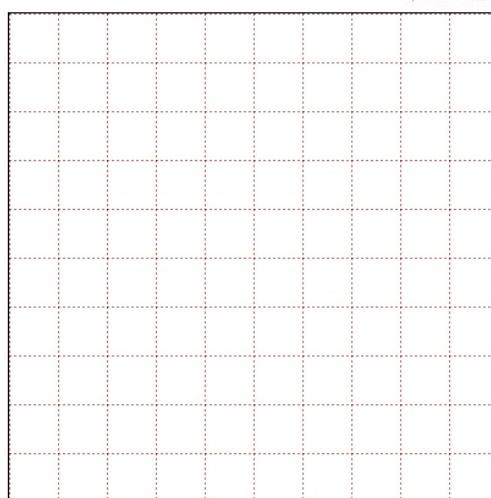
Draw xy-grid on selected images.

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

**Default values:** '`offset_x=offset_y=0'`, '`opacity=1'`, '`pattern=(undefined)`' and '`color1=0'`.



**Example 359:** `image.jpg -grid 10%,10%,0,0,0.5,255`  
 (400x400x1x3)



**Example 360:** `400,400,1,3,255 -grid 10%,10%,0,0,0.3,0xCCCCCCCC,128,32,16`

### 2.10.11 *-image (+)*

**Arguments:** `[sprite], -x[%], -y[%], -z[%], -c[%], -opacity, -[sprite_mask], -max_opacity_mask`

Draw specified sprite image on selected images.  
*(eq. to '-j').*

**Default values:**   `'x=y=z=c=0', 'opacity=1', 'sprite_mask=(undefined)' and  
 'max_opacity_mask=1'.`



**Example 361 :** `image.jpg --crop 40%,40%,60%,60% -resize[-1] 200%,200%,1,3,5 -frame[-1] 2,2,0 -image[0] [-1],30%,30% -keep[0]`

### 2.10.12 *-line (+)*

**Arguments:** `x0[%],y0[%],x1[%],y1[%],_opacity,_pattern,_color1,..`

Draw specified colored line on selected images.

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

**Default values:** '`opacity=1'`, '`pattern=(undefined)'` and '`color1=0'`.



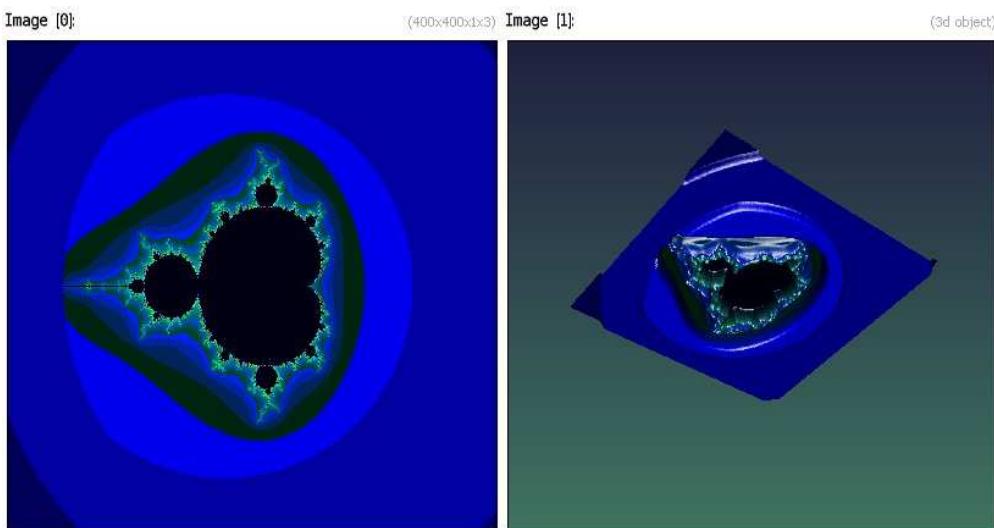
```
Example 362: image.jpg -repeat 500 -line 50%,50%,{?(w)},{?(h)},0.5,@{-RGB} -done
-line 0,0,100%,100%,1,0xCCCCCCCC,255 -line 100%,0,0,100%,1,0xCCCCCCCC,255
```

### 2.10.13 **-mandelbrot (+)**

**Arguments:** `z0r,z0i,z1r,z1i,_iteration_max>=0,_is_julia={ 0 | 1 },_c0r,_c0i,_opacity`

Draw mandelbrot/julia fractal on selected images.

**Default values:** `'iteration_max=100', 'is_julia=0', 'c0r=c0i=0' and 'opacity=1'.`



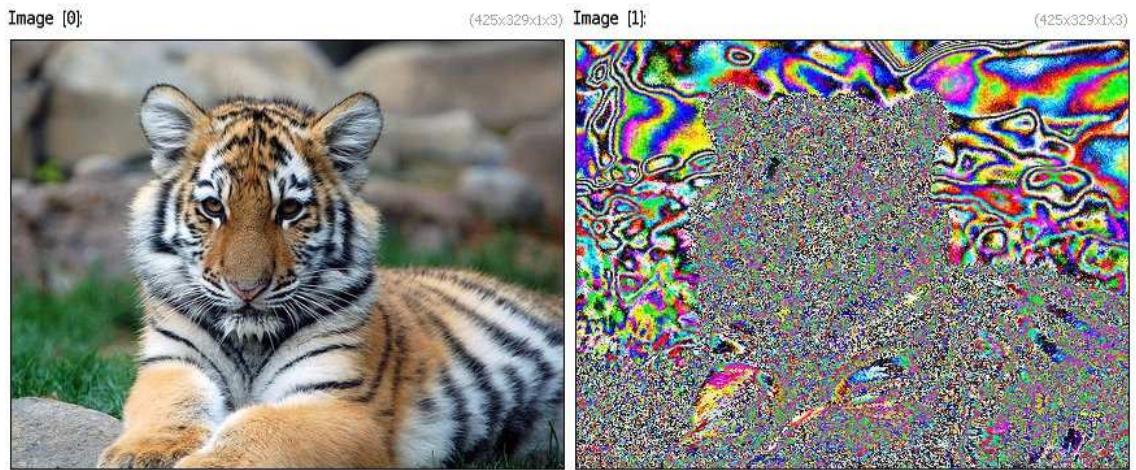
```
Example 363: 400,400 -mandelbrot -2.5,-2,2,2,1024 -map 0 --blur 2
-elevation3d[-1] -0.2
```

### 2.10.14 **-marble**

**Arguments:** `_image_weight,_pattern_weight,_angle,_amplitude,_sharpness>=0,_anisotropy>=0,_alpha>=0,_sigma>=0,_cut_low=_cut_high=0`

Render marble like pattern on selected images.

**Default values:** `'image_weight=0.2', 'pattern_weight=0.1', 'angle=45', 'amplitude=0', 'sharpness=0.4', 'anisotropy=0.8', 'alpha=0.6', 'sigma=1.1' and 'cut_low=cut_high=0'.`

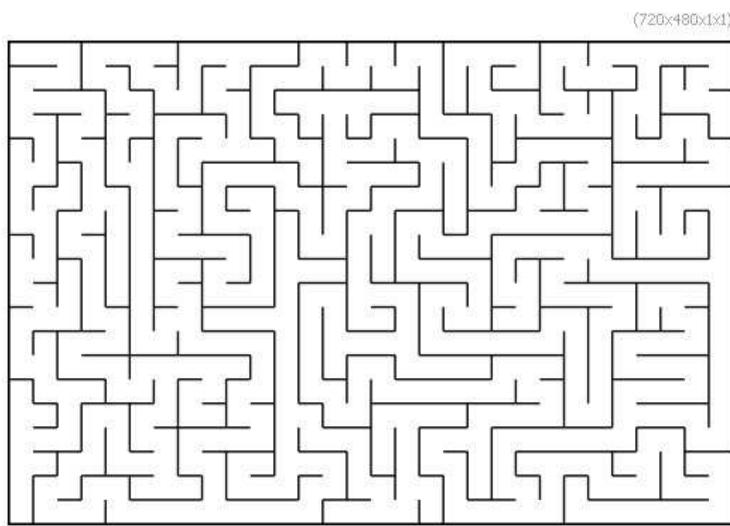


**Example 364:** `image.jpg --marble ,`

### 2.10.15 *-maze*

**Arguments:** `_width>0, _height>0, _cell_size>0`

Input maze with specified size.

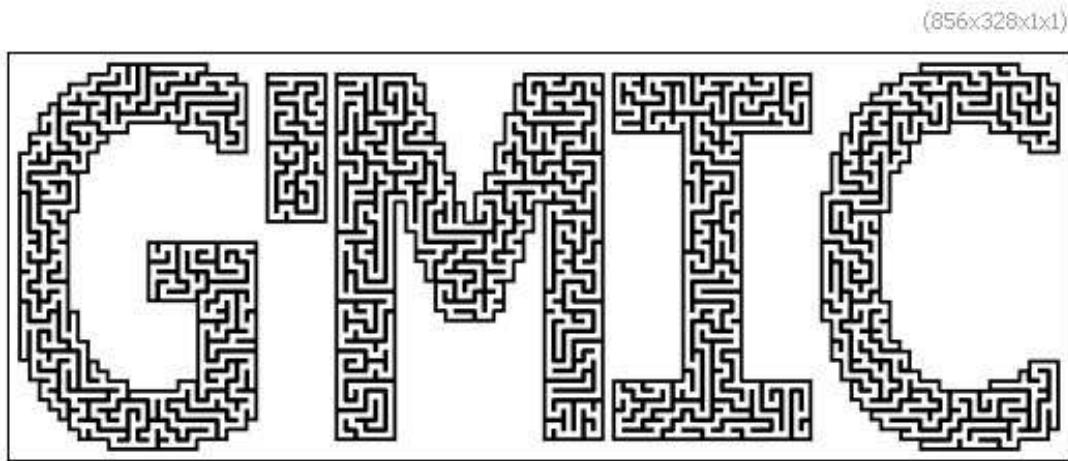


**Example 365 :** `-maze 30,20 -negative -n 0,255`

### 2.10.16 *-maze mask*

**Arguments:** `_cellsize>0`

Input maze according to size and shape of selected mask images.  
Mask may contain disconnected shapes.



```
Example 366 : 0 -text "G'MIC",0,0,57,1,1 -dilate 3 -autocrop 0 -frame 1,1,0  
-maze_mask 8 -dilate 3 -negative -* 255
```

### 2.10.17 *-object3d* (+)

**Arguments:** [*object3d*], *-x[%]*, *-y[%]*, *-z*, *-opacity*, *-rendering\_mode*, *-is\_double3d*=  
0 | 1, *-is\_zbuffer*= { 0 | 1 }, *-focale*, *-light\_x*, *-light\_y*, *-light\_z*, *-specular\_lightness*, *-spec*

Draw specified 3d object on selected images.

'*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 367 : image.jpg -torus3d 100,10 -cone3d 30,-120 -add3d[-2,-1]
-rotate3d[-1] 1,1,0,60 -object3d[0] [-1],50%,50% -keep[0]
```

### 2.10.18 *-pack\_sprites*

**Arguments:**            `_nb_scales>=0, 0<=_min_scale<=100, _allow_rotation={ 0=0 | 1=180 | 2=90 | 3=any }, _spacing, _precision>=0, ,max_iterations>=0`

Try to randomly pack as much sprites as possible onto the 'empty' areas of an image.

Sprites can be eventually rotated and scaled during the packing process.

First selected image is the canvas that will be filled with the sprites.

Its last channel must be a binary mask whose zero values represent potential locations for drawing the sprites.

All other selected images represent the sprites considered for packing.

Their last channel must be a binary mask that represents the sprite shape (i.e. a 8-connected component).

The order of sprite packing follows the order of specified sprites in the image list.

Sprite packing is done on random locations and iteratively with decreasing scales.

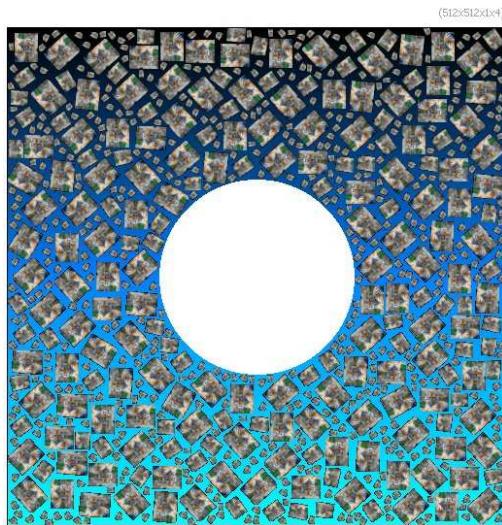
'nb\_scales' sets the number of decreasing scales considered for all specified sprites to be packed.

'min\_scale' (in %) sets the minimal size considered for packing (specified as a percentage of the original sprite size).

'spacing' can be positive or negative.

'precision' tells about the desired number of failed trials before ending the filling process.

**Default values:**            `'nb_scales=5', 'min_scale=25', 'allow_rotation=3', 'spacing=1', 'precision=7' and 'max_iterations=256'.`

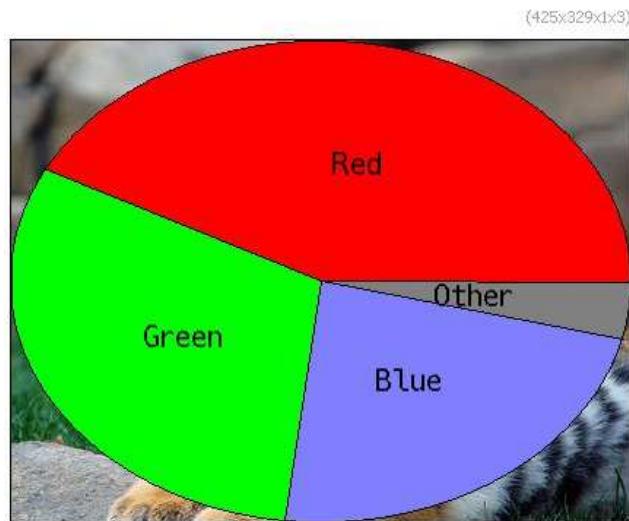


```
Example 368 : 512,512,1,3,"min(255,y*c/2)" 100%,100% -circle 50%,50%,100,1,255  
-append c image.jpg -resize2dy[-1] 24 -to_rgba -pack_sprites 3,25
```

### 2.10.19 -piechart

**Arguments:** label\_height $\geq$ 0, label\_R, label\_G, label\_B, "label1", value1, R1, G1, B1, ..., "labelN", v

Draw pie chart on selected (RGB) images.



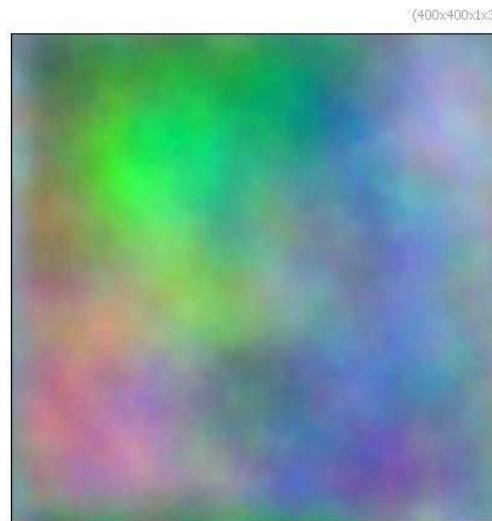
```
Example 369 : image.jpg -piechart  
25,0,0,0,"Red",55,255,0,0,"Green",40,0,255,0,"Blue",30,128,128,255,"Other",5,128,128,128
```

**2.10.20 *-plasma* (+)**

**Arguments:** `alpha, -beta, -scale >= 0`

Draw a random colored plasma on selected images.

**Default values:** '`beta=1`' and '`scale=8`' .



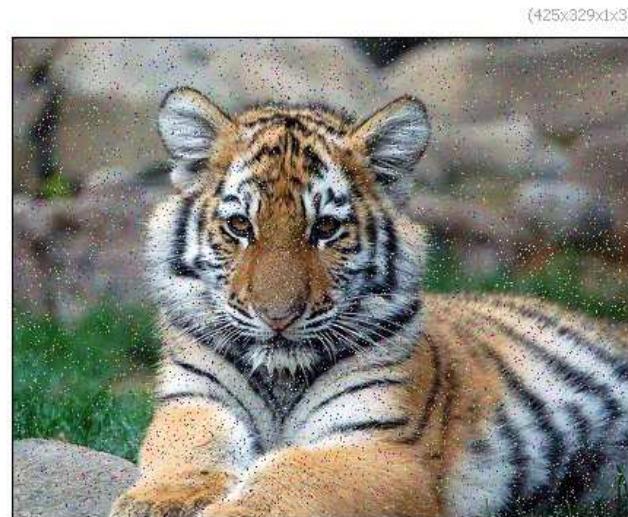
**Example 370 :** `400, 400, 1, 3 -plasma 1`

**2.10.21 *-point* (+)**

**Arguments:** `x[%], y[%], -z[%], -opacity, -color1, ..`

Set specified colored pixel on selected images.

**Default values:** '`z=0`', '`opacity=1`' and '`color1=0`' .



**Example 371 :** image.jpg -repeat 10000 -point {(100)%,{(100)%,0,1,@{-RGB}} -done

### 2.10.22 *-polka\_dots*

**Arguments:** diameter>=0, -density, -offset1, -offset2, -angle, -aliasing, -shading, -opacity, -color

Draw dots pattern on selected images.

**Default values:** 'density=20', 'offset1=offset2=50', 'angle=0',  
'aliasing=10', 'shading=1', 'opacity=1' and 'color=255'.



**Example 372 :** image.jpg -polka\_dots 10,15,0,0,20,10,1,0.5,0,128,255

### 2.10.23 *-polygon (+)*

**Arguments:** `N>=1, x1[%], y1[%], ..., xN[%], yN[%], _opacity, _pattern, _color1, ..`

Draw specified colored N-vertices polygon on selected images.

'pattern' is 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 373 :** `image.jpg -polygon 4,20%,20%,80%,30%,80%,70%,20%,80%,0.3,0,255,0 -polygon 4,20%,20%,80%,30%,80%,70%,20%,80%,1,0xCCCCCCCC,255`



**Example 374 :** `image.jpg 2,16,1,1,'?(if(x,@{-1,h},@{-1,w}))' -polygon[-2] {h},@-1,0.6,255,0,255 -remove[-1]`

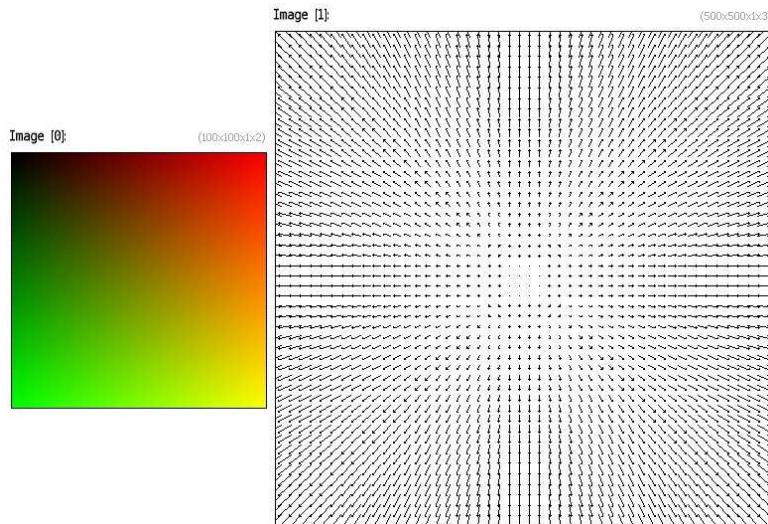
### 2.10.24 -quiver (+)

**Arguments:** [function\_image], \_sampling>0, \_factor, \_is\_arrow={ 0 | 1 }, \_opacity, \_pattern, \_color1, ..

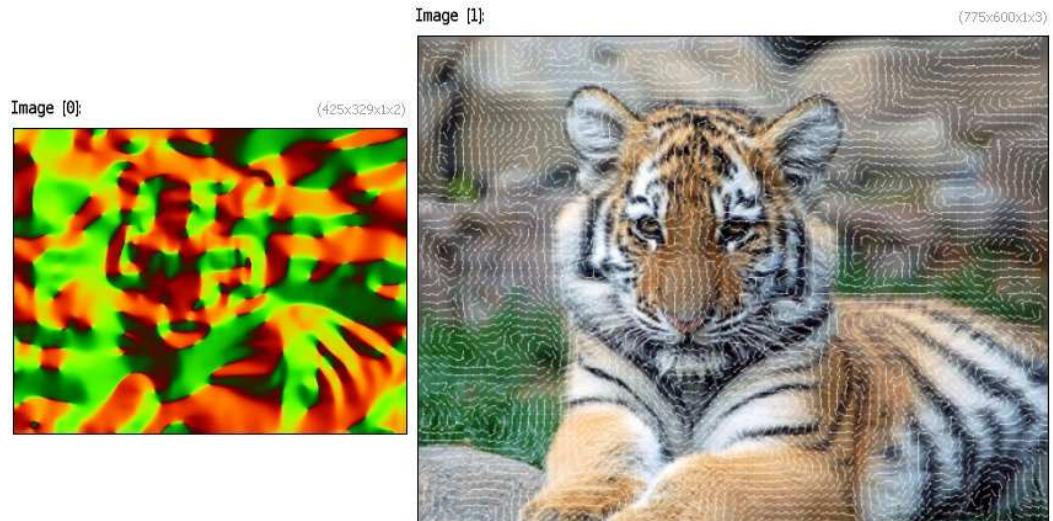
Draw specified 2d vector/orientation field on selected images.

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

**Default values:** 'sampling=25', 'factor=-20', 'is\_arrow=1', 'opacity=1', 'pattern=(undefined)' and 'color1=0'.



**Example 375 :** 100,100,1,2,'if(c==0,x-w/2,y-h/2)' 500,500,1,3,255 -quiver[-1]  
[-2],10



```
Example 376 : image.jpg --resize2dy 600 -luminance[0] -gradient[0] -mul[1] -1
-reverse[0,1] -append[0,1] c -blur[0] 8 -orientation[0] -quiver[1]
[0],10,10,1,0.8,255
```

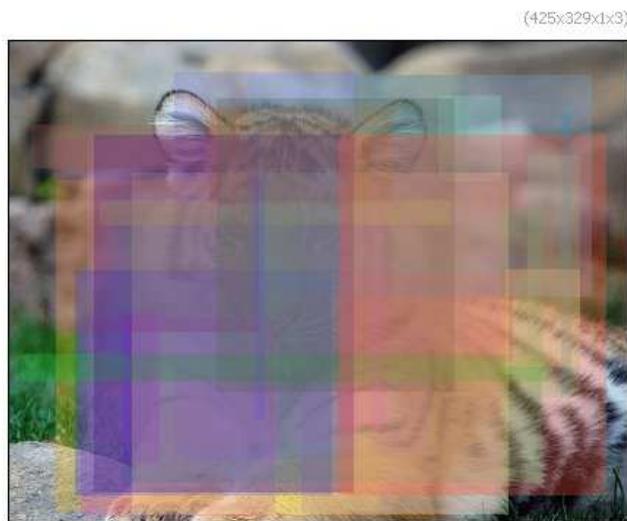
### 2.10.25 *-rectangle*

**Arguments:** `x0[%],y0[%],x1[%],y1[%],-opacity,-pattern,-color1,..`

Draw specified colored rectangle on selected images.

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

**Default values:** '`opacity=1'`, '`pattern=(undefined)`' and '`color1=0`'.



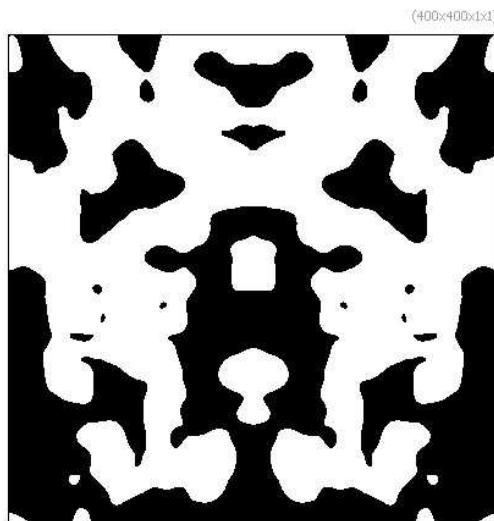
```
Example 377 : image.jpg -repeat 30 -rectangle
{?(100){%,{?(100){%,{?(100){%,{?(100){%,0.3,@{-RGB} -done
```

### 2.10.26 *-rorschach*

**Arguments:** `'smoothness[%]>=0','mirrorng={ 0=none | 1=x | 2=y | 3=xy }`

Render rorschach-like inkblots on selected images.

**Default values:** '`smoothness=5%`' and '`mirrorng=1`'.



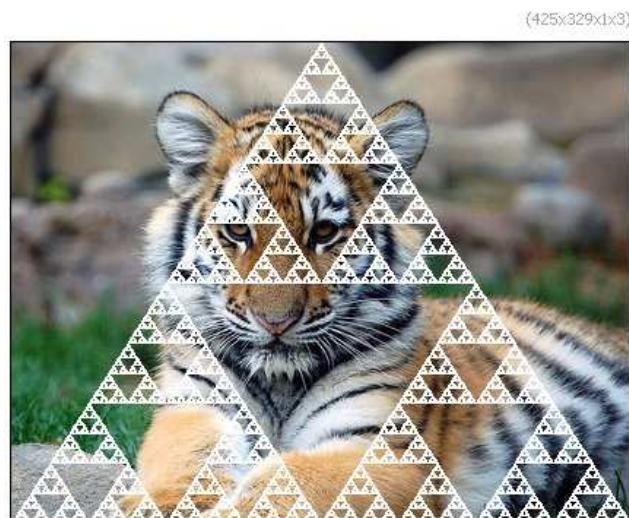
**Example 378 :** 400, 400 -rorschach 3%

### 2.10.27 -sierpinski

**Arguments:** recursion\_level>=0

Draw Sierpinski triangle on selected images.

**Default value:** 'recursion\_level=7' .



**Example 379 :** image.jpg -sierpinski 7

### 2.10.28 *-snowflake*

**Arguments:** `_recursion>=0, -x0, -y0, -x1, -y1, -x2, -y2, -opacity, -col1, ... -colN`

Draw a Koch snowflake on selected images.

**Default values:**     `'recursion=4', 'x0=20', 'y0=70', 'x1=80', 'y1=70', 'x2=50', 'y2=10', 'opacity=1' and 'col1=255'.`



**Example 380 :** `image.jpg -snowflake 4`

### 2.10.29 *-spiralbw*

Draw (squared) spiral on selected images.



**Example 381 :** `16, 16 -spiralbw`

### 2.10.30 -spline

**Arguments:** `x0[%],y0[%],u0[%],v0[%],x1[%],y1[%],u1[%],v1[%],_nb_vertices>=2,_opacity,_color1`

Draw specified colored spline curve on selected images.

**Default values:** '`nb_vertices=256`', '`opacity=1`' and '`color1=0`'.



**Example 382 :** `image.jpg -repeat 30 -spline {?(100)}%,{?(100)}%,{?(-600,600)},{?(-600,600)},{?(100)}%,{?(100)}%,{?(-600,600)},{?(-600,600)},256,-done`

### 2.10.31 -text (+)

**Arguments:** `text,_x[%],_y[%],_font_height>=0,_opacity,_color1,..`

Draw specified colored text string on selected images.

(*eq. to '-t'*).

Exact pre-defined sizes are '13','24','32' and '57'. Any other size is interpolated.

Specifying an empty target image resizes it to new dimensions such that the image contains the entire text string.

**Default values:** '`opacity=1`' and '`color1=0`'.



**Example 383 :** `image.jpg -resize2dy 600 y=0 -repeat 30 -text {2*$>}" : This is a nice text, isn't it ?",10,$y,{2*$>},0.9,255 y={$y+2*$>} -done`



**Example 384 :** `0 -text "G'MIC",0,0,24,1,255`

### 2.10.32 *-text\_outline*

**Arguments:** `text,-x[%],-y[%],-font_height>0,-outline>=0,-opacity,-color1,..`

Draw specified colored and outlined text string on selected images.

**Default values:** `'x=y=2', 'font_height=13', 'outline=2', 'opacity=1'` and `'color1=255'`.

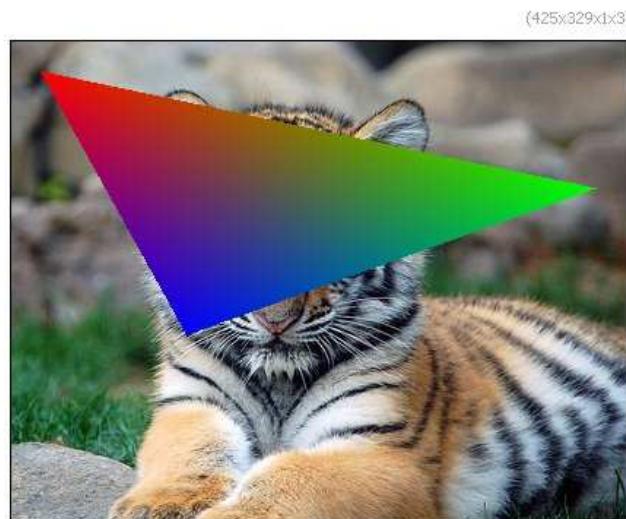


**Example 385 :** image.jpg -text\_outline "Hi there!",10,10,52,3

### 2.10.33 *-triangle\_shade*

**Arguments:** `x0,y0,x1,y0,x2,y2,R0,G0,B0,...,R1,G1,B1,...,R2,G2,B2,...`

Draw triangle with interpolated colors on selected images.



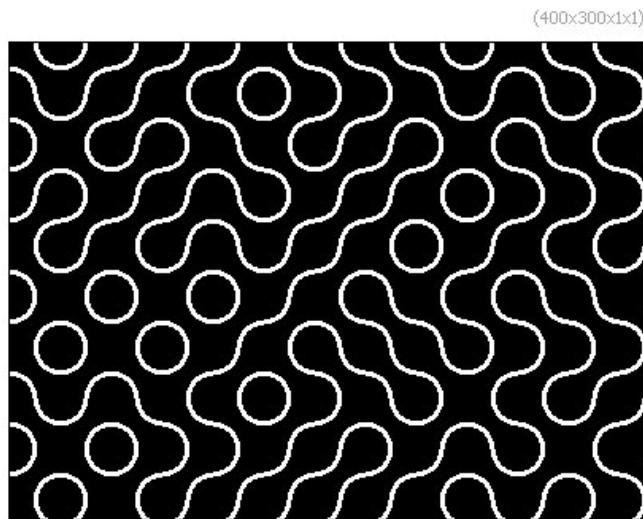
**Example 386 :** image.jpg -triangle\_shade  
20,20,400,100,120,200,255,0,0,0,255,0,0,0,255

### 2.10.34 *-truchet*

**Arguments:** `_scale>0,_radius>=0,_pattern_type={ 0=straight | 1=curved }`

Fill selected images with random truchet patterns.

**Default values:** '`scale=32'`, '`radius=5'` and '`pattern_type=1'`.



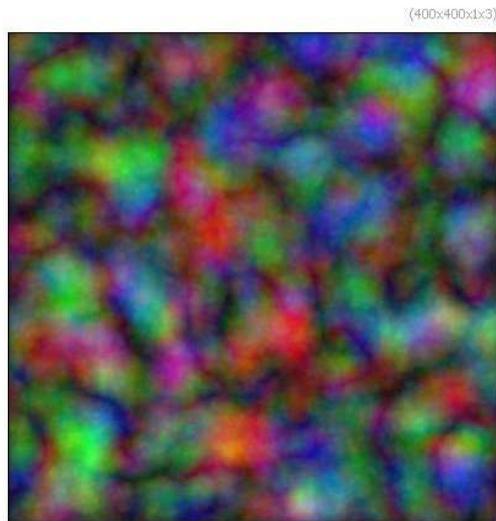
**Example 387:** `400,300 -truchet ,`

### 2.10.35 *-turbulence*

**Arguments:** `_radius>0`, `_octaves={1,2,3...,12}`, `_alpha>0`, `_difference={-10,10}`, `_mode={0,`

Render fractal noise or turbulence on selected images.

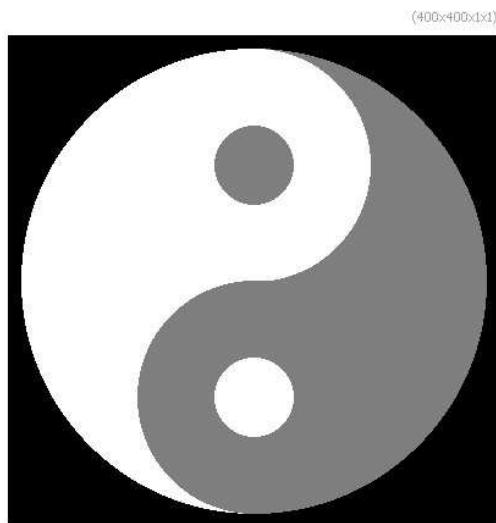
**Default values:** '`radius=32'`', '`octaves=6'`', '`alpha=3'`', '`difference=0'`' and '`mode=0'`.



**Example 388 :** 400, 400, 1, 3 -turbulence 16

### 2.10.36 -yinyang

Draw a yin-yang symbol on selected images.



**Example 389 :** 400, 400 -yinyang

## 2.11 Matrix computation

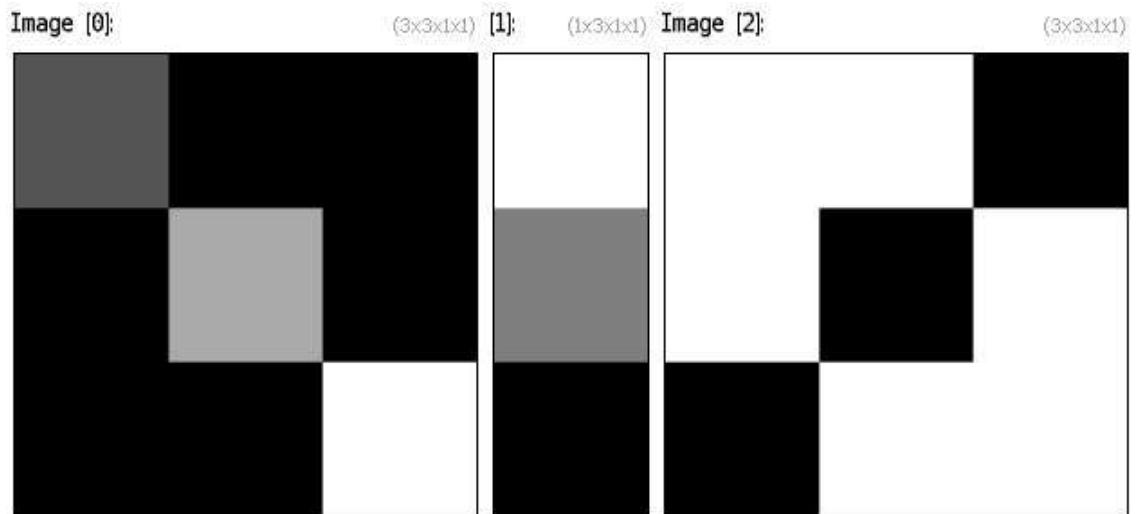
### 2.11.1 -dijkstra (+)

**Arguments:** starting\_node>=0, ending\_node>=0

Compute minimal distances and pathes from specified adjacency matrices by the Dijkstra algorithm.

### 2.11.2 *-eigen* (+)

Compute the eigenvalues and eigenvectors of selected symmetric matrices or matrix fields. If one selected image has 3 or 6 channels, it is regarded as a field of 2x2 or 3x3 symmetric matrices, whose eigen elements are computed at each point of the field.



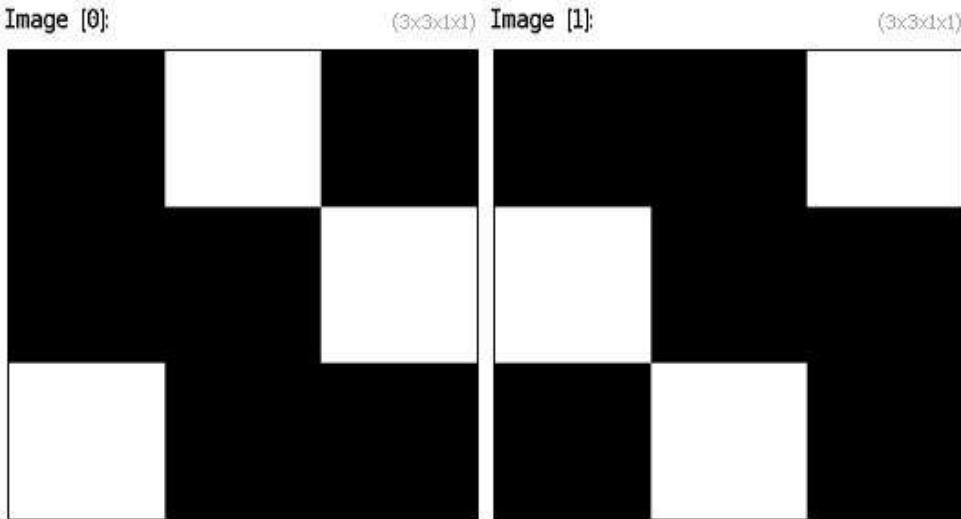
**Example 390 :** `(1, 0, 0; 0, 2, 0; 0, 0, 3) --eigen`



**Example 391 :** `image.jpg -structuretensors -blur 2 -eigen -split[0] c`

### 2.11.3 *-invert* (+)

Compute the inverse of the selected matrices.

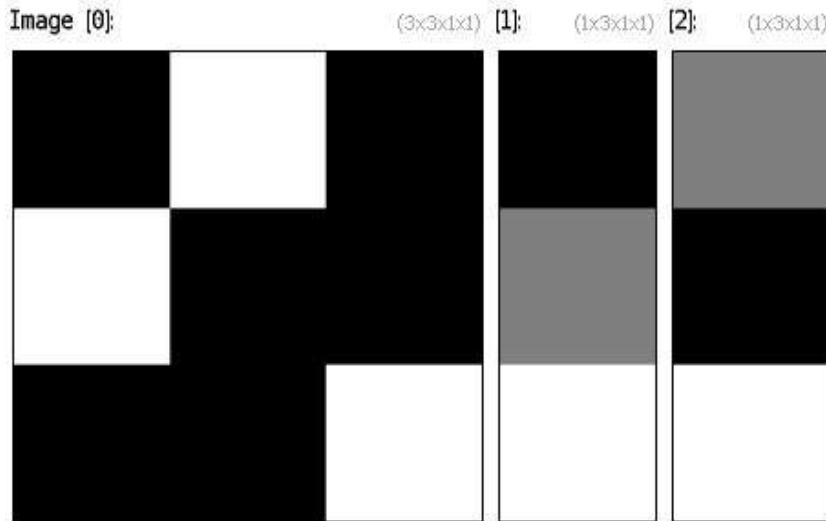


**Example 392 :**  $(0, 1, 0; 0, 0, 1; 1, 0, 0)$  --invert

#### 2.11.4 -solve (+)

**Arguments:** [image]

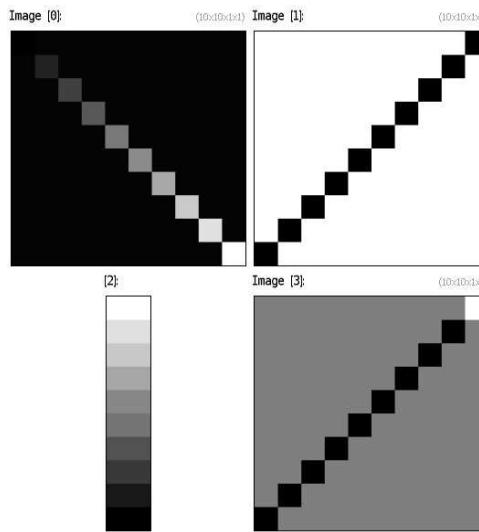
Solve linear system  $AX = B$  for selected  $B$ -vectors and specified  $A$ -matrix.  
If the system is under- or over-determined, least square solution is returned.



**Example 393 :**  $(0, 1, 0; 1, 0, 0; 0, 0, 1)$  (1;2;3) --solve[-1] [-2]

#### 2.11.5 -svd (+)

Compute SVD decomposition of selected matrices.



**Example 394 :** `10, 10, 1, 1, 'if (x==y, x+?(-0.2, 0.2), 0)' --svd`

### 2.11.6 *-transpose*

Transpose selected matrices.

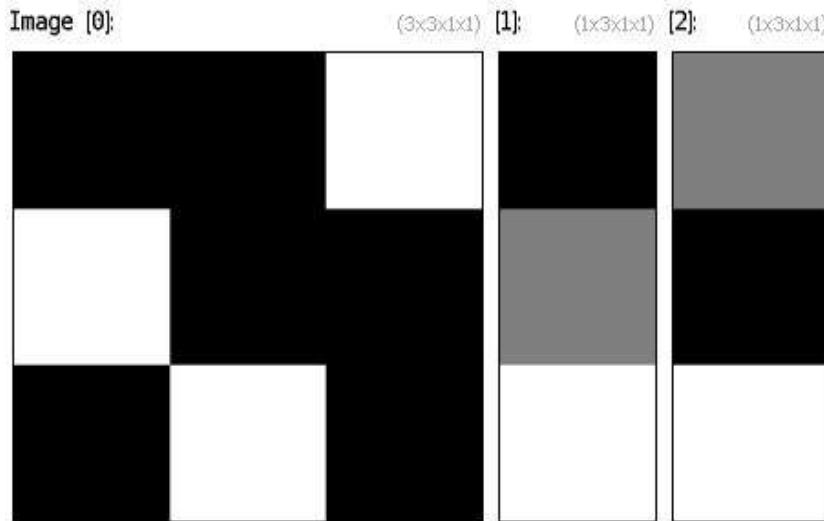


**Example 395 :** `image.jpg --transpose`

### 2.11.7 *-trisolve (+)*

**Arguments:** [image]

Solve tridiagonal system  $AX = B$  for selected B-vectors and specified tridiagonal A-matrix. Tridiagonal matrix must be stored as a 3 column vector, where 2nd column contains the diagonal coefficients, while 1st and 3rd columns contain the left and right coefficients.



**Example 396 :** (0,0,1;1,0,0;0,1,0) (1;2;3) --trisolve[-1] [-2]

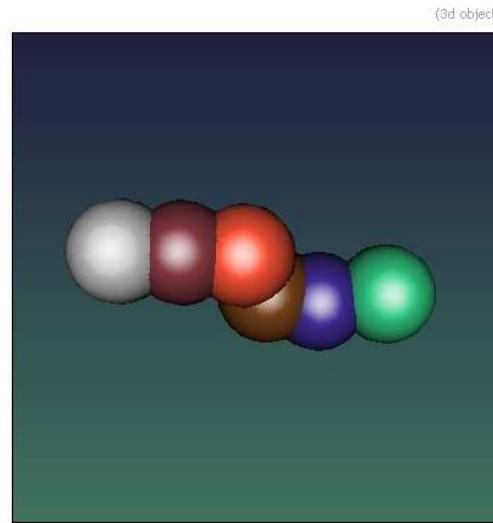
## 2.12 3d rendering

### 2.12.1 -add3d (+)

**Arguments:** tx,-ty,-tz |  
 [object3d] |  
 (noargs)

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 397 : -sphere3d 10 -repeat 5 --add3d[-1] 10,{?(-10,10)},0 -color3d[-1]
@{-RGB} -done -add3d
```



```
Example 398 : -repeat 20 -torus3d 15,2 -color3d[-1] @{-RGB} -mul3d[-1] 0.5,1 -if
{$>%2} -rotate3d[-1] 0,1,0,90 -endif -add3d[-1] 70 -add3d -rotate3d[-1]
0,0,1,18 -done -double3d 0
```

