darktable 3.5 (dev) user manual

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 and enhance your raw images.

The source repository for this documentation may be found at https://github.com/darktable-org/dtdocs.git . Any feedback relating to this documentation can be provided by creating a ticket or a pull request against this repository.

1. Overview

1.1. user interface

1.1.1. views

There are five different views available in darktable:

lighttable
Manage images and collections.

darkroom
Develop a single image.

map
Show images with a geo-tag data on a map and manually geo-tag new images.

print
Send images to your printer.

slideshow
Display images as a slideshow, processing them on-the-fly.

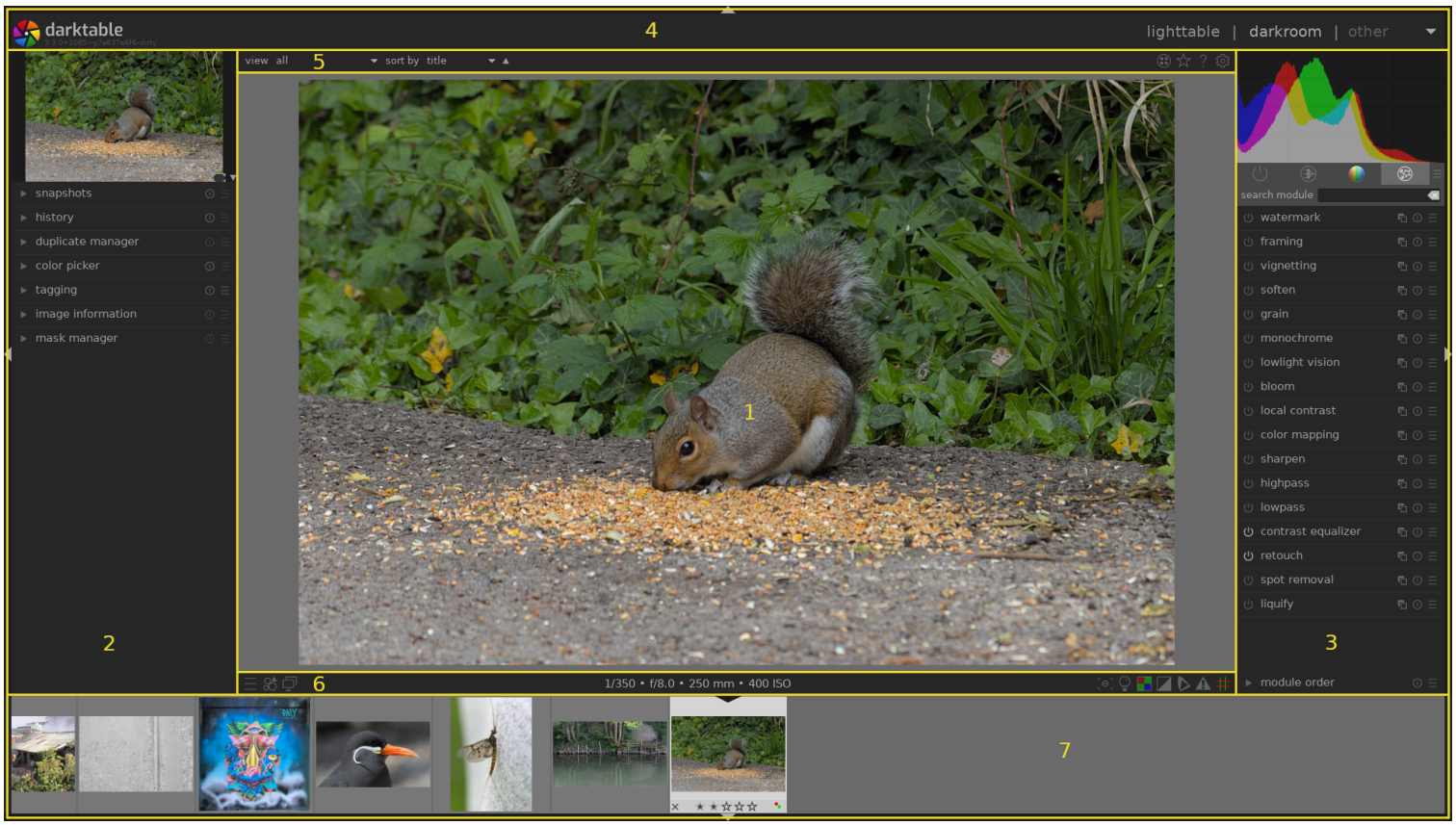
tethering
Remotely capture and save images taken with a connected camera.

You can switch between views by clicking the view name at the top of the right panel (the currently active view is highlighted) or by using one of the following keyboard shortcuts:

```
L       switch to lighttable
D       switch to darkroom
M       switch to map
P       switch to print
S       switch to slideshow
T       switch to tethering
```

1.1.2. screen layout

The layout of all views is similar and consists of a center area with panels at the edges:



1. center area : Contains information and functionality specific to the current view.
2. left panel : Contains modules primarily used to provide information.
3. right panel : Contains modules primarily used for image processing.
4. top banner : Contains information about the current darktable version and allows you to switch between views. Also used by some modules to show hints and messages.
5. top panel : Provides access to global settings and shortcuts
6. bottom panel : Provides access to view-specific settings and shortcuts.
7. filmstrip / timeline panel : An optional panel that can be enabled at the bottom of the screen to display a timeline (in the lighttable view) or a filmstrip (in other views) of images in the current collection.

panel size and visibility

The left, right and filmstrip/timeline panels can be resized by dragging their inner borders.

