

darktable 1.6

darktable 1.6

Copyright © 2010-2012 P.H. Andersson

Copyright © 2010-2011 Olivier Tribout

Copyright © 2012-2014 Ulrich Pegelow

Copyright © 2013-2014 Jérémy Rosen

The owner of the darktable project is Johannes Hanika. Main developers are Johannes Hanika, Henrik Andersson, Tobias Ellinghaus, Pascal de Bruijn and Ulrich Pegelow.

darktable has been developed with major contribution by Aldric Renaudin, Alexandre Prokoudine, Ammon Riley, Andrew Toskin, Andrey Kaminsky, Antony Dovgal, Artur de Sousa Rocha, Brian Teague, Bruce Guenter, Cherrot Luo, Christian Himpe, Christian Tellefsen, Dan Torop, David Bremner, Dennis Gnad, Dimitrios Psychogios, Eckhart Pedersen, Edouard Gomez, Frédéric Grollier, Gaspard Jankowiak, Ger Siemerink, Gianluigi Calcaterra, Guilherme Brondani Torri, Igor Kuzmin, Ivan Tarozzi, James C. McPherson, Jean-Sébastien Pédrón, Jérémy Rosen, Jesper Pedersen, Jochen Schröder, Johannes Schneider, José Carlos Casimiro, Jose Carlos Garcia Sogo, Josep Vicenç Moragues Pastor, Kaminsky Andrey, Kanstantsin Shautsou, Karl Mikaelsson, Loic Guibert, Marcel Laubach, Mauro Bartoccelli, Michal Babej, Michel Leblond, Mikko Ruohola, Milan Knížek, Moritz Lipp, Olivier Tribout, Pascal Obry, Pedro Côrte-Real, Petr Styblo, Richard Hughes, Richard Levitte, Richard Wonka, Robert Bieber, Roman Lebedev, Rostyslav Pidgorny, Sergey Pavlov, Simon Spannagel, Stuart Henderson, Tatica Leandro, Thomas Pryds, Victor Lamoine, Wolfgang Goetz, Wyatt Olson and many others.

darktable is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

darktable is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with darktable. If not, see *the GNU homepage* [<http://www.gnu.org/licenses/>].

The present user manual is under license *cc by-sa*, meaning *Attribution Share Alike*. You can visit *the creative commons page* [<http://creativecommons.org/about/licenses/>] to get more information.

Table of Contents

Preface to this manual	vii
1. Overview	1
1.1. Program invocation	3
1.1.1. darktable binary	3
1.1.2. darktable-cli binary	4
1.2. User interface	6
1.2.1. Views	6
1.2.2. Screen layout	6
1.2.3. Filmstrip	7
1.2.4. Preferences	7
1.3. darktable basic workflow	8
1.3.1. Importing images	8
1.3.2. Basic development steps	8
1.3.3. Exporting images	10
2. Lighttable	13
2.1. Overview	14
2.2. Lighttable concepts	16
2.2.1. Film rolls	16
2.2.2. Collections	16
2.2.3. Thumbnails	16
2.2.4. Star ratings and color labels	17
2.2.5. Filtering and sort order	18
2.2.6. Image grouping	18
2.2.7. Sidecar files	19
2.2.8. Importing sidecar files generated by other applications	20
2.2.9. Local copies	21
2.3. Lighttable panels	22
2.3.1. Import	22
2.3.2. Collect images	24
2.3.3. Recently used collections	26
2.3.4. Image information	26
2.3.5. Select	26
2.3.6. Selected image(s)	27
2.3.7. History stack	28
2.3.8. Styles	29
2.3.9. Geotagging	30
2.3.10. Metadata editor	32
2.3.11. Tagging	32
2.3.12. Export selected	33
3. Darkroom	37
3.1. Overview	38
3.2. Darkroom concepts	39
3.2.1. Pixelpipe, module order, and history stack	39
3.2.2. Interacting with modules	39
3.2.3. Module presets	41
3.2.4. Multiple instances	42
3.2.5. Blending	43
3.2.6. Blending operators	45
3.2.7. Drawn mask	47
3.2.8. Parametric mask	50
3.2.9. Combining drawn and parametric masks	53
3.2.10. Color management	54

3.3. Darkroom panels	57
3.3.1. Navigation	57
3.3.2. Snapshots	57
3.3.3. History stack	57
3.3.4. Global color picker	58
3.3.5. Mask manager	59
3.3.6. Histogram	61
3.3.7. Module groups	61
3.3.8. More modules	62
3.3.9. Bottom panel	63
3.4. Modules	65
3.4.1. Basic group	65
3.4.2. Tone group	74
3.4.3. Color group	81
3.4.4. Correction group	90
3.4.5. Effect group	100
3.5. Examples	117
3.5.1. Converting to black and white	117
3.5.2. Cross-processing	118
3.5.3. Cyan toned image	119
3.5.4. Removal of red-eye effect	120
4. Tethering	123
4.1. Overview	124
4.1.1. Tethering	124
4.2. Tethering panels	125
4.2.1. Session	125
4.2.2. Live view	125
4.2.3. Camera settings	125
4.3. Examples	126
4.3.1. Studio setup with screening	126
4.3.2. Capturing a timelapse	126
4.4. Troubleshoot	127
4.4.1. Verify that your camera is supported	127
4.4.2. So, now what?	127
5. Map	129
5.1. Overview	130
5.1.1. Center map view	130
5.2. Map panels	131
5.2.1. Left panels	131
5.2.2. Find location	131
5.2.3. Map settings	131
5.2.4. Tagging	132
6. Slideshow	133
6.1. Overview	134
6.2. Usage	135
7. Preferences and settings	137
7.1. GUI options	138
7.2. Core options	141
7.3. Session options	143
7.4. Shortcuts	145
7.5. Presets	148
8. Scripting with Lua	151
8.1. Lua usage	152
8.1.1. Basic principles	152
8.1.2. A simple lua example	152

8.1.3. Printing labeled images	152
8.1.4. Adding a simple shortcut	154
8.1.5. Exporting images with Lua	156
8.1.6. Sharing scripts	157
8.1.7. Calling Lua from Dbus	159
8.1.8. Using Darktable from a lua script	159
8.2. Lua API	160
8.2.1. darktable	160
8.2.2. types	188
8.2.3. events	209
8.2.4. attributes	213
8.2.5. system	213
9. Special topics	215
9.1. darktable and memory	216
9.1.1. Total system memory	216
9.1.2. Available address space	216
9.1.3. Memory fragmentation	216
9.1.4. Further limitations	217
9.1.5. Setting up darktable on 32-bit systems	217
9.1.6. darktable on 64-bit systems	218
9.2. darktable and OpenCL	219
9.2.1. The background	219
9.2.2. How OpenCL works	219
9.2.3. How to activate OpenCL in darktable	219
9.2.4. Setting up OpenCL on your system	220
9.2.5. Possible problems and solutions	221
9.2.6. Setting up OpenCL for AMD/ATI devices	222
9.2.7. OpenCL performance optimization	223
9.2.8. Multiple OpenCL devices	225
9.2.9. OpenCL still does not run for me!	227
Index	229

Preface to this manual

User manual version and applicable darktable version are listed below:

	version	date
user manual	1.6.0	November 2014
darktable	1.6	November 2014

Translations of this manual to local languages are brought to you by Federico Bruni, Victor Lamoine, and Michel Leblond.

Many thanks to all contributors to this user manual. Special thanks for proof reading, style improvement, constructive criticism, and valuable contributions go to Colin Adams, Mark Garrow, Simon Harhues, István Kovács, Michel Leblond, Rudolf Martin, Ammon Riley, Rob Z. Smith, Andrew Toskin, and David Vincent-Jones.

Chapter 1. Overview

darktable is an open source photography workflow application and RAW developer, a virtual lighttable and darkroom for photographers.

It manages your digital negatives in a database, lets you view them through a zoomable lighttable and enables you to develop raw images and enhance them.

General Features

- darktable runs on GNU/Linux / GNOME, Mac OS X / macports and Solaris 11 / GNOME.
- Fully non-destructive editing.
- All darktable core functions operate on 4x32-bit floating point pixel buffers for high accuracy processing, preventing banding and color breaks.
- darktable makes heavy use of *Streaming SIMD Extensions 2* (SSE2) instructions of the CPU to speed up processing. In fact, darktable will only run on a CPU that supports SSE2.
- GPU acceleration via OpenCL (runtime detection and enabling).
- Most image processing is done in CIE Lab color space, which is much larger than the gamut of modern displays, printers or even human vision.
- Full color managed display with softproofing and gamut-check. Built-in ICC profile support for export: sRGB, Adobe RGB, XYZ and linear RGB.
- A collect module allows you to execute flexible database queries, search your images by tags, image rating (stars), color labels and many more. Filtering and sorting your collections within the base query or simple tagging by related tags are useful tools in your every-day photo workflow.
- Import a variety of standard, raw and high dynamic range image formats (e.g. JPG, CR2, OpenEXR, PFM, ...).
- darktable has a zero-latency fullscreen, zoomable user interface through multi-level software caches.
- Tethered shooting.
- The powerful export system supports Picasa webalbum, flickr upload, disk storage, 1:1 copy, email attachments and can generate a simple html-based web gallery. darktable allows you to export to low dynamic range (JPEG, JPEG2000, PNG, TIFF), 16-bit (PPM, TIFF), or linear high dynamic range (PFM, EXR) images.
- darktable uses both XMP sidecar files as well as its fast database for saving metadata and processing settings. All Exif data is read and written using libexif2.
- darktable comes with more than 50 image operation modules which cover everything from basic operations, tonal value changes, color manipulation, correction of common image defects to artistic effects.
- Many darktable modules can be combined with blending operators for even more development options.

- A powerful mask feature gives you fine control on module's effect to different parts of an image. You can at your choice draw a mask using various shapes or define a parametric mask based on pixel values.
- Most modules can exist as multiple instances. Together with the mask feature, you can let an operation have different effects on different parts of the image.
- darktable introduces a highly efficient, yet simple "single-click" denoiser that always just works™. It's designed as a module where the denoising performance only depends on camera and ISO setting. A database of profiles contains parameters for over 100 popular camera models.
- darktable comes with a versatile scripting interface for functionality enhancement using Lua as a scripting language.

1.1. Program invocation

darktable comes with two binaries: the standard GUI variant which is started by calling `darktable` and a command line interface variant which is started by calling `darktable-cli`.

1.1.1. darktable binary

This binary starts darktable with its GUI and full functionality; it is the standard way of using darktable.

`darktable` is called with the following command line parameters:

```
darktable [-d {all,cache,camctl,control,dev,fswatch,
             input,lighttable,masks,memory,nan,opencl,
             perf,pwstorage,sql}]
          [IMG_1234.{RAW,..}|image_folder/]
          [--version]
          [--disable-opencl]
          [--library <library file>]
          [--datadir <data directory>]
          [--moduledir <module directory>]
          [--tmpdir <tmp directory>]
          [--configdir <user config directory>]
          [--cachedir <user cache directory>]
          [--localedir <locale directory>]
          [--luacmd <lua command>]
          [--conf <key>=<value>]
```

All parameters are optional; in most cases users will start darktable without any additional parameters in which case darktable uses suitable defaults.

`-d`

This option enables debug output to the terminal. There are several subsystems of darktable and debugging of each of them can be activated separately. You can use this option multiple times if you want debugging output of more than one subsystem.

`IMG_1234.{RAW,..}|image_folder/`

You may optionally supply the filename of an image or the name of a folder containing image files. If a filename is given darktable starts in darkroom view with that file opened. If a folder is given darktable starts in lighttable view with the content of that folder as the current collection.

`--version`

This option causes darktable to print its version number, a copyright notice, some other useful information, and then terminate.

`--disable-opencl`

This option prevents darktable from initializing the OpenCL subsystem. Use this option in case darktable crashes at startup due to a defective OpenCL implementation.

`--library <library file>`

darktable keeps image information in an sqlite database for fast access. The default location of that database file is `“$HOME/.config/darktable/library.db”`. You may give an alternative location, e.g. if you want to do some experiments without compromising your original `library.db`. If the database file does not exist, darktable creates it for

you. You may also give “:memory:” as a library file in which case the database is kept in system memory – all changes are discarded when darktable terminates.

`--datadir <data directory>`

This option defines the directory where darktable finds its runtime data. The default place depends on your installation. Typical places are “/opt/darktable/share/darktable/” and “/usr/share/darktable/”.

`--moduledir <module directory>`

darktable has a modular structure and organizes its modules as shared libraries for loading at runtime. With this option you tell darktable where to look for its shared libraries. The default place depends on your installation; typical places are “/opt/darktable/lib64/darktable/” and “/usr/lib64/darktable/”.

`--tmpdir <tmp directory>`

The place where darktable stores its temporary files. If this option is not supplied darktable uses the system default.

`--configdir <config directory>`

This option defines the directory where darktable stores the user specific configuration. The default place is “\$HOME/.config/darktable/”.

`--cachedir <cache directory>`

darktable keeps a cache of image thumbnails for fast image preview and of pre-compiled OpenCL binaries for fast startup. By default the cache is located in “\$HOME/.cache/darktable/”. There may exist multiple thumbnail caches in parallel – one for each library file.

`--localedir <locale directory>`

The place where darktable finds its language specific text strings. The default place depends on your installation. Typical places are “/opt/darktable/share/locale/” and “/usr/share/locale/”

`--luacmd <lua command>`

A string containing lua commands to execute after lua initialization. These commands will be run after your “luarc” file.

If lua is not compiled in, this option will be accepted but won't do anything

`--conf <key>=<value>`

darktable supports a rich set of configuration parameters which the user defines in “darktable.rc” – darktable's configuration file in the user config directory. You may temporarily overwrite individual settings on the command line with this option – however, these settings will not be stored in “darktable.rc”.

1.1.2. darktable-cli binary

This binary starts the command line interface variant of darktable which allows exporting images.

`darktable-cli` is called with the following command line parameters:

```
darktable-cli <input file>
               [<xmp file>]
               <output file>
               [--width <max width>]
               [--height <max height>]
```



```
[--bpp <bpp>]
[--hq <0|1|true|false>]
[--verbose]
[--core <darktable options>]
```

The user needs to supply an input filename and an output filename. All other parameters are optional.

<input file>

The name of the input file to export.

<xmp file>

The optional name of an XMP sidecar file containing the history stack data to be applied during export. If this option is not given darktable will search for an XMP file that belongs to the given input file.

<output file>

The name of the output file. darktable derives the export file format from the file extension.

--width <max width>

This optional parameter allows to limit the width of the exported image to that number of pixels.

--height <max height>

This optional parameter allows to limit the height of the exported image to that number of pixels.

--bpp <bpp>

An optional parameter to define the bit depth of the exported image; allowed values depend on the file format. Currently this option is not yet functional. If you need to define the bit depth you need to use the following workaround:

```
--core
--conf plugins/imageio/format/<FORMAT>/bpp=<VALUE>
```

where <FORMAT> is the name of the selected output format.

--hq <0|1|true|false>

A flag that defines whether to use high quality resampling during export (see Section 7.2, "Core options"). Defaults to true.

--verbose

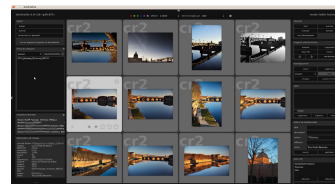
Enables verbose output.

--core <darktable options>

All command line parameters following "--core" are passed to the darktable core and handled as standard parameters. See Section 1.1.1, "darktable binary" for a detailed description.

1.2. User interface

This section describes the layout of the user interface.



1.2.1. Views

darktable consists of several views or modes. There are five available views as described in this section. You can switch between views by clicking the view name at the top of the right panel – the active view is highlighted – or by using one of the key accelerators:

<i>l</i>	switches to lighttable
<i>d</i>	switches to darkroom
<i>t</i>	switches to camera tethering
<i>m</i>	switches to map
<i>s</i>	switches to slideshow

1.2.1.1. Lighttable

The lighttable view is where images and film rolls are managed. It's also where you rate images, add tags and colorlabels, and export images among other actions (see Chapter 2, *Lighttable*).

1.2.1.2. Darkroom

In the darkroom view you develop a single image using the available modules (see Chapter 3, *Darkroom*).

1.2.1.3. Tethering

This view is for shooting with the camera connected to the computer and remotely capturing images that will be downloaded and shown on computer screen (see Chapter 4, *Tethering*).

1.2.1.4. Map


This view shows images with geo-tag data on a map and allows manually geo-tagging new images (see Chapter 5, *Map*).

1.2.1.5. Slideshow

This view shows images as a slideshow, processing them on-the-fly (see Chapter 6, *Slideshow*).

1.2.2. Screen layout

The general screen layout of all views is similar. There is a center area which contains most of the relevant information of that view. Then there are panels left, right, top and bottom to the center area. The left panel typically has an informational purpose. The right panel offers functions to modify an image. The top and bottom panel give access to several

settings and shortcuts. Each of the panels can be collapsed or expanded by pressing a triangle like , located close to the panel.

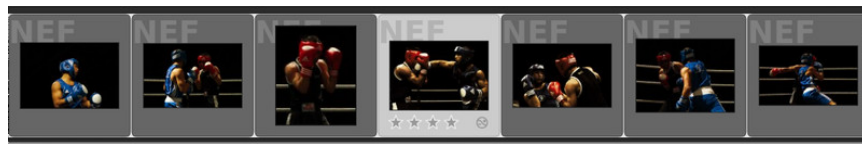
By pressing the *TAB* key all panels get collapsed, allowing the center area to occupy all available space. Pressing *TAB* again brings you back to the previous view.

Fullscreen view can be toggled by pressing *F11*.


darktable's contrast can be changed by using *F7* and *F8* and darktable's lightness by using *F9* and *F10*.

1.2.3. Filmstrip

The filmstrip along the bottom shows the same images as lighttable, with respect to filters and sort order. It is turned on/off with key accelerator *ctrl-f*. You can navigate along the filmstrip by scrolling with the mouse wheel. The filmstrip allows you to interact with images while you are not in lighttable mode. For example, you can, while developing an image in darkroom mode, switch to another image to develop, by double clicking the thumbnail in the filmstrip. You can also rate the images as you do in lighttable, copy/paste history stack, etc.



1.2.4. Preferences

A button  located in the upper panel allows you to define various parameters which control darktable's behavior.

The options are fairly self-explanatory. If you need more information, hover the mouse cursor over the text label or entry box, to display a popup tool-tip. All configuration parameters are explained in Chapter 7, *Preferences and settings*.

1.3. darktable basic workflow

This section describes a typical darktable workflow which novice users may take as a starting point. We describe how to get an image into darktable, the basic steps of a raw development workflow and how to export the final result.

1.3.1. Importing images

To begin with darktable, you first need to import images. The import module is in the left pane of the lighttable view (Section 2.3.1, “Import”). You can either import from the filesystem or, if darktable supports your camera model, directly from camera.

1.3.1.1. Importing images from filesystem

When importing from disk, you can import either a single image or a folder. darktable will analyse its content, detect images that are already imported and only import new images.

1.3.1.2. Importing from camera

Connect your camera to your system. If your distribution tries to automount it, select the option to abort the mount operation. Otherwise the camera will be locked and not accessible from within darktable. If you don't see your camera in the import pane, hit the “scan for devices” button. Your camera will then appear in the same pane with additional choices: *import* and *tethering*.

1.3.2. Basic development steps

1.3.2.1. Introduction

This section will guide you through the basics of developing an image in the darkroom view.

To begin, open an image in darkroom mode by double clicking an image thumbnail on the lighttable. The darkroom mode is where the actual adjustments for an image are made, where an arsenal of modules are at hand to help you reach your goal.

Each change made on a module while developing an image is turned into a *history stack* item. The history is stored in a database and in an XMP sidecar file for the specific image.

All changes are stored automatically. You can safely leave darkroom mode or quit darktable at any time and come back later to continue your work. That said darktable does not need a “save” button and it does not have one.

On the left panel in darkroom mode is the *history stack*, showing changes starting from bottom, and building up with each change made to the image. You can select a point in this history to show how the image looked at that point, for comparison of changes. The stack can be compressed: it will be optimized and redundant changes will be discarded. When you think you are done and are happy with what you have done, just compress the history stack.

darktable ships with a number of modules, arranged into groups. These module groups are accessed via toggle buttons in the right panel, just under the histogram. There are also two special module groups named “active” and “favorites”, which only show modules enabled in the history for the current image, and modules selected as a favorite, respectively. Marking a module as a favorite is done in the *more modules* dialog (Section 3.3.8, “More modules”), at the bottom of the right panel, by clicking a module until a star is displayed in front of the icon.

1.3.2.2. White balance

The *white balance* module controls the white balance or color temperature of the image. It's always enabled and reads its default values from camera metadata embedded in the image. The most common change is fine-tuning the white balance, which is done using the "temperature" slider. Moving this slider left will make the color balance cooler, and moving it right will make it warmer.

1.3.2.3. Exposure correction

The *exposure* module is probably the most basic module of them all. Exposure is fine-tuned either by using the slider, or by dragging with the mouse in the *histogram*. You can also boost the black level to enhance contrast; but be careful: use small amounts, like steps of 0.005. There is also an auto-correct feature.

1.3.2.4. Noise reduction

The best starting point for noise reduction is *profiled denoise*. This module offers an almost "single-click" solution to fight noise. From a user perspective the effect only depends on camera type and ISO value, both derived from EXIF data. All other settings are taken from a database of noise profiles that the darktable team has collected – now covering already over 100 popular camera models. In addition you have several other options in darktable to reduce noise. There is *raw denoise*, *denoising based on bilateral filter*, *denoising based on non-local means*, and *equalizer*, which is based on wavelets. If your camera is not yet supported by *profiled denoise*, *denoising based on non-local means* is probably the most convenient, as it allows you to treat color and luminance noise separately.

1.3.2.5. Fixing spots

Sometimes you will need to remove spots caused by sensor dirt. The *spot removal* module is at hand and can also correct other disturbing elements like skin blemishes. If your camera has stuck pixels or tends to produce hot pixels at high ISO values, or longer exposure times, have a look at the *hot pixels* module for automatic correction.

1.3.2.6. Geometrical corrections

Quite frequently you want to only show part of the captured scene in your image, e.g. to take away some disturbing feature close to the frame. In other cases, the horizon in the image may need levelling, or there are perspective distortions. All this can be corrected in the *crop and rotate* module. If you need to correct typical camera lens flaws like cushion distortion, transversal chromatic aberrations or vignetting, there is a *lens correction* module.

1.3.2.7. Bringing back detail

Digital RAW images often contain more information than you can see at first sight. Especially in the shadows of an image, there are lots of hidden details. The *shadows and highlights* module helps bring these details back into visible tonal values. Structural details in fully blown-out highlights, by nature of the digital sensor, can not be recovered. However, you can correct unfavorable color casts in these areas with the *highlight reconstruction* module.

1.3.2.8. Adjusting tonal values

Almost every workflow is likely to include adjusting the image's tonal range. darktable offers several alternative modules to take care of that. The most basic one is the *contrast brightness saturation* module. In the *tone curve* module, tonal values are adjusted by con-

structuring a gradient curve. The *levels* module offers a concise interface, with three markers in a histogram. In addition, there is a *zone system* module which allows control over tonal values by zones, inspired by the work of Ansel Adams.

1.3.2.9. Enhancing local contrast

Local contrast enhancement can emphasize detail and clarity in your image. Carefully used, it can give your photograph the right pop. darktable offers several modules for this task. The *local contrast* module is easy to handle, with just a few parameters. A much more versatile, but also more complex technique, is offered by the *equalizer* module. Have a look at its presets, to get a feeling for how it works. Equalizer is darktable's "Swiss Army Knife" for many adjustments where spatial dimension plays a role.

1.3.2.10. Color adjustments

darktable offers many modules for adjusting colors in an image. A very straightforward technique is implemented in the *color correction* module. Use it to give an image an overall tint or to adjust overall color saturation. The *color zones* module offers a much finer control to adjust saturation, or lightness, and even hue, in user defined zones. darktable's *tone curve* module – in addition to the classical adjustment of tonal values – gives you fine control over the colors in an image. Finally, if you intend to convert an image into black & white, a good starting point, with an easy to use and intuitive user interface, is offered by the *monochrome* module. Alternatively, you might consider using darktable's *channel mixer*.

1.3.2.11. Sharpening

If you start your workflow from a RAW image, you will need to have your final output sharpened. The *sharpen* module can do this with the classical USM (unsharp mask) approach, available in most image processing software. Another very versatile way to enhance edges in an image is offered by the *highpass* module, in combination with darktable's rich set of blending operators.

1.3.2.12. Artistic effects

darktable comes with a rich set of artistic effect modules. To name just a few: with the *watermark* module you add an individual watermark to your image. The *grain* module simulates the typical noise of classical analogue footage. Use the *color mapping* module to transfer the look and feel of one color image onto another. darktable's *low light* module allows to simulate human vision to make lowlight pictures look closer to reality. The *graduated density* filter adds a neutral or colored gradient to your image for exposure and color correction.

1.3.3. Exporting images

Changes to an image are not saved as in a regular image editor. darktable is a non-destructive editor, which means all changes are stored in a database, and the original image is untouched. Therefore, you need to export images to bake the processing options into an output file that can be distributed outside of darktable.

Images are exported from the lighttable view, using the *export selected* dialog in the right panel (Section 2.3.12, "Export selected"). In general, export means: save my developed RAW image as a JPEG.

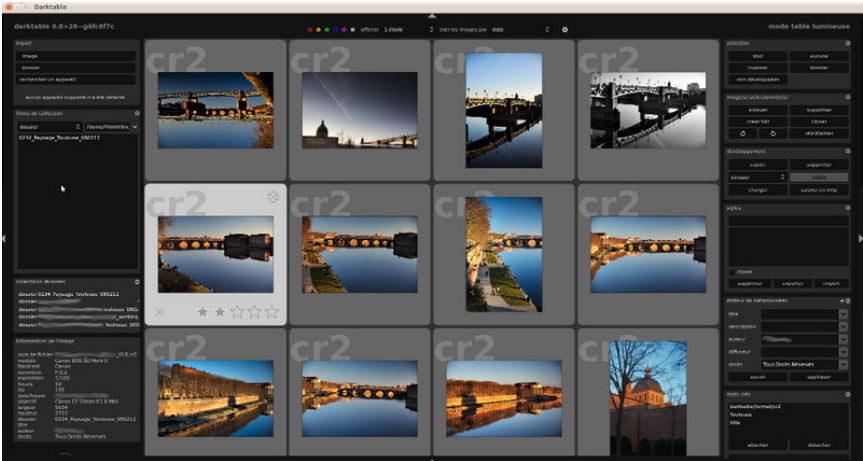
The export is modularized into *storage* and *format*. darktable ships with several storage modules such as *save on disk*, various webalbums, a LaTeX photo book template and more.

Format modules are the actual image formats such as JPEG, PNG, TIFF, OpenEXR and more.


Select images on the lighttable, choose the target storage and format, and set the maximum width and height image restraints. This means that none of the images will be bigger than any of the width/height restraints and hit the export button. Leave the width and height restraints at zero, if you want the original resolution.

Chapter 2. Lighttable

The lighttable is where you manage all your images, ratings, export and much more.





2.1. Overview

In the central view, your images are shown as thumbnails, surrounded by a frame. When the mouse is over an image, its rating and color labels are shown in the frame, along with an indicator  of whether the image has already been altered in darkroom. Also, when the mouse hovers over an image frame, image information (EXIF data, metadata) is shown in the *image information* panel at the bottom left.



While the mouse is over an image frame, there are a number of actions you can perform. Here is a table of keyboard shortcuts and assigned actions.

<code>0 – 5</code>	set the rating of the image; if an image has 1 star and you hit the <code>1</code> key, the image will be unrated. Pressing <code>r</code> rejects the image.
<code>F1 – F5</code>	set a color label
<code>ctrl-c</code>	copy the history stack
<code>ctrl-v</code>	paste the copied history stack
<code>d</code>	open in darkroom view for developing
<code>z</code>	fully zoom into the image while the key is pressed
<code>ctrl-z</code>	fully zoom into the image and show areas in focus

The overlay button  located in the upper panel activates the permanent display of star ratings and image-changed indicator  on all thumbnails. By default these overlays are only visible on the thumbnail under the mouse cursor. An overlay button is also available in the other views where it affects the filmstrip accordingly (see Section 1.2.3, “Filmstrip” and Section 3.3.9.4, “Filmstrip”).

In the bottom panel you have an option to choose between zoomable lighttable view or filemanager view of the thumbnails. In zoomable lighttable view, scroll with your mouse wheel to zoom in and out. Moving the mouse while *pressing the left mouse button* allows you to navigate through your collection. In filemanager view, you can change the number of images in each row, using the slider next to the filemanager option, or by using `ctrl-(mouse wheel)`. Use your mouse wheel to navigate through your collection.

While in filemanager mode, you can scroll (not select) up and down through your collection using `↑/↓`. In zoomable lighttable `←/→/↑/↓` allow you to move left/right/up/down through your collection. Pressing `g` goes to the top, `shift-g` to the bottom.

To locate where you are in a collection, there are indicators at the extreme borders of the window: left/right for your position when you are in filemanager mode, left/right and top/bottom for your vertical and your horizontal position, respectively, when you are in the zoomable lighttable view.

While holding down the `z` key a fully zoomed preview of the image under the mouse cursor is displayed. You can use this feature for a quick inspection of image quality while rating and selecting images.

Holding down the `ctrl-z` key fully zooms into the image and additionally activates an analysis for focus regions. Areas of high sharpness are indicated by a red border – the higher

the color intensity the better the sharpness. In case that no area of *high* sharpness is detected darktable indicates areas of *moderate* sharpness with a blue border. For this tool to work the input image needs to hold an embedded JPEG thumbnail which is the case for most RAW files.

Sometimes pressing *z* or *ctrl-z* may not reveal an immediate effect – in that case please click into the center area and press the corresponding key again.

Fully zoomed-in image view while holding down the *ctrl-z* key with indication of the areas in focus. Focus detection is based on an embedded JPEG thumbnail of the original RAW file independent of any processing steps within darktable.



2.2. Lighttable concepts

This section explains some of the underlying concepts on how darktable organizes images in the lighttable.

2.2.1. Film rolls

The basic element for organizing images in darktable is called a film roll – a kind of virtual folder. Whenever you import images from disk, the images are organized in a film roll whose name is derived from the name of the disk folder. Re-importing a disk folder will add any new images to the existing film roll; images already present in the film roll are not touched.

It is important to note that importing images in darktable does not involve a physical copy step. Importing a folder into darktable is therefore *not* a backup operation of that folder.

2.2.2. Collections

darktable offers a versatile feature to organize your images according to various user defined selection criteria. A set of images which is defined by a specific combination of selection criteria is called a collection. The most basic kind of collection is a film roll – covering all the images which have been imported from a specific folder on disk.

You can easily construct other kinds of collections based on various image attributes like EXIF data, filename, tags etc. Multiple criteria can be logically combined to narrow or extend your collection (see Section 2.3.2, “Collect images”).

darktable keeps a list of the most recently used collections for quick access (see Section 2.3.3, “Recently used collections”).

2.2.3. Thumbnails

Each image of the current collection is represented by a thumbnail in the lighttable view. darktable keeps a cache of thumbnails on disk, a so-called mipmap cache, the size of which can be adjusted in the core preferences dialog (see Section 7.2, “Core options”).

2.2.3.1. Thumbnail creation

Thumbnails get created whenever darktable imports an image for the first time, after an image has been modified in the darkroom, or when revisiting an “old” image whose thumbnail has already been dropped from the cache.

When darktable imports an image for the first time, there are two possible sources from where to take a thumbnail. darktable can either try to extract an embedded thumbnail out of the input image – most RAW files contain these kind of thumbnails generated by the camera – or process the image by itself using default settings. You can define how darktable gets its thumbnails in the gui preferences dialog (see Section 7.1, “GUI options”).

Extracting an embedded thumbnail from the input image has the advantage of being very fast. However, as those thumbnails were generated without knowing your display profile, the colors may be rendered incorrectly. You may notice differences when you open the image in the darkroom view, where colors are corrected for your monitor.

As the thumbnail cache has a pre-defined maximum size it will eventually get filled up. Then if new thumbnails are added, old ones need to be dropped. This typically passes unnoticed to the user unless a too small cache size is selected. In that case you might observe adverse effects like continuous regeneration of thumbnails each time you move in your collection, flickering of thumbnail images, or even darktable becoming unresponsive

while continuously generating new thumbnails and dropping them again. A good choice of the cache size is 512MB or higher. Please mind that the inherent limits of *32-bit systems* will force you to go for a much lower cache size (see Section 9.1, “darktable and memory” for more details on these limitations).

2.2.3.2. Skulls

If for some reason darktable is not able to generate a thumbnail, it displays a skull. Don't panic!



There are three main reasons for this to happen.


One possible cause is that the input image has been renamed or physically deleted from disk. darktable remembers all images ever imported, as long as they have not been removed from your database. In case darktable wants to create a thumbnail but is not able to open the input file, a skull is displayed instead. Users are advised to remove images from the database (see Section 2.3.6, “Selected image(s)”) *before* physically removing them from disk. Alternatively you may occasionally run the script `purge_non_existing_images.sh` from darktable's toolset to clean-up your database.

Another possible cause is sometimes darktable encounters an input image having an extension that seems valid for darktable but which has a file format that darktable does not support yet. darktable tries to process the image but is not able to get the job done.

The third possible cause for getting skulls is memory shortage: If darktable runs out of memory while generating a thumbnail, it will warn you and display a skull – this can happen if darktable is run with suboptimal settings on a 32-bit system. Please consult Section 9.1, “darktable and memory” for more information.

2.2.4. Star ratings and color labels

Star ratings and color labels help you to sort and rank images according to your own criteria. An image's star rating and its color labels are displayed in the thumbnail.

You can give a rating from zero to five stars to an image. The quality criteria which lead to a rating are fully up to you. Whenever you import images, each image receives a default rating which you can define in GUI options (see Section 7.1, “GUI options”). You can later revise the rating at any time. You can also mark an image as “rejected” by pressing the  icon or typing the *r* key. This will remove all stars. You can reverse the rejection by applying a new star rating.

There are several ways to change a rating. While hovering the cursor over an image thumbnail, you can press a number key *0–5* for the number of stars, or type *r* to “reject” an image; this is probably the fastest way when rating your images on first inspection of a film roll.

You can also directly click on the stars displayed in the thumbnails; click the *x* to reject, click the fifth star to get a rating of five stars, etc. Clicking either the *x*



or the first star for a second time resets the image rating to unranked, or zero stars.

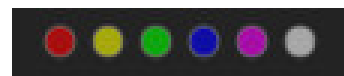
To rate one or more images at once, select those images (see Section 2.3.5, “Select”) and then press the key for your rating, or click the desired star rating in *the bottom panel* of the lighttable view.



Color labels are another way to sort images, and can be used as an alternative to star ratings or work alongside star ratings. Each image can carry any combination of one or more color labels in red, yellow, green, blue, and purple.

You can set the color labels for a single image by hovering your cursor over the thumbnail and pressing the function keys *F1 – F5*, which correspond with the labels in the order given above.

To toggle the color labels of one or more images, select the desired images (see Section 2.3.5, “Select”) and then press the corresponding color button in the bottom panel. To remove all labels from the selected images, press the grey button.



2.2.5. Filtering and sort order

The filtering and sort order of images in the lighttable view are controlled in the top panel.



With filtering you limit the number of displayed images in your current collection (see also Section 2.3.2, “Collect images”). Filtering is mainly based on the star ratings of your images. A typical filtering rule will show all images that have a rating at or above the given number of stars (from one to five). The comparison operators is not restricted to “≥”. By clicking on the operator field you toggle between “≥”, “>”, “≠”, “<”, “≤”, and “=”.

Alternatively you can make darktable display “all” images, only the “unstarred” ones, only the “rejected” images and all but the rejected ones.

Images in the lighttable view can be displayed in different sort orders, depending on “filename”, “time” (when the photo was taken), “rating” (i.e. stars), “id” (an internal ordering number of darktable), or “color labels”. You can sort in reversed order by toggling the triangle button to the right of the “sort by” combobox.

2.2.6. Image grouping

Grouping images helps improve structure and clarity of your image collection when displayed in lighttable view.

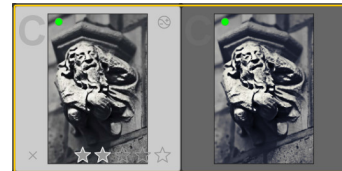
You can combine images into a group by selecting them, and clicking the “group” button in the *selected image(s)* panel (Section 2.3.6, “Selected image(s)”), or by typing *ctrl-g*. Likewise, you can remove selected images from a group by clicking the “ungroup” button, or typing *shift-ctrl-g*. Images generated by duplicating an existing one, are automatically grouped. If you import images from the file system or camera, images with the same base name, but different extensions (eg. IMG_1234.CR2 and IMG_1234.JPG), will form a group.

Images which are members of a group are labeled with a “G” symbol in their thumbnails.



The group button **G** in the top panel of the lightroom view toggles grouping mode on and off. If grouping is off, each image is displayed as an individual thumb. If grouping is on, images of a group are collapsed, which means they are represented by a single thumbnail. This thumbnail you see is called the group head. If you press the “G” symbol in the group's thumbnail, only this group gets expanded; if you then expand another group, the first group collapses. To collapse an expanded group again, just click on the “G” symbol of its group head.

An expanded group in the filemanager mode of light-table view is indicated by an orange frame which appears as soon as your mouse pointer hovers over one of the images.



You can define which image constitutes the group head, while in an expanded view of a group, clicking on the “G” symbol of the desired image.

If you collapse an image group and then enter darkroom mode (e.g., by double-clicking on the thumbnail), the *group head* will be opened for developing.

Image groups are a convenient way to protect an existing history stack against unintentional changes. Suppose you have just finalized an image and want to protect its current version: all you need to do is select the image, click “duplicate” in the *selected images* panel, and make sure that grouping is switched on and that the group is collapsed. Now, whenever you open the image group again in darkroom, only the group head will be altered. The underlying duplicate remains unchanged.

Please note that “duplicating images” here only generates a copy of your history stack, stored in another small XMP file. There is still only one RAW file, so you don't waste a lot of disk space.

2.2.7. Sidecar files

darktable is a non-destructive image editor. This means that darktable opens images read-only. Any newly added metadata, tags, and parameters of image operations (the “history stack”) are stored in separate *.xmp* files, so-called sidecars, allowing you to store information about the images as well as the full editing history without touching the original RAW files. When you import an image into darktable for the first time, an XMP file with default settings is generated automatically.

For a given source image, multiple editing versions, called duplicates, can co-exist, sharing the same input (RAW) data but each having their own metadata, tags and history stack. Each duplicate is represented by a separate XMP sidecar file with a filename construct-

ed in the form “<basename>_nn.<extension>.xmp”, where *nn* represents the (minimum two-digit) version number of that edit. Information for the initial edit – the “duplicate” with version number zero – is stored in the sidecar file “<basename>.<extension>.xmp”. The version number of a duplicate is displayed in the image information panel in each of darktable’s views (see an example in Section 2.3.4, “Image information”).

Sidecar files automatically synchronize with your work without the need to press a “save” button. When backing up your data, make sure you also keep the XMP files, as these are needed to fully reconstruct your work in case of a disaster.

In addition to the sidecar files, darktable keeps all image-related data in its database for fast access. An image can only be viewed and edited from within darktable if its data is loaded in that database. This automatically happens when you first import an image or at any later time by re-importing it (see Section 2.3.1, “Import”). In the latter case the database gets updated with data that darktable finds in the sidecar files belonging to that image.

Once an image has been imported into darktable the database entries take precedence over the XMP file. Subsequent changes to the XMP file by any other software are not visible to darktable – any changes get overwritten the next time darktable synchronizes the file. This behavior can be changed in the preferences dialog (see Section 7.2, “Core options”). On request darktable looks for updated XMP files at startup and offers a choice whether to update the database or overwrite the XMP file.

2.2.8. Importing sidecar files generated by other applications

When importing an image, darktable automatically checks if this is accompanied by a sidecar file. Other than the formats “<basename>.<extension>.xmp” and “<basename>_nn.<extension>.xmp” darktable also checks for the presence of a file in the form “<basename>.xmp”. darktable’s own sidecar files are always stored in the first format – the latter one would only be read – not overwritten.

At present, darktable is able to deal with the following metadata of Lightroom generated sidecar files during the import phase:

- tags and hierarchical tags
- color labels
- ratings
- GPS information

In addition, darktable has been designed to help migrate some image operations from specific other applications. The aim is not to make darktable a drop-in replacement for any other software; it’s just meant to help you recover part of the work you have invested into your image in case you migrate to darktable. It is very important to understand that this import process will never give identical results. The underlying development engines are very different from application to application, and additionally depend a lot on the specific image. In some cases, it will probably be close, and in some cases, the development will need manual adjustment in darktable.

The migration happens automatically when entering the darkroom view, provided that a corresponding XMP sidecar is found.

At present, darktable is able to deal with the following development steps from Lightroom-generated XMP files (with the corresponding darktable module in parentheses):

- crop and rotate (*crop and rotate*)
- black level (*exposure*)
- exposure (*exposure*)
- vignette (*vignette*)
- clarity (*local contrast*)
- tone curve (*tone curve*)
- HSL (*color zones*)
- split toning (*split toning*)
- grain (*grain*)
- spot removal (*spot removal*)

2.2.9. Local copies

Many users have huge image collections which they store on extra hard drives in their desktop computer, or on an external storage medium like a RAID NAS, etc. So it is a common use case to develop some images while travelling using a notebook and then later synchronize them to the original storage medium. But copying images manually from the main storage to the notebook and back is cumbersome and prone to errors.

The “local copies” feature of darktable has been designed to directly support those use cases. You can create local copies of selected pictures on your computer's local drive. This local copy is then used when the original image file is not accessible. At a later point, when connected again with your main storage medium, you can synchronize the XMP sidecar files, deleting the local copy of your input image. These operations can be found in the *selected images* panel (see Section 2.3.6, “Selected image(s)”).

For safety reasons, if local copies exist and the external storage is available, the local XMP sidecars are automatically synchronized at start up.

The local copies are stored into the `$HOME/.cache/darktable` directory and named “img-`<N>`-`<SIGNATURE>`.`<EXT>`”, where:

<i>N</i>	is the image number in darktable
<i>SIGNATURE</i>	is a hash signature (SHA-1) of the full pathname
<i>EXT</i>	is the original filename extension

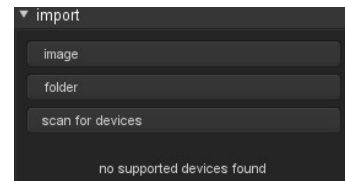
A local copy is identified in the lighttable view with a white marker on the right of the thumbnail. In addition all local copies carry the `darktable/local-copy` tag for selecting them easily.



2.3. Lighttable panels

2.3.1. Import

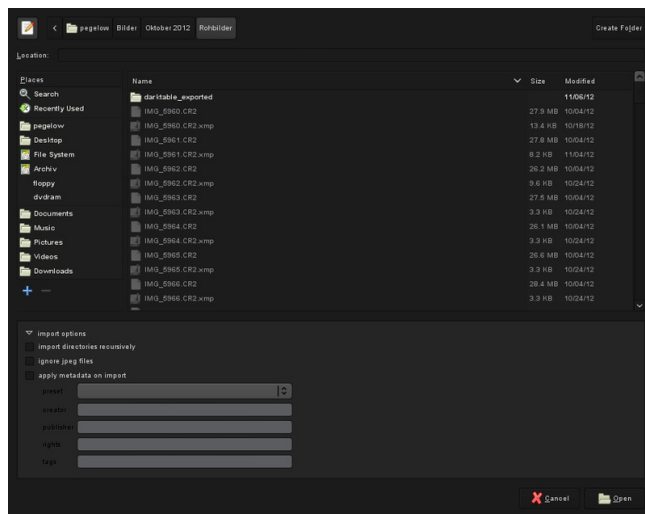
This panel is for importing images into film rolls. You can either import a complete folder, by pressing “folder”, or a single image with “image”. You can also cause darktable to search for connected cameras by pressing “scan for devices”. If a supported camera is connected you can import directly from that camera or control the camera from within darktable in tethering mode.



Imported images are organized as film rolls (see Section 2.2.1, “Film rolls”). All film rolls are accessible through the *collect images* module (see Section 2.3.2, “Collect images”). If you set the selection attribute to “film roll” you get a list of available film rolls, which can be filtered using the editbox to quickly find the one of interest. Double-click on a film roll in the list and it will open in the lighttable. You can also click the items in *recently used collections* (see Section 2.3.3, “Recently used collections”) to open the latest ones you have worked with.

2.3.1.1. Import from filesystem

You can import either a single image, or a folder. The imported image(s) will show up in a film roll with the name of the filesystem folder.



Clicking on “image” or “folder” opens a file selector dialog. Navigate through the filesystem, and select the item to import. On the lower part of the dialog, are some further import options.

As the name implies, checking “import directories recursively” will import images in the currently selected directory, and all subdirectories. It is not recommended, and wastes resources, to do this on an exhaustive list of images. darktable would generate thumbnails for all of them, but in the end only keep the recent ones in its cache. It is better to import images in smaller chunks, making logical film rolls.