### 2.12.2 *-animate3d*

**Arguments:** *\_width>0, \_height>0, \_dx, \_dy, \_dz, \_zoom>=0, \_filename*

Animate selected 3d objects in a window.

**2.12.3 -apply\_camera3d**

**Arguments:** pos\_x, pos\_y, pos\_z, target\_x, target\_y, target\_z, up\_x, up\_y, up\_z

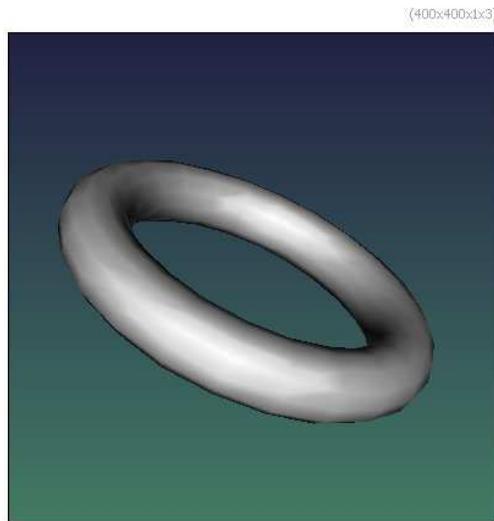
Apply 3d camera matrix to selected 3d objects.

**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\_pose3d**

**Arguments:** p1, ..., p12

Apply 3d pose matrix to selected 3d objects.



**Example 399 :** -torus3d 100,20 -apply\_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
```

**2.12.5 -axes3d**

**Arguments:** \_size\_x, \_size\_y, \_size\_z, \_font\_size>0, \_label\_x, \_label\_y, \_label\_z

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

**Default values:** 'size\_x=size\_y=size\_z=1', 'font\_size=24', 'label\_x=X', 'label\_y=Y' and 'label\_z=Z' .



**Example 400 :** `-axes3d ,`

### 2.12.6 *-background3d* (+)

**Arguments:** `R, _G, _B` |  
`[image]` |  
`(no args)`

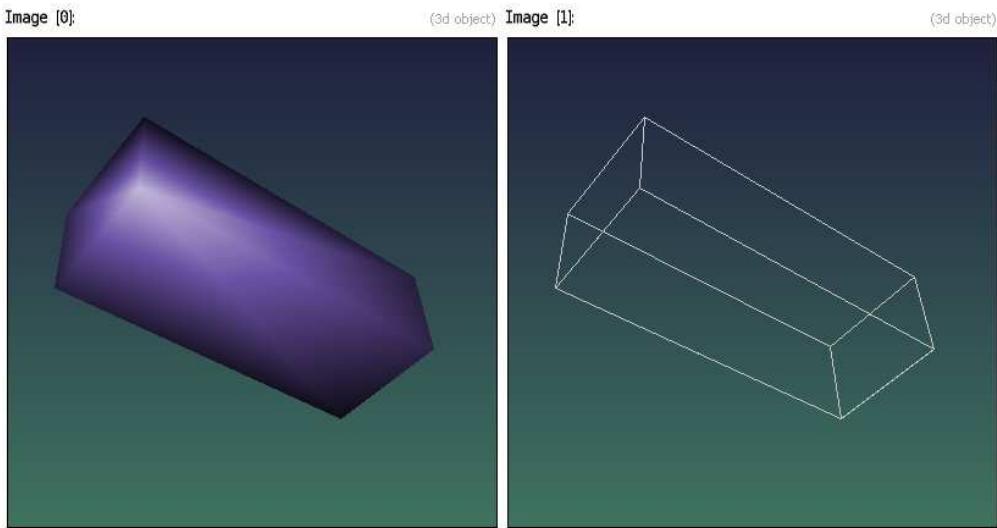
Define background from specified color or existing image for interactive 3d viewer.  
`(eq. to '-b3d').`  
`(no args)` resets the background to default.

### 2.12.7 *-box3d*

**Arguments:** `_size_x, _size_y, _size_z`

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

**Default values:** '`size_x=1`' and '`size_z=size_y=size_x`' .

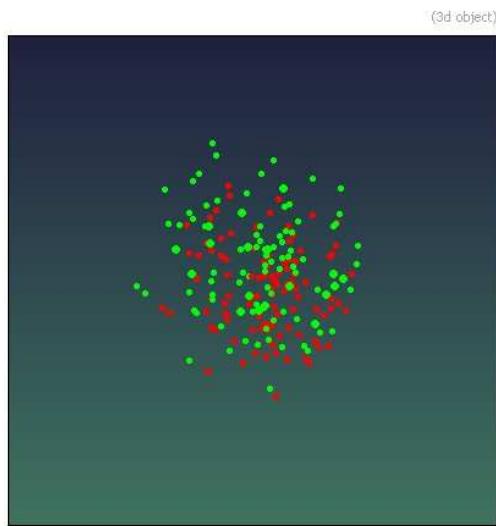


**Example 401 :** -box3d 100,40,30 --primitives3d 1 -color3d[-2] @{-RGB}

### 2.12.8 *-center3d*

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

(eq. to '*-c3d*').



**Example 402 :** -repeat 100 -circle3d {?(100)},{?(100)},{?(100)},2 -done -add3d -color3d[-1] 255,0,0 --center3d -color3d[-1] 0,255,0 -add3d

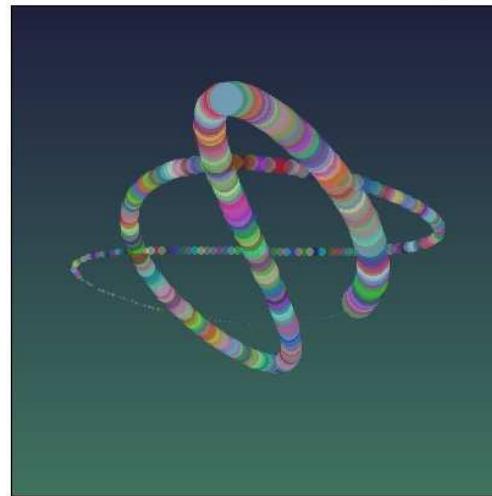
### 2.12.9 *-circle3d*

**Arguments:** *\_x0, \_y0, \_z0, \_radius>=0*

Input 3d circle at specified coordinates.

**Default values:** '`x0=y0=z0=0`' and '`radius=1`' .

(3d object)



```
Example 403 : -repeat 500 a={$>*pi/250} -circle3d
{cos(3*$a)},{sin(2*$a)},0,{\$a/50} -color3d[-1] @{-RGB},0.4 -done -add3d
```

### 2.12.10 `-circles3d`

**Arguments:** `_radius>=0`

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

(3d object)



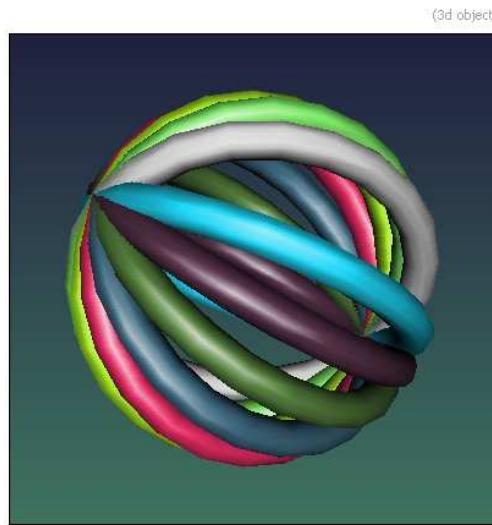
```
Example 404 : image.jpg -luminance -resize2dy 40 -threshold 50% -* 255
-pointcloud3d -color3d[-1] 255,255,255 -circles3d 0.7
```

**2.12.11 -color3d (+)**

**Arguments:** `R, _G, _B, _opacity`

Set color and opacity of selected 3d objects.  
(*eq. to '-col3d'*).

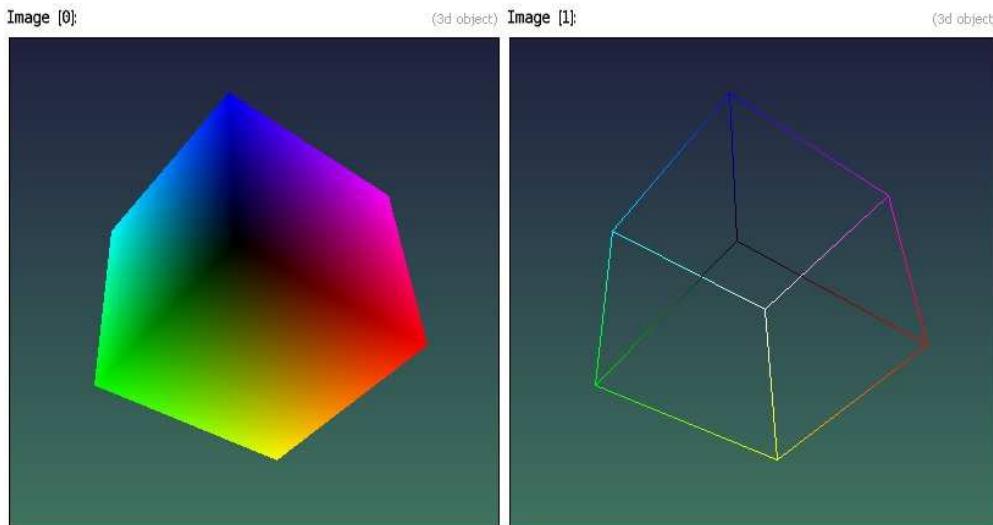
**Default value:** '`opacity=(undefined)`'.



**Example 405 :** `-torus3d 100,10 -double3d 0 -repeat 7 --rotate3d[-1] 1,0,0,20 -color3d[-1] @{-RGB} -done -add3d`

**2.12.12 -colorcube3d**

Input 3d color cube.



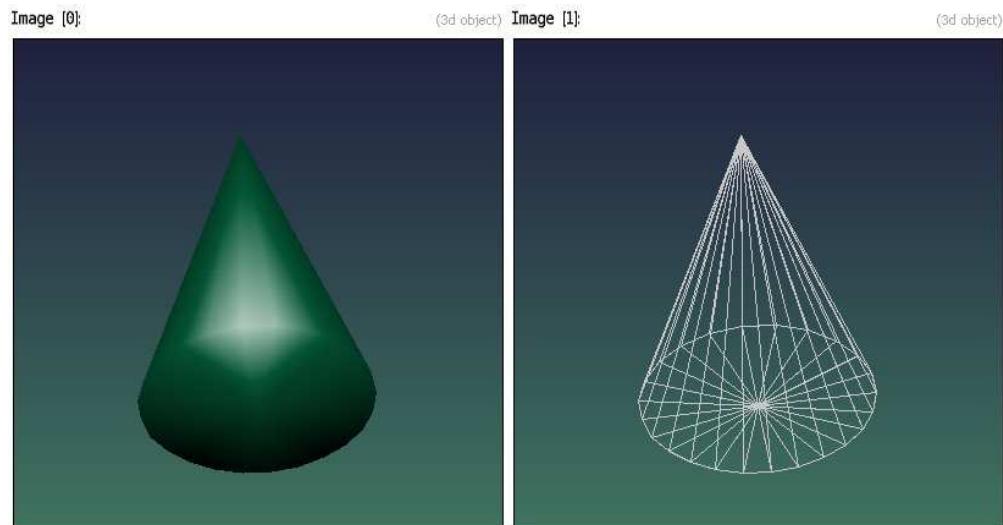
**Example 406 :** `-colorcube3d -mode3d 2 --primitives3d 1`

### 2.12.13 *-cone3d*

**Arguments:** `_radius, _height, _nb_subdivisions>0`

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

**Default value:** '`radius=1'`, '`height=1'` and '`nb_subdivisions=24'`.

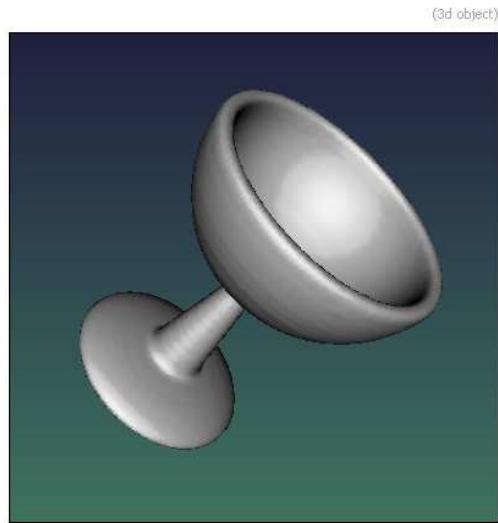


**Example 407 :** `-cone3d 10, 40 --primitives3d 1 -color3d[-2] @{-RGB}`

### 2.12.14 *-cup3d*

**Arguments:** `_resolution>0`

Input 3d cup object.



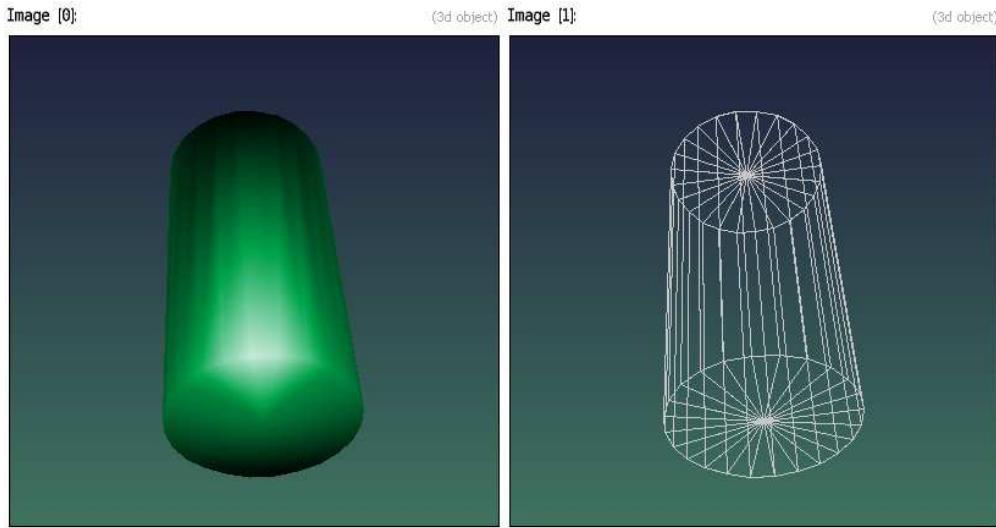
**Example 408 :** `-cup3d` ,

### 2.12.15 *-cylinder3d*

**Arguments:** `_radius, _height, _nb_subdivisions>0`

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

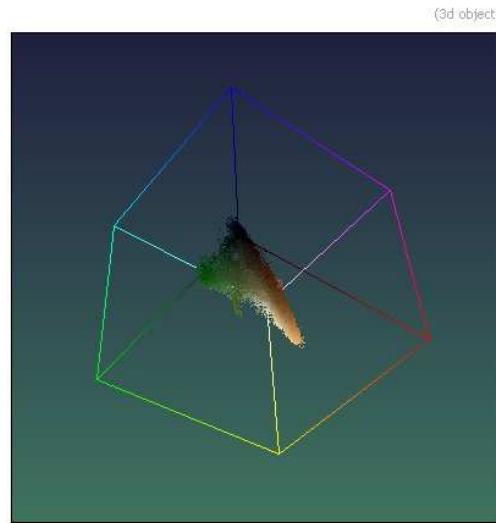
**Default value:** '`radius=1`', '`height=1`' and '`nb_subdivisions=24`' .



**Example 409 :** `-cylinder3d 10,40 --primitives3d 1 -color3d[-2] @{-RGB}`

### 2.12.16 *-distribution3d*

Get 3d color distribution of selected images.



**Example 410 :** `image.jpg -distribution3d -colorcube3d -primitives3d[-1] 1 -add3d`

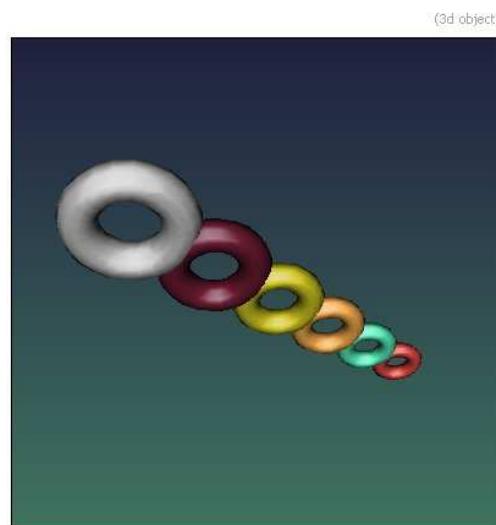
### 2.12.17 `-div3d (+)`

**Arguments:** `factor | factor_x, factor_y, factor_z`

Scale selected 3d objects isotropically or anisotropically, with the inverse of specified factors.

(*eq. to ' $-/3d'$ '*).

**Default value:** '`factor_z=0`'.

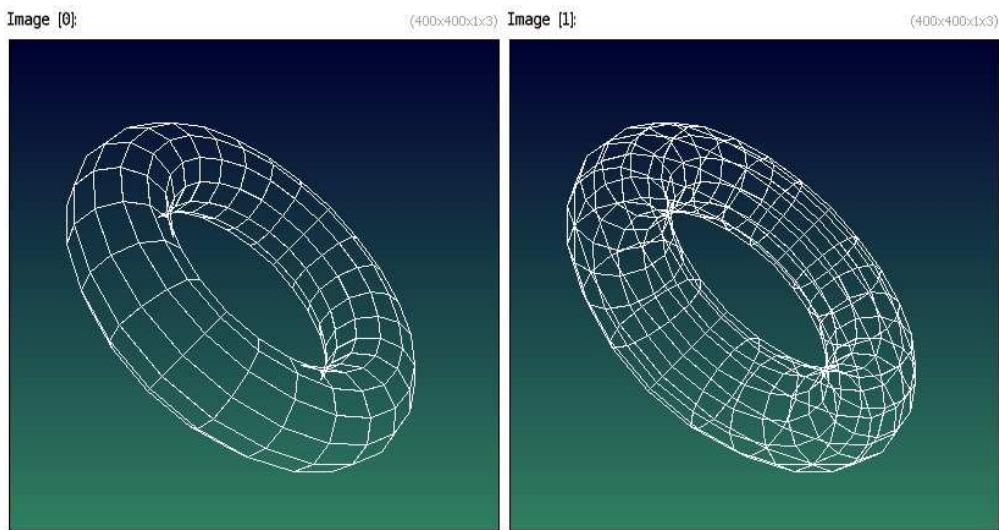


```
Example 411 : -torus3d 5,2 -repeat 5 --add3d[-1] 12,0,0 -div3d[-1] 1.2
                  -color3d[-1] @{-RGB} -done -add3d
```

### 2.12.18 **-double3d** (+)

**Arguments:** is\_doubled={ 0 | 1 }

Enable/disable double-sided mode for 3d rendering.  
(*e.g.* to '`-db3d`').



```
Example 412 : -mode3d 1 -repeat 2 -torus3d 100,30 -rotate3d[-1] 1,1,0,60
                  -double3d $> -snapshot3d[-1] 400 -done
```

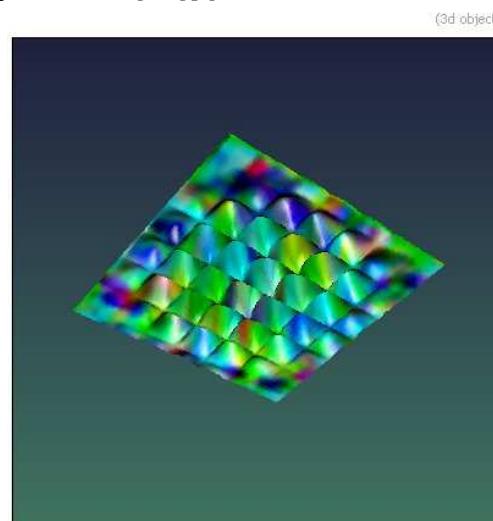
### 2.12.19 **-elevation3d** (+)

**Arguments:** z-factor |  
[elevation\_map] |  
'formula' |  
(no args)

Build 3d elevation of selected images, with a specified elevation map.  
When invoked with (no args) or 'z-factor', the elevation map is computed as the pointwise L2 norm of the pixel values. Otherwise, the elevation map is taken from the specified image or formula.



**Example 413 :** image.jpg -blur 5 -elevation3d 0.5



**Example 414 :** 128,128,1,3,?(255) -plasma 10,3 -blur 4 -sharpen 10000  
-elevation3d[-1]  
'X=(x-64)/6;Y=(y-64)/6;100\*exp(-(X^2+Y^2)/30)\*abs(cos(X)\*sin(Y))'

### 2.12.20 -empty3d

Input empty 3d object.



**Example 415 :** `-empty3d`

### 2.12.21 *-extrude3d*

**Arguments:** `_depth>0`, `_resolution>0`, `_smoothness[%]>=0`

Generate extruded 3d object from selected binary XY-profiles.

**Default values:** '`depth=16`', '`resolution=1024`' and '`smoothness=0.5%`'.



**Example 416 :** `image.jpg -threshold 50% -extrude3d 16`

### 2.12.22 **-focale3d** (+)

**Arguments:** `focale`

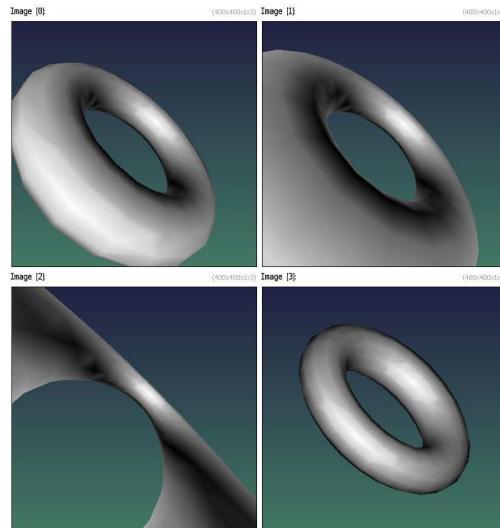
Set 3d focale.

(*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`' .

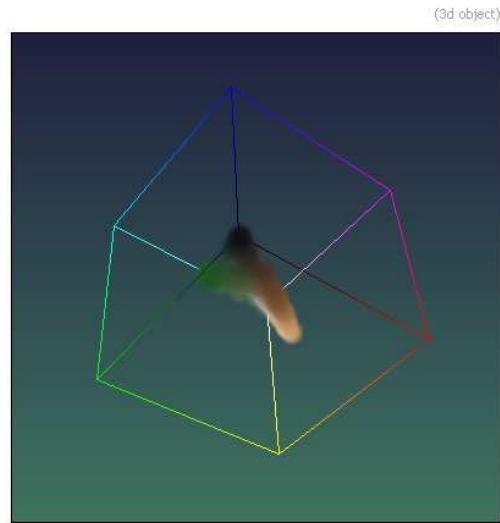


**Example 417 :** `-repeat 5 -torus3d 100,30 -rotate3d[-1] 1,1,0,60 -focale3d {$<*90}`  
`-snapshot3d[-1] 400 -done -remove[0]`

### 2.12.23 **-gaussians3d**

**Arguments:** `_size>0, _opacity`

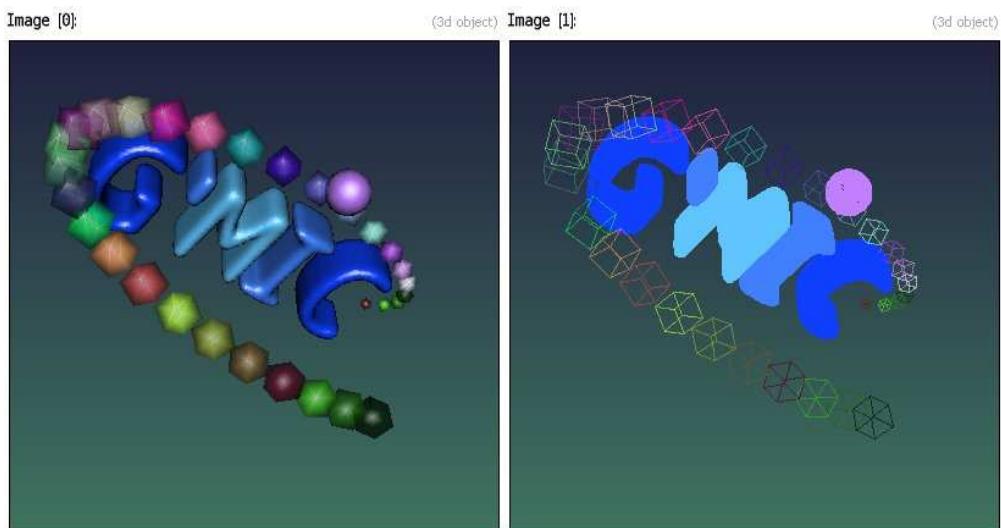
Convert selected 3d objects into set of 3d gaussian-shaped sprites.