Each of the panels can be expanded or collapsed by pressing a triangle located at the outside edge of the panel. Panel visibility can also be adjusted using keyboard shortcuts, as follows:

```
TAB             Temporarily expand the centre view to fill the whole window.
                Press again to return to the previous view.
F11             Toggle fullscreen mode
Shift+Ctrl+t    Toggle the top panel (between the image and the top banner)
Shift+Ctrl+b    Toggle the bottom panel (beween the image and the filmstrip/timeline if shown)
```

```
Shift+Ctrl+l    Toggle the left panel
Shift+Ctrl+r    Toggle the right panel
Ctrl+f          Toggle the filmstrip/timeline
Ctrl+h          Toggle the top banner
b               Toggle all borders and panel-collapse controls
```

Note: Size and visibility of panels are stored independently for each view.

1.1.3. filmstrip

The filmstrip, when enabled, is shown at the bottom of the screen (except in the lighttable view, where it is replaced with the timeline module) and displays the images from the collection that is currently selected in the lighttable view. 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 the lighttable view. For example, while developing an image in darkroom mode, you can switch to another image to by clicking its thumbnail in the filmstrip. You can also rate and color-classify the images as you do in lighttable, as well as copy & paste the history stack using keyboard shortcuts.

See the filmstrip module documentation for more information.

1.1.4. top panel

The top panel is common to all darktable views and provides a number of common utility functions.

On the left-hand-side

view
Choose which images to view in the lighttable/filmstrip, based on star rating and reject status.

sort by
Choose the property to sort by from the dropdown.

sort order
Switch the sort order (ascending / descending) with the arrow-toggle.

On the right-hand-side

grouping
Expand or collapse grouped images

thumbnail overlays
Define what information is displayed over thumbnails in the lighttable/filmstrip.
You can define different settings depending on the thumbnail size in preferences > lighttable

context-sensitive help
Click on this icon and then click on a control element to be directed to the appropriate online help page.

preferences
Open the preferences & settings dialog

1.1.5. keyboard shortcuts

Much of the functionality in darktable can be controlled via keyboard shortcuts which can be customised in preferences > shortcuts .

Press the H key (for help) in any darktable view to show a list of all shortcuts that are applicable to the current view.

1.2. supported file formats

darktable 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, ARI, ARW, BAY, BMQ, CAP, CINE, CR2, CRW, CS1, DC2, DCR, DNG, GPR, 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 the open source library RawSpeed , originally developed by Klaus Post and now maintained as part of the darktable project. The number of supported cameras and file formats is constantly increasing. Most modern camera models are supported, and new ones tend to get added very quickly. It is beyond the scope of this manual to give an exhaustive list.

With the exception of Fujifilm X-Trans cameras, darktable does not decode images from cameras with non-Bayer sensors (e.g. Sigma cameras with the Foveon X3 sensor).

other image files

darktable natively reads “ordinary” images in JPEG, 8-bit/16-bit PNG and 8-bit/16-bit TIFF format, as well as 16-bit/32-bit floating point TIFF formats. JPEG2000 is also supported if the required libraries are present at compile time. Similarly, if darktable was compiled with GraphicsMagick support, there are further supported formats, such as GIF, Dicom DCM, additional “exotic” TIFF formats, and some of Sun’s “portable xyz-map” family.

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

1.3. sidecar files & non-destructive editing

1.3.1. sidecar files

darktable is a non-destructive image editor and opens all images in read-only mode. Any data created within darktable (metadata, tags, and image processing steps) are stored in separate `.XMP` *sidecar* files. These files allow darktable 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 is automatically generated (XMP generation can be disabled in preferences > storage but this is not recommended in normal use).

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 processing steps. Each duplicate is represented by a separate XMP sidecar file (with a filename constructed in the form `<basename>_nn.<extension>.xmp`, where nn represents the 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 each duplicate is displayed in the image information module in each of darktable's views.

Sidecar files are automatically synchronized with your work without the need to press a "save" button. When backing up your data, make sure you also retain 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 library database for fast access. An image can only be viewed and edited from within darktable if its data is loaded in the library database. This happens automatically when you first import an image. If an image is subsequently re-imported, the database will be updated from the data in the XMP file.

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 – such changes will be overwritten the next time darktable synchronizes the file. On request darktable can be configured to search for updated XMP files at startup, offering a choice to update the database or overwrite the XMP file where changes are identified. This configuration can also be changed in preferences > storage .

1.3.2. importing sidecar files generated by other applications

When importing an image, darktable automatically checks if it is accompanied by a sidecar file. As well as looking for files named <basename>.<extension>.xmp and <basename>_nn.<extension>.xmp (darktable's XMP file naming formats) darktable also checks for the presence of a file in the form <basename>.xmp (the naming format for Lighroom's XMP sidecar files). Files with the latter naming format will be read by darktable but will not be written to. Once the image has been imported, darktable will generate an additional XMP file using its own naming convention.

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

- tags (including 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 here is not to make darktable a drop-in replacement for any other software, but rather to help you recover part of the work you have invested into your image on migration to darktable. It is 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, the results may be similar but often, further adjustment will be required in darktable.

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

At present, darktable is able to handle 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 (*vignetting*)
- clarity (*local contrast*)
- tone curve (*tone curve*)
- HSL (*color zones*)
- split-toning (*split-toning*)
- grain (*grain*)
- spot removal (*spot removal*)

1.3.3. local copies

Many users have huge image collections stored on extra hard drives in their desktop computer, or on an external storage medium like a RAID NAS, etc.

It is a common requirement to be able to develop a number of images while travelling using a laptop and then later synchronize them back to the original storage medium. However, copying images manually from the main storage to the laptop and back is cumbersome and prone to errors. The "local copies" feature of darktable has been designed to directly support these use cases.