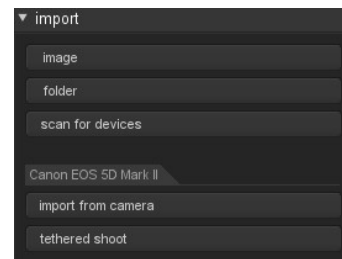
Checking “ignore jpeg files” is a good choice if there are JPEG images in the same folder that you do not want to process; eg. if the camera stores RAW+JPEG, you often only want to work on the RAWs, leaving the JPEG images alone.

You can also apply some metadata during import; see Section 2.3.10, “Metadata editor” for more details.

Importing a folder does not mean that darktable copies your images into another folder. It just means that the images are visible in lighttable and thus can be developed. If you delete an image or a folder from disk after having imported them, darktable cannot access them anymore. Importing an image or folder in darktable is not a backup of your filesystem! Moreover, darktable does not watch for changes in the filesystem. Thus, if you add an image to a folder, after having imported that folder in darktable, the new image will not be shown until explicitly imported.

2.3.1.2. Importing from a connected camera

When a camera is detected, it will show up in the device panel after pressing “scan for devices”. If you hover your mouse over the camera tab label, a tooltip will pop up with information about the camera, such as model, firmware version, and more. Depending on the camera's support, buttons with actions will be available such as “import from camera” and “tethered shoot”.



Import from camera

This will bring up an import dialog, showing the images on camera that can be selected for import into a film roll in darktable.

You define the base directory for storing imported images and the naming pattern of sub directories and individual images in the preferences dialog (see Section 7.3, “Session options”).

Tethered shoot

Tethering is used to integrate darktable with your camera. While you take images with your camera, they are automatically imported into darktable, so you can review the result of the shoot. You can also setup remote capture jobs, controlling the number of images and time between captures, along with camera settings such as exposure time, aperture and more.

If supported by your camera, tethering will take you into the capture view for tethered shooting. Read more about tethering in Chapter 4, *Tethering*.

2.3.1.3. Supported file formats

darktable is focused on managing and developing camera RAW files. It supports a huge number of file formats from various camera manufacturers. In addition darktable can read specific *low dynamic range* and *high dynamic range* images – mainly for data exchange between darktable and other software.

In order for darktable to consider a file for import, it must have one of the following extensions (case independent): 3FR, ARW, BAY, BMQ, CAP, CINE, CR2, CRW, CS1, DC2, DCR, DNG, ERF, FFF, EXR, IA, IIQ, JPEG, JPG, K25, KC2, KDC, MDC, MEF, MOS, MRW, NEF, NRW, ORF, PEF, PFM, PNG, PXN, QTK, RAF, RAW, RDC, RW1, RW2, SR2, SRF, SRW, STI, TIF, TIFF, X3F.

If darktable was compiled with JPEG2000 support, these extensions are also recognized: J2C, J2K, JP2, JPC.

If darktable was compiled with GraphicsMagick support, the following extensions are recognized in addition to the standard ones: BMP, DCM, GIF, JNG, JPC, JP2, MIFF, MNG, PBM, PGM, PNM, PPM.

Camera RAW files

darktable reads RAW files using two open source libraries: RawSpeed (developed by Klaus Post) and – failing that – with LibRaw. The number of supported cameras and file formats is constantly increasing. It is beyond the scope of this manual to give an exhaustive list. Most modern camera models are supported, and new ones tend to get added very quickly. With the exception of Fujifilm X-Trans cameras, darktable does not decode images from cameras with non-Bayer sensors (e.g. Sigmas with the Foveon X3 sensor).

LDR image files

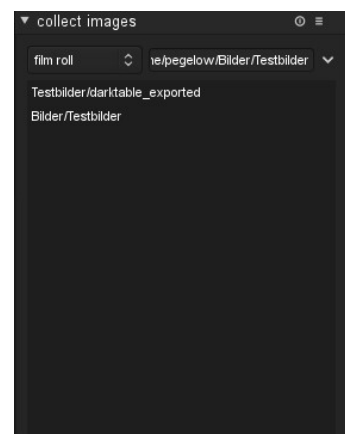
darktable natively reads “ordinary” images in JPEG, 8-bit/16-bit PNG and 8-bit/16-bit TIFF format. JPEG2000 is also supported if the required libraries are built into darktable at compile time. Similarly, if darktable was compiled with GraphicsMagick support, there are further import formats, like GIF, Dicom DCM, additional exotic TIFF formats, and some of Sun's “portable xyz-map” family.

HDR image files

darktable reads high dynamic range images in OpenEXR, RGBE and PFM format.

2.3.2. Collect images

The current view in lighttable is called a collection. The *collect images* panel lets you narrow down the list of visible images to just the ones you want to work with.



Information about all images imported into darktable are kept in a database, with various attributes describing each image. You define a collection by applying certain filtering rules to these attributes, creating a subset of images to display in the lighttable view.

The default collection is based on the film roll attribute – it displays all images of the last imported film roll or any other film roll chosen.

2.3.2.1. Usage

image attributes

The left combobox lets you choose from the available attributes:

film roll the film roll to which the image belongs

folders	the disk folder where the input image file is located
camera	the EXIF data entry describing the camera make and model
tag	any tag that is attached to the image; if activated a hierarchical list of known tags is displayed for quick selection
date	the date when the photo was taken, in the format <i>YYYY:MM:DD</i>
time	the time (date <i>and</i> time of day) when the photo was taken, in the format <i>YYYY:MM:DD hh:mm:ss</i>
history	choose images whose history stacks have been altered or not altered
color label	any color label that is attached to the image: "red", "yellow", "green", "blue", "purple"
title	the title, as listed in the image's metadata "title" field
description	the description, as listed in the image's metadata "description" field
creator	the creator, as listed in the image's metadata "creator" field
publisher	the publisher, as listed in the image's metadata "publisher" field
rights	the copyrights statement, as written in the image's metadata "rights" field
lens	the EXIF data entry describing the lens
ISO	the ISO value, as derived from EXIF data
aperture	the aperture value, as derived from EXIF data
filename	the filename of the physical input image

search rules

In the text field to the right of the attribute, you write a pattern. The pattern is compared against all database entries with the selected attribute. This search detects a match if the images' attribute contains the pattern in its full text. You may use "%" as wildcard character. The collection gets limited to those images where the query matches. Leaving the text field empty matches all images for that attribute.

The box below the search rule will list all matching database entries of the query you are currently working on. This list gets updated continuously as you type. You may also choose sorting criteria by scrolling through the list and double-clicking.

Clicking on the triangle button to the right of the text field opens a drop-down menu with options to finetune your current collection by adding further rules, or removing them:

clear this rule

Removes this rule – or resets to default if this is the only rule defined.

narrow down search

Adds a new rule which is combined with the previous rule(s) in a logical *AND* operation. An image is only part of the collection if it additionally fulfills the added rule.




add more images

Adds a new rule which is combined with the previous rule(s) in a logical *OR* operation. Images that fulfill the new rule are added to the collection.

exclude images

Adds a new rule which is combined with the previous rule(s) in a logical *EXCEPT* operation. Images which are selected by the new rule are exempted from the collection.

combining rules

The logical operators defining the combination of rules are displayed to the right of the rule: *AND* by the , *OR* by the , and *EXCEPT* by the . Clicking on either symbol gives a choice to change the logical operation.

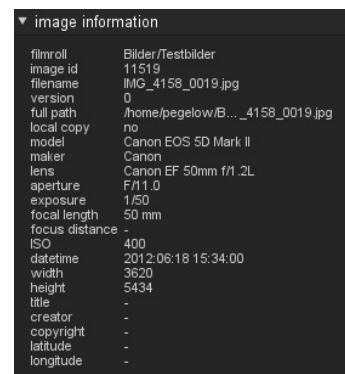
2.3.3. Recently used collections

This panel keeps track of the latest collections you have used, so you can jump to recently used collections without remembering what rules were specified in the collection.



2.3.4. Image information

This panel shows information embedded within an image's EXIF data. When hovering with the mouse over thumbnails, darktable will update this view, displaying information of the image currently under the mouse cursor. This panel is also available in darkroom, tethering and map view.



2.3.5. Select

This panel allows for a quick selection of images, according to some common criteria.



2.3.5.1. Usage

select all

Select all images in the current view (collection), with respect to the filters.

select none

De-select all images.

invert selection

Select all images that are not currently selected.

select film roll

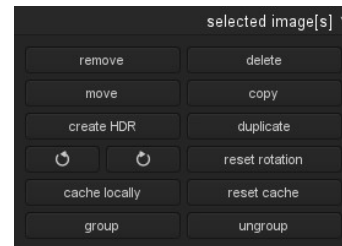
Select all images that are in the same film roll as the currently selected images.

select untouched

Select all images that have not yet been developed.

2.3.6. Selected image(s)

This panel provides some actions that operate on selected images.



2.3.6.1. Usage

remove

Remove the selected images from the darktable database. Those images will not be shown in lighttable anymore, but remain on the filesystem. As darktable stores XMP files with your development parameters on disk, you can later fully reconstruct your work by just re-importing the images.

When backing up your RAWs make sure to also save the XMP files!

delete

Physically delete selected images from filesystem. See also preference option "ask before erasing images from disk" (Section 7.1, "GUI options"). If this configuration option is not active, darktable will delete the file(s) without further question! This is irreversible, and will also erase your development work of these images.

When deleting an image with duplicates, darktable keeps the original input file on disk until the last of the duplicates gets deleted.

move

Physically move selected images (parent file plus all accompanying XMP sidecar files) to another filesystem folder. darktable does not overwrite images in the target folder. If an input image with the given filename already exists in the target folder the source image is not moved but kept where it is.

copy

Physically copy selected images (parent file plus accompanying XMP sidecar file) to another filesystem folder. If an image with the given name already exists in the target folder it does not get overwritten – instead a new duplicate with the given history stack is generated.

create hdr

Create a high dynamic range image from the selected images, and store it as a new source file in DNG format. Images need to be properly aligned, which implies that they have been taken on a sturdy tripod. You can also generate HDRs with programs like *Luminance HDR* [<http://qtpfsgui.sourceforge.net/>], and later import them into darktable for further processing (see Section 2.3.1.3, “Supported file formats”). Note that darktable cannot currently create HDR files from images made with non-Bayer (e.g. Fujifilm X-Trans) sensors.

duplicate

Create a virtual copy of selected images within darktable. It allows testing different developments for the same image, for example. Duplicate images share the same parent input file, but each have their own XMP sidecar file.

rotation

Perform a counter-clockwise or clockwise rotation on selected images. The third button resets the image rotation to the value in the EXIF data. This feature is directly linked to the orientation module (see Section 3.4.1.2, “Orientation”) – adjustments are converted into history stack items of that module.

copy locally

This action will create local copies of the selected images into the local drive. These copies will then be used when the original images are not accessible (see Section 2.2.9, “Local copies”).

resync local copy

This action will synchronize the XMP sidecars of the temporary local copy and the copy in external storage, if needed, and will remove the local copies. Note that if a local copy has been modified and the external storage is not accessible the local copy won't be deleted (see Section 2.2.9, “Local copies”).

group

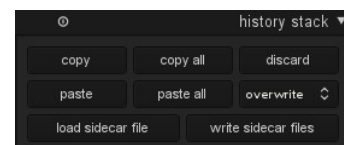
Create a new group from selected images (see Section 2.2.6, “Image grouping”).

ungroup

Remove selected images from the group (see Section 2.2.6, “Image grouping”).

2.3.7. History stack

This panel allows manipulating the history stack (development) of images. For each image, development is written in a sidecar file (.xmp), and is fully non-destructive.



2.3.7.1. Usage

copy

Copy the history stack of the selected image. You will be prompted for which items are to be include. If more than one image is selected, the history stack is taken from the image that has been selected first.

copy all

Copy the complete history stack of the first selected image; all items will be included. If more than one image is selected, the history stack is taken from the image that has been selected first.

discard

Physically delete the history stack of the selected images. Beware, this action can not be undone!

overwrite/append

This setting defines how a new history stack behaves when pasted on an image that already has a history stack. "Overwrite" will delete the previous history stack, whereas "append" will concatenate the two history stacks.

Caution: only in "append" mode you keep the option to later reconstruct your existing history stack whereas in "overwrite" mode your previous edit gets irrevocably lost. Beware, the setting of this parameter remains effective when you quit darktable.

paste

Paste a previously copied history stack onto all selected images. You will be prompted for which items to include. This button is greyed out, until a history stack is copied from another image.

paste all

Paste all previously copied items of a history history stack onto all selected images. This button is greyed out, until a history stack is copied from another image.

load sidecar file

Opens a dialog box to select an XMP file, thus loading a history stack that you can paste on images.

Files that were exported by darktable typically contain the full history stack if the file format supports embedded metadata (see Section 2.3.12, "Export selected" about this feature and its limitations). You can load an exported image as a sidecar file in the same way as you do with an XMP file. This feature allows you to recover all parameter settings in case you have accidentally lost or overwritten the XMP file. All you need is the source image, typically a RAW, and the exported file.

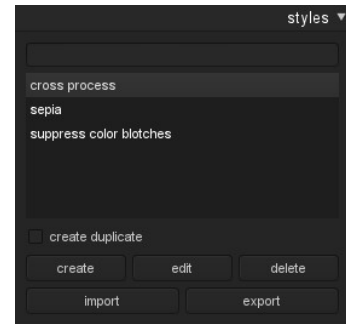
write sidecar files

Write XMP sidecar files for all selected images. The filename is generated by appending ".xmp" to the name of the underlying input file.

By default darktable generates and updates sidecar files automatically whenever you work on an image and change the history stack. You can disable automatic sidecar file generation in the preferences dialog (see Section 7.2, "Core options"). However, this is not recommended.

2.3.8. Styles

This panel provides a powerful functionality in darktable: storing a history stack as a style, and applying it to other images. Styles are created within this panel or in the darkroom (see Section 3.3.3, “History stack”). They are managed within this lighttable panel, which allows you to create, apply, edit and delete styles.



2.3.8.1. Usage

This panel displays a list of all available styles. A search field above the list allows you to input a text string which is compared against the styles' names and descriptions, thus limiting the list to the matching ones.

Double clicking on a style name applies the style to all selected images.

create duplicate

When applying a style to selected images, activating this box creates a duplicate of the image before applying the style. Disable this option if you want to try various styles without creating multiple duplicates; beware that in this case any existing history stack gets overwritten and cannot be recovered.

create

This creates new styles out of the history stacks of the selected images. For each image a style creation window pops up. You need to supply a unique name for the new style and you can add an additional descriptive text. You have the option to de-select those history stack items which you want to not be part of the newly created style.

edit

Styles are collections of history stack items. After pressing “edit”, you are prompted with a dialogue to include or exclude specific items from the stack. Check option “duplicate” if you want to create a new style, instead of overwriting the existing one; you need to provide a new unique style name in this case.

delete

This deletes the selected style, without further question.

import

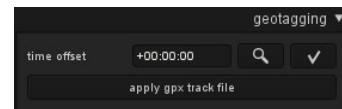
You can import a style which has been previously saved by darktable. darktable stores styles as XML files with the extension “.dtstyle”.

export

This option saves a selected style to disk as a “.dtstyle” file. This allows publishing styles and sharing styles with other users.

2.3.9. Geotagging

Use this panel to import and apply GPX track data on a selection of images. Alternatively, you can manually geotag images within the *Map* view (see Chapter 5, *Map*).



2.3.9.1. Usage

A GPS receiver calculates its current position based on the information it receives from satellites and records them in a GPX file – together with the current date and time. The EXIF data of the images also contains a time stamp defined by the camera settings. darktable takes the time stamp of the image, looks-up the position in the GPX file at that time, and stores the coordinates (latitude/longitude) in its database and the image's XMP sidecar file.

There may occur two problems. In contrast to GPS devices, most cameras don't record the accurate time. Secondly, the time stored in the EXIF data doesn't contain a time zone. Most people set their camera to local time, while the GPS devices store the time in UTC (Universal Time, Coordinated), i.e. Greenwich (London) time zone. If the time zones of camera and GPX file differs, than the related location will be wrong.

If your image already carry the UTC time stamp you can directly apply the GPX track without further adjustments.

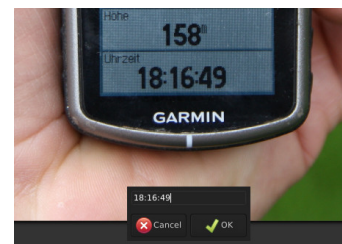
Otherwise we take two steps to correlate the time of camera and GPS tracker, first the offset, then the time zone.

To fix the drift of the camera time setting you can either enter it manually into the offset input field or let darktable calculate it. All you need is a picture taken of a reliable time source.



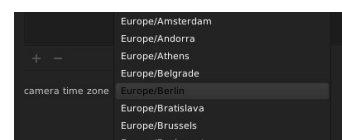
This can be any precise clock or even better the time displayed on your GPS device (normally it shows the local time, although it stores universal time).

When you have this image selected you can click on the (looking glass) button and darktable will present you an entry box. Just enter the time that is shown on the image. As a result you will get the difference between the time you entered and the one associated with the image in its EXIF data.



Now you can select all the images you want to geotag and click the apply button (currently represented by a check mark). This will alter the time in darktable's internal database for these pictures – so you will also see the change in the image information module on the left.

Now you can apply a GPX track. Click the corresponding button and navigate to the GPX file. Before confirming that dialog you should select the corresponding time zone for your camera in the drop-down-menu.



Should you ever make a mistake with the time zone selection you can just come back and reapply the GPX file with a different time zone.

2.3.10. Metadata editor

Edit metadata of an image, like *title*, *description*, *creator*, *publisher*, or *rights*. You can define your own presets, if you want to apply specific settings frequently.



2.3.10.1. Usage

clear

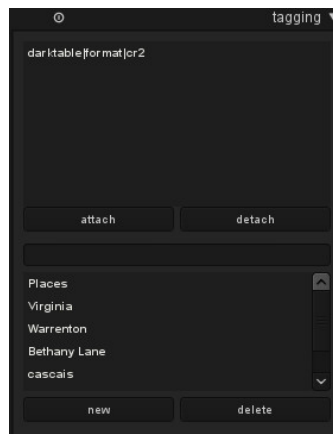
Delete existing metadata from the selected image(s).

apply

Apply new settings, as defined in the fields above, to the selected image(s).

2.3.11. Tagging

This panel is for managing tags on images. Tags are stored in both, sidecar files (.xmp), and within the darktable database for a faster access. The panel is divided into two parts: the upper part contains the tag(s) currently attached to the image under the cursor, or the selected image(s) if the mouse is outside the lighttable. The lower part contains a list of all available tags in the database, which can be filtered in the text box above.



New tags get added to the list either by typing into the text box and pressing the “new” button, or if an automatically generated tag gets attached to an image, or on import of already tagged images. darktable sorts the list of tags by relevance and frequency of usage.

Hierarchical tags are supported and can be create using the pipe symbol “|”.

As you may notice there are automatically created tags like “darktable|exported” or “darktable|styles|your style”, which keep track of your actions and let you easily find already exported images or images with specific styles applied.

Tip: ctrl-t opens a small text box at the bottom of the central view for quickly tagging your selected images by typing a tag phrase and hitting return.

2.3.11.1. Usage

attach

Attach the selected tag from the list below to all selected images.

detach

Detach the selected tag from all selected images. This also can be done by double clicking the specific tag.

new

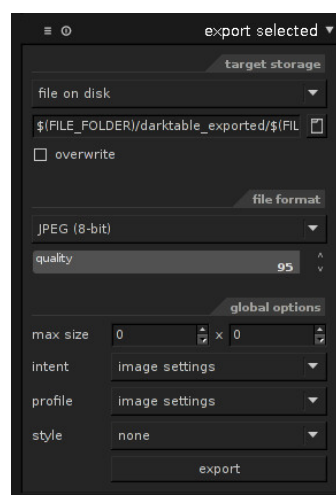
Create a new tag for the list.


delete

Delete a tag from the list and from all images. A warning will be displayed on how many images have this tag attached. Take this warning serious as there is no way to recover or later find the affected images.

2.3.12. Export selected

Each workflow ends in this panel: the export of your developed images. You can export either to a file on disk, or to various on-line storage places. Tip: you can use *ctrl-e* from within darkroom mode to export.



All settings of this panel can be saved for later reuse. Press the  button to manage your presets.

2.3.12.1. Usage

target storage

Where to store your selected images. Different back-ends are implemented, including file on disk, a LaTeX book template and various web albums. Depending on the selected target, you will be asked to give additional information, like filenames, or account name and password.


filename template

You can define filenames that darktable generates for export. Several pre-defined variables can be used as placeholders:

<code>\$(ROLL_NAME)</code>	roll of the input image
<code>\$(FILE_FOLDER)</code>	folder containing the input image
<code>\$(FILE_NAME)</code>	basename of the input image

\$(FILE_EXTENSION)	extension of the input image
\$(VERSION)	the duplicate version (see Section 2.2.7, "Sidecar files")
\$(SEQUENCE)	a sequence number within export job
\$(YEAR)	year at date of export
\$(MONTH)	month at date of export
\$(DAY)	day at date of export
\$(HOUR)	hour at time of export
\$(MINUTE)	minute at time of export
\$(SECOND)	second at time of export
\$(EXIF_YEAR)	exif year
\$(EXIF_MONTH)	exif month
\$(EXIF_DAY)	exif day
\$(EXIF_HOUR)	exif hour
\$(EXIF_MINUTE)	exif minute
\$(EXIF_SECOND)	exif second
\$(STARS)	star rating
\$(LABELS)	colorlabels
\$(PICTURES_FOLDER)	pictures folder
\$(HOME)	home folder
\$(DESKTOP)	desktop folder

output directory

Pressing button  opens a dialog to select the parent directory for export.

overwrite

Enabling the checkbox allows darktable to overwrite existing images during export. If this option is not checked darktable automatically selects a unique new file name in case of naming conflicts. The overwrite option gets reset after each export in order to protect you from accidental data loss.

file format

darktable can export to various file formats. Depending on the output format you can define additional parameters. For some formats you need to decide on the desired bit depth and the compression method, respectively.

For some export formats like JPEG you can define an output quality. Higher values will lead to larger file sizes. The default quality "95" is a good setting for very high quality exports, e.g. for archiving or printing purposes. If you need a good compromise between size and quality, e.g. for online image display or uploads, you should consider a value of "90" instead.

If the file format supports embedded metadata, like JPEG, JPEG2000 and TIFF, darktable will try to store the history stack as XMP tags within the output file. This information can later be used to reconstruct your parameters and settings that have produced the exported image (see Section 2.3.7, "History stack").

Caution: for various reasons embedding XMP tags into output files may fail without notice, eg. if certain size limits are exceeded. Users are therefore advised to not rely their backup strategy on this feature. To backup your data make sure to save your input (RAW) file as well as all of darktable's XMP sidecar files.

If you don't want to distribute history stack data with your images, there are various tools to delete embedded XMP tags. As an example you can use the program *exiftool* [<http://www.sno.phy.queensu.ca/~phil/exiftool/>] with:

```
exiftool -XMP:all= image.jpg
```

max size

Set the maximum width and height of the output images in pixels. Set both to a value of "0" to export with full resolution. darktable currently can only do down-scaling; the maximum output resolution is defined by the parent image.

Caution: it's a frequent pitfall to accidentally put low values, like 1 or 10, in these fields, causing darktable to produce miniature output files. You might think darktable's output is broken, but in fact it only generated what you asked for.

intent

This option lets you define the intent, i.e. the way darktable will deal with out-of-gamut colors. See Section 3.4.3.3, "Output color profile" for a more detailed description of the available options.

profile

This defines the output color profile. Select "image settings" if you want the settings in the *output color profile* (see Section 3.4.3.3, "Output color profile") module of the individual images to take precedence.

style

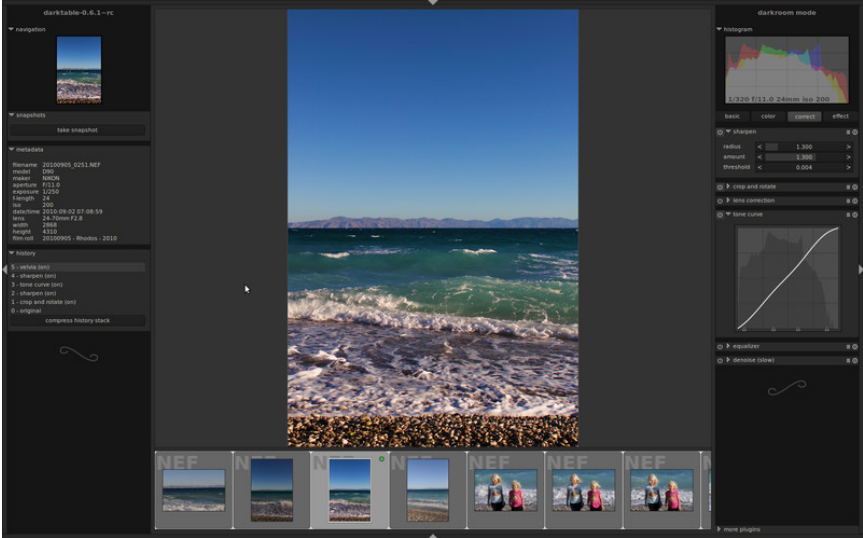
This option lets you choose a style, i.e. a collection of history stack items, which darktable concatenates with the existing history stack to generate the output image. These history items are only added temporarily; the original history stack is not overwritten. You can use this feature to add processing steps and parameters that you want to be applied specifically to exported images, e.g. you may define a style that adds a stronger level of sharpening when you produce scaled-down JPEG files for the internet. Learn more about styles in Section 2.3.8, "Styles", and Section 3.3.3, "History stack".

export

Pressing this button starts a background job to export all selected images. A bar at the bottom of the left side panel displays the progress. Whenever a file is successfully exported, a notification message pops up for a few seconds. You may click on the pop-up to make it disappear.

Chapter 3. Darkroom

The darkroom view is where you develop your image.



3.1. Overview

Darkroom mode is for photographic development of the image that you selected from the lighttable. Numerous tools, named modules, are available for processing that image.

On the left hand side you have navigation, snapshot, history stack, color picker, image information and mask manager panels, described in Section 3.3, "Darkroom panels". In the right hand panel you can see the histogram and a list of modules available for working with the image. At the bottom of the right hand panel you can enable/disable the view of individual modules.

You can use *middle-click* to zoom 1:1. A double *middle-click* takes you to 2:1. Alternatively you can zoom in and out between 1:1 and fit-to-screen by mouse scrolling. Mouse scrolling while holding the control key pressed gives an extended zoom range between 2:1 and 1:10.

You normally export multiple images from the lighttable view but you can also export the current image directly from the darkroom by using the shortcut *ctrl-e*. Export parameters are then those currently selected in the lighttable.

3.2. Darkroom concepts

This section tries to explain some of the basic concepts on how darktable develops images in the darkroom.

The basic element of an image operation in darktable is called a module. darktable comes with a rich set of over 50 modules for all kind of image manipulations. In Section 3.4, “Modules” you will find a description for each of the available modules.

3.2.1. Pixelpipe, module order, and history stack

darktable processes images – from input to output – in a so called “pixelpipe”. Within the pixelpipe image processing consists of consecutive operations which are implemented as “modules”.

Modules are applied in a fixed order. This differentiates darktable, as a non-destructive image editor, from classical image manipulation programs like The Gimp. As module order is fixed, you are free to activate, deactivate or change the parameters of a module at arbitrary points in time; the order of activation in your workflow does not have any impact on the outcome.

Users frequently ask why the module order is fixed and if there are plans to change that restriction. There are several reasons why darktable works in the way described:

- The sequence of modules has been selected with great care in order to give highest output quality. Changes to the sequence would generally worsen the result rather than improving it.
- Certain image processing steps just don't make sense if they are shifted in the pixelpipe. To mention just a few: highlight reconstruction needs to be done on raw data before demosaicing and the demosaic step needs to be performed before any input color profile can be applied.
- Most of darktable's modules are designed to work within a specific color model (see Section 3.2.10, “Color management” for more details). Full flexibility would require modules to support different parallel algorithms depending on the color space they are working in – this would drastically increase complexity.



That said, the fixed sequence of modules is not likely to change in the near future.

Whenever you activate or deactivate a module or go back to a module and change the parameters, this adds an item on top of the “history stack”.


For example, when working on a raw file, the history stack on the left panel might say that you first enabled *bilateral filtering*, then disabled *base curve*, then adjusted *white balance*. But at any time, the processing took the RAW image, adjusted *white balance* on it, then *demosaic*, then *base curve* (if enabled), then *bilateral filtering* (if enabled), as shown bottom to top on the right panel.


The history stack records your workflow in the order in which you made changes to the pipeline. It allows you to go back to an earlier stage of development if needed. The history stack represents your personal workflow and is not to be confused with the sequence in which modules are applied in the pixelpipe (see above). For more details on the history stack see Section 3.3.3, “History stack”.


3.2.2. Interacting with modules


A module has an expander bar  sharpen . Clicking on the name of the module expands the module's GUI with all parameters.

In its default setting darktable will only expand one GUI at a time. If you click the expander bar of another module, the previous GUI gets collapsed. If you want to see more than one GUI expanded, you may expand further modules with *shift-click* – all previously expanded GUIs remain opened. The expander bar behavior on *click* and *shift-click*, respectively, is controlled by a preference setting in gui options (see Section 7.1, “GUI options”).

Expanding a module does *not* activate it. You need to click the  icon to turn a module on or off.

Icon  accesses the module's available presets or creates a new preset from your current settings (see Section 3.2.3, “Module presets”).

The  icon is used to reset the module parameters to their default values.

Many of darktable's modules can have multiple instances, each with different settings. Click on the  icon to generate new instances and control existing ones (see Section 3.2.4, “Multiple instances”).

The most frequently used control elements of modules are sliders, comboboxes and curves.

3.2.2.1. Sliders

Sliders offer four different ways of interaction, depending on the level of control you need.

1. Triangular marker

Left-click the slider's triangular marker and drag it to the left or right.

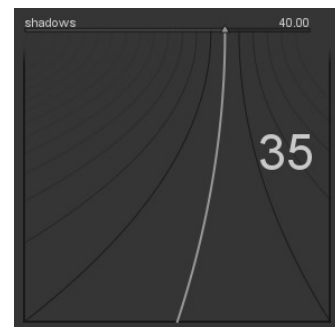
2. Mouse wheel

Hover over any place on the slider with your mouse, then use your mouse wheel to adjust the value step by step.

3. Right-click

When your mouse is over a slider right-click gives you a multi-functional pop-up below the slider for fine control with your mouse or numerical entry using the keyboard.

darktable's innovative input method: for both coarse and fine value adjustments in a single control element combined with keyboard input.



A bent line extending from the triangular marker moves as you move your mouse. The closer your mouse pointer is to the triangular marker the coarser the control; the

further away from the triangular marker the finer is your control. Left-click with your mouse to accept the new value and go back to normal control.

Alternatively you can type in a new value using your keyboard and commit by hitting the enter key. You may even supply the new value in the form of an arithmetic expressions which darktable will calculate for you – the old value is referenced as “x”.

4. Double-click

You can double-click on a parameter label to reset its value to default.


3.2.2.2. Comboboxes

Clicking on a combobox will open a list of available options. Click on the item you want to select. Sometimes the selection list opens close to the bottom or top of the screen and only part of the items are visible; scroll with your mouse wheel to bring up the full list.

3.2.2.3. Curves


Some modules are controlled by adjusting curves. More detail is given later in this chapter when the respective modules are explained.

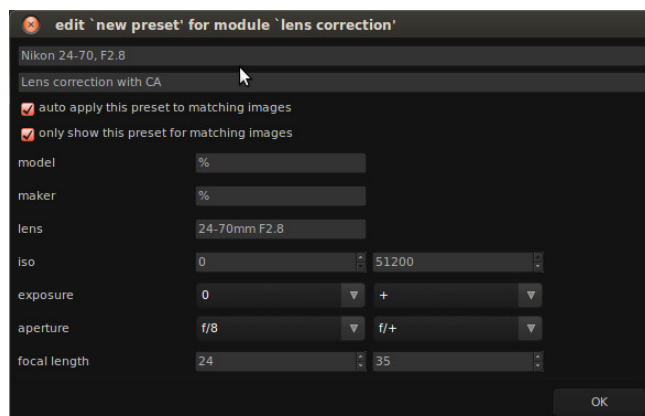
3.2.3. Module presets

Presets are stored configurations for a module's parameters. Some modules already have internal pre-defined presets but you can also define your own. Both internal and user-defined presets are displayed by clicking the  icon with the currently activated preset shown in bold text.

The preset system also supports automatic preset selection based on image data such as focal length, ISO, camera model and other fields.

3.2.3.1. Creating a new preset

First configure the module's parameters then click the  icon and select “store new preset”. The following dialog will be shown for configuring the preset:



The first two fields are used to name and describe the preset.

In the example above we have also checked the auto apply option. This brings up additional selection fields where you can define a filter used to decide if the preset should be

automatically applied when opening other similar images in darkroom for the first time. The example dialog above sets up following rules: if lens name matches and aperture is greater or equal to f/8 and focal length is between 24 and 35mm the preset will be automatically applied. Also the second checkbox is clicked so this preset will only be shown in the preset list if the image matches the rule.

darktable finds this data in your image's EXIF information. If you want a preset to be applied to all images from a specific camera leave all fields at default values except for the model field.

Tip: The *image information* panel for your image displays your model name, use this to ensure you have the correct spelling (see Section 2.3.4, "Image information").

3.2.3.2. Managing Presets

Both user created and pre-defined presets can be viewed and managed from within the presets menu (Section 7.5, "Presets") in the preferences dialog (see Chapter 7, *Preferences and settings*).

3.2.4. Multiple instances

Many of darktable's modules can be applied multiple times. Each instance of that module behaves like any other module, taking its input from the module below in the pixelpipe delivering its output to the module above.

3.2.4.1. Typical use cases

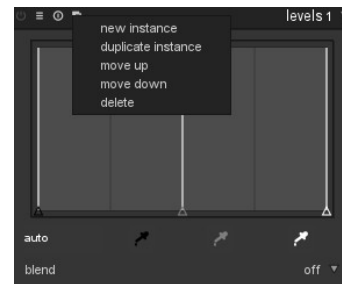
There are many occasions where it makes sense to have an operation act more than once in the pixelpipe. Here are a few use cases.

- Most of our modules are highly versatile and depending on parameters can deliver quite varying effects. For example the *fill light* module (Section 3.4.2.1, "Fill light") allows local modification of lightness based on pixel values. You might want to do two lightness corrections in your image at the same time – one for dark tones and another one for lighter tones.
- You might want to apply a denoising module like *denoise (profile)* (Section 3.4.4.3, "Denoise – profiled") with two different parameter sets. One to do luma denoising and another set of parameters to do chroma denoising. You could do so by generating two instances and use the first one only on luma by selecting blend mode "lightness" and use the second one just for chroma by selecting blend mode "color" (see Section 3.2.6, "Blending operators").
- In an even more elaborate case you could have a module act on different parts of your image. As an example you might want to apply a certain gradation curve with module *tone curve* (Section 3.4.2.3, "Tone curve") to your complete image and have a second curve being applied specifically to skin tones. All of the controls offered by *drawn masks* (Section 3.2.7, "Drawn mask") and *parametric masks* (Section 3.2.8, "Parametric mask") may be used to select those parts of an image where each of the module instances are applied.

Please be aware that of course each instance also adds to the workload of your pixelpipe. Generating too many instances – especially of the more demanding modules – will certainly cause some noticeable slow-down.

3.2.4.2. Managing instances

When clicking on the  icon a drop-down menu will appear.



Selecting “new instance” generates a new module instance above any existing ones. All parameters are set to default values. The new instance gets its own complete set of GUI controls and a number appended to the base module name for distinction.

Selecting “duplicate instance” behaves in a similar way. The only difference: the new instance will inherit all parameter settings from its parent.

darktable applies all modules in a defined order according to their type. Therefore all instances of a particular module will occur together in the pixelpipe. You can however decide on the relative order in which the different instances of a module are applied by selecting “move up” or “move down” to shift the position of the instance among its peers.

To delete an instance just press “delete” from the drop-down menu.

3.2.5. Blending

3.2.5.1. Overview

By default a module takes its input from the preceding module, performs its calculations and handles its output over to the next module in the pixelpipe. On demand you can activate an additional step where a module's output is reprocessed with its input before giving the result to the next module. This additional processing step is called blending. Input and output can be processed with different algorithms, called blending operators or blend modes.

Each blend mode is further controlled by a parameter called opacity, which can have a value between 0% and 100% and defines how input and output image contribute to the final result. Typically an opacity value of 0% gives as a result an image that is identical to the input image – the module remains without effect. An opacity value of 100% delivers the maximum effect of the module with the blend mode chosen.

The opacity value can be the same for all image pixels. In this case blending acts uniformly on the image. Alternatively you can make opacity values to vary between different image locations or pixel values. This is called a mask and gives fine control over what parts of an image are affected by a module and to what extent. At your choice you may activate a drawn mask or a parametric mask or a combination of both.

3.2.5.2. Usage

Modules with blending support exhibit an additional combobox “blend” at the bottom of their GUI.



blend

Blending is activated with this combobox. Depending on the value selected additional control elements will show up.

off

module's output is passed to the next module in pixelpipe without additional reprocessing. No further controls are displayed.

uniformly

reprocessing takes place with the chosen blend mode and opacity value – the same for all pixels. Additional controls to select blend mode and opacity value are displayed. The default blend mode is “normal” with an opacity of 100%.

drawn mask

reprocessing takes place with the chosen blend mode and opacity. Additional controls are displayed which allow you to draw a mask. If no mask elements are drawn all pixels have the same opacity, defined by the opacity slider. If you draw a mask element, e.g. a circle, the inner area of the circle will get maximum opacity, surrounded by a transition area or border with a gradual decay of opacity and the remaining image with an opacity of 0%. Different graphical shapes can be used. See Section 3.2.7, “Drawn mask” for further details.

parametric mask

reprocessing takes place with the chosen blend mode and opacity. Additional controls are displayed which allow you to adjust the opacity on a per-pixel basis determined by pixel values. In previous versions of darktable this was called “conditional blending”. See Section 3.2.8, “Parametric mask” for further details.

drawn and parametric mask

this option combines drawn and parametric masks and shows the full set of both controls. See Section 3.2.9, “Combining drawn and parametric masks” to learn how to best use this combination.

Mask combination and inversion

invert mask

When “drawn mask” is selected there is an additional combobox to invert the mask by switching mask inversion “on” or “off”.

combine masks

When either “parametric masks”, or “drawn and parametric mask” are selected an additional combobox is shown that controls how the individual masks are combined to form the final mask. Details on the combination of individual masks can be found in Section 3.2.9, “Combining drawn and parametric masks”.

Additional controls


When blending with a mask there are some additional options to deal with the final mask: you may blur the mask, temporarily disable it, or display it as an overlay image.

mask blur


Blurring the mask creates a softer transition between blended and unblended parts of an image and avoids artifacts. The mask blur slider controls the radius of a gaussian blur

applied to the final blend mask. The higher the radius, the stronger the blur – or set to 0 for an unblurred mask.

temporarily switch off mask

Sometimes it is useful to visualize the module's effect without the mask taking action. You can do so by clicking on the  symbol, which will temporarily deactivate the mask – the selected blend mode and opacity remain in effect. Switch this button on and off to see if the mask is acting on the image as intended.

display mask

Clicking on the  symbol will display the current mask as a yellow overlay over a black-and-white version of your image. Solid yellow indicates an opacity of 100%; a fully visible gray background image without yellow overlay indicates an opacity of 0%.

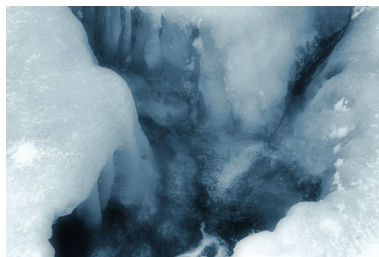
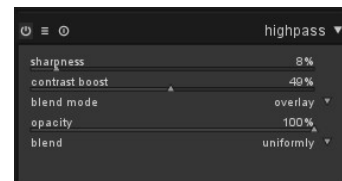
3.2.5.3. Examples

Texturing an image

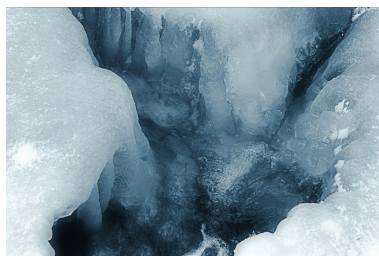
The watermark module supports SVG files with embedded images that can be used as a texture source. Blending operators then allow control of how that texture is expressed.

Gritty details

When blending operators were introduced into darktable, a new module named *highpass* (see Section 3.4.5.7, “Highpass”) was added. It provides a highpass filter of the image to be implicitly used with blending. It is used to produce a gritty detailed image and is a method widely used in the workflow of other imaging software.



This is the original image, pretty heavily processed: first *monochrome*, then some blue *splittoning* but as you see it lacks pop in details and is a bit out of focus...



Here we applied the highpass filter with the values shown above. You can now see that the details are greatly boosted and we now have a really gritty detailed image.

3.2.6. Blending operators

There are several blend modes implemented and more might be added in future. For now all of the most commonly used ones are included and you will recognize a few of them from other imaging software. A good introduction on many common blend modes is giv-

en in *The Gimp Manual (Chapter 8.2, "Layer Modes")* [<http://docs.gimp.org/2.8/en/gimp-concepts-layer-modes.html>]. Therefore we only discuss a few blend modes here in more detail.

3.2.6.1. blend modes

normal

This will probably be the most used blend mode. It just mixes input and output and, depending on the opacity value, it reduces the strength of a module's effect. Generally this is also the blend mode of choice if you want to apply a module's effect locally using masks.

normal bounded

This blend mode acts similarly to blend mode "normal", except that input and output data are clamped to a particular min/max value range. Out-of-range values are effectively blocked and do not pass to the following modules. Sometimes this helps to prevent artifacts. However, in most cases (e.g. highly color saturated extreme highlights) it is better to let unbound values travel through the pixelpipe in order to properly deal with them at the right place (e.g. in module *output color profile*). Blend mode "normal" is most often the preferred choice.

lightness

This blend mode mixes lightness from the input and output images. Color data (chroma and hue) are taken unaltered from the input image.

chroma

This blend mode mixes chroma (saturation) from the input and output images. Lightness and hue are taken unaltered from the input image.

hue

This blend mode mixes hue (color tint) from the input and output images. Lightness and chroma are taken unaltered from the input image. Caution: When modules drastically modify hue (e.g. when generating complementary colors) this blend mode can result in strong color noise.

color

This blend mode mixes color (chroma and hue) from the input and output images. Lightness is taken unaltered from the input image. Caution: When modules drastically modify hue (e.g. when generating complementary colors) this blend mode can result in strong color noise.

Lab lightness

Only available with modules that work in the Lab color space; this blend mode mixes lightness from the input and output images, while color data are taken unaltered from the input image. In contrast to blend mode "lightness" this blend mode does not involve any color space conversion and does not clamp any data. In some cases this is less prone to artifacts in comparison to "lightness".

Lab color

Only available with modules that work in the Lab color space; this blend mode mixes Lab color channels a and b from the input and output images, while lightness data are taken

unaltered from the input image. In contrast to blend mode “color” this blend mode does not involve any color space conversion and does not clamp any data. In some cases this is less prone to artifacts in comparison to “color”.

HSV lightness

Only available with modules that work in the RGB color space; this blend mode mixes lightness from the input and output images, while color data are taken only from the input image. In contrast to blend mode “lightness” this mode does not involve clamping.

HSV color

Only available with modules that work in the RGB color space; this blend mode mixes color from the input and output images, while lightness data are taken only from the input image. In contrast to blend mode “color” this mode does not involve clamping.

color adjustment

Some modules act predominantly on the tonal values of an image but also perform some color saturation adjustments, e.g. module *levels* and *tone curve*. The color adjustment blend mode takes the lightness only from output data and mixes colors from input and output enabling control of the module's color adjustments.

3.2.7. Drawn mask

Almost all darktable modules have the option to narrow down their effect by a drawn mask and thus allowing local adjustments.

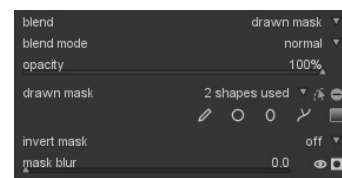
3.2.7.1. Overview


With the drawn mask feature you can construct a mask by drawing directly on the image base. Different drawing operators, called shapes, are available and can be used alone or in combination. A flexible editing feature allows you to change single aspects of a shape, remove shapes or import shapes already defined in other modules.

Internally shapes are stored as vector graphics and rendered with the needed resolution during pixelpipe processing. Shapes are expressed in the coordinate system of the original image and transformed with all distorting modules. This way a shape will always work on the same image area regardless of warping or other modifications that may be applied.