```
Example 418 : image.jpg -r2dy 32 -distribution3d -gaussians3d 20 -colorcube3d  
-primitives3d[-1] 1 -+3d
```

### 2.12.24 *-gmic3d*

Input a 3d G'MIC logo.



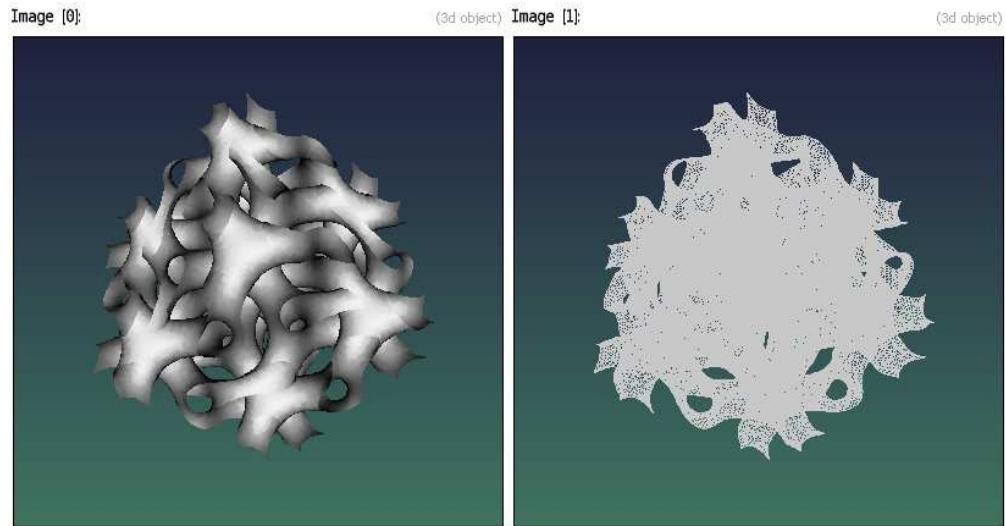
```
Example 419 : -gmic3d --primitives3d 1
```

### 2.12.25 *-gyroid3d*

**Arguments:** `_resolution>0, _zoom`

Input 3d gyroid at (0,0,0), with specified resolution.

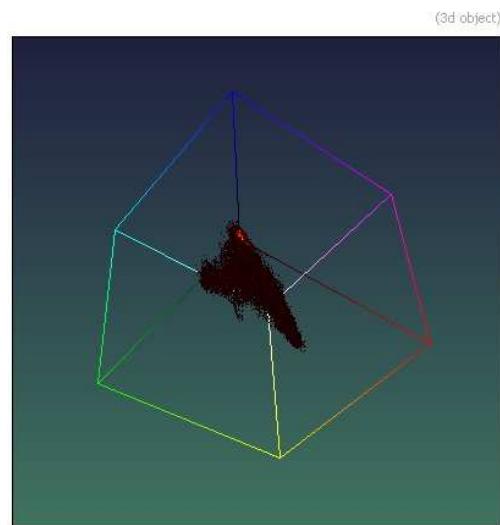
**Default values:** 'resolution=32' and 'zoom=5'.



**Example 420 :** -gyroid3d 48 --primitives3d 1

### 2.12.26 *-histogram3d*

Get 3d color histogram of selected images.



**Example 421 :** image.jpg -histogram3d -colorcube3d -primitives3d[-1] 1 -add3d

### 2.12.27 *-image6cube3d*

Generate 3d mapped cubes from 6-sets of selected images.



**Example 422 :** `image.jpg -animate flower,"30,0","30,5",6 -image6cube3d`

### 2.12.28 *-imagecube3d*

Generate 3d mapped cubes from selected images.



**Example 423 :** `image.jpg -imagecube3d`

### 2.12.29 *-imageplane3d*

Generate 3d mapped planes from selected images.



**Example 424 :** image.jpg -imageplane3d

### 2.12.30 *-imagepyramid3d*

Generate 3d mapped pyramids from selected images.



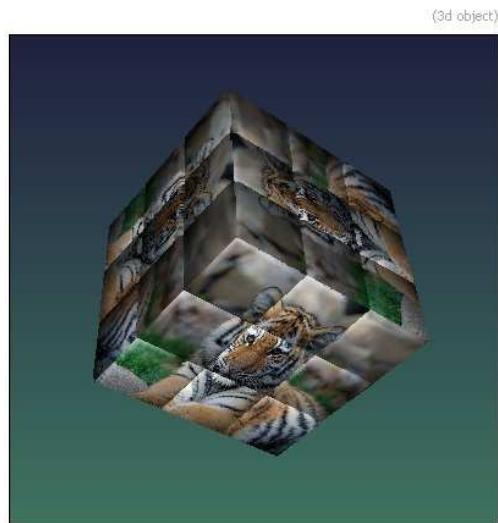
**Example 425 :** image.jpg -imagepyramid3d

### 2.12.31 *-imagerubik3d*

**Arguments:** `xy_tiles>=1, 0<=xy_shift<=100, 0<=z_shift<=100`

Generate 3d mapped rubik's cubes from selected images.

**Default values:** '`xy_tiles=3`', '`xy_shift=5`' and '`z_shift=5`'.



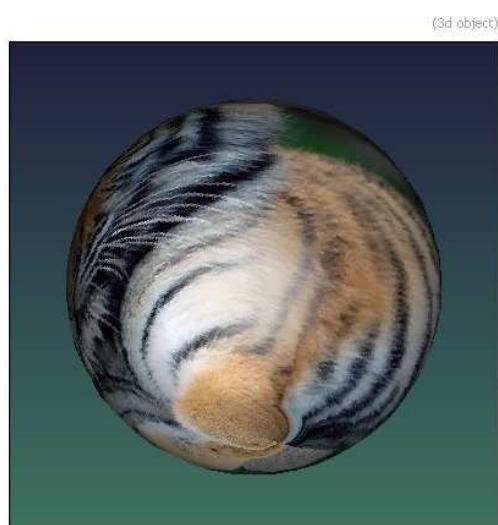
**Example 426 :** image.jpg -imagerubik3d ,

### 2.12.32 -imagesphere3d

**Arguments:** `_resolution1>=3, _resolution2>=3`

Generate 3d mapped sphere from selected images.

**Default values:** '`resolution1=32`' and '`resolutions2=16`' .



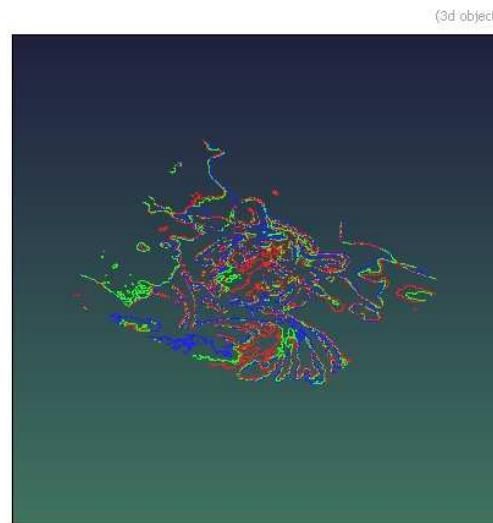
**Example 427 :** image.jpg -imagesphere3d 32,16

### 2.12.33 -isoline3d (+)

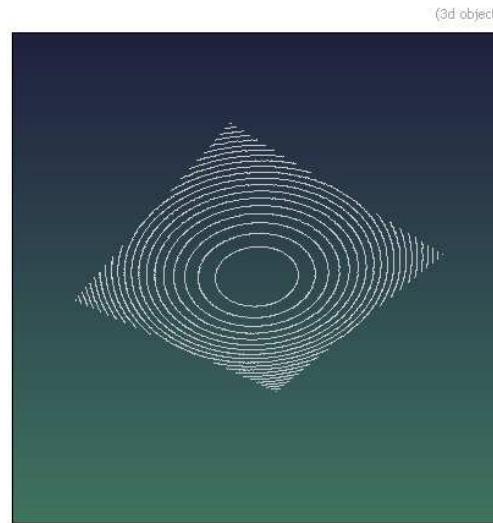
**Arguments:** isovalue[%] |  
     'formula', value, -x0, -y0, -x1, -y1, -size\_x>0[%], -size\_y>0[%]

Extract 3d isolines with specified value from selected images or from specified formula.

**Default values:** 'x0=y0=-3', 'x1=y1=3' and 'size\_x=size\_y=256'.



**Example 428 :** image.jpg -blur 1 -isoline3d 50%



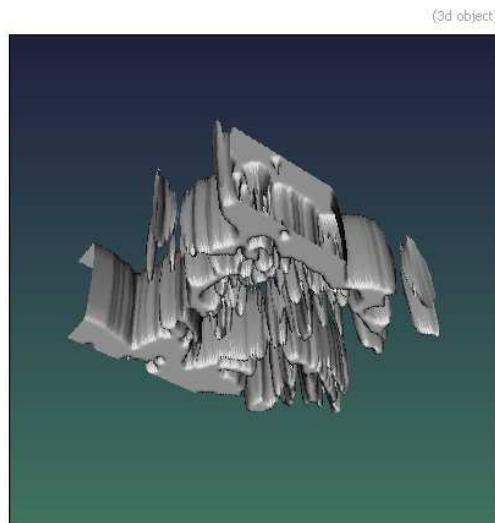
**Example 429 :** -isoline3d 'x=x-w/2; y=y-h/2; (X^2+Y^2)%20', 10, -10, -10, 10, 10

### 2.12.34 *-isosurface3d* (+)

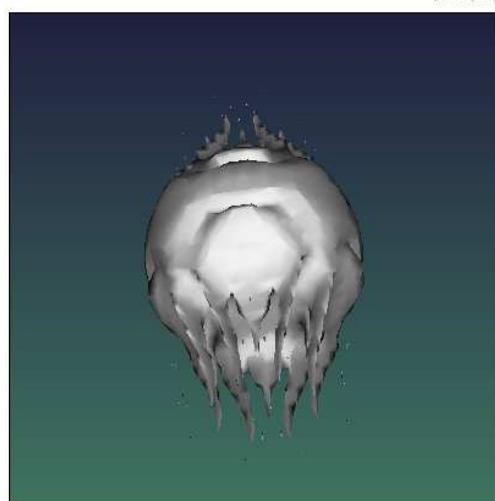
**Arguments:** `isovalue[%] | 'formula', value, -x0, -y0, -z0, -x1, -y1, -z1, -size_x>0[%], -size_y>0[%], -size_z>0[%]`

Extract 3d isosurfaces with specified value from selected images or from specified formula.

**Default values:** '`x0=y0=z0=-3`', '`x1=y1=z1=3`' and '`size_x=size_y=size_z=32`'.



**Example 430 :** `image.jpg -resize2dy 128 -luminance -threshold 50% -expand_z 2,0 -blur 1 -isosurface3d 50% -mul3d 1,1,30`



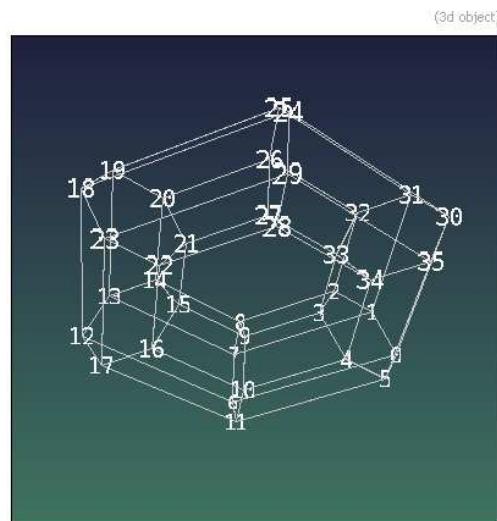
**Example 431 :** `-isosurface3d 'x^2+y^2+abs(z)^abs(4*cos(x*y*z*3))', 3`

### 2.12.35 *-label\_points3d*

**Arguments:** `_label_size>0, _opacity`

Add a numbered label to all vertices of selected 3d objects.

**Default values:** '`label_size=13'` and '`opacity=0.8'`.



**Example 432 :** `-torus3d 100,40,6,6 -label_points3d 24,1 -mode3d 1`

### 2.12.36 *-lathe3d*

**Arguments:** `_resolution>0, _smoothness [%]>=0, _max_angle>=0`

Generate 3d object from selected binary XY-profiles.

**Default values:** '`resolution=128', 'smoothness=0.5%`' and '`max_angle=361'`.



**Example 433 :** 300,300 -rand -1,1 -blur 40 -sign -normalize 0,255 -lathe3d ,

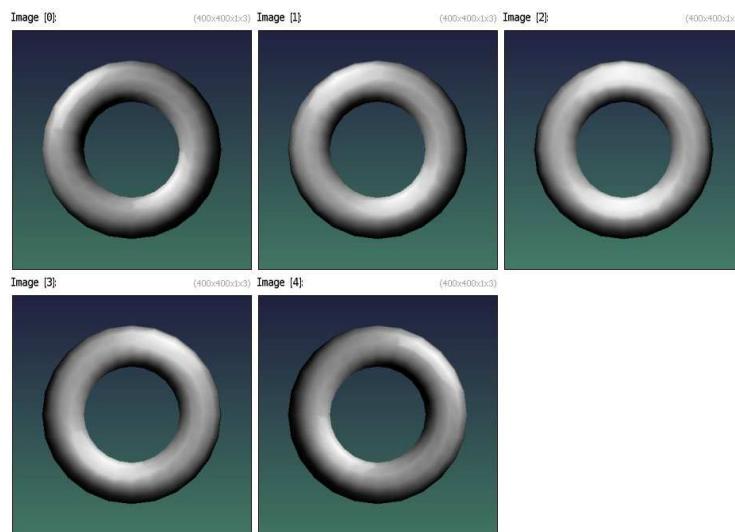
### 2.12.37 -light3d (+)

**Arguments:** position\_x,position\_y,position\_z |  
[texture] |  
(no args)

Set the light coordinates or the light texture for 3d rendering.

(eq. to '-l3d').

(noargs) resets the 3d light to default.

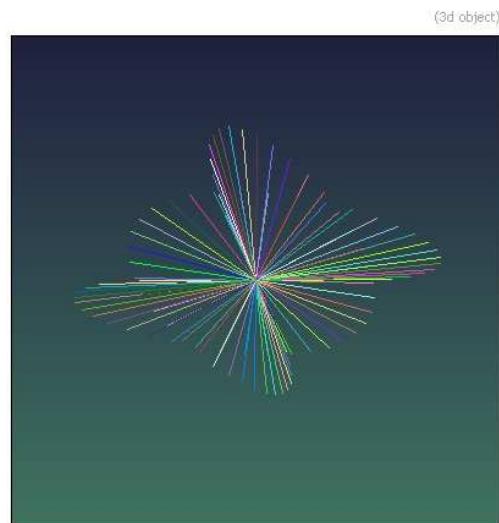


**Example 434 :** -torus3d 100,30 -double3d 0 -specs3d 1.2 -repeat 5 -light3d  
{\$>\*100},0,-300 --snapshot3d[0] 400 -done -remove[0]

### 2.12.38 *-line3d*

**Arguments:** `x0,y0,z0,x1,y1,z1`

Input 3d line at specified coordinates.



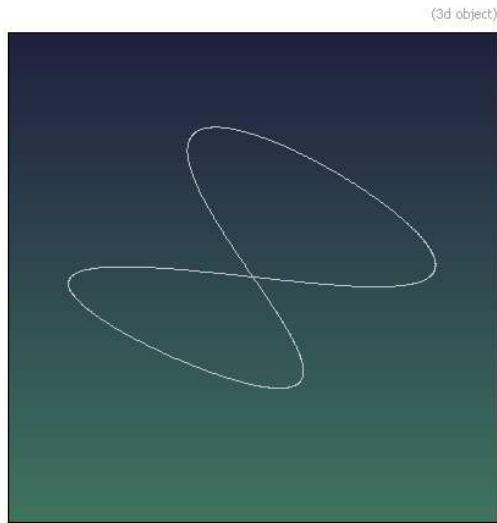
**Example 435 :** `-repeat 100 a={$>*pi/50} -line3d 0,0,0,{cos(3*$a)},{sin(2*$a)},0 -color3d[-1] @{-RGB} -done -add3d`

### 2.12.39 *-lissajous3d*

**Arguments:** `resolution>1,a,A,b,B,c,C`

Input 3d lissajous curves ( $x(t)=\sin(a*t+A*2*pi)$ , $y(t)=\sin(b*t+B*2*pi)$ , $z(t)=\sin(c*t+C*2*pi)$ ).

**Default values:** '`resolution=1024'`, '`a=2'`, '`A=0'`, '`b=1'`, '`B=0'`, '`c=0'` and '`C=0'`.



**Example 436 :** `-lissajous3d` ,

#### 2.12.40 `-mode3d` (+)

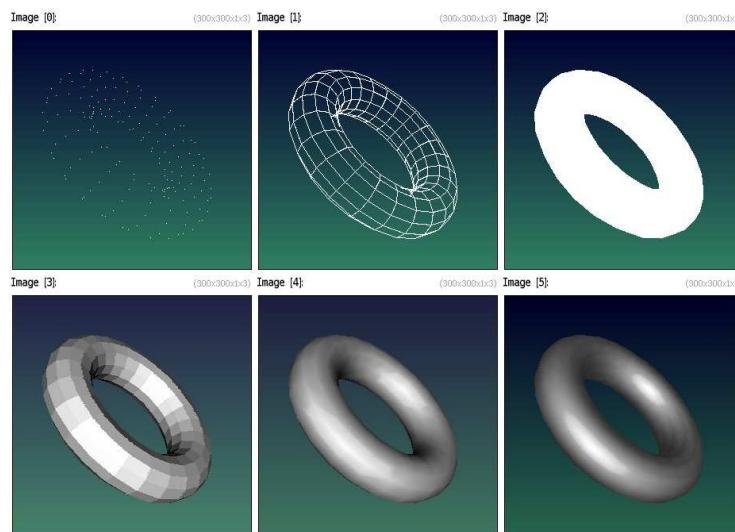
**Arguments:** mode

Set static 3d rendering mode.

(eq. to '`-m3d`').

'mode' can be { -1=bounding-box | 0=dots | 1=wireframe | 2=flat | 3=flat-shaded | 4=gouraud-shaded | 5=phong-shaded }.");

Bounding-box mode ('mode==1') is active only for the interactive 3d viewer.



**Example 437 :** `(0,1,2,3,4,5) -double3d 0 -repeat {w} -torus3d 100,30 -rotate3d[-1] 1,1,0,60 -mode3d @{0,$>} -snapshot3d[-1] 300 -done -remove[0]`

### 2.12.41 ***-moded3d*** (+)

**Arguments:** mode

Set dynamic 3d rendering mode for interactive 3d viewer.

(eq. to '`-md3d`').

'mode' can be { -1=bounding-box | 0=dots | 1=wireframe | 2=flat | 3=flat-shaded | 4=gouraud-shaded | 5=phong-shaded }.

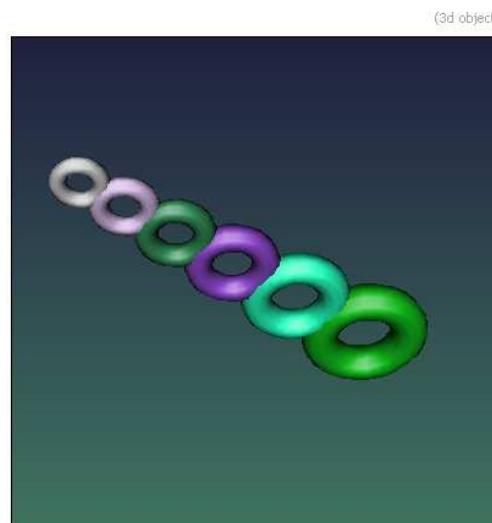
### 2.12.42 ***-mul3d*** (+)

**Arguments:** factor |  
factor\_x, factor\_y, factor\_z

Scale selected 3d objects isotropically or anisotropically, with specified factors.

(eq. to '`-*3d`').

**Default value:** '`factor_z=0`'.

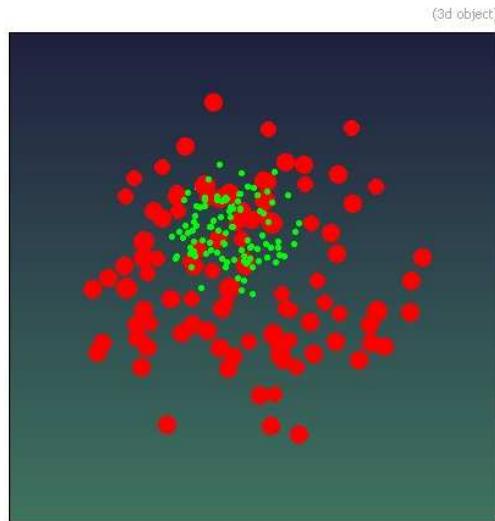


**Example 438 :** `-torus3d 5,2 -repeat 5 --add3d[-1] 10,0,0 -mul3d[-1] 1.2 -color3d[-1] @{-RGB} -done -add3d`

### 2.12.43 ***-normalize3d***

Normalize selected 3d objects to unit size.

(eq. to '`-n3d`').



**Example 439 :** -repeat 100 -circle3d {? (3)},{? (3)},{? (3)},0.1 -done -add3d -color3d[-1] 255,0,0 --normalize3d[-1] -color3d[-1] 0,255,0 -add3d

#### 2.12.44 -*opacity3d* (+)

**Arguments:** opacity

Set opacity of selected 3d objects.  
(*eq. to* '-o3d').



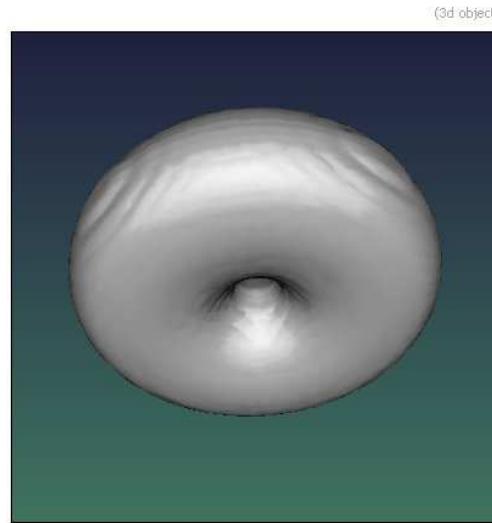
**Example 440 :** -torus3d 100,10 -double3d 0 -repeat 7 --rotate3d[-1] 1,0,0,20 -opacity3d[-1] {?} -done -add3d

### 2.12.45 -parametric3d

**Arguments:** `_x(a,b),_y(a,b),_z(a,b),_amin,_amax,_bmin,_bmax,_res_a>0,_res_b>0,_res_x>0`

Input 3d object from specified parametric surface ( $x(a,b), y(a,b), z(a,b)$ ).

**Default values:** `'x=(2+cos(b))*sin(a)', 'y=(2+cos(b))*cos(a)', 'c=sin(b)', 'amin=-pi', 'amax='pi', 'bmin=-pi', 'bmax='pi', 'res_a=512', 'res_b=res_a', 'res_x=64', 'res_y=res_x', 'res_z=res_y', 'smoothness=2%' and 'isovalue=10%'`.



**Example 441 :** `-parametric3d ,`

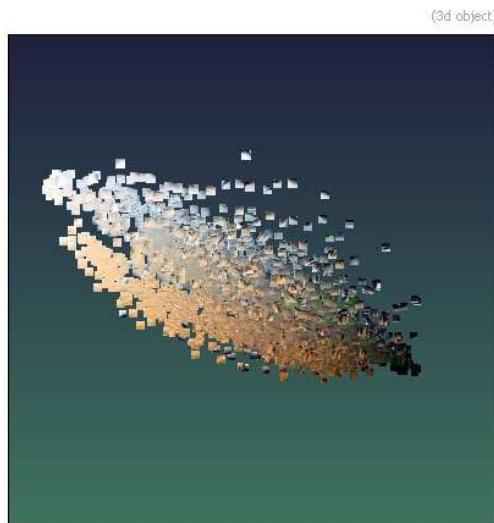
### 2.12.46 -pca\_patch3d

**Arguments:** `_patch_size>0, _M>0, _N>0, _normalize_input={ 0 | 1 }, _normalize_output={ 0 | 1 }, _lambda_xy`

Get 3d patch-pca representation of selected images.

The 3d patch-pca is estimated from M patches on the input image, and displayed as a cloud of N 3d points.

**Default values:** `'patch_size=7', 'M=1000', 'N=3000', 'normalize_input=1', 'normalize_output=0', and 'lambda_xy=0'`.



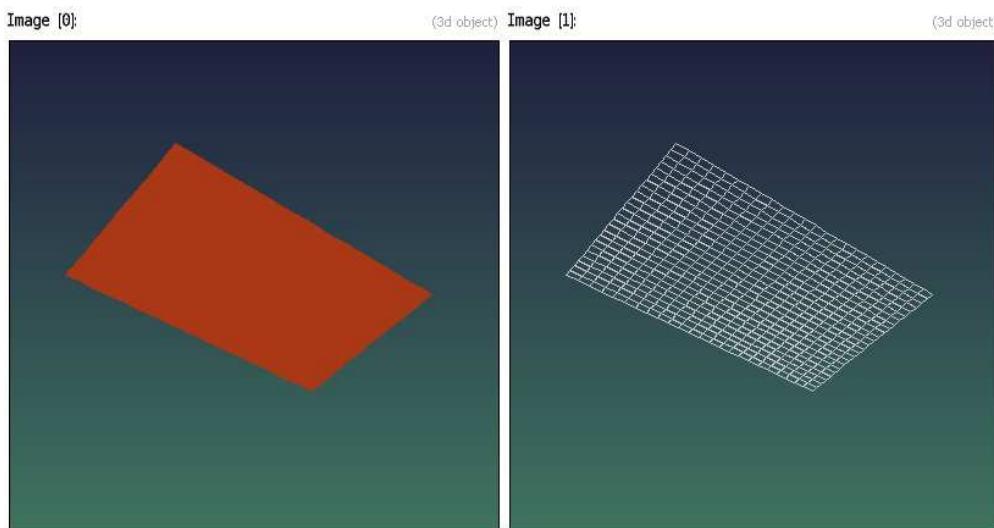
**Example 442 :** `image.jpg -pca-patch3d 7`

#### 2.12.47 *-plane3d*

**Arguments:** `_size_x, _size_y, _nb_subdivisions_x>0, _nb_subdivisions_y>0`

Input 3d plane at (0,0,0), with specified geometry.

**Default values:** '`size_x=1`', '`size_y=size_x`' and '`nb_subdivisions_x=nb_subdivisions_y=24`'.

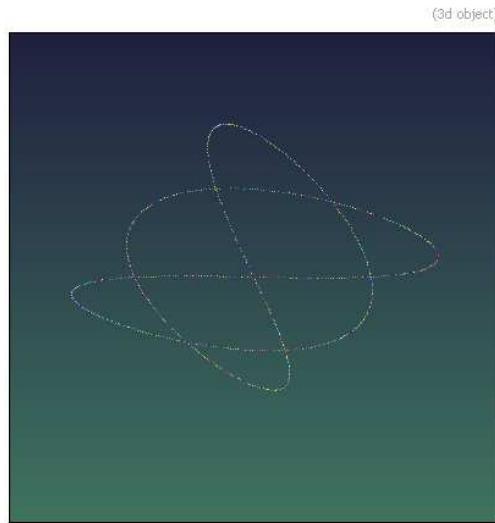


**Example 443 :** `-plane3d 50,30 --primitives3d 1 -color3d[-2] @{-RGB}`

**2.12.48 -point3d**

**Arguments:**  $x_0, y_0, z_0$

Input 3d point at specified coordinates.



**Example 444 :** `-repeat 1000 a={$>*pi/500} -point3d {\cos(3*a)}, {\sin(2*a)}, 0 -color3d[-1] @{-RGB} -done -add3d`

**2.12.49 -pointcloud3d**

Convert selected planar or volumetric images to 3d point clouds.



**Example 445 :** `image.jpg -luminance -resize2dy 100 -threshold 50% -* 255 -pointcloud3d -color3d[-1] 255,255,255`

**2.12.50 -pose3d (+)**

**Arguments:** value1, ..., value16 |  
(noargs)

Set the coefficients of the 3d pose matrix.  
(noargs) resets the 3d pose matrix to default.

**2.12.51 -primitives3d (+)**

**Arguments:** mode

Convert primitives of selected 3d objects.  
(eq. to '-p3d').  
'mode' can be { 0=points | 1=segments | 2=non-textured }.



**Example 446 :** -sphere3d 30 -primitives3d 1 -torus3d 50,10 -color3d[-1] @{-RGB}  
-add3d

**2.12.52 -projections3d**

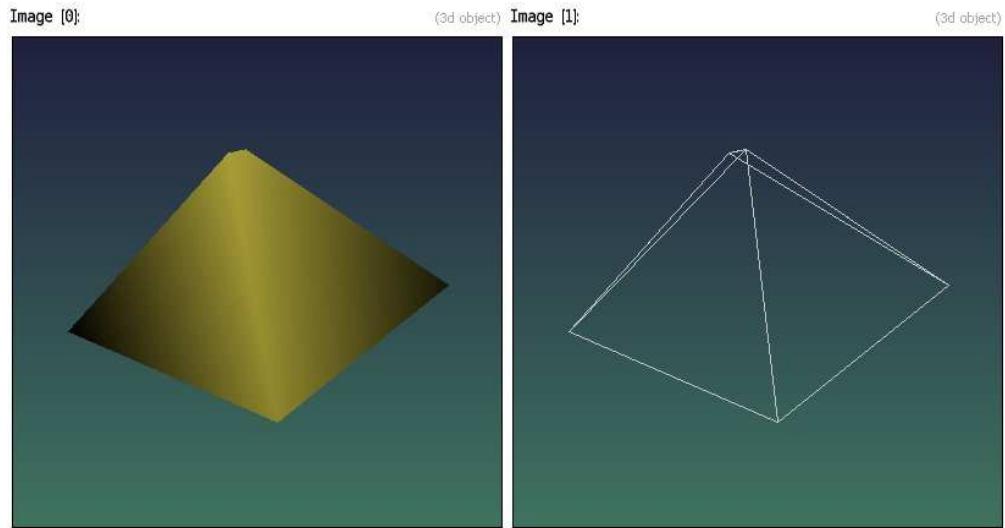
**Arguments:** -x [%], -y [%], -z [%], -is\_bounding\_box={ 0 | 1 }

Generate 3d xy,xz,yz projection planes from specified volumetric images.

**2.12.53 -pyramid3d**

**Arguments:** width, height

Input 3d pyramid at (0,0,0), with specified geometry.

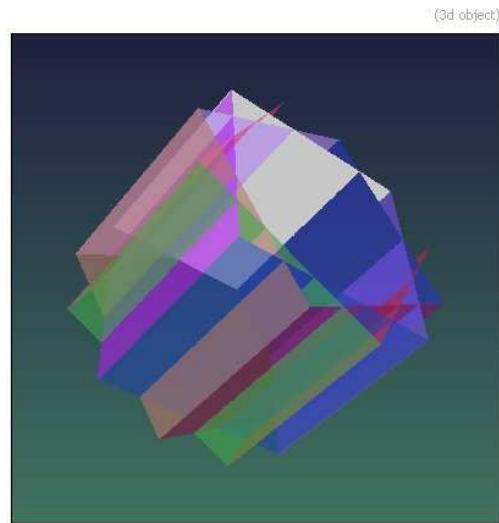


**Example 447 :** -pyramid3d 100,100 --primitives3d 1 -color3d[-2] @{-RGB}

### 2.12.54 -quadrangle3d

**Arguments:**  $x_0, y_0, z_0, x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3$

Input 3d quadrangle at specified coordinates.

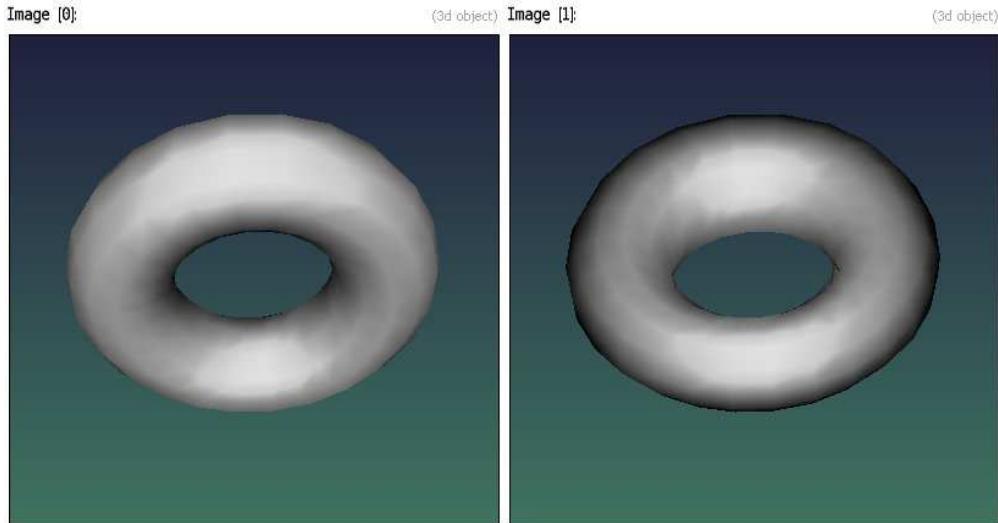


**Example 448 :** -quadrangle3d -10,-10,10,10,-10,10,10,10,10,-10,10,10 -repeat 10  
--rotate3d[-1] 0,1,0,30 -color3d[-1] @{-RGB},0.6 -done -add3d -mode3d 2

**2.12.55 *-reverse3d* (+)**

Reverse primitive orientations of selected 3d objects.

(eq. to '*-rv3d*').



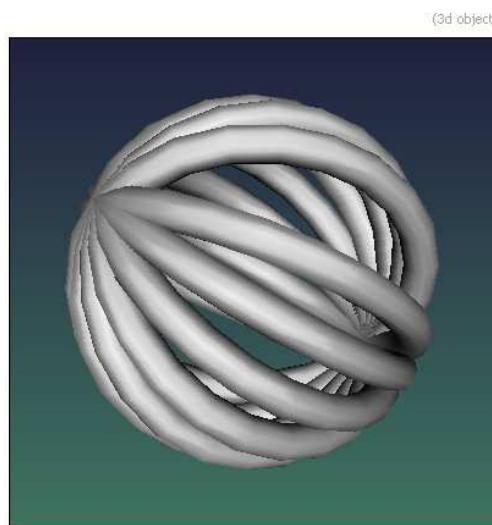
**Example 449 :** `-torus3d 100,40 -double3d 0 --reverse3d`

**2.12.56 *-rotate3d* (+)**

**Arguments:** *u, v, w, angle*

Rotate selected 3d objects around specified axis with specified angle (in deg.).

(eq. to '*-r3d*').

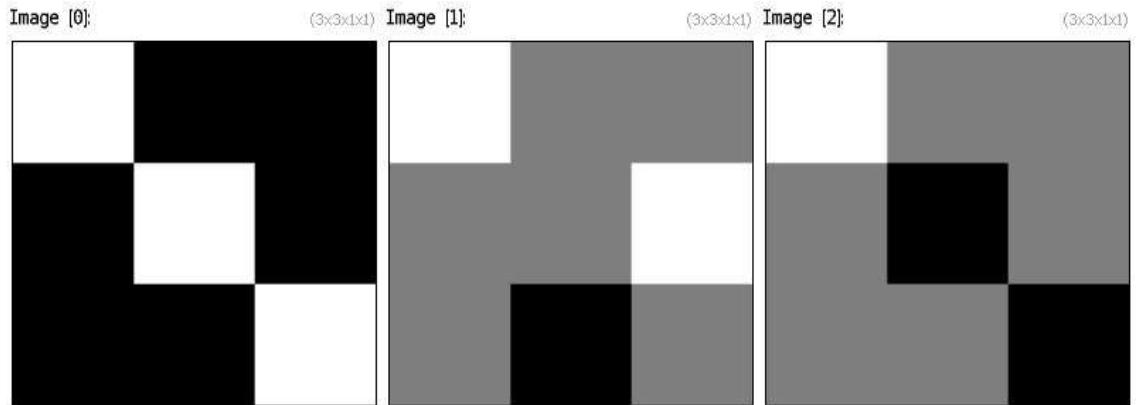


**Example 450 :** `-torus3d 100,10 -double3d 0 -repeat 7 --rotate3d[-1] 1,0,0,20 -done -add3d`

### 2.12.57 *-rotation3d*

**Arguments:** *u, v, w, angle*

Input 3x3 rotation matrix with specified axis and angle (in deg).

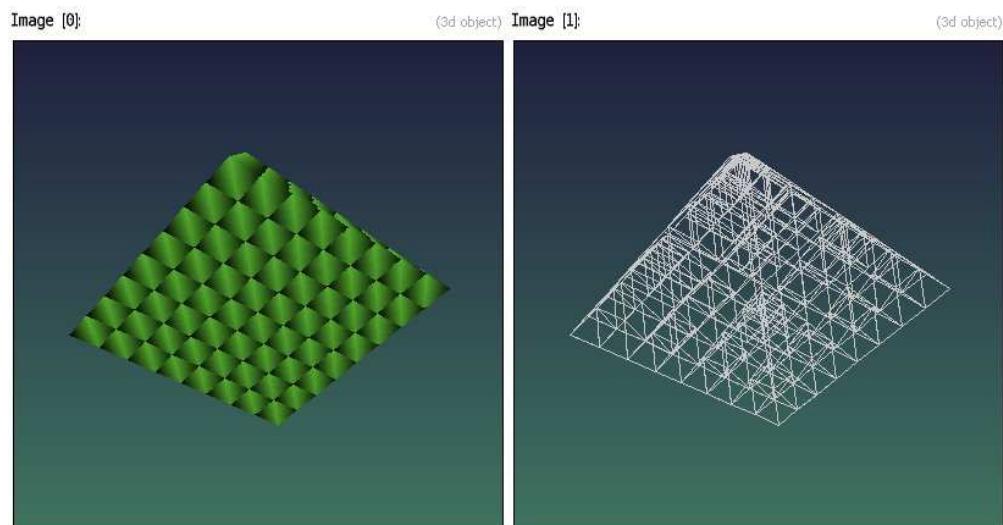


**Example 451 :** `-rotation3d 1,0,0,0 -rotation3d 1,0,0,90 -rotation3d 1,0,0,180`

### 2.12.58 *-sierpinski3d*

**Arguments:** *\_recursion\_level>=0, \_width, \_height*

Input 3d Sierpinski pyramid.



**Example 452 :** `-sierpinski3d 3 --primitives3d 1 -color3d[-2] @{-RGB}`

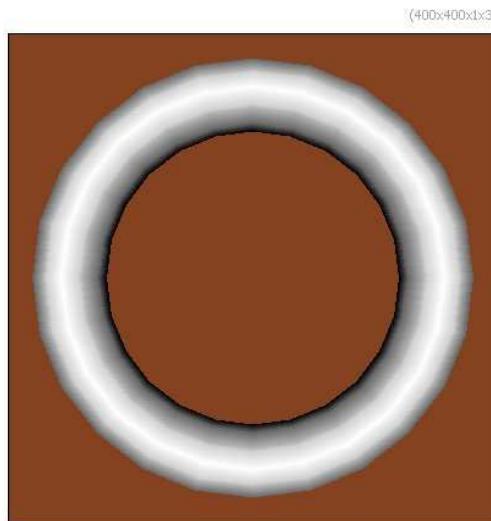
**2.12.59 -snapshot3d**

**Arguments:** `_size>0, _zoom>=0, _backgroundR, _backgroundG, _backgroundB`

Take 2d snapshots of selected 3d objects.

Set 'zoom' to 0 to disable object auto-scaling.

**Default values:** '`size=512`', '`zoom=1`' and '`backgroundR=backgroundG=backgroundB=(undefined)`'.



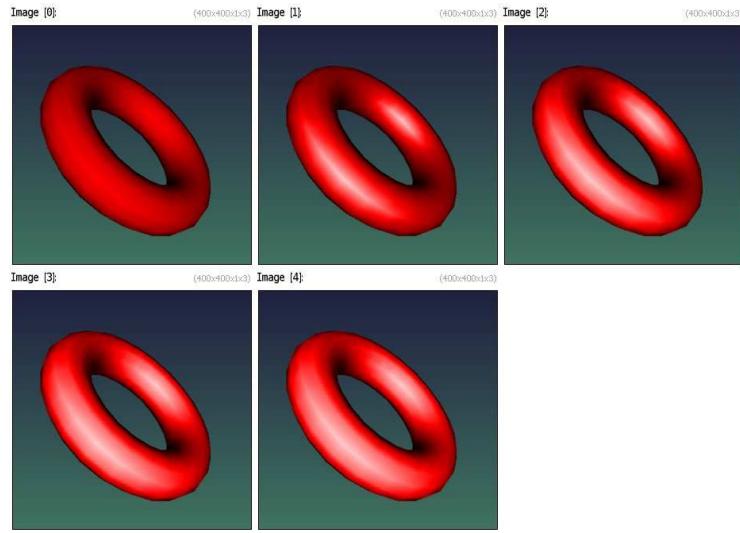
**Example 453 :** `-torus3d 100,20 -snapshot3d 400,1.2,128,64,32`

**2.12.60 -spec13d (+)**

**Arguments:** `value>=0`

Set lightness of 3d specular light.

(*eq. to '-sl3d'*).

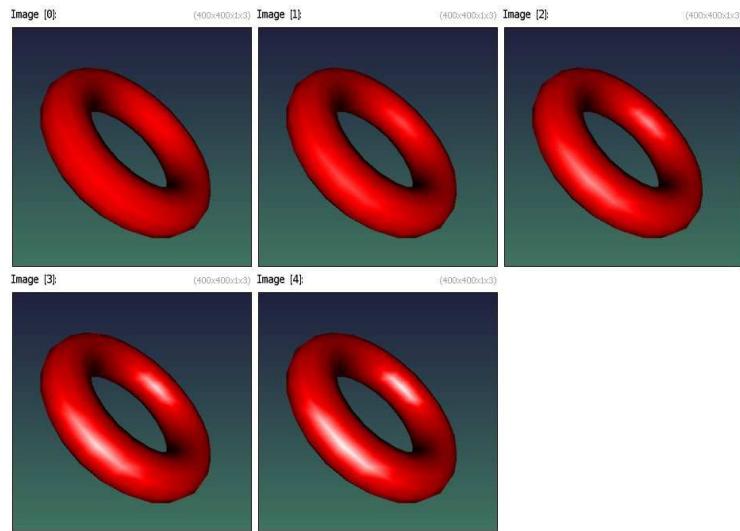


**Example 454 :** `(0,0.3,0.6,0.9,1.2) -repeat {w} -torus3d 100,30 -rotate3d[-1] 1,1,0,60 -color3d[-1] 255,0,0 -spec13d @{0,$>} -snapshot3d[-1] 400 -done -remove[0]`

### 2.12.61 `-specs3d (+)`

**Arguments:** `value>=0`

Set shininess of 3d specular light.  
(eq. to '`-ss3d`' ).



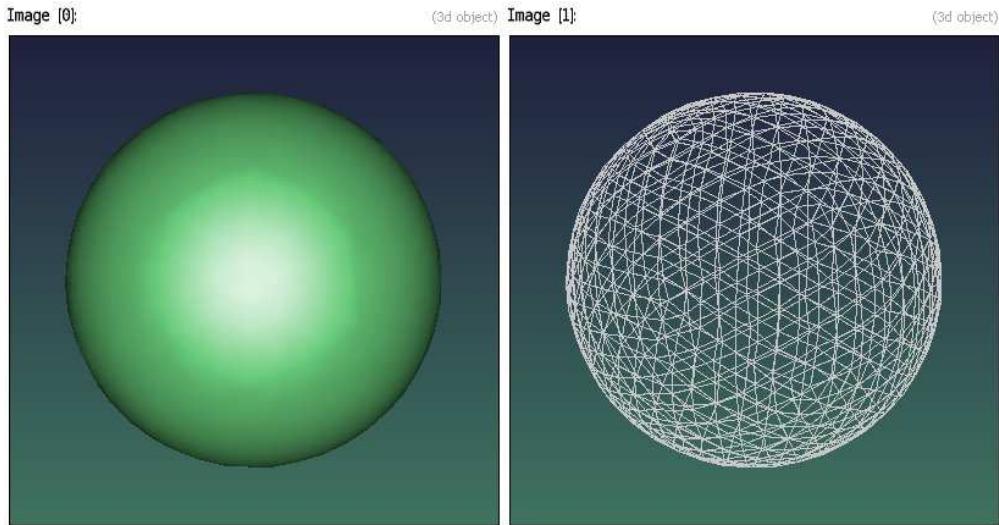
**Example 455 :** `(0,0.3,0.6,0.9,1.2) -repeat {w} -torus3d 100,30 -rotate3d[-1] 1,1,0,60 -color3d[-1] 255,0,0 -specs3d @{0,$>} -snapshot3d[-1] 400 -done -remove[0]`

**2.12.62 -sphere3d (+)**

**Arguments:** `radius, _nb_recursions>=0`

Input 3d sphere at (0,0,0), with specified geometry.

**Default value:** '`nb_recursions=3`' .



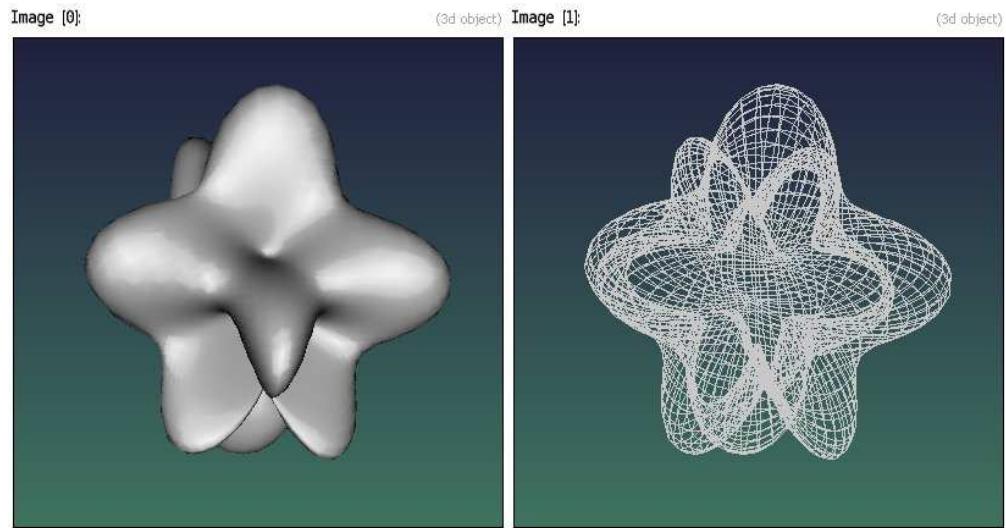
**Example 456 :** `-sphere3d 100 --primitives3d 1 -color3d[-2] @{-RGB}`

**2.12.63 -spherical3d**

**Arguments:** `_nb_azimuth>=3, _nb_zenith>=3, _radius_function(phi,theta)`

Input 3d spherical object at (0,0,0), with specified geometry.

**Default values:** '`nb_zenith=nb_azimuth=64`' and '`radius_function="abs(1+0.5*cos(3*phi))*sin(4*theta)"`'



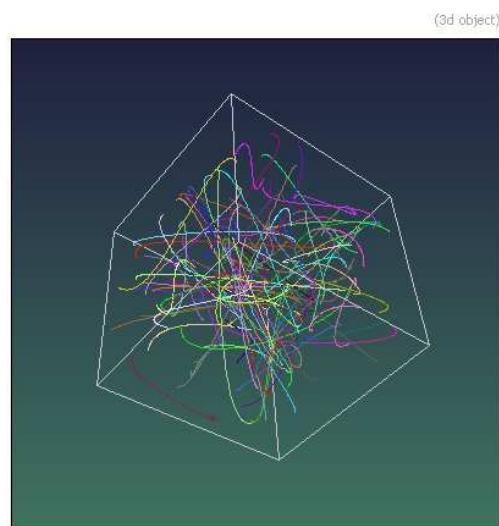
**Example 457 :** -spherical3d 64 --primitives3d 1

### 2.12.64 -spline3d

**Arguments:** x0[%], y0[%], z0[%], u0[%], v0[%], w0[%], x1[%], y1[%], z1[%], u1[%], v1[%], w1[%]

Input 3d spline with specified geometry.

**Default values:** 'nb\_vertices=128'.



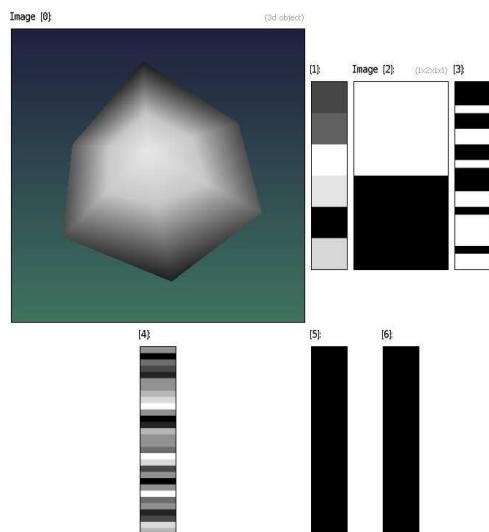
**Example 458 :** -repeat 100 -spline3d  
`{?},{?},{?},{?},{?},{?},{?},{?},{?},{?},{?},{?},{?},{?},{?},128 -color3d[-1] @{-RGB} -done  
-box3d 1 -primitives3d[-1] 1 -+3d

### 2.12.65 `-split3d (+)`

Split selected 3d objects into 6 feature vectors : { header, sizes, vertices, primitives, colors, opacities }.

(eq. to '`-s3d`' ).

To recreate the 3d object, append these 6 images along the y-axis.

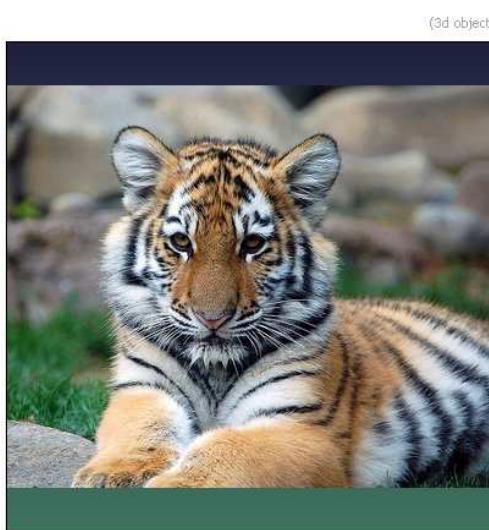


**Example 459 :** `-box3d 100 --split3d`

### 2.12.66 `-sprite3d`

Convert selected images as 3d sprites.

Selected image with alpha channels are managed.



**Example 460 :** `image.jpg -sprite3d`

### 2.12.67 *-sprites3d*

Convert selected 3d objects as sprites clouds, where the specified 2d sprite is the last selected image.

If the selected sprite has a 4th channel, it stands for the sprite alpha-channel (in [0,255]).



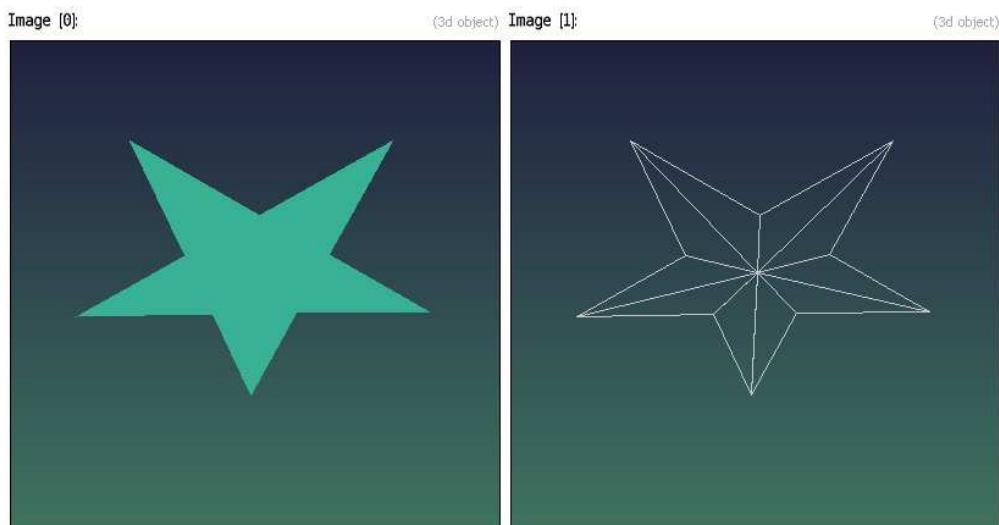
```
Example 461 : -torus3d 100,20 image.jpg -resize2dy[-1] 64 100%,100% -gaussian[-1] 30%,30% -*[-1] 255 -append[-2,-1] c --sprites3d -drgba[-2]
```

### 2.12.68 *-star3d*

**Arguments:** nb\_branches>0, 0<=thickness<=1

Input 3d star at (0,0,0), with specified geometry.

**Default values:** 'nb\_branches=5' and 'thickness=0.38' .