You can create local copies of selected images from within the lighttable. Local copies are always used when present, giving continued access to images even if the external storage is no longer connected. At a later point, when your primary storage medium has been reconnected, you can synchronize the XMP sidecar files back to this storage, deleting any local copies. These operations can be found in the selected images module in the lighttable.

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

Local copies are stored within the `$HOME/.cache/darktable` directory and named `img-<SIGNATURE>.<EXT>` (where `SIGNATURE` is a hash signature (MD5) of the full pathname, and `EXT` is the original filename extension).

Local copies can be identified in the lighttable view by a white marker on the top right of the thumbnail. In addition, all local copies carry the `darktable|local-copy` tag to allow them to be easily selected.

1.4. basic workflow

1.4.1. import, rate & tag images

When you want to edit some new images in darktable, the first step is to import them. This will create entries for the imported images in darktable's library database so it can keep track of the changes you make to them. There are two main methods for importing images:

import images from the filesystem

You can import a single image or a directory full of images (optionally recursing through subdirectories) from the filesystem. When importing images, darktable will read the image's internal metadata and any accompanying XMP sidecar file . If an image has already been imported, it will be ignored (though any updates to the sidecar file will be loaded). The location of each image is recorded in the library database, but darktable will not copy or move the files anywhere. If you want a program that will copy files into a specific directory, you can use a separate program like rapid photo downloader for this.

import images from a camera

To import images from a camera, first connect the camera to your system with a USB cable. If your system tries to automount the camera's files, you should choose to abort the mount operation, otherwise the camera cannot be accessed from within darktable. If you don't see your camera listed in the import module, press the "scan for devices" button. Once your camera is detected the import module should offer the ability to *import* images or *tether* your camera while shooting. Unlike when importing of images from the filesystem, darktable will physically copy files imported from the camera into a specified directory following the file naming pattern defined in preferences > import .

Once images are imported, they will appear in the lighttable view. By default, the images will all be given a one-star rating.

There are many different ways to manage a set of newly imported photos, such as giving them tags and adjusting their ratings. Please refer to the lighttable section of this guide for a full list of *digital asset management* features.

One example workflow might be:

1. Set the lighttable view to show photos with exactly a 1 star rating.
2. Perform a quick first-level screening of your photos. If any photos are badly out-of-focus or otherwise useless, *reject* them with the R key, or give them a 0-star rating. If a photo looks reasonable and should pass to the next phase, press 2 to give it a 2 star rating. Any photos that no longer have a 1 star rating will automatically disappear from view. Continue in this manner until you have completed the first level of assessment.
3. Now set the lighttable view to show only photos with exactly a 2 star rating. Go through these photos more carefully, and decide whether to promote them to a 3 star rating, or put them back down to a 1 star or rejected rating.
4. You can now spend some time performing a quick edit on your 3 star photos, to see if they are worth keeping. If you are happy with the results, you can create a tag for the photo, and promote it to a 4 or even 5 star rating.
5. Go through your 4 and 5 star photos, perform any final edits on them, print them out, publish on your portfolio site, etc. and bask in the copious amounts of critical acclaim you will receive!
6. If space is at a premium you might want to consider permanently deleting your rejected or 0-star images. Select these images in the lighttable and use the 'trash' option in the selected images module. You should probably only do this on photos you are certain you will never need again (badly focussed, significantly overexposed etc.).

1.4.2. editing an image: workflow overview

This section will guide you through the basics of developing an image in the darkroom view, where an arsenal of modules is at hand to help you reach your creative goals.

To begin, open an image in the darkroom by double clicking an image thumbnail in the lighttable view.

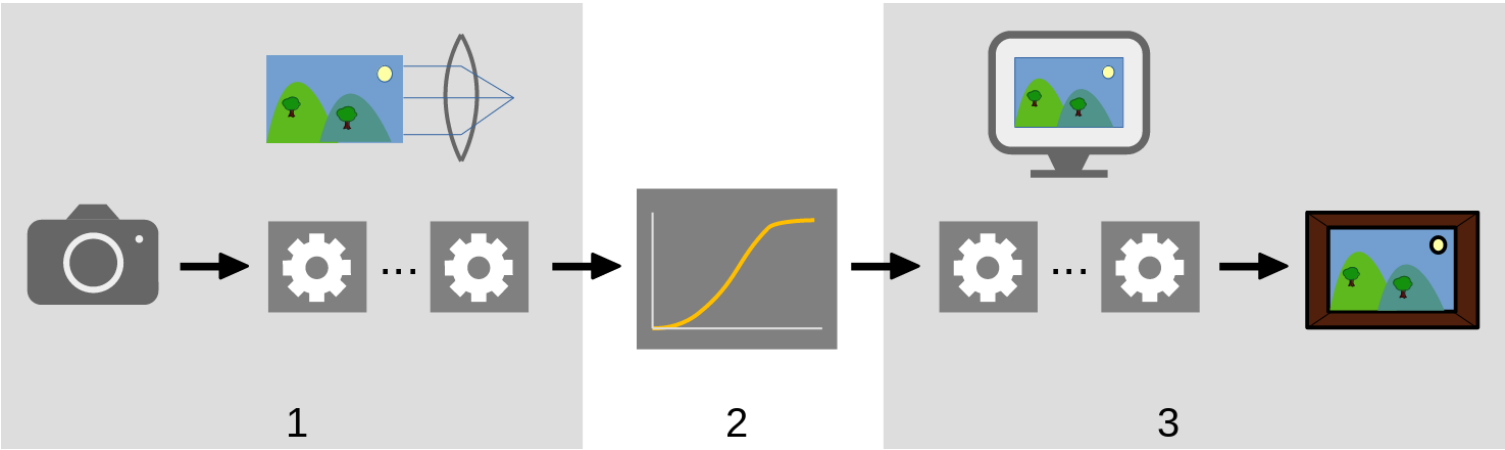
Each change you make to the image in the darkroom is turned into a history stack item. The history is stored in a database and in an XMP sidecar file for that image. All changes are stored automatically when you switch images or go from one darktable view to another. You can safely leave darkroom mode or quit darktable at any time and come back later to continue your work. For this reason darktable does not need a “save” button and it does not have one.