3.2.7.2. Usage

To draw a shape you need to click on one of the shape symbols. You will automatically be moved into the edit mode in which you generate a new instance of the selected shape and afterwards change its properties.




You leave edit mode by clicking on the  symbol. You can at any time go back to edit mode and do further adjustments by clicking the edit symbol again. In edit mode you can also remove a shape by right-clicking on it – the shape is removed from the current mask but it's still in the list of defined shapes.

If you *ctrl-click* on the edit mode symbol you enter a restricted edit mode. Certain actions like dragging a complete shape or changing its size are blocked. Only finetuning changes like dragging a node are allowed.

Currently five shapes are implemented.

brush

Clicking the  symbol adds a brush stroke.

Start drawing by *left-clicking* into the canvas and moving the mouse while keeping the button pressed. The brush stroke is finalized once you release the mouse button. Brush size, hardness and opacity can be changed by *scrolling*, *shift+scrolling*, and *ctrl+scrolling*, respectively, either before you start drawing or at any time on your way.

If you have a graphic tablet with pen pressure sensitivity, darktable can apply the recorded pen pressure to certain attributes of the brush stroke. See Section 7.1, “GUI options” for more details.

On lifting the tablet pen or releasing the left mouse button the drawn figure is converted into a number of connected nodes which define the shape. A configuration option (see Section 7.1, “GUI options”) controls how much smoothing is applied. A higher level of smoothing leads to less nodes – this eases subsequent editing at the expense of a lower initial accuracy.

Nodes and segments of a brush stroke can be modified individually. See the documentation on *path* below for more details.


Rendering a complex brush shape can consume a significant number of CPU cycles; consider to revert to the circle, ellipse or path shape if possible.

A brush stroke with controls and activated mask display.

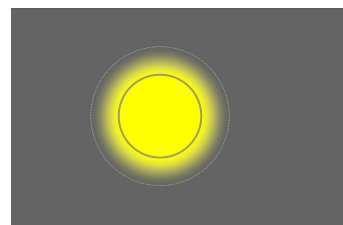


circle


Clicking the  symbol adds a circle shape.

Click into the canvas to place the circle. Left-click and drag the circle to a different position if needed. Use the scroll-wheel of your mouse while in the circle to change the diameter; scroll within the circle border to adjust the width of the gradual decay. With *ctrl+scroll* you can adjust the opacity of the circle – this is best observed with the mask displayed by pressing the  button.

A circle shape with controls and activated mask display.

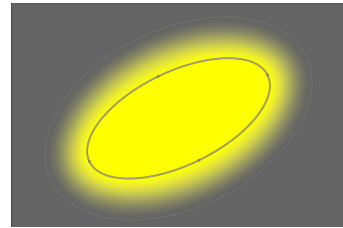


ellipse


Clicking the  symbol adds an ellipse shape.

The general principle is the same as for the circle shape. In addition you get four nodes on the ellipse line. Click on the nodes to adjust the ellipse's eccentricity. *ctrl+click* on them to rotate the ellipse.

An ellipse shape with controls and activated mask display.



path

Clicking the  symbol adds a shape defined by a user defined closed path.

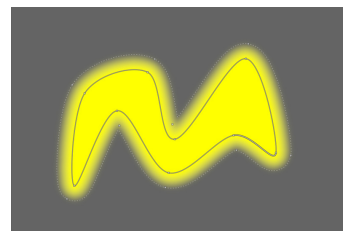
Left-click into the canvas to define path nodes; terminate the path by *right-clicking* after having set the last point. Per default nodes are connected by smooth lines. If you want a node to define a sharp corner, you can do so by creating it with *ctrl+click*.

In the edit mode you can convert existing nodes from smooth to sharp corners and vice versa by *ctrl-clicking* on them. You can insert additional nodes by *ctrl-clicking* on one of the line segments. Single nodes can be deleted by *right-clicking* on them; make sure that the mouse pointer is over the desired node and the node is highlighted, or else you might accidentally remove the whole path.


The size of the complete shape can be modified by scrolling – analogous to the circle shape. The same holds true for the width of the border, i.e. the area with a gradual opacity decay. Single nodes as well as path segments can be moved by mouse dragging. If a node is selected by clicking on it, a further control point appears – you can move it around to modify the curvature of the line. Dragging one of the control points on the border adjusts the border width just in that part of the shape.

Consider to finetune a path in the restricted edit mode (see above), which allows you to adjust single nodes and segments without the risk of accidentally shifting or resizing the whole shape.


A path shape with controls and activated mask display.



gradient

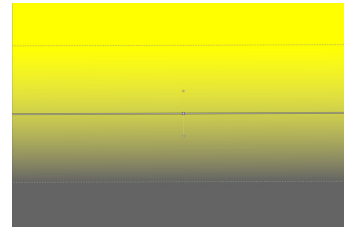
Clicking the  symbol adds a gradient to the mask. This does not generate a confined shape but produces a linear gradient extending the whole image.

Click into the canvas to define the position of the line where opacity is at 50%. The line has two anchor nodes which you can drag to change the rotation of the gradient.

Scrolling close to the center line changes the steepness of the gradient. Dotted lines indicate the distance beyond which the opacity is 100% and 0%, respectively. Between these dotted lines the opacity changes linearly. The gradient is best seen and modified when the mask is displayed by pressing the  button.

Depending on the module and the underlying image using a gradient shape might provoke banding artifacts. You should consider to activate the *dithering* module (see Section 3.4.4.11, “Dithering”)

A gradient with controls and activated mask display.



drawn mask

The number of shapes that are used in the current mask is displayed in the “drawn mask” field. Clicking on that field opens a dropdown box with all shapes that have already been defined in the context of the current image but are not yet used in the current mask. You can click on any of these items in order to add it to the current mask. The list also contains shapes once generated but no longer in use. This way you can even get back a deleted shape.

A polarity button (☕ and ☹, respectively) allows the user to toggle between the normal and the inverted state of the drawn mask, i.e. the opacity values get inverted – 100% becomes 0% and vice versa. You need this feature when combining drawn and parametric masks (see Section 3.2.9, “Combining drawn and parametric masks”).

mask manager

More functionality to control the interaction of multiple shapes within a mask can be found in the mask manager panel (see Section 3.3.5, “Mask manager”). Here you can give individual names to your shapes which will help you to keep track of your shapes. You can also select individual shapes for editing – a helpful feature if your masks happens to contain several shapes with overlapping control elements.

3.2.8. Parametric mask

The parametric mask feature, formerly called “conditional blending”, offers fine-grained selective control over how individual pixels are blended. It does so by automatically generating an intermediate blend mask from user defined parameters. These parameters are color coordinates not the geometrical coordinates used in drawn masks.

The parametric mask is a powerful tool with a certain level of complexity.

3.2.8.1. Overview

For each data channel of a module (Lab, RGB) and additionally for several virtual data channels (e.g. hue, saturation) users can construct a per-channel opacity function. Depending on the pixel's value for this data channel this function determines a blending factor between 0 and 1 (or 100%) for that pixel.

Each pixel of an image thus has different blending factors for each of its data channels (real and virtual). All blending factors are finally pixel-wise multiplied together with the value of the global opacity slider (see Section 3.2.6, “Blending operators”) to form a blend mask for the image.

If for a given pixel the blend mask has a value of 0, the input of the module is left unchanged. If for a pixel the blend mask has its maximum value of 1 (or 100%), the module has full effect.

3.2.8.2. Usage

When *parametric mask* is activated in combobox “blend” an additional set of tabbed controls is shown.



Channel tabs

Each tab selects a data channel – real or virtual. Modules acting in Lab color space have data channels for L, a, b, C (chroma of LCh) and h (hue of LCh). Modules acting in RGB color space have data channels for g (gray), R, G, B, H (hue of HSL), S (saturation of HSL), and L (lightness of HSL). Consult for example Wikipedia's article on color spaces [http://en.wikipedia.org/wiki/Color_space] for a deeper look.

Each tab provides two sliders for its data channels: one for the input data that the module receives and one for the output data that the module produces prior to blending.

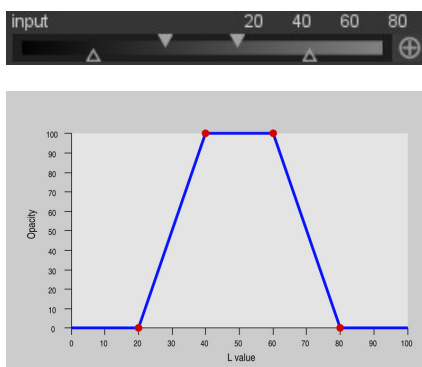
Color channel sliders

With the color channel slider you construct a trapezoidal opacity function. For this purpose there are four markers per slider. Two triangles above the slider mark the range of values where opacity is 1. Two triangles below the slider mark the range values where opacity is zero. Intermediate points between full and zero opacities are given a proportional opacity.

The filled triangles, or inside markers, indicate the closed (mostly narrower) edge of the trapezoidal function. The open triangles, or outside markers, indicate the open (mostly wider) edge of the trapezoidal function. The sequence of the markers always remains unchanged: they can touch but they can not switch position.

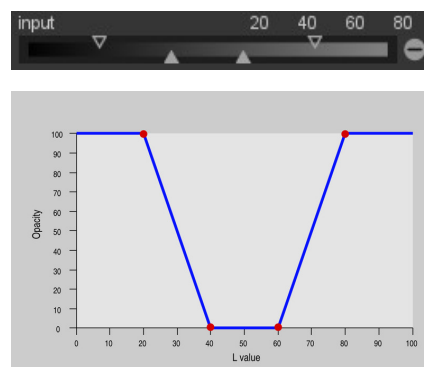
A polarity button (⊕ and ⊖, respectively) to the right of the slider switches between *range select* and *range de-select* function modes with visual confirmation provided by exchanging the upper and the lower triangle markers. These two types of trapezoidal functions are represented graphically in the following images.

Range select function



A trapezoidal function that selects a narrow range of values for blending.

Range de-select function



A trapezoidal function that de-selects a narrow range of values from blending.


In their default state all markers are at their extreme positions, maximum left and maximum right, respectively. In this state a *range select* function selects the whole range of values giving an “all at 100%” mask. Starting from there one can move the sliders inwards to gradually take out more and more parts of the image except of the remaining narrow range.


A *range de-select* function per default deselects the whole range of values, giving an “all-zero” mask as a starting point. Moving the sliders inwards gradually extends the mask more and more except of the remaining narrow range.


For more information on the polarity feature read Section 3.2.9, “Combining drawn and parametric masks”.

Control buttons

Control buttons help you when designing a parametric mask.

With the color picker button  you can select a probe from your image. The corresponding values for the real and virtual data channels are then displayed within each color channel slider.

With the invert button  you can toggle the polarities of all channels (including a potentially activated drawn mask) and change the method how channels are combined into the final mask. More on that topic can be found in Section 3.2.9, “Combining drawn and parametric masks”.

With the reset button  you can put all settings back to their default state.

3.2.8.3. Examples

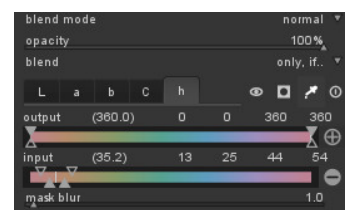
Colorkey effect

To create a colorkey effect with this poppy blossom in red and the remainder of the image in monochrome, we could apply module *monochrome* to all parts of the image except for of the saturated red petals.

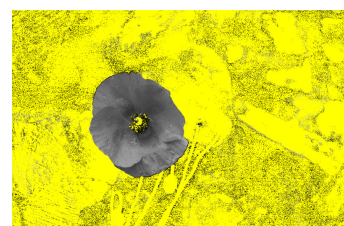


We choose the hue channel to control our mask as hue provides good separation between the petals and background.

These settings in hue channel construct a parametric blend mask that excludes the red petals. The small white bar in the gradient was obtained by using the color picker on one of the petals and the markers then closely centered on the indicated hue to increase the selectivity of our mask.



The resulting blend mask.



The final image after module *monochrome* is applied.



3.2.9. Combining drawn and parametric masks

This section describes how darktable combines individual masks to form the final mask of a module. Individual masks are the drawn mask and all the single channels of the parametric mask. The topic is rather advanced – if you don't want to go through all the theoretical details just jump down where we describe two typical use cases.



3.2.9.1. Overview

There are two main elements which control how individual masks are combined: the polarity setting of each individual mask, defined by the plus or minus buttons, and the setting in the “combine masks” combobox (see Section 3.2.5, “Blending”).

Masks can be regarded as grayscale images which take up values between 0 and 1.0 (or from 0% to 100%) for each pixel.

A straightforward way to combine masks is by multiplying the individual pixel values. The final mask will have a pixel value of 0 whenever one of the individual masks is 0 at that pixel location. The final mask can only reach a maximum pixel value of 1.0 if each and every of the individual masks has a value of 1.0 at that location. We call this way of combination “exclusive”. Any individual mask can exclude a pixel by setting its value to zero, regardless of what the other individual masks do. Once a pixel is excluded (its value is 0) by any mask there is no way to include it again by any other individual mask.

An alternative way to combine masks is the following: we first invert each individual mask – calculating 1.0 minus its value – then we multiply these inverted masks and as a last step invert the final mask again. Now if one of the non-inverted individual masks has a value of 1.0 at a pixel location the final will also be 1.0. The final mask can only reach a pixel value of 0 if all the individual masks have a value of 0. We call this way of combination “inclusive”. Any individual mask can include a pixel by setting its value to 1.0, regardless of what the other individual masks do. Once a pixel is included (its value is 1.0) by any mask there is no way to exclude it again by any other individual mask.

These two combination methods alone would still be rather limiting. We gain maximum flexibility by allowing an additional inversion step for each individual mask. This is governed by the polarity buttons  and  that you find close to the individual channels. Toggling the polarity button of a mask inverts its values, i.e. it recalculates the pixel values to 1.0 minus the original value.

Finally within the “combine masks” combobox you may once again invert the final result to fit your needs by selecting the *exclusive & inverted* or *inclusive & inverted* options.


3.2.9.2. Usage

You will typically want to combine drawn and parametric masks to first select a certain region of your image – either by the drawn or the parametric mask – and use the other mask type to finetune your selection. Finetuning can either mean that you want to include further parts of the image, which are not included in the first place, or you want to exclude parts of the image that were previously included.





This gives two typical use cases:

Inclusive mode

For this mode you set the “combine masks” combobox to *inclusive* and make sure that all polarity buttons of all the individual channels and of the drawn mask are set to negative (). Your starting point is a mask where all pixels have a value of zero, i.e. no pixel is selected. You now adjust the parametric mask sliders to bring more and more pixels into the selection or you draw shapes on the canvas to select specific areas of your image.

Exclusive mode

In the opposite case you set the “combine masks” combobox to *exclusive* and make sure that all polarity buttons are set to positive (). Your starting point is a mask with all values at 1.0, i.e. all pixels selected. You now gradually change the parametric mask sliders to exclude parts of your image as needed or you directly draw shapes on the canvas to specifically exclude these areas.

For your convenience you find in the parametric masks GUI a  toggle button that inverts all channel polarities and toggles between inclusive and exclusive mode in the “combine masks” combobox.

For novice users it is recommended to stick to these two use cases. This implies that you should decide beforehand how you want to construct your mask. Advanced users will find their way to take advantage of the many possible combinations of polarities and mask combination modes.

3.2.10. Color management

The darkroom employs a fully color managed workflow:

- Input color specifications are taken from embedded or user supplied ICC profiles or – in case of raw files – alternatively from a library of camera specific color matrices.
- darktable automatically reads the display profile of your monitor (if properly configured) for an accurate color rendition on screen. Multi-screen setups are fully supported as long as a system service like *colord* is in place and properly set up to inform darktable about the correct monitor profile.
- Output files can be encoded in one of darktable's built-in profiles, like sRGB [<http://en.wikipedia.org/wiki/SRGB>] or AdobeRGB (compatible) [<http://en.wikipedia.org/wiki/AdobeRGB>], or into any other output color space that the user supplies to darktable as an ICC profile.

To investigate your display profile configuration you can call the `darktable-cmstest` binary which prints out useful information like profile name per monitor and tells you if the system is correctly configured.

3.2.10.1. darktable's color spaces

darktable's input images are either RGB files (like JPEGs or TIFFs) or pre-demosaiced camera raws – both represent colors by a combination of red, green and blue. Most part of our modules act in the CIELAB color space [http://en.wikipedia.org/wiki/Lab_color_space] (often just called “Lab”) which describes color as a combination of lightness data (L) and a point in the a-b color plane. The final output of the image processing pipeline is once again in RGB, either shaped for the monitor display or the output file.

This process implies that the pixelpipe has two color conversion steps: *input color profile* and *output color profile*. In addition there is the *demosaic* step for raw images, where the colors of each pixel are reconstructed by interpolation.

Each module has a fixed position in the pixelpipe which tells you which color space the module lives in:

up to <i>demosaic</i>	Image is in raw data format with only latent colors. Each pixel carries lightness and color information for only one base color. Please mind that some of the modules in this part can also act on non-RAW input images in RGB format with full information on all three color channels.
between <i>demosaic</i> and <i>input color profile</i>	Image is in RGB format within the color space of the specific camera or input file.
between <i>input color profile</i> and <i>output color profile</i>	Image is in Lab format. This is a very huge universal color space which covers all colors visible to the human eye (and even more). As darktable processes images in 4x32-bit floating point buffers, we can handle the Lab color space without risking banding or tonal breaks.
after <i>output color profile</i>	Image is in RGB format as defined by the selected display or output ICC profile.

3.2.10.2. Unbounded colors

Theoretically the individual components of color data are confined to certain minimum and maximum levels. As an example the intensity of an individual red, green or blue color channel in RGB can be anything in the range between 0% and 100% (or between 0.0 and 1.0). Likewise the L channel in Lab can be anything between 0 (pure black) and 100 (pure white).

In practice sometimes the image processing steps in darktable's modules can lead to values which lie outside of these confined ranges. In fact even the well established color matrices, which convert from camera RGB to Lab, may sometimes produce pixels with negative L values.

Pixels with these kind of values are said to have “unbounded colors”. One could *clamp* (i.e. confine) those values to the allowed range at every processing step. However, it has been found that it is much preferred and less prone to artifacts to not clamp in-between unbounded colors, but treat them just like any other color data.

In darktable there is no technical requirement for clamping; due to the fact that we are using floating point arithmetics for all processing steps, unbounded values can be passed along like any other value until the final color conversion module makes sure that they are transferred into the most reasonable color within the selected output color space. Therefore darktable avoids clamping as far as the underlying algorithms allow.

3.2.10.3. Possible color artifacts

That said there are some infrequent situations which still can lead to problematic results unless the user takes some action. Some modules in Lab color space, like *levels* and *monochrome*, need to rely on the fact that the L channels carries all lightness information and the a and b channels purely represent chroma and hue. Unbounded colors with negative L values are especially problematic to these modules and can lead to black pixel artifacts.

It has been found that highly saturated blue light sources in the image frame are hot candidates for pixels with negative L values. If you are engaged in stage photography you should pay close attention to image areas like that.

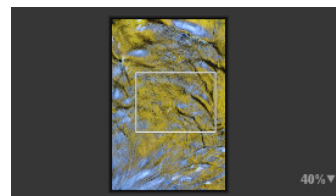
In order to mitigate that issue the *input color profile* module (see Section 3.4.3.10, "Input color profile") has a *gamut clipping* option. It is off by default but can be activated in case you observe artifacts. Depending on the settings, colors get confined to one of the offered RGB gamuts. In effect black pixel artifacts are prevented at the costs of losing some color dynamics. See Section 3.4.3.10, "Input color profile" for an example.

3.3. Darkroom panels

This section contains documentation for panels that are specific to the darkroom view.

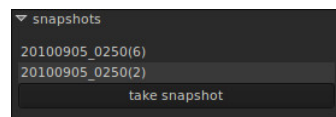
3.3.1. Navigation

On the top left hand side the navigation panel displays a full preview of your image with a rectangle showing the currently visible zoom area in the central panel. Drag the rectangle around to pan the zoomed-in view. The current zoom scale is displayed to the right of the preview image. Click on that figure for a quick access to some common zoom levels.



3.3.2. Snapshots

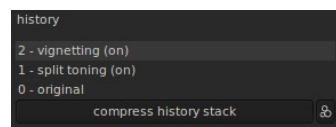
You can take snapshots of images as you process them. A snapshot is stored as a bitmap of the current center view and is kept as long as you stay in the darkroom. A snapshot can then be selected and overlaid in the current center view to help you with a side by side comparison (left: snapshot, right: active) when you are tuning parameters of a module. This can also be combined with history (see Section 3.3.3, "History stack") to compare the snapshot against different stages of development.



You can control the split view by moving the splitline back and forth. If you hover with the mouse over the splitline, a small rotation icon will appear on the center of the line. Click it to change between vertical and horizontal split view.


3.3.3. History stack

The history stack lists every change of state (activate/de-activated) for all modules. Here you can *select* a point in stack to return to that point of development history. If you then activate a new module or change a module parameter, all modules above the current point will be discarded.




Caution: activating any module action using key accelerators will discard all modules above the currently selected one. It is easy to lose all development work on an image this way!

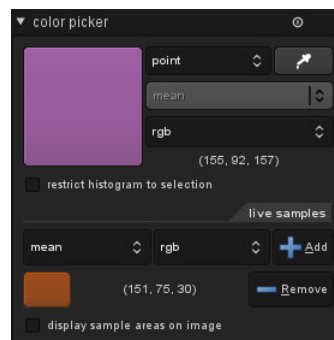
Hitting "compress history stack" generates the shortest history stack that produces the current image, i.e. suppressing all de-activated modules. This also will discard all modules above the currently selected one.

The button to the right  lets you create a new style for applying your history stack to other images. Use the first line of the popup dialog window to name your style and the second to add a searchable description. You are prompted for which of the current history stack modules to include in the style.

Once created styles are then managed and applied to other images in the lighttable's *styles* panel (see Section 2.3.8, "Styles").

3.3.4. Global color picker

Using the global color picker you can take color samples from your image, display their values in multiple ways and compare colors from different locations. The color picker is activated by pressing the  icon. There are multiple parameters for controlling how the color picker works, whose settings remain in effect until you leave the darkroom mode.



Besides the global color picker described here there are also local color pickers in some of the modules (eg. *tone curve*). Global and local color pickers are different. The global color picker works in monitor color space and takes samples after the complete pixelpipe has been processed. The local color pickers run in the color space of the individual module, which is usually Lab; they reflect the input and output data of that specific module within pixelpipe.

The global color picker can be run in point or area mode. When in point mode only a small spot under your cursor is taken as a sample. In area mode you can draw a rectangle and darktable samples the area within that rectangle. The combobox to switch between point and area mode can also be used to toggle the mode of local color pickers.

If samples are taken in area mode, darktable will calculate mean, min and max color channel values. A combobox allows you to select which of those are displayed. For obvious statistical reasons mean, min and max are identical for the single sample of point mode.

A color swatch representing the sampled point or area is displayed. Numerical values are shown as well. As said before global color picker works in monitor RGB color space. You can also let darktable translate these numerical values into Lab color space. Beware that Lab values are approximated here; depending on monitor color profile there can be some deviations from the real values.

When the checkbox "restrict histogram to selection" is ticked, only the values of your selected area or point are taken into account by the main histogram at the top of the right hand panel (see Section 3.3.6, "Histogram"). This is a way to show which tonal values are present in a specific area.

The sampled colors in either area or point mode can be "stored" as live samples by pressing the "add" button. darktable will then show a color swatch and numerical values for each stored sample. You can once again select which numerical value (mean, min, max) is to be displayed and if this is to be done in RGB or Lab color space.

Newly created live samples are not locked. If you change your image the changes will be reflected in your live samples. Use this if you want see how changing parameters effects different parts of an image. Clicking on a live sample's color swatch locks it and a lock symbol is displayed. Further image changes will then no longer affect the sample. You can for example take two live samples from the same location and lock just one of them to provide a before and after sample comparison.

Live sample locations are indicated in your image if you check option "display sample areas on image".

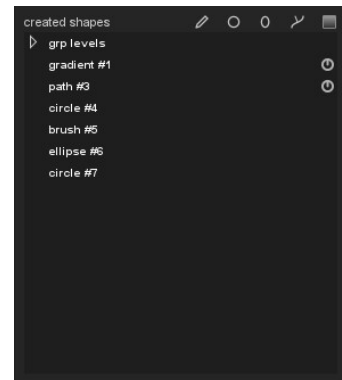
3.3.5. Mask manager


3.3.5.1. Overview

The masks manager panel is the central place where you manage all masks and shapes within the context of the current image. Here you create, delete and change shapes or give them unique names. You can add shapes to and remove shapes from a mask, and you define how multiple shapes interact within a mask.

3.3.5.2. Usage

In the top line of the masks manager panel you find buttons to create new shapes. They are the same as in the *drawn mask* GUI (see Section 3.2.7, “Drawn mask” for more details).



The lines below list all masks used and all individual shapes defined. The masks are noted with a headline in the form “grp levels” indicating the module in which they are used. The list of masks is followed by a list of all individual shapes which have been generated in the context of the given image. If a shape is in use by any of the masks it is marked by the  symbol to the right of the shape name.

3.3.5.3. Shapes

By default shapes receive an automatically generated name, consisting of the shape type (“brush”, “circle”, “ellipse”, “path”, “gradient”) and a number that is incremented automatically. You can replace these automatically generated names by more meaningful ones. Double-clicking on the existing shape name prompts you for a new one. Giving a meaningful name is a good habit, especially if you are going to use the same selection in different masks. A name like “house front” makes it easier to get the right shape, rather than something like “path #32”.

Clicking on the shape name shows the selected shape in the center canvas with all its controls. This is a convenient way to edit the properties of a specific shape. Especially if there are so many shapes within a mask that their controls overlap and make it difficult to hit the right target.

Right-clicking on a shape name gives you a drop-down menu with the options to remove the current shape or to remove all shapes currently not in use.

All shapes ever defined for the current image are kept in the list unless you remove them explicitly. If you have worked a lot with shapes on one image, this list can get quite long. All settings – with all defined shapes – are part of the XMP tags of an image and are included in exported files. If the list of shapes is very long the needed space to store all shapes might exceed the given limits of certain file formats, like JPEG. In that case storing the XMP tags might fail during export. This is normally not a problem – however, you can no longer

rely on the exported file to contain your full history stack (see Section 2.3.12, “Export selected”).

3.3.5.4. Masks

Clicking on the name of a mask expands a list showing the individual shapes which constitute that mask.

Right-clicking on the shape name opens a drop-down menu. Here you define the way that the individual shapes interact to form the mask. You can also remove shapes from that mask.



Masks are constructed by adding the shapes in the order that they are listed from top to bottom. Each shape adds to the mask by using at your choice one out of four logical set operators.

As order matters when combining shapes you may move each shape up or down if needed.


Each shape before being added can be inverted and is then depicted by the  symbol.

3.3.5.5. Set operators

We use as an example a combination of a gradient followed by a path to demonstrate the effect of the set operator which we apply to the path shape. As a convention we say that a pixel is “selected” in a mask or shape if it has a value higher than zero.




union

This is the default set operator. It is depicted by the  symbol left to the shape name. The shape adds to the existing mask in such a way that the resulting mask contains the pixels that are either selected in the existing mask or in the added shape. In overlapping areas the maximum value is taken.




intersection

This set operator is depicted by the  symbol left to the shape name. The shape adds to the existing mask in such a way that the resulting mask contains only pixels that are selected in both, the existing mask and the added shape. In overlapping areas the minimum value is used. In the given example we use this operator to “imprint” the path with a gradient.




difference

This set operator is depicted by the  symbol. In the non-overlapping area the existing mask is unchanged. In the resulting mask pixels get selected only if they are selected in the existing mask but *not* in the added shape. This set operator can be chosen if you want to “cut out” a region from within an existing selection.



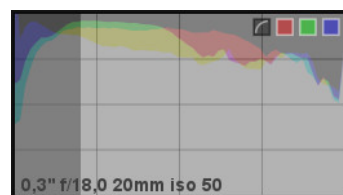
exclusion

This set operator is depicted by the  symbol. The resulting mask has all pixels selected that are either selected in the existing mask and not in the added shape or vice versa. This is equivalent to an “exclusive or”.



3.3.6. Histogram

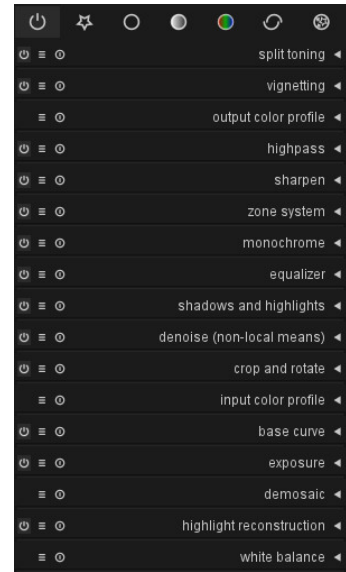
This shows a histogram of the developed image's light levels. In its default state curves for all three RGB color channels are displayed. You can toggle the colored squares to enable or disable specific color channels. A curve button is also provided to toggle between linear view, logarithmic view and wavefront view.










The histogram is directly linked to the *exposure* module described in Section 3.4.1.5, “Exposure”, and you can operate some of the exposure module's controls from the histogram. You can left-click towards the right hand side of the histogram and then drag right to increase or drag left to decrease the exposure. In a similar manner you can control the black level by clicking and dragging in the left hand side. Double-clicking in the histogram resets the exposure module's parameters to their defaults.

3.3.7. Module groups

The module groups button bar gives you quick access to darktable's processing modules.



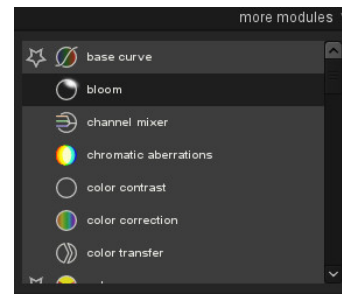
Here follows a description of the module groups available:

	Active	Modules you have activated and are using on the current image.
	Favorites	Modules you have marked as favorites using <i>more modules</i> (see Section 3.3.8, "More modules").
	Basic	Modules that are frequently used, such as exposure, temperature etc. (see Section 3.4.1, "Basic group").
	Tone	Modules for working with the image's tonal values, e.g. levels, tonemap etc. (see Section 3.4.2, "Tone group").
	Color	Modules for processing colors, such as color correction, vibrance etc. (see Section 3.4.3, "Color group").
	Correction	Modules making corrections to the image, e.g. denoise, CA correction etc. (see Section 3.4.4, "Correction group").
	Effect	Modules with a more artistic output, such as vignetting, softening etc. (see Section 3.4.5, "Effect group").

Clicking on one of the group symbols will show the modules in that group. If you once again click on the symbol, grouping will be de-activated and all non-hidden modules will be shown in one long list. This list shows the sequence in which modules are applied from bottom to top. As a general principle darktable applies modules in a pre-defined sequence.

3.3.8. More modules

More modules at the bottom of the right panel is used to show the less frequently used modules. By default only standard modules are shown to the user but you can use this function to make the extra modules visible, or alternatively to hide away modules you don't typically use.




Each module is shown with a small icon next to its name. Left-click with your mouse to toggle the status between visible, hidden and favorite. Favorite modules are indicated by a star in front of the icon and in addition to appearing in their normal module group will also be visible in the module group *favorites*. This is a good way to get fast access to modules that you use very frequently. Visible modules are indicated in the list by a light grey background whilst hidden modules have a dark grey background and do not display any of their controls.

Hiding or un-hiding modules is not meant to be part of your daily workflow, you should only occasionally need to review the modules you typically use.


3.3.9. Bottom panel

The bottom panel provides quick access to apply presets and styles to your image and allows to activate the over/underexposure warning. You can also activate a filmstrip for quick navigation within the current collection.


3.3.9.1. Quick access to favorite presets

Clicking the  icon opens a combobox that gives you quick access to your favorite module's presets. Click on the preset name to apply it to the image.

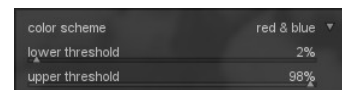
3.3.9.2. Quick access to styles

Clicking the  icon opens a combobox with your styles. Hovering with the mouse over a style name opens a tooltip showing the involved modules. Click on a style name to apply that style to the image.

3.3.9.3. Over/underexposed warning

By clicking the  icon an over/underexposed warning is toggled on or off. Pixels outside the dynamic range, close to pure white or close to pure black, are prominently displayed in a signal color. You can also activate the over/underexposure warning with the keyboard shortcut "o".

Right-clicking on the icon opens a dialog with configuration parameters.



color scheme

In the default color scheme underexposed pixels are shown in blue and overexposed pixels in red. These colors are easy to identify in most cases. In some cases you may want to change the color scheme to "black & white" or "purple & green", eg. if you experience overexposed highlights in red blossoms.

lower threshold

Sets the threshold for underexposure warning, expressed as a percentage of the maximal brightness.

upper threshold

Sets the threshold for overexposure warning, expressed as a percentage of the maximal brightness.

3.3.9.4. Filmstrip

The optional filmstrip can be used to quickly switch between images while remaining in the darkroom view. The images viewed are the same as the ones in the lighttable view.

The filmstrip can be switched on and off using the shortcut *ctrl-f*. You can scroll with your mouse to quickly navigate through the images and change the height of the filmstrip panel by dragging its top.



3.4. Modules

Modules are organized into five functional groups: basic, tone, color, correction and effect. You either view all modules in one long list or instead click on a group to just display modules belonging to that group.

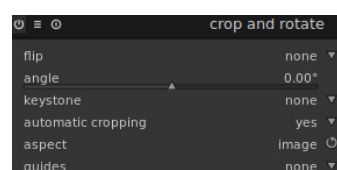
3.4.1. Basic group

The basic group of modules contains the modules for basic development. These are ones you probably will use most often, such as exposure, white balance etc.

3.4.1.1. Crop and rotate

Overview

This module is used to crop, rotate and correct perspective distortions of your image. You can overlay your image with various helpful guidelines that assist you using the tools.



Some of the tools of this module, namely adjustment of angle and corrections of perspective distortion, will require the original image data to be interpolated. For best sharpness results set “lanczos3” as pixel interpolator in *core options* (see Section 7.2, “Core options”).

Usage

Whenever the user interface of this module is in focus, you will see the full uncropped image overlaid with handles and guiding lines.

First off, select what aspect ratio you want and size the crop boundaries by dragging border and corner handles. Use the button right of the aspect box, to swap between portrait and landscape mode. You can move around the crop rectangle by holding down left mouse button and move around. When you are done and want to execute the crop, just give focus to another module or double-click into the image base. You can at any time change your crop area by just revisiting this module.

flip

This tool is used to flip the image on the horizontal, vertical or both axis.

angle

This tool corrects the rotation angle helping you level an image. You can either set a numerical value or use your mouse directly on the image. To use your mouse, right-click, hold it down and draw a line along the horizon; as soon as you release the mouse button the image is rotated so the line you drew matches the horizontal axis.

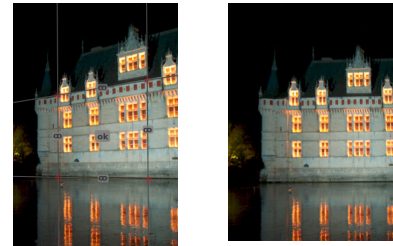
keystone

This tool is used to correct perspective distortions in your image. Useful for example when you shoot a high building from ground with a short focal length, aiming upwards with your camera. The combobox lets you select the type of correction you want to use :

vertical if you want to limit the correction to vertical lines

horizontal limit the correction to horizontal lines
free if you want to correct horizontal and vertical lines

Depending on the selected correction type you will see two or four straight adjustment lines overlaid to your image. Two red circles on every line let you modify the line positions with your mouse. Each line additionally carries a "symmetry" button. If activated (and highlighted in red) all movements of the affected line will be mirrored by the opposite line.



In order to correct perspective distortions, you need to find suitable horizontal and/or vertical features in your image and align the adjustment lines with them. When done, press the "OK" button, which is located close to the center of your image. The image will be corrected immediately. You can at any time come back and refine your corrections by selecting "correction applied" in combobox keystone.

automatic cropping

Use this options to avoid black edges on the image borders. Useful when you rotate the image.

aspect

Here you can change what aspect ratio you want to have on the result, thus constraining the ability to drag and crop rectangle out of the aspect ratio of your choice. Many common numerical ratios are pre-defined. You can also select any other ratio after opening the combobox and typing it in the form of "x:y". A few special aspect ratios deserve explanation:

free free forming the rectangle without any ratio restrictions
image this option constrains the ratio to be equal to image ratio
golden cut this option constrains the ratio to be equal the golden number
square this option constrains the ratio to be 1

guides

Many self-explaining guides are available to help you compose your image.

Examples

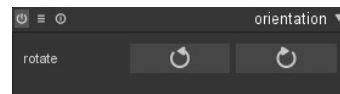


A cropped image in center view when module crop and rotate is in focus. The cropped area is visible as well as some guiding lines.

3.4.1.2. Orientation

Overview

This module allows the user to modify the orientation of an image. By default it is active for all images and receives its standard settings from the camera's orientation flag stored in the EXIF data.



Usage

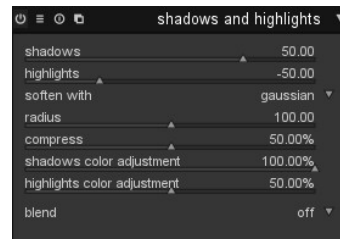
rotate

Clicking on one of the buttons causes a counter-clockwise or clockwise rotation in steps of 90 degrees.

3.4.1.3. Shadows and Highlights

Overview

The shadows and highlights module allows adjustment to the tonal range of darker parts of an image (shadows) and lighter parts (highlights); it can bring back details in shadows and highlights by enhancing local contrast.



Usage

shadows

This slider controls the effect on shadows; positive values will lighten up shadows while negative values will darken them.

highlights

This slider controls the effect on highlights; negative values will darken highlights while positive values will lighten them up.

soften with

This combobox chooses the underlying blurring filter, gaussian or bilateral. Try bilateral filter if you experience halos with gaussian blur.

radius

This slider controls the radius of the involved blurring filter. Higher values give softer transitions between shadows and highlights but might introduce halos. Lower values will reduce the size of halos but may lead to an artificial look. As said, bilateral filter is much less prone to halo artifacts.

compress

This slider controls how strong the effect extends to midtones; high values reduce the effect to the extreme shadows and highlights; low values cause strong adjustments also to

midtone. You normally only need to touch this parameter if you want to limit the effects to the extreme shadows and highlights; increase the value in this case. At 100% this module has no visible effect any longer as only absolute black and absolute white are affected.

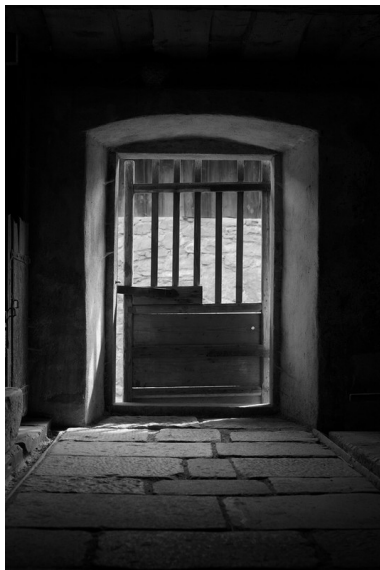
shadows color adjustment

This slider controls the color saturation adjustment made to shadows; high values cause saturation enhancements on lightened shadows; low values cause desaturation on lightened shadows. It is normally safe to leave this at its default of 100%. This gives a natural saturation boost on shadows – similar to the one you would also expect in nature if shadows would receive more light.

highlights color adjustment

This slider controls the color saturation adjustment made to highlights; high values cause saturation enhancements on darkened highlights; low values cause desaturation on darkened highlights. Often highlights do not contain enough color information to give convincing colors when darkened. You might need to play a bit with this parameter in order to find the best fitting value depending on your specific image; but be aware that sometimes results still might not be fully satisfying.

Examples



Original image exposed for the outer sunlit wall to avoid clipped highlights. As a consequence the interior of the barn has pitch black shadows.



Shadows get lightened; highlights are untouched; overall effect toned down a bit by *blend mode* "normal" and an opacity of 65%.

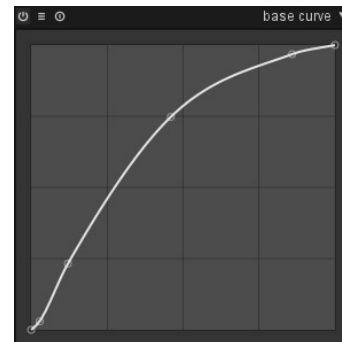


Resulting image.

3.4.1.4. Base curve

Overview

Camera sensors provide data in linear RGB format, the original image appears flat and dull. That's the reason why camera manufacturers apply their characteristic base curves to the RAW data when they generate in-camera JPEG images with better colors and contrast. darktable comes with base curve presets that mimic the curves of various manufacturers. These are automatically applied to RAW images according to the manufacturer ID found in EXIF data.



Usage

You can adjust an existing base curve or create a new one. The base curve is defined by two or more nodes. You can click on any node and drag to modify it. You can also create additional nodes by clicking on a curve segment between two nodes. In order to remove a node drag it outside of the widget area.

Tip: If you intend to take full manual control of the tonal values with the *tone curve* module or the *zone system* module (see Section 3.4.2.3, “Tone curve” and Section 3.4.2.4, “Zone system”) it may be easier to leave the image in linear RGB. Disable the *base curve* module in this case.

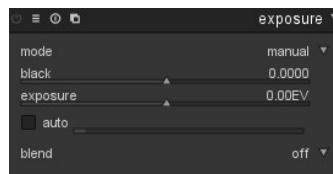
scale

This combobox toggles between “linear” and “logarithmic” view. In the double logarithmic view more space is given to the lower values allowing a more fine-grained adjustment of the shadows.

3.4.1.5. Exposure

Overview

This module is used to tweak the exposure. It is directly linked to the histogram panel. Indeed, if you correct exposure graphically, using the *histogram* (see Section 3.3.6, “Histogram”), you automatically activate the exposure module. The histogram simply acts as a view for the exposure module.



You can activate multiple instances of this module each with different parameters acting on different parts of the image which you select by a drawn mask (see Section 3.2.4, “Multiple instances” and Section 3.2.7, “Drawn mask”). The histogram is always linked to the lowest instance in pixelpipe.

Usage

This module is responsible for one of the most basic steps in each raw image development. An exposure adjustment value allows you – within certain limits – to correct for under- or overexposure. A shift by 1EV is equivalent to a change of exposure time by a factor of 2.

Positive exposure corrections will make the image brighter. As a side effect noise level gets higher. Depending on the basic noise level of your camera and the ISO value of your image, positive exposure compensations with up to 1EV or 2EV still give reasonable results.

Negative exposure corrections will make the image darker. Given the nature of digital images this can not correct for fully blown out highlights but allows to reconstruct data in case that only some of the RGB channels are clipped (see also Section 3.4.1.8, “Highlight reconstruction”).

A black level adjustment is a basic tool to increase contrast and pop of an image. The value defines at what threshold dark gray values are cut off to pure black. Use with care as the clipped values can not be recovered in other modules further down the pixelpipe. Please also have a look at the *tone curve* module (see Section 3.4.2.3, “Tone curve”) and the *levels* module (see Section 3.4.2.2, “Levels”) which can produce similar results with less side effects as they come later in pixelpipe.

mode

Defines the mode of operation. Currently darktable only supports “manual” operation. Future versions of darktable will additionally allow automatic exposure correction.

black

Adjust the black level.

exposure

Adjust the exposure correction [EV].

auto

Activating this checkbox makes darktable calculate a correct exposure for the rectangular view appearing in the centre of the image. You can draw your own selection using your mouse. An adjustment slider to the right of the auto exposure checkbox lets you define what percentage of bright values are to be clipped out in the calculation.

3.4.1.6. Contrast Brightness Saturation

Overview

This module offers a very basic tool for adjusting an image's contrast, brightness and saturation.



Usage

The module has sliders for each of the three affected attributes. In their neutral position (zero) the image remains unchanged. Shifting sliders left to negative values reduces contrast, brightness and saturation, respectively. Shifting right to positive values leads to an increase.

Much more versatility for contrast and brightness adjustment is offered by the *tone curve*, *levels*, and *zone system* modules (see Section 3.4.2.3, "Tone curve", Section 3.4.2.2, "Levels", and Section 3.4.2.4, "Zone system"). Likewise you may adjust color saturation in a more detailed way with the *tone curve*, *color contrast*, and *color zones* modules (see Section 3.4.2.3, "Tone curve", Section 3.4.3.4, "Color contrast", and Section 3.4.3.7, "Color zones").

contrast

This slider adjusts the image's contrast.

brightness

This slider adjusts the image's brightness.

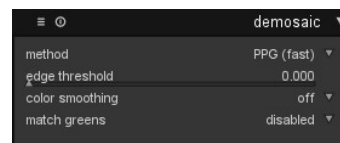
saturation

This slider adjusts the color saturation.

3.4.1.7. Demosaic

Overview

This module allows you to control how the demosaic is processed.



Usage

Demosaic is an essential step of any raw image development process.