```
Example 462 : -star3d , --primitives3d 1 -color3d[-2] @{-RGB}
```

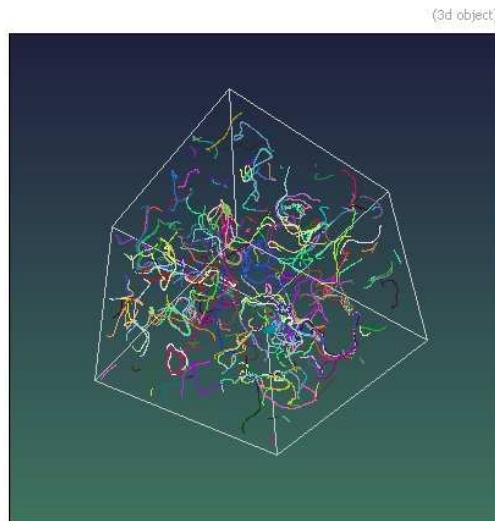
**2.12.69 -streamline3d (+)**

**Arguments:**     $x[\%], y[\%], z[\%], _L>=0, _dl>0, _interpolation, _is\_backward=\{ 0 | 1 \}, _is\_oriented=\{ 0 | 1 \} | 'formula', x, y, z, _L>=0, _dl>0, _interpolation, _is\_backward=\{ 0 | 1 \}, _is\_oriented=\{ 0 | 1 \}$

Extract 3d streamlines from selected vector fields or from specified formula.

'interpolation' can be { 0=nearest integer | 1=1st-order | 2=2nd-order | 3=4th-order }.

**Default values:**    'dl=0.1', 'interpolation=2', 'is\_backward=0'    and  
 'is\_oriented=0'.



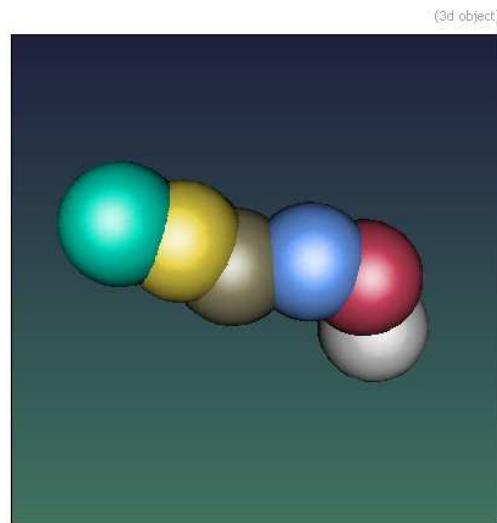
**Example 463 :** 100,100,100,3 -rand -10,10 -blur 3 -repeat 300 --streamline3d[0] {?(100)},{? (100)},{? (100)},1000,1,1 -color3d[-1] @{-RGB} -done -rm[0] -box3d 100 -primitives3d[-1] 1 -add3d

**2.12.70 -sub3d (+)**

**Arguments:**  $tx, _ty, _tz$

Shift selected 3d objects with the opposite of specified displacement vector.  
 (eq. to '--3d').

**Default values:** ' $ty=tz=0$ '.



**Example 464 :** -sphere3d 10 -repeat 5 --sub3d[-1] 10,{?(-10,10)},0 -color3d[-1] @{-RGB} -done -add3d

### 2.12.71 *-superformula3d*

**Arguments:** resolution>1,m>=1,n1,n2,n3

Input 2d superformula curve as a 3d object.

**Default values:** 'resolution=1024', 'm=8', 'n1=1', 'n2=5' and 'n3=8'.



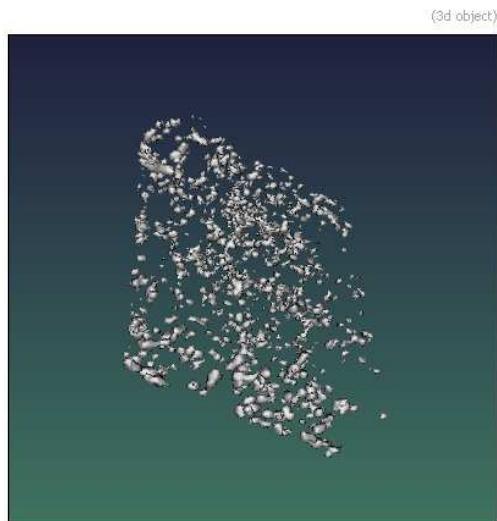
**Example 465 :** -superformula3d ,

**2.12.72 -text\_pointcloud3d**

**Arguments:** `_("text1"), _("text2"), _smoothness`

Input 3d text pointcloud from the two specified strings.

**Default values:** `'text1="text1'", 'text2="text2'"` and `'smoothness=1'`.



**Example 466 :** `-text_pointcloud3d "G'MIC", "Rocks!"`

**2.12.73 -text3d**

**Arguments:** `text, _font_height>0, _depth>0, _smoothness`

Input a 3d text object from specified text.

**Default values:** `'font_height=57', 'depth=10'` and `'smoothness=1.5'`.



**Example 467 :** `-text3d "GMIC as a\n3D logo!"`

### 2.12.74 `-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)'`.



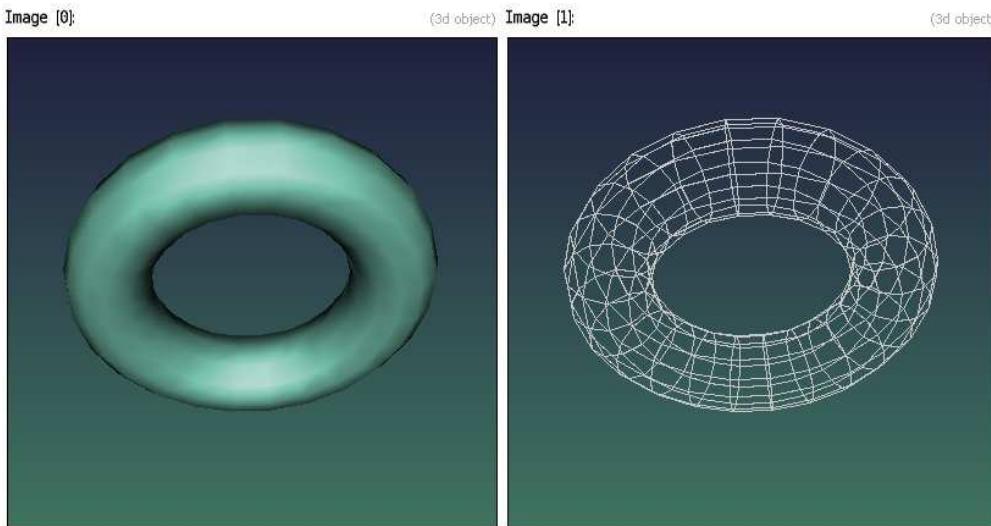
**Example 468 :** `image.jpg -torus3d 100,30 -texturize3d[-1] [-2] -keep[-1]`

**2.12.75 -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'`.

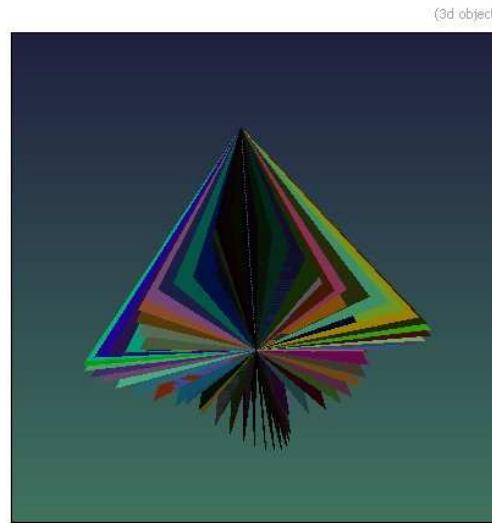


**Example 469 :** `-torus3d 10,3 --primitives3d 1 -color3d[-2] @{-RGB}`

**2.12.76 -triangle3d**

**Arguments:** `x0,y0,z0,x1,y1,z1,x2,y2,z2`

Input 3d triangle at specified coordinates.



```
Example 470 : -repeat 100 a={$>*pi/50} -triangle3d  
0,0,0,0,0,3,{cos(3*$a)},{sin(2*$a)},0 -color3d[-1] @{-RGB} -done -add3d
```

### 2.12.77 *-volume3d*

Transform selected 3d volumetric images as 3d parallelepipedic objects.



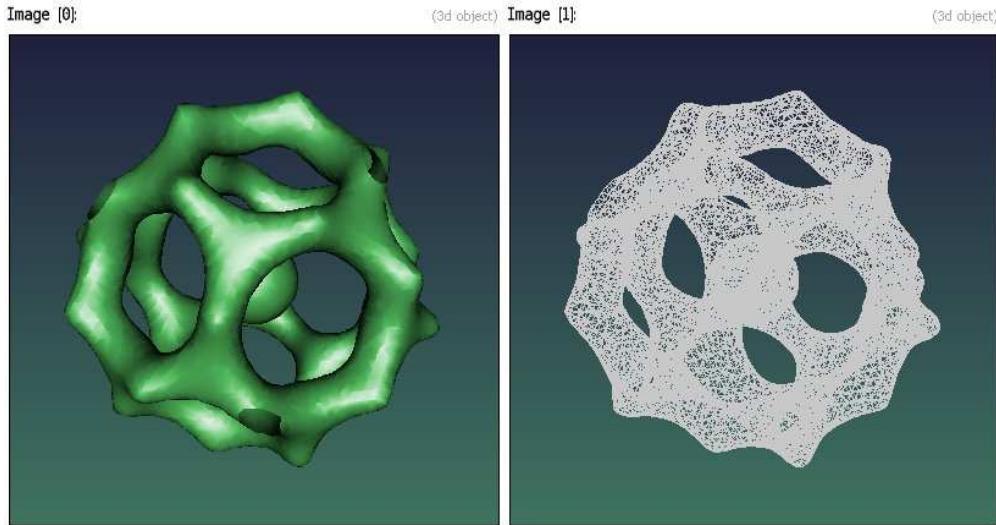
```
Example 471 : image.jpg -animate blur,0,5,30 -a z -volume3d
```

### 2.12.78 *-weird3d*

**Arguments:** *\_resolution>0*

Input 3d weird object at (0,0,0), with specified resolution.

**Default value:** 'resolution=32'.



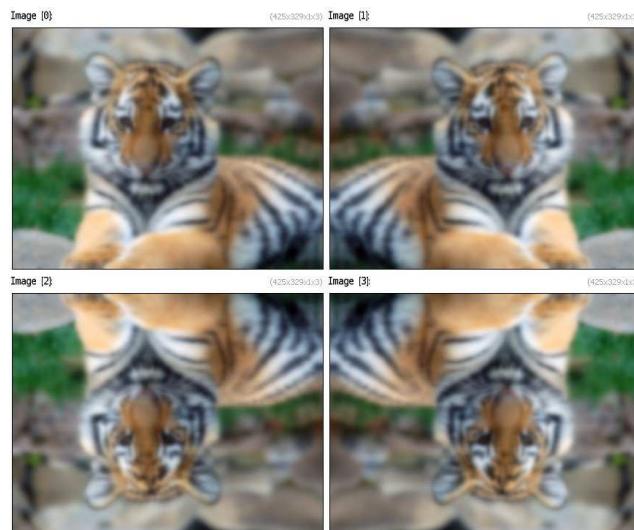
**Example 472 :** -weird3d 48 --primitives3d 1 -color3d[-2] @{-RGB}

## 2.13 Program controls

### 2.13.1 -apply\_parallel (\*)

**Arguments:** "command"

Apply specified command on each of the selected images, by parallelizing it for each image of the list.



**Example 473 :** image.jpg --mirror x --mirror y -apply\_parallel "-blur 3"

### 2.13.2 *-apply\_parallel2*

**Arguments:** `overlap[%], "command"`

Apply specified command on each of the selected images, by parallelizing it on two sub-image (2 threads used).

Default values: 'overlap=0','split\_axis=x'.



**Example 474 :** `image.jpg --apply_parallel2 16, "-smooth 500,0,1"`

### 2.13.3 *-apply\_parallel4*

**Arguments:** `overlap[%], "command"`

Apply specified command on each of the selected images, by parallelizing it on the four image quarters (4 threads used).

Default values: 'overlap=0'.



**Example 475 :** `image.jpg --apply_parallel4 16, "-smooth 500,0,1"`

#### 2.13.4 *-apply\_parallel*

**Arguments:** `overlap[%], "command"`

Apply specified command on each of the selected images, by parallelizing it on the four image quarters (8 threads used).

Default values: 'overlap=0'.



**Example 476 :** `image.jpg --apply_parallel8 16, "-smooth 500,0,1"`

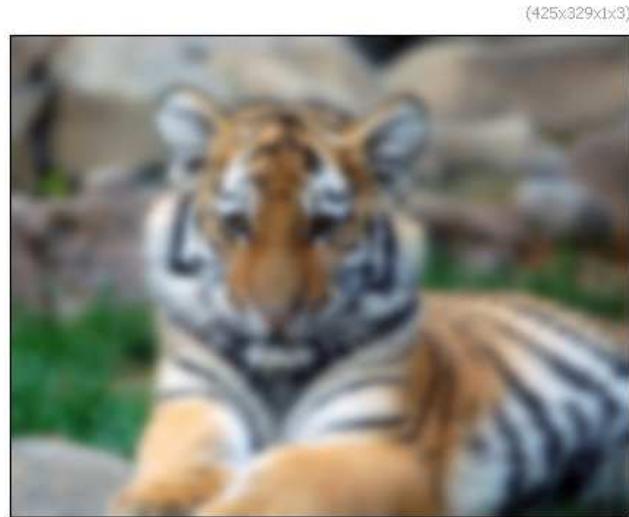
#### 2.13.5 *-check* (\*)

**Arguments:** `expression`

Evaluate specified expression and display an error message if evaluated to false.  
If 'expression' is not evaluable, it is regarded as a filename and checked if it exists.

#### 2.13.6 *-continue* (\*)

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



**Example 477 :** `image.jpg -repeat 10 -blur 1 -if {1==1} -continue -endif -deform 10 -done`

### 2.13.7 ***-break*** (\*)

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



**Example 478 :** `image.jpg -repeat 10 -blur 1 -if {1==1} -break -endif -deform 10 -done`

### 2.13.8 ***-do*** (\*)

Start a 'do..while' block.