On the left panel in darkroom mode is the history stack module, listing the changes you have made, starting from the bottom – each edit adds a new item to the top of the stack. You can select an earlier point in this history to show how the image looked at that point, for comparison between changes. The stack can be compressed to remove redundant intermediate points in your edits – when you are happy with what you have done, just compress the history stack. Note that all edits above the selected history entry are permanently deleted when compressing the history stack.

A large number of processing modules are shipped with darktable, arranged into module groups . These groups are accessed via toggle buttons at the top of the right panel, just below the histogram .

choosing a workflow

When processing an image, we apply a sequence of modules, known as the pixelpipe .



1. The *scene-referred* modules are intended to process pixel values that are proportional to the amount of light collected by the camera at the scene. The dynamic range of an image in the scene-referred section of the pixelpipe is often larger than that of the display medium.
2. At some point in the pixelpipe, these pixel values are compressed by a tone mapping module into a smaller dynamic range more suitable for display on a monitor or hardcopy print.
3. The remaining modules operate in this non-linear *display-referred* section of the pixelpipe to produce the final output image.

There are two standard workflows offered in darktable (these can be changed in preferences > processing > auto-apply pixel workflow defaults):

- *scene-referred workflow* : Here the emphasis is on doing as much processing as possible in the scene-referred part of the pipeline, and performing dynamic range compression to display-referred space as late as possible. This is the preferred workflow for darktable. It uses the *filmic rgb* module to perform the tone mapping compression.
- *display-referred workflow* : This is the legacy workflow and is still the default setting when darktable is first installed. It performs the tone mapping compression much earlier in the pixelpipe, and many of the modules therefore operate in display-referred space. It uses the *base curve* module to perform the tone mapping compression.

A third option is to set the workflow presets option to *none*. In this case, the scene-referred workflow module order will be used by default but none of the above modules will be automatically applied. It is up to the user to arrange for appropriate tone mapping and reorder the modules where required.

Note: when changing the preferences between scene-referred and display-referred workflow, the new setting will only apply for newly imported images. If you have already imported an image using a different workflow setting, go to the *history stack* module in the darkroom view, select “original image”, and click “compress history stack”. This will discard any previous edits and reset the workflow for that image.

1.4.3. editing an image: scene-referred workflow

The *scene-referred* workflow places an emphasis on performing image processing in the linear scene-referred part of the pixelpipe. This helps to reduce artifacts and color shifts that can result from processing non-linear pixel values and, by decoupling the image processing from the characteristics of a specific display, it makes it easier to adapt your work in the future to new display media, such as high dynamic range displays.

This being the recommended way to process images in versions 3.0 and above, this section will provide a much more comprehensive overview than the next section on the *display-referred* workflow.

basic steps

Basic image processing in scene-referred workflow requires you, as a minimum, to consider the following steps in order to render a reasonable image on your display:

1. Capture an image

   Use your camera to take a properly exposed image. Normally you can rely on the camera’s metering and automatic exposure features. However, for some scenes you may need to use the camera’s exposure compensation dial or manual settings to get an optimal exposure. In general, you want to make the exposure in camera as bright as possible without clipping the highlights. This is known as “exposing to the right” (ETTR), and it ensures you take best advantage of the sensor’s dynamic range. Many cameras have features like “zebras” or “blinkies” to warn you when you are in danger of clipping.

2. Adjust mid-tones using the *exposure* module

   Use the exposure slider in the *exposure* module to adjust the mid-tones in the image to an appropriate brightness level. At this stage, don’t worry about highlights and shadows – these will be handled later.

   The *exposure* module is enabled by default, and will include an initial exposure boost of +0.5EV to mimic the standard processing of most in-camera JPEGs. The metering systems in cameras vary, and some camera models might need a slightly larger exposure boost (eg. +0.8EV ~ 1.5EV), in which case you can create an auto-apply preset as required. The exposure module will detect if the camera’s exposure compensation dial was used (see above remarks about ETTR), and will re-adjust the exposure accordingly.

   You *can* tweak the black level in the *exposure* module to supply more contrast, but you need to be very careful doing this as you can end up with negative RGB values, which can cause the *filmic rgb* module to malfunction.

3. Adjust the *white balance*

   It is important that the white balance is set correctly to form a solid basis for subsequent processing. The camera will normally store the selected white balance setting inside the raw file’s metadata, and darktable will use this as a starting point. To get a more accurate white balance, you can either use the color picker to select a neutral gray tone in the image, or you can switch to a different white balance preset from your camera, where available. Fine adjustments to the global white balance are made using the *temperature* slider and, less often, the *tint* slider. Moving the *temperature* slider to the left makes the image cooler (more blue), and moving it to the right makes it warmer (more orange).

   The *white balance* module is only able to make *global* adjustments to the white balance of the image. The *color balance* module, among other things, gives you even more control in cases where a scene was illuminated by multiple light sources at different color temperatures.