A detailed description would be beyond the scope of this manual. In a nutshell, the sensor cells of a digital camera are only able to record different levels of lightness, not different color. In order to get a color image, each cell is covered by a color filter, either in red, green or blue. Due to the color sensitivity of the human vision, there are two times more green cells than red or blue. Filters are arranged in a certain mosaic, called Bayer pattern. Therefore each pixel of your image originally only has information about one color channel. Demosaic reconstructs the missing color channels by interpolation with data of the neighboring pixels. For further reading see the Wikipedia article on the Bayer filter [http://en.wikipedia.org/wiki/Bayer_filter].

As interpolation is prone to produce artifacts, various different demosaic algorithms have been developed in the past. Artifacts would typically be visible as moiré-like patterns when you strongly zoom into your image. Currently darktable supports PPG, AMAZE, and VNG4. All these algorithms produce high quality output with a low tendency to artifacts. AMAZE is reported to sometimes give slightly better results. However, as AMAZE is significantly slower, darktable uses PPG as a default. VNG4 produces the softest results of the three algorithms, but if you see "maze" artifacts, try VNG4 to eliminate them.

There are a few cameras whose sensors do not use a Bayer filter. Cameras with an "X-Trans" sensor have their own set of demosaicing algorithms. The default algorithm for X-Trans sensors is VNG. For noticeably better quality (at the cost of slower processing), choose Markesteijn 1-pass or (for the best quality, but the slowest processing) Markesteijn 3-pass.

Some further parameters of this module can activate additional averaging and smoothing steps. They might help to reduce remaining artifacts in special cases.

Demosaic is always applied when exporting images. Demosaic is done on monitor display only when zoom is greater than 50% or when the according preference setting "demosaicing for zoomed out darkroom mode" (see Section 7.2, "Core options") is set accordingly. Else color channels are taken from neighboring pixels without an expensive interpolation.

method

Set the demosaic method. darktable currently supports PPG, AMAZE, and VNG4 for Bayer sensors. For X-Trans sensors darktable supports VNG, Markesteijn 1-pass, and Markesteijn 3-pass.

edge threshold

Set the threshold for an additional median pass. Defaults to "0" which disables median filtering. This option is not shown for X-Trans sensors.

color smoothing

Activates a number of additional color smoothing passes. Defaults to "off".

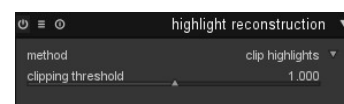
match greens

In some cameras the green filters have slightly varying properties. This parameter adds an additional equalization step to suppress artifacts. Available options are "disabled", "local average", "full average" and "full and local average". This option is not shown for X-Trans sensors.

3.4.1.8. Highlight reconstruction

Overview

This module tries to reconstruct color information that is usually clipped because of incomplete information in some of the channels. If you do nothing, your clipped areas are often toned to the not clipped channel. For example, if your green and blue channels are clipped, then your image will appear red in the clipped areas.



Usage

You can choose between three methods for highlight reconstruction.

“Clip highlights” analyses each pixel having at least one channel clipped. Then it sets all channels to the minimum value found among the channels. Effectively this converts all clipped highlights to neutral grey tones. This method is most useful in cases where clipped highlights occur in non-colored areas like clouds in the sky.

“Reconstruct in LCh” analyses each pixel having at least one channel clipped and transforms the information in LCh color space to linearly mix the channels. This method is not able to reconstruct any color information – the reconstructed highlights will all be neutral grey. This works fairly well with a high-contrast base curve (such as most manufacturers apply to their jpg), which renders highlights desaturated. This method is a good option on naturally desaturated things such as clouds.

“Reconstruct color” uses an algorithm that transfers color information from the unclipped surroundings into the clipped highlights. This method works very well on areas with homogeneous colors and is especially useful on skin tones with smoothly fading highlights. It fails in certain cases where it produces maze-like artifacts at highlights behind high-contrast edges, such as fine, well-exposed structures in front of overexposed background (for instance ship masts or flags in front of the blown-out sky).

Tip: for highlight reconstruction to be effective you need to apply a negative EV correction in the exposure module (see Section 3.4.1.5, “Exposure”). If you want to avoid a general darkening of your image you can use darktable's mask feature in that module to limit the EV correction to only the highlights (see Section 3.2.7, “Drawn mask” and Section 3.2.8, “Parametric mask”).

method

Choose the method for highlight reconstruction.

clipping threshold

Manually adjust the clipping threshold against magenta highlights. The default is usually satisfactory without any need for additional adjustments.

3.4.1.9. White balance

Overview

This module is used to set the white balance. You have three ways to interact with it: (a) Set up tint and temperature, (b) define the value of each channel, or (c) choose from predefined white balances.



Usage

tint

Alter the colour tint of the image, from magenta (value < 1) to green (value > 1). The channel sliders will be updated when you adjust this parameter.

temperature

Set the color temperature (in Kelvin). The channel sliders will be updated when you adjust this parameter. darktable derives the color temperature from the EXIF data using some model assumptions. The value given is not meant to be authoritative. In the end only the updated channel values determine how the image is modified.

red, green and blue channels

Set the channel values on a scale from 0 to 8.

preset

Select a preset white balance.

camera white balance (default)

White balance reported by the camera.

spot white balance

Select a square area in your image containing mostly grey pixels. The white balance is calculated based on the selected area.

passthrough

Show without adjusting for white balance.

camera presets

Camera specific white balance presets. Examples: direct sunlight, flash, cloudy, shade and a number of indoor lighting options.

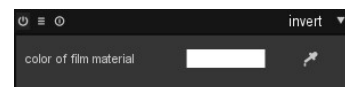
finetune

Some cameras offer additional finetuning parameters if one of the camera presets is selected. Depending on camera white balance, can be adjusted in steps within a certain range. The adjustments are usually towards yellow (value < 1) or blue (value > 1).

3.4.1.10. Invert


Overview

The main purpose of this module is to invert scanned negatives.



Usage

color of film material

The only control element of this module is a color selector which allows to adjust for different colors of your film material. Clicking on the colored field will open a color selector dialog which allows to define a color in HSL or RGB color space. You can also activate a color picker by pressing  and take a color probe from your image – preferably from the unexposed border of your negative.

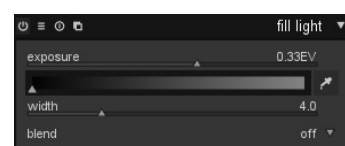
3.4.2. Tone group

This group contains modules that operate on the tonal values of an image, modulating brightness while leaving color values intact.

3.4.2.1. Fill light

Overview

This module allows local modification of the exposure based on pixel lightness.




Usage

Pushes exposure by increasing lightness with a Gaussian curve of a specified width, centered on a given lightness.

exposure

Sets fill-light exposure in [EV].

center

Sets the median lightness impacted by the fill-light. A color picker is activated by pressing . It shows the picked lightness value in the gradient bar, which helps find the desired center value.

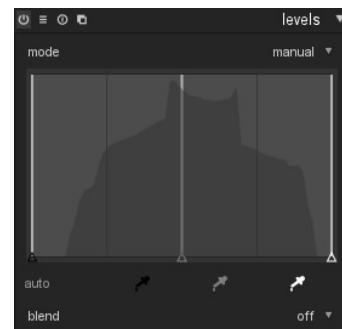
width

Sets the width of the Gaussian curve. This number is expressed in zones, with the whole dynamic range being 10 zones. As the Gaussian curve is symmetric, only even numbers can be entered.

3.4.2.2. Levels

Overview

A tool for adjusting black, white, and mid-gray points. This module is especially useful if the histogram of an image does not span the whole horizontal range, from pure black to pure white.




Usage

The levels tool offers two modes of operation: "manual" and "automatic".

In "manual" mode the levels tool shows a histogram of the image, and displays three bars with handles. Dragging the handles modifies the tones in the image. Those bars control the black, middle gray and white points in absolute values of image lightness (the L value from Lab).

You can move the black and white bars to match the left and right borders of the histogram, which will make the output image span the full available tonal range. A previously flat looking image will get more contrast and pop.

Moving the middle bar will modify the middle gray tones. Shifting it left will make the image look brighter, shifting it right will make it darker. This is often referred to as a change of image gamma.

There are three color pickers in black, gray, and white, available by pressing the respectively colored  icon. You can use them to sample the corresponding level directly from the image.

The “auto” button autoadjusts the black and white point and puts the gray point exactly in the mean between them.

In “automatic” mode the module automatically analyses the histogram of the image, detects the left and right histogram borders, and lets you define the black point, the gray point and the white point in terms of *percentiles* [<http://en.wikipedia.org/wiki/Percentile>] relative to these borders.

Tip: Under certain conditions, especially highly saturated blue light sources in the frame, the levels module may produce black pixel artifacts. See the *gamut clipping* option (Section 3.4.3.10, “Input color profile”) on how to mitigate this issue.

mode

Set the mode of operation of this module. Defaults to “manual”.

black

Sets the black point in percentiles relative to the left border of the histogram (only “automatic” mode).

gray

Sets the gray point in percentiles relative to the left and right borders of the histogram *after* having applied the black point and white point corrections (only “automatic” mode).

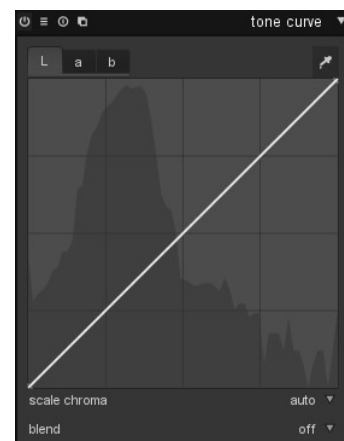
white

Sets the white point in percentiles relative to the right border of the histogram (only “automatic” mode).


3.4.2.3. Tone curve

Overview

This module is a classic digital photography tool. Unlike other image manipulation software, however, darktable's tone curve acts in Lab color space. Thus, it offers three independent curves for L, a, and b channels.



Usage

In its default state, curves will be straight lines, defined by few anchor nodes. You can move the nodes with your mouse to modify the curve. You can also generate new nodes by clicking on the curve. Up to 20 nodes per curve can be defined. To remove a node, move it out of the widget area. A color picker is activated by pressing  and will show the picked

values in the graph. Numerical Lab values of input and output (see below) at the selected spot are shown on top left of the widget.

L-channel curve

The tone curve in L-channel works on Lightness. For a better overview a lightness histogram is displayed in the diagram.

The horizontal line represents the input image pixels' lightness. The vertical line represents the lightness of the output image pixels. A straight line does not change anything. A point above the default diagonal increases the lightness, whereas a point under decreases it. Shifting the center of the curve upwards will lighten the image, shifting it downwards will darken the image. An S-like curve will enhance the contrast of the image.

a/b-channel curves

The curves in the a and b channels work on color values. They are only displayed and active if the *scale chroma* combobox is set to "manual". The horizontal line represents the color channel value of the input image pixels. The vertical line represents the color channel value of the output image pixels. Positive a-values correspond to more magenta colors; negative a-values correspond to more greenish colors. Positive b-values correspond to more yellowish colors; negative b-values correspond to more blueish colors.

A straight line does not change anything. Shifting the center of the curve will give the image a color tint: shifting a-channel upwards gives a magenta tint; shifting b-channel upwards gives a yellow tint; shifting a-channel downwards gives a green tint; shifting b-channel downwards gives a blue tint.

Increasing/decreasing the steepness of the curves, without shifting its center, will increase/decrease the color saturation of the respective channel. With properly defined curves you can exert fine control on color saturation, depending on the input pixel's colors.

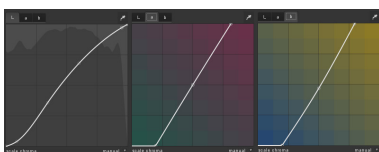
scale chroma

darktable does an automatic adjustment of color saturation, if this combobox is set to "auto". The level of this adjustment depends on the pixel's color values and its L-value modification by the L-channel tone curve. It is designed to give an overall boost in color saturation, if the L-curve gives a contrast boost. Look at blend mode "coloradjustment" to adjust the strength of this effect (see Section 3.2.6, "Blending operators"). If this combobox is set to "manual", you can modify color saturation using the curves in channels a and b.

Examples



Original image



Tone curve settings. Please note how the center node of our b-curve was shifted down to negative values. This gives the image its blue tint.



Resulting image

3.4.2.4. Zone system

Overview

This module is another way to change the lightness of your image, based on Ansel Adams' system. It allows modification of a zone's lightness taking into account the effect on the adjacent zones. It divides the lightness range into a user-defined number of zones.



Usage

Following the concept of Ansel Adams the lightness (based on the L channel from Lab) is divided into a number of zones ranging from pure black to pure white. These zones are displayed in a zonebar. The number of zones can be changed by mouse-scrolling on that bar (defaults to 10 zones).

The zonebar is split horizontally with the upper part showing the zones of the module's output and the lower part the according zones of the module's input. In its default state both parts are fully aligned. While the output zones are static you can left click and drag a control point in the input zones to modify the zonemapping. Shifting a control point proportionally expands the zones on one side and compresses the zones on the other side. Any already existing control point stays in place, effectively preventing changes to the zones beyond. Use right click to remove a control point.

The preview shows the image broken down in zones. When hovering above a zone on the zonebar, that zone – either from input or output – is highlighted on the preview.

Examples



The original image.



Here, the darker and lighter zones were compressed to increase contrast, then the upper parts of darker zones were expanded to increase their visual impact.



3.4.2.5. Local contrast

Overview

This module allows enhancing local contrast. It uses the unnormalized bilateral filter, and works on the L channel from Lab.



Usage

Local contrast boosts details of your image, much like the *equalizer* does (see Section 3.4.4.2, "Equalizer"). However, it is easier to use as it does not require you to work on different frequency bands.

coarseness

Make the details you want to adjust finer or coarser.

contrast

How strongly the algorithm distinguishes between brightness levels. Increasing the value results in a more contrasty look.

detail

Add or remove detail. Higher values will increase local contrast.

Example

Before



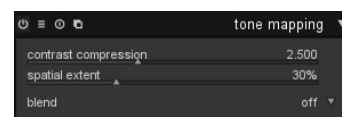
After, a little overdone to demonstrate the effect. Use this sparingly, to avoid a cheap, overprocessed look.



3.4.2.6. Tonemapping

Overview

This module compresses the tonal range of HDR images, so they fit into the limits of a normal, low dynamic range image, using Durand's 2002 algorithm.



darktable can import HDR images if they come in OpenEXR, RGBE or PFM format or as a DNG generated by darktable's HDR creation mechanism (see Section 2.3.6, "Selected image(s)").

Usage

The underlying algorithm uses a bilateral filter to decompose an image into a coarse base layer and a detail layer. The contrast of the base layer is compressed, while the detail layer is preserved, then both layers are re-combined.

contrast compression

Sets the contrast compression level of the base layer. A higher compression will make the image fit a lower dynamic range.

spatial extent

Sets the spatial extent of the bilateral filter. Lower values cause the contrast compression to have stronger effects on image details.

3.4.2.7. Global tonemap

Overview

This module implements another approach to compressing the tonal range of an HDR image into the limited tonal range of a typical LDR output file. It offers several implementations of global tonemap operators.



Usage

Global tonemapping processes each pixel of an HDR image, without taking the local surrounding into account. This is generally faster than local tonemapping, as implemented in the *tonemapping* module but might lead to less convincing results with very high dynamic scenes. As an enhancement to the original operators, darktable can preserve detail of the input image, and transfer it back to the output image.

operator

Reinhard, Filmic and Drago global tonemap operators are available for use. Depending on the selected operator, different parameters can be adjusted. Some operators are fully self-adjusting, and do not require specific controls.

bias

Only offered for the *Drago* operator. This parameter influences the contrast of the output image. It is an essential parameter for adjusting the compression of high values and the visibility of details in dark areas. According to the original paper, a value of 0.85 is recommended as a starting point.

target

Only offered for the *Drago* operator. This is a scale factor to adjust the global image brightness to the brightness of the intended display. It is measured in cd/m^2 , and should match the according value of your output device. Higher values lead to a brighter image, while lower values lead to a darker image.

detail

Offered as an addition to all operators. This parameter controls how much detail is preserved and transferred back into the output image after tonemapping.

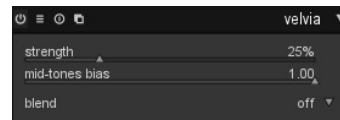
3.4.3. Color group

Modules for working specifically on your image's color are found here in the color group.

3.4.3.1. Velvia

Overview

The velvia module enhances image saturation. Its effect is tailored to increase saturation less on lower saturated pixels than on highly saturated pixels.



Usage

strength

This slider controls the strength of the effect.

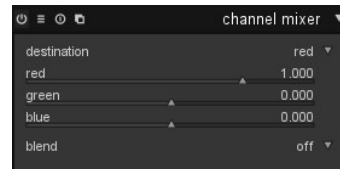
mid-tones bias

Velvia can reduce its effect for mid-tones to avoid unnatural skin tones. The mid-tones bias slider controls this selectivity; reducing its value reduces mid-tone protection and makes the overall velvia effect stronger.

3.4.3.2. Channel mixer

Overview

This module is a powerful tool to manage channels. It accepts red, green and blue channels as an input. As output it provides red, green, blue, gray, hue, saturation and lightness channels.



Usage

First select your output channel and then set the amount each input channel feeds into that output channel.

Examples



For skin tones the blue channel tends to represent detail, with red tending to also have smoother tones than green. Therefore tonal rendering is controlled by how we blend of the three input channels.



Here a monochrome portrait is produced by simply selecting the grey channel as output. A smooth skin tone is achieved by reducing the blue channels input and also emphasizing the red channels input relative to green. An RGB mix of 0.9, 0.3, -0.3 was used together with an 0.1 EV exposure increase to lighten the image.



In this example an RGB mix of 0.4, 0.75, -0.15 uses more green than red, bringing back some features. We still reduce the blue channel in the mix to de-emphasize unwanted skin texture.

Table of mixing values for some b/w films

Classic black and white films have different characteristic color responses. Select gray as output mixing channel, and try out the values suggested below for your favorite film type.

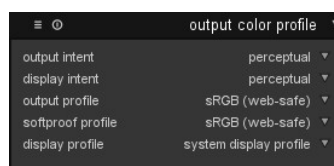
Film Type	Red	Green	Blue
AGFA 200X	0.18	0.41	0.41
Agfapan 25	0.25	0.39	0.36
Agfapan 100	0.21	0.40	0.39
Agfapan 400	0.20	0.41	0.39
Ilford Delta 100	0.21	0.42	0.37
Ilford Delta 400	0.22	0.42	0.36
Ilford Delta 3200	0.31	0.36	0.33
Ilford FP4	0.28	0.41	0.31
Ilford HP5	0.23	0.37	0.40
Ilford Pan F	0.33	0.36	0.31
Ilford SFX	0.36	0.31	0.33
Ilford XP2 Super	0.21	0.42	0.37
Kodak T-Max 100	0.24	0.37	0.39
Kodak T-Max 400	0.27	0.36	0.37
Kodak Tri-X 400	0.25	0.35	0.40
Normal Contrast	0.43	0.33	0.30
High Contrast	0.40	0.34	0.60
Generic B/W	0.24	0.68	0.08

3.4.3.3. Output color profile

Overview

This module manages the output profiles for display and export as well as the rendering intent to be used when mapping between the different color spaces.

darktable comes with pre-defined profiles sRGB, AdobeRGB, XYZ and linear RGB but you can provide additional profiles by placing these in `$DARKTABLE/share/darktable/color/out` and `$HOME/.config/darktable/color/out`. `$DARKTABLE` is used here to represent your chosen darktable installation directory and `$HOME` your home directory.



Usage

A configuration parameter “always try to use littlecms2” in darktable's *core options* (see Section 7.2, “Core options”) defines how darktable renders colors for display and export. If the configuration parameter is disabled darktable uses a simplified and very fast internal rendering algorithm. If the option is checked the external library *littlecms2* [<http://www.littlecms.com/>] with higher accuracy and significantly higher processing overhead is used instead.

output intent

Sets the rendering intent for output/export. You can easily override this setting whenever you do exports from lighttable mode.

<i>perceptual</i>	Suited to pictures as it maintains the relative position of colors. This is usually the best choice.
<i>relative colorimetric</i>	Out-of-gamut colors are converted to colors having the same lightness, but different saturation. Other colors remain unmodified.
<i>saturation</i>	Saturation is kept but lightness is slightly changed.
<i>absolute colorimetric</i>	Keep the white point.

Only rendering with littlecms2 gives you a choice of rendering intent. The option is hidden, if darktable's internal rendering routines are used.

display intent

Sets the rendering intent for your display. See above for available options.

output profile

Sets the color profile for output/export, causing darktable to render colors with this profile. darktable embeds the profile data into the output file if supported by the file format – this allows other applications reading the file to correctly interpret its colors.

Some file formats – namely PNG – do not support embedded profiles by their specification. However, there is an established de-facto standard to embed profiles which dark-

table adheres to. Beware that some applications might fail to correctly handle those profiles and render colors incorrectly.

As not all applications, e.g. image viewers, are aware of color profiles, a general recommendation is to stick to *sRGB* as the default output profile. You should only deviate from *sRGB* if this is really required and if you know what you are doing.

softproof profile

Sets the color profile for softproofing.

softproof allows you to preview your image rendered using a printer profile so as to see how colors will end up on the final print and is toggled on and off by pressing *ctrl-s*.

Likewise *gamut check* is toggled on and off by pressing *ctrl-g*; this function highlights in cyan all pixels out of gamut with respect to the selected softproof profile.

softproof and *gamut check* are mutually exclusive modes that can be activated at any place in darkroom mode and when this module is in focus the status is displayed below the image.

Tip: at other times it may not be obvious if the *softproof* mode is still active or not. If in doubt press *ctrl-g*, which will switch to *gamut check* and be clearly distinguishable by cyan marked pixels. Press *ctrl-g* again and you are back to normal display mode.

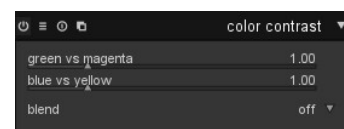
display profile

Sets the color profile for the display. The additional option “system display profile” is the preferred setting when working with a calibrated display; the profile is taken either from your system’s color manager or from your X display server. In *gui options* (see Section 7.1, “GUI options”) you can specify which method to use.

3.4.3.4. Color contrast

Overview

The color contrast module provides simplified control for changing the contrast or separation of colors between either green/magenta or blue/yellow axis.



Usage

Higher values increase color contrast, lower values decrease it. The effect of this module's sliders are similar to applying a steepened or flattened a- or b-curve in module *tone curve* (see Section 3.4.2.3, “Tone curve”).

green vs. magenta

Changes color contrast for green versus magenta.

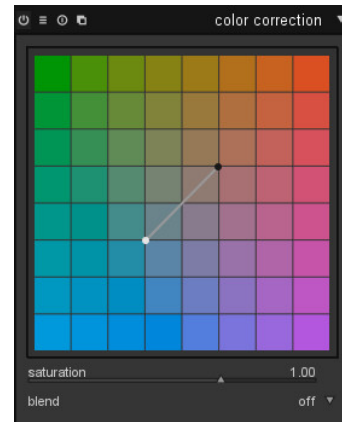
blue vs. yellow

Changes color contrast for blue versus yellow.

3.4.3.5. Color correction

Overview

This module can be used to modify the global saturation, give a tint to the image or to split tone it.



Usage

color board

For split toning drag the white dot to the desired highlight tint and then select a tint for shadows with the dark spot. For a simple global tint set both spots to the same color.

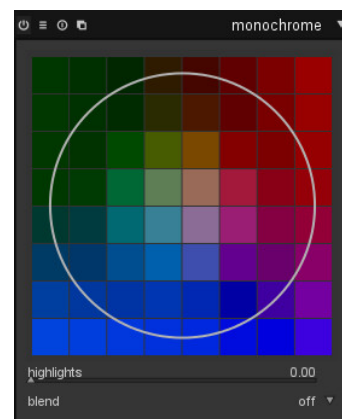
saturation

Use the saturation slider to correct the global saturation.

3.4.3.6. Monochrome

Overview

This module is a quick way to convert an image into black and white and provides a variable color filter for that conversion.



Usage

The default central location of the filter has a neutral effect but dragging it to an alternate position applies a filter analogously to taking a b&w photograph through a conventional color filter.

As well as position you can change the filter size by scrolling with your mouse wheel. This makes the filter's range of hues more or less selective.

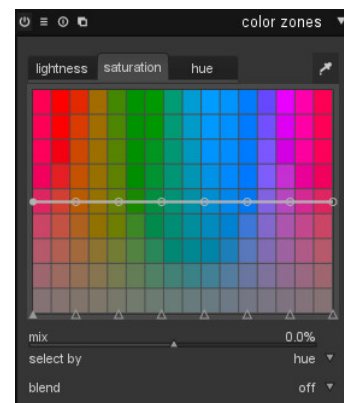
Tip: First reduce the filter size to concentrate its effect and move it across the hue pallet to find the best filter value for your desired image rendition. Then expand the filter to include more hues and thus more natural tonality.

Under certain conditions, especially highly saturated blue light sources in the frame, this module may produce black pixel artifacts. See the *gamut clipping* option (Section 3.4.3.10, "Input color profile") on how to mitigate this issue.

3.4.3.7. Color zones

Overview

This module selectively modifies the colors in your image. It is highly versatile and allows every transformation possible in the LCh colorspace.



Usage

The horizontal axis represents the range of values you can work on. The vertical axis shows the modifications you can apply by designing a curve. For both horizontal and vertical axes you can work on lightness, saturation or hue.

You can click on any of the eight nodes on the curve and drag to adjust it vertically. A circle indicates how strong adjacent nodes will be affected. Use the scroll wheel of your mouse to change the circle diameter. You can also use the eight controlpoints (triangles which define the vertical value of the nodes) at the bottom to adjust the curve.

A color picker is activated by pressing  and will show the picked values in the diagram.

tabbed controls

You can define curves for each of the three channels "lightness", "saturation", and "hue" individually.

select by

Defines the horizontal axis, i.e. the range of values you work on. You can chose between "lightness", "saturation", and "hue" (default). Changing this parameter resets any defined curve to a straight horizontal line.

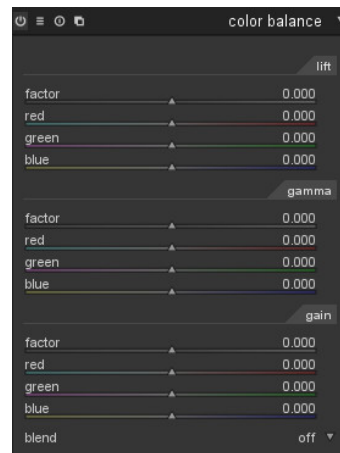
mix

Use this parameter to tune the strength of the overall effect.

3.4.3.8. Color balance

Overview

This module offers a versatile tool for adjusting an image's color balance.



Usage

For each of the three primary colors – red, green, and blue – there are three sliders “lift”, “gamma” and “gain” to control dark tones, mid tones and high tones, respectively. Changing these parameters adds offsets to the individual RGB channels and influences the color balance of the image.

Additional “factor” sliders act on all colors at once. Their effect is similar to the controls of the *levels* module (see Section 3.4.2.2, “Levels”).

The way to best envision the module's way of working is if you consider “lift”, “gamma” and “gain” as parameters influencing the properties of a tone curve which – in its original state – is a straight line going diagonally from bottom left to top right.

The lift parameter adjusts the left axis intercept of the line – this point can be shifted from its default location with zero abscissa to points with negative or positive ones. The gamma parameter modifies the line shape – the default shape is linear and can be turned into a convex curve with increased or into a concave curve with lowered gamma values. Finally, the gain parameter defines the steepness of the curve – by default the curve runs diagonally, higher values lead to a steeper curve and lower values to a flatter one.

Side note: although this module acts on RGB colors its location in pixelpipe puts it into the Lab color space. Accordingly the module converts from Lab to RGB, does its color adjustments, and then converts back to Lab.

lift

Adjusts the value of dark colors with individual sliders for each RGB color and with a “factor” slider acting on all three colors at once.

gamma

Adjusts the mid tones with individual sliders for each RGB color and with a “factor” slider acting on all three colors at once.

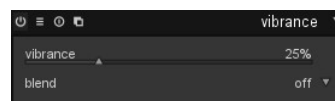
gain

Adjusts the high tones with individual sliders for each RGB color and with a “factor” slider acting on all three colors at once.

3.4.3.9. Vibrance

Overview

Vibrance is a widely used term in image processing but the mechanism and end result differ from program to program. Vibrance in darktable saturates and brings down the lightness of the most saturated pixels to make the colors more vivid.



Usage

Vibrance only has one parameter which controls the amount applied.

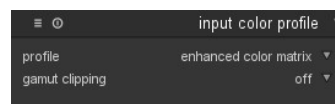
vibrance

The amount of vibrance to apply to the image.

3.4.3.10. Input color profile

Overview

This module can be used to override darktable's automatic allocation of input color profile if there is an alternative that more closely matches your original image's color space.



Usage

In this module you define the input color profile, i.e. how colors of your input image are to be interpreted. You also have an option to have colors confined to a certain gamut in order to mitigate some (infrequent) color artifacts.

profile

Choose the profile or color matrix to apply, darktable offers many widespread matrices along with an enhanced matrix for some camera models. The enhanced matrices were processed by the darktable team in order to provide a look closer to the manufacturer's.

You can also supply your own input ICC profiles and put them into `$DARKTABLE/share/darktable/color/in` or `$HOME/.config/darktable/color/in`. `$DARKTABLE` is used here to represent darktable's installation directory and `$HOME` your home directory. One common source of ICC profiles is the software that is shipped with your camera; it often contains profiles specific to your camera model. You may need to activate module *unbreak input profile* (see Section 3.4.3.11, "Unbreak input profile") to use your extra profiles.

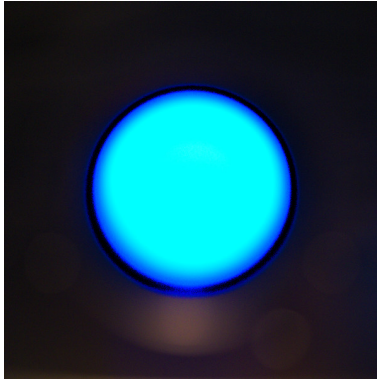
If your input image is a low dynamic range file like JPEG, or a raw in DNG format, it might already contain an embedded ICC profile which darktable will use as a default. You can always overrule darktable and select a different profile. Select "embedded icc profile" to restore the default.

gamut clipping

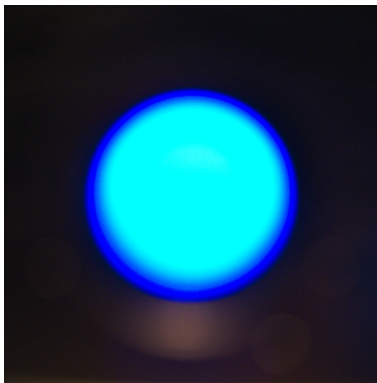
This combobox lets you activate a color clipping mechanism. In most cases you can leave it at its default "off" state. However, if your image shows some specific features like highly saturated blue light sources, gamut clipping might be useful to avoid black pixel artifacts. See Section 3.2.10.3, "Possible color artifacts" for more background information.

You can select from a list of RGB profiles. Input colors with a saturation that exceeds the permissible range of the selected profile get clipped to a maximum value. "linear Rec2020 RGB" and "Adobe RGB (compatible)" allow for a broader range of unclipped colors, while "sRGB" and "linear Rec709 RGB" produce a tighter clipping. You should select the profile that prevents artifacts while still maintaining highest color dynamics.

Examples



Close-up of a blue light source (LED) with gamut clipping off. The activated levels module produces a ring of black pixel artifacts.

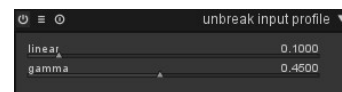


The same image and processing with gamut clipping set to "linear Rec2020 RGB".

3.4.3.11. Unbreak input profile

Overview

This module adds a correction curve to image data, which is required if you have selected certain input profiles in module *input color profile*.



Usage

If you decide in module *input color profile* to use an ICC profile from the camera manufacturer, a correction curve very frequently needs to be pre-applied to image data – or else the final output looks much too dark. This extra processing is not required if you use darktable's standard or enhanced color matrices. The correction curve is defined with a linear part extending from the shadows to some upper limit and a gamma curve covering mid-tones and highlights. For further reading please also have a look at darktable's neighbouring project UFRaw [<http://ufraw.sourceforge.net>].

linear

Set the upper limit for the region counted as shadows and where no gamma correction is performed. Typically values between 0.0 and 0.1 are required by the profile.

gamma

Set the gamma value to compensate your input profile. Often the required value is 0.45 (the reciprocal of 2.2 gamma used by some manufacturer's profile).

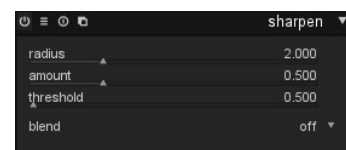
3.4.4. Correction group

The correction group contains the modules that will correct typical problems in an photo such as hotpixels, spot removal, noise, lens correction among others. This group also includes the basic sharpening tools.

3.4.4.1. Sharpen

Overview

This is a standard UnSharp Mask (USM) tool for sharpening the details of an image.



Usage

This module works by enhancing the contrast around edges and thereby enhances the impression of sharpness of an image. In darktable this module is only applied to the L-channel in Lab color space.

radius

USM applies a gaussian blur to your image as part of its algorithm. This controls the blur radius which in turn defines the spatial extent of edge enhancement. Too high values will lead to ugly over-sharpening.

amount

This controls the strength of the sharpening.

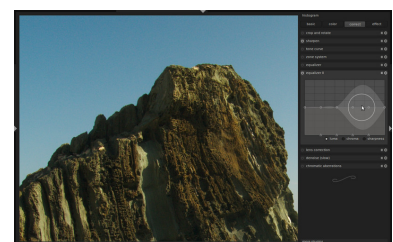
threshold

Contrast differences below this threshold are excluded from sharpening. Use this to avoid amplification of noise.

3.4.4.2. Equalizer

Overview

This versatile module can be used to achieve a variety of effects, such as: bloom, denoising, clarity, and local contrast enhancement. It works in the wavelet domain and parameters can be tuned for each frequency band separately.



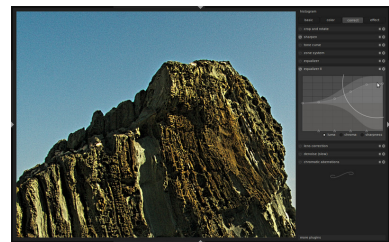
Usage

Each frequency band can be tweaked independently. In particular, you can adjust contrast boost and denoise threshold splines for both lightness and chromaticity, as well as the acuteness ("edge") of the wavelet basis on each frequency scale.

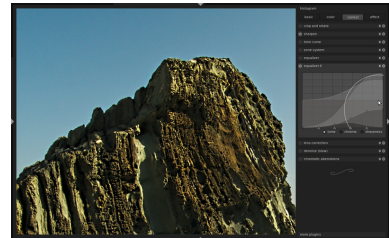
Each spline can be dragged with a proportional edit approach; use the mouse wheel to adjust the radius in which your changes will have an effect. The transparent area indicates where you would drag the spline with the current mouse position and radius. The small little triangles on the x-axis can be moved to alter the x-position of the spline nodes.



Drag the upper line (bright circles, here for the lightness channel) to affect local contrast. Pulling it up, as shown here, will result in a contrast boost for that frequency band. Higher frequencies, i.e. smaller details, are to the right of the grid. Pulling it down works, too.



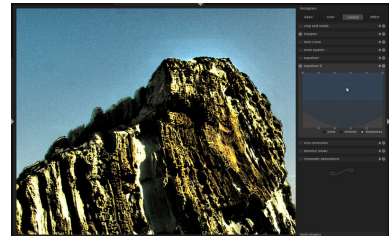
The bottom spline (black circles) is used to perform denoising. It adjusts the wavelet shrinkage threshold for each frequency band. Pull it up to see the effect. In this example, the noise which has been amplified by local contrast enhancement is removed.



This screen shows the effect of the edge parameter. It is here pulled down to zero for all bands. This is effectively a regular à trous wavelet, without edge detection, and results in the characteristic halos around sharp edges in the image.



This image is the other extreme. The wavelet basis now oversharpen, which results in ugly gradient reversals near the ridge of the rock.



Note that the edge parameter only affects the wavelet basis, not the image directly. You will have to change some denoise/contrast boost parameters to see an effect following adjustments to the edge parameter.

This module additionally has a “mix” slider below the spline GUI. Adjusting the slider will upscale or downscale the splines on the y-axis. The slider was added as a convenience tool to help you modify the strength of the effect. It is not a module parameter in itself; when you leave darkroom mode all changes will be consolidated into the spline curves.

Have a look at the presets where there are a broad variety of examples that will provide a good starting point to gain an intuitive understanding of the controls. Among others there are two presets to enhance an image's “clarity”.

3.4.4.3. Denoise – profiled

Overview

This module offers an easy to use and – at the same time – highly efficient denoise operation. Under the hood it applies (your choice of) a non-local means or edge-aware wavelet denoise algorithm with parameters specifically profiled for certain camera models and ISO settings.



Usage

The darktable team, with the help of many users, has measured noise profiles for various cameras. Differentiated by ISO settings we evaluated how the noise statistics develop with brightness for the three color channels. Our set of profiles already covers about 100 popular camera models from all major manufacturers.

profile

Based on EXIF data of your RAW file, darktable will automatically determine the camera model and ISO setting. If found in its database, the corresponding noise profile will be used. If your image has an intermediate ISO value, the statistical properties will be interpolated between the two closest datasets in the database, and this interpolated setting will show up as the first line in the combo box. You also have the option to manually overwrite this selection to suit your personal preferences better. The top-most entry in the combo box brings you back to the profile darktable deems most suited.

mode

This module can eliminate noise with two different core algorithms. “non-local means” is a bit better suited to tackle luma (lightness) noise; “wavelet” has its strength in eliminating chroma (color) noise. If needed you can apply two instances of this module (see Section 3.2.4, “Multiple instances”). The “non-local means” instance should be combined with blend mode “lightness” or “HSV lightness”; the “wavelet” instance with blend mode “color” or “HSV color”. For more information on blend modes have a look at Section 3.2.6, “Blending operators”.

patch size

This slider is only available if mode “non-local means” is selected. It controls the size of the patches being matched when deciding which pixels to average (see also Section 3.4.4.4, “Denoise – Non local means”). Setting this to higher values can give more sharpness. Processing time will stay about the same.

strength

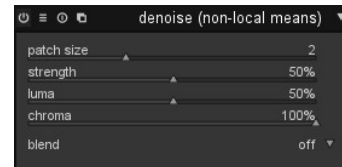
This parameter is here to fine-tune the strength of the denoise effect. The default value has been chosen to maximize the peak signal to noise ratio. It's mostly a matter of taste

if you prefer a rather low noise level at the costs of a higher loss of detail, or if you accept more remaining noise in order to have finer structures better preserved within your image.

3.4.4.4. Denoise – Non local means

Overview

This is a denoise algorithm, which will work on chroma and/or luma.



Usage

This module reduces noise in your image but preserves structures. This is accomplished by averaging each pixel with some surrounding pixels in the image. The weight of such a pixel in the averaging process depends on the similarity of its neighborhood with the neighborhood of the one pixel to be denoised. A patch with a certain size is used to measure that similarity. As denoising is a resource hungry process, it slows down pixelpipe processing significantly; consider activating this module late in your workflow.

patch size

The radius of the patch for similarity evaluation.

strength

The strength of the denoise. Higher values lead to a stronger effect.

luma

Amount of denoise to apply to luma. Select carefully in order not to lose too much structure.

chroma

Amount of denoise to apply to chroma. You can be much more aggressive with this parameter compared to luma.

3.4.4.5. Denoise – bilateral

Overview

This module is used to denoise high ISO pictures. It is flagged as a slow module due to its high resource consumption, both in terms of CPU cycles and in terms of memory usage. Quite counter-intuitively, the greater the values for sliders, the lesser resources.



Usage

This module reduces noise in your image but preserves sharp edges. This is accomplished by averaging pixels with their neighbors, taking into account not only the geometric dis-

tance but also the distance on the range scale, i.e. differences in the RGB values. As de-noising is a resource hungry process, it slows down pixelpipe processing significantly; consider to activate this module late in your workflow.

radius

Set the spatial extent of the gaussian blur.

red

Blur intensity for red channel.

green

Blur intensity for green channel.

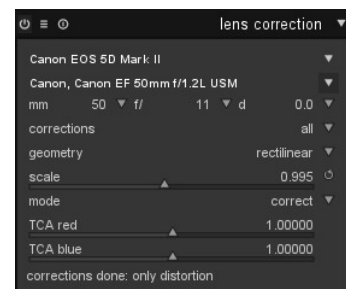
blue

Blur intensity for blue channel.

3.4.4.6. Lens correction

Overview

This module is able to correct certain lens flaws, namely distortions, transversal chromatic aberrations (TCA) and vignetting. It relies on the external library lensfun, which comes with correction profiles for many (but not all) common cameras end lenses.



Usage

In order to perform lens corrections the module uses EXIF data of your image to identify the specific camera/lens combination and collects the needed correction parameters from a profile in lensfun's database.

camera

The camera make and model as determined by EXIF data. You can override this manually and select your camera from a hierarchical menu.

lens

The lens make and model as determined by EXIF data. You can override this manually and select your lens from a hierarchical menu. This is mainly needed for pure mechanical lenses.

photometric parameters: focal length, aperture, focal distance

Corrections additionally depend on certain photometric parameters that are read from EXIF-data: focal length (needed for distortion, TCA, vignetting), aperture (needed for TCA,

vignetting) and focal distance (needed for vignetting). Many cameras do not record focal distance in their EXIF data; most likely you need to set this manually.

You can manually override all automatically selected parameters. Either take one of the predefined values from the pull-down menu; or – with the pull-down menu still open – just type in your own value.

If your system's lensfun library has no correction profile for the automatically identified camera/lens combination the controls for the three photometric parameters are not displayed, and you get a warning message instead. You may try to find the right profile yourself by searching for it in the menu. If there is no matching profile for your lens, please visit this lens calibration service [<http://www.darktable.org/2013/07/have-your-lens-calibrated/>] offered by Torsten Bronger, one of darktable's users. Alternatively you may go to lensfun's home page [<http://lensfun.berlios.de>] and learn how to generate your own set of correction parameters. Don't forget to share your profile with the lensfun team!

corrections

This combobox gives you a choice about which corrections (out of distortion, TCA and vignetting) darktable shall apply. Change this from its default "all", if your camera has already done some internal corrections (e.g. of vignetting), or if you plan to do certain corrections with a separate program.

geometry

In addition to the correction of lens flaws, this module can change the projection type of your image. Set this combobox to the aimed projection type, like "rectilinear", "fish-eye", "panoramic", "equirectangular".

scale

This slider allows you to adjust the scaling factor of your image. Pressing the auto scale button (right to the slider) will let darktable find the best fit to avoid black corners.

mode

The default behavior of this module is to correct lens flaws. Switch this togglebutton to "distort" in order to simulate the behavior of a specific lens (inverted effect).

TCA red

This slider allows to override the correction parameter for TCA. You can also use this slider to manually set the parameter in case the lens profile does not support TCA correction. Look out for colored seams at features with high contrast edges and adjust this parameter and the following one to minimize those seams.

TCA blue

This slider allows to override the correction parameter for TCA. You can also use this slider to manually set the parameter in case the lens profile does not support TCA correction.

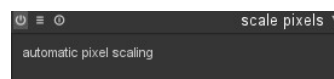
corrections done

You will sometimes observe that for a given camera/lens combination only part of the possible corrections (distortion, TCA, vignetting) are supported by lensfun's profiles. This message box will tell you what corrections have actually been applied.

3.4.4.7. Scale pixels

Overview

Some cameras like the Nikon D1X have rectangular instead of the usual square sensor cells. Without correction this would lead to distorted images. This module applies the needed scaling.



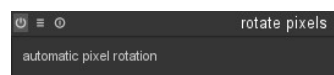
Usage

darktable detects images that need correction by their EXIF data and automatically activates this module. For other images the module always remains disabled. The module has no parameters.

3.4.4.8. Rotate pixels

Overview

The sensors of some cameras like the Fujifilm FinePix S2Pro, F700, and E550 have a diagonally oriented Bayer pattern instead of the usual orthogonal layout. Without correction this would lead to a tilted image with black corners. This module applies the needed rotation.



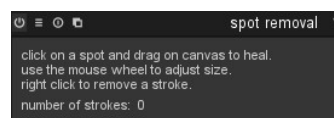
Usage

darktable detects images that need correction by their EXIF data and automatically activates this module. For other images the module always remains disabled. The module has no parameters.

3.4.4.9. Spot removal

Overview

Spot removal allows you to correct an area in your image by using another area as model.



Usage

This module uses some of the shapes that are offered in drawn masks, namely circles, ellipses and path shapes. The user interface and the controls are the same as in drawn mask and explained in more detail in Section 3.2.7, "Drawn mask".