```
Example 479 : image.jpg -luminance i={ia+2} -do -set 255,{?(100)}%,{?(100)}%
-while {ia<$i}
```

### 2.13.9 **-done** (\*)

End a 'repeat..done' block, and go to associated '-repeat' position, if iterations remain.

### 2.13.10 **-elif** (\*)

**Arguments:** boolean |  
filename

Start a 'elif..[else]..endif' block if previous '-if' was not verified and test if specified boolean is true, or if specified filename exists.

'boolean' can be a float number standing for { 0=false | other=true }.

### 2.13.11 **-else** (\*)

Execute following commands if previous '-if' or '-elif' conditions failed.

### 2.13.12 **-endif** (\*)

End a 'if..[elif]..[else]..endif' block.

### 2.13.13 **-endlocal** (\*)

End a 'local..endlocal' block.

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

### 2.13.14 ***-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 subset (if any) stands for displayed scope indices instead of image indices.

### 2.13.15 ***-exec*** (\*)

**Arguments:** command

Execute external command using a system call.

The status value is then set to the error code returned by the system call.  
(*eq. to '-x'*).

### 2.13.16 ***-if*** (\*)

**Arguments:** boolean |  
filename

Start a 'if..[elif]..[else]..endif' block and test if specified boolean is true, or if specified filename exists.

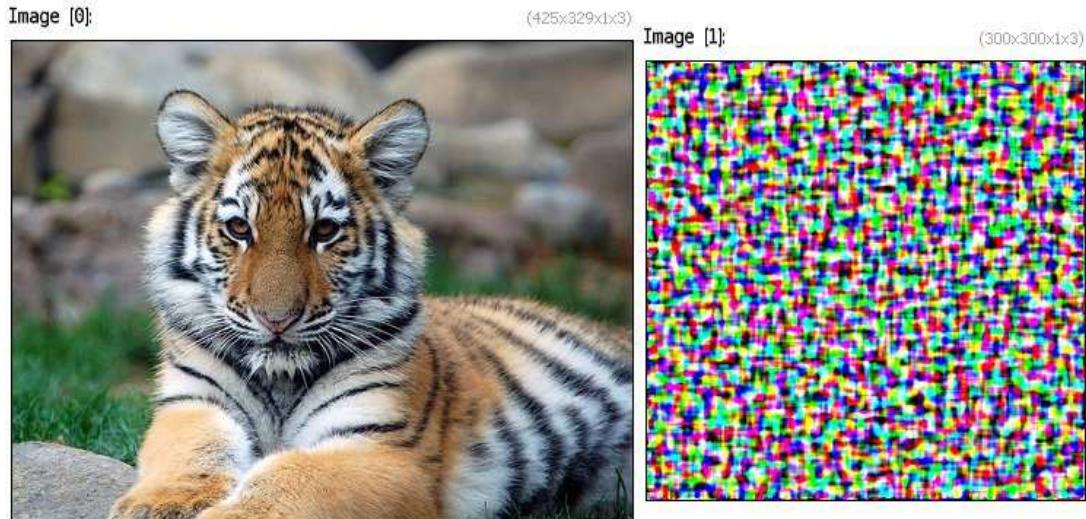
'boolean' can be a float number standing for { 0=false | other=true }.



**Example 480 :** image.jpg -if {ia<64} -add 50% -elif {ia<128} -add 25% -elif {ia<192} -sub 25% -else -sub 50% -endif -cut 0,255

### 2.13.17 *-local* (\*)

Start a 'local..[onfail]..endlocal' block, with selected images.  
(*eq. to '-1'*).



**Example 481 :** `image.jpg -local[] 300,300,1,3 -rand[0] 0,255 -blur 4 -sharpen 1000 -endlocal`

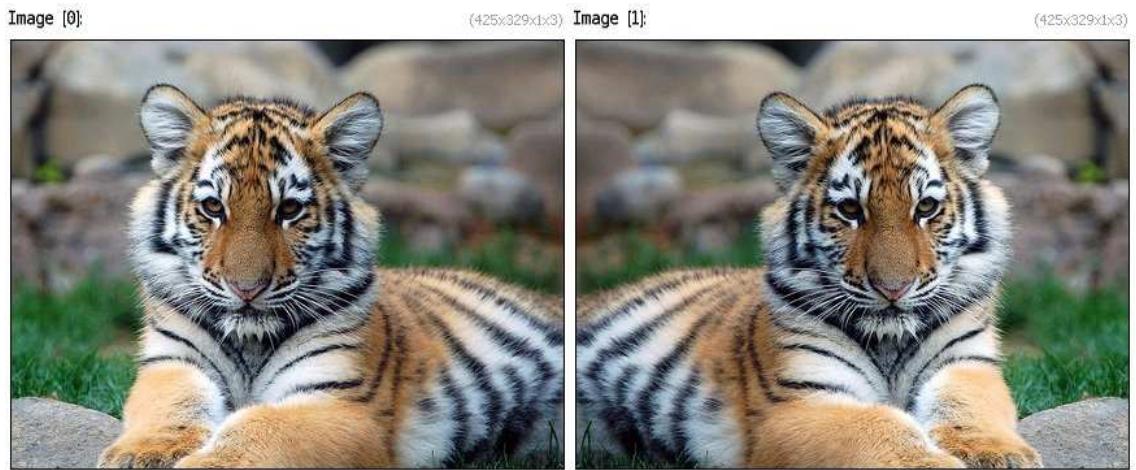


**Example 482 :** `image.jpg --local -repeat 3 -deform 20 -done -endlocal`

### 2.13.18 *-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.

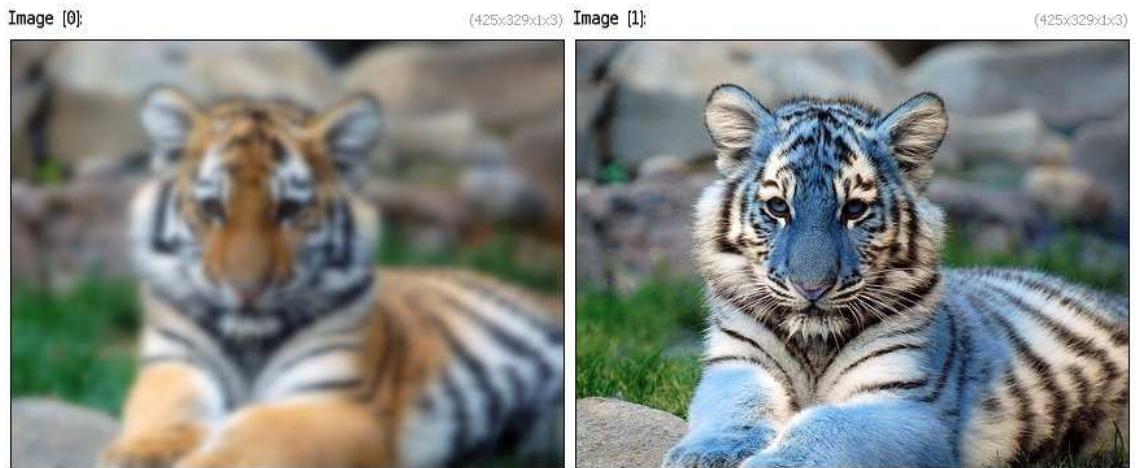


**Example 483:** `image.jpg --local -blur -3 -onfail -mirror x -endlocal`

### 2.13.19 *-parallel* (\*)

**Arguments:** "command1", "command2", ...

Execute specified commands in parallel, each in a different thread.  
All running threads share the current list of images.



**Example 484:** `image.jpg [0] -parallel "-blur[0] 3", "-mirror[1] c"`

### 2.13.20 *-progress* (\*)

**Arguments:** 0<=value<=100 |  
-1

Set the progress indice of the current processing pipeline.

This command is useful only when G'MIC is used by an embedding application.

### 2.13.21 **-quit** (\*)

Quit interpreter.

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

### 2.13.22 **-repeat** (\*)

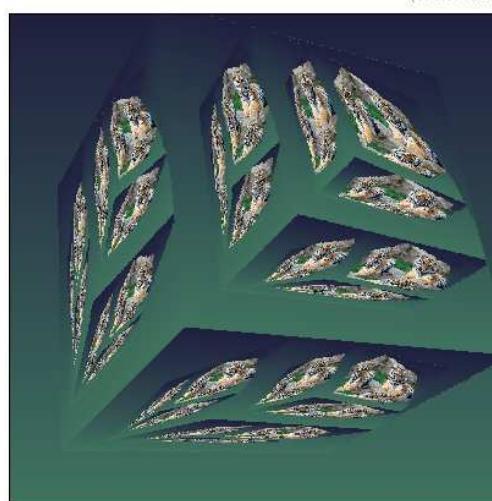
**Arguments:** nb\_iterations

Start iterations of a 'repeat..done' block.



**Example 485 :** image.jpg -split y -repeat 0% -shift[\$>] \$<,0,0,0,2 -done -append y

(400x400x1x3)



**Example 486:** `image.jpg -mode3d 2 -repeat 4 -imagecube3d -rotate3d 1,1,0,40 -snapshot3d 400,1.4 -done`

### 2.13.23 `-return` (\*)

Return from current custom command.

### 2.13.24 `-rprogress`

**Arguments:** `0<=value<=100 | -1 | "command", 0<=value_min<=100, 0<=value_max<=100`

Set the progress indice of the current processing pipeline (relatively to previously defined progress bounds), or call the specified command with specified progress bounds.

### 2.13.25 `-skip` (\*)

**Arguments:** `item`

Do nothing but skip specified item.

### 2.13.26 `-status` (\*)

**Arguments:** `value`

Set current status value. Used to define a returning value in a function.  
(*eq. to '-u'*).



**Example 487:** `image.jpg -command "foo : u0=Dark u1=Bright -status ${u{ia>=128}}" -text_outline @{-foo},2,2,24,2,1,255`

### 2.13.27 *-while* (\*)

**Arguments:** boolean |  
filename

End a 'do..while' block and go back to associated '-do' if specified boolean is true or if specified filename exists.

'boolean' can be a float number standing for { 0=false | other=true }.

## 2.14 Arrays, tiles and frames

### 2.14.1 *-array*

**Arguments:** M>0, \_N>0, \_expand\_type={ 0=min | 1=max | 2=all }

Create MxN array from selected images.

**Default values:** ' N=M' and ' expand\_type=0' .



**Example 488 :** image.jpg -array 3,2,2

### 2.14.2 *-array fade*

**Arguments:** M>0, \_N>0, 0<=\_fade\_start<=100, 0<=\_fade\_end<=100, \_expand\_type={ 0=min | 1=max | 2=all }

Create MxN array from selected images.

**Default values:** 'N=M', 'fade\_start=60', 'fade\_end=90' and 'expand\_type=1'.



**Example 489 :** image.jpg -array\_fade 3,2

### 2.14.3 *-array\_mirror*

**Arguments:** N>=0, \_dir={ 0=x | 1=y | 2=xy | 3=tri-xy }, \_expand\_type={ 0 | 1 }

Create 2^Nx2^N array from selected images.

**Default values:** 'dir=2' and 'expand\_type=0'.



**Example 490 :** image.jpg -array\_mirror 2

### 2.14.4 *-array\_random*

**Arguments:** `_Ms>0, _Ns>0, _Md>0, _Nd>0`

Create  $MdxNd$  array of tiles from selected  $MsxNs$  source arrays.

**Default values:** '`Ns=Ms'`, '`Md=Ms'` and '`Nd=Ns'`'.



**Example 491 :** `image.jpg --array_random 8,8,15,10`

### 2.14.5 *-frame\_blur*

**Arguments:** `_sharpness>0, _size>=0, _smoothness, _shading, _blur`

Draw RGBA-colored round frame in selected images.

**Default values:** '`sharpness=10'`', '`size=30'`', '`smoothness=0'`', '`shading=1'` and '`blur=3%`'.



**Example 492 :** image.jpg -frame.blur 3,30,8,10%

### 2.14.6 *-frame\_cube*

**Arguments:**                    $_{\text{depth}}>=0, _{\text{x_center}}, _{\text{y_center}}, _{\text{left\_side}}=\{0=\text{normal}$   
   |   $1=\text{mirror-x}$  |   $2=\text{mirror-y}$  |   $3=\text{mirror-xy}\}, _{\text{right\_side}}, _{\text{lower\_side}}, _{\text{upper\_side}}$

Insert 3d frames in selected images.

**Default       values:**                    $'\text{depth}=1'$ ,  $'\text{x_center}=\text{y_center}=0'$            and  
    $'\text{left\_side}=\text{right\_side}, \text{lower\_side}=\text{upper\_side}=0'$ .



**Example 493 :** image.jpg -frame\_cube ,

### 2.14.7 *-frame\_fuzzy*

**Arguments:**  $\text{size\_x}>=0, \text{size\_y}>=0, \text{fuzzyness}>=0, \text{smoothness}>=0, _{\text{R}}, _{\text{G}}, _{\text{B}}, _{\text{A}}$

Draw RGBA-colored fuzzy frame in selected images.

**Default       values:**                    $'\text{size\_y}=\text{size\_x}', '\text{fuzzyness}=5'$ ,  $'\text{smoothness}=1'$            and  
    $'\text{R}=\text{G}=\text{B}=\text{A}=255'$ .



**Example 494 :** image.jpg -frame\_fuzzy 20

#### 2.14.8 *-frame\_painting*

**Arguments:** `_size[%]>=0, 0<=_contrast<=1, _profile_smoothness[%]>=0, _R, _G, _B, _vignette_size[%]`

Add a painting frame to selected images.

**Default values:** `'size=10%', 'contrast=0.4', 'profile_smoothness=6%', 'R=225', 'G=200', 'B=120', 'vignette_size=2%', 'vignette_contrast=400', 'defects_contrast=50', 'defects_density=10', 'defects_size=1', 'defects_smoothness=0.5%' and 'serial_number=123456789'.`



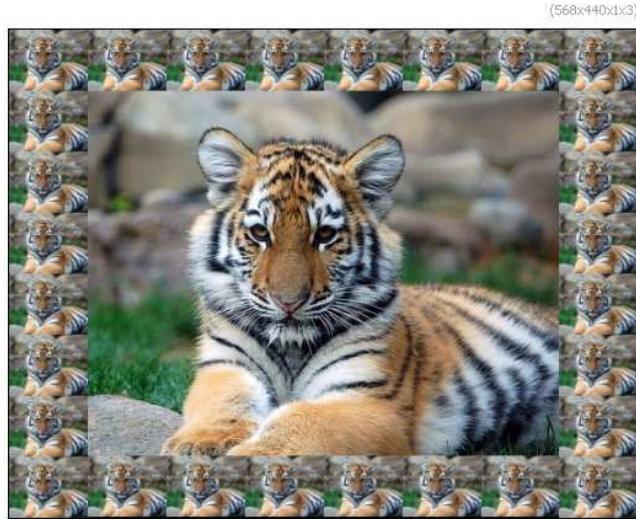
**Example 495 :** image.jpg -frame\_painting ,

### 2.14.9 *-frame\_pattern*

**Arguments:** `M>=3, -pattern = { 0=first image | 1=self }, -constrain_size = { 0 | 1 }`

Insert selected pattern frame in selected images.

**Default values:** '`pattern=0`' and '`constrain_size=0`'.



**Example 496 :** `image.jpg -frame_pattern 8`

### 2.14.10 *-frame\_round*

**Arguments:** `_sharpness>0, _size>=0, _smoothness, _shading, _R, _G, _B, _A`

Draw RGBA-colored round frame in selected images.

**Default values:** '`sharpness=10`', '`size=10`', '`smoothness=0`', '`shading=0`' and '`R=G=B=A=255`'.



**Example 497 :** image.jpg -frame\_round 10

#### 2.14.11 *-frame\_x*

**Arguments:** size\_x[%]>=0,\_col1,...,\_colN

Insert colored frame along the x-axis in selected images.

**Default values:** 'col1=col2=col3=255' and 'col4=255' .



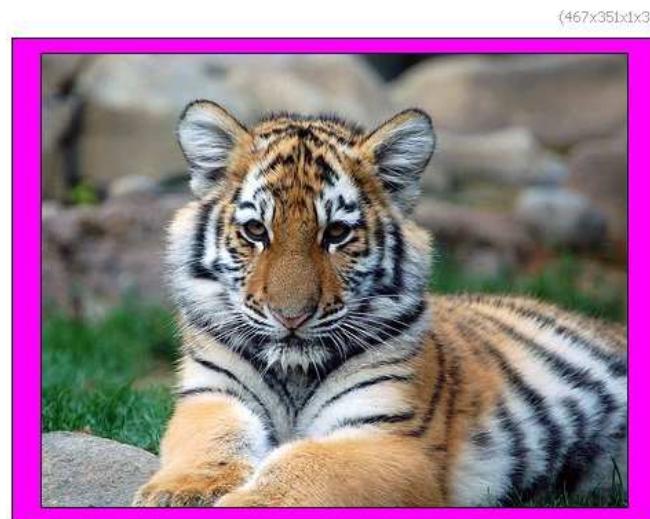
**Example 498 :** image.jpg -frame\_x 20,255,0,255

### 2.14.12 *-frame\_xy*

**Arguments:** `size_x[%]>=0, size_y[%]>=0, col1, ..., colN`

Insert colored frame along the x-axis in selected images.

**Default values:** '`size_y=size_x`', '`col1=col2=col3=255`' and '`col4=255`'.  
(eq. to '`-frame`').



**Example 499:** `image.jpg -frame_xy 1,1,0 -frame_xy 20,10,255,0,255`

### 2.14.13 *-frame\_xyz*

**Arguments:** `size_x[%]>=0, size_y[%]>=0, size_z[%]>=0, col1, ..., colN`

Insert colored frame along the x-axis in selected images.

**Default values:** '`size_y=size_x=size_z`', '`col1=col2=col3=255`' and '`col4=255`'.

### 2.14.14 *-frame\_y*

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

Insert colored frame along the y-axis in selected images.

**Default values:** '`col1=col2=col3=255`' and '`col4=255`'.



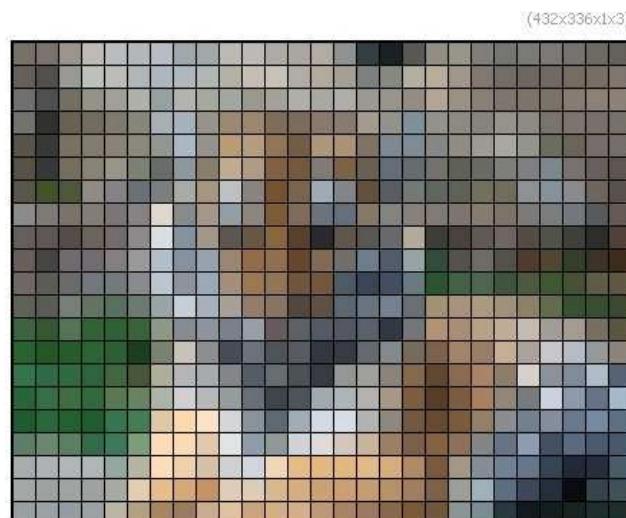
**Example 500 :** `image.jpg -frame_y 20,255,0,255`

#### 2.14.15 *-imagegrid*

**Arguments:**  $M > 0, N > 0$

Create  $M \times N$  image grid from selected images.

**Default value:** ' $N=M$ ' .



**Example 501 :** `image.jpg -imagegrid 16`

### 2.14.16 *-linearizetiles*

**Arguments:**  $M > 0, N > 0$

Linearize  $M \times N$  tiles on selected images.

**Default value:** ' $N=M$ ' .



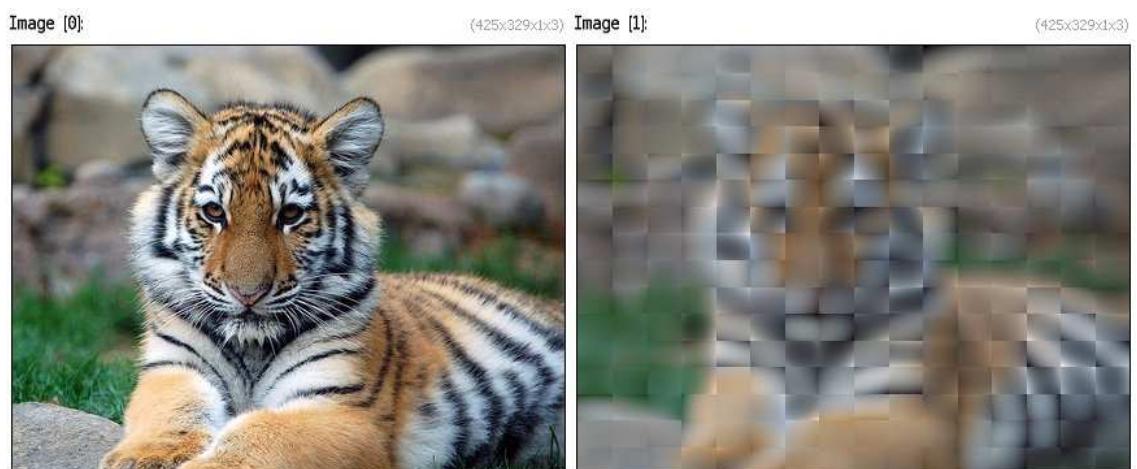
**Example 502 :** `image.jpg --linearizetiles 16`

### 2.14.17 *-quadratizetiles*

**Arguments:**  $M > 0, N > 0$

Quadratize  $M \times N$  tiles on selected images.

**Default value:** ' $N=M$ ' .



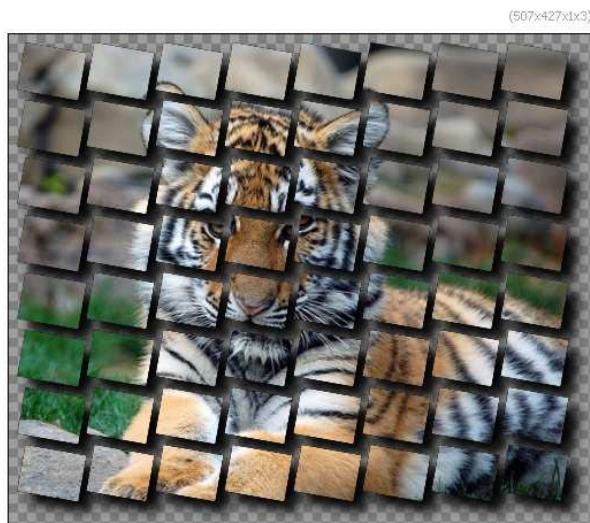
**Example 503 :** `image.jpg --quadratize_tiles 16`

### 2.14.18 *-rotate\_tiles*

**Arguments:** `angle, M>0, N>0`

Apply MxN tiled-rotation effect on selected images.

**Default values:** '`M=8`' and '`N=M`' .



**Example 504 :** `image.jpg -to_rgba -rotate_tiles 10,8 -drop_shadow 10,10 -display_rgba`

### 2.14.19 *-shift\_tiles*

**Arguments:** `M>0, N>0, amplitude`

Apply MxN tiled-shift effect on selected images.

**Default values:** '`N=M`' and '`amplitude=20`' .



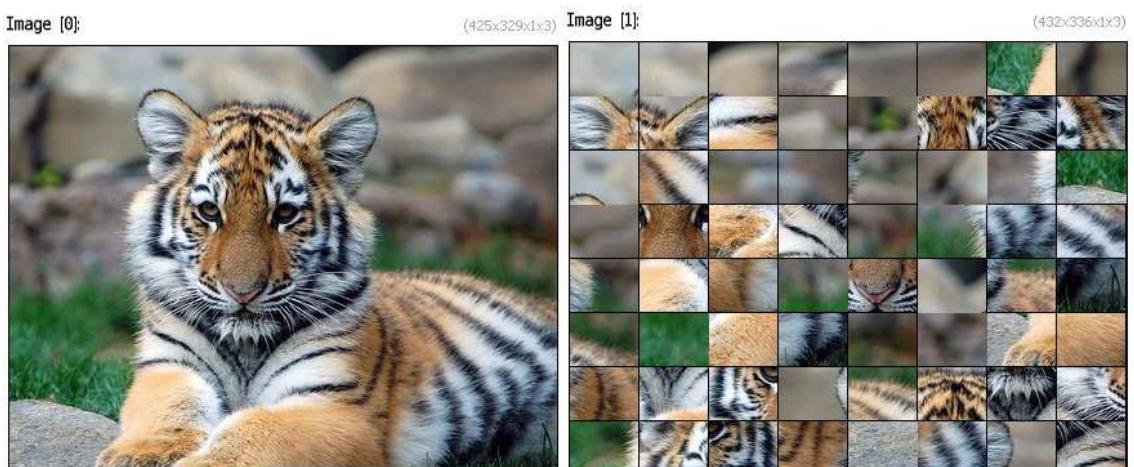
**Example 505 :** `image.jpg --shift_tiles 8,8,10`

#### 2.14.20 -taquin

**Arguments:** `M>0, N>0`

Create MxN taquin puzzle from selected images.

**Default value:** '`N=M`' .



**Example 506 :** `image.jpg --taquin 8`

#### 2.14.21 -tunnel

**Arguments:** `_level>=0, _factor>0, _cx, _cy, _opacity, _angle`

Apply tunnel effect on selected images.

**Default values:**    'level=9', 'factor=80%', 'cx=cy=0.5', 'opacity=1'    and  
   'angle=0'



**Example 507 :** image.jpg --tunnel 20

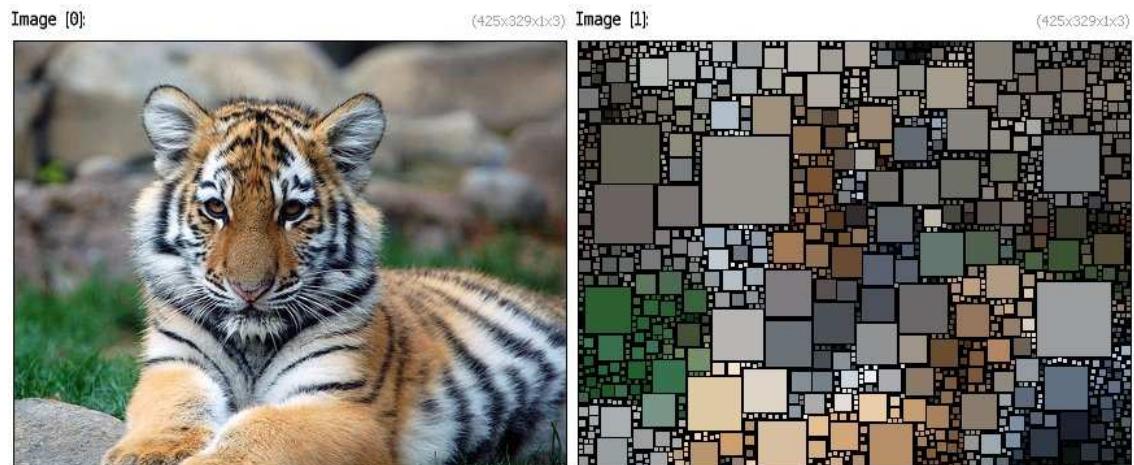
## 2.15 Artistic

### 2.15.1 *-boxfitting*

**Arguments:** `_min_box_size>=1, _max_box_size>=0, _initial_density>=0, _nb_attempts>=1`

Apply box fitting effect on selected images, as displayed the web page:  
[\[http://www.complexification.net/gallery/machines/boxFittingImg/\]](http://www.complexification.net/gallery/machines/boxFittingImg/)

**Default values:**    'min\_box\_size=1', 'max\_box\_size=0', 'initial\_density=0.1'  
   and 'nb\_tries=3'.



**Example 508 :** `image.jpg --boxfitting ,`

### 2.15.2 *-cartoon*

**Arguments:** `_smoothness, _sharpening, _threshold>=0, _thickness>=0, _color>=0, quantizat`

Apply cartoon effect on selected images.

**Default values:**     `'smoothness=3', 'sharpening=150', 'threshold=20',  
'thickness=0.25', 'color=1.5' and 'quantization=8'.`



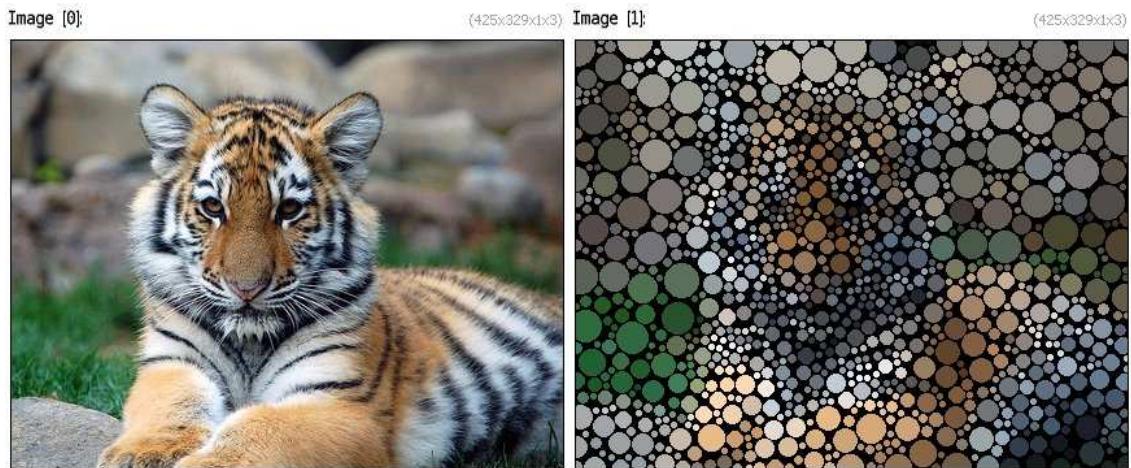
**Example 509 :** `image.jpg --cartoon 3,80,15`

### 2.15.3 *-circlism*

**Arguments:** `_radius_min>0, _radius_max>0, _smoothness[%]>=0, _radius_linearity>=0, _loc  
| 1=diamonds | 2=circle }`

Apply circlism effect on selected images (effect inspired by Ben Heine).

**Default values:**     `'radius_min=2', 'radius_max=20', 'smoothness=1',  
'radius_linearity=0.4', 'location_linearity=3' and 'shape=1'.`



**Example 510 :** image.jpg --circlism ,

#### 2.15.4 -color\_ellipses

**Arguments:** `_count>0, _radius>=0, _opacity>=0`

Add random color ellipses to selected images.

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



**Example 511 :** image.jpg --color\_ellipses ,, 0.15

#### 2.15.5 -cubism

**Arguments:** `_density>=0, 0<=_thickness<=50, _max_angle, _opacity, _smoothness>=0`

Apply cubism effect on selected images.

**Default values:** 'density=50', 'thickness=10', 'max\_angle=75', 'opacity=0.7' and 'smoothness=0'.



**Example 512 :** `image.jpg --cubism ,`

### 2.15.6 *-dotsbw*

**Arguments:** `_nb_scales>0, 0<=_resolution<=100, _radius_factor>=0`

Apply B&W dots effect on selected images.

**Default values:** 'nb\_scales=16', 'resolution=10' and 'radius\_factor=1'.



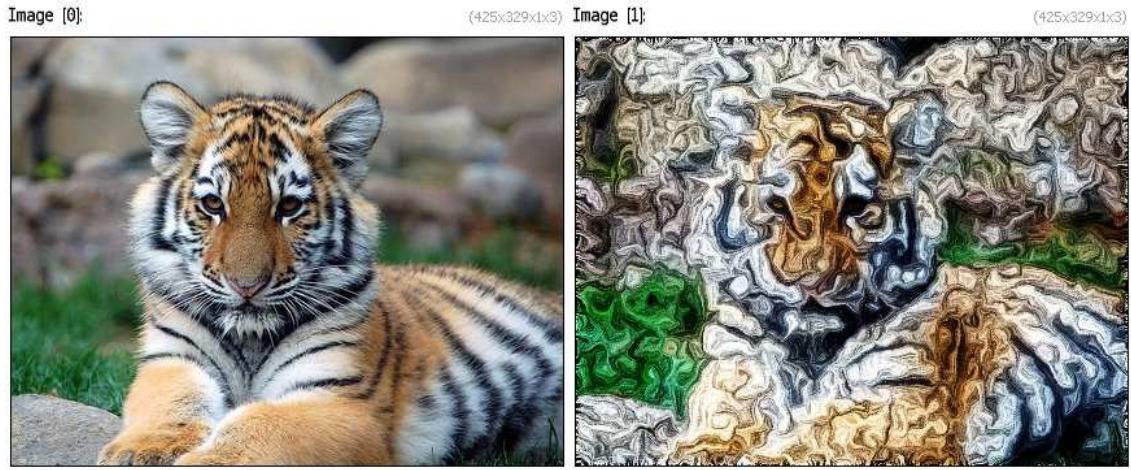
**Example 513 :** `image.jpg --dotsbw , --blend shapeaverage0`

### 2.15.7 *-draw\_whirl*

**Arguments:** `_amplitude>=0`

Apply whirl drawing effect on selected images.

**Default value:** ' amplitude=100' .



**Example 514 :** image.jpg --draw\_whirl ,

### 2.15.8 -drawing

**Arguments:** \_amplitude>=0

Apply drawing effect on selected images.

**Default value:** ' amplitude=200' .



**Example 515 :** image.jpg --drawing ,

### 2.15.9 -drop\_shadow

**Arguments:** \_offset\_x[%], \_offset\_y[%], \_smoothness[%]>=0, 0<=\_curvature<=1, \_expand\_size={

```
0 | 1 }
```

Drop shadow behind selected images.

**Default values:** 'offset\_x=20', 'offset\_y=offset\_x', 'smoothness=5', 'curvature=0' and 'expand\_size=1'.



**Example 516:** `image.jpg -drop_shadow 10,20,5,0.5 -expand_xy 20,0 -display_rgba`

### 2.15.10 -ellipsionism

**Arguments:** `_R>0 [%], _r>0 [%], _smoothness>=0 [%], _opacity, _outline>0, _density>0`

Apply ellipsionism filter to selected images.

**Default values:** 'R=10', 'r=3', 'smoothness=1%', 'opacity=0.7', 'outline=8' and 'density=0.6'.



**Example 517 :** `image.jpg --ellipsisism ,`

### 2.15.11 *-fire\_edges*

**Arguments:** `_edges>=0, 0<=_attenuation<=1, _smoothness>=0, _threshold>=0, _nb_frames>0, _starting`

Generate fire effect from edges of selected images.

**Default values:** `'edges=0.7', 'attenuation=0.25', 'smoothness=0.5', 'threshold=25', 'nb_frames=1', 'starting_frame=20' and 'frame_skip=0'.`



**Example 518 :** `image.jpg -fire_edges ,`

### 2.15.12 *-glow*

**Arguments:** `_amplitude>=0`

Add soft glow on selected images.

**Default value:** ' amplitude=1% ' .



**Example 519 :** image.jpg --glow ,

### 2.15.13 -halftone

**Arguments:** nb\_levels>=2, \_size\_dark>=2, \_size\_bright>=2, \_shape={0=square | 1=diamond | 2=circle | 3=inv-square | 4=inv-diamond | 5=inv-circle }, \_smoothness[%]>=0

Apply halftone dithering to selected images.

**Default values:** ' nb\_levels=5 ', ' size\_dark=8 ', ' size\_bright=8 ', ' shape=5 ' and ' smoothness=0 ' .



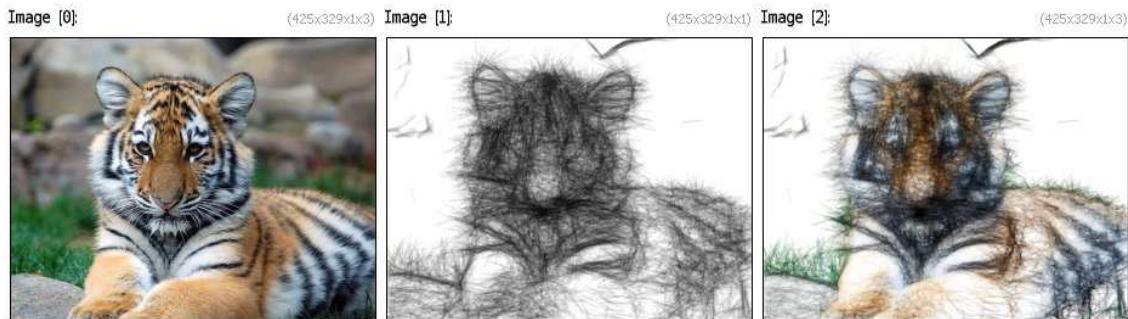
**Example 520 :** `image.jpg --halftone ,`

### 2.15.14 *-hardsketchbw*

**Arguments:** `_amplitude>=0, _density>=0, _opacity, 0<=_edge_threshold<=100, _is_fast={0 | 1 }`

Apply hard B&W sketch effect on selected images.

**Default values:** `'amplitude=1000', 'sampling=3', 'opacity=0.1', 'edge_threshold=20' and 'is_fast=0'.`



**Example 521 :** `image.jpg --hardsketchbw 200,70,0.1,10 -median[-1] 2 --local -reverse -blur[-1] 3 -blend overlay -endl`

### 2.15.15 *-hearts*

**Arguments:** `_density>=0`

Apply heart effect on selected images.

**Default value:** `'density=10'`.

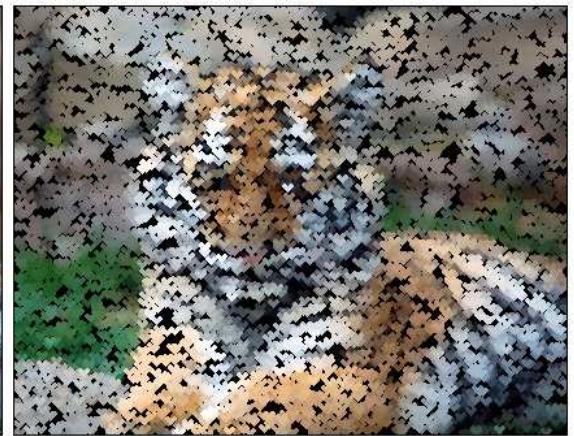
Image [0]:

(425x329x1x3)



Image [1]:

(425x329x1x3)



**Example 522 :** image.jpg --hearts ,

### 2.15.16 -houghsketchbw

**Arguments:** `_density>=0, _radius>0, 0<=_threshold<=100, 0<=_opacity<=1, _votesize[%]>0`

Apply hough B&W sketch effect on selected images.

**Default values:** `'density=8', 'radius=5', 'threshold=80', 'opacity=0.1'` and `'votesize=100%'`.

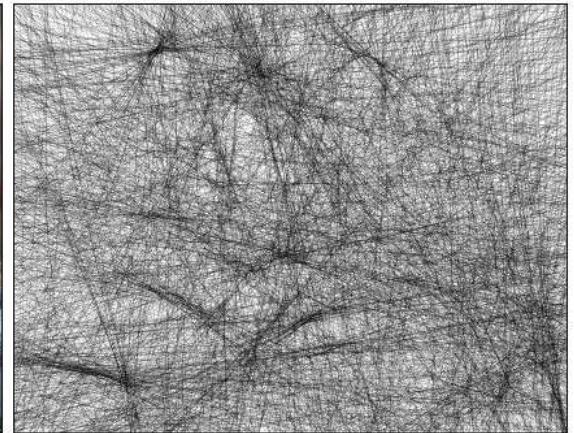
Image [0]:

(425x329x1x3)



Image [1]:

(425x329x1x1)



**Example 523 :** image.jpg --houghsketchbw ,

### 2.15.17 -lightrays

**Arguments:** `100<=_density<=0, _cx, _cy, _ray_length>=0, _ray_attenuation>=0`

Generate ray lights from the edges of selected images.

Defaults values : 'density=50%', 'cx=0.5', 'cy=0.5', 'ray\_length=0.9' and 'ray\_attenuation=0.5'.



**Example 524:** image.jpg --lightrays , -+ -c 0,255

### 2.15.18 *-light\_relief*

**Arguments:** `-ambient_light`, `-specular_lightness`, `-specular_size`, `-light_smoothness`, `-darkness`, `-xl`,  
0 | 1 }

Apply relief light to selected images.

**Default values(s):** 'ambient\_light=0.3', 'specular\_lightness=0.5',  
'specular\_size=0.2', 'darkness=0', 'xl=0.2', 'yl=zl=0.5',  
'zscale=1', 'opacity=1' and 'opacity\_bumpmap=0'.



**Example 525 :** `image.jpg --blur 2 -light_relief[-1] 0.3,4,0.1,0`

### 2.15.19 *-mosaic*

**Arguments:** `_density>=0, _edges={ 0 | 1 }`

Create random mosaic from selected images.

**Default values:** '`density=0.8`' and '`edges=1`'.



**Example 526 :** `image.jpg --mosaic ,`

### 2.15.20 *-old\_photo*

Apply old photo effect on selected images.



**Example 527 :** `image.jpg --old_photo`

**2.15.21 *-pencilbw***

**Arguments:** `_size>=0, _amplitude>=0`

Apply B&W pencil effect on selected images.

**Default values:** '`size=0.3`' and '`amplitude=60`' .



**Example 528 :** `image.jpg --pencilbw ,`

**2.15.22 *-polaroid***

**Arguments:** `_size1>=0, _size2>=0`

Create polaroid effect in selected images.

**Default values:** '`size1=10`' and '`size2=20`' .



**Example 529 :** `image.jpg -to_rgba -polaroid 5,30 -rotate 20 -drop_shadow , -display_rgba`

### 2.15.23 *-poster\_edges*

**Arguments:** `0<=edge_threshold<=100, 0<=edge_shade<=100, edge_thickness>=0, edge_antialiasing>=0`

Apply poster edges effect on selected images.

**Default values:** `'edge_threshold=40', 'edge_shade=5', 'edge_thickness=0.5', 'edge_antialiasing=10', 'posterization_level=12'` and `'posterization_antialiasing=0'`.



**Example 530 :** `image.jpg --poster_edges ,`

**2.15.24 -rodilius**

**Arguments:** `0<=_amplitude<=100, _0<=thickness<=100, _sharpness>=0, _nb_orientations>0, _offset  
0=darker | 1=brighter }`

Apply rodilius (fractalius-like) filter on selected images.

**Default values:** `'amplitude=10', 'thickness=10', 'sharpness=400',  
'nb_orientations=7', 'offset=0' and 'color_mode=1'.`



**Example 531 :** `image.jpg --rodilius 12,10,300,10 -normalize_local[-1] 10,6`

**2.15.25 -stained\_glass**

**Arguments:** `_edges [%]>=0, shading>=0, is_thin_separators={ 0 | 1 }`

Generate stained glass from selected images.

**Default values:** `'edges=40%', 'shading=0.2' and 'is_precise=0'`.



**Example 532 :** `image.jpg --stained-glass ,`

### 2.15.26 -stars

**Arguments:** `_density[%]>=0, _depth>=0, _size>0, _nb_branches>=1, 0<=_thickness<=1, _smoothness=0.5, _R=G=B=200 and _opacity=1`

Add random stars to selected images.

**Default values:** `'density=10%', 'depth=1', 'size=32', 'nb_branches=5', 'thickness=0.38', 'smoothness=0.5', 'R=G=B=200' and 'opacity=1'.`



**Example 533 :** `image.jpg -stars ,`

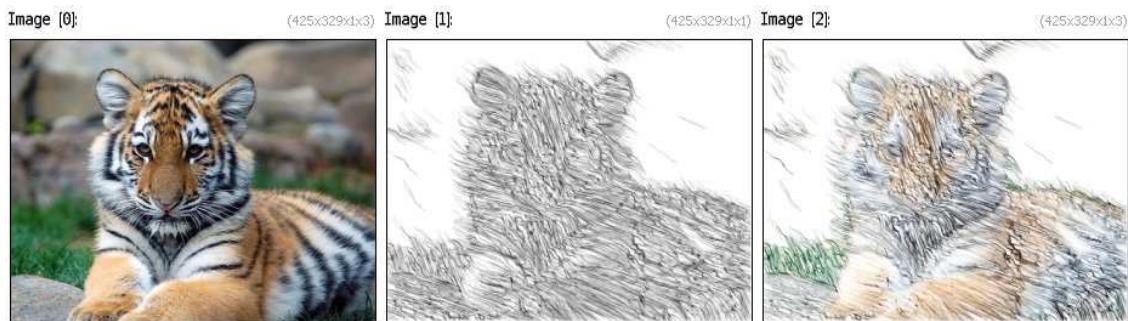
### 2.15.27 -sketchbw

`_nb_orient>0, _start_angle, _angle_range>=0, _length>=0, _threshold>=0, _opacity, _bgfactor>=0, _density>0, _sharpness>=0, _anisotropy>=0, _smoothness>=0, _coherence>=0, _is_borders>0`