4. Adjust the white and black points using the *filmic rgb* module

   This module performs tone mapping compression from the high-dynamic-range of the captured image, to the lower dynamic range of the display medium. The mid-gray tone level has already been set (above) with the *exposure* module. Filmic will propose, on its *scene* tab, an appropriate white point and black point for the image – you may need to adjust these for a particular scene. On the *look* tab you can adjust the midtone contrast and saturation settings if required.

other recommended modules

In addition to the basic modules described above, you may want to consider using the following modules to make your image look even prettier. These modules are known to work well with the scene-referred workflow:

crop and rotate* / *perspective correction

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 may be perspective distortions. All of this can be corrected with full manual control in the *crop and rotate* module. For a fully automatic correction of perspective distortions you may alternatively visit the *perspective correction* module.

retouch* / *spot removal* / *hot pixels

Sometimes you will need to remove spots caused by sensor dirt. The new *retouch* and the older *spot removal* modules are at hand for this 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, take a look at the *hot pixels* module for automatic correction.

color balance

This is a versatile module that can be used to further adjust the contrast and saturation of an image, and can also be used to perform color grading (e.g. emulate “orange and teal” grading used in hollywood films, remove redness in skin tones, adjust for uneven color balance in shadows/midtones/highlights, etc.). The *color zones* module can also be helpful in some cases where you are unable to achieve the desired effect using the *color balance* module.

tone equalizer

Use this module to perform “dodging and burning” operations and recover detail in the shadows and highlights. This module generates a mask to average out the luminance in different parts of the image, and the equalizer allows you to selectively increase and decrease luminance levels using that mask. It is recommended that you first check the mask is set up appropriately, then you can use the sliders or the spline curve to adjust the various brightness levels. You can also place the mouse cursor over different parts of the image to see the EV level of the mask at that point, and then use the mouse wheel to adjust the brightness of that EV level accordingly.

local contrast

This module can emphasise detail and improve clarity, and is a good way to improve the general sharpness of an image. It is recommended that you use this module in *local laplacian* mode.

A more versatile but also more complex technique is to use the *contrast equalizer* module, which is very useful for making adjustments where spatial dimension plays a role. It has a number of pre-defined presets that may be helpful as a starting point in understanding this module.

denoise (profiled)

The *denoise (profiled)* module is usually your best option for reducing noise in an image. This module offers an almost “single-click” solution to remove 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 well above 300 popular camera models. The simplest way to use this module is *non-local means (auto)* mode. The wavelet feature of this module is also quite effective against color noise. It is recommended that you use this module at 100% zoom so that you can accurately see the effects of your changes.

Other modules that allow for image denoising include *raw denoise* , *surface blur* , *astrophoto denoise* , and the *contrast equalizer* module, which is based on wavelets. If your camera is not yet supported by *denoise (profiled)*, *astrophoto denoise* is probably the most convenient alternative, as it allows you to treat color and luminance noise separately.

haze removal

Does what is says on the tin – removes atmospheric haze.

color calibration

This module offers a range of presets for making black and white images emulating classic film. It can also be used to tweak your color profile matrices, for example, to deal with color gamut issues.

lens correction

If your camera/lens combination is supported, use this module to correct for standard lens distortions, where corrections have not already been performed in-camera. The *crop and rotate* or *perspective correction* modules can also be used to simulate the effects of a tilt-shift lens.

modules to be used with care

There are some modules for which there is not yet an alternative that is well-suited to the scene-referred workflow. If required, these modules should be used sparingly and with care.

vibrance

Tends to darken colors. Consider using *color zones* with a saturation parametric mask to give more control.

color zones

Transitions may not be graceful. An alternative can be to use *color balance* with a parameteric mask.

vignetting

This module can produce unnatural-looking results with too strong a fall-off. You may be better off using the *exposure* module with an elliptical mask with large transition area, and perhaps adding *color balance* with the same mask to reduce saturation at the edges.

Note: When using blend modes on any module, you should be aware that many of the blend modes are optimized for display-referred space and assume a mid-gray value of 50%. For the linear scene-referred space, stick with blend modes based on arithmetic operations (addition, multiplication, division, subtraction, average), on maximum/minimum comparisons (screen) or on channel separations (hue, color, chroma, etc.).

---

other artistic effects

There are also a number of artistic effect modules available in darktable. To name just a few:

- Use the *watermark* module to add an individual watermark to your image.
- Use the *grain* module to simulate the typical noise of classical analogue photos.
- Use the *color mapping* module to transfer the look and feel of one color image onto another.
- Use the *lowlight vision* module to simulate human vision making low light pictures look closer to reality.
- Use the *graduated density* filter to add a neutral or colored gradient to your image for exposure and color correction.

Please see the processing module reference for a list of the available modules.

modules to avoid

There are a number of modules which are no longer recommended for use within a scene-referred workflow. This doesn't mean they can't be used, but they can produce undesirable effects when their sliders are pushed too far, and there are better alternatives. In each case, the preferred alternative module is listed along with a brief explanation.

local tone mapping (deprecated)

prefer tone equalizer

This module applies a bilateral blur over a non-linear (log) mapping that can provoke halos and fringing. This is common issue for modules performing blurs and occlusions that operate over a non-linear encoding.

global tonemap (deprecated)

prefer filmic rgb

This module tries to deal with HDR images using the Lab color space, which is not well suited for high dyanamic ranges. The *filmic rgb* module operates in a linear space and can easily scale over a wide range of input values from the scene and fit them into the narrower dynamic range demanded by display and printing devices.

shadows and highlights

prefer tone equalizer

This module works with blurs in Lab color space, resulting in problems including halos, high local contrast in highlights and hue shifts towards blue in the shadows.

lowpass

prefer contrast equalizer or tone equalizer

Another module doing blurs in Lab space. Prefer the *contrast equalizer* for blurring, or the *tone equalizer* if local dynamic range compression is needed.