Select the desired shape by clicking the corresponding icon, then click on the canvas to choose the area to be healed, i.e. the *target area*.

The *source area* is preliminary positioned at a location with a default distance to the target. Source area and target area can then be shifted independently until the result matches your expectations. An arrow helps to tell source from target area.

Use the shape specific controls to adjust its size, its border width, and other attributes.

Right-click on a shape to delete it.

Collapse the module to complete the changes.

Examples



Let's use this portrait as example; we want to remove some dirt and unwanted catchlight from camera pop-up strobe.



I have marked all the spots that I want to remove from the image with circle shapes and appropriately selected source areas.



And here is the result image of the spotremoval.

3.4.4.10. Raw denoise

Overview

Raw denoise allows you to perform denoising on pre-demosaic data. It is ported from *dcraw* [<http://www.cybercom.net/~dcoffin/dcraw/>].



Usage

noise threshold

Set the threshold for noise detection. Higher values lead to stronger noise removal and higher loss of image detail.

3.4.4.11. Dithering

Overview

This module eliminates some of the typical banding artifacts which can occur, when darktable's internal 32-bit floating point data are transferred into a discrete 8-bit or 16-bit integer output format for display or file export.



Banding is a problem which can arise, when an image is downsampled into a lower bit-depth. Downsampling happens regularly, when darktable displays or exports the results of a pipeline. In order to avoid banding, you may activate this module. As dithering consumes significant resources this module is disabled by default.

Although banding is not an inherent problem of any of darktable's modules, some operations may provoke it as they produce a lightness gradient in the image. To mitigate possible artifacts you should consider to activate dithering when using the *vignette* and the *graduated density* module, respectively (see Section 3.4.5.4, "Vignetting" and Section 3.4.5.13, "Graduated Density"). This is especially relevant for images with extended homogeneous areas like cloudless sky. Also when using a *gradient mask* (see the section called "gradient") you should watch out for possible banding artifacts.

Usage

If you export images with a reduced width/height and want best dithering results, please make sure that you de-activate option "do high quality resampling during export" in *core options* (see Section 7.2, "Core options"), else the final scaling step will counteract dithering.

Viewing from some distance an image dithered into a very low bit depth (like "floyd-steinberg 1-bit b&w") will give the impression of a homogeneous grayscale image. We try to mimic this impression in darktable when you look at zoomed-out images in the center view, in the navigation window and for thumbnails. This is accomplished by dithering those images into a higher number of grayscale levels. Note that as a consequence the histogram – which is derived from the navigation window – will show this increased number of levels and is no longer a full match of the output image.

method

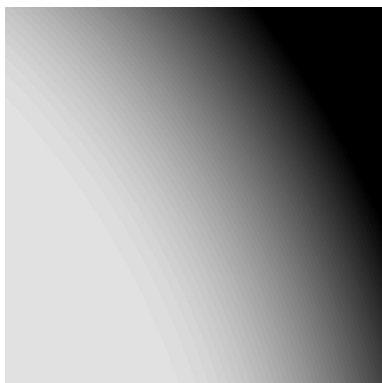
This combobox sets the dithering method. Floyd-Steinberg error diffusion – with some typical output bit depths – and random noise dithering are both supported. Floyd-Steinberg systematically distributes quantization errors over neighboring pixels, whereas random dithering just adds some level of randomness to break sharp tonal value bands. The default setting is "floyd-steinberg auto", which automatically adapts to the desired output format.

damping

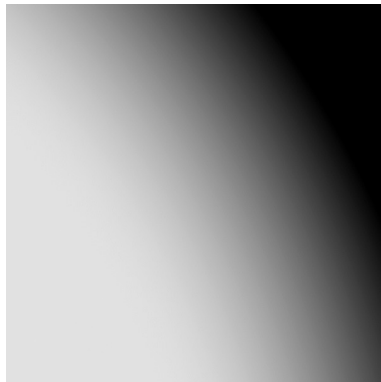
This slider is only displayed if you choose method "random". It controls the level of added random noise expressed as a damping factor in a $10 \cdot \log_2$ basis. A value of -80 is a good fit for 8-bit output formats and -160 for 16-bit ones.

Examples

The visibility of the following examples depends on the quality of your monitor or the print quality.



Banding artifact caused by vignetting (100% crop of a 8-bit PNG; effect heavily exaggerated by strong contrast enhancement).

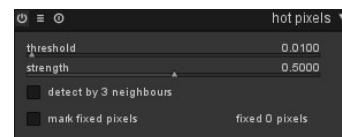


The same image area, processed as above but with activated Floyd-Steinberg dithering.

3.4.4.12. Hotpixels

Overview

This module is able to automatically detect and eliminate hotpixels. Hotpixels are pixels which failed to record light level correctly. Detected hotpixels are replaced by an average value of their neighbors.



Usage

You control the detection sensitivity with the threshold parameter and the level of elimination with the strength parameter.

threshold

The threshold of the detection, i.e. how strong a pixel's value needs to deviate from its neighbors to be regarded as a hotpixel.

strength

The strength of blending hotpixels with their surrounding.

detect by 3 neighbours

This will extend the detection of hotpixels, it will even regard a pixel as hot if a minimum of only three (instead of four) neighbor pixels deviate by more than the threshold level.

mark fixed pixels

This options will mark the pixels that have been corrected. It also displays the count of detected and fixed pixels.

3.4.4.13. Chromatic aberrations

Overview

This module allows you to correct chromatic aberrations.



Usage

The module has no parameters. On activation it will automatically try to optimize away visible CA's.

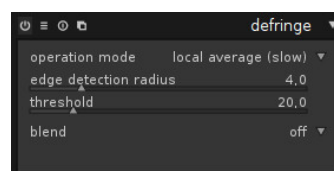
The underlying model assumes as input an uncropped photographic image. The module is likely to fail when you zoom into the image, as in that case it will only receive parts of your photograph as input in darktable's pixelpipe. As a consequence, chromatic aberrations do not get corrected properly in the center view. This limitation only applies to interactive work, not to file export.

This module currently only works for images recorded with a Bayer sensor (which is the sensor used in the majority of cameras).

3.4.4.14. Defringe

Overview

This module is designed to remove purple or any other color of fringing which often results from Longitudinal Chromatic Abberations (LCA), also known as Axial Chromatic Abberations.



Usage

This module helps removing fringe via edge-detection. Where pixels are detected as a fringe, it rebuilds the color from lower-saturated neighboring pixels.

operation mode

Set the operation mode for detecting fringes. "global average" is usually the fastest but might show slightly incorrect previews in high magnification. It might also protect the wrong regions of color too much or too little by comparison with local averaging. "local average" is slower because it computes local color references for every pixel, which might protect color better than global average and allows for rebuilding color where actually required. The "static" method does not use a color reference but directly uses the threshold as given by the user.

edge detection radius

Set the spatial extent of the gaussian blur used for an edge detection. The algorithm uses the difference of gaussian-blurred and original image as an indicator for edges (a special case of the "difference of gaussians" edge detection). Try increasing this value if you either want a stronger detection of the fringes or the thickness of the fringe edges is too high.

threshold

Sets the threshold over which the edge of a pixel is counted as a "fringe". The colors of the affected pixels will be rebuild from neighboring pixels. Try lowering this value if there is not enough fringe detected and try increasing this value if too many pixels are desaturated. You may additionally want to play around with the edge detection radius.

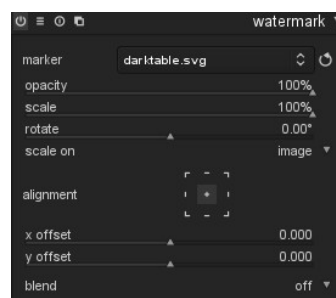
3.4.5. Effect group

In the effect group you will find modules with a more artistic touch.

3.4.5.1. Watermark

Overview

The watermark module provides a way to render a vector-based overlay onto your image. Watermarks are standard SVG documents and can be designed using *Inkscape* [<http://www.inkscape.org>].



The SVG processor of darktable also substitutes strings within the SVG document, which gives the opportunity to include image dependent information in the watermark such as aperture, exposure time and other metadata.

User-designed watermarks are placed into the directory `$HOME/.config/darktable/watermarks`. Once in place, use the reload button at the right of the watermark file name to update the list of available watermarks to use.

Here follows a list of available variable strings that is supported for substitution within the svg document.

<code>\$(DARKTABLE.NAME)</code>	The application name
<code>\$(DARKTABLE.VERSION)</code>	The version of darktable
<code>\$(IMAGE.ID)</code>	The unique image id within current library
<code>\$(IMAGE.FILENAME)</code>	The image filename
<code>\$(IMAGE.EXIF)</code>	The image exif string
<code>\$(EXIF.DATE)</code>	The image date
<code>\$(EXIF.DATE.SECOND)</code>	Seconds from the image EXIF data
<code>\$(EXIF.DATE.MINUTE)</code>	Minutes from the image EXIF data
<code>\$(EXIF.DATE.HOUR)</code>	Hours from the image EXIF data (24h)
<code>\$(EXIF.DATE.HOUR_AMPM)</code>	Hours from the image EXIF data (12h, AM/PM)
<code>\$(EXIF.DATE.DAY)</code>	Day from the image EXIF data
<code>\$(EXIF.DATE.MONTH)</code>	
<code>\$(EXIF.DATE.SHORT_MONTH)</code>	
<code>\$(EXIF.DATE.LONG_MONTH)</code>	
<code>\$(EXIF.DATE.SHORT_YEAR)</code>	Abbreviated year from the image EXIF data (2013 is "13")
<code>\$(EXIF.DATE.LONG_YEAR)</code>	Full year from the image EXIF data
<code>\$(DATE)</code>	Current system date
<code>\$(DATE.SECOND)</code>	Current system time seconds
<code>\$(DATE.MINUTE)</code>	Current system time minutes
<code>\$(DATE.HOUR)</code>	Current system time hours (24h)
<code>\$(DATE.HOUR_AMPM)</code>	Current system time hours (12, AP/PM)

<code>\$(DATE.DAY)</code>	Current system time day
<code>\$(DATE.MONTH)</code>	Current system time month
<code>\$(DATE.SHORT_MONTH)</code>	
<code>\$(DATE.LONG_MONTH)</code>	
<code>\$(DATE.SHORT_YEAR)</code>	Current system time year (abbreviated)
<code>\$(DATE.LONG_YEAR)</code>	Current system time year
<code>\$(EXIF.MAKER)</code>	The maker of camera model
<code>\$(EXIF.MODEL)</code>	The camera model
<code>\$(EXIF.LENS)</code>	The specific lens used
<code>\$(Xmp.dc.creator)</code>	The creator string
<code>\$(Xmp.dc.publisher)</code>	The publisher string
<code>\$(Xmp.dc.title)</code>	The title of the image
<code>\$(Xmp.dc.description)</code>	The description of the image
<code>\$(Xmp.dc.rights)</code>	The rights assigned to the image

Usage

marker

Choose the watermark of interest. You can use the reload button next to the combobox to update the list with all newly added watermarks.

opacity

Set the opacity of the watermark's rendering.

scale

Scale the watermark pixel-independently.

rotate

Set the rotation angle of the watermark.

scale on

Defines the reference for the scale parameter. The default setting "image" scales the watermark relative to the horizontal image size. Alternatively you can scale the watermark relative to the "larger border" and "smaller border", respectively.

alignment

Use these controls to align the watermark to any edge or center of the image.

x offset

Pixel-independent offset relative to the choice of alignment on the x-axis.

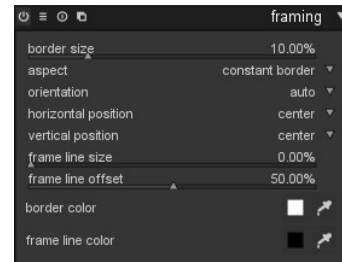
y offset

Pixel-independent offset relative to the choice of alignment on the y-axis.

3.4.5.2. Framing

Overview

This module is an artistic feature to generate a frame around your image. The frame consists of a border with a user defined color and a frame line inside that border, which has another user defined color. There are various options for you to control the geometry of your frame.



Usage

border size

This slider controls the size of the frame in percent of the underlying full image.

aspect

With this combobox you can choose between different aspect ratios for the final output of this module, i.e. underlying image plus frame.

orientation

If you select a non-square aspect ratio this combobox defines the orientation – portrait or landscape. Set to “auto” if you want darktable to select the most reasonable orientation based on the underlying image.

horizontal position

Select from a set of pre-defined ratios where you want your underlying image be positioned on the horizontal axis. You can also right click and enter your own ratio as “x/y”.

vertical position

Select from a set of pre-defined ratios where you want your underlying image be positioned on the vertical axis. You can also right click and enter your own ratio as “x/y”.


frame line size

The percentage of the frame line size relative to the border size at its smallest part.


frame line offset

Where the frame line is positioned relative to the underlying image. Select a value of 0 for a frame line touching the image, 100% for a frame line touching the outer border limits.

border color

Clicking on the colored field will open a color selector dialog which allows one to define a color for the border in HSL or RGB color space. You can also activate a color picker by pressing  and take a color probe from your image.

frame line color

Clicking on the colored field will open a color selector dialog which allows one to define a color for the frame in HSL or RGB color space. You can also activate a color picker by pressing  and take a color probe from your image.

Examples

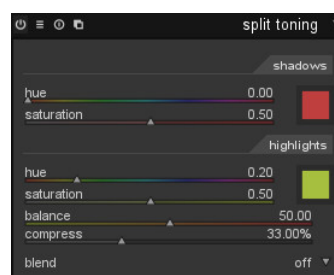


Example image with a user defined frame.

3.4.5.3. Splittoning

Overview

darktable's splittoning method creates a two color linear toning effect where the shadows and highlights are represented by two different colors. In the example image below you can see an original black and white image and one where a splittoning effect is applied with blue in shadows and a yellowish color in highlights.



Compared to traditional splittoning our module has more parameters to influence its behavior. We have parameter "balance", which offsets the 50% gray level in your image – at your choice – more to the shadows or more to the highlights. Additionally, with parameter "compression" you can compress toning in the shadows and highlights and leave a gap in the mid-tones, which remain untouched by the effect.

The splittoning module does not convert images to black and white and has limited benefits on color images. So, if you want to do traditional splittoning, use the *monochrome* module (see Section 3.4.3.6, "Monochrome") to make the image black and white before playing around with splittoning effect.

Usage

shadows and highlights color

These controls are used to set the color of the splittoning effect, you select the desired color and saturation for both shadows and highlights, you can also click the color preview box to bring up a common color picker dialog.

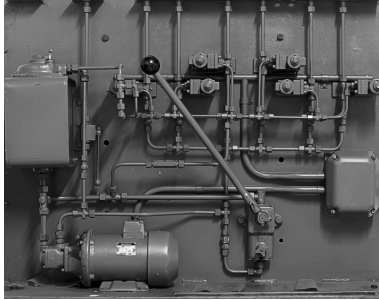
balance

This parameter represents the ratio of toning between shadows and highlights. For a value of 50% half of the lightness range in image is used for shadows toning and the other half for highlights toning.

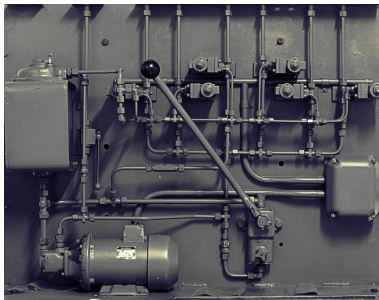
compression

Compression is a percentage of total lightness range that is not affected by color toning. Default value is set to 33%; this is not the default behavior of an original splittoning which would be 0% compression. The choice of 33% as a default is to invite you experimenting with these parameters and how it extends the original splittoning method.

Examples



Original black and white image.

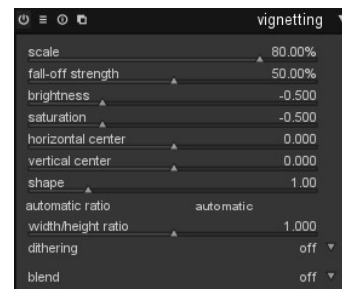


Splittoning with blue shadows and yellow highlights.

3.4.5.4. Vignetting

Overview

This module is an artistic feature which creates vignetting (modification of the brightness/saturation at the borders).



Usage

The vignetting module has an extensive set of parameters to precisely tune its effect. It also will display graphical controls within the image if the module is in focus. Give it a try to get a feeling of how it works. On-screen controls and parameter sliders will stay in sync.

This module is known to provoke banding artifacts under certain conditions; you should consider to activate the *dithering* module (see Section 3.4.4.11, "Dithering").

scale

Set the radius of the vignetting area.

fall-off strength

Sets the progressiveness of the fall-off. Higher values will cause a steeper transition.

brightness

Sets the intensity of brightening (positive values) or darkening (negative values).

saturation

Controls how strong colors become when desaturated or saturated in the darkened or brightened vignetting area.

horizontal center

Shifts the center of the vignetting area horizontally.

vertical center

Shifts the center of the vignetting area vertically.

shape

Influences the shape of the vignetting area. The default value of 1 causes a circular or elliptical area. Smaller values will shift the shape into a more square one; higher values turn it into a cross-like shape.

automatic ratio

Click this button to automatically adjust the width/height ratio of the vignetting area to the aspect ratio of the underlying image. The vignetting area will typically become elliptical.

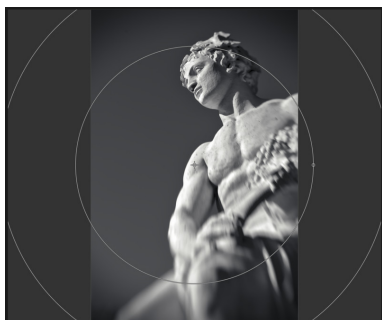
width/height ratio

Manually adjust the width/height ratio of the vignetting area.

dithering

With this combobox you can activate random noise dithering to overcome banding artifacts caused by vignette gradients. Select “8-bit output” to prevent banding on monitor display and for JPEGs. When set to “16-bit output”, only a little dithering will be applied, just strong enough to compensate for banding on the fine grained 16-bit level. This feature is mostly obsolete by our new module *dithering* (see Section 3.4.4.11, “Dithering”).

Examples



An image with vignetting and with graphical vignetting controls displayed.

3.4.5.5. Soften

Overview

This module is an artistic feature that creates a softened image, commonly known as the Orton effect.



Usage

Michael Orton achieved his results on slide film by using two exposures of the same scene: one well exposed and one overexposed; he then used a darkroom technique to blend those into a final image where the overexposed image was blurred.

This module is almost a copy of Orton's analogue process into the digital domain. You can control brightness and blur with the provided parameters; we also add a control for saturation of the overexposed image for more play.

size

Set the size of blur of the overexposed image in the process, the bigger the softer.

saturation

Set the saturation of the overexposed image.

brightness

Expressed in [EV], the brightness slider selects the increase in brightness.

mix

Controls the mix of the overexposed image and the overall effect.

Examples



This is the original image, use it as reference for the changes below...



In this image I used the default values, and added 0.33EV to brightness for a little more light in the soft layer.

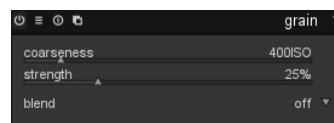


This version is the same as above but with 25% saturation.

3.4.5.6. Grain

Overview

This module is an artistic feature which simulates the grain of a film.



Usage

The grain is processed on the L channel from CIELAB.

coarseness

Set the grain size, which has been scaled to simulate an ISO number.

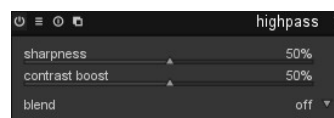
strength

Set the strength of the effect.

3.4.5.7. Highpass

Overview

Highpass acts as a high pass filter. The primary usage for this filter is in combination with a blending operator. Try out blend mode "soft light" to get highpass sharpening. Use the opacity slider to adjust the strength of the effect or use a *drawn mask* (see Section 3.2.7, "Drawn mask") or *parametric mask* (see Section 3.2.8, "Parametric mask") to limit the effect to only parts of your image.



Usage

sharpen

Set the sharpness. The higher, the more details.

contrast boost

Set the contrast boost.

3.4.5.8. Lowpass

Overview

A lowpass filter (eg. gaussian blur) with additional control of the outcome both of contrast and saturation. The primary usage for lowpass filter is in combination with a *blending operator* (see Section 3.2.6, "Blending operators"). Try out the preset named "local contrast mask" with an "overlay" blending operation.



Usage

This module offers enormous artistic potential, albeit, with results that are sometimes difficult to predict.

radius

Set the radius of the blur.

soften with

This combobox defines the blur algorithm; you can choose between "gaussian" blur (default) and "bilateral" filter. The latter leads to an edge preserving blur. "gaussian" will blur all image channels: L, a and b. "bilateral" will only blur L channel.

contrast

Changes the contrast. Negative values result in an inverted negative image. Higher absolute values increase contrast; lower absolute values reduce contrast. A value of zero leads to a neutral plane.

brightness

Changes the brightness. Negative values result in a darker image. Positive values increase the image's brightness.

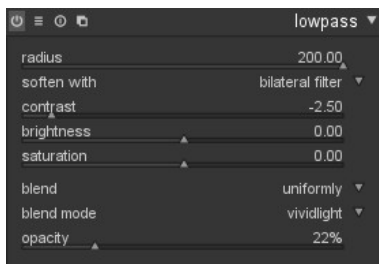
saturation

Changes the color saturation. Negative values result in complementary colors by inverting the a/b-channels. Higher absolute values increase color saturation; lower absolute values reduce color saturation. A value of zero leads to a desaturated black&white image.

Examples



The original image, already heavily processed. The boat is almost a silhouette.



Bilateral blur with high radius. Desaturated, inverted and with high contrast.



Intermediate result after lowpass filter ...



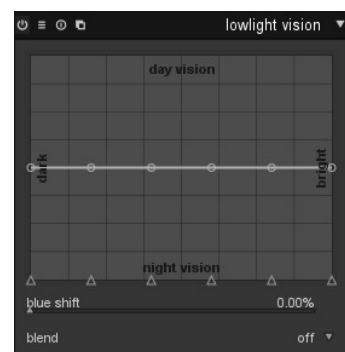
... and final image after this was applied with blend mode "vividlight".

3.4.5.9. Low light

Overview

The *low light* module allows one to simulate human lowlight vision, thus providing ability to make low-light pictures look closer to reality. It can also be used to perform a day to night conversion.

The idea is to calculate a scotopic vision [http://en.wikipedia.org/wiki/Scotopic_vision] image which is perceived by rods rather than than cones in the eye under low light. Scotopic lightness then is mixed with photopic value (regular color image pixel) using some blending function. Also this module is able to simulate the Purkinje effect [http://en.wikipedia.org/wiki/Purkinje_effect] by adding some blueness to the dark parts of the image.



Usage

This module comes with several presets. Give them a try to get a better feeling how it works.

curve

The horizontal axis is about pixel lightness from dark (left) to bright (right). The vertical axis represents the kind of vision from night vision (bottom) to day vision (top)

blue

Set the blue hint in shadows (Purkinje effect).

Examples



Image 1. This is the original image.



Image 1. With low light module on.

Image 2. This is the original image.



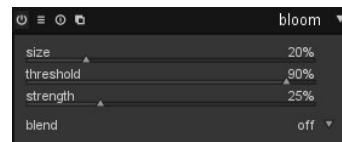
Image 2. With low light module on.



3.4.5.10. Bloom

Overview

This module boost highlights and creates a soft blooms them over the image, hence the name of the effect. There are numerous ways to use this module depending on the image's actual scenery lighting.



Usage

Starting from the default settings change the strength value for a pleasant look, then change the size to control the spread of light.

size

Represents the spatial extent of the bloom effect.

threshold

Set the threshold for the increase in brightness.

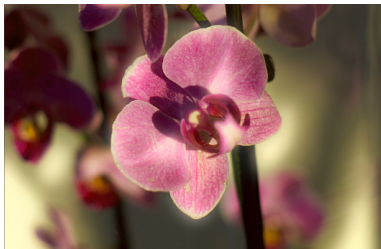
strength

Set the strength of overlightning for the effect.

Examples



This is the original image, use it as reference for the changes below...

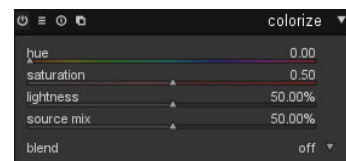


Here we have chosen to use a size of 10%, which is a rather small radius for the soft light spread. We boosted up the strength to 50% for a more exaggerated effect.

3.4.5.11. Colorize

Overview

This module is an artistic feature that adds a solid layer of color to your image.



Usage

Several parameters control the effect of this module. Much more versatility can be reached if you apply blending and masks. (see Section 3.2.5, "Blending").

hue

Selects the hue of the color layer.

saturation

Selects the color saturation of shadow tones.

lightness

Selects the lightness of the color layer.

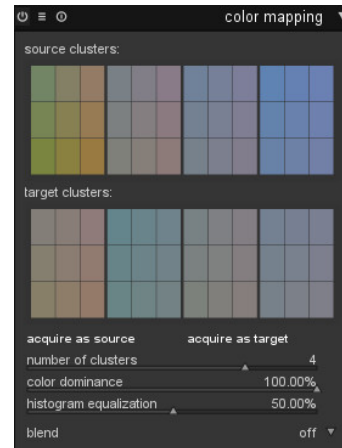
source mix

This slider controls how the lightness of the input image is mixed in. If you set this to zero a uniformly colored plane will result.

3.4.5.12. Color mapping

Overview


This module transfers the look and feel from one image onto another. It statistically analyses the color characteristics of a source image and a target image. The colors of the source image are then mapped to corresponding colors of the target image.



Usage

To use this module two steps are required.

First you open the source image in darkroom mode and acquire its color characteristics by pressing the “acquire as source” button. A set of color clusters is generated and displayed in the “source clusters” area. Each cluster is represented by a set of color swatches with the mean value in the center surrounded by swatches indicating the color variance within that cluster. The clusters are sorted in ascending order by their weight, i.e. the number of pixels that contribute to the clusters.

Next you open your target image in darkroom mode. darktable has remembered the previously collected source clusters; if they are not yet displayed, press the  button. You now press the “acquire as target” button to generate a corresponding set of color clusters for your target image, which gets displayed in the “target clusters” area.

When both source and target clusters are collected an automatic color mapping is applied to the target image. In its default settings the overall effect is quite exaggerated. A set of sliders gives you control of the effect's strength. You can also use blending operator “normal” to tame the effect (see Section 3.2.6, “Blending operators”). As the *color mapping* module comes early in the pixelpipe, you can finetune the colors with modules like *tone curve* (see Section 3.4.2.3, “Tone curve”) or *color correction* (see Section 3.4.3.5, “Color correction”).

acquire as source/target

Press these buttons to generate color clusters for the source and target image, respectively. The processing takes a few seconds during which the GUI remains unresponsive.

number of clusters

Sets the number of color clusters to use. If you change this parameter all collected color clusters are reset and need to be acquired anew.

color dominance

This parameter controls the mapping between source and target clusters. At the lowest value mapping is based on color proximity. This typically leads to very subtle effects on the target image. At the maximum value mapping is based on the relative weight of the

color clusters – dominant colors of the source image are mapped to dominant colors of the target image. This typically leads to a very bold effect. In-between values incrementally shift between the extremes.

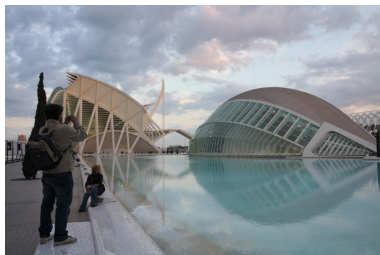
histogram equalization

Besides mapping of color characteristics this module can modify the target image's contrast by matching its histogram with the histogram of the source image. This slider controls the extent of the effect.

Examples



The source image taken shortly after sunset under front lighting conditions.



The target image taken in the afternoon with a partly cloudy sky. Our goal is to transfer the evening atmosphere of the source image.

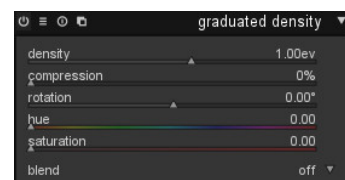


The target image with color mapping applied. A number of 2 clusters was used. "color dominance" is set to 100% for a bold but still credible effect. "histogram equalization" is set to 80%.

3.4.5.13. Graduated Density

Overview

This module aims at simulating a graduated density filter, in order to correct exposure and color in a progressive manner.



Usage

The module uses a gradient to modify the exposure and the color cast of the image in a non-homogeneously manner.

This module is known to provoke banding artifacts under certain conditions; you should consider to activate the *dithering* module (see Section 3.4.4.11, "Dithering").

density

Set the density of the filter in [ev]. A low value underexposes slightly whereas a high value creates a strong filter.

It is expressed as [ev] that is equivalent to *f-stops*. Lens filters are often referred as ND2, ND4, ND8 and so on. Each time you add an [ev] you double the ND. So ND2 is 1 ev, ND4 is 2 ev, and so on. You can also express it in optical density or transmittance. The table below sums up the different approach for the most common filters:

ND	[ev] or f-stop	absorbance	transmittance
ND2	-1	0.3	50%
ND4	-2	0.6	25%
ND8	-3	0.9	12.5%
ND400	-9	2.7	0.195%

compression

Set progressiveness of the gradient. A low value creates a smooth transition, whereas a high value makes the transition abrupt.

hue

Set the hue to add a color cast to the gradient.

saturation

Set the saturation to add a color cast to the gradient.

position

You can set the position of the gradient directly on the image by moving the white line. For fine tuning, you can also use the rotation slider. Negative values turn clockwise.



Examples

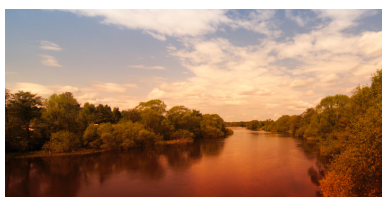
Here is an example that shows various options of darktable's graduated density filter:



This is the original image with a pretty overexposed sky, use it as reference for the changes below...



And now we have added a neutral ND8 filter which does a pretty good job on the image..



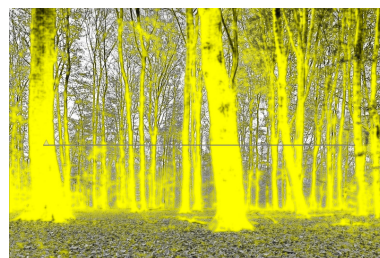
And at last, we added an orange colored filter rotating it -180 degrees, applying it on water/trees for a more artistic use of the filter.

darktable's graduated density filter is a powerful tool. Nevertheless, hardware filters have some advantages over a pure software solution. With a physical GND filter you can in fact reduce the dynamic range of your scene to make it better fit the limits of your camera sensor.

In this example a hardware GND filter (Hitech ND0.6, soft edge) helped to prevent over-exposure in the sky and tree tops, while at the same time getting a well exposed image of the ground. A rather disturbing element is the decay of brightness in the tree trunks from bottom to top.



darktable's graduated density filter together with the parametric mask feature (see Section 3.2.8, "Parametric mask") comes in handy. We can add a brightness gradient that is just inverted in relation to the hardware filter. As we only want to compensate the unnatural decay of brightness in the tree trunks, we combine the module with a suited blend mask.



The resulting image.



Tip: If you know you intend to use the graduated density filter before actually making a shot with your camera you might want to underexpose by one or two thirds of an f-stop to make sure detail remains in the highlights. When all detail has truly been blown out the graduated density filter cannot produce a pleasing results, this is a limitation that is inherent to digital postprocessing. For instructions on how to intentionally underexpose, please consult your camera's manual, look for "exposure compensation".

3.5. Examples

3.5.1. Converting to black and white

3.5.1.1. Overview

Black and white conversion can be achieved in several ways with darktable. Indeed, darktable comes with a lot of modules, especially for color manipulation. In this manual, I will show you 4 ways to perform a black and white conversion.



3.5.1.2. The obvious way: monochrome module

To perform the conversion, just activate the *monochrome* module (Section 3.4.3.6, “Monochrome”). You can then simulate a color filter, by dragging the circle above the colours you want to filter. Filter size can be modified thanks to wheel scrolling.

3.5.1.3. The simple way: color correction module

To perform such conversion we use the *color correction* module (Section 3.4.3.5, “Color correction”).

1. Activate the color correction module
2. Use the bottom slider to set saturation to zero

3.5.1.4. The artistic way: color zones module

To perform the conversion we use the *color zone* module (Section 3.4.3.7, “Color zones”).

1. Activate the colour zones module
2. By default, the active tabbed control is set to “saturation” whereas the “select by” combobox is set to “hue”. This means that color are selected according to their hue (horizontal scale) and you can change for each hue its “saturation” (vertical scale). You simply need to set all points to the minimum of the vertical scale to de-saturate every hue.
3. But now if you want, you can keep some hues a little bit saturated, so your image will be all black and white but some hue. A classical use for portrait is to keep red hue saturated in order to make the lips standing out.

You can also use one of the available presets that perform black and white conversion, keeping some hues saturated.

3.5.1.5. The sophisticated way: channel mixer module

To perform the conversion we use the *channel mixer* module (Section 3.4.3.2, “Channel mixer”).

1. Activate the channel mixer module
2. Select the gray output channel
3. Set the proportion of each color, the sum having to equal 1 if you want to keep your global lightness.

3.5.2. Cross-processing

3.5.2.1. Overview

Cross-processing is an analog processing technique where slide film (normally developed thanks to an E6 solution) is processed in chemicals used for processing print film (C41). The resulting images get skewed colors usually a cyan hue and increased contrast and saturation.



The standard way for doing digital cross-processing is to use a channel curve tool but darktable lacks this tool for the moment and another way to accommodate the effect is used.

3.5.2.2. Procedure

This procedure uses tone curve, channel mixer and splittoning modules.

1. Image preparation

Prepare the image for the cross process steps by adjusting the base settings such as exposure, whitebalance etc. for a correctly looking image.

2. Boost contrast

Select the medium contrast curve preset for *tone curve* module (Section 3.4.2.3, "Tone curve") to boost the overall contrast in the image. You might later return here to tune the curve for better result.

3. Color cast

This step changes the color cast as the base for the effect using the *channel mixer* module (Section 3.4.3.2, "Channel mixer"). You might later again return to this and finetune the colorcast of the final result.

- a. Enable the channel mixer module
- b. Select the blue channel and set the blue color value to 0.8
- c. Select the red channel and change the blue color value to 0.1
- d. Select the green channel and change the blue color value to 0.1

4. Splittoning

We use *splittoning* (Section 3.4.5.3, "Splittoning") to add some more coloring to the result for cyan/blue shadows and yellow highlight.

- a. Enable the splittoning module
- b. Select a cyan/blue tone for shadows and set saturation around 50%
- c. Select a yellow/orange tone for highlights and set saturation around 70%
- d. Set compression to 10%
- e. Use the balance slider to tune the splittoning effect. This differs on every image due to its exposure, motive etc.

3.5.3. Cyan toned image

3.5.3.1. Overview

Cyan is a nice color touchup for black and white images, this example guides you through how to make this with darktable and how to control the tone. You can choose any tone of your like for this tutorial...



3.5.3.2. Procedure

This procedure uses monochrome, channel mixer and splittoning modules.

1. Image preparation

Prepare the image for the cyan toned steps by adjusting the base settings such as exposure, black level, contrast etc. for a correctly looking image.

2. Black and white

Enable the *monochrome* module (Section 3.4.3.6, “Monochrome”) to make the image black and white.

3. Add color tone

This step selects the base tone of the image using *channel mixer* (Section 3.4.3.2, “Channel mixer”), we are going for cyan tone but you can choose any tone that you like here.

- a. Enable the channel mixer module
- b. Select the red channel destination and set the red color value to 0.7
- c. Select the green channel destination and set the green color value to 1.15
- d. Select the blue channel destination and set the blue color value to 1.15

As you notice we mix blue and green color to get a cyan tone, we subtract 0.3 from the red channel and add them to blue and green.

4. Splittoning

The result of the previous step does also add a color cast on highlights that we actually want to have white for a prettier result. We also want to add some blue color cast to the shadows in order to emphasize them. Module *splittoning* (Section 3.4.5.3, “Splittoning”) can accomplish this.

- a. Enable the splittoning module
- b. Select a blue/cyan tone for shadows and set saturation around 50%
- c. Set highlights saturation to zero, to remove saturation on highlights
- d. Set compression to zero

- e. Use the balance slider to tune the effect, our example uses a balance of 70/30

3.5.4. Removal of red-eye effect

3.5.4.1. Overview

The red-eye effect in photography is the common appearance of red pupils in color photographs of eyes. It occurs when using a photographic flash very close to the camera lens (as with most cameras with build-in flash), in ambient low light.

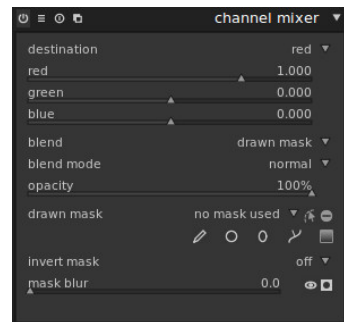
We use the channel mixer module to reduce the red color of the pupil. To limit the processing to the pupils we have to apply two masks.



3.5.4.2. Masking

1. Enable the channel mixer module (see Section 3.4.3.2, "Channel mixer").
2. Activate blending

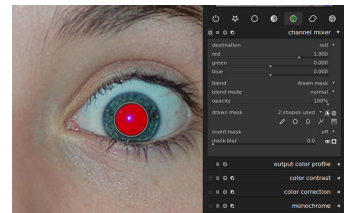
Modules with blending support exhibit an additional combobox "blend" at the bottom of their GUI. Blending is activated with this combobox. Set value to "drawn mask". Additional controls are displayed which allow you to draw a mask.



3. Masking the pupil


Clicking the  symbol adds a circle shape.

Click into the canvas to place the circle. Left-click and drag the circle to the position of the first pupil. Use the scroll-wheel of your mouse while in the circle to change the diameter. Scroll within the circle border to minimize the width of the gradual decay.



Alternatively you can use an elliptical shape. See Section 3.2.7, "Drawn mask" for further details.

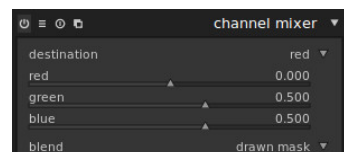
4. Repeat step 3.

Clicking the  symbol adds another circle shape.

Mark the second pupil.

3.5.4.3. Desaturation

1. Approach - Modifying the output channel "red".
 - a. Set output channel "destination" to "red" (default)
 - b. Set red color value to 0.00



c. Set green color value to 0.50

d. Set blue color value to 0.50

You are free to experiment what gives you the most realistic pupil, but this is a good starting point. Another proposal is 0.10/0.60/0.30. The sum of the three values should be 1.



2. Approach - Modifying the output channel "gray".

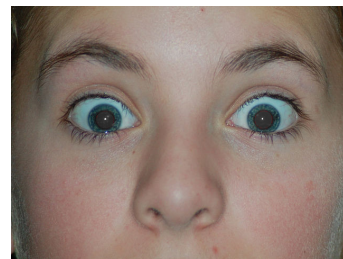
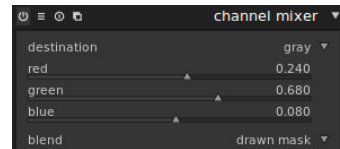
a. Set output channel "destination" to "gray"

b. Set red color value to 0.24

c. Set green color value to 0.68

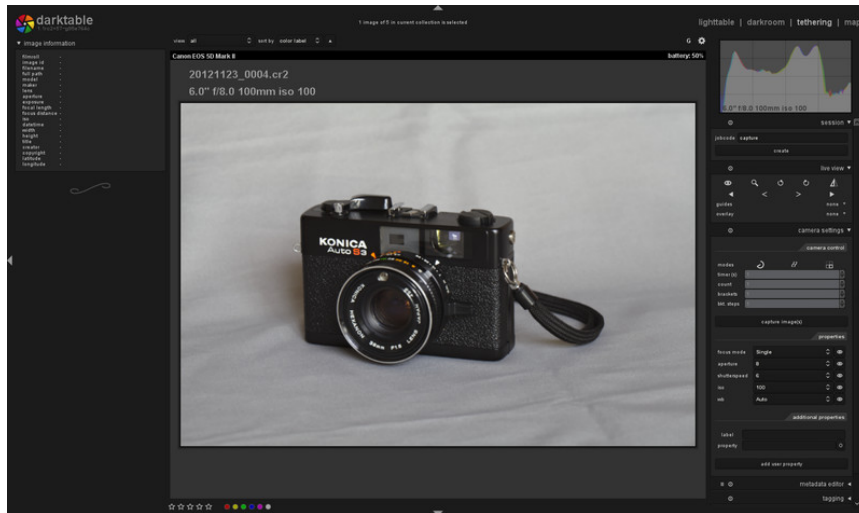
d. Set blue color value to 0.08

You are free to experiment what gives you the most realistic pupil, but this is a good starting point. The sum of the three values should be 1.



Chapter 4. Tethering

The tethering view allows you to capture images directly into darktable from your connected camera.



4.1. Overview

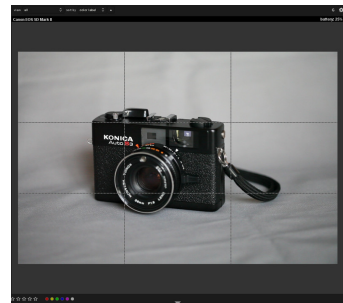
To use the tethering feature you need to connect your camera to your PC using a USB cable. Your computer might ask to mount or view the connected camera. Do not mount or view the camera. If that happens automatically, you will need to “unmount/eject” the camera. This is required to unlock the camera so darktable can lock it for usage.

After the USB cable is connected, look at the import panel in lighttable mode (see Section 2.3.1, “Import”). If your camera is not visible in this panel, click the “scan devices” button and it will appear with two functions: “import from camera” and “tethered shoot”. Click “tethered shoot” to enter the tethering mode.

darktable uses gphoto2 to interface with your camera. If you have problems finding the connected camera as described above, check the troubleshoot section in this chapter in order to verify your camera has tethering support.

4.1.1. Tethering

In the center view images are shown while you capture them. You can get an exposure by either using darktable's userinterface or manually triggering a capture on your camera. If you are using LiveView it will be shown in darktable's center view.



When entering tethering view, a film roll will be created using the same structure as defined when you import from the camera. Job code will be predefined as “capture”.

If you want to group your captures into different film rolls, you should use the session panel in right side. When entering a new name and pressing enter, a new film roll will be created and captured images will go into this new film roll.

darktable provides some nifty tools to setup a capture in the user interface. You can setup timelapse captures and brackets for HDR creations. The configuration is so dynamic that you can create sequential capture of brackets – go figure... For more information read the documentation about the capture panel and the examples in this chapter.

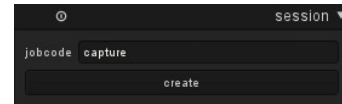
4.2. Tethering panels

This section contains documentation for panels that are specific to the tethering view.

4.2.1. Session

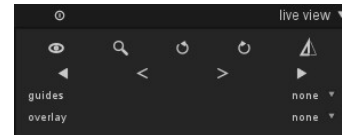
A session is a sequence of exposures taken in tethering mode and going into a single film roll. A new session equals a new film roll. A film roll is created with the same storage structure that is used when you import images from camera into darktable.

It's a bit awkward, but configuring this storage structure is done in the camera import dialog for now.



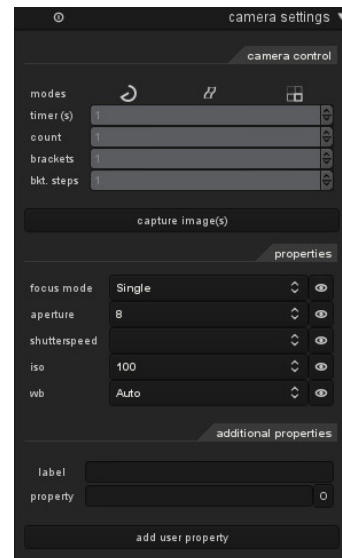
4.2.2. Live view

This panel gives you control of your camera's live view mode. Functionality such as focus setting, rotation, adding guides and overlays are supported.



4.2.3. Camera settings

The camera settings section allows you to set up a capture job. This can include sequence, bracket and delayed captures. You are also able to control other camera settings such as focus mode, aperture, shutter speed, ISO and white balance.



4.3. Examples

This section contains examples of typical usages of tethering.

4.3.1. Studio setup with screening

This is a pretty common use case. You have your studio and subject set up, the camera is connected to your computer and tethering view is active in darktable. You work at the camera and take images. Whenever you want, you can screen the image directly on your computer monitor instead of using the camera LCD for validation.

This workflow is efficient and effective, since you can immediately review your captures instead of waiting until after the shoot when everyone is gone. If you're shooting a model this is a pretty nice way to pre-view the captures with the client instead of fumbling around with your camera.

Working in the tethered mode can save you time and aggravation. Set a session name, shoot your images and they will save in the correct film roll for the session for easy on-site review.

4.3.2. Capturing a timelapse

A timelapse is a video clip composed of images taken in a time sequence. A typical example is to take a timelapse of cityscapes where you capture clouds and traffic etc.