```
0 | 1 },'is_curved={ 0 | 1 }\n
```

Apply sketch effect to selected images.

**Default values:** 'nb\_orientents=2', 'start\_angle=45', 'angle\_range=180', 'length=30', 'threshold=1', 'opacity=0.03', 'bgfactor=0', 'density=0.6', 'sharpness=0.1', 'anisotropy=0.6', 'smoothness=0.25', 'coherence=1', 'is\_boost=0' and 'is\_curved=1'.



**Example 534:** image.jpg --sketchbw 1 --local -reverse -blur[-1] 3 -blend overlay -endl

### 2.15.28 -sponge

**Arguments:** \_size>0

Apply sponge effect on selected images.

**Default value:** 'size=13' .



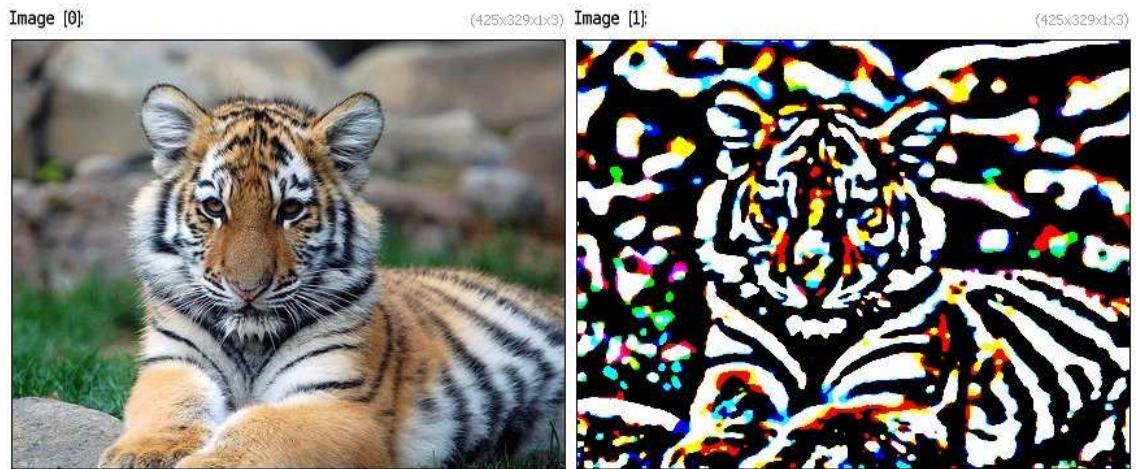
**Example 535:** image.jpg --sponge ,

### 2.15.29 *-stencil*

**Arguments:** `_radius[%]>=0, _smoothness>=0, _iterations>=0`

Apply stencil filter on selected images.

**Default values:** '`radius=3`', '`smoothness=1`' and '`iterations=8`'.



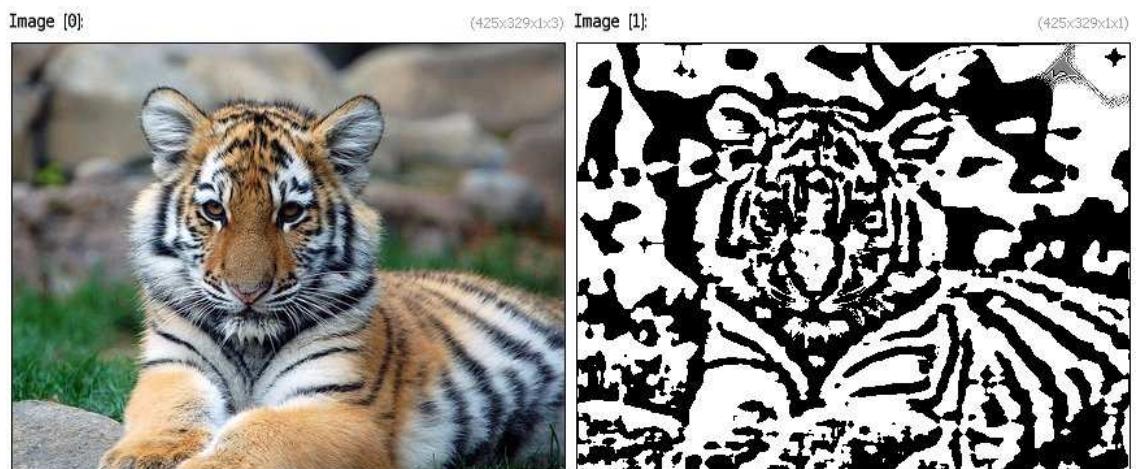
**Example 536 :** `image.jpg --stencil 1,10,3`

### 2.15.30 *-stencilbw*

**Arguments:** `_edges>=0, _smoothness>=0`

Apply B&W stencil effect on selected images.

**Default values:** '`edges=15`' and '`smoothness=10`'.



**Example 537 :** image.jpg --stencilbw 40,4

### 2.15.31 -*tetris*

**Arguments:** \_scale>0

Apply tetris effect on selected images.

**Default value:** 'scale=10'.



**Example 538 :** image.jpg --tetris 10

### 2.15.32 -*warhol*

**Arguments:** \_M>0, \_N>0, \_smoothness>=0, \_color>=0

Create MxN Andy Warhol-like artwork from selected images.

**Default values:** 'M=3', 'N=M', 'smoothness=2' and 'color=20'.



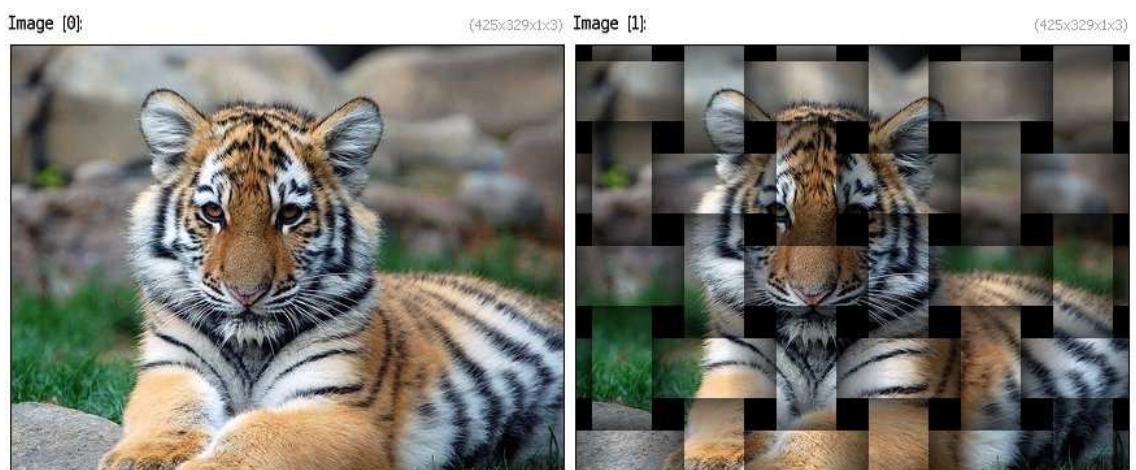
**Example 539 :** `image.jpg --warhol 5,3,3,40`

### 2.15.33 -weave

**Arguments:** `-density>=0, 0<=_thickness<=100, 0<=_shadow<=100, -shading>=0, -fibers_amplitude>=0, -fibers_smoothness>=0, -angle>=0, -curvature_x=>0, -curvature_y=>0`

Apply weave effect to the selected images.  
`'angle'` can be { 0=0 | 1=22.5 | 2=45 | 3=67.5 }.

**Default values:** `'density=6', 'thickness=65', 'shadow=40', 'shading=0.5', 'fibers_amplitude=0', 'fibers_smoothness=0', 'angle=0' and 'curvature_x=curvature_y=0'`



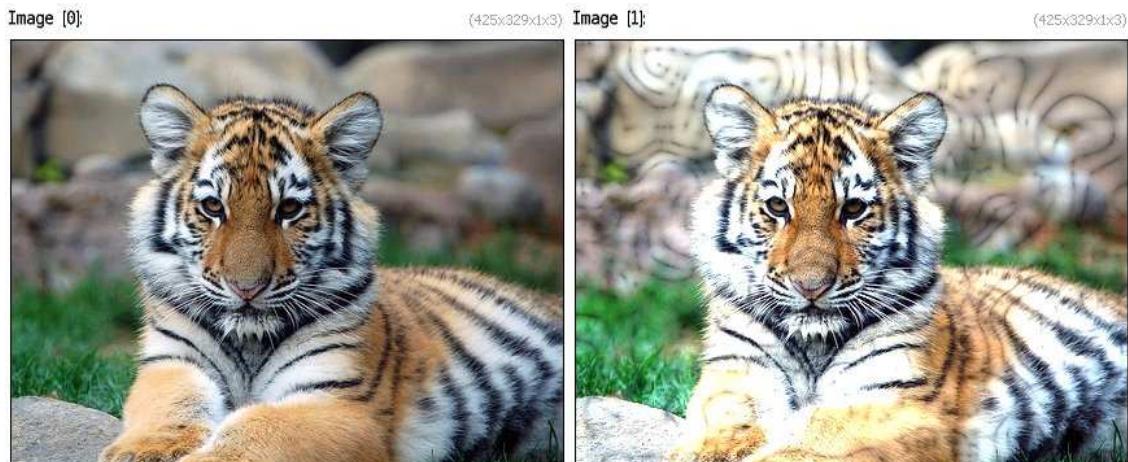
**Example 540 :** `image.jpg --weave ,`

### 2.15.34 *-whirls*

**Arguments:** `_texture>=0, _smoothness>=0, _darkness>=0, _lightness>=0`

Add random whirl texture to selected images.

**Default values:** `'texture=3', 'smoothness=6', 'darkness=0.5' and 'lightness=1.8'`.



**Example 541:** `image.jpg --whirls ,`

## 2.16 Warpings

### 2.16.1 *-euclidean2polar*

**Arguments:** `_cx, _cy, _n>0, _boundary={ 0=dirichlet | 1=neumann | 2=cyclic }`

Apply euclidean to polar transform on selected images.

**Default values:** `'cx=cy=0.5', 'n=1' and 'boundary=1'`.



**Example 542 :** `image.jpg --euclidean2polar ,`

### 2.16.2 *-deform*

**Arguments:** `_amplitude>=0`

Apply random smooth deformation on selected images.

**Default value:** `' amplitude=10 '`.



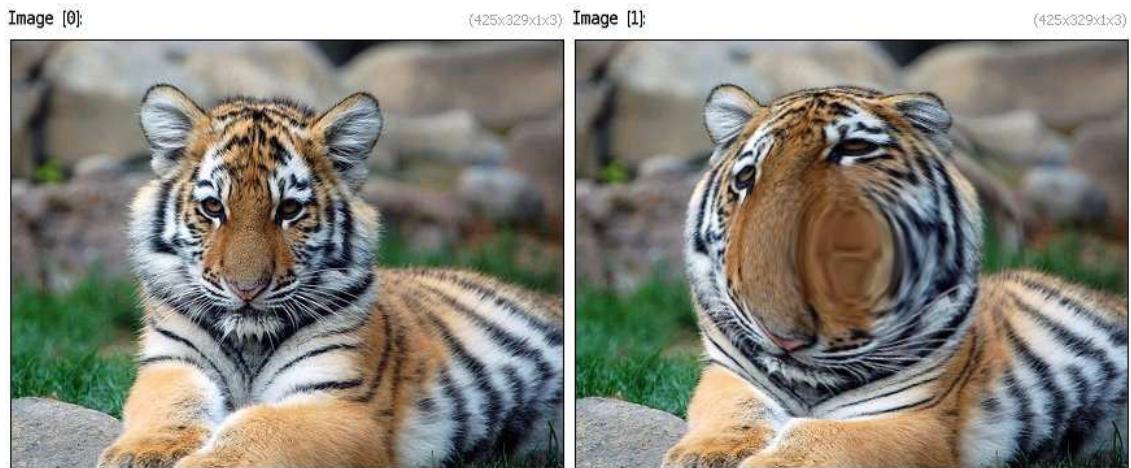
**Example 543 :** `image.jpg --deform[0] 10 --deform[0] 20`

### 2.16.3 *-fisheye*

**Arguments:** `_x, _y, 0<=_radius<=100, _amplitude>=0`

Apply fish-eye deformation on selected images.

**Default values:** `' x=y=50 ', ' radius=50 ' and ' amplitude=1.2 '`.



**Example 544:** image.jpg --fisheye ,

#### 2.16.4 -flower

**Arguments:** `-amplitude, -frequency, -offset_r[%], -angle, -cx, -cy, -boundary={0=dirichlet | 1=neumann | 2=cyclic }`

Apply flower deformation on selected images.

**Default values:** `'amplitude=30', 'frequency=6', 'offset_r=0', 'angle=0', 'cx=cy=0.5'` and `'boundary=2'`.



**Example 545:** image.jpg -flower ,

### 2.16.5 *-kaleidoscope*

**Arguments:** `_cx, _cy, _radius, _angle, boundary={ 0=dirichlet | 1=neumann | 2=cyclic }`

Create kaleidoscope effect from selected images.

**Default values:** '`cx=cy=0.5'`, '`radius=100'`, '`angle=30'` and '`boundary=1'`.



**Example 546 :** `image.jpg --kaleidoscope ,`

### 2.16.6 *-map\_sphere*

**Arguments:** `_width>0, _height>0, _radius, _dilation>0, _fading>=0, _fading_power>=0`

Map selected images on a sphere.

**Default values:** '`width=height=512'`, '`radius=100'`, '`dilation=0.5'`, '`fading=0'` and '`fading_power=0.5'`.



**Example 547:** `image.jpg --map_sphere ,`

### 2.16.7 -polar2euclidean

**Arguments:** `-cx, -cy, -n>0, -boundary={ 0=dirichlet | 1=neumann | 2=cyclic }`

Apply polar to euclidean transform on selected images.

**Default values:** '`cx=cy=0.5`', '`n=1`' and '`boundary=1`' .



**Example 548:** `image.jpg --euclidean2polar , -mirror[-1] x -polar2euclidean[-1] ,`

### 2.16.8 -raindrops

**Arguments:** `-amplitude, -density>=0, -wavelength>=0, -merging_steps>=0`

Apply raindrops deformation on selected images.

**Default values:** 'amplitude=80', 'density=0.1', 'wavelength=1' and 'merging\_steps=0'.



**Example 549 :** image.jpg --raindrops ,

### 2.16.9 -ripple

**Arguments:** \_amplitude, \_frequency, \_shape={ 0=bloc | 1=triangle | 2=sine | 3=sine+ | 4=random }, \_angle, \_offset

Apply ripple deformation on selected images.

**Default values:** 'amplitude=10', 'frequency=10', 'shape=2', 'angle=0' and 'offset=0'.



**Example 550 :** `image.jpg --ripple ,`

### 2.16.10 *-rotoidoscope*

**Arguments:** `-cx,-cy,-tiles>0,-smoothness[%]>=0,-boundary={ 0=dirichlet | 1=neumann | 2=cyclic }`

Create rotational kaleidoscope effect from selected images.

**Default values:** '`cx=cy=50%`', '`tiles=10`', '`smoothness=1`' and '`boundary=1`' .



**Example 551 :** `image.jpg --rotoidoscope ,`

### 2.16.11 *-symmetrize*

**Arguments:** `-x[%],-y[%],-angle,-boundary={ 0=dirichlet | 1=neumann | 2=cyclic },-is_antisymmetry={ 0 | 1 },-swap_sides={ 0 | 1 }`

Symmetrize selected image regarding specified axis.

**Default values:** '`x=y=50%`', '`angle=90`', '`boundary=1`', '`is_antisymmetry=0`' and '`swap_sides=0`' .



**Example 552 :** `image.jpg --symmetrize 50%,50%,45 --symmetrize[-1] 50%,50%,-45`

### 2.16.12 *-transform polar*

**Arguments:** `"expr_radius", "expr_angle", _x_center, _y_center, _boundary={0=dirichlet | 1=neumann }`

Apply user-defined transform on polar representation of selected images.

**Default values:** `'expr_radius=R-r', 'expr_angle=a', 'x_center=y_center=50' and 'boundary=1'.`



**Example 553 :** `image.jpg --transform_polar[0] R*(r/R)^2,a --transform_polar[0] r,2*a`

### 2.16.13 *-twirl*

**Arguments:** `_amplitude, _cx, _cy, _boundary={ 0=dirichlet | 1=neumann | 2=cyclic }`

Apply twirl deformation on selected images.

**Default values:** `'amplitude=1', 'cx=cy=0.5' and 'boundary=1'.`



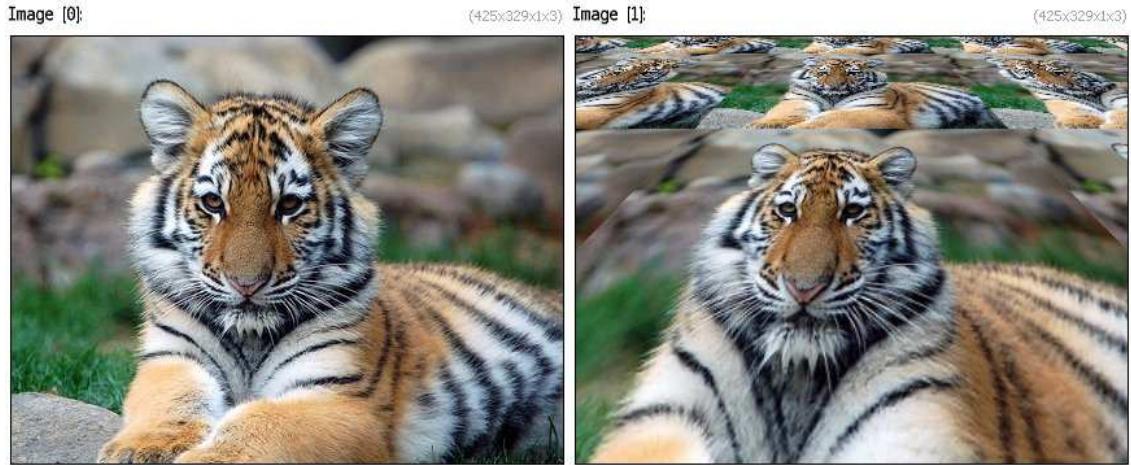
**Example 554 :** image.jpg --twirl 0.6

### 2.16.14 -warp\_perspective

**Arguments:** `-x-angle, -y-angle, -zoom>0, -x-center, -y-center, -boundary={0=dirichlet | 1=neumann | 2=cyclic }`

Warp selected images with perspective deformation.

**Default values:** `'x-angle=1.5', 'y-angle=0', 'zoom=1', 'x-center=y-center=50' and 'boundary=2'.`



**Example 555 :** image.jpg --warp-perspective ,

### 2.16.15 -water

**Arguments:** `_amplitude>=0, _smoothness>=0`

Apply water deformation on selected images.

**Default values:** ' amplitude=30' and ' smoothness=1.5' .



**Example 556 :** image.jpg --water ,

### 2.16.16 -wave

**Arguments:** \_amplitude>=0, \_frequency>=0, \_center\_x, \_center\_y

Apply wave deformation on selected images.

**Default values:** ' amplitude=4' , ' frequency=0.4' and ' center\_x=center\_y=50' .



**Example 557 :** image.jpg --wave ,

### 2.16.17 -wind

**Arguments:** `-amplitude>=0, -angle, 0<=attenuation<=1, -threshold`

Apply wind effect on selected images.

**Default values:** `'amplitude=20', 'angle=0', 'attenuation=0.7' and 'threshold=20'.`



**Example 558 :** `image.jpg --wind ,`

### 2.16.18 -zoom

**Arguments:** `-factor, -cx, -cy, -cz, -boundary={ 0=dirichlet | 1=neumann | 2=cyclic }`

Apply zoom factor to selected images.

**Default values:** `'factor=1', 'cx=cy=cz=0.5'` and `'boundary=0'`.



**Example 559 :** `image.jpg --zoom[0] 0.6 --zoom[0] 1.5`

## 2.17 Degradations

### 2.17.1 *-cracks*

**Arguments:** `_density>=0, _amplitude, _relief={ 0 | 1 }`

Add random cracks to selected images.

**Default values:** `'density=0.2', 'amplitude=40'` and `'relief=0'`.



**Example 560 :** `image.jpg --cracks 0.2,60,1`

### 2.17.2 *-light patch*

**Arguments:** `_density>0, _darkness>=0, _lightness>=0`

Add light patches to selected images.

**Default values:** `'density=10', 'darkness=0.9'` and `'lightness=1.7'`.



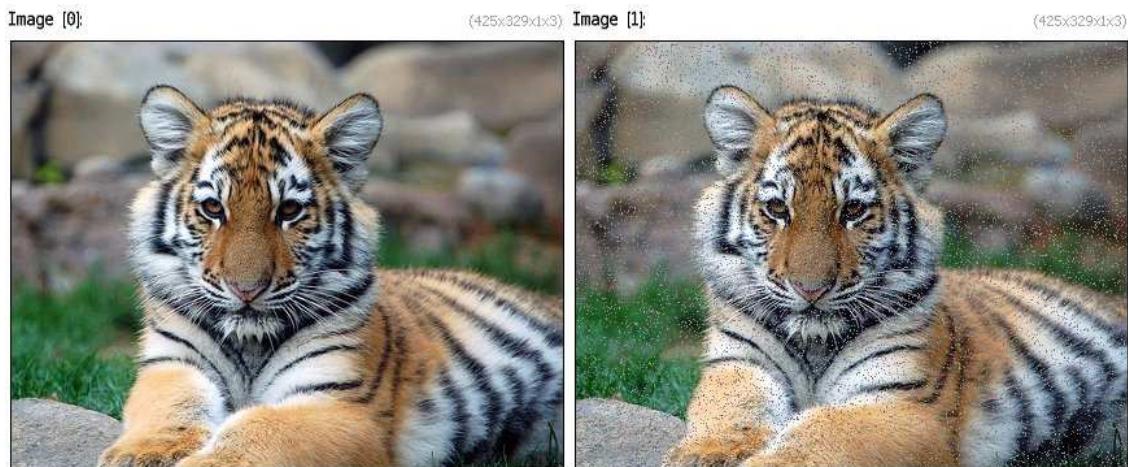
**Example 561 :** `image.jpg --light_patch 20,0.9,4`

### 2.17.3 -noise\_hurl

**Arguments:** `_amplitude>=0`

Add hurl noise to selected images.

**Default value:** `'amplitude=10'`.



**Example 562 :** `image.jpg --noise_hurl ,`

### 2.17.4 -pixelize

**Arguments:** `_scale_x>0, _scale_y>0, _scale_z>0`

Pixelize selected images with specified scales.

**Default values:** 'scale\_x=20' and 'scale\_y=scale\_z=scale\_x' .



**Example 563:** image.jpg --pixelize ,

### 2.17.5 *-shade\_stripes*

**Arguments:** `-frequency>=0, -direction={ 0=horizontal | 1=vertical }`  
`, -darkness>=0, -lightness>=0`

Add shade stripes to selected images.

**Default values:** 'frequency=5', 'direction=1', 'darkness=0.8' and  
'lightness=2' .



**Example 564 :** image.jpg --shade\_stripes 30

### 2.17.6 *-shadow\_patch*

**Arguments:** `_opacity>=0`

Add shadow patches to selected images.

**Default value:** `' opacity=0.7'`.



**Example 565 :** `image.jpg --shadow_patch 0.4`

### 2.17.7 *-spread*

**Arguments:** `_dx>=0, _dy>=0, _dz>=0`

Spread pixel values of selected images randomly along x,y and z.

**Default values:** `' dx=3'`, `' dy=dx'` and `' dz=0'`.



**Example 566 :** `image.jpg --spread 3`

### 2.17.8 *-stripes\_y*

**Arguments:** `_frequency>=0`

Add vertical stripes to selected images.

**Default value:** `'frequency=10'`.



**Example 567 :** `image.jpg --stripes_y ,`

### 2.17.9 *-texturize\_canvas*

**Arguments:** `_amplitude>=0, _fibrousness>=0, _emboss_level>=0`

Add paint canvas texture to selected images.

**Default values:** `'amplitude=20', 'fibrousness=3' and 'emboss_level=0.6'`.



**Example 568 :** `image.jpg --texturize_canvas ,`

### 2.17.10 *-texturize\_paper*

Add paper texture to selected images.



**Example 569 :** `image.jpg --texturize_paper`

### 2.17.11 *-vignette*

**Arguments:** `_strength>=0, 0<=_radius_min<=100, 0<=_radius_max<=100`

Add vignette effect to selected images.

**Default values:** '`strength=100`', '`radius_min=70`' and '`radius_max=90`'.



**Example 570:** `image.jpg --vignette ,`

### 2.17.12 *-watermark\_visible*

**Arguments:** `_text, 0<_opacity<1, _size>0, _angle, _mode={ 0=remove | 1=add }, _smoothness>=0`

Add or remove a visible watermark on selected images (value range must be [0,255]).

**Default values:** `'text=(C) GMIC', 'opacity=0.3', 'size=57', 'angle=25', 'mode=1' and 'smoothness=0'.`



**Example 571 :** `image.jpg --watermark_visible , 0.7`

## 2.18 Blending and fading

### 2.18.1 *-blend*

**Arguments:** `blending_mode, 0<=opacity<=1, _revert_layer_order={ 0 | 1 }`

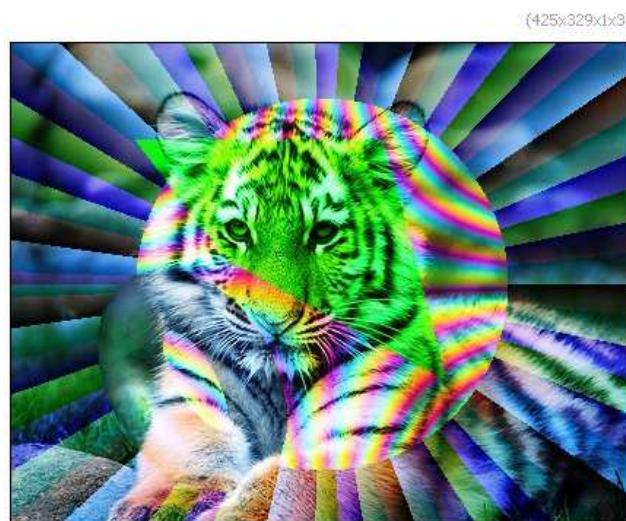
Blend selected G,GA,RGB or RGBA images, two-by-two, using specified mode.

'blending\_mode' can be { add | alpha | and | average | blue | burn | darken | difference | divide | dodge | exclusion | freeze | grainextract | grainmerge | green | hardlight | hardmix | hue | interpolation | lighten | lightness | linearburn | linearlight | luminance | multiply | negation | or | overlay | pinlight | red | reflect | saturation | screen | shapeaverage | shapeaverage0 | softburn | softdodge | softlight | stamp | subtract | value | vividlight | xor }.

**Default values:** 'blending\_mode=alpha', 'opacity=1' and 'revert\_layers=0'.



**Example 572:** `image.jpg --drop_shadow , -resize2dy[-1] 200 -rotate[-1] 20 --blend alpha -drgba[-2]`



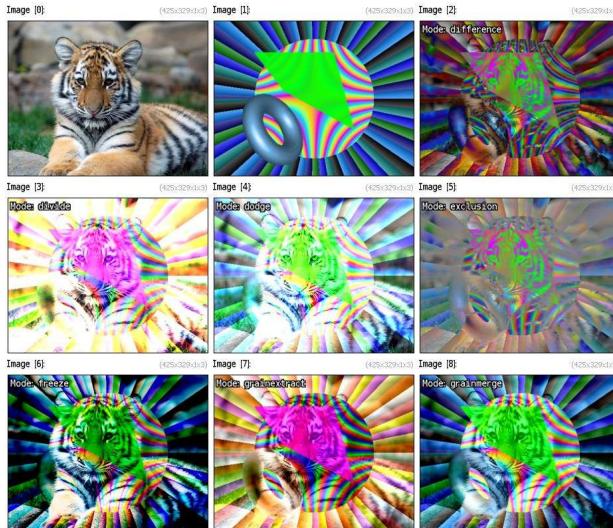
**Example 573 :** `image.jpg -testimage2d {w},{h} -blend overlay`



```
Example 574 : -m "ex : $""=arg -repeat $""% --blend[0,1] ${arg{$>+1}}
```

```
-text_outline[-1] Mode:\\" \"$${arg{$>+1}},2,2,24,2,1,255 -done" image.jpg
```

```
-testimage2d {w}, {h} -ex add,alpha,and,average,blue,burn,darken
```



```
Example 575 : -m "ex : $""=arg -repeat $""% --blend[0,1] ${arg{$>+1}}
```

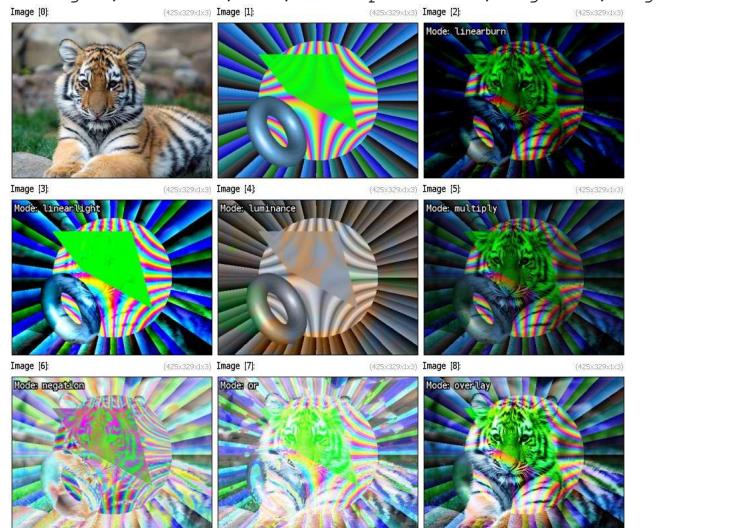
```
-text_outline[-1] Mode:\\" \"$${arg{$>+1}},2,2,24,2,1,255 -done" image.jpg
```

```
-testimage2d {w}, {h} -ex
```

```
difference,divide,dodge,exclusion,freeze,grainextract,grainmerge
```



```
Example 576 : -m "ex : $""=arg -repeat $"% --blend[0,1] ${arg{$>+1}}
-text_outline[-1] Mode:\\" \"$${arg{$>+1}},2,2,24,2,1,255 -done" image.jpg
-testimage2d {w},{h} -ex
green,hardlight,hardmix,hue,interpolation,lighten,lightness
```

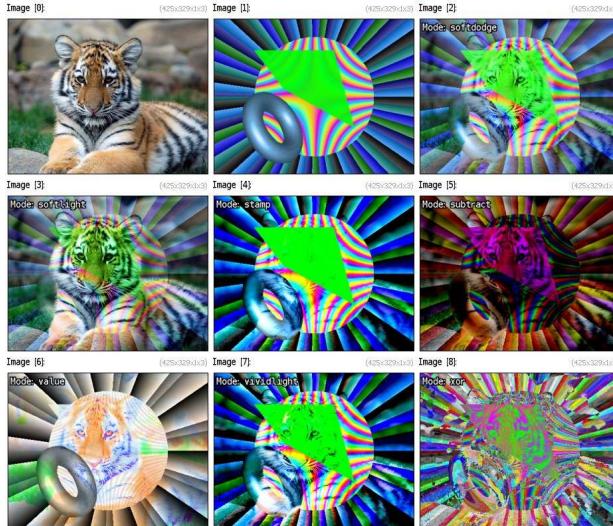


```
Example 577 : -m "ex : $""=arg -repeat $"% --blend[0,1] ${arg{$>+1}}
-text_outline[-1] Mode:\\" \"$${arg{$>+1}},2,2,24,2,1,255 -done" image.jpg
-testimage2d {w},{h} -ex
linearburn,linearlight,luminance,multiply,negation,or,overlay
```



```
Example 578 : -m "ex : $""=arg -repeat $""% --blend[0,1] ${arg{$>+1}}
-text_outline[-1] Mode:\\" \"$\{arg{$>+1}\},2,2,24,2,1,255 -done" image.jpg
-testimage2d {w},{h} -ex
```

pinlight,red,reflect,saturation,screen,shapeaverage,softburn



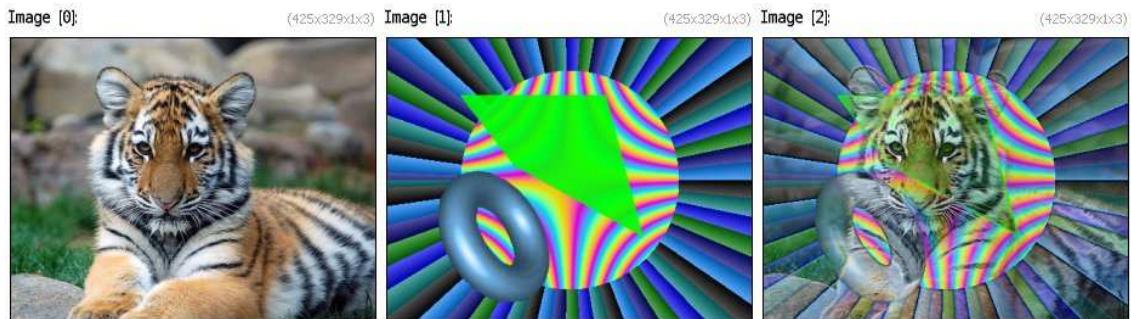
```
Example 579 : -m "ex : $""=arg -repeat $""% --blend[0,1] ${arg{$>+1}}
-text_outline[-1] Mode:\\" \"$\{arg{$>+1}\},2,2,24,2,1,255 -done" image.jpg
-testimage2d {w},{h} -ex
```

softdodge,softlight,stamp,subtract,value,vividlight,xor

## 2.18.2 *-blend\_edges*

**Arguments:** smoothness [%] >=0

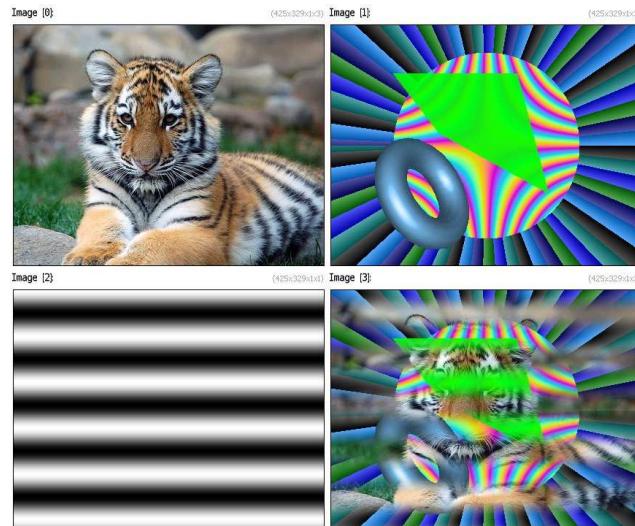
Blend selected images together using 'edges' mode.