highpass

prefer contrast equalizer or local contrast

Uses a blur performed in Lab space, so has the same problems as with the *lowpass* module. Use *contrast equalizer* for fine sharpness, or *local contrast* for general sharpness.

sharpen

prefer contrast equalizer or local contrast

The USM algorithm used in the *sharpen* module suffers from same issues as the *highpass* module, and can easily cause artifacts. Use the presets offered by the *contrast equalizer* for de-blurring, or *local contrast* for general sharpness.

monochrome

prefer color calibration (or color balance)

The *monochrome* module can be quite fiddly to use. The *color calibration* presets better emulate what physically happens with film, or you can set the *output saturation* slider in the *color balance* module to 0% for a more perceptual approach.

fill light (deprecated)

prefer tone equalizer (or exposure)

Used to add light to a scene, this module again uses blurs in Lab space. The *tone equalizer* works in linear space, or you can also achieve a similar effect by using the *exposure* module with a drawn mask .

bloom

prefer tone equalizer (or exposure)

Again, this module uses blurs in Lab space. Either use the *tone equalizer* module or the *exposure* module with a parametric mask , both of which operate with linear encodings.

zone system (deprecated)

prefer tone equalizer (or exposure)

This module again operates in Lab space, and becomes problematic if you push it too far. It is better to use the *tone equalizer* or multiple instances of the *exposure* module with parametric masks to narrow down on a zone.

color correction

prefer color balance

Prefer the *color balance* module, which works in RGB color space and allows easy adjustment of the white balance in shadows (*offset*), midtones (*power*) and highlights (*slope*). Note the *offset*, *power* and *slope* that we normally use in linear spaces roughly correspond to the *lift*, *gamma* and *gain* parameters used in non-linear gamma-encoded spaces.

velvia

prefer color balance

The *output saturation* slider of the *color balance* module uses similar logic as the *velvia* module, but without the hue and brightness shifts, which can be difficult to manage.

***levels* / rgb levels**

prefer color balance

These modules basically implement a subset of the functions of the *color balance* module, which pretty much makes them redundant.

***tone curve* / rgb curve**

prefer color balance

These modules are normally used to adjust contrast. Their user interface assumes the mid-gray level is around 50%, but in linear scene-referred space mid-gray is much lower at around 18%. It is better to adjust the contrast in *color balance* module, where the mid-gray reference point can be set with the *contrast fulcrum* slider.

contrast brightness saturation

prefer color balance

This module works in Lab color space (with the limitations that implies) and basically duplicates functions already provided by *color balance*.

1.4.4. editing an image: display-referred workflow

This is a legacy mode which is retained to provide backward-compatibility with edits in older darktable versions, and to allows users to continue with their former way of working without forcing them to use the newer *scene-referred* workflow.

The *display-referred* workflow places more emphasis on performing image processing in the non-linear *display-referred* part of the pixelpipe . By default it uses the *base curve* module to tone map images from the linear *scene-referred* space into *display-referred* space, although other tone-mapping tools (such as the *tone curve* module) can also be used. Many modules are moved later in the pipeline (after this tone mapping transition) so that they work with gamma-encoded (*display-referred*) pixel values rather than linearly-encoded (*scene-referred*) pixel values.

Most of the basic steps required to develop images under the *display-referred* workflow are quite similar to the *scene-referred* workflow. The main differences lie in the choice of modules, and the order in which they appear in the pixelpipe. To see the difference in the ordering of the modules between the *display-referred* and *scene-referred* workflows, please refer to the default module order section.

Note: This section discusses a number of legacy modules that are no longer recommended for use in the scene-referred workflow, and new users are recommended to instead refer to the scene-referred section for a guide to how best to process images in darktable.

---

white balance

The *white balance* module works the same as in *scene-referred* workflow and, by default, uses the white balance coefficients provided by the camera. If this does not give acceptable results, use the camera presets or take the white balance from a neutral spot in your image. The temperature slider can be used to make the image “warmer” or “cooler”. More advanced color grading is better left to other modules, as discussed later.

exposure correction

The *exposure* module works the same as in *scene-referred* mode, but the way you use it is a little different. In *display-referred* mode, you need to make sure you don’t blow out your highlights too much, and use the *base curve* module to adjust the middle tones if needed.

While you can use the *exposure* module to tweak the black level to supply more contrast, you need to be very careful doing this as you can end up with negative RGB values. It is better to increase the contrast by adjusting the toe of the *base curve*, however this can be a little fiddly and it is one of the reasons why the *filmic rgb* module was introduced to darktable.

noise reduction

As with the *scene-referred* workflow, the best starting point for noise reduction is the *denoise (profiled)* module. Similarly, you may also choose to use *raw denoise* , *surface blur* , *astrophoto denoise* , or the *contrast equalizer* module.

fixing spots

As with the *scene-referred* workflow, you can use the *retouch* , *spot removal* and *hot pixels* modules to correct artifacts in your image.

geometrical corrections

As with the *scene-referred* workflow you can use the *crop and rotate* , *perspective correction* and *lens correction* modules to correct distortions and crop your image.

bringing back detail

Raw images often contain more information than you can see at first sight, especially in the shadows. 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. The *color reconstruction* module is able to fill overexposed areas with suitable colors based on their surroundings. The *filmic rgb* module also offers highlight reconstruction, but be sure to disable *base curve* first.

adjusting tonal values

Almost every workflow is likely to include adjusting the image’s tonal range and darktable offers several modules to assist with this. The most basic is the *contrast brightness saturation* module. In the *tone curve* module, tonal values are adjusted by constructing a curve. The *levels* and *rgb levels* modules offer a concise interface, with three markers in a histogram. And of course, there is nothing to stopping you from using the *filmic rgb* module in a *display-referred* workflow if you so wish (after disabling *base curve*).