To setup a timelapse capture, create a new session as described previously. Now decide if you want to shoot in manual or auto mode. Only use auto in situations where the ambient light will change significantly during the time of the shoot, eg. shooting a timelapse over 24 hours might give you easier control of light in that kind of captured sequence.

The camera settings panel is where you define delay and sequence. Sequence will give you the opportunity to choose how many images you want to capture and delay will set the time in seconds between captures.

To start the capture click the capture button in the same panel and watch the filmstrip fill up with images. The latest captured image is always displayed in center view.

4.4. Troubleshoot

4.4.1. Verify that your camera is supported

This troubleshooting guide will give you steps to verify your camera can be used with tethering. This is done using the `gphoto2` commandline tools. This is what `darktable` uses to interface with your camera.

1. Verify that camera is detected

The following command will verify a camera that is connected to the computer and detected by `gphoto2`. Find your camera port name to use it in the following tests below. Usually port "usb:" will be enough and therefore used in these examples.

```
env LANG=C gphoto2 --auto-detect
```

2. Verify camera driver abilities

Execute the following command and verify that the *capture choices* ability supports "Image" and *configuration support* is "yes". `darktable` will check these two abilities and decide if "tethered shoot" button should be shown or not.

```
env LANG=C gphoto2 --port usb: --abilities
```

3. Verify camera remote capture

This step will verify that your camera can be remotely controlled; that it can capture an image, download it to your computer and display it within `darktable`.

```
env LANG=C gphoto2 --port usb: --capture-image-and-download
```

4. Verify camera tethered capture

And this last step tests if your camera supports events which `darktable` heavily relies on. Running this command will make the `gphoto2` process wait for an image capture event which you must manually trigger on your camera. If successful, the image will be downloaded to your computer.

```
env LANG=C gphoto2 --port usb: --capture-tethered
```

4.4.2. So, now what?

If any of the steps above failed, there are problems with your specific camera and driver. Please report the issues to `gphoto2` mailing list for further help. You can find the mailing list at www.gphoto.org [<http://www.gphoto.org/maillinglists/>]. Add the following flags to the failed command above for better support and attach the log output to your mail:

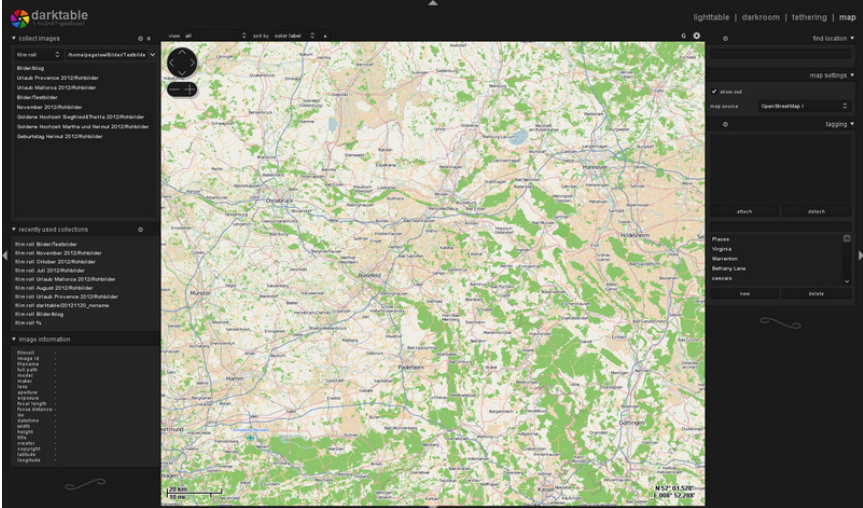
```
--debug --debug-file gphoto2_debug.log
```

If you successfully went through all the tests above, your camera will most likely be supported by `darktable`. Even if successful, if you stumble upon a problem in `darktable`, please file a bug at `redmine` [<http://www.darktable.org/redmine>]. Please attach the log outputs from the steps above and the log file output generated after starting `darktable` with the following command.

```
darktable -d camctl 2>1 >camctl.log
```


Chapter 5. Map

The Map view is where you geotag your images.



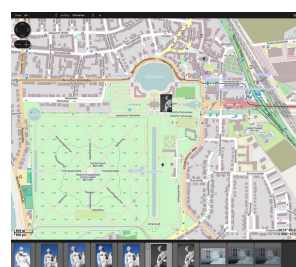
5.1. Overview

Map view will show you a world map with the currently open image, or film roll of images, pinned to their geotagged location. This requires that the image was geotagged by a camera with that feature. Some newer cameras, including smartphones, are already equipped with GPS receivers. Other cameras may need additional GPS hardware to do this.

Even if your camera doesn't support this feature, there is an alternative method. darktable can match the EXIF time and date data in your image(s) to a separate GPX data tracking file created by a GPS tracker recording your movements. These can be handheld devices or a GPS tracker app on your smartphone. This is all done in the lighttable view (see Section 2.3.9, "Geotagging").

5.1.1. Center map view

In the center of the map view you will see a map.



Map data are taken from open map sources on the internet. New map data are only available if you are connected to the internet. darktable keeps a disk cache of previously loaded map data.

Your mouse will allow navigation in the map. Left-click will drag the map; using the scroll-wheel will zoom in or out.

There are on-screen controls and displays that assist you to find your way. A navigation area is located on top left of the map. Use it as an alternative to mouse-dragging and scrolling. The scale of your map is displayed on bottom left. On bottom right you see the geographical coordinates for the center of the map.

Images that already have geo location attributes in their metadata are displayed as small icons on the map.

In order to assign geo coordinates to an image, activate the film-strip on the lower panel (press *Ctrl-f*). You can simply assign a geo location to an image by dragging the image icon from the film-strip and position it onto the map. darktable will record the new geo location (longitude and latitude) as part of the image metadata. Exported images will include this data.

In order to remove geo coordinates from an image just drag it from the map and drop it onto the filmstrip.

Left and right to the central map there are panels for additional control.

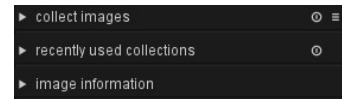
5.2. Map panels

This section contains documentation for panels that are specific to the map view.

5.2.1. Left panels

The panels on the left side we already know from lighttable mode (Section 2.3, “Lighttable panels”).

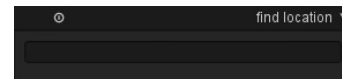
Choose your image selection rules with the *collect images* panel. Recently used collections can be chosen with their respective names in a separate panel. You can also get an overview of the information of the image under your mouse cursor in a panel labeled *image information*.



5.2.2. Find location

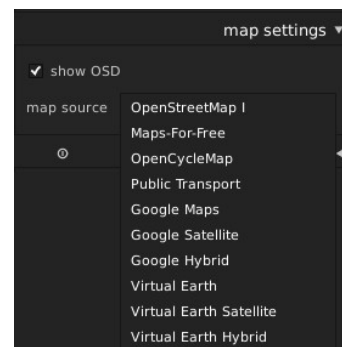
The *find location* module is used to search for a place on map. You need to be connected to the internet to use this feature.

To use, type in a place or address, press enter and a list of results will be shown. Click on one of the resulting items and the map will zoom to that location. Now drag images from the film strip at the bottom of the screen to their location on the map. The GPS location will be embedded in the image.



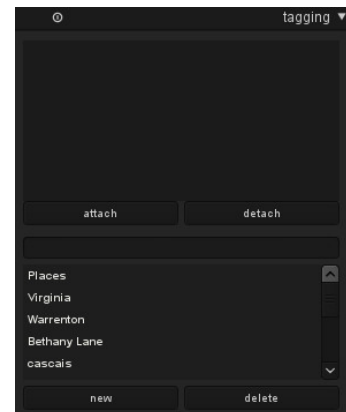
5.2.3. Map settings

In the *map settings* panel you can select your preferred map data from various providers. Some will provide different layers, such as satellite view etc., which you can toggle.



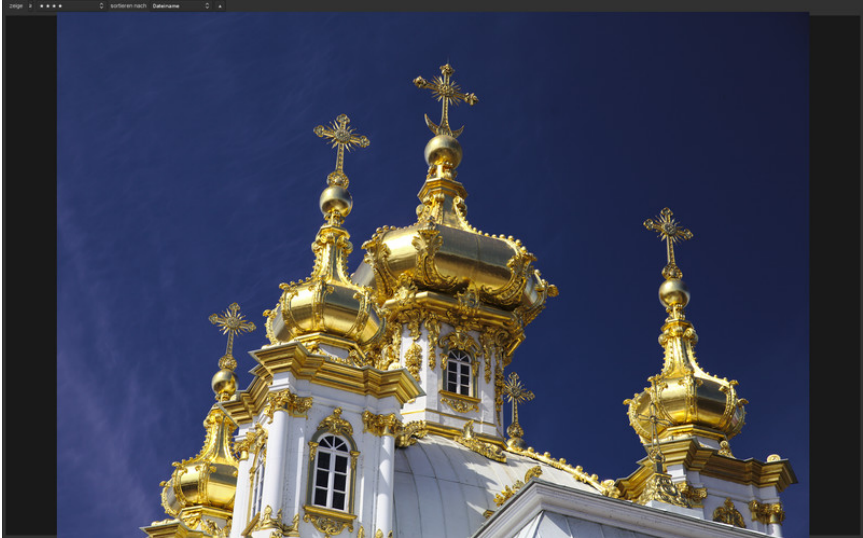
5.2.4. Tagging

The *tagging* panel allows you to attach or detach different tags to an image as well as creating or deleting a tag. It is divided into two parts. The upper part contains tags that are currently attached to the image. The lower part contains all available tags. You must select or mouse over an image for the data to be displayed. See Section 2.3.11, “Tagging” for more details.



Chapter 6. Slideshow

The slideshow view starts a slideshow of your current collection.



6.1. Overview

Entering the slideshow view starts a slideshow of the images of the current collection with filtering rules and sort order applied. To learn more on how to define the collection, the filtering and the sort order see Section 2.3.2, “Collect images” and Section 2.2.5, “Filtering and sort order”.

The image display is optimized to take full advantage of your screen size. You should therefore put darktable into fullscreen mode which is toggled by pressing *F11*. You may press *TAB* in order to hide all remaining panels – namely the filtering option in the top panel.


6.2. Usage

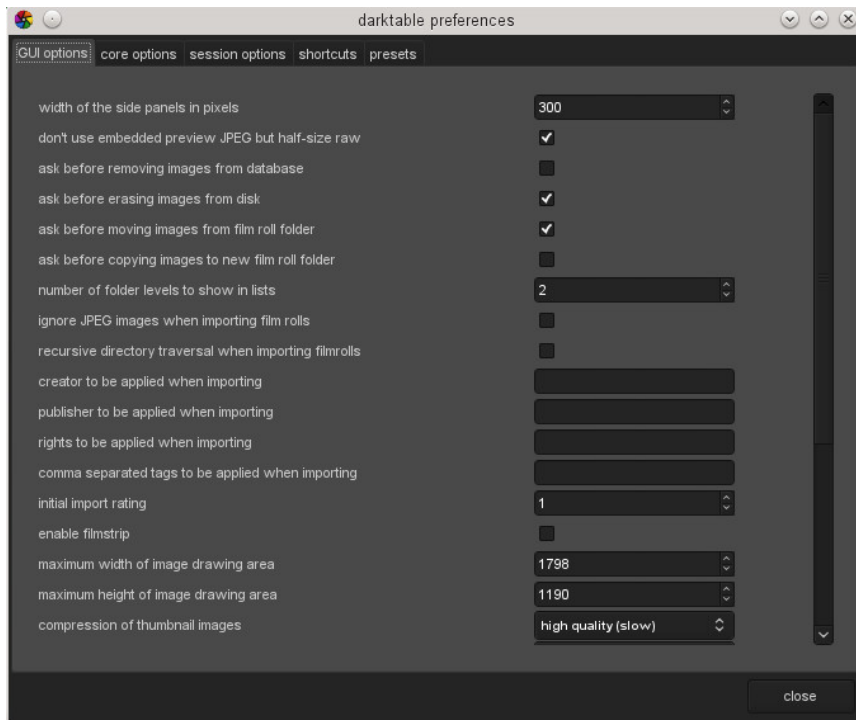
The slideshow view is in an early stage of development with a basic set of features.

<i>left-click</i>	switch to the next image of the collection
<i>right-click</i>	switch to the previous image of the collection
<i>space</i>	start and stop auto-advance mode which automatically switches to the next images every five seconds and after the last image loops back to the beginning
<i>ESC</i>	leave slideshow mode and return to lighttable view

Depending on the complexity of the history stack and the power of your hardware, processing an image with high resolution can take a significant amount of time. darktable prefetches the next image in the background in order to minimize latencies. If you still experience long delays when switching between images or if you intend to quickly advance in your collection consider to disable the option “do high quality processing for slideshow” which gives a much higher processing speed at the expense of a possible slight loss in quality (see Section 7.1, “GUI options”).

Chapter 7. Preferences and settings

darktable comes with a number of settings that can be configured by users. You reach the configuration menu by clicking  at the top of the screen.



7.1. GUI options

These options control the look and feel of darktable.

width of the side panels in pixels

This controls the size of side panels in pixels. Side panels are found left and right to the center view (default 300).

don't use embedded preview jpeg but half-size raw

Check this option to not use the embedded jpeg from the raw file but process the raw data. This is slower but gives you color managed thumbnails (default off).

ask before removing images from database

Always ask the user before any image is removed from the database (default on).

ask before erasing images from disk

Always ask the user before any image file is deleted (default on).

ask before moving images from film roll folder

Always ask the user before any image file is moved (default on).

ask before copying images to new film roll folder

Always ask the user before any image file is copied (default on).

number of folder levels to show in lists

The number of folder levels to show in film roll names, starting from the right (default 1).

ignore jpeg images when importing film rolls

When having RAW+JPEG images together in one directory it makes no sense to import both. With this flag one can ignore all jpegs found (default off).

recursive directory traversal when importing film rolls

Not only import images from the directory selected but recursively go through all subdirectories as well (default off).

creator to be applied when importing

If provided, automatically add this string as a creator tag when importing images (default none).

publisher to be applied when importing

If provided, automatically add this string as a publisher tag when importing images (default none).

rights to be applied when importing

If provided, automatically add this string as a copyrights tag when importing images (default none).

comma separated tags to be applied when importing

If you want to add further tags when importing images, you can give them as a comma separated list (default none).

initial import rating

Initial star rating (from 0 to 5) for all images when importing a film roll (default 1).

enable filmstrip

Enable the filmstrip in darkroom, tethering and geomapping modes (default on).

maximum width of image drawing area

Maximum width of the image drawing area in darkroom mode – adjust to your screen. Needs a restart and will invalidate current thumbnail caches (default 1300).

maximum height of image drawing area

Maximum height of the image drawing area in darkroom mode – adjust to your screen. Needs a restart and will invalidate current thumbnail caches (default 1000).

compression of thumbnail images

Controls the compression of thumbnail images in memory and on disk. Options: “off”, “low quality (fast)”, “high quality (slow)”; (default off).

pen pressure control for brush masks

Controls how the pressure reading of a graphics tablet impacts newly generated brush strokes (see Section 3.2.7, “Drawn mask”). You can control the brush width, its hardness and its opacity. “Absolute” control means that the pressure reading directly defines the attribute with a value between 0% and 100%. “Relative” means that the pressure reading adjusts the attribute between zero and the pre-defined default value (default off).

smoothing of brush strokes

Sets the level for smoothing of brush strokes. Stronger smoothing leads to less nodes and easier editing at the expense of lower accuracy.

ask before deleting a tag

Always ask user before deleting a tag from an image (default on).

only draw images on map that are currently collected and filtered

Use the current filter settings to select the geotagged images drawn in the map view. This limits the images drawn to those currently shown in the filmstrip and thus reduces the time needed (default off).

maximum number of images drawn on map

The maximum number of geotagged images drawn on the map. Increasing this number can slow down the drawing of the map. Needs a restart if changed (default 100).

pretty print the image location

Show a more readable representation of the geo location in the image information module (default on).

expand a single lighttable module at a time

Controls how lighttable panels are expanded. If this option is enabled, expanding a panel by *clicking* collapses any currently expanded panel. If you want to expand a panel without collapsing the others you do so with *shift+click*. Disabling this option reverts the meaning of *click* and *shift+click* (default off).

expand a single darkroom module at a time

Controls how darkroom modules are expanded. If this option is enabled, expanding a module by *clicking* collapses any currently expanded module. If you want to expand a module without collapsing the others you do so with *shift+click*. Disabling this option reverts the meaning of *click* and *shift+click* (default on).

do high quality processing for slideshow

Controls how images are processed for the slideshow view. If this option is enabled, the image will first be processed in full resolution, and downscaled at the very end. This can result in better quality sometimes, but will always be slower (default on).

method to use for getting the display profile

This option allows to force a specific way of getting the current display profile in the output color profile module (see Section 3.4.3.3, "Output color profile"). In the default setting "all" darktable will either query the X display server's xatom or the *colord* system service. You can set this option to "xatom" or "colord" to enforce a specific method if the alternative gives wrong results.

7.2. Core options

These options control some of the internals of darktable.

memory in megabytes to use for mipmap cache

On order to speed-up display of film rolls, darktable stores thumbnails in a cache on disk and loads it into memory at startup. This value controls the cache size in megabytes. It needs a restart if changed (default 512MB).

number of background threads

This controls how many parallel threads are used to create thumbnails during import. On 32bit systems it is strongly recommended to set this to 1. Needs a restart if changed (default 2).

host memory limit (in MB) for tiling

In order to manage large images on systems with limited memory darktable does tile-wise processing. This variable controls the maximum amount of memory (in MB) a module may use during image processing. Lower values will force memory hungry modules to process an image with increasing number of tiles. Setting this to 0 will omit any limits. Values below 500 will be treated as 500. On a 32bit system you should set this to 500. Needs a restart if changed (default 1500).

minimum amount of memory (in MB) for a single buffer in tiling

If set to a positive, non-zero value, this variable defines the minimum amount of memory (in MB) that darktable should take for a single tile. On a 32bit system you should set this to 8. 64bit systems can live with higher values. Needs a restart if changed (default 16).

write sidecar file for each image

These redundant XMP files can later be re-imported into a different database, preserving your changes to the image. It's strongly recommended to have this option activated so you don't lose data in case of a database corruption. Backing up your RAW file plus the accompanying XMP file will allow you to fully restore your work (default on).

store xmp tags in compressed format

Entries in XMP tags can get rather large and may exceed the available space to store the history stack in output files. This option allows binary XMP tags to be compressed in order to save space. Available options are "never", "always", and "only large entries" (default).

activate opengl support

darktable can use your GPU to speed up processing significantly. Interface OpenCL requires suitable hardware and matching OpenCL drivers on your system. If one of those is not found the option is greyed out. Can be switched on and off at any time and takes immediate effect (default on).

always try to use littlecms2

If this option is activated, darktable will use system library littlecms2 instead of its own routines. This is about 28x slower than the default but might give more accurate results in some cases (default off).

do high quality resampling during export

The image will first be processed in full resolution, and downscaled at the very end. This can result in better quality sometimes, but will always be slower (default off).

demosaicing for zoomed out darkroom mode

Interpolation when not viewing 1:1 in darkroom mode: “always bilinear (fast)” is fastest, but not as sharp. “at most ppg (reasonable)” is using ppg + interpolation modes specified below, “full (possibly slow)” will use exactly the settings for full-size export (default “at most ppg (reasonable)”).

pixel interpolator

Pixel interpolator used in rotation, lens correction, up- and downscaling; options are “bilinear”, “bicubic”, “lanczos2”, “lanczos3” (default).

password storage backend to use

The storage backend for password storage. Options: “none”, “kwallet”, “gnome keyring” (default none).

look for updated xmp files on startup

Check file modification times of all XMP files on startup to find out if any got updated in the meantime by some other software. If updated XMP files are found a menu opens for the user to decide which of the XMP files to be reloaded – replacing darktable’s database entries by the XMP file contents – and which of the XMP files to be overwritten by darktable’s database (default off).

executable for playing audio files

Defines an external program which is used in the lighttable view to play audio files that some cameras record to keep notes for images (default “aplay”).

7.3. Session options

These options define a naming pattern to organize images on disk when importing from a connected camera (see Section 2.3.1, “Import”) and when taking photos in the tethering view (see Chapter 4, *Tethering*).

The naming pattern consists of three parts: a base part defining the parent folder, a session part defining a sub directory which is specific to the individual import session, and a file name part defining the filename structure for each imported image.

Several pre-defined variables can be used in the pattern as placeholders:

<code>\$(HOME)</code>	the home folder as defined by the system
<code>\$(PICTURES_FOLDER)</code>	the pictures folder as defined by the system (usually “ <code>\$(HOME)/Pictures</code> ”)
<code>\$(DESKTOP)</code>	the desktop folder as defined by the system (usually “ <code>\$(HOME)/Desktop</code> ”)
<code>\$(USERNAME)</code>	your user account name on the system
<code>\$(FILE_NAME)</code>	basename of the imported image
<code>\$(FILE_EXTENSION)</code>	extension of the imported image
<code>\$(JOBCODE)</code>	unique identifier of the import job
<code>\$(SEQUENCE)</code>	a sequence number within the import job
<code>\$(ID)</code>	unique identification number of the image in darktable's database
<code>\$(YEAR)</code>	year at the date of import
<code>\$(MONTH)</code>	month at the date of import
<code>\$(DAY)</code>	day at the date of import
<code>\$(HOUR)</code>	hour at the time of import
<code>\$(MINUTE)</code>	minute at the time of import
<code>\$(SECOND)</code>	second at the time of import
<code>\$(EXIF_YEAR)</code>	year the photo was taken (from EXIF data)
<code>\$(EXIF_MONTH)</code>	month the photo was taken (from EXIF data)
<code>\$(EXIF_DAY)</code>	day the photo was taken (from EXIF data)
<code>\$(EXIF_HOUR)</code>	hour the photo was taken (from EXIF data)
<code>\$(EXIF_MINUTE)</code>	minute the photo was taken (from EXIF data)
<code>\$(EXIF_SECOND)</code>	seconds the photo was taken (from EXIF data)
<code>\$(EXIF_ISO)</code>	ISO value of the photo (from EXIF data)

base directory naming pattern

The base directory part of the naming pattern (default “`$(PICTURES_FOLDER)/Darktable`”).

sub directory naming pattern

The sub directory part of the naming pattern (default “`$(YEAR)$(MONTH)$(DAY)_$(JOBCODE)`”).

file naming pattern

The file name part of the naming pattern (default “\$(YEAR)\$ (MONTH)\$ (DAY)_\$(SEQUENCE).\$(FILE_EXTENSION)”).

7.4. Shortcuts

darktable has a large set of keyboard shortcuts that, with the work of Robert Bieber, is now user configurable through the preference pane.

When you open the shortcuts menu you are presented with a hierarchical list of all actions that can receive a keyboard shortcut. Go to the action you want to change and double click. You are then prompted to press the new key combination to be mapped to the selected action. In order to remove an existing keyboard shortcut, click on the action and press backspace.

You can export your mappings to a file or import mappings from a file. Press “default” to reset all keyaccelerators to their default state.

Below is a table with the default keybindings for actions available in darktable.

<darktable>/global/lighttable view	l
<darktable>/global/darkroom view	d
<darktable>/global/capture view	t
<darktable>/global/map view	m
<darktable>/global/increase brightness	F10
<darktable>/global/decrease brightness	F9
<darktable>/global/increase contrast	F8
<darktable>/global/decrease contrast	F7
<darktable>/global/leave fullscreen	Escape
<darktable>/global/quit	<Primary>q
<darktable>/global/switch view	period
<darktable>/global/toggle fullscreen	F11
<darktable>/global/toggle header	<Primary>h
<darktable>/global/toggle side borders	Tab
<darktable>/views/lighttable/color red	F1
<darktable>/views/lighttable/color yellow	F2
<darktable>/views/lighttable/color green	F3
<darktable>/views/lighttable/color blue	F4
<darktable>/views/lighttable/color purple	F5
<darktable>/views/lighttable/navigate down	<Shift>g
<darktable>/views/lighttable/navigate page down	Page_Down
<darktable>/views/lighttable/navigate page up	Page_Up
<darktable>/views/lighttable/navigate up	g
<darktable>/views/lighttable/preview	z
<darktable>/views/lighttable/preview with focus detection	<Primary>z
<darktable>/views/lighttable/rate 1	1
<darktable>/views/lighttable/rate 2	2
<darktable>/views/lighttable/rate 3	3
<darktable>/views/lighttable/rate 4	4
<darktable>/views/lighttable/rate 5	5

<darktable>/views/lighttable/rate desert	0
<darktable>/views/lighttable/rate reject	r
<darktable>/views/lighttable/realign images to grid	l
<darktable>/views/lighttable/scroll center	apostrophe
<darktable>/views/lighttable/scroll down	Down
<darktable>/views/lighttable/scroll left	Left
<darktable>/views/lighttable/scroll right	Right
<darktable>/views/lighttable/scroll up	Up
<darktable>/views/darkroom/export	<Primary>e
<darktable>/views/darkroom/image forward	space
<darktable>/views/darkroom/image back	BackSpace
<darktable>/views/darkroom/overexposed	o
<darktable>/views/darkroom/toggle film strip	<Primary>f
<darktable>/views/darkroom/zoom close-up	<Alt>1
<darktable>/views/darkroom/zoom fill	<Alt>2
<darktable>/views/darkroom/zoom fit	<Alt>3
<darktable>/views/map/redo	<Primary>r
<darktable>/views/map/undo	<Primary>z
<darktable>/views/capture/toggle film strip	<Primary>f
<darktable>/modules/copy_history/copy all	<Primary>c
<darktable>/modules/copy_history/copy	<Primary><Shift>c
<darktable>/modules/copy_history/paste all	<Primary>v
<darktable>/modules/copy_history/paste	<Primary><Shift>v
<darktable>/modules/export/export	<Primary>e
<darktable>/modules/filmstrip/color red	F1
<darktable>/modules/filmstrip/color yellow	F2
<darktable>/modules/filmstrip/color green	F3
<darktable>/modules/filmstrip/color blue	F4
<darktable>/modules/filmstrip/color purple	F5
<darktable>/modules/filmstrip/duplicate image	<Primary>d
<darktable>/modules/filmstrip/copy history parts	<Primary><Shift>c
<darktable>/modules/filmstrip/copy history	<Primary>c
<darktable>/modules/filmstrip/paste history parts	<Primary><Shift>v
<darktable>/modules/filmstrip/paste history	<Primary>v
<darktable>/modules/filmstrip/rate 1	1
<darktable>/modules/filmstrip/rate 2	2
<darktable>/modules/filmstrip/rate 3	3
<darktable>/modules/filmstrip/rate 4	4
<darktable>/modules/filmstrip/rate 5	5
<darktable>/modules/filmstrip/rate desert	0
<darktable>/modules/filmstrip/rate reject	r

<darktable>/modules/image/group	<Primary>g
<darktable>/modules/image/ungroup	<Primary><Shift>g
<darktable>/modules/image/remove from collection	Delete
<darktable>/modules/lighttable_mode/zoom max	<Alt>1
<darktable>/modules/lighttable_mode/zoom in	<Alt>2
<darktable>/modules/lighttable_mode/zoom out	<Alt>3
<darktable>/modules/lighttable_mode/zoom min	<Alt>4
<darktable>/modules/live_view/toggle live view	v
<darktable>/modules/select/select all	<Primary>a
<darktable>/modules/select/invert selection	<Primary>i
<darktable>/modules/select/select none	<Primary><Shift>a
<darktable>/modules/tagging/tag	<Primary>t
<darktable>/image operations/clipping/commit	Return
<darktable>/image operations/colorout/toggle gamutcheck	<Primary>g
<darktable>/image operations/colorout/toggle softproofing	<Primary>s
<darktable>/image operations/flip/rotate 90 degrees ccw	bracketleft
<darktable>/image operations/flip/rotate 90 degrees cw	bracketright

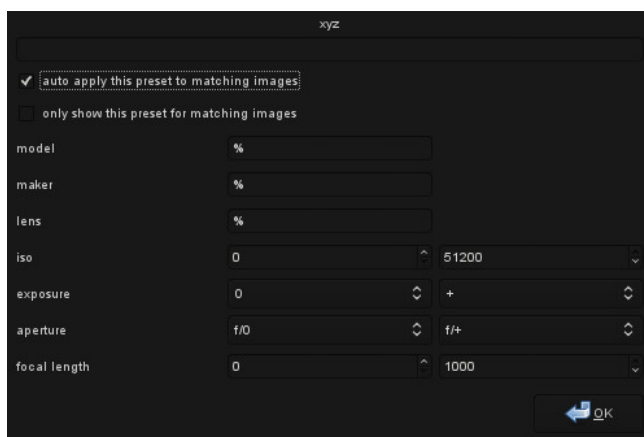
7.5. Presets

This menu gives you an overview of the presets that are defined for darktable's modules. In this dialog you can select whether a certain user defined preset shall be auto-applied to matching images.

darktable already comes with a set of pre-defined presets for several modules. In addition you can define your own presets from within each module in darkroom mode (see Section 3.2.3, "Module presets").

Pre-defined presets are shown with a lock symbol. Their auto-apply properties can not be changed.

Double clicking on a user-defined preset will open a menu.



auto apply this preset to matching images

activate this checkbox to automatically apply the preset on newly imported images; a set of fields is displayed where you can define patterns to be matched against EXIF data.

only show this preset for matching images

activate this checkbox to hide the preset in darkroom mode if it does not match the defined patterns.

model

a pattern to be matched against the EXIF field that describes your camera model; use "%" as wildcard.

maker

a pattern to be matched against the EXIF field that describes the maker of your camera; use "%" as wildcard.

lens

a pattern to be matched against the EXIF field that describes your lens; use "%" as wildcard.

iso

only apply the preset if the ISO value of your image lies within the given range.

exposure

only apply the preset if the exposure time of your image lies within the given range; set "+" as the upper value to match arbitrarily long exposures.

aperture

only apply the preset if the aperture of your image lies within the given range; set "f/0" as the lower value to match arbitrarily open apertures; set "f/+" as the upper value to match arbitrarily closed apertures.

focal length

only apply the preset if the focal length of your image lies within the given range (from 0 to 1000).

Chapter 8. Scripting with Lua

darktable comes with a versatile scripting interface for functionality enhancement.

```
dt = require "darktable"
dt.register_storage("module_stitcher", "mosaic generator", nil,
function(storage, image_table)
    dt.print_error("Will try to stitch now")
    command = "gm convert "
    for _,v in pairs(image_table) do
        command = command..v.." "
    end
    command = command..".append -resize 15% "..dt.configuration.tmp_dir.."/tmp.png"
    dt.print_error("this is the command: "..command)
    os.execute(command)

    dt.print("Stitching saved to "..dt.configuration.tmp_dir.."/tmp.png")
end
```

8.1. Lua usage

Lua can be used to define actions which darktable will perform whenever a specified event is triggered. One example might be calling an external application during file export in order to apply additional processing steps outside of darktable.

darktable uses *Lua* [<http://www.lua.org/>], which is an independent project founded in 1993, providing a powerful, fast, lightweight, embeddable scripting language. Lua is widely used by many open source applications, in commercial programs, and for games programming.

darktable uses Lua version 5.2. Describing the principles and syntax of Lua is beyond the scope of this usermanual. For a detailed introduction see the Lua reference manual [<http://www.lua.org/manual/5.2/manual.html>].

8.1.1. Basic principles

At startup, darktable will automatically run two Lua scripts:

- a script called *luarc* in *\$DARKTABLE/share/darktable*
- a script called *luarc* in the user's configuration directory

\$DARKTABLE is used here to represent your system's darktable installation directory.

This is the only time darktable will run Lua scripts by itself. The script can register callbacks to perform actions on various darktable events. This callback mechanism is the primary way to trigger lua actions.

8.1.2. A simple lua example

Let's start with a simple example. We will print some code on the console. Create a file called *luarc* in darktable's configuration directory (usually *~/.config/darktable/*) and add the following line to it:

```
print("Hello World !")
```

Start darktable and you will see the sentence *Hello World!* printed on the console. Nothing fancy but it's a start...

At this point, there is nothing specific to darktable in the script. We simply use the standard `print` function to print a string. That's nice and all, but we can do better than that. To access the darktable API you first need to `require` it and save the returned object in a variable. Once this is done you can access the darktable API as subfields of the returned object. All of this is documented in Section 8.2.1, "darktable".

```
local darktable = require "darktable"  
darktable.print_error("Hello World !")
```

Run the script ... and nothing happens. The function `darktable.print_error` is just like `print` but will only print the message if you have enabled lua traces with **-d lua** on the command line. This is the recommended way to do traces in a darktable lua script.

8.1.3. Printing labeled images

This first example showed us the very basics of lua and allowed us to check that everything is working properly. Let's do something a little bit more complex. Let's try to print the list of images that have a red label attached to them. But first of all, what is an image?

```

local darktable = require "darktable"
local debug = require "darktable.debug"
print(darktable.debug.dump(darktable.database[1]))

```

Running the code above will produce a lot of output. We will look at it in a moment, but first let's look at the code itself.

We know about requiring `darktable`. Here, we need to separately require `darktable.debug` which is an optional section of the API that provides helper functions to help debug lua scripts.

`darktable.database` is a table provided by the API that contains all images in the database (currently visible or not, duplicate or not...). Each entry in the database is an image object. Image objects are complex objects that allow you to manipulate your image in various ways (it is all documented in Section 8.2.2.1, "types.dt_lua_image_t"). To display our images, we use `darktable.debug.dump` which is a function that will take anything as its parameter and recursively dump its content. Since images are complex objects that indirectly reference other complex objects, the resulting output is huge. Below is a cut down example of the output.

```

toplevel (userdata,dt_lua_image_t) : /images/100.JPG
  publisher (string) : ""
  path (string) : "/images"
  move (function)
  exif_aperture (number) : 2.7999999523163
  rights (string) : ""
  make_group_leader (function)
  exif_crop (number) : 0
  duplicate_index (number) : 0
  is_raw (boolean) : false
  exif_iso (number) : 200
  is_ldr (boolean) : true
  rating (number) : 1
  description (string) : ""
  red (boolean) : false
  get_tags (function)
  duplicate (function)
  creator (string) : ""
  latitude (nil)
  blue (boolean) : false
  exif_datetime_taken (string) : "2014:04:27 14:10:27"
  exif_maker (string) : "Panasonic"
  drop_cache (function)
  title (string) : ""
  reset (function)
  create_style (function)
  apply_style (function)
  film (userdata,dt_lua_film_t) : /images
    1 (userdata,dt_lua_image_t): .toplevel
    [.....]
  exif_exposure (number) : 0.0062500000931323
  exif_lens (string) : ""

```

```

detach_tag (function): toplevel.film.2.detach_tag
exif_focal_length (number) : 4.5
get_group_members (function): toplevel.film.2.get_group_members
id (number) : 1
group_with (function): toplevel.film.2.group_with
delete (function): toplevel.film.2.delete
purple (boolean) : false
is_hdr (boolean) : false
exif_model (string) : "DMC-FZ200"
green (boolean) : false
yellow (boolean) : false
longitude (nil)
filename (string) : "100.JPG"
width (number) : 945
attach_tag (function): toplevel.film.2.attach_tag
exif_focus_distance (number) : 0
height (number) : 648
local_copy (boolean) : false
copy (function): toplevel.film.2.copy
group_leader (userdata,dt_lua_image_t): .toplevel

```

As we can see, an image has a large number of fields which provide all sort of information about it. We are interested in the red label (documented in the section called "types.dt_lua_image_t.red"). This field is a boolean, and the documentation tells us that it can be written. We now just need to find all images with that field and print them out.

```

darktable = require "darktable"
for _,v in ipairs(darktable.database) do
  if v.red then
    print(tostring(v))
  end
end
end

```

This code should be quite simple to understand at this point, but it contains a few interesting aspects about lua that are worth highlighting:

- `ipairs` is a standard lua function that will iterate through all numeric indices of a table. We use it here because `darktable.database` has non-numeric indices which are functions to manipulate the database itself (adding or deleting images, for example).
- Iterating through a table will return both the key and the value used. It is conventional in lua to use a variable named `"_"` to store values that we don't care about.
- Note that we use the standard lua function `tostring` here and not the darktable specific `darktable.debug.dump`. The standard function will return a name for the object whereas the debug function will print the content. The debug function would be too verbose here. Once again, it is a great debug tool but should not be used for anything else.

8.1.4. Adding a simple shortcut

So far, all our scripts have done things during startup. This is of limited use and doesn't allow us to react to real user actions. To do more advanced things we need to register

a function that will be called on a given event. The most common event to react to is a keyboard shortcut.

```
darktable = require "darktable"

local function hello_shortcut(event, shortcut)
darktable.print("Hello, I just received '"..event..
               "' with parameter '"..shortcut.."'")
end

darktable.register_event("shortcut",hello_shortcut,
                        "A shortcut that print its parameters")
```

Now start darktable, go to preferences => shortcut => lua => A shortcut that print its parameters assign a shortcut and try it. You should have a nice message printed on the screen.

Let's look at the code in details. We first define a function with two parameters. These parameters are strings (this is documented in the section called "events.shortcut.callback"). The first one is the type of event that is triggered ("shortcut") and the second one is what shortcut specifically ("A shortcut that print its parameters"). The function itself calls `darktable.print` that will print the message as an overlay in the main window.

Once that function is defined, we register it as a shortcut callback. To do that we call `darktable.register_event` which is a generic function for all types of events. We tell it that we are registering a shortcut event, then we give the callback to call and last, we give the string to use to describe the shortcut in the preference window.

Let's try a shortcut a little more interactive. This one will look at the images the user is currently interested in (selected our moused-over) and will increase their rating.

```
darktable = require "darktable"

darktable.register_event("shortcut",function(event,shortcut)
    local images = darktable.gui.action_images
    for _,v in pairs(images) do
        v.rating = v.rating + 1
    end
end,"Increase the rating of an image")
```

At this point, most of this code should be self explanatory. Just a couple of notes:

- Instead of declaring a function and referencing it, we declare it directly in the call to `darktable.register_event` this is strictly equivalent but slightly more compact.
- `image.rating` is a field of any image that gives its rating (between 0 and 5 stars, -1 means rejected).
- `darktable.gui.action_images` is a table containing all the images of interest. Darktable will act on selected images if any image is selected, and on the image under the mouse if no image is selected. This function allows to easily follow darktable's UI logic in lua.

If you select an image and press your shortcut a couple of time, it will work correctly at first but when you have reached five stars, darktable will start showing the following error on the console:

```
LUA ERROR : rating too high : 6
stack traceback:
 [C]: in ?
 [C]: in function '__newindex'
 ./configdir/luarc:10: in function <./configdir/luarc:7>
      LUA ERROR : rating too high : 6
```

This is lua's way of reporting errors. We have attempted to set a rating of 6 to an image, but a rating can only go as high as 5. It would be trivial to add a check, but let's go the complicated way and catch the error instead.

```
darktable.register_event("shortcut",function(event,shortcut)
    local images = darktable.gui.action_images
    for _,v in pairs(images) do
        result,message = pcall(function()
            v.rating = v.rating + 1
        end)
        if not result then
            darktable.print_error("could not increase rating of image "..
                tostring(v).. " : "..message)
        end
    end
end,"Increase the rating of an image")
```

`pcall` will run its first argument and catch any exception thrown by it. If there is no exception it will return `true` plus any result returned by the function; if there is an exception it will return `false` and the error message of the exception. We simply test these results and print them on the console...

8.1.5. Exporting images with Lua

We have learned to use lua to adapt darktable to our particular workflow, let's look at how to use lua to easily export images. Darktable can easily export images to quite a few online services but there are always more. If you are able to upload an image to a service via the command line then you can use lua to integrate it into darktable's user interface.

In this next example we will use lua to export via **scp**. A new storage will appear in darktable's UI which will export images to a remote target via ssh's copy mechanism.

```
darktable = require "darktable"
```



```

darktable.preferences.register("scp_export", "export_path",
    "string", "target SCP path",
    "Complete path to copy to. Can include user and hostname", "")

darktable.register_storage("scp_export", "Export via scp",
    function( storage, image, format, filename,
        number, total, high_quality, extra_data)
        if coroutine.yield("RUN_COMMAND", "scp ../filename.." ..
            darktable.preferences.read("scp_export",
                "export_path", "string")) then
            darktable.print_error("scp failed for " .. tostring(image))
        end
    end)
end)

```

`darktable.preferences.register` will add a new preference to darktable's preference menu. `scp_export` and `export_path` allow us to uniquely identify our preference. These fields are reused when we read the value of the preference. The `string` field tells the lua engine that the preference is a string. It could also be an integer, a filename or any of the types detailed in Section 8.2.2.34, "types.lua_pref_type". We then have the label for the preference in the preference menu, the tooltip when hovering over the value and a default value.

`darktable.register_storage` is the call that actually registers a new storage. The first argument is a name for the storage, the second is the label that will be displayed in the UI and last is a function to call on each image. This function has a lot of parameters, but `filename` is the only one we use in this example. It contains the name of a temporary file where the image was exported by darktable's engine.

This code will work but it has a couple of limitations. This is just a simple example after all:

- We use preferences to configure the target path. It would be nicer to add an element to the export UI in darktable but the lua interface doesn't support this yet. Stay tuned...
- We check the returned value of `scp`. That command might fail, in particular if the user has not correctly set the preference.
- This script can't read input from the user. The remote `scp` must use password-less copy. Reading input from the user is doable but beyond the scope of this example.
- There is no message displayed once the example is done, only the progressbar on the lower left side tells the user that the job is done.
- We use `coroutine.yield` to call an external program. The normal `os.execute` would block other lua codes from happening.

8.1.6. Sharing scripts

So far, our lua code was in *luarc*. That's a good way to develop your script but not very practical for distribution. We need to make this into a proper lua module. To do that, we save the code in a separate file (**scp-storage.lua** in our case):

```

--[ [
SCP STORAGE
a simple storage to export images via scp

AUTHOR

```

Jérémy Rosen (jeremy.rosen@enst-bretagne.fr)

INSTALLATION

```
* copy this file in $CONFIGDIR/luarc/ where CONFIGDIR
is your darktable configuration directory
* add the following line in the file $CONFIGDIR/luarc
require "scp-storage"
```

USAGE

```
* set the target directory for scp in preferences =>lua
=> target SCP path
* select "Export via SCP" in the storage selection menu
* export your images
```

LICENSE

GPLv2

```
]]
darktable = require "darktable"

darktable.configuration.check_version(...,{3,0,0})

darktable.preferences.register("scp_export","export_path",
"string","target SCP path",
"Complete path to copy to. Can include user and hostname","")

darktable.register_storage("scp_export","Export via scp",
function( storage, image, format, filename,
number, total, high_quality, extra_data)
if coroutine.yield("RUN_COMMAND","scp "..filename.." "..
darktable.preferences.read("scp_export",
"export_path","string")) then
darktable.print_error("scp failed for "..tostring(image))
end
end)
end)
```

Darktable will look for scripts (following the normal lua rules) in the standard directories plus \$CONFIGDIR/luarc/?.lua . So our script can be called by simply adding `require "scp-storage"` in the *luarc* file. A couple of extra notes...

- The function `darktable.configuration.check_version` will check compatibility for you. the `...` will turn into your script's name and `{2,0,0}` is the API version you have tested your script with. You can add multiple API versions if you update your script for multiple versions of darktable.
- Make sure to declare all your functions as `local` to not pollute the general namespace.
- Make sure you do not leave debug prints in your code. `darktable.print_error` in particular allows you to leave debug prints in your final code without disturbing the console.
- You are free to choose any license for your script but scripts that are uploaded on darktable's website need to be GPLv2.

Once you have filled all the fields, checked your code, you can upload it to our script page here [<http://darktable.org/redmine/projects/darktable/wiki/LuaScripts>].

8.1.7. Calling Lua from DBus

It is possible to send a lua command to darktable via its DBus interface. The method *org.darktable.service.Remote.Lua* takes a single string parameter which is interpreted as a lua command. The command will be executed in the current lua context and should return either *nil* or a string. The result will be passed back as the result of the DBus method.

If the Lua call results in an error, the DBus method call will return an error *org.darktable.Error.LuaError* with the lua error message as the message attached to the DBus error.

8.1.8. Using Darktable from a lua script

Warning: This feature is very experimental. It is known that several elements don't work in library mode yet. Careful testing is highly recommended.

The lua interface allows you to use darktable from any lua script. This will load darktable as a library and provide you with most of the lua API (darktable is configured headless, so the functions related to the user interface are not available).

As an example, the following program will print the list of all images in your library:

```
#!/usr/bin/env lua
package = require "package"
package.cpath=package.cpath..";/lib/darktable/lib?.so"

dt = require("darktable")(
  "--library", "./library.db",
  "--datadir", "./share/darktable",
  "--moduledir", "./lib/darktable",
  "--configdir", "./configdir",
  "--cachedir", "cachedir",
  "--g-fatal-warnings")

require("darktable.debug")

for k,v in ipairs(dt.database) do
  print(tostring(v))
end
```

Note the third line that points to the location of the `libdarktable.so` file.