```
Example 580 : image.jpg -testimage2d {w},{h} --blend_edges 0.8
```

### 2.18.3 -blend\_fade

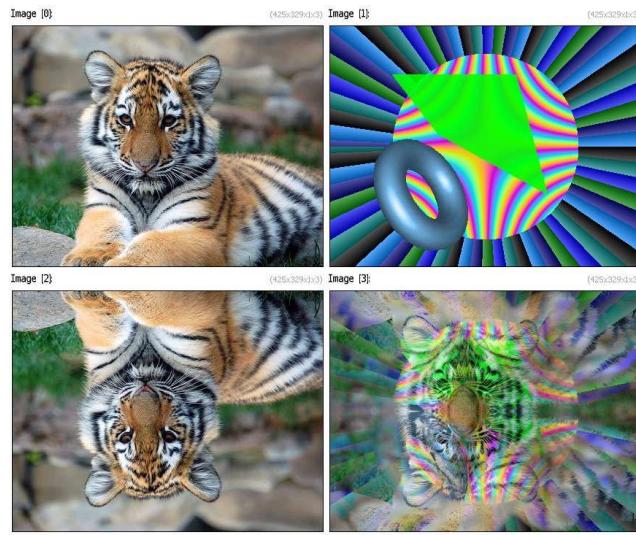
Blend selected images together using a given fading pattern (defined as the latest image).



```
Example 581 : image.jpg -testimage2d {w},{h} 100%,100%,1,1,'cos(y/10)',  
-normalize[-1] 0,1 --blend_fade
```

### 2.18.4 -blend\_median

Blend selected images together using 'median' mode.



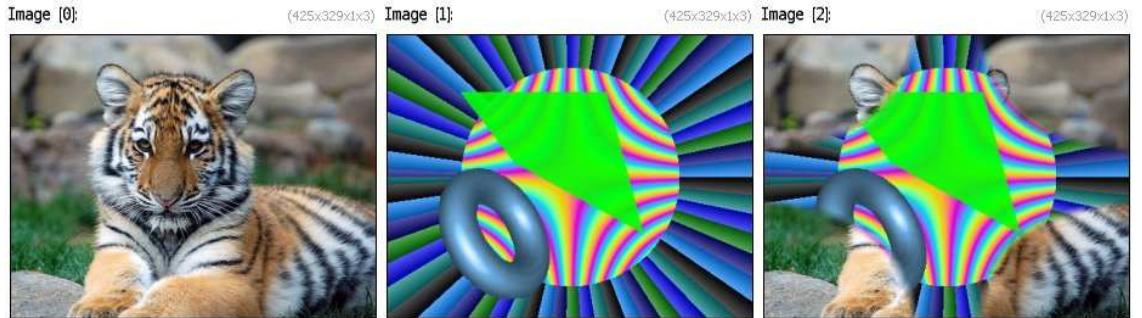
**Example 582 :** `image.jpg -testimage2d {w}, {h} --mirror[0] y --blend_median`

### 2.18.5 *-fade\_diamond*

**Arguments:**  $0 \leq \text{start} \leq 100$ ,  $0 \leq \text{end} \leq 100$

Create diamond fading from selected images.

**Default values:** 'start=80' and 'end=90'.



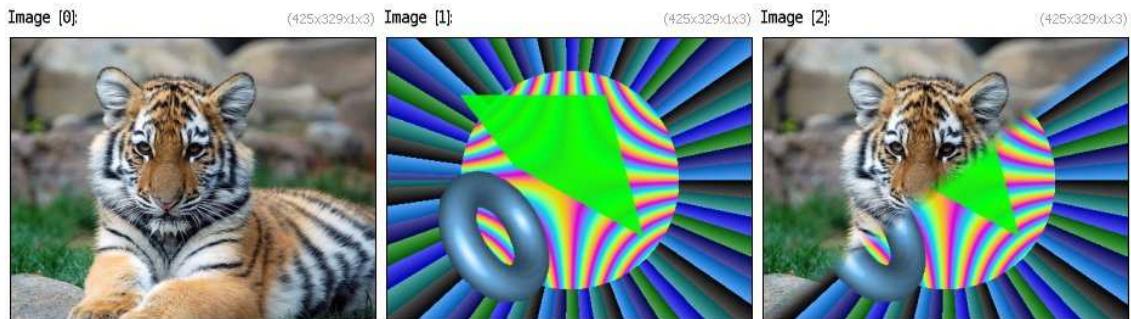
**Example 583 :** `image.jpg -testimage2d {w}, {h} --fade_diamond 80, 85`

### 2.18.6 *-fade\_linear*

**Arguments:**  $\text{angle}$ ,  $0 \leq \text{start} \leq 100$ ,  $0 \leq \text{end} \leq 100$

Create linear fading from selected images.

**Default values:** 'angle=45', 'start=30' and 'end=70'.



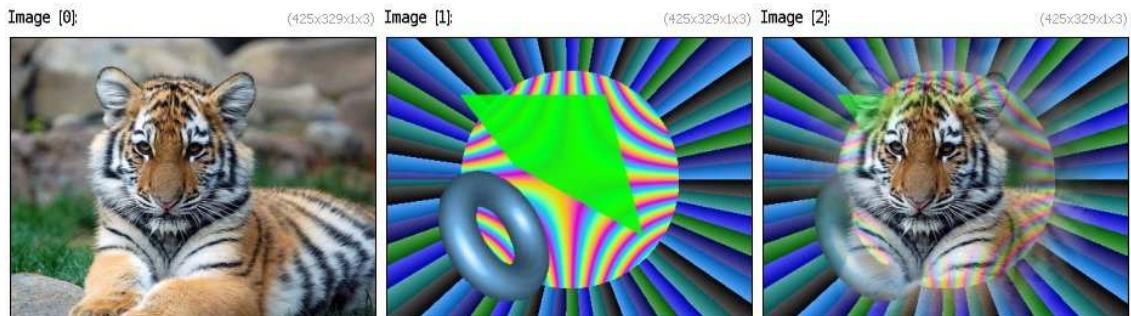
**Example 584 :** image.jpg -testimage2d {w}, {h} --fade\_linear 45, 48, 52

### 2.18.7 -fade\_radial

**Arguments:**  $0 \leq \text{start} \leq 100$ ,  $0 \leq \text{end} \leq 100$

Create radial fading from selected images.

**Default values:** 'start=30' and 'end=70'.



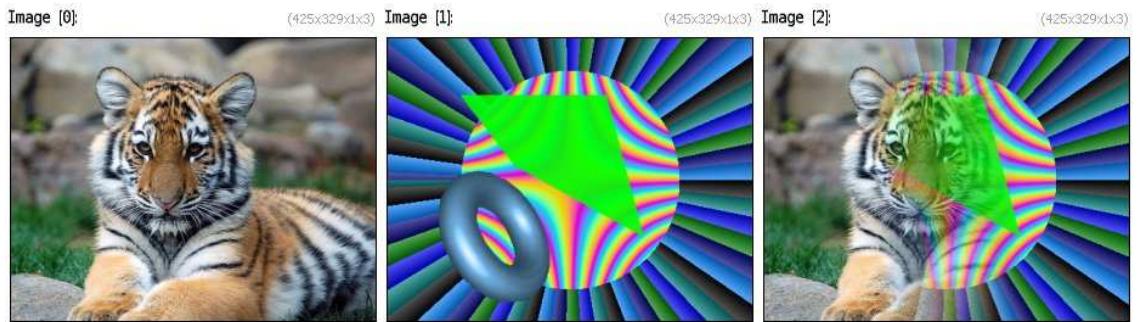
**Example 585 :** image.jpg -testimage2d {w}, {h} --fade\_radial 30, 70

### 2.18.8 -fade\_x

**Arguments:**  $0 \leq \text{start} \leq 100$ ,  $0 \leq \text{end} \leq 100$

Create horizontal fading from selected images.

**Default values:** 'start=30' and 'end=70'.



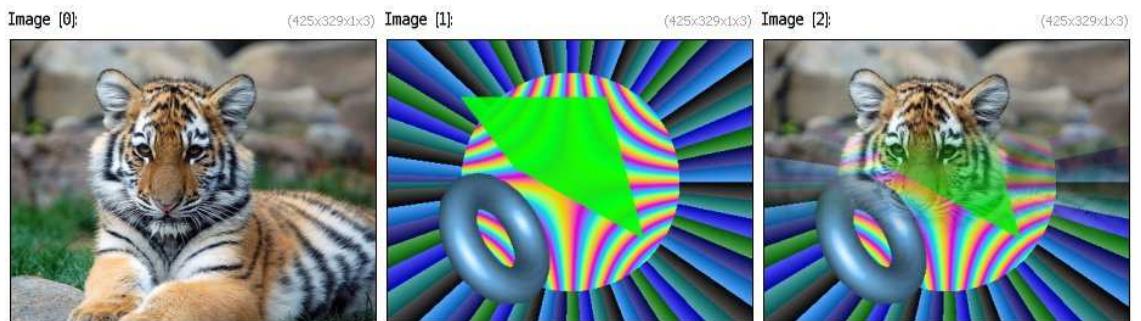
**Example 586:** `image.jpg -testimage2d {w},{h} --fade_x 30,70`

### 2.18.9 *-fade\_y*

**Arguments:**  $0 \leq \text{start} \leq 100, 0 \leq \text{end} \leq 100$

Create vertical fading from selected images.

**Default values:** 'start=30' and 'end=70'.



**Example 587:** `image.jpg -testimage2d {w},{h} --fade_y 30,70`

### 2.18.10 *-fade\_z*

**Arguments:**  $0 \leq \text{start} \leq 100, 0 \leq \text{end} \leq 100$

Create transversal fading from selected images.

**Default values:** 'start=30' and 'end=70'.

## 2.19 Image sequences

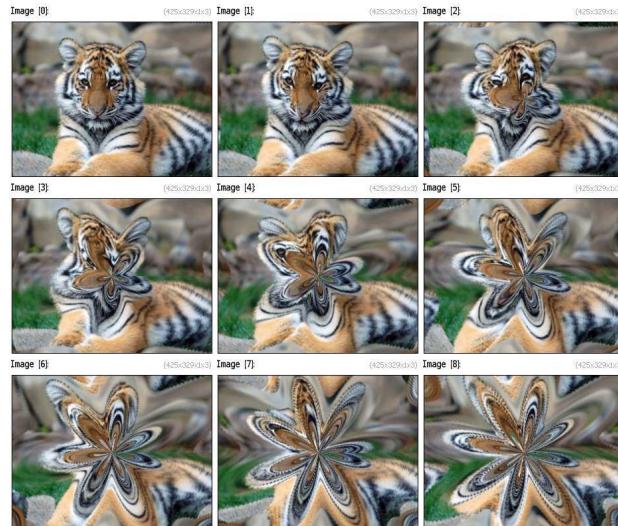
### 2.19.1 *-animate*

**Arguments:** `filter_name, "param1_start,...,paramN_start", "param1_end,...,paramN_end", n`

```
0 | 1 }, _output_filename |
    delay>0
```

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

**Default value:** 'delay=30'.



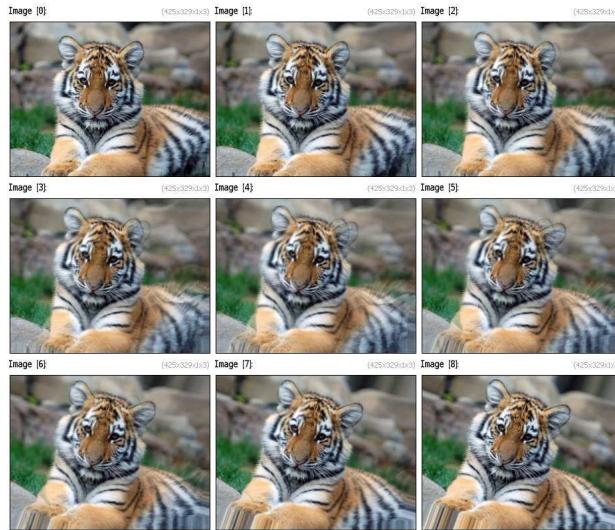
**Example 588 :** image.jpg -animate flower,"0,3","20,8",9

## 2.19.2 -morph

**Arguments:** nb\_frames>0, \_smoothness>=0, \_precision>0

Create morphing sequence between selected images.

**Default values:** 'smoothness=0.1' and 'precision=5'.



**Example 589 :** `image.jpg --rotate 20,1,1,50%,50% -morph 9`

### 2.19.3 *-register\_nonrigid*

**Arguments:** `_smoothness>=0`, `_precision>0`, `_nb_scale>=0`

Register selected images with non-rigid warp.

**Default values:** '`smoothness=0.2`', '`precision=6`' and '`nb_scale=0 (auto)`'.



**Example 590 :** `image.jpg --rotate 20,1,1,50%,50% --register_nonrigid , -remove[-2]`

### 2.19.4 *-register\_rigid*

**Arguments:** `_smoothness>=0`

Register selected images with rigid warp.

**Default value:** '`smoothness=1`'.



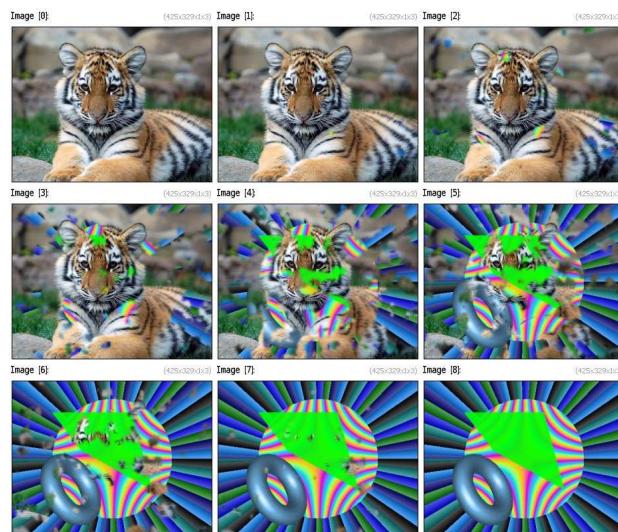
**Example 591 :** `image.jpg --shift 30,20 --register_rigid , -remove[-2]`

### 2.19.5 -transition plasma

**Arguments:** `_nb_frames>=2, _scale>=0, _smoothness [%] >=0`

Create plasma transition sequence between consecutive images.

**Default values:** '`nb_frames=10`', '`scale=5`' and '`smoothness=0.5%`'.



**Example 592 :** `image.jpg -testimage2d {w}, {h} -transition_plasma 9`

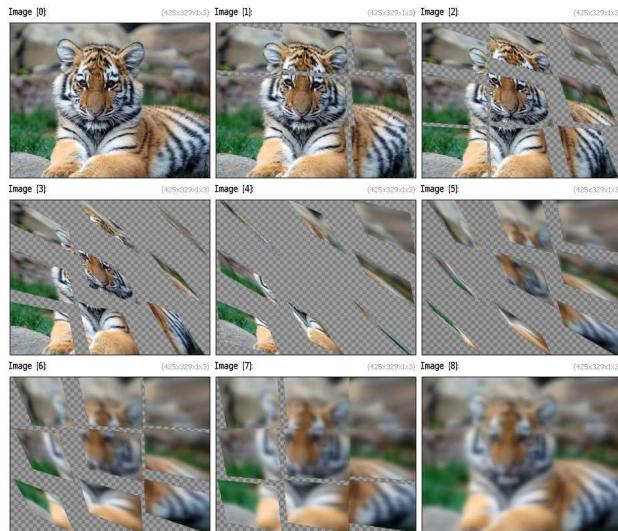
### 2.19.6 -transition3d

**Arguments:** `_nb_frames>=2, _nb_xtiles>0, _nb_ytiles>0, _axis_x, _axis_y, _axis_z, _is_antialias`

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 593 :** `image.jpg --blur 5 -transition3d 9 -rgba`

## 2.20 Interactive demos

### 2.20.1 *-x blobs*

Launch the blobs editor.

### 2.20.2 *-x fire*

Launch the fire demo.

### 2.20.3 *-x fireworks*

Launch the fireworks demo.

### 2.20.4 *-x fisheye*

Launch fish-eye demo.

### 2.20.5 *-x fourier*

Launch fourier filtering demo.

### 2.20.6 *-x histogram*

Launch histogram demo.

### 2.20.7 *-x hough*

Launch hough transform demo.

**2.20.8 -x *jawbreaker***

**Arguments:**  $0 < \text{width} < 20, 0 < \text{height} < 20, 0 < \text{balls} \leq 8$

Launch the Jawbreaker game.

**2.20.9 -x *life***

Launch the game of life.

**2.20.10 -x *light***

Launch the light demo.

**2.20.11 -x *mandelbrot***

**Arguments:**  $-\text{julia} = \{ 0 \mid 1 \}, -\text{c0r}, -\text{c0i}$

Launch Mandelbrot/Julia explorer.

**2.20.12 -x *minesweeper***

**Arguments:**  $8 \leq \text{width} \leq 20, 8 \leq \text{height} \leq 20$

Launch the Minesweeper game.

**2.20.13 -x *minimal\_path***

Launch the minimal path demo.

**2.20.14 -x *pacman***

Launch pacman game.

**2.20.15 -x *paint***

Launch the interactive painter.

**2.20.16 -x *plasma***

Launch the plasma demo.

**2.20.17 -x *quantize\_rgb***

**Arguments:**  $\text{nbcolors} \geq 2$

Launch RGB colors quantization demo.

**2.20.18 *-x\_reflection3d***

Launch the 3d reflection demo.

**2.20.19 *-x\_rubber3d***

Launch the 3d rubber demo.

**2.20.20 *-x\_shadebobs***

Launch the shade bobs demo.

**2.20.21 *-x\_spline***

Launch spline curve editor.

**2.20.22 *-x\_tetris***

Launch tetris game.

**2.20.23 *-x\_tictactoe***

Launch tic-tac-toe game.

**2.20.24 *-x\_whirl***

**Arguments:** *\_opacity>=0*

Launch fractal whirl demo.

**Default values:** '*opacity=0.2*' .

## 2.21 PINK-library operators

**2.21.1 *-output pink3d***

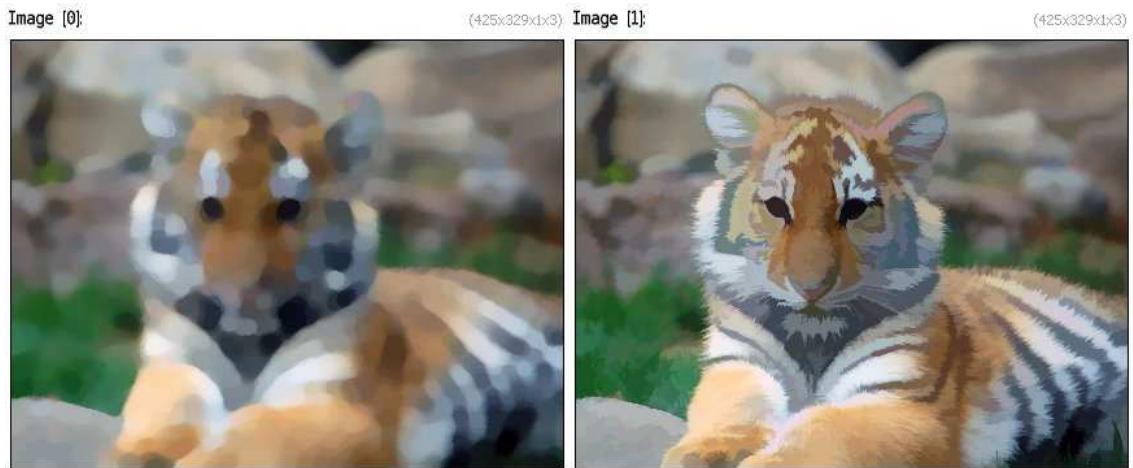
**Arguments:** *filename*

Save selected images as P5-coded PPM files (PINK extension for 3d volumetric images).

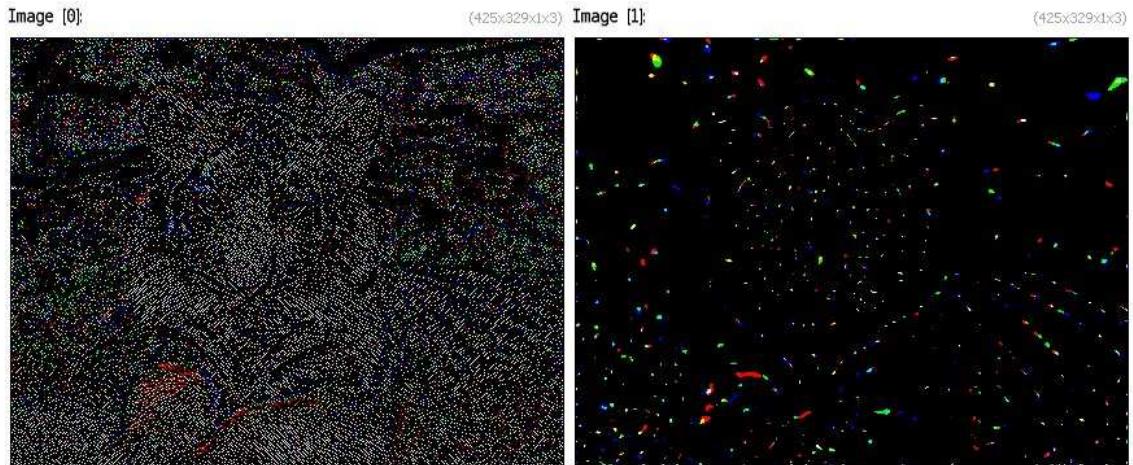
**2.21.2 *-pink***

Pink wrapper name,p1, .. ,pn (requires the PINK library to be installed).

(<http://pinkhq.com/>) prepares input, calls external "name input p1 ... pn output" and reads output (/tmp)



**Example 594 :** image.jpg --pink asfr,5 -pink[0] asf,5



**Example 595 :** image.jpg --blur 2 -pink maxima,4

### 2.21.3 -pink\_grayskel

**Arguments:** \_connectivity={ 4 | 8 | 6 | 26 }, \_lambda=0

([http://pinkhq.com/grayskel\\_8c.html](http://pinkhq.com/grayskel_8c.html))

Grayscale homotopic skeleton (requires the PINK library to be installed).

**Default values:** 'connectivity=4' and 'lambda=0'.



**Example 596 :** `image.jpg --pink_grayskel , --pink_grayskel[0] ,10 --pink_grayskel[0] ,100 -append_tiles 2`

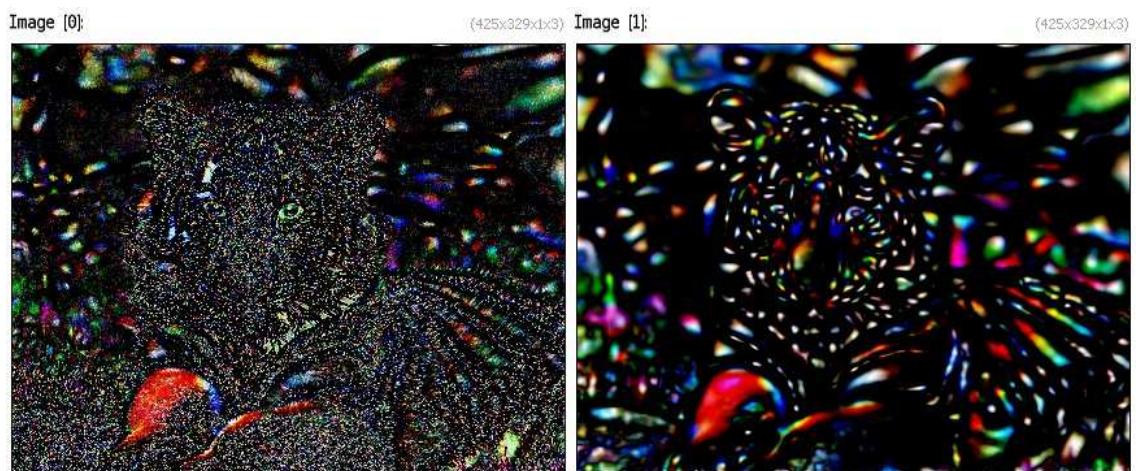
#### 2.21.4 *-pink heightmaxima*

**Arguments:** `_connectivity={ 4 | 8 | 6 | 26 },_height=1`

([http://pinkhq.com/heightmaxima\\_8c.html](http://pinkhq.com/heightmaxima_8c.html))

Heightmaxima filtering (requires the PINK library to be installed).

**Default values:** '`connectivity=4`' and '`height=1`'.



**Example 597 :** `image.jpg --blur 2 --pink_heightminima ,15 --pink_heightmaxima[0,1] ,15 --[-3,-1] --[-3,-1] -k[-1,-2]`

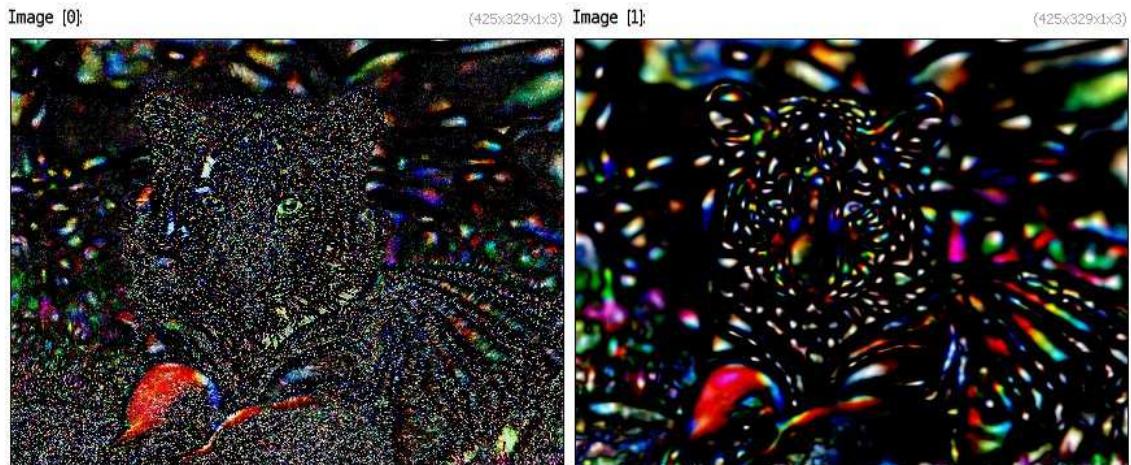
### 2.21.5 *-pink\_heightminima*

**Arguments:** `_connectivity={ 4 | 8 | 6 | 26 },_height=1`

([http://pinkhq.com/heightminima\\_8c.html](http://pinkhq.com/heightminima_8c.html))

Heightminima filtering (requires the PINK library to be installed).

**Default values:** '`connectivity=4`' and '`height=1`'.



**Example 598:** `image.jpg --blur 2 --pink_heightminima ,15 --pink_heightmaxima[0,1]`  
`,15 --[-3,-1] --[-3,-1] -k[-1,-2]`

### 2.21.6 *-pink\_htkern*

**Arguments:** `_connectivity={ 4 | 8 | 6 | 26 }, _type={" " | u}`

([http://pinkhq.com/htkern\\_8c.html](http://pinkhq.com/htkern_8c.html))

([http://pinkhq.com/htkernu\\_8c.html](http://pinkhq.com/htkernu_8c.html))

Grayscale ultimate homotopic thinning/thickening without condition (requires the PINK library to be installed).

**Default values:** '`connectivity=4`' and '`type=""`'.



**Example 599 :** image.jpg --pink\_htkern ,u --pink\_htkern[0] , ---[-1,-2] -rm[0]

### 2.21.7 *-pink\_lvkern*

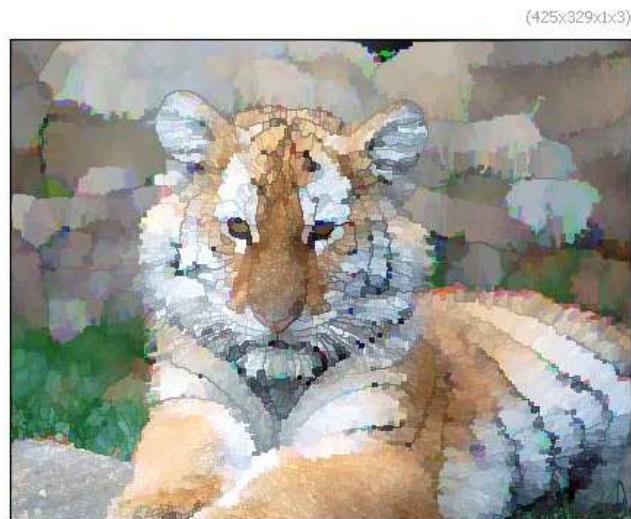
**Arguments:** `_connectivity={ 4 | 8 | 6 | 26 }`, `_type={" " | u}`

([http://pinkhq.com/lvkern\\_8c.html](http://pinkhq.com/lvkern_8c.html))

([http://pinkhq.com/lvkernu\\_8c.html](http://pinkhq.com/lvkernu_8c.html))

Grayscale ultimate leveling thinning/thickening without condition (requires the PINK library to be installed).

**Default values:** `'connectivity=4'` and `'type=""'`.



**Example 600 :** image.jpg -pink\_lvkern ,u

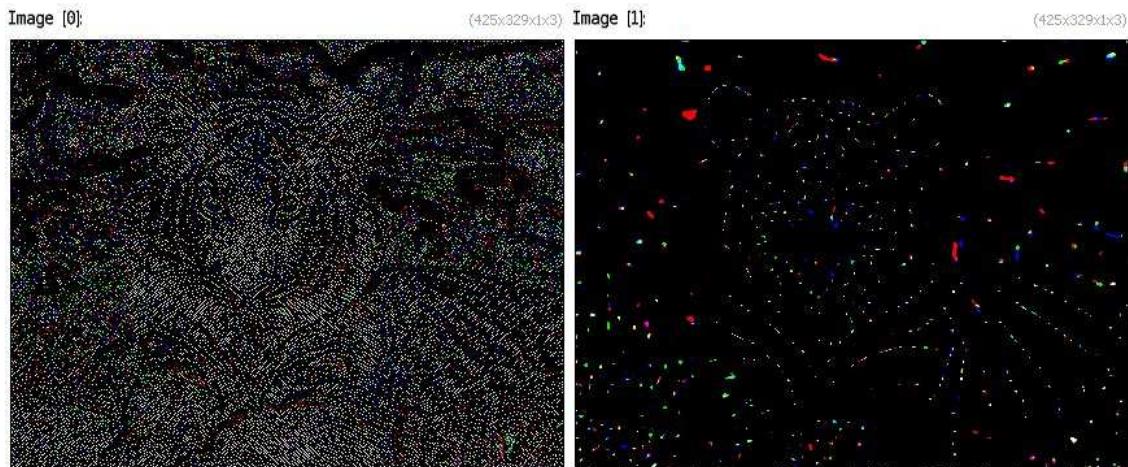
### 2.21.8 *-pink reg minima*

**Arguments:** `_connectivity={ 4 | 8 | 6 | 26 }`

([http://pinkhq.com/minima\\_8c.html](http://pinkhq.com/minima_8c.html))

Regional minima (requires the PINK library to be installed).

**Default values:** 'connectivity=4'.



**Example 601:** `image.jpg --blur 2 -pink_reg_minima ,`

### 2.21.9 -pink\_skelcurv

**Arguments:** `-prio={0 | 1 | 2 | 3 | 4 | 8 | 6 | 26}, -connectivity={ 4 | 8 | 6 | 26 }, -inhibit={"")}`

([http://pinkhq.com/skelcurv\\_8c.html](http://pinkhq.com/skelcurv_8c.html))

Curvilinear binary skeleton guided by a priority function or image (requires the PINK library to be installed).

**Default values:** 'prio=0', 'connectivity=4' and 'inhibit=""'.



**Example 602:** `image.jpg -threshold 50% {w},{h} -fill[-1] 'if(x>w/2,255,0)' tp=@{-path_tmp} -output[-1] ${tp}/inhibit.pgm -remove[-1] --pink_skelcurv[0] , --pink_skelcurv[0] , ${tp}/inhibit.pgm -exec "rm \"${tp}\"/inhibit.pgm"`



**Example 603 :** `image.jpg -threshold 50% --pink_skelcurv , --pink_skelcurv[-2] , 8`

### 2.21.10 *-pink\_skelend*

**Arguments:** `_connectivity={ 4 | 8 | 6 | 26 },_n=0`

([http://pinkhq.com/skelend\\_8c.html](http://pinkhq.com/skelend_8c.html))

Homotopic skeleton of a 2d or 3d binary image with dynamic detection of end points (requires the PINK library to be installed).

**Default values:** '`connectivity=4`' and '`n=0`' .



**Example 604 :** `image.jpg -threshold 50% --pink_skelend , --pink_skelend[-2] ,-1`

### 2.21.11 *-pink\_skeleton*

**Arguments:** `_prio={0 | 1 | 2 | 3 | 4 | 8 | 6 | 26},_connectivity={ 4 | 8 | 6 | 26 },_inhibit={" "}`

([http://pinkhq.com/skeleton\\_8c.html](http://pinkhq.com/skeleton_8c.html))

Ultimate binary skeleton guided by a priority image (requires the PINK library to be installed).

**Default values:** '`prio=0`', '`connectivity=4`' and '`inhibit=""`' .



**Example 605 :** `image.jpg -threshold 50% --pink_skeleton[-1]`,

### 2.21.12 *-pink\_skelpar*

**Arguments:** `_algorithm={0..29}`, `_nsteps=-1`, `_inhibit=""`

([http://pinkhq.com/skelpar\\_8c.html](http://pinkhq.com/skelpar_8c.html))

Parallel binary skeleton (requires the PINK library to be installed).

**Default values:** `'algorithm=4'`, `'nsteps=-1'` and `'inhibit=""'`.



**Example 606 :** `image.jpg -threshold 50% --pink_skelpar[-1] 0 --pink_skelpar[-1] 2`

### 2.21.13 *-pink\_wshed*

**Arguments:** `_connectivity={ 4 | 8 | 6 | 26 }`, `_inverse={ 0 | 1 }`, `_height=0`

([http://pinkhq.com/wshedtopo\\_8c.html](http://pinkhq.com/wshedtopo_8c.html))

Watershed (requires the PINK library to be installed).

**Default values:** 'connectivity=4', 'inverse=0' and 'height=0'.



**Example 607 :** `image.jpg --pink_wshed ,1,5 -pink_wshed[0] , ,5`

## 2.22 Convenience functions

### 2.22.1 *-alert*

**Arguments:** `_title, _message, _label_button1, _label_button2, ...`

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

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

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

### 2.22.2 *-arg*

**Arguments:** `n, _arg1, ..., _argN`

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

'n' can be an expression.

### 2.22.3 *-at*

**Arguments:** `_x, _y, _z`

Return a specified vector-valued point (x,y,z) from the latest of the selected images.

**2.22.4 *-autocrop\_coords***

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

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

**Default value:** 'auto'

**2.22.5 *-average\_color***

Return the average color of the latest of the selected images.

**2.22.6 *-basename***

**Arguments:** file\_path,\_variable\_name\_for\_folder

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

When specified 'variable\_name\_for\_folder' must starts by an underscore (global variable accessible from calling function).

**2.22.7 *-bin***

**Arguments:** binary\_int1,...

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

**2.22.8 *-bin2dec***

**Arguments:** binary\_int1,...

Convert specified binary integers into their decimal representations.

**2.22.9 *-dec***

**Arguments:** decimal\_int1,...

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

**2.22.10 -dec2str**

**Arguments:** decimal\_int1, ...

Convert specifial decimal integers into its string representation.

**2.22.11 -dec2bin**

**Arguments:** decimal\_int1, ...

Convert specified decimal integers into their binary representations.

**2.22.12 -dec2hex**

**Arguments:** decimal\_int1, ...

Convert specified decimal integers into their hexadecimal representations.

**2.22.13 -dec2oct**

**Arguments:** decimal\_int1, ...

Convert specified decimal integers into their octal representations.

**2.22.14 -fact**

**Arguments:** value

Return the factorial of the specified value.

**2.22.15 -file\_mv**

**Arguments:** filename\_src, filename\_dest

Rename or move a file from a location \$1 to another location \$2.

**2.22.16 -file\_rand**

Return a random filename for storing temporary data.

**2.22.17 *-file\_rm*****Arguments:** `filename`

Delete a file.

**2.22.18 *-file\_slash***

Return '/' or '\' as a path separator for filenames.

**2.22.19 *-filename*****Arguments:** `filename, number1, number2, ..., numberN`

Return a filename numbered with specified indices.

**2.22.20 *-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.

**2.22.21 *-fitscreen*****Arguments:** `width, height, depth`

Return the 'ideal' size WxH for a window intended to display an image of specified size on screen.

**2.22.22 *-gcd*****Arguments:** `a, b`

Return the GCD (greatest common divisor) between a and b.

**2.22.23 *-hex*****Arguments:** `hexadecimal_int1, ...`

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

**2.22.24 -hex2dec**

**Arguments:** `hexadecimal_int1, ...`

Convert specified hexadecimal integers into their decimal representations.

**2.22.25 -hex2str**

**Arguments:** `hexadecimal_string`

Convert specified hexadecimal string into a string.

**2.22.26 -img2str**

Return the content of the latest of the selected image as a special G'MIC input string.

**2.22.27 -img2text**

**Arguments:** `_line_separator`

Return text contained in a multi-line image.

**Default value:** `'line_separator= '`.

**2.22.28 -img82hex**

Convert selected 8bits-valued vectors into their hexadecimal representations (ascii-encoded).

**2.22.29 -hex2img8**

Convert selected hexadecimal representations (ascii-encoded) into 8bits-valued vectors.

**2.22.30 -is\_3d**

Return 1 if all of the selected image are 3d objects, 0 otherwise.

**2.22.31 -is\_percent**

**Arguments:** `string`

Return 1 if specified string ends with a '%', 0 otherwise.

**2.22.32 -is\_windows**

Return 1 if current computer OS is Windows, 0 otherwise.

**2.22.33 *-mad***

Return the MAD (Maximum Absolute Deviation) of the last selected image.

The MAD is defined as  $MAD = \text{med\_i} | x_{-i} - \text{med\_j}(x_{-j}) |$

**2.22.34 *-max\_w***

Return the maximal width between selected images.

**2.22.35 *-max\_h***

Return the maximal height between selected images.

**2.22.36 *-max\_d***

Return the maximal depth between selected images.

**2.22.37 *-max\_s***

Return the maximal spectrum between selected images.

**2.22.38 *-max\_wh***

Return the maximal wxh size of selected images.

**2.22.39 *-max\_whd***

Return the maximal wxhxd size of selected images.

**2.22.40 *-max\_whds***

Return the maximal wxhdxs size of selected images.

**2.22.41 *-med***

Return the median value of the last selected image.

**2.22.42 *-min\_w***

Return the minimal width between selected images.

**2.22.43 *-min\_h***

Return the minimal height between selected images.

**2.22.44 *-min\_d***

Return the minimal depth between selected images.

**2.22.45 *-min\_s***

Return the minimal s size of selected images.

**2.22.46 *-min\_wh***

Return the minimal wxh size of selected images.

**2.22.47 *-min\_whd***

Return the minimal wxhxd size of selected images.

**2.22.48 *-min\_whds***

Return the minimal wxhdxs size of selected images.

**2.22.49 *-oct***

**Arguments:** octal\_int1, ...

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

**2.22.50 *-oct2dec***

**Arguments:** octal\_int1, ...

Convert specified octal integers into their decimal representations.

**2.22.51 *-padint***

**Arguments:** number, \_size>0

Return a integer with 'size' digits (eventually left-padded with '0').

**2.22.52 *-path\_tmp***

Return a path to store temporary files (whose value is OS-dependent).

**2.22.53 *-path\_user***

Return a path to store persistent configuration files for one user (whose value is OS-dependent).

**2.22.54 *-quote***

**Arguments:** string

Return a "quotified" version of the string.

**2.22.55 *-region feature***

**Arguments:** `region_label, feature, _default_value`

Return feature for a specified region.

This function requires two images [img,region\_label] in the selection.

Argument 'feature' is a string that corresponds to the way the feature would be asked for the entire image.

**Default value:** '`default_value=0`' .