enhancing local contrast

Local contrast enhancement can emphasize detail and clarity in your image. Carefully used, it can give your photograph the right pop. Several modules are available 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 *contrast equalizer* module. Take a look at its presets to get a feeling for how it works. The *contrast equalizer* is darktable's "Swiss Army Knife" for many adjustments where spatial dimension plays a role. Note that the location of this module in the pixel pipeline differs significantly between the *scene-referred* and *display-referred* workflows.

color adjustments

darktable offers many modules for adjusting colors in an image. A very straightforward technique is implemented in the *color correction* module. Use this module 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, lightness and hue, in user defined zones. The *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 the *color calibration* module.

sharpening

If you start your workflow from a raw image, your final output will need to be 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.

artistic effects

darktable comes with a rich set of artistic effect modules. For example you can use the *watermark* module to add a 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. The *low light* module allows you 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.

1.4.5. exporting and uploading images with metadata

Changes to an image are not saved directly to the image file contrary to a regular image editor. Rather, darktable is a non-destructive editor, which means that all changes are recorded in darktable's library database, and the original image is left untouched. Therefore, you need to export images in order to bake your processing options and metadata changes into an output file that can be distributed outside of darktable.

Images are exported using the export module which is available in the lighttable and darkroom views. This module offers a lot of options, but by far the most common use is to "save my developed raw image as a JPEG".

When exporting images in darktable, there are two basic questions you need to answer:

- *Where shall I send the exported images?* Most often you will choose to write the files to a folder on your local disk, but other options include writing them to a LaTeX photo book template or sending them to another program such as Hugin (for panarama stitching) or GIMP (for further editing).
- *In what format shall I save the exported images?* This covers not only the image file format (JPEG, PNG, TIFF, OpenEXR etc.) but also the quality, compression, resolution, picture profile settings and which metadata to embed in the exported image.

The most common steps would therefore be:

1. Select one or more images in the lighttable view
2. Choose the target storage and file format
3. Set the maximum width and height image restraints. Leave the width and height restraints at zero to export at full resolution.

By default, an image will be saved to local disk as a high-quality JPEG at full resolution. For further information on all the available export options, please refer to the export module reference section.

2. Lighttable

2.1. overview

The lighttable view allows you to view and manage your image collection.

The centre view contains thumbnails of your images – how they are displayed depends on which mode you are working in.

While the mouse is over an image thumbnail or images are selected, there are a number of actions you can perform:

```
0 – 5       set the rating of the image
            if an image has 1 star and you hit the 1 key, the image will become unrated
R           rate the image as "rejected"
F1 – F5     set a color label
Ctrl+C      copy the history stack
Ctrl+V      paste the copied history stack
D           open in darkroom view for developing
W           fully zoom into the current image while the key is pressed
Ctrl+W      fully zoom into the current image and show areas in focus
```

2.2. lighttable view layout

left panel

From top to bottom:

import
Import images from the filesystem or a connected camera.

collect images
Filter the images displayed in the lighttable center panel – also used to control the images displayed in the filmstrip and timeline modules.

recently used collections
View recently used collections of images

image information
Display image information

lua scripts installer **(optional)**
Install lua scripts

right panel

From top to bottom:

select
Select images in the lighttable using simple criteria.

selected images(s)
Perform actions on selected images.

history stack
Manipulate the history stack of selected images.

styles
Store an image’s history stack as a named style and apply it to other images.

metadata editor
Edit metadata for selected images.

tagging
Tag selected images.
geotagging
Import and apply GPX track data to selected images.
export
Export selected images to local files or external services.

bottom panel

From left to right:

star ratings
Apply star ratings to images
color labels
Apply color categories to images
mode selector
Choose lighttable mode using the dropdown
zoom
Use the slider to change the size of thumbnails
 **enable focus-peaking mode**
Click to enable focus-peaking mode
 **set display profile**
Click to choose display profiles

2.3. undo/redo

Most changes made within the lighttable are recorded and can be reverted to a previous state. This includes modifications to color labels, ratings, geo-localization, tags, metadata, orientation, copy/paste of history, image duplication, or application of a style. Note that the facility to undo/redo actions is unlimited in the number of steps while in the lighttable view, but it is reset each time you switch to a different view.

Press Ctrl+Z to undo the last modification and Ctrl+Y to redo the last undone modification (if any).

2.4. lighttable modes

2.4.1. filemanager

In the default mode images are displayed in a grid with an adjustable number of images per row.

controls

zoom
The number of images in each row can be altered using the slider in the bottom panel, or by holding Ctrl while scrolling over the center view with your mouse.
navigate
You can navigate through the images using the arrow keys (←/→/↑/↓) or by scrolling with your mouse. Press the Home key to scroll to the top of the collection, the End key to scroll to the bottom, and PageUp/PageDown to scroll up/down by a page.

select

You can select the image under the pointer by clicking on its thumbnail or by pressing Enter. A range of images can be selected by clicking on the first image and then Shift+clicking on the last one. Images can be added or removed from a selection by Ctrl+clicking on their thumbnails or by pressing Spacebar.

2.4.2. zoomable lighttable

The *zoomable lighttable* mode provides a different way to navigate large collections of images.

controls

zoom

Scroll with the mouse wheel to zoom in and out of the lighttable (compared to Ctrl+scroll in filemanager mode).