Also note that the call to `require` returns a function that can be called only once and allows you to set darktable's command line parameter. The `:memory:` parameter to `--library` is usefull here if you don't want to work on your personal library.

8.2. Lua API

To access the darktable specific functions you must load the darktable environment:

```
darktable = require "darktable"
```

All functions and data are accessed through the darktable module.

This documentation for API version 2.0.0-dev.

8.2.1. darktable

The darktable library is the main entry point for all access to the darktable internals.

8.2.1.1. darktable.print

```
function(  
  message : string  
)
```

Will print a string to the darktable control log (the long overlaid window that appears over the main panel).

message

```
string
```

The string to display which should be a single line.

8.2.1.2. darktable.print_error

```
function(  
  message : string  
)
```

This function will print its parameter if the Lua logdomain is activated. Start darktable with the "-d lua" command line option to enable the Lua logdomain.

message

```
string
```

The string to display.

8.2.1.3. darktable.register_event

```
function(  
  event_type : string,  
  callback : function,  
  ... : variable  
)
```

This function registers a callback to be called when a given event happens.

Events are documented in the event section.

event_type

```
string
```

The name of the event to register to.

callback

function

The function to call on event. The signature of the function depends on the type of event.

...

variable

Some events need extra parameters at registration time; these must be specified here.

8.2.1.4. `darktable.register_storage`

```
function(  
  plugin_name : string,  
  name : string,  
  [store : function],  
  [finalize : function],  
  [supported : function],  
  [initialize : function]  
)
```

This function will add a new storage implemented in Lua.

A storage is a module that is responsible for handling images once they have been generated during export. Examples of core storages include filesystem, e-mail, facebook...

`plugin_name`

string

A Unique name for the plugin.

`name`

string

A human readable name for the plugin.

`store`

```
function(  
  storage : types.dt_imageio_module_storage_t,  
  image : types.dt_lua_image_t,  
  format : types.dt_imageio_module_format_t,  
  filename : string,  
  number : integer,  
  total : integer,  
  high_quality : boolean,  
  extra_data : table  
)
```

This function is called once for each exported image. Images can be exported in parallel but the calls to this function will be serialized.

storage

`types.dt_imageio_module_storage_t`

The storage object used for the export.

image

`types.dt_lua_image_t`

The exported image object.

format

`types.dt_imageio_module_format_t`

The format object used for the export.

filename

`string`

The name of a temporary file where the processed image is stored.

number

`integer`

The number of the image out of the export series.

total

`integer`

The total number of images in the export series.

high_quality

`boolean`

True if the export is high quality.

extra_data

`table`

An empty Lua table to take extra data. This table is common to the `initialize`, `store` and `finalize` calls in an export serie.

finalize

```
function(  
  storage : types.dt_imageio_module_storage_t,  
  image_table : table,  
  extra_data : table  
)
```

This function is called once all images are processed and all store calls are finished.

storage

`types.dt_imageio_module_storage_t`

The storage object used for the export.

image_table

table

A table keyed by the exported image objects and valued with the corresponding temporary export filename.

extra_data

table

An empty Lua table to store extra data. This table is common to all calls to store and the call to finalize in a given export series.

supported

```
function(  
  storage : types.dt_imageio_module_storage_t,  
  format  : types.dt_imageio_module_format_t  
) : boolean
```

A function called to check if a given image format is supported by the Lua storage; this is used to build the dropdown format list for the GUI.

Note that the parameters in the format are the ones currently set in the GUI; the user might change them before export.

storage

`types.dt_imageio_module_storage_t`

The storage object tested.

format

`types.dt_imageio_module_format_t`

The format object to report about.

return

boolean

True if the corresponding format is supported.

initialize

```
function(  
  storage : types.dt_imageio_module_storage_t,  
  format  : types.dt_imageio_module_format_t,  
  images  : table of types.dt_lua_image_t,  
  high_quality : boolean,  
  extra_data : table
```

) : table or nil

A function called before storage happens

This function can change the list of exported functions

storage

`types.dt_imageio_module_storage_t`

The storage object tested.

format

`types.dt_imageio_module_format_t`

The format object to report about.

images

table of `types.dt_lua_image_t`

A table containing images to be exported.

high_quality

boolean

True if the export is high quality.

extra_data

table

An empty Lua table to take extra data. This table is common to the initialize, store and finalize calls in an export serie.

return

table or nil

The modified table of images to export or nil

If nil (or nothing) is returned, the original list of images will be exported

If a table of images is returned, that table will be used instead. The table can be empty. The images parameter can be modified and returned

8.2.1.5. darktable.films

A table containing all the film objects in the database.

darktable.films.#

`types.dt_lua_film_t`

Each film has a numeric entry in the database.

darktable.films.new


```
function(  
    directory : string  
) : types.dt_lua_film_t
```

Creates a new empty film

see `darktable.database.import` to import a directory with all its images and to add images to a film

directory

string

The directory that the new film will represent. The directory must exist

return

types.dt_lua_film_t

The newly created film, or the existing film if the directory is already imported

darktable.films.delete

see `types.dt_lua_film_t.delete`

8.2.1.6. darktable.new_format

```
function(  
    type : string  
) : types.dt_imageio_module_format_t
```

Creates a new format object to export images

type

string

The type of format object to create, one of :

- copy
- exr
- j2k
- jpeg
- pfm
- png
- ppm
- tiff
- webp

return

types.dt_imageio_module_format_t

The newly created object. Exact type depends on the type passed

8.2.1.7. **darktable.new_storage**

```
function(  
  type : string  
) : types.dt_imageio_module_storage_t
```

Creates a new storage object to export images

type

string

The type of storage object to create, one of :

- disk
 - email
 - facebook
 - flickr
 - gallery
 - latex
 - picasa
- (Other, lua-defined, storage types may appear.)

return

```
types.dt_imageio_module_storage_t
```

The newly created object. Exact type depends on the type passed

8.2.1.8. **darktable.gui**

This subtable contains function and data to manipulate the darktable user interface with Lua.

Most of these function won't do anything if the GUI is not enabled (i.e you are using the command line version `darktbl-cli` instead of `darktable`).

darktable.gui.action_images

table

A table of `types.dt_lua_image_t` on which the user expects UI actions to happen.

It is based on both the hovered image and the selection and is consistent with the way darktable works.

It is recommended to use this table to implement Lua actions rather than `darktable.gui.hovered` or `darktable.gui.selection` to be consistent with darktable's GUI.

darktable.gui.hovered

The image under the cursor or nil if no image is hovered.

darktable.gui.selection

```
function(  
  [selection : table of types.dt_lua_image_t]  
) : table of types.dt_lua_image_t
```

Allows to change the set of selected images.

Attributes: • *implicit_yield*

selection

table of types.dt_lua_image_t

A table of images which will define the selected images. If this parameter is not given the selection will be untouched. If an empty table is given the selection will be emptied.

return

table of types.dt_lua_image_t

A table containing the selection as it was before the function was called.

darktable.gui.current_view

```
function(  
  [view : types.dt_view_t]  
) : types.dt_view_t
```

Allows to change the current view.

view

types.dt_view_t

The view to switch to. If empty the current view is unchanged

return

types.dt_view_t

the current view

darktable.gui.create_job

```
function(  
  text : string,  
  [percentage : boolean],  
  [cancel_callback : function]  
) : types.dt_lua_backgroundjob_t
```

Create a new progress_bar displayed in darktable.gui.libs.backgroundjobs

text

string

The text to display in the job entry

percentage

boolean

Should a progress bar be displayed

cancel_callback

```
function(  
  job : types.dt_lua_backgroundjob_t  
)
```

A function called when the cancel button for that job is pressed

note that the job won't be destroyed automatically. You need to set `types.dt_lua_backgroundjob_t.valid` to false for that

job

`types.dt_lua_backgroundjob_t`

The job who is being canceled

return

`types.dt_lua_backgroundjob_t`

The newly created job object

darktable.gui.views

The different views in darktable

darktable.gui.views.map

The map view

Attributes:

- *has_tostring*
- *parent* : `types.dt_view_t`

darktable.gui.views.map.latitude

number

The latitude of the center of the map

Attributes:

- *write*

darktable.gui.views.map.longitude

number

The longitude of the center of the map

Attributes:

- *write*

darktable.gui.views.map.zoom

number

The current zoom level of the map

Attributes: • *write*

darktable.gui.views.darkroom

The darkroom view

Attributes: • *has_tostring*
 • *parent* : types.dt_view_t

darktable.gui.views.lighttable

The lighttable view

Attributes: • *has_tostring*
 • *parent* : types.dt_view_t

darktable.gui.views.tethering

The tethering view

Attributes: • *has_tostring*
 • *parent* : types.dt_view_t

darktable.gui.views.slideshow

The slideshow view

Attributes: • *has_tostring*
 • *parent* : types.dt_view_t

darktable.gui.libs

This table allows to reference all lib objects

lib are the graphical blocks within each view.

To quickly figure out what lib is what, you can use the following code which will make a given lib blink.

```
local tested_module="global_toolbox"
dt.gui.libs[tested_module].visible=false
coroutine.yield("wait_ms",2000)
while true do
  dt.gui.libs[tested_module].visible = not dt.gui.libs[tested_module].visible
  coroutine.yield("wait_ms",2000)
end
```

darktable.gui.libs.snapshots

The UI element that manipulates snapshots in darkroom

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.snapshots.ratio

number

The place in the screen where the line separating the snapshot is. Between 0 and 1

- Attributes:
- *write*

darktable.gui.libs.snapshots.direction

types.snapshot_direction_t

The direction of the snapshot overlay

- Attributes:
- *write*

darktable.gui.libs.snapshots.#

types.dt_lua_snapshot_t

The different snapshots for the image

darktable.gui.libs.snapshots.selected

The currently selected snapshot

darktable.gui.libs.snapshots.take_snapshot

```
function(  
)
```

Take a snapshot of the current image and add it to the UI

The snapshot file will be generated at the next redraw of the main window

darktable.gui.libs.snapshots.max_snapshot

number

The maximum number of snapshots

darktable.gui.libs.styles

The style selection menu

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.metadata_view

The widget displaying metadata about the current image

- Attributes:
- *has_tostring*

- *parent* : types.dt_lib_module_t

darktable.gui.libs.metadata

The widget allowing modification of metadata fields on the current image

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.hinter

The small line of text at the top of the UI showing the number of selected images

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.modulelist

The window allowing to set modules as visible/hidden/favorite

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.filmstrip

The filmstrip at the bottom of some views

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.viewswitcher

The labels allowing to switch view

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.darktable_label

The darktable logo in the upper left corner

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.tagging

The tag manipulation UI

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.geotagging

The geotagging time synchronisation UI

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.recentcollect

The recent collection UI element

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.global_toolbox

The common tools to all view (settings, grouping...)

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.global_toolbox.grouping

boolean

The current status of the image grouping option

- Attributes:
- *write*

darktable.gui.libs.global_toolbox.show_overlays

boolean

the current status of the image overlays option

- Attributes:
- *write*

darktable.gui.libs.filter

The image-filter menus at the top of the UI

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.import

The buttons to start importing images

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.ratings

The starts to set the rating of an image

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.select

The buttons that allow to quickly change the selection

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.collect

The collection UI element that allows to filter images by collection

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.colorlabels

The color buttons that allow to set labels on an image

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.lighttable_mode

The navigation and zoom level UI in lighttable

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.copy_history

The UI element that manipulates history

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.image

The UI element that manipulates the current image

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.modulegroups

The icons describing the different iop groups

- Attributes:
- *has_tostring*

- *parent* : types.dt_lib_module_t

darktable.gui.libs.module_toolbox

The tools on the bottom line of the UI (overexposure)

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.session

The session UI when tethering

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.histogram

The histogram widget

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.export

The export menu

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.history

The history manipulation menu

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.colorpicker

The colorpicker menu

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.navigation

The full image preview to allow navigation

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.masks

The masks window

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.view_toolbox

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.live_view

The liveview window

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.map_settings

The map setting window

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.camera

The camera selection UI

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.location

The location ui

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

darktable.gui.libs.backgroundjobs

The window displaying the currently running jobs

- Attributes:
- *has_tostring*
 - *parent* : types.dt_lib_module_t

8.2.1.9. darktable.tags

Allows access to all existing tags.

darktable.tags.#

```
types.dt_lua_tag_t
```

Each existing tag has a numeric entry in the tags table - use `ipairs` to iterate over them.

darktable.tags.create

```
function(  
  name : string  
)
```

Creates a new tag and return it. If the tag exists return the existing tag.

name

```
string
```

The name of the new tag.

darktable.tags.find

```
function(  
  name : string  
) : types.dt_lua_tag_t
```

Returns the tag object or nil if the tag doesn't exist.

name

```
string
```

The name of the tag to find.

return

```
types.dt_lua_tag_t
```

The tag object or nil.

darktable.tags.delete

```
function(  
  tag : types.dt_lua_tag_t  
)
```

Deletes the tag object, detaching it from all images.

tag

```
types.dt_lua_tag_t
```

The tag to be deleted.

darktable.tags.attach

```
function(  
  tag : types.dt_lua_tag_t,  
  image : types.dt_lua_image_t  
)
```

Attach a tag to an image; the order of the parameters can be reversed.

tag

```
types.dt_lua_tag_t
```

The tag to be attached.

image

```
types.dt_lua_image_t
```

The image to attach the tag to.

darktable.tags.detach

```
function(  
  tag : types.dt_lua_tag_t,  
  image : types.dt_lua_image_t  
)
```

Detach a tag from an image; the order of the parameters can be reversed.

tag

```
types.dt_lua_tag_t
```

The tag to be detached.

image

```
types.dt_lua_image_t
```

The image to detach the tag from.

darktable.tags.get_tags

```
function(  
  image : types.dt_lua_image_t  
) : table of types.dt_lua_tag_t
```

Gets all tags attached to an image.

image

```
types.dt_lua_image_t
```

The image to get the tags from.

return

```
table of types.dt_lua_tag_t
```

A table of tags that are attached to the image.

8.2.1.10. darktable.configuration

table

This table regroups values that describe details of the configuration of darktable.

darktable.configuration.version

string

The version number of darktable.

darktable.configuration.has_gui

boolean

True if darktable has a GUI (launched through the main darktable command, not darktable-cli).

darktable.configuration.verbose

boolean

True if the Lua logdomain is enabled.

darktable.configuration.tmp_dir

string

The name of the directory where darktable will store temporary files.

darktable.configuration.config_dir

string

The name of the directory where darktable will find its global configuration objects (modules).

darktable.configuration.cache_dir

string

The name of the directory where darktable will store its mipmaps.

darktable.configuration.api_version_major

number

The major version number of the lua API.

darktable.configuration.api_version_minor

number

The minor version number of the lua API.

darktable.configuration.api_version_patch

number

The patch version number of the lua API.

darktable.configuration.api_version_suffix

string

The version suffix of the lua API.

darktable.configuration.api_version_string

string

The version description of the lua API. This is a string compatible with the semantic versioning convention

darktable.configuration.check_version

```
function(  
  module_name : string,  
  ... : table...  
)
```

Check that a module is compatible with the running version of darktable

Add the following line at the top of your module :

```
darktable.configuration.check(..., {M,m,p}, {M2,m2,p2})
```

To document that your module has been tested with API version M.m.p and M2.m2.p2.

This will raise an error if the user is running a released version of DT and a warning if he is running a development version

(the ... here will automatically expand to your module name if used at the top of your script

module_name

string

The name of the module to report on error

...

table...

Tables of API versions that are known to work with the scrip

8.2.1.11. darktable.preferences

table

Lua allows you do manipulate preferences. Lua has its own namespace for preferences and you can't access nor write normal darktable preferences.

Preference handling functions take a `_script_` parameter. This is a string used to avoid name collision in preferences (i.e namespace). Set it to something unique, usually the name of the script handling the preference.

Preference handling functions can't guess the type of a parameter. You must pass the type of the preference you are handling.

Note that the directory, enum and file type preferences are stored internally as string. The user can only select valid values, but a lua script can set it to any string

darktable.preferences.register

```

function(
  script : string,
  name : string,
  type : types.lua_pref_type,
  label : string,
  tooltip : string,
  [default : depends on type],
  [min : int or float],
  [max : int or float],
  [step : float],
  values : string...
)

```

Creates a new preference entry in the Lua tab of the preference screen. If this function is not called the preference can't be set by the user (you can still read and write invisible preferences).

script

string

Invisible prefix to guarantee unicity of preferences.

name

string

A unique name used with the script part to identify the preference.

type

types.lua_pref_type

The type of the preference - one of the string values described above.

label

string

The label displayed in the preference screen.

tooltip

string

The tooltip to display in the preference menu.

default

depends on type

Default value to use when not set explicitly or by the user.

For the enum type of pref, this is mandatory

min

int or float

Minimum value (integer and float preferences only).

max

int or float

Maximum value (integer and float preferences only).

step

float

Step of the spinner (float preferences only).

values

string...

Other allowed values (enum preferences only)

darktable.preferences.read

```
function(  
  script : string,  
  name : string,  
  type : types.lua_pref_type  
) : depends on type
```

Reads a value from a Lua preference.

script

string

Invisible prefix to guarantee unicity of preferences.

name

string

The name of the preference displayed in the preference screen.

type

types.lua_pref_type

The type of the preference.

return

depends on type

The value of the preference.

darktable.preferences.write

```
function(  
  script : string,  
  name : string,
```

```
type : types.lua_pref_type,  
value : depends on type  
)
```

Writes a value to a Lua preference.

script

```
string
```

Invisible prefix to guarantee unicity of preferences.

name

```
string
```

The name of the preference displayed in the preference screen.

type

```
types.lua_pref_type
```

The type of the preference.

value

```
depends on type
```

The value to set the preference to.

8.2.1.12. darktable.styles

This pseudo table allows you to access and manipulate styles.

darktable.styles.#

```
types.dt_style_t
```

Each existing style has a numeric index; you can iterate them using ipairs.

darktable.styles.create

```
function(  
  image : types.dt_lua_image_t,  
  name : string,  
  description : string  
) : types.dt_style_t
```

Create a new style based on an image.

image

```
types.dt_lua_image_t
```

The image to create the style from.

name

```
string
```

The name to give to the new style.

description

string

The description of the new style.

return

types.dt_style_t

The new style object.

darktable.styles.delete

```
function(  
  style : types.dt_style_t  
)
```

Deletes an existing style.

style

types.dt_style_t

the style to delete

darktable.styles.duplicate

```
function(  
  style : types.dt_style_t,  
  name : string,  
  description : string  
) : types.dt_style_t
```

Create a new style based on an existing style.

style

types.dt_style_t

The style to base the new style on.

name

string

The new style's name.

description

string

The new style's description.

return

types.dt_style_t

The new style object.

darktable.styles.apply

```
function(  
  style : types.dt_style_t,  
  image : types.dt_lua_image_t  
)
```

Apply a style to an image. The order of parameters can be inverted.

style

```
types.dt_style_t
```

The style to use.

image

```
types.dt_lua_image_t
```

The image to apply the style to.

darktable.styles.import

```
function(  
  filename : string  
)
```

Import a style from an external .dtstyle file

filename

```
string
```

The file to import

darktable.styles.export

```
function(  
  style : types.dt_style_t,  
  directory : string,  
  overwrite : boolean  
)
```

Export a style to an external .dtstyle file

style

```
types.dt_style_t
```

The style to export

directory

```
string
```

The directory to export to

overwrite

boolean

Is overwriting an existing file allowed

8.2.1.13. **darktable.database**

Allows to access the database of images. Note that duplicate images (images with the same RAW but different XMP) will appear multiple times with different duplicate indexes. Also note that all images are here. This table is not influenced by any GUI filtering (collections, stars etc..).

darktable.database.#

`types.dt_lua_image_t`

Each image in the database appears with a numerical index; you can iterate them using `ipairs`.

darktable.database.duplicate

```
function(  
  image : types.dt_lua_image_t  
) : types.dt_lua_image_t
```

Creates a duplicate of an image and returns it.

image

`types.dt_lua_image_t`

the image to duplicate

return

`types.dt_lua_image_t`

The created image if an image is imported or the toplevel film object if a film was imported.

darktable.database.import

```
function(  
  location : string  
)
```

Imports new images into the database.

location

string

The filename or directory to import images from. NOTE: If the images are set to be imported recursively in preferences only the toplevel film is returned (the one whose path was given as a parameter). NOTE2: If the parameter is a directory the call is non-blocking; the film object will not have the newly imported images yet. Use a post-import-film filtering on that film to react when images are actually imported.

darktable.database.move_image

```
function(  
  image : types.dt_lua_image_t,  
  film : types.dt_lua_film_t  
)
```

Physically moves an image (and all its duplicates) to another film.

This will move the image file, the related XMP and all XMP for the duplicates to the directory of the new film

Note that the parameter order is not relevant.

image

```
types.dt_lua_image_t
```

The image to move

film

```
types.dt_lua_film_t
```

The film to move to

darktable.database.copy_image

```
function(  
  image : types.dt_lua_image_t,  
  film : types.dt_lua_film_t  
) : types.dt_lua_image_t
```

Physically copies an image to another film.

This will copy the image file and the related XMP to the directory of the new film

If there is already a file with the same name as the image file, it will create a duplicate from that file instead

Note that the parameter order is not relevant.

image

```
types.dt_lua_image_t
```

The image to copy

film

```
types.dt_lua_film_t
```

The film to copy to

return

```
types.dt_lua_image_t
```

The new image

darktable.database.delete

see `types.dt_lua_image_t.delete`

8.2.1.14. darktable.debug

table

This section must be activated separately by calling `require "darktable.debug"`

darktable.debug.dump

```
function(  
  object : anything,  
  [name : string],  
  [known : table]  
) : string
```

This will return a string describing everything Lua knows about an object, used to know what an object is. This function is recursion-safe and can be used to dump `_G` if needed.

object

anything

The object to dump.

name

string

A name to use for the object.

known

table

A table of object,string pairs. Any object in that table will not be dumped, the string will be printed instead.

defaults to `darktable.debug.known` if not set

return

string

A string containing a text description of the object - can be very long.

darktable.debug.debug

boolean

Initialized to false; set it to true to also dump information about metatables.

darktable.debug.max_depth

number

Initialized to 10; The maximum depth to recursively dump content.

darktable.debug.known

table

A table containing the default value of `darktable.debug.dump.known`

darktable.debug.type

```
function(  
  object : anything  
) : string
```

Similar to the system function `type()` but it will return the real type instead of "userdata" for darktable specific objects.

object

anything

The object whos type must be reported.

return

string

A string describing the type of the object.

8.2.2. types

This section documents types that are specific to darktable's Lua API.

8.2.2.1. types.dt_lua_image_t

dt_type

Image objects represent an image in the database. This is slightly different from a file on disk since a file can have multiple developements. Note that this is the real image object; changing the value of a field will immediately change it in darktable and will be reflected on any copy of that image object you may have kept.

Attributes: • *has_tostring*

types.dt_lua_image_t.attach_tag

see `darktable.tags.attach`

types.dt_lua_image_t.detach_tag

see `darktable.tags.detach`

types.dt_lua_image_t.get_tags

see `darktable.tags.get_tags`

types.dt_lua_image_t.create_style

see `darktable.styles.create`

types.dt_lua_image_t.apply_style

see `darktable.styles.apply`

types.dt_lua_image_t.duplicate

see `darktable.database.duplicate`

types.dt_lua_image_t.move

see `darktable.database.move_image`

types.dt_lua_image_t.copy

see `darktable.database.copy_image`

types.dt_lua_image_t.id

number

A unique id identifying the image in the database.

types.dt_lua_image_t.path

string

The file the directory containing the image.

types.dt_lua_image_t.film

`types.dt_lua_film_t`

The film object that contains this image.

types.dt_lua_image_t.filename

string

The filename of the image.

types.dt_lua_image_t.duplicate_index

number

If there are multiple images based on a same file, each will have a unique number, starting from 0.

types.dt_lua_image_t.publisher

string

The publisher field of the image.

Attributes: • *write*

types.dt_lua_image_t.title

string

The title field of the image.

Attributes: • *write*

types.dt_lua_image_t.creator

string

The creator field of the image.

Attributes: • *write*

types.dt_lua_image_t.rights

string

The rights field of the image.

Attributes: • *write*

types.dt_lua_image_t.description

string

The description field for the image.

Attributes: • *write*

types.dt_lua_image_t.exif_maker

string

The maker exif data.

Attributes: • *write*

types.dt_lua_image_t.exif_model

string

The camera model used.

Attributes: • *write*

types.dt_lua_image_t.exif_lens

string

The id string of the lens used.

Attributes: • *write*

types.dt_lua_image_t.exif_aperture

number

The aperture saved in the exif data.

Attributes: • *write*

types.dt_lua_image_t.exif_exposure

number

The exposure time of the image.

Attributes: • *write*

types.dt_lua_image_t.exif_focal_length

number

The focal length of the image.

Attributes: • *write*

types.dt_lua_image_t.exif_iso

number

The iso used on the image.

Attributes: • *write*

types.dt_lua_image_t.exif_datetime_taken

string

The date and time of the image.

Attributes: • *write*

types.dt_lua_image_t.exif_focus_distance

number

The distance of the subject.

Attributes: • *write*

types.dt_lua_image_t.exif_crop

number

The exif crop data.

Attributes: • *write*

types.dt_lua_image_t.latitude

GPS latitude data of the image, nil if not set.

Attributes: • *write*

types.dt_lua_image_t.longitude

GPS longitude data of the image, nil if not set.

Attributes: • *write*

types.dt_lua_image_t.is_raw

boolean

True if the image is a RAW file.

types.dt_lua_image_t.is_ldr

boolean

True if the image is a ldr image.

types.dt_lua_image_t.is_hdr

boolean

True if the image is a hdr image.

types.dt_lua_image_t.width

number

The width of the image.

types.dt_lua_image_t.height

number

The height of the image.

types.dt_lua_image_t.rating

number

The rating of the image (-1 for rejected).

Attributes: • *write*

types.dt_lua_image_t.red

boolean

True if the image has the corresponding colorlabel.

Attributes: • *write*

types.dt_lua_image_t.blue

see types.dt_lua_image_t.red

types.dt_lua_image_t.green

see types.dt_lua_image_t.red

types.dt_lua_image_t.yellow

see `types.dt_lua_image_t.red`

types.dt_lua_image_t.purple

see `types.dt_lua_image_t.red`

types.dt_lua_image_t.reset

```
self: function(  
)
```

Removes all processing from the image, resetting it back to its original state

self

```
types.dt_lua_image_t
```

The image whose history will be deleted

types.dt_lua_image_t.delete

```
self: function(  
)
```

Removes an image from the database

self

```
types.dt_lua_image_t
```

The image to remove

types.dt_lua_image_t.group_with

```
self: function(  
  [image : types.dt_lua_image_t]  
)
```

Puts the first image in the same group as the second image. If no second image is provided the image will be in its own group.

self

```
types.dt_lua_image_t
```

The image whose group must be changed.

image

```
types.dt_lua_image_t
```

The image we want to group with.

types.dt_lua_image_t.make_group_leader

```
self: function(  
)
```

Makes the image the leader of its group.

self

types.dt_lua_image_t

The image we want as the leader.

types.dt_lua_image_t.get_group_members

```
self: function(  
    ) : table of types.dt_lua_image_t
```

Returns a table containing all types.dt_lua_image_t of the group. The group leader is both at a numeric key and at the "leader" special key (so you probably want to use ipairs to iterate through that table).

self

types.dt_lua_image_t

The image whose group we are querying.

return

table of types.dt_lua_image_t

A table of image objects containing all images that are in the same group as the image.

types.dt_lua_image_t.group_leader

types.dt_lua_image_t

The image which is the leader of the group this image is a member of.

types.dt_lua_image_t.local_copy

boolean

True if the image has a copy in the local cache

Attributes: • *write*

types.dt_lua_image_t.drop_cache

```
self: function(  
    )
```

drops the cached version of this image.

This function should be called if an image is modified out of darktable to force DT to regenerate the thumbnail

Darktable will regenerate the thumbnail by itself when it is needed

self

types.dt_lua_image_t

The image whose cache must be dropped.

8.2.2.2. `types.dt_imageio_module_format_t`

`dt_type`

A virtual type representing all format types.

`types.dt_imageio_module_format_t.plugin_name`

`string`

A unique name for the plugin.

`types.dt_imageio_module_format_t.name`

`string`

A human readable name for the plugin.

`types.dt_imageio_module_format_t.extension`

`string`

The typical filename extension for that format.

`types.dt_imageio_module_format_t.mime`

`string`

The mime type associated with the format.

`types.dt_imageio_module_format_t.max_width`

`number`

The max width allowed for the format (0 = unlimited).

Attributes: • *write*

`types.dt_imageio_module_format_t.max_height`

`number`

The max height allowed for the format (0 = unlimited).

Attributes: • *write*

`types.dt_imageio_module_format_t.write_image`

```
self:function(  
  image : types.dt_lua_image_t,  
  filename : string  
) : boolean
```

Exports an image to a file. This is a blocking operation that will not return until the image is exported.

Attributes: • *implicit_yield*

self

`types.dt_imageio_module_format_t`

The format that will be used to export.

image

`types.dt_lua_image_t`

The image object to export.

filename

string

The filename to export to.

return

boolean

Returns true on success.

8.2.2.3. types.dt_imageio_module_format_data_png

`dt_type`

Type object describing parameters to export to png.

Attributes: • *parent*: `types.dt_imageio_module_format_t`

types.dt_imageio_module_format_data_png.bpp

number

The bpp parameter to use when exporting.

Attributes: • *write*

8.2.2.4. types.dt_imageio_module_format_data_tiff

`dt_type`

Type object describing parameters to export to tiff.

Attributes: • *parent*: `types.dt_imageio_module_format_t`

types.dt_imageio_module_format_data_tiff.bpp

number

The bpp parameter to use when exporting.

Attributes: • *write*

8.2.2.5. types.dt_imageio_module_format_data_exr

dt_type

Type object describing parameters to export to exr.

Attributes: • *parent*: types.dt_imageio_module_format_t

types.dt_imageio_module_format_data_exr.compression

string

The compression parameter to use when exporting.

Attributes: • *write*

8.2.2.6. types.dt_imageio_module_format_data_copy

dt_type

Type object describing parameters to export to copy.

Attributes: • *parent*: types.dt_imageio_module_format_t

8.2.2.7. types.dt_imageio_module_format_data_pfm

dt_type

Type object describing parameters to export to pfm.

Attributes: • *parent*: types.dt_imageio_module_format_t

8.2.2.8. types.dt_imageio_module_format_data_jpeg

dt_type

Type object describing parameters to export to jpeg.

Attributes: • *parent*: types.dt_imageio_module_format_t

types.dt_imageio_module_format_data_jpeg.quality

number

The quality to use at export time.

Attributes: • *write*

8.2.2.9. types.dt_imageio_module_format_data_ppm

dt_type

Type object describing parameters to export to ppm.

Attributes: • *parent*: types.dt_imageio_module_format_t

8.2.2.10. types.dt_imageio_module_format_data_webp

dt_type

Type object describing parameters to export to webp.

Attributes: • *parent* : `types.dt_imageio_module_format_t`

types.dt_imageio_module_format_data_webp.quality

`number`

The quality to use at export time.

Attributes: • *write*

types.dt_imageio_module_format_data_webp.comp_type

`types.comp_type_t`

The overall quality to use; can be one of "webp_lossy" or "webp_lossless".

Attributes: • *write*

types.dt_imageio_module_format_data_webp.hint

`types.hint_t`

A hint on the overall content of the image.

Attributes: • *write*

8.2.2.11. types.dt_imageio_module_format_data_j2k

`dt_type`

Type object describing parameters to export to jpeg2000.

Attributes: • *parent* : `types.dt_imageio_module_format_t`

types.dt_imageio_module_format_data_j2k.quality

`number`

The quality to use at export time.

Attributes: • *write*

types.dt_imageio_module_format_data_j2k.bpp

`number`

The bpp parameter to use when exporting.

Attributes: • *write*

types.dt_imageio_module_format_data_j2k.format

`types.dt_imageio_j2k_format_t`

The format to use.

Attributes: • *write*

types.dt_imageio_module_format_data_j2k.preset

`types.dt_imageio_j2k_preset_t`

The preset to use.

Attributes: • *write*

8.2.2.12. types.dt_imageio_module_storage_t

`dt_type`

A virtual type representing all storage types.

types.dt_imageio_module_storage_t.plugin_name

`string`

A unique name for the plugin.

Attributes: • *write*

types.dt_imageio_module_storage_t.name

`string`

A human readable name for the plugin.

Attributes: • *write*

types.dt_imageio_module_storage_t.width

`number`

The currently selected width for the plugin.

Attributes: • *write*

types.dt_imageio_module_storage_t.height

`number`

The currently selected height for the plugin.

Attributes: • *write*

types.dt_imageio_module_storage_t.recommended_width

`number`

The recommended width for the plugin.

Attributes: • *write*

types.dt_imageio_module_storage_t.recommended_height

number

The recommended height for the plugin.

Attributes: • *write*

types.dt_imageio_module_storage_t.supports_format

```
self: function(  
  format : types.dt_imageio_module_format_t  
) : boolean
```

Checks if a format is supported by this storage.

self

```
types.dt_imageio_module_storage_t
```

The storage type to check against.

format

```
types.dt_imageio_module_format_t
```

The format type to check.

return

```
boolean
```

True if the format is supported by the storage.

8.2.2.13. types.dt_imageio_module_storage_data_email

```
dt_type
```

An object containing parameters to export to email.

Attributes: • *parent* : types.dt_imageio_module_storage_t

8.2.2.14. types.dt_imageio_module_storage_data_flickr

```
dt_type
```

An object containing parameters to export to flickr.

Attributes: • *parent* : types.dt_imageio_module_storage_t

8.2.2.15. types.dt_imageio_module_storage_data_facebook

```
dt_type
```

An object containing parameters to export to facebook.

Attributes: • *parent* : types.dt_imageio_module_storage_t

8.2.2.16. types.dt_imageio_module_storage_data_latex

dt_type

An object containing parameters to export to latex.

Attributes: • *parent* : types.dt_imageio_module_storage_t

types.dt_imageio_module_storage_data_latex.filename

string

The filename to export to.

Attributes: • *write*

types.dt_imageio_module_storage_data_latex.title

string

The title to use for export.

Attributes: • *write*

8.2.2.17. types.dt_imageio_module_storage_data_picasa

dt_type

An object containing parameters to export to picasa.

Attributes: • *parent* : types.dt_imageio_module_storage_t

8.2.2.18. types.dt_imageio_module_storage_data_gallery

dt_type

An object containing parameters to export to gallery.

Attributes: • *parent* : types.dt_imageio_module_storage_t

types.dt_imageio_module_storage_data_gallery.filename

string

The filename to export to.

Attributes: • *write*

types.dt_imageio_module_storage_data_gallery.title

string

The title to use for export.

Attributes: • *write*

8.2.2.19. types.dt_imageio_module_storage_data_disk

dt_type

An object containing parameters to export to disk.

Attributes: • *parent* : types.dt_imageio_module_storage_t

types.dt_imageio_module_storage_data_disk.filename

string

The filename to export to.

Attributes: • *write*

8.2.2.20. types.dt_lua_film_t

dt_type

A film in darktable; this represents a directory containing imported images.

Attributes: • *has_tostring*

types.dt_lua_film_t.move_image

see darktable.database.move_image

types.dt_lua_film_t.copy_image

see darktable.database.copy_image

types.dt_lua_film_t.#

types.dt_lua_image_t

The different images within the film.

types.dt_lua_film_t.id

number

A unique numeric id used by this film.

Attributes: • *write*

types.dt_lua_film_t.path

string

The path represented by this film.

Attributes: • *write*

types.dt_lua_film_t.delete

```
self: function(  
  [force : Boolean]  
)
```

Removes the film from the database.

self

`types.dt_lua_film_t`

The film to remove.

force

Boolean

Force removal, even if the film is not empty.

8.2.2.21. `types.dt_style_t`

`dt_type`

A style that can be applied to an image.

Attributes: • *has_tostring*

`types.dt_style_t.delete`

see `darktable.styles.delete`

`types.dt_style_t.duplicate`

see `darktable.styles.duplicate`

`types.dt_style_t.apply`

see `darktable.styles.apply`

`types.dt_style_t.export`

see `darktable.styles.export`

`types.dt_style_t.name`

string

The name of the style.

`types.dt_style_t.description`

string

The description of the style.

`types.dt_style_t.#`

`types.dt_style_item_t`

The different items that make the style.

8.2.2.22. `types.dt_style_item_t`

`dt_type`

An element that is part of a style.

Attributes: • *has_tostring*

types.dt_style_item_t.name

string

The name of the style item.

types.dt_style_item_t.num

number

The position of the style item within its style.

8.2.2.23. types.dt_lua_tag_t

dt_type

A tag that can be attached to an image.

Attributes: • *has_tostring*

types.dt_lua_tag_t.delete

see darktable.tags.delete

types.dt_lua_tag_t.attach

see darktable.tags.attach

types.dt_lua_tag_t.detach

see darktable.tags.detach

types.dt_lua_tag_t.name

string

The name of the tag.

types.dt_lua_tag_t.#

types.dt_lua_image_t

The images that have that tag attached to them.

8.2.2.24. types.dt_lib_module_t

dt_type

The type of a UI lib

types.dt_lib_module_t.id

string

A unit string identifying the lib

types.dt_lib_module_t.name

string

The translated title of the UI element

types.dt_lib_module_t.version

number

The version of the internal data of this lib

types.dt_lib_module_t.visible

boolean

Allow to make a lib module completely invisible to the user.

Note that if the module is invisible the user will have no way to restore it without lua

Attributes: • *implicit_yield*
 • *write*

types.dt_lib_module_t.expandable

boolean

True if the lib can be expanded/retracted

types.dt_lib_module_t.expanded

boolean

True if the lib is expanded

Attributes: • *write*

types.dt_lib_module_t.reset

```
self: function(  
)
```

A function to reset the lib to its default values

This function will do nothing if the lib is not visible or can't be reset

self

```
    types.dt_lib_module_t
```

The lib to reset

types.dt_lib_module_t.on_screen

boolean

True if the lib is currently visible on the screen

8.2.2.25. types.dt_view_t

dt_type

A darktable view

types.dt_view_t.id

string

A unique string identifying the view

types.dt_view_t.name

string

The name of the view

8.2.2.26. types.dt_lua_backgroundjob_t

dt_type

A lua-managed entry in the backgroundjob lib

types.dt_lua_backgroundjob_t.percent

number

The value of the progress bar, between 0 and 1. will return nil if there is no progress bar, will raise an error if read or written on an invalid job

Attributes: • *write*

types.dt_lua_backgroundjob_t.valid

boolean

True if the job is displayed, set it to false to destroy the entry

An invalid job cannot be made valid again

Attributes: • *write*

8.2.2.27. types.dt_lua_snapshot_t

dt_type

The description of a snapshot in the snapshot lib

Attributes: • *has_tostring*

types.dt_lua_snapshot_t.filename

string

The filename of an image containing the snapshot

types.dt_lua_snapshot_t.select

self: function(

)

Activates this snapshot on the display. To deactivate all snapshot you need to call this function on the active snapshot

self

`types.dt_lua_snapshot_t`

The snapshot to activate

types.dt_lua_snapshot_t.name

string

The name of the snapshot, as seen in the UI

8.2.2.28. types.hint_t

enum

a hint on the way to encode a webp image

- Attributes:
- *values* :
 - hint_default
 - hint_picture
 - hint_photo
 - hint_graphic

8.2.2.29. types.snapshot_direction_t

enum

Which part of the main window is occupied by a snapshot

- Attributes:
- *values* :
 - left
 - right
 - top
 - bottom

8.2.2.30. types.dt_imageio_j2k_format_t

enum

J2K format type

- Attributes:
- *values* :
 - j2k

- jp2

8.2.2.31. `types.dt_imageio_j2k_preset_t`

enum

J2K preset type

- Attributes:
- *values* :
 - off
 - cinema2k_24
 - cinema2k_48
 - cinema4k_24

8.2.2.32. `types.yield_type`

enum

What type of event to wait for

- Attributes:
- *values* :
 - WAIT_MS
 - FILE_READABLE
 - RUN_COMMAND

8.2.2.33. `types.comp_type_t`

enum

Type of compression for webp

- Attributes:
- *values* :
 - webp_lossy
 - webp_lossless

8.2.2.34. `types.lua_pref_type`

enum

The type of value to save in a preference

- Attributes:
- *values* :
 - string
 - bool
 - integer

- float
- file
- directory
- enum

8.2.2.35. `types.dt_imageio_exr_compression_t`

enum

undocumented `types.dt_imageio_exr_compression_t`

- Attributes:
- *values*:
 - off
 - rle
 - zips
 - zip
 - piz
 - pxr24
 - b44
 - b44a

8.2.3. events

This section documents events that can be used to trigger Lua callbacks.

8.2.3.1. `events.intermediate-export-image`

event

This event is called each time an image is exported, once for each image after the image has been processed to an image format but before the storage has moved the image to its final destination.

`events.intermediate-export-image.callback`

```
function(
  event : string,
  image : types.dt_lua_image_t,
  filename : string,
  format : types.dt_imageio_module_format_t,
  storage : types.dt_imageio_module_storage_t
)
```

event

string

The name of the event that triggered the callback.

image

`types.dt_lua_image_t`

The image object that has been exported.

filename

`string`

The name of the file that is the result of the image being processed.

format

`types.dt_imageio_module_format_t`

The format used to export the image.

storage

`types.dt_imageio_module_storage_t`

The storage used to export the image (can be nil).

events.intermediate-export-image.extra registration parameters

This event has no extra registration parameters.

8.2.3.2. events.post-import-image

event

This event is triggered whenever a new image is imported into the database. This event can be registered multiple times, all callbacks will be called.

events.post-import-image.callback

```
function(  
  event : string,  
  image : types.dt_lua_image_t  
)
```

event

`string`

The name of the event that triggered the callback.

image

`types.dt_lua_image_t`

The image object that has been exported.

events.post-import-image.extra registration parameters

This event has no extra registration parameters.

8.2.3.3. `events.shortcut`

event

This event registers a new keyboard shortcut. The shortcut isn't bound to any key until the users does so in the preference panel. The event is triggered whenever the shortcut is triggered. This event can only be registered once per value of shortcut.

`events.shortcut.callback`

```
function(  
    event : string,  
    shortcut : string  
)
```

event

string

The name of the event that triggered the callback.

shortcut

string

The tooltip string that was given at registration time.

`events.shortcut.extra registration parameters`

tooltip

string

The string that will be displayed on the shortcut preference panel describing the shortcut.

8.2.3.4. `events.post-import-film`

event

This event is triggered when an film import is finished (all post-import-image callbacks have already been triggered). This event can be registered multiple times.

`events.post-import-film.callback`

```
function(  
    event : string,  
    film : types.dt_lua_film_t  
)
```

event

string

The name of the event that triggered the callback.

film

`types.dt_lua_film_t`

The new film that has been added. If multiple films were added recursively only the top level film is reported.

events.post-import-film.extra registration parameters

This event has no extra registration parameters.

8.2.3.5. events.view-changed

event

This event is triggered after the user changed the active view

events.view-changed.callback

```
function(  
  old_view : types.dt_view_t,  
  new_view : types.dt_view_t  
)
```

old_view

`types.dt_view_t`

The view that we just left

new_view

`types.dt_view_t`

The view we are now in

events.view-changed.extra registration parameters

This event has no extra registration parameters.

8.2.3.6. events.global_toolbox-grouping_toggle

event

This event is triggered after the user toggled the grouping button.

events.global_toolbox-grouping_toggle.callback

```
function(  
  toggle : boolean  
)
```

toggle

boolean

the new grouping status.

events.global_toolbox-grouping_toggle.extra registration parameters

This event has no extra registration parameters.

8.2.3.7. events.global_toolbox-overlay_toggle

event

This event is triggered after the user toggled the overlay button.

events.global_toolbox-overlay_toggle.callback

```
function(  
  toggle : boolean  
)
```

toggle

boolean

the new overlay status.

events.global_toolbox-overlay_toggle.extra registration parameters

This event has no extra registration parameters.

8.2.4. attributes

This section documents various attributes used throughout the documentation.

8.2.4.1. attributes.write

This object is a variable that can be written to.

8.2.4.2. attributes.has_tostring

This object has a specific reimplementation of the "tostring" method that allows pretty-printing it.

8.2.4.3. attributes.implicit_yield

This call will release the Lua lock while executing, thus allowing other Lua callbacks to run.

8.2.4.4. attributes.parent

This object inherits some methods from another object. You can call the methods from the parent on the child object

8.2.5. system

This section documents changes to system functions.

8.2.5.1. system.coroutine

system.coroutine.yield

```
function(  
  ...  
)
```

```
type : types.yield_type,  
extra : variable  
) : variable
```

Lua functions can yield at any point. The parameters and return types depend on why we want to yield.

A callback that is yielding allows other Lua code to run.

- `wait_ms`: one extra parameter; the execution will pause for that many milliseconds; yield returns nothing;
- `file_readable`: an opened file from a call to the OS library; will return when the file is readable; returns nothing;
- `run_command`: a command to be run by "sh -c"; will return when the command terminates; returns the return code of the execution.

type

```
types.yield_type
```

The type of yield.

extra

```
variable
```

An extra parameter: integer for "wait_ms", open file for "file_readable", string for "run_command".