```
Example 608 : image.jpg --luminance -quantize[-1] 2 -label[-1] 0,1
mean=@{"-region_feature[0,1] 10,\\"{ia}\\""} sum=@{"-region_feature[0,1]
10,\\"@{-1,+}\\""}
```

**2.22.56 *-reset***

Reset global parameters of the interpreter environment.

**2.22.57 *-RGB***

Return a random int-valued RGB color.

**2.22.58 *-RGBA***

Return a random int-valued RGBA color.

**2.22.59 *-str***

**Arguments:** `string`

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

**2.22.60 -str2hex****Arguments:** string

Convert specified string into a sequence of hexadecimal values.

**2.22.61 -stresc****Arguments:** val1, ..., valN

Return escaped string from specified ascii codes.

**2.22.62 -strcat****Arguments:** string1, string2, ...

Return the concatenation of all strings passed as arguments.

**2.22.63 -strcmp****Arguments:** string1, string2

Return 1 if the two strings are equal, 0 otherwise.

**2.22.64 -strlen****Arguments:** string1

Return the length of specified string argument.

**2.22.65 -strreplace****Arguments:** string, search, replace

Search and replace substrings in an input string.

**2.22.66 -struncase****Arguments:** string

Return a lower-case version of the specified string.

**2.22.67 -strver**

Return the current version number of the G'MIC interpreter, as a string.

**2.22.68 -tic**

Initialize tic-toc timer.

Use it in conjunction with '-toc'.

**2.22.69 -toc**

Display elapsed time of the tic-toc timer since the last call to '-tic'.

Use it in conjunction with '-tic'.

**2.22.70 -variance noise**

Return the estimated noise variance of the last selected image.

## 2.23 Others

**2.23.1 -gpt**

**Arguments:** \_scorefile, \_number\_of\_sessions>=0

Generate score board for the GPT championship (GREYC Poker Tour).

** GREYC Poker Tour ** Session #93 (avg. #player = 6.5)														
Session winner : Edouard														
Rank	Name	Score	Short Score	Hand	Card	Face	Blinds	Cash	Net Total	Player	Player (Avg)	Hand (Avg)	Blinds (Avg)	Net Total (Avg)
1	Edouard	+1230	+1230.00	73	28.7	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
2	Lata	+932	+466.00	12	5.9	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
3	Pierre-Anthony	+777	+388.50	84	11.8	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
4	Mounaf	+564	+282.00	36	13.8	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
5	Xavier	+497	+248.50	49	16.8	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
6	Scarlett	-	-	-	-	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
7	Youssef	-63	-31.50	3	0.0	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
8	Mohamed	-123	-61.50	42	7.5	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
9	Soh Berry	-155	-77.50	8	0.0	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
10	Tuc	-178	-89.00	2	0.0	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
11	Salah	-223	-111.50	29	7.7	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
12	Mathilde	-295	-148.00	80	10.0	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
13	Bernit	-322	-161.00	36	4.9	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
14	Alessia	-553	-276.50	2	0.0	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
15	Pierre	-610	-305.00	37	1.4	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
16	Soh B.	-620	-310.00	2	0.0	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
17	Lamine	-716	-353.00	3	0.0	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
18	Olivier	-717	-353.00	45	6.0	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
19	Soh F.	-1585	-792.50	12	3.8	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00
20	Nicolas	-1818	-909.00	2	0.0	Hand	100 100	+100.00	+100.00	100	100.00	100	100.00	+100.00

**Example 609 :** -gpt ,

## 2.24 Commands shortcuts

- `'-h'` (\*) is equivalent to `'-help'`.
- `'-m'` (\*) is equivalent to `'-command'`.
- `'-d'` (+) is equivalent to `'-display'`.
- `'-d0'` is equivalent to `'-display0'`.
- `'-d3d'` (+) is equivalent to `'-display3d'`.
- `'-da'` is equivalent to `'-display_array'`.
- `'-dffft'` is equivalent to `'-display_fft'`.
- `'-dg'` is equivalent to `'-display_graph'`.
- `'-dh'` is equivalent to `'-display_histogram'`.
- `'-dp'` is equivalent to `'-display_polar'`.
- `'-rgba'` is equivalent to `'-display_rgba'`.
- `'-dt'` is equivalent to `'-display_tensors'`.
- `'-dw'` is equivalent to `'-display_warp'`.
- `'-e'` (\*) is equivalent to `'-echo'`.
- `'-i'` (\*) is equivalent to `'-input'`.
- `'-o'` (\*) is equivalent to `'-output'`.
- `'-on'` is equivalent to `'-outputn'`.
- `'-op'` is equivalent to `'-outputp'`.
- `'-ow'` is equivalent to `'-outputw'`.
- `'-p'` (\*) is equivalent to `'-print'`.
- `'-sh'` (\*) is equivalent to `'-shared'`.
- `'-up'` is equivalent to `'-update'`.
- `'-v'` (\*) is equivalent to `'-verbose.'`.
- `'-w'` (+) is equivalent to `'-window'`.
- `'-k'` (\*) is equivalent to `'-keep'`.
- `'-mv'` (\*) is equivalent to `'-move'`.
- `'-nm'` (\*) is equivalent to `'-name'`.
- `'-rm'` (\*) is equivalent to `'-remove'`.
- `'-rv'` (\*) is equivalent to `'-reverse'`.
- `'-+'` (+) is equivalent to `'-add'`.
- `'-<<'` (+) is equivalent to `'-bsl'`.
- `'->>'` (+) is equivalent to `'-bsr'`.
- `'-/'` (+) is equivalent to `'-div'`.
- `'==='` (+) is equivalent to `'-eq'`.
- `'->='` (+) is equivalent to `'-ge'`.
- `'->'` (+) is equivalent to `'-gt'`.
- `'-<='` (+) is equivalent to `'-le'`.
- `'-<'` (+) is equivalent to `'-lt'`.
- `'-//'` (+) is equivalent to `'-mdiv'`.
- `'-%'` (+) is equivalent to `'-mod'`.
- `'-**'` (+) is equivalent to `'-mmul'`.
- `'-*'` (+) is equivalent to `'-mul'`.

- `'- !='` (+) is equivalent to `'-neq'`.
- `'- ^'` (+) is equivalent to `'-pow'`.
- `'--'` (+) is equivalent to `'-sub'`.
- `'-c'` (+) is equivalent to `'-cut'`.
- `'-f'` (+) is equivalent to `'-fill'`.
- `'-n'` (+) is equivalent to `'-normalize'`.
- `'-='` (+) is equivalent to `'-set'`.
- `'-t2'` is equivalent to `'-threshold2'`.
- `'-fc'` is equivalent to `'-fill_color'`.
- `'-a' (*)` is equivalent to `'-append'`.
- `'-z' (*)` is equivalent to `'-crop'`.
- `'-r' (*)` is equivalent to `'-resize'`.
- `'-rr2d'` is equivalent to `'-resize_ratio2d'`.
- `'-r2dx'` is equivalent to `'-resize2dx'`.
- `'-r2dy'` is equivalent to `'-resize2dy'`.
- `'-r3dx'` is equivalent to `'-resize3dx'`.
- `'-r3dy'` is equivalent to `'-resize3dy'`.
- `'-r3dz'` is equivalent to `'-resize3dz'`.
- `'-s' (*)` is equivalent to `'-split'`.
- `'-y' (*)` is equivalent to `'-unroll'`.
- `'-b'` (+) is equivalent to `'-blur'`.
- `'-g'` (+) is equivalent to `'-gradient'`.
- `'-j'` (+) is equivalent to `'-image'`.
- `'-t'` (+) is equivalent to `'-text'`.
- `'-+3d'` (+) is equivalent to `'-add3d'`.
- `'-b3d'` (+) is equivalent to `'-background3d'`.
- `'-c3d'` is equivalent to `'-center3d'`.
- `'-col3d'` (+) is equivalent to `'-color3d'`.
- `'-/3d'` (+) is equivalent to `'-div3d'`.
- `'-db3d'` (+) is equivalent to `'-double3d'`.
- `'-f3d'` (+) is equivalent to `'-focale3d'`.
- `'-l3d'` (+) is equivalent to `'-light3d'`.
- `'-m3d'` (+) is equivalent to `'-mode3d'`.
- `'-md3d'` (+) is equivalent to `'-moded3d'`.
- `'-*3d'` (+) is equivalent to `'-mul3d'`.
- `'-n3d'` is equivalent to `'-normalize3d'`.
- `'-o3d'` (+) is equivalent to `'-opacity3d'`.
- `'-p3d'` (+) is equivalent to `'-primitives3d'`.
- `'-rv3d'` (+) is equivalent to `'-reverse3d'`.
- `'-r3d'` (+) is equivalent to `'-rotate3d'`.
- `'-s13d'` (+) is equivalent to `'-spec13d'`.
- `'-ss3d'` (+) is equivalent to `'-specs3d'`.
- `'-s3d'` (+) is equivalent to `'-split3d'`.
- `'-3d'` (+) is equivalent to `'-sub3d'`.

- ‘-t3d’ (+) is equivalent to ‘-texturize3d’.
- ‘-endl’ (\*) is equivalent to ‘-endlocal’.
- ‘-x’ (\*) is equivalent to ‘-exec’.
- ‘-l’ (\*) is equivalent to ‘-local’.
- ‘-q’ (\*) is equivalent to ‘-quit’.
- ‘-u’ (\*) is equivalent to ‘-status’.
- ‘-frame’ is equivalent to ‘-frame\_xy’.

## 2.25 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 -o output.jpg
```

– Create a volumetric image from a movie sequence:

```
gmic input.mpg -a z -o output.hdr
```

– Compute image gradient norm:

```
gmic input.bmp -gradient_norm
```

– Denoise a color image:

```
gmic image.jpg --denoise 30,10 -o denoised.jpg
```

– Compose two images using overlay layer blending:

```
gmic image1.jpg image2.jpg -blend overlay -o blended.jpg
```

– Evaluate a mathematical expression:

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

– Plot a 2d function:

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

– Plot a 3d elevated function in random colors:

```
gmic 128,128,1,3,"?(0,255)" -plasma 10,3 -blur 4 -sharpen 10000 \
-elevation3d[-1] "X=(x-64)/6;Y=(y-64)/6;100*exp(-(X^2+Y^2)/30)*abs(cos(X)*sin(Y))"
```

– Plot the isosurface of a 3d volume:

```
gmic -m3d 5 -md3d 5 -db3d 0 -isosurface3d ""x^2+y^2+abs(z)^abs(4*cos(x*y*z*3))"",3
```

– Render a G'MIC 3d logo:

```
gmic 1 -text G\`MIC,0,0,57,1,1,1 -expand_xy 10,0 -blur 2 -n 0,100 --plasma 0.4 -+ \
```

```
–blur 1 –elevation3d –0.1 –md3d 4
```

- Generate a 3d ring of torii:

```
gmic –repeat 20 –torus3d 15,2 –col3d[–1] ”{(60,255)},{(60,255)},{(60,255)}” \  
–*3d[–1] 0.5,1 –if ”{$>%2}” –r3d[–1] 0,1,0,90 –endif –+3d[–1] 70 –+3d \  
–r3d 0,0,1,18 –done –md3d 3 –m3d 5 –db3d 0
```

- Create a vase from a 3d isosurface:

```
gmic –md3d 4 –isosurface3d ”x^2+2*abs(y/2)*sin(2*y)^2+z^2-3”,0” –sphere3d 1.5 \  
–3d[–1] 0.5 –plane3d 15,15 –r3d[–1] 1,0,0,90 –c3d[–1] –+3d[–1] 0,3.2 \  
–col3d[–1] 180,150,255 –col3d[–2] 128,255,0 –col3d[–3] 255,128,0 –+3d
```

- Display filtered webcam stream:

```
gmic –apply_camera \”–mirror x –mirror y –+ –/ 4\”
```

- Launch a set of G'MIC interactive demos:

```
gmic –x_fisheye –x_fire G\’MIC –x_tictactoe –rm –x_spline –x_mandelbrot 0 –x_light \  
–x_whirl , –x_life –x_jawbreaker , –x_blobs –x_tetris
```

# Index of commands

-RGB, 377  
-RGBA, 377  
-abs, 41  
-acos, 41  
-add, 42  
-add3d, 243  
-alert, 370  
-and, 44  
-animate, 356  
-animate3d, 244  
-append, 125  
-append\_tiles, 127  
-apply\_camera, 17  
-apply\_camera3d, 245  
-apply\_channels, 103  
-apply\_curve, 81  
-apply\_gamma, 82  
-apply\_parallel, 289  
-apply\_parallel2, 290  
-apply\_parallel4, 290  
-apply\_parallel8, 291  
-apply\_pose3d, 245  
-area, 197  
-area\_fg, 198  
-arg, 370  
-array, 299  
-array\_fade, 299  
-array\_mirror, 300  
-array\_random, 301  
-asin, 45  
-at, 370  
-atan, 46  
-atan2, 47  
-autocrop, 128  
-autocrop\_components, 129  
-autocrop\_coords, 371  
-autocrop\_seq, 129  
-autoindex, 104  
-average\_color, 371  
-axes, 214  
-axes3d, 245  
-background3d, 246  
-balance\_gamma, 82  
-ball, 214  
-bandpass, 153  
-barycenter, 198  
-basename, 371  
-bayer2rgb, 105  
-bilateral, 153  
-bin, 371  
-bin2dec, 371  
-blend, 349  
-blend\_edges, 352  
-blend\_fade, 353  
-blend\_median, 353  
-blur, 154  
-blur-angular, 154  
-blur\_linear, 155  
-blur\_radial, 155  
-blur\_selective, 156  
-blur\_x, 157  
-blur\_xy, 157  
-blur\_xyz, 158  
-blur\_y, 158  
-blur\_z, 159  
-bokeh, 159  
-box3d, 246  
-boxfitting, 311  
-break, 292  
-bsl, 47  
-bsr, 48  
-camera, 18

-cartoon, 312  
-center3d, 247  
-channels, 130  
-check, 291  
-chessboard, 215  
-cie1931, 216  
-circle, 216  
-circle3d, 247  
-circles3d, 248  
-circlism, 312  
-cmy2rgb, 105  
-cmyk2rgb, 106  
-color3d, 249  
-color\_ellipses, 313  
-colorcube3d, 249  
-colormap, 106  
-columns, 131  
-command, 18  
-complex2polar, 83  
-compose\_channels, 106  
-compose\_freq, 160  
-cone3d, 250  
-continue, 291  
-convolve, 160  
-convolve\_fft, 161  
-correlate, 162  
-cos, 49  
-cosh, 50  
-cracks, 342  
-crop, 131  
-cross\_correlation, 162  
-cubism, 313  
-cumul, 83  
-cup3d, 250  
-curvature, 163  
-cut, 83  
-cylinder3d, 251  
-deblur, 163  
-deblur\_goldmeinel, 164  
-deblur\_richardsonlucy, 164  
-debug, 17  
-dec, 371  
-dec2bin, 372  
-dec2hex, 372  
-dec2oct, 372  
-dec2str, 372  
-deconvolve\_fft, 165  
-deform, 332  
-deinterlace, 165  
-denoise, 166  
-denoise\_haar, 166  
-deriche, 167  
-diagonal, 133  
-diffusontensors, 171  
-dijkstra, 239  
-dilate, 168  
-dilate\_circ, 169  
-dilate\_oct, 169  
-direction2rgb, 107  
-discard, 84  
-displacement, 199  
-display, 18  
-display0, 19  
-display3d, 19  
-display\_array, 19  
-display\_fft, 19  
-display\_graph, 19  
-display\_histogram, 20  
-display\_polar, 21  
-display\_rgba, 22  
-display\_tensors, 22  
-display\_warp, 23  
-distance, 200  
-distribution3d, 251  
-ditheredbw, 107  
-div, 51  
-div3d, 252  
-divergence, 170  
-do, 292  
-document\_gmic, 24  
-dog, 170  
-done, 293  
-dotsbw, 314  
-double3d, 253  
-draw\_whirl, 314  
-drawing, 315  
-drop\_shadow, 315  
-echo, 24

-echo\_file, 24  
-echo\_stdout, 25  
-edges, 171  
-eigen, 240  
-eigen2tensor, 85  
-eikonal, 172  
-elevate, 133  
-elevation3d, 253  
-elif, 293  
-ellipse, 217  
-ellipsionism, 316  
-else, 293  
-empty3d, 254  
-endian, 85  
-endif, 293  
-endlocal, 293  
-eq, 52  
-equalize, 85  
-erode, 172  
-erode\_circ, 173  
-erode\_oct, 174  
-error, 294  
-euclidean2polar, 331  
-exec, 294  
-exp, 53  
-expand\_x, 133  
-expand\_xy, 134  
-expand\_xyz, 135  
-expand\_y, 135  
-expand\_z, 135  
-extrude3d, 255  
-fact, 372  
-fade\_diamond, 354  
-fade\_linear, 354  
-fade\_radial, 355  
-fade\_x, 355  
-fade\_y, 356  
-fade\_z, 356  
-fft, 174  
-fft82float, 201  
-fftpolar, 201  
-file\_mv, 372  
-file\_rand, 372  
-file\_rm, 373  
-file\_slash, 373  
-filename, 373  
-fill, 86  
-fill\_color, 108  
-fire\_edges, 317  
-fisheye, 332  
-fitratio\_wh, 373  
-fitscreen, 373  
-float2fft8, 201  
-float2int8, 88  
-flood, 218  
-flower, 333  
-focale3d, 256  
-frame\_blur, 301  
-frame\_cube, 302  
-frame\_fuzzy, 302  
-frame\_painting, 303  
-frame\_pattern, 304  
-frame\_round, 304  
-frame\_x, 305  
-frame\_xy, 306  
-frame\_xyz, 306  
-frame\_y, 306  
-function1d, 25  
-gaussian, 218  
-gaussians3d, 256  
-gcd, 373  
-ge, 54  
-glow, 317  
-gmic3d, 257  
-gmicky, 25  
-gmicky\_wilber, 26  
-gpt, 379  
-gradient, 175  
-gradient2rgb, 108  
-gradient\_norm, 176  
-gradient\_orientation, 176  
-graph, 219  
-grid, 220  
-gt, 55  
-gyroid3d, 257  
-haar, 176  
-halftone, 318  
-hardsketchbw, 319

-hearts, 319  
-heat\_flow, 177  
-help, 17  
-hessian, 177  
-hex, 373  
-hex2dec, 374  
-hex2img8, 374  
-hex2str, 374  
-histogram, 202  
-histogram3d, 258  
-histogram\_cumul, 203  
-histogram\_pointwise, 203  
-hough, 203  
-houghsketchbw, 320  
-hs12rgb, 109  
-hs182rgb, 109  
-hs1l2rgb, 109  
-hs182rgb, 109  
-hs2vrgb, 109  
-hs2v8rgb, 110  
-iee, 178  
-if, 294  
-ifft, 178  
-ifftpolar, 204  
-ihaar, 178  
-image, 221  
-image6cube3d, 258  
-image\_integral, 89  
-imagecube3d, 259  
-imagegrid, 307  
-imageplane3d, 259  
-imagepyramid3d, 260  
-imagerubik3d, 260  
-imagesphere3d, 261  
-img2str, 374  
-img2text, 374  
-img82hex, 374  
-index, 88  
-inn, 178  
-inpaint, 179  
-inpaint\_flow, 179  
-input, 26  
-int82float, 88  
-invert, 240  
-is\_3d, 374  
-is\_percent, 374  
-is\_windows, 374  
-isoline3d, 262  
-isophotes, 204  
-isosurface3d, 263  
-kaleidoscope, 334  
-keep, 36  
-kuwahara, 180  
-lab2lch, 110  
-lab2rgb, 110  
-lab82rgb, 111  
-label, 205  
-label\_fg, 206  
-label\_points3d, 264  
-laplacian, 180  
-lathe3d, 264  
-lch2lab, 111  
-lch2rgb, 111  
-lch82rgb, 111  
-le, 56  
-lic, 181  
-light3d, 265  
-light\_patch, 342  
-light\_relief, 321  
-lightrays, 320  
-line, 222  
-line3d, 266  
-linearize\_tiles, 308  
-lissajous3d, 266  
-local, 295  
-log, 58  
-log10, 59  
-log2, 60  
-lt, 57  
-luminance, 111  
-mad, 375  
-mandelbrot, 223  
-map, 90  
-map\_clut, 91  
-map\_sphere, 334  
-map\_tones, 182  
-map\_tones\_fast, 182  
-marble, 223

-max, 61  
-max\_d, 375  
-max\_h, 375  
-max\_patch, 206  
-max\_s, 375  
-max\_w, 375  
-max\_wh, 375  
-max\_whd, 375  
-max\_whds, 375  
-maze, 224  
-maze\_mask, 224  
-mdiv, 62  
-meancurvature\_flow, 183  
-med, 375  
-median, 183  
-min, 62  
-min\_d, 375  
-min\_h, 375  
-min\_patch, 206  
-min\_s, 375  
-min\_w, 375  
-min\_wh, 376  
-min\_whd, 376  
-min\_whds, 376  
-minimal\_path, 207  
-mirror, 135  
-mix\_channels, 91  
-mix\_rgb, 111  
-mmul, 64  
-mod, 63  
-mode3d, 267  
-moded3d, 268  
-morph, 357  
-mosaic, 322  
-move, 37  
-mse, 207  
-mul, 65  
-mul3d, 268  
-name, 37  
-negative, 92  
-neq, 67  
-noise, 92  
-noise\_hurl, 343  
-norm, 93  
-normalize, 94  
-normalize3d, 268  
-normalize\_local, 184  
-normalize\_sum, 94  
-normalized\_cross\_correlation, 184  
-object3d, 225  
-oct, 376  
-oct2dec, 376  
-old\_photo, 322  
-onfail, 295  
-opacity3d, 269  
-or, 67  
-orientation, 95  
-otsu, 95  
-output, 28  
-output\_pink3d, 362  
-outputn, 28  
-outputp, 28  
-outputw, 29  
-pack\_sprites, 226  
-padint, 376  
-parallel, 296  
-parametric3d, 270  
-patches, 208  
-path\_tmp, 376  
-path\_user, 376  
-pca\_patch3d, 270  
-pde\_flow, 185  
-pencilbw, 323  
-permute, 136  
-phase\_correlation, 185  
-piechart, 227  
-pink, 362  
-pink\_grayskel, 363  
-pink\_heightmaxima, 364  
-pink\_heightminima, 365  
-pink\_htkern, 365  
-pink\_lvkern, 366  
-pink\_reg\_minima, 366  
-pink\_skelcurv, 367  
-pink\_skelend, 368  
-pink\_skeleton, 368  
-pink\_skelpar, 369  
-pink\_wshed, 369

-pixelize, 343  
-plane3d, 271  
-plasma, 228  
-plot, 29  
-plot2value, 208  
-point, 228  
-point3d, 272  
-pointcloud, 209  
-pointcloud3d, 272  
-polar2complex, 96  
-polar2euclidean, 335  
-polaroid, 323  
-polka\_dots, 229  
-polygon, 230  
-pose3d, 273  
-poster\_edges, 324  
-pow, 68  
-pow2, 138  
-primitives3d, 273  
-print, 29  
-progress, 296  
-projections3d, 273  
-pseudogray, 112  
-psnr, 210  
-pyramid3d, 273  
-quadrangle3d, 274  
-quadratize\_tiles, 308  
-quantize, 96  
-quit, 297  
-quiver, 231  
-quote, 376  
-rainbow\_lut, 29  
-raindrops, 335  
-rand, 97  
-rectangle, 232  
-red\_eye, 186  
-region\_feature, 377  
-register\_nonrigid, 358  
-register\_rigid, 358  
-remove, 38  
-remove\_duplicates, 30  
-remove\_empty, 30  
-remove\_hotpixels, 186  
-remove\_opacity, 121  
-remove\_pixels, 187  
-repair, 188  
-repeat, 297  
-replace, 98  
-replace\_color, 112  
-replace\_inf, 98  
-replace\_nan, 99  
-replace\_seq, 99  
-reset, 377  
-resize, 137  
-resize2dx, 139  
-resize2dy, 140  
-resize3dx, 141  
-resize3dy, 141  
-resize3dz, 142  
-resize\_ratio2d, 139  
-return, 298  
-reverse, 39  
-reverse3d, 275  
-rgb2bayer, 113  
-rgb2cmy, 113  
-rgb2cmyk, 114  
-rgb2hsi, 115  
-rgb2hsi8, 115  
-rgb2hsl, 116  
-rgb2hsl8, 116  
-rgb2hsv, 117  
-rgb2hsv8, 117  
-rgb2lab, 118  
-rgb2lab8, 118  
-rgb2lch, 119  
-rgb2lch8, 119  
-rgb2luv, 119  
-rgb2srgb, 119  
-rgb2xyz, 120  
-rgb2xyz8, 120  
-rgb2ycbcr, 120  
-rgb2yuv, 121  
-rgb2yuv8, 121  
-ripple, 336  
-roddy, 29  
-rodilius, 325  
-rol, 69  
-ror, 70

-rorschach, 232  
-rotate, 142  
-rotate3d, 275  
-rotate\_tileable, 143  
-rotate\_tiles, 309  
-rotation3d, 276  
-rotoidoscope, 337  
-round, 100  
-roundify, 100  
-rows, 143  
-rprogress, 298  
-scale2x, 144  
-scale3x, 144  
-segment\_watershed, 210  
-select, 30  
-select\_color, 121  
-sepia, 122  
-set, 101  
-shade\_stripes, 344  
-shadow\_patch, 345  
-shared, 31  
-sharpen, 188  
-shift, 145  
-shift\_tiles, 309  
-shrink\_x, 146  
-shrink\_xy, 146  
-shrink\_xyz, 147  
-shrink\_y, 147  
-shrink\_z, 147  
-sierpinski, 233  
-sierpinski3d, 276  
-sign, 71  
-sin, 71  
-sinc, 72  
-sinh, 73  
-skeleton, 211  
-sketchbw, 326  
-skip, 298  
-slices, 147  
-smooth, 189  
-snapshot3d, 277  
-snowflake, 234  
-solarize, 122  
-solidify, 191  
-solidify\_linear, 191  
-solve, 241  
-solve\_poisson, 192  
-sort, 148  
-sort\_list, 40  
-sort\_str, 40  
-spec13d, 277  
-specs3d, 278  
-sphere3d, 279  
-spherical3d, 279  
-spiralbw, 234  
-spline, 235  
-spline3d, 280  
-split, 148  
-split3d, 281  
-split\_freq, 190  
-split\_opacity, 123  
-split\_tiles, 150  
-sponge, 327  
-spread, 345  
-sprite3d, 281  
-sprites3d, 282  
-sqr, 74  
-sqrt, 75  
-rand, 32  
-srgb2rgb, 123  
-ssd\_patch, 212  
-stained\_glass, 325  
-star3d, 282  
-stars, 326  
-status, 298  
-stencil, 328  
-stencilbw, 328  
-str, 377  
-str2hex, 378  
-strcat, 378  
-strcmp, 378  
-streamline3d, 283  
-stresc, 378  
-stripes\_y, 346  
-strlen, 378  
-strreplace, 378  
-structuretensors, 193  
-struncase, 378

-strver, 379  
-sub, 76  
-sub3d, 283  
-superformula3d, 284  
-svd, 241  
-symmetrize, 337  
-tan, 78  
-tanh, 79  
-taquin, 310  
-testimage2d, 32  
-tetris, 329  
-text, 235  
-text2img, 33  
-text3d, 285  
-text\_outline, 236  
-text\_pointcloud3d, 285  
-texturize3d, 286  
-texturize\_canvas, 346  
-texturize\_paper, 347  
-thinning, 212  
-threshold, 102  
-threshold2, 103  
-tic, 379  
-to\_a, 123  
-to\_color, 123  
-to\_colormode, 123  
-to\_gray, 123  
-to\_graya, 124  
-to\_pseudogray, 124  
-to\_rgb, 124  
-to\_rgba, 124  
-toc, 379  
-tones, 213  
-topographic\_map, 213  
-torus3d, 287  
-transfer\_colors, 124  
-transform\_polar, 338  
-transition3d, 359  
-transition\_plasma, 359  
-transpose, 242  
-triangle3d, 287  
-triangle\_shade, 237  
-trisolve, 242  
-truchet, 237  
-tunnel, 310  
-turbulence, 238  
-tv\_flow, 193  
-twirl, 338  
-type, 34  
-uncommand, 34  
-uniform\_distribution, 34  
-unroll, 151  
-unsharp, 194  
-unsharp\_octave, 194  
-update, 34  
-upscale\_smart, 151  
-vanvliet, 195  
-variance\_noise, 379  
-vector2tensor, 103  
-verbose, 35  
-version, 17  
-vignette, 347  
-volume3d, 288  
-wait, 35  
-warhol, 329  
-warn, 35  
-warp, 152  
-warp\_perspective, 339  
-water, 339  
-watermark\_fourier, 196  
-watermark\_visible, 348  
-watershed, 197  
-wave, 340  
-weave, 330  
-weird3d, 288  
-while, 299  
-whirls, 331  
-wind, 341  
-window, 35  
-x\_blobs, 360  
-x\_fire, 360  
-x\_fireworks, 360  
-x\_fisheye, 360  
-x\_fourier, 360  
-x\_histogram, 360  
-x\_hough, 360  
-x\_jawbreaker, 361  
-x\_life, 361

-x\_light, 361  
-x\_mandelbrot, 361  
-x\_minesweeper, 361  
-x\_minimal\_path, 361  
-x\_pacman, 361  
-x\_paint, 361  
-x\_plasma, 361  
-x\_quantize\_rgb, 361  
-x\_reflection3d, 362  
-x\_rubber3d, 362  
-x\_shadebobs, 362  
-x\_spline, 362  
-x\_tetris, 362  
-x\_tictactoe, 362  
-x\_whirl, 362  
-xor, 80  
-xyz2rgb, 125  
-xyz82rgb, 125  
-ycbcr2rgb, 125  
-yinyang, 239  
-yuv2rgb, 125  
-yuv82rgb, 125  
-zoom, 341

□ End of document.