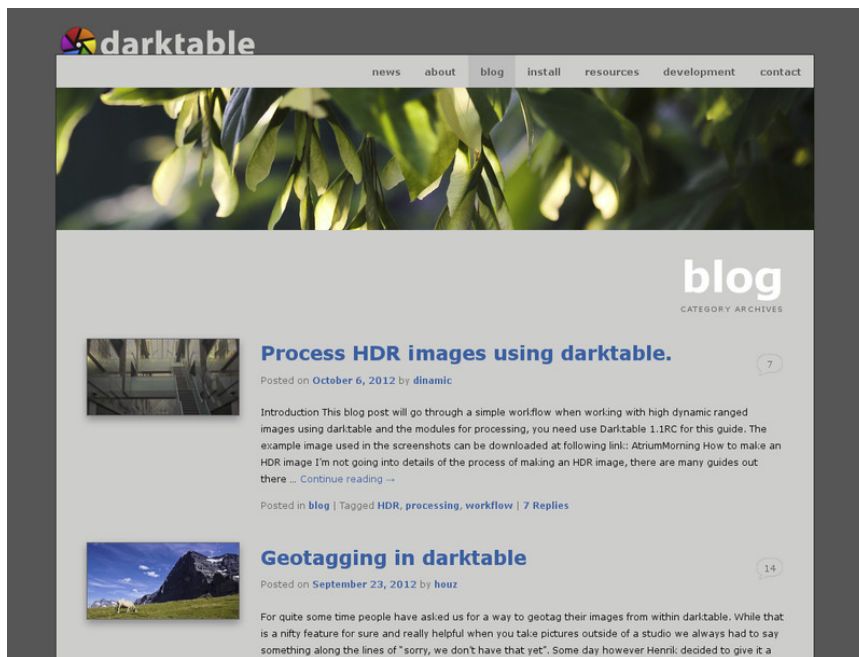
return

```
variable
```

Nothing for "wait_ms" and "file_readable"; the returned code of the command for "run_command".

Chapter 9. Special topics

This chapter touches several technical topics which might help you to get darktable running on specific hardware or optimize its performance. A lot of additional technical background information and many tips and tricks are also covered in an extensive blog section that you can find on our homepage [<http://www.darktable.org>].



9.1. darktable and memory

darktable's memory requirements are high. A simple calculation makes this clear. If you have a 20MPx image, darktable for precision reasons will store this internally as a 4×32 -bit floating point cell for each pixel. Each full image of this size will need about 300MB of memory. As we want to process the image, we will at least need two buffers for each module – one for input and one for output. If we have a more complex module, its algorithm might additionally require several intermediate buffers of the same size. Without further optimization, anything between 600MB and 3GB would be needed only to store and process image data. On top we have darktable's code segment, the code and data of all dynamically linked system libraries, and not to forget further buffers where darktable stores intermediate images for quick access during interactive work (mip map cache). All in all, darktable would like to see a minimum of about 4GB to run happily.

9.1.1. Total system memory

From what I said before, it is evident that your computer needs a sane memory setup to properly run darktable. We suggest that you have a least 4GB of physical RAM plus 4 to 8GB of additional swap space installed. The latter is required, so that your system can swap out temporarily unneeded data to disk in order to free physical RAM.

Theoretically, you could also run darktable with lower amounts of physical RAM and balance this with enough swap space. However, you should be prepared that your system could then heavily "thrash", as it reads or writes data pages to and from the hard disk. We have positive reports that this functions well for several users, but it still might get extremely slow for others...

9.1.2. Available address space

Besides the total amount of system memory there is another limiting factor: the available address space of your hardware architecture. How much memory can be addressed by a process depends on the number of address bits your CPU offers. For a CPU with 32-bit address registers, this is 2^{32} bytes, which makes a total of 4GB. This is the absolute upper limit of memory that can be used by a process and it constitutes a tight situation for darktable as we have seen above.

darktable's escape route is called tiling. Instead of processing an image in one big chunk, we split the image into smaller parts for every processing step (module). This will still require one full input and output buffer, but intermediate buffers can be made small enough to have everything fit into the hardware limits.

9.1.3. Memory fragmentation

Unfortunately this is not the full story yet. There is an effect called memory fragmentation, which can and will hit software that needs to do extensive memory management. If such a program allocates 5 times 300MB at a time and frees it again, that memory should normally be available for one big 1.5GB allocation afterwards. This however is often not the case. The system's memory allocator may no longer see this area as one contiguous 1.5GB block but as a row of 300MB areas. If there is no other free area of 1.5GB available, the allocation would fail. During a program run this mechanism will take away more and more of the larger memory blocks in favor of smaller ones. darktable 1.0 introduces a caching algorithm to address this problem. It pre-allocates blocks of memory and makes them available on request.

9.1.4. Further limitations

As if this were not challenging enough, there are further things that might limit your access to memory. On some older boards you need to activate BIOS option “memory remapping” in order to have all physically installed memory enabled. In addition if you are on a 32-bit OS you will probably need a kernel version that has “Physical Address Extension” (PAE) enabled. This is often but not always the case for Linux. Many distributions deliver different kernels, some with and some without PAE activated; you need to choose the right one. To check if the system is setup correctly, use the command “free” in a terminal and examine the output. If the output reports less RAM than you have installed, you have an issue needing correction; for example you have 4GB on your board, but your kernel is only seeing 3GB or less. You need to consult your BIOS manual and the information about your Linux variant for further help.

9.1.5. Setting up darktable on 32-bit systems

As we’ve seen 32-bit systems are difficult environments for darktable. Still many users are successfully running darktable on them, if the basic requirements in terms of total system memory and the topics mentioned in the paragraphs above are addressed properly.

There are several adjustment parameters to get it running. If you install fresh, darktable will detect your system and set conservative values by default. However, if you upgrade darktable from an older version (e.g. coming from 0.9.3 and going to 1.0), chances are you have unfavorable settings in your preferences. The consequences might be darktable aborting due to allocation failures or – very typically – darktable not being able to properly import a new film roll. As a frequent symptom you get skulls displayed instead of thumbs for many of your pictures.

If this is the case, take a minute to optimize the preference settings in this case. You will find them under “core options” (Section 7.2, “Core options”) in darktable’s preference dialog. You should also find these parameters as configuration variables in `$HOME/.config/darktable/darktable/darktable.rc` and edit them there.

Here is a short explanation of the relevant parameters and their proposed settings:

number of background threads

This parameter defines the maximum number of threads that are allowed in parallel when importing film rolls or doing other background stuff. For obvious reasons on 32-bit systems you can only have one thread eating resources at a time. So you need set this parameter to 1; anything higher will kill you. For the same reason you also must set the number of parallel export threads to 1.

host memory limit (in MB) for tiling

This parameter tells darktable how much memory (in MB) it should assume is available to store image buffers during module operations. If an image can not be processed within these limits in one chunk, tiling will take over and process the image in several parts, one after the other. Set this to the lowest possible value of 500 as a starting point. You might experiment later whether you can increase it a bit in order to reduce the overhead of tiling.

minimum amount of memory (in MB) for a single buffer in tiling

This is a second parameter that controls tiling. It sets a lower limit for the size of intermediate image buffers in megabytes. The parameter is needed to avoid excessive tiling in some cases (for some modules). Set this parameter to a low value of 8. You might tentatively increase it to 16 later.

memory in megabytes to use for mipmap cache

This controls how many thumbnails (or mip maps) can be stored in memory at a time. As a starting point set this to something like 256MB. To avoid the problem of memory fragmentation during longer runs of darktable, the new caching scheme frontloads the memory costs and allocates this cache once at the beginning. Some Linux kernels use over-committing memory allocation, which means you don't immediately pay for all of the memory in terms of RSS (resident set size, the non-swapped physical memory), but in any case you pay for the address space. As explained before, this poses a problem for 32-bit systems and will, at first sight, appear as a regression over the 0.9.3-style cache. In the long run however, this is all the memory that's ever going to be allocated for thumbnails. So if we can successfully grab this portion once, we are relieving a lot of pressure on fragmentation for long sessions.

9.1.6. darktable on 64-bit systems

There's not much to be said here. Of course also 64-bit systems require a sufficient amount of main memory, so the 4GB plus swap recommendation holds true. On the other hand, 64-bit architectures do not suffer from the specific 32-bit limitations like small address space and fragmentation madness.

Most modern Intel or AMD 64-bit CPUs will have available address space in the range of several Terabytes. The word "modern" is relative in this context: all AMD and Intel CPUs introduced since 2003 and 2004, respectively, offer a 64-bit mode. Linux 64-bit has been available for many years.

All relevant Linux distributions give you the choice to install a 32-bit or a 64-bit version with no added costs. You can even run old 32-bit binaries on a 64-bit Linux. The only thing you need to do: invest some time into the migration. In the end we strongly recommend moving to a 64-bit version of Linux. There really is no reason not to upgrade to 64-bit.

On a 64-bit system you can safely leave the tiling related configuration parameters at their defaults: "host memory limit (in MB) for tiling" should have a value of 1500 and "minimum amount of memory (in MB) for a single buffer in tiling" should be set to 16. In case you are migrating from a 32-bit to a 64-bit system you will need to check these settings and manually change them if needed in darktable's preference dialog.

Typically there is no need to restrict oneself in the number of background threads on a 64-bit system. On a multi-processor system a number of two to eight threads can speed up thumbnail generation considerably versus only one thread. The reason is not so much taking maximum advantage of all your CPU cores – darktable's pixelpipe anyhow uses all of them in parallel – but hiding I/O latency.

One exception is worth to be mentioned. If you use darktable to process stitched panoramas, e.g. TIFFs as generated by Hugin, these images can reach considerable sizes. Each background thread needs to allocate enough memory to keep one full image plus intermediates and output in its buffers. This may quickly run even a well equipped 64-bit system out of memory. In that case lower the number of background threads to only one.

9.2. darktable and OpenCL

darktable can use GPU acceleration via OpenCL to improve performance.

9.2.1. The background

Processing high resolution images is a demanding task needing a modern computer. Both in terms of memory requirements and in terms of CPU power, getting the best out of a typical 15, 20 or 25 Megapixel image can quickly bring your computer to its limits.

darktable's requirements are no exception. Our decision to not compromise processing quality, has led to all calculations being done on 4×32 bit floating point numbers. This is slower than "ordinary" 8 or 16bit integer algebra, but eliminates all problems of tonal breaks or loss of information.

A lot of hand optimization has been invested to make darktable as fast as possible. If you run a current version of darktable on a modern computer, you might not notice any "slowness". However, there are conditions and certain modules where you will feel (or hear from the howling of your CPU fan) how much your poor multi-core processor has to struggle.

That's where OpenCL comes in. OpenCL allows us to take advantage of the enormous power of modern graphics cards. Gamer's demand for high detailed 3D worlds in modern ego shooters has fostered GPU development. ATI, NVIDIA and Co had to put enormous processing power into their GPUs to meet these demands. The result is modern graphics cards with highly parallelized GPUs to quickly calculate surfaces and textures at high frame rates.

You are not a gamer and you don't take advantage of that power? Well, then you should at least use it in darktable! For the task of highly parallel floating point calculations modern GPUs are much faster than CPUs. That is especially true, when you want to do the same few processing steps over millions of items. Typical use case: processing of megapixel images.

9.2.2. How OpenCL works

As you can imagine, hardware architectures of GPUs can vary significantly. There are different manufacturers, and even different generations of GPUs from the same manufacturer may differ. At the same time GPU manufacturers don't normally disclose all hardware details of their products to the public. One of the known consequences is the need to use proprietary drivers under Linux, if you want to take full advantage of your graphics card.

Fortunately an industry consortium lead by The Khronos Group has developed an open, standardized interface called OpenCL. It allows the use of your GPU as a numerical processing device. OpenCL offers a C99-like programming language with a strong focus on parallel computing. An application that wants to use OpenCL will need OpenCL source code that it hands over to a hardware specific OpenCL compiler at run-time. This way the application can use OpenCL on different GPU architectures (even at the same time). All hardware "secrets" are hidden in this compiler and are normally not visible to the user (or the application). The compiled OpenCL code is loaded onto your GPU and – with certain API calls – it is ready to do calculations for you.

9.2.3. How to activate OpenCL in darktable

Using OpenCL in darktable requires that your PC is equipped with a suitable graphics card and that it has the required libraries in place. Namely modern graphics cards from NVIDIA and ATI come with full OpenCL support. The OpenCL compiler is normally shipped as part of the proprietary graphics driver; it is used as a dynamic library called "libOpenCL.so". This library must be in a folder where it is found by your system's dynamic linker.

When darktable starts, it will first try to find and load libOpenCL.so and – on success – check if the available graphics card comes with OpenCL support. A sufficient amount of graphics memory (1GB+) needs to be available to take advantage of the GPU. If that is OK, darktable tries to setup its OpenCL environment: a processing context needs to be initialized, a calculation pipeline to be started, OpenCL source code files (extension is .cl) need to be read and compiled and the included routines (called OpenCL kernels) need to be prepared for darktable's modules. If all that is done, the preparation is finished.

Per default OpenCL support is activated in darktable if all the above steps were successful. If you want to de-activate it you can do so in "core options" (Section 7.2, "Core options") by unchecking "activate opencl support". This configuration parameter also tells you if OpenCL initialization failed: it is greyed out in that case.

You can at any time switch OpenCL support off and on; this will happen immediately. Depending on the type of modules you are using, you will notice the effect as a general speed-up during interactive work and during export. Most modules in darktable can take advantage of OpenCL but not all modules are demanding enough to make a noticeable difference. In order to feel a real difference, take modules like *shadows and highlights*, *sharpen*, *lowpass*, *highpass* or even more extreme *equalizer* and *profiled denoise*.

If you are interested in profiling figures, you can start darktable with command line parameters "-d opencl -d perf". After each run of the pixelpipe you will get a detailed allocation of processing time to each module plus an even more fine grained profile for all used OpenCL kernels.

Besides the speed-up you should not see any difference in the results between CPU and GPU processing. Except of rounding errors, the results are designed to be identical. If, for some reasons, darktable fails to properly finish a GPU calculation, it will normally notice and automatically (and transparently) fall back to CPU processing.

9.2.4. Setting up OpenCL on your system

The huge diversity of systems and the marked differences between OpenCL vendors and driver versions makes it impossible to give an comprehensive overview of how to setup OpenCL. We only can give you an example, in this case for NVIDIA driver version 331.89 on Ubuntu 14.04. We hope that this will serve you as a first impression and will help to solve possible problems of your specific setup.

The principle OpenCL function flow is like this:

darktable -> libOpenCL.so -> libnvidia-opencl.so.1 -> kernel driver module(s) -> GPU

- darktable dynamically links `libOpenCL.so`, a system library which must be accessible to the system's dynamic loader (`ld.so`).
- `libOpenCL.so` will read the vendor specific information file (`/etc/OpenCL/vendors/nvidia.icd`) to find the library which contains the vendor specific OpenCL implementation.
- The vendor specific OpenCL implementation comes as a library `libnvidia-opencl.so.1` (which in our case is a symbolic link to `libnvidia-opencl.so.331.89`).
- `libnvidia-opencl.so.1` needs to talk to the vendor specific kernel modules `nvidia` and `nvidia_uvm` via device special files `/dev/nvidia0`, `/dev/nvidiaact1`, and `/dev/nvidia-uvm`.

At system startup the required device special files (`/dev/nvidia*`) need to be created. If this does not happen on your system by default, the easiest way to set them up and make sure all modules are loaded is installing the `nvidia-modprobe` package (which, at the time of this writing, is only available for “utopic”, but works well with “trusty” and “Mint 17”). You can grab it at this site [<http://packages.ubuntu.com/utopic/nvidia-modprobe>].

A user account which wants to make use of OpenCL from within darktable needs to have read-write access to NVIDIA's device special files. On some systems these files allow world read-write access by default, which avoids permission issues but might be debatable in terms of system security. Other systems restrict the access to a user group, e.g. “video”. In that case your user account has to be member of that group.

To summarise, the packages which needed to be installed in this specific case were:

- `nvidia-331` (331.89-0ubuntu1~xedgers14.04.2)
- `nvidia-331-dev` (331.89-0ubuntu1~xedgers14.04.2)
- `nvidia-331-uvvm` (331.89-0ubuntu1~xedgers14.04.2)
- `nvidia-libopencl1-331` (331.89-0ubuntu1~xedgers14.04.2)
- `nvidia-modprobe` (340.24-1)
- `nvidia-opencl-dev:amd64` (5.5.22-3ubuntu1)
- `nvidia-opencl-icd-331` (331.89-0ubuntu1~xedgers14.04.2)
- `nvidia-settings` (340.24-0ubuntu1~xedgers14.04.1)
- `nvidia-settings-304` (340.24-0ubuntu1~xedgers14.04.1)
- `nvidia-libopencl1-331` (331.89-0ubuntu1~xedgers14.04.2)
- `nvidia-opencl-dev:amd64` (5.5.22-3ubuntu1)
- `nvidia-opencl-icd-331` (331.89-0ubuntu1~xedgers14.04.2)
- `opencl-headers` (1.2-2013.10.23-1)

The list of NVIDIA related kernel modules as reported by `lsmod` is:

```
nvidia
nvidia_uvm
```

The list of NVIDIA related device special files (`ls -l /dev/nvidia*`) should read like:

```
crw-rw-rw- 1 root root 195,  0 Jul 28 21:13 /dev/nvidia0
crw-rw-rw- 1 root root 195, 255 Jul 28 21:13 /dev/nvidiaactl
crw-rw-rw- 1 root root 250,  0 Jul 28 21:13 /dev/nvidia-uvvm
```

Beware that the major/minor numbers (e.g. 250/0 for `/dev/nvidia-uvvm` in this example) may vary depending on your system.

9.2.5. Possible problems and solutions

darktable will detect OpenCL run-time problems automatically. It will then reprocess everything on CPU; only speed is affected, the final result should not be endangered.

There can be various reasons why OpenCL could fail during initialization phase. We depend on hardware requirements and on the presence of certain drivers and libraries. In addition

all these have to fit in terms of maker model and revision number. If anything does not fit, e.g. your graphics driver (loaded as a kernel module) does not match the version of your libOpenCL.so, OpenCL support is likely not available.

In that case, the best thing to do is start darktable from a console with

```
darktable -d openc1
```

This will give additional debugging output about the initialization and use of OpenCL. First see if you find a line that starts with “[openc1_init] FINALLY ...” This should tell you, if OpenCL support is available for you or not. If initialization failed, look at the messages above for anything that reads like “could not be detected” or “could not be created”. Check if there is a hint about where it failed.

Here are a few cases observed in the past:

- darktable might tell you that no OpenCL aware graphics card is detected or that the available memory on your GPU is too low and the device is discarded. In that case you might need to buy a new card, if you really want OpenCL support.
- darktable might find your libOpenCL.so but then tell you that it couldn't get a platform. NVIDIA drivers will often give error code -1001 in that case. This happens because libOpenCL.so is only a wrapper library. For the real work further libraries – specific to vendor, device and driver – need to be loaded. This failed for some reason. There is a structure of files in /etc/OpenCL on your system that libOpenCL.so consults to find these libraries. Check if you find something fishy in there and try to fix it. Often the needed libraries cannot be found by your system's dynamic loader. Giving full path names might help.
- darktable might also tell you that a context could not be created. This often indicates a version mismatch between (loaded) graphics driver and libOpenCL. Check if you have left-over kernel modules or graphics libraries of an older install and take appropriate action. In doubt, make a clean reinstall of your graphics driver. Sometimes, immediately after a driver update, the loaded kernel driver does not match the newly installed libraries: reboot your system in that case.
- darktable might crash in very rare cases directly during startup. This can happen if your OpenCL setup is completely broken or if driver/library contains a severe bug. If you can't fix it, you can still use darktable with option “--disable-openc1”, which will skip the entire OpenCL initialization step.
- darktable might fail to compile its OpenCL source files at run-time. In that case you will get a number of error messages looking like typical compiler errors. This could indicate an incompatibility between your OpenCL implementation and our interpretation of the standard. In that case visit us at darktable-devel@sourceforge.net and report the problem. Chances are good that we can help you. Please also report if you see significant differences between CPU and GPU processing of an image!

There also exists a few on-CPU implementations of OpenCL. These come as drivers provided by INTEL or AMD. We observed that they do not give us any speed gain versus our hand-optimized CPU code. Therefore we simply discard these devices by default. This behavior can be changed by setting the configuration variable `openc1_use_cpu_devices` to TRUE.

9.2.6. Setting up OpenCL for AMD/ATI devices

While NVIDIA devices and most modern AMD/ATI devices will most often run out of the box, there is more to do for older AMD/ATI graphics cards, namely those prior to the HD7xxx series. This starts with the fact that those devices will only report to darktable

part of their total GPU memory. For a 1GB device this typically amounts to 512MB, a value which darktable in its standard configuration will refuse as not being sufficient for its tasks. Consequence: the device will not be used.

On the web you might find as a tip to set environment variable `GPU_MAX_HEAP_SIZE` to a value of 100 in this case. Indeed this will cause the AMD/ATI driver to report the full installed memory to darktable. However, there is a problem. On many (most?) cards this will cause buffers to be allocated on your computer (host) not on the video card! In this case all memory accesses will need to go through the slow PCIe bus. This will cost you a factor of 10x or more in performance and will render OpenCL useless for you, especially when exporting files.

Another environment variable which changes driver behavior is `GPU_MAX_ALLOC_PERCENT`. You could set this to 100 in order to allow memory allocations as high as 1GB on your AMD/ATI card. The problem is, this tends to cause darktable to crash sooner or later.

Our recommendation is to leave these settings untouched. Often your card will be recognized with 512MB memory and a maximum allocation size of 128MB. There are three configuration parameters which you set in file `$HOME/.config/darktable/darktable.rc` to get things running. Here are the details:

`opengl_memory_requirement`

Set this parameter to 500 so that darktable will accept your 512MB graphics memory as being sufficient in memory.

`opengl_memory_headroom`

This parameter controls how much graphics memory (out of the reported one) darktable should leave untouched for driver and display use. As for AMD/ATI devices we anyhow only can get half of the available RAM it's safe to set this to zero. So all of the 512MB can be used by darktable.

`opengl_avoid_atomics`

Atomic operations in OpenCL are a special way of data synchronization. They are only used in a few kernels. Unfortunately, some (most?) AMD/ATI devices are extremely slow in processing atomics. It's better to process the affected modules on CPU rather than accepting an ultra-slow GPU codepath. Set this parameter to TRUE if you experience slow processing of modules like *shadows and highlights*, *monochrome*, *local contrast*, or *global tonemap* or if you even get intermittent system freezes.

These recommendations do not apply to the more recent Radeon HD7xxx series with GCN architecture. Besides being very fast in terms of GPU computing they normally run out of the box. You only might consider to try some of the performance optimization options which are described in the following section.

9.2.7. OpenCL performance optimization

There are some configuration parameters in `$HOME/.config/darktable/darktable.rc` that help to finetune your system's OpenCL performance. Performance in this context mostly means the latency of darktable during interactive work, i.e. how long it takes to reprocess your pixelpipe. For a comfortable workflow it is essential to keep latency low.

In order to get profiling info you start darktable from a terminal with

```
darktable -d opengl -d perf
```

After each reprocessing of pixelpipe – caused by module parameter change, zooming, panning, etc. – you will get the total time and the time spent in each of our OpenCL kernels.

The most reliable value is the total time spent in pixelpipe. Please note that the timings given for each individual module are unreliable when running the OpenCL pixelpipe asynchronously (see `opencl_async_pixelpipe` below).

To allow for a fast pixelpipe processing with OpenCL it is essential that we keep the GPU busy. Any interrupts or a stalled data flow will add to the total processing time. This is especially important for the small image buffers we need to handle during interactive work. They can be processed quickly by a fast GPU. However, even short-term stalls of the pixelpipe will easily become a bottleneck.

On the other hand darktable's performance during file exports is more or less only governed by the speed of our algorithms and the horse-power of your GPU. Short-term stalls will not have a noticeable effect on the total time of an export.

darktable comes with default settings that should deliver a decent GPU performance on most systems. However, if you want to fiddle around a bit by yourself and try to optimize things further, here is a description of the relevant configuration parameters.

`opencl_async_pixelpipe`

This boolean flag controls how often we block the OpenCL pixelpipe and get a status on success/failure of all the kernels that have been run. For optimum latency set this to `TRUE`, so darktable runs the pixelpipe asynchronously and tries to use as few interrupts as possible. If you experience OpenCL errors like failing kernels, set the parameter to `FALSE`. darktable will then interrupt after each module so you can more easily isolate the problem. Problems have been reported with some older ATI/AMD cards, like HD57xx, which can produce garbled output if this parameter is set to `TRUE`. If in doubt, leave it at its default `FALSE`.

`opencl_number_event_handles`

Event handles are used so we can monitor success/failure of kernels and profiling info even if the pixelpipe is run asynchronously. The number of event handles is a limited resource of your OpenCL driver. For sure we can recycle them, but there is a limited number that we can use at the same time. Unfortunately, there is no way to find out what the resource limits are; so we need to guess. Our default value of 25 is quite conservative. You might want to try if higher values like 100 give better OpenCL performance. If your driver runs out of free handles you would experience failing OpenCL kernels with error code `"-5 (CL_OUT_OF_RESOURCES)"` or even crashes or system freezes; reduce the number again in that case. A value of 0 will block darktable from using any event handles. This will prevent darktable from properly monitoring the success of your OpenCL kernels but saves some driver overhead. The consequence is that any failures will likely lead to garbled output without darktable taking notice; only recommended if you know for sure that your system runs rock-solid. You can also set this parameter to `-1`, which means that darktable assumes no restriction in the number of event handles; this is not recommended.

`opencl_synch_cache`

This parameter, if set to `TRUE`, will force darktable to fetch image buffers from your GPU after each module and store them in its pixelpipe cache. This is a very resource consuming operation. It only makes sense if you have a rather slow GPU. In that case darktable might in fact save some time when module parameters have changed, as it can go back to some cached intermediate state and reprocess only part of the pixelpipe. In most cases this parameter should be set to `FALSE` (default).

`opencl_micro_nap`

In an ideal case you keep your GPU busy at 100% when reprocessing the pixelpipe. That's good. On the other hand your GPU is also needed to do regular GUI updates. It

might happen that there is no sufficient time left for this task. Consequence would be a jerky reaction of your GUI on panning, zooming or when moving sliders. darktable can add small naps into its pixelpipe processing to have the GPU catch some breath and do GUI related stuff. Parameter `opencl_micro_nap` controls the duration of these naps in microseconds. You need to experiment in order to find an optimum value for your system. Values of 0, 100, 500 and 1000 are good starting points to try. Defaults to 1000.

`opencl_use_pinned_memory`

During tiling huge amounts of memory need to be transferred between host and device. On some devices (namely AMD) direct memory transfers to and from an arbitrary host memory region may give a huge performance penalty. This is especially noticeable when exporting large images. Setting this configuration parameter to TRUE tells darktable to use a special kind of intermediate buffer for host-device data transfers. On some devices this can speed up exporting of large files by a factor of 2 to 3. NVIDIA devices and drivers seem to have a more efficient memory transfer technique even for arbitrary memory regions. As they may not show any performance gain and even may produce garbled output, `opencl_use_pinned_memory` should be left at its default FALSE for those devices.

9.2.8. Multiple OpenCL devices

While most systems will only have one OpenCL capable GPU installed, darktable is also able to make use of multiple devices in parallel. There is a configuration parameter which helps to optimize GPU priorities in that case.

It is important to understand how darktable uses OpenCL devices. Each processing sequence of an image – to convert an input to the final output using a certain history stack – is run in a so called pixelpipe. There are four different types of pixelpipe in darktable. One type is responsible to process the center image view (or full view) in darkroom mode, another pixelpipe processes the preview image (navigation window) top left in darkroom mode. Of each of these two pixelpipe there can be one at a time – with the full and the preview pixelpipe running in parallel. In addition there can be multiple parallel pixelpipes doing file exports and there can be multiple parallel pixelpipes generating thumbnails. If an OpenCL device is available darktable dynamically allocates it to one specific pixelpipe for one run and releases it afterwards.

The computational demand depends a lot on the pixelpipe type. Preview image and thumbnails have a low resolution and can be processed quickly; center image view is more demanding, let alone the pixelpipe doing a file export. If you have a reasonably fast GPU and want to get a low latency during interactive work, it is therefore important that your GPU is allocated to do the more demanding center image (full) pixelpipe, while the smaller preview image can be processed in parallel by the CPU. Older versions of darktable would therefore not allow the preview pixelpipe to grab any OpenCL device.

Starting with darktable 1.2 there is a more flexible scheme to allocate and prioritize your OpenCL device(s). Configuration parameter `opencl_device_priority` holds a string with the following structure:

```
a,b,c.../k,l,m.../o,p,q.../x,y,z...
```

Each letter represents one specific OpenCL device. There are four fields in the parameter string separated by a slash, each representing one type of pixelpipe. `“a,b,c...”` defines the devices that are allowed to process the center image (full) pixelpipe. Likewise devices `“k,l,m...”` can process the preview pixelpipe, devices `“o,p,q...”` the export pixelpipes and finally devices `“x,y,z...”` the thumbnail pixelpipes. An empty field means that no OpenCL device may serve this type of pixelpipe.

darktable has an internal numbering system, where the first available OpenCL device will receive number "0". All further devices are numbered consecutively. This number together with the device name is displayed when you start darktable with "darktable -d opencl". You can specify a device either by number or by name (upper/lower case and whitespace do not matter). If you have more than one device – all with the same name – you need to use the device numbers in order to differentiate them.

A device specifier can be preceded by an exclamation mark "!", in which case the device is excluded from processing this pixelpipe. You can also give an asterisk "*" as a wildcard, representing all devices not mentioned explicitly before in that group.

Sequence order within a group matters. darktable will read the list from left to right and whenever it tries to allocate an OpenCL device to a pixelpipe it will scan the devices in that order, taking the first free device it finds.

darktable's default setting for `opencl_device_priority` is:

```
*/!0,*/*/*
```

Any detected OpenCL device is allowed to process our center view image. The first OpenCL device (0) is not allowed to process the preview pixelpipe. As a consequence, if there is only one GPU owned by your system, preview pixelpipe will always be processed on CPU, keeping your single GPU exclusively for the more demanding center image view. This is reasonable and identical to the old behavior. No restrictions apply to export and thumbnail pixelpipes.

The default is a good choice if you have only one device. If you have several devices it forms a reasonable starting point. However, as your devices might have quite different levels of processing power, it makes sense to invest a few thoughts and optimize your priority list.

Here is an example. Let's assume we have a system with two devices, a fast Radeon HD7950 and an older and slower GeForce GTS450. darktable (started with "darktable -d opencl") will report the following devices:

```
[opencl_init] successfully initialized.
[opencl_init] here are the internal numbers and names of
                OpenCL devices available to darktable:
[opencl_init]          0          'GeForce GTS 450'
[opencl_init]          1          'Tahiti'
[opencl_init] FINALLY: opencl is AVAILABLE on this system.
```

So the GeForce GTS 450 is detected as the first device; the Radeon HD7950 ('Tahiti') as the second one. This order will normally not change unless the hardware or driver configuration is modified. But it's better to use device names rather than numbers to be on the safe side.

As the GTS450 is slower than the HD7950, an optimized `opencl_device_priority` could look like:

```
!GeForce GTS450,*/!Tahiti,*/Tahiti,*/Tahiti,*
```

The GTS450 is explicitly excluded from doing the center image pixelpipe; this is reserved to "all" other devices (i.e. the HD7950/Tahiti). Completely the opposite for our preview pixelpipe. Here the Tahiti is excluded, so that only the GTS450 will be allowed to do the work.

For file export and thumbnail generation we want all hands on deck. However, darktable should first look if device Tahiti is free, because it's faster. If that's not the case, all other devices – in fact only the GTS450 – are checked.

9.2.9. OpenCL still does not run for me!

As has been said before OpenCL systems come with a huge variety of setups: different GPU manufacturers, different GPU models, varying amounts of GPU memory, different drivers, different distributions etc. Many of the potential problems will only appear with a very specific combination of those factors.

As we developers of darktable on our computers only have access to a small fraction of those variations, please understand that we might not be able to fix your specific problem. There is not much we can do, if there is no way for us to reproduce.

If nothing else helps, the best option might be to start darktable with

```
darktable --disable-opengl
```

In the end there is nothing in darktable which only runs on GPU. Don't let OpenCL discourage you; also darktable's CPU code is highly optimized for performance!

Index

A

artifact mitigation

- banding, 49, 97, 105, 114
- black pixels, 56, 75, 85, 88
- blue light sources, 56, 75, 85, 88
- halos, 67, 90
- magenta highlights, 72

B

- banding, 97
- base curve, 69
- basic workflow, 8
- blending, 43
- blending operators, 45
- bloom, 111
- brightness, 71
 - (see also levels)
 - (see also tone curve)
- brush, 48

C

- camera import, 22
- channel mixer, 81
- chromatic aberration, 99
 - (see also lens correction)
- circle, 48
- clarity (see equalizer)
- cloning, 96
- collect images, 24
- collections, 16, 24
- color balance, 87
- color contrast, 84
- color correction, 85
- color labels, 17
- color management, 54
 - display profile, 84
 - gamut check, 84
 - input color profile, 88
 - output color profile, 83
 - softproof, 84
 - unbreak input profile, 89
- color mapping, 112
- color picker, 58
- color spaces, 55
- color zones, 86
- colorize, 112
- comboboxes, 41
- command line parameters, 3
- conditional blending, 50
- contrast, 71
 - (see also levels)

- (see also tone curve)
- copy images, 27
- create images, 28
- crop and rotate, 65
- cropping an image, 65

D

- darkroom, 37
- darkroom panels, 57
 - bottom panel, 63
 - color picker, 58
 - filmstrip, 64
 - histogram, 61
 - history stack, 57
 - mask manager, 59
 - module groups, 61
 - more modules, 62
 - navigation, 57
 - overexposed warning, 63
 - snapshots, 57
 - underexposed warning, 63
- darktable, vii
- darktable-cli, 4
- defringe, 100
- delete images, 27
- demosaic, 71
- denoise
 - bilateral, 93
 - equalizer, 90
 - non local means, 93
 - profiled, 92
 - raw, 97
- display profile, 84
- dithering, 97

E

- ellipse, 48
- equalizer, 90
- EXIF data, 26
- export, 33
- exposure, 69

F

- file export, 33
- file import, 22
- fill light, 74
- film rolls, 16
- filmstrip, 7
- filtering, 18
- focus detection, 14
- framing, 102

G

- gamut clipping, 56, 88

- geo tagging, 30
- global tonemap, 80
- GPU computing, 219
- gradient, 49
- graduated density, 114
- grain, 107
- group images, 18, 28

H

- HDR images, 24, 27
- highlight reconstruction, 72
- highpass, 108
- histogram, 61
- history stack, 28, 39, 57
- hotpixels, 99

I

- image information, 26
- import, 22
 - HDR files, 24
 - LDR files, 24
 - RAW files, 24
- input color profile, 88
- invert, 74

L

- lens correction, 94
- levels, 75
- lighttable panels
 - collect images, 24
 - export, 33
 - geo tagging, 30
 - history stack, 28
 - image information, 26
 - import, 22
 - metadata editor, 32
 - recently used collections, 26
 - selected images, 27
 - selecting images, 26
 - styles, 29
 - tagging, 32
- lighttable view, 13
- local adjustments, 47
- local contrast, 79
 - (see also equalizer)
- local copies, 21, 28
- lowlight, 110
- lowpass, 108
- Lua, 151
- Lua API, 160
 - #, 164, 170, 175, 182, 185, 202, 203, 204
 - action_images, 166
 - api_version_major, 178
 - api_version_minor, 178

- api_version_patch, 178
- api_version_string, 179
- api_version_suffix, 178
- apply, 184, 203
- apply_style, 188
- attach, 176, 204
- attach_tag, 188
- attributes, 213
- backgroundjobs, 175
- blue, 192
- bpp, 196, 196, 198
- cache_dir, 178
- callback, 209, 210, 211, 211, 212, 212, 213
- camera, 175
- check_version, 179
- collect, 173
- colorlabels, 173
- colorpicker, 174
- compression, 197
- comp_type, 198
- comp_type_t, 208
- configuration, 177
- config_dir, 178
- copy, 189
- copy_history, 173
- copy_image, 186, 202
- coroutine, 213
- create, 176, 182
- create_job, 167
- create_style, 188
- creator, 190
- current_view, 167
- darkroom, 169
- darktable, 160
- darktable_label, 171
- database, 185
- debug, 187, 187
- delete, 165, 176, 183, 187, 193, 202, 203, 204
- description, 190, 203
- detach, 177, 204
- detach_tag, 188
- direction, 170
- drop_cache, 194
- dt_imageio_exr_compression_t, 209
- dt_imageio_j2k_format_t, 207
- dt_imageio_j2k_preset_t, 208
- dt_imageio_module_format_data_copy, 197
- dt_imageio_module_format_data_exr, 196
- dt_imageio_module_format_data_j2k, 198
- dt_imageio_module_format_data_jpeg, 197
- dt_imageio_module_format_data_pfm, 197
- dt_imageio_module_format_data_png, 196
- dt_imageio_module_format_data_ppm, 197
- dt_imageio_module_format_data_tiff, 196
- dt_imageio_module_format_data_webp, 197

dt_imageio_module_format_t, 195
 dt_imageio_module_storage_data_disk, 201
 dt_imageio_module_storage_data_email, 200
 dt_imageio_module_storage_data_facebook, 200
 dt_imageio_module_storage_data_flickr, 200
 dt_imageio_module_storage_data_gallery, 201
 dt_imageio_module_storage_data_latex, 200
 dt_imageio_module_storage_data_picasa, 201
 dt_imageio_module_storage_t, 199
 dt_lib_module_t, 204
 dt_lua_backgroundjob_t, 206
 dt_lua_film_t, 202
 dt_lua_image_t, 188
 dt_lua_snapshot_t, 206
 dt_lua_tag_t, 204
 dt_style_item_t, 203
 dt_style_t, 203
 dt_view_t, 205
 dump, 187
 duplicate, 183, 185, 189, 203
 duplicate_index, 189
 events, 209
 exif_aperture, 190
 exif_crop, 191
 exif_datetime_taken, 191
 exif_exposure, 191
 exif_focal_length, 191
 exif_focus_distance, 191
 exif_iso, 191
 exif_lens, 190
 exif_maker, 190
 exif_model, 190
 expandable, 205
 expanded, 205
 export, 174, 184, 203
 extension, 195
 extra registration parameters, 210, 210, 211, 212, 212, 213, 213
 filename, 189, 201, 201, 202, 206
 film, 189
 films, 164
 filmstrip, 171
 filter, 172
 find, 176
 format, 198
 geotagging, 172
 get_group_members, 194
 get_tags, 177, 188
 global_toolbox, 172
 global_toolbox-grouping_toggle, 212
 global_toolbox-overlay_toggle, 213
 green, 192
 grouping, 172
 group_leader, 194
 group_with, 193
 gui, 166
 has_gui, 178
 has_tostring, 213
 height, 192, 199
 hint, 198
 hinter, 171
 hint_t, 207
 histogram, 174
 history, 174
 hovered, 166
 id, 189, 202, 204, 206
 image, 173
 implicit_yield, 213
 import, 172, 184, 185
 intermediate-export-image, 209
 is_hdr, 192
 is_ldr, 192
 is_raw, 192
 known, 188
 latitude, 168, 191
 libs, 169
 lighttable, 169
 lighttable_mode, 173
 live_view, 175
 local_copy, 194
 location, 175
 longitude, 168, 191
 lua_pref_type, 208
 make_group_leader, 193
 map, 168
 map_settings, 175
 masks, 174
 max_depth, 187
 max_height, 195
 max_snapshot, 170
 max_width, 195
 metadata, 171
 metadata_view, 170
 mime, 195
 modulegroups, 173
 modulelist, 171
 module_toolbox, 174
 move, 189
 move_image, 186, 202
 name, 195, 199, 203, 204, 204, 204, 206, 207
 navigation, 174
 new, 164
 new_format, 165
 new_storage, 166
 num, 204
 on_screen, 205
 parent, 213

- path, 189, 202
- percent, 206
- plugin_name, 195, 199
- post-import-film, 211
- post-import-image, 210
- preferences, 179
- preset, 199
- print, 160
- print_error, 160
- publisher, 189
- purple, 193
- quality, 197, 198, 198
- rating, 192
- ratings, 172
- ratio, 170
- read, 181
- recentcollect, 172
- recommended_height, 199
- recommended_width, 199
- red, 192
- register, 179
- register_event, 160
- register_storage, 161
- reset, 193, 205
- rights, 190
- select, 173, 206
- selected, 170
- selection, 167
- session, 174
- shortcut, 211
- show_overlays, 172
- slideshow, 169
- snapshots, 169
- snapshot_direction_t, 207
- styles, 170, 182
- supports_format, 200
- system, 213
- tagging, 171
- tags, 175
- take_snapshot, 170
- tethering, 169
- title, 189, 201, 201
- tmp_dir, 178
- type, 188
- types, 188
- valid, 206
- verbose, 178
- version, 178, 205
- view-changed, 212
- views, 168
- viewswitcher, 171
- view_toolbox, 175
- visible, 205
- width, 192, 199
- write, 181, 213

- write_image, 195
- yellow, 192
- yield, 213
- yield_type, 208
- zoom, 168

M

- map view, 129
 - find location, 131
 - map settings, 131
- mask manager, 59
- mask shapes, 48
 - brush, 48
 - circle, 48
 - ellipse, 48
 - gradient, 49
 - path, 49
- masks, 47
 - combined masks, 53
 - drawn masks, 47
 - parametric masks, 50
- memory setup, 216
 - 32-bit systems, 217
 - 64-bit systems, 218
- metadata editor, 32
- module groups, 61
- module presets, 41
- module processing order, 39
- module usage, 39
 - comboboxes, 41
 - sliders, 40
- modules, 65
 - base curve, 69
 - bloom, 111
 - channel mixer, 81
 - chromatic aberration, 99
 - color balance, 86
 - color contrast, 84
 - color correction, 84
 - color mapping, 112
 - color zones, 86
 - colorize, 112
 - contrast brightness saturation, 70
 - crop and rotate, 65
 - defringe, 100
 - demosaic, 71
 - denoise – bilateral, 93
 - denoise – non local means, 93
 - denoise – profiled, 92
 - dithering, 97
 - equalizer, 90
 - exposure, 69
 - fill light, 74
 - framing, 102
 - global tonemap, 80

- graduated density, 114
- grain, 107
- highlight reconstruction, 72
- highpass, 108
- hotpixels, 99
- input color profile, 88
- invert, 74
- lens correction, 94
- levels, 75
- local contrast, 79
- lowlight, 110
- lowpass, 108
- monochrome, 85
- orientation, 66
- output color profile, 82
- raw denoise, 97
- rotate pixels, 96
- scale pixels, 95
- shadows and highlights, 67
- sharpen, 90
- soften, 106
- splittoning, 104
- spot removal, 96
- tone curve, 76
- tonemapping, 79
- unbreak input profile, 89
- velvia, 81
- vibrance, 88
- vignetting, 105
- watermark, 101
- white balance, 73
- zone system, 78
- monitor profile, 84
- monochrome, 85
- more modules, 62
- move images, 27
- multiple instances, 42

N

- navigation, 57
- noise removal (see denoise)

O

- OpenCL, 219
- orientation, 67
- output color profile, 83
- overexposed warning, 63
- overlay, 14

P

- panoramas, 218
- path, 49
- pixelpipe, 39
- Preface, vii

- preferences and settings, 137
 - core options, 141
 - GUI options, 138
 - presets, 148
 - session options, 143
 - shortcuts, 145
- presets, 148
 - module presets, 41
- program invocation, 3
- purple fringing, 100

R

- recently used collections, 26
- red-eye removal, 120
- remove images, 27
- rotate pixels, 96
- rotating an image, 65

S

- saturation, 71
 - (see also color contrast)
 - (see also color zones)
 - (see also tone curve)
- save button, 8, 19
- scale pixels, 95
- scripting, 151
- selected images, 27
- selecting images, 26
- shadows and highlights, 67
- shapes (see mask shapes)
- sharpen, 90
- sidecar files, 19, 29
- skulls, 17
- sliders, 40
- slideshow view, 133
- snapshots, 57
- soften, 106
- sort order, 18
- splittoning, 104
- spot removal, 96
- star ratings, 17
- stitched panoramas, 218
- styles, 30

T

- tags, 32
- tethering view, 123
 - camera settings, 125
 - live view, 125
 - sessions, 125
- thumbnails, 16
- tiling, 217
- tone curve, 76
- tonemapping, 79

(see also global tonemap)

U

unbounded colors, 55
unbreak input profile, 89
underexposed warning, 63

V

velvia, 81
vibrance, 88
vignetting, 105

W

watermark, 101
white balance, 73

X

XMP files, 19, 29

Z

zone system, 78
zoom, 38, 57