Zooming the thumbnails does not change the number of thumbnails per row, so the lighttable can exceed the visible area on all sides.

navigate

Hold down the left mouse button to drag the images around on the lighttable to navigate through your collection.

select

As with the filemanager mode, you can select the image under the pointer by clicking on its thumbnail or by pressing Enter. A range of images can be selected by clicking on the first image and then Shift+clicking on the last one. Images can be added or removed from a selection by Ctrl+clicking on their thumbnails or by pressing Spacebar.

Hint: you may find that the images are slow to load when zooming quickly through a large collection. One way to speed up the navigation is to generate a cache containing all the thumbnails using the `darktable-generate-cache` command.

2.4.3. culling

Culling mode allows you to display images side by side to easily compare them.

controls

zoom

In culling mode, you can zoom into images (up to 100%) by holding Ctrl while scrolling with the mouse wheel.

Pan within zoomed images with click+drag.

By default, zooming and panning are synchronized between all visible images. If you want to zoom or pan only a specific image, add the Shift modifier to the above actions.

navigate

Use the mouse wheel or arrow keys (←/→) to scroll through your collection.

modes

There are two different ways to define how many images are shown at the same time: “fixed” and “dynamic”. Switch between these methods while in culling mode by pressing the “<” key.

fixed mode

The number of images displayed is always the same, independent of the selection length. This number can be set with a slider on the bottom panel.

In this mode, you will navigate through all selected images. If no selection is set (or if only one image is selected), you will navigate through all images.

The default keyboard shortcut to enter culling in fixed mode is X.

dynamic mode

All of the selected images are shown. If no selection is set (or if only one image is selected) the last value from fixed mode is used.

The default keyboard shortcut to enter culling in dynamic mode is Ctrl+X.

Hint: To enhance performance when loading zoomed images, you can enable (preferences > cpu/gpu/memory > enable disk backend for full preview cache). Bear in mind that this can occupy a lot of disk space.

2.4.4. full preview

From any of the lighttable modes, you can display a fully zoomed preview of the image that is currently under the mouse pointer by pressing and holding down W. This is useful to more closely inspect an image while rating and selecting images.

Pressing and holding Ctrl+W fully zooms into the image and also identifies any regions of sharpness in the image that may indicate the image is in focus. For this tool to work the input image needs to hold an embedded JPEG thumbnail, which is the case for most raw files.

Regions in the image with a high level of sharpness are indicated with red borders. If there are no such regions found, any regions of moderate sharpness are identified with a blue border. Note that this is not the same as the focus peaking indicator, which is another way to identifying areas of sharpness within an image.

Sometimes pressing W or Ctrl+W may not appear to have any effect – in such cases, click on the image thumbnail and press the corresponding key again.

If you want the full preview to stay in place without having to hold the W key, you can use sticky preview mode by pressing F. In sticky preview mode, you can zoom and pan within the image in a similar way to the culling mode. Press F or ESC again to return to the original view.

2.5. digital asset management

2.5.1. 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 the filesystem, the images are organized in a film roll whose name is derived from the name of their parent folder. Re-importing a folder will add any new images to the existing film roll; images already present in the film roll are not reimported (though the contents of their XMP sidecar files are reloaded into the library database).

Note: Importing images in darktable does not physically copy them to another location: *importing images will not back up your files*.

2.5.2. collections

A collection is a set of images defined by a specific combination of selection criteria. The most basic kind of collection is a film roll, which contains all of the images that have been imported from a specific folder on disk.

You can easily construct other kinds of collections based on various image attributes (Exif data, filename, tags etc.). Multiple criteria can be logically combined to narrow or extend your collection (see collect images).

darktable keeps a list of the most recently used collections for quick access (see recent collections).

2.5.3. thumbnails

Each image in the current collection is represented by a thumbnail in the lighttable view and filmstrip module. A cache of the last recently used thumbnails is stored in a file on disk and loaded into memory at startup. The size of this cache can be adjusted in preferences > cpu/gpu/memory .

thumbnail creation

Thumbnails are created whenever an image is imported into darktable for the first time, after an image has been modified in the darkroom, or when revisiting an image whose thumbnail is no longer available.

When an image is imported into darktable for the first time darktable can either try to extract an embedded thumbnail from the input image (most raw files contain these, usually in JPEG format) or process the raw image itself using default settings. You can define how darktable obtains its thumbnails in preferences > lighttable .

Extracting an embedded thumbnail from the input image has the advantage of being very fast. However, these thumbnails have been generated by the raw converter of the camera and do not represent darktable's "view" of that image. You will notice the difference as soon as you open the image in the darkroom mode, at which point darktable replaces the lighttable's thumbnail with its own internally processed version.

After importing a new film roll darktable automatically generates thumbnails for new images as they are needed. When importing a large set of new images, thumbnail generation can slow down navigation in the lighttable view. Alternatively you may terminate darktable and generate the thumbnail cache separately by running the darktable-generate-cache binary. This program will generate all missing thumbnails in one go.

As the thumbnail cache has a pre-defined maximum size it will eventually get filled up. If new thumbnails are subsequently added, old thumbnails are dropped from the cache. However, darktable will keep all thumbnails on disk if the corresponding disk backend option is activated in preferences > cpu/gpu/memory . Access to the thumbnails in this secondary cache is slower than the primary cache, but still much faster than reprocessing thumbnails from scratch. The size of the secondary cache is limited only by available disk space.

Thumbnails are never removed from the secondary cache. You can manually clean the secondary cache by recursively deleting all images in the `$HOME/.cache/darktable/mipmaps-xyz.d` folder (where `xyz` denotes an alphanumeric identifier of the cache). After clearing the secondary cache you can simply allow darktable to re-generate thumbnails as needed, or you can generate all thumbnails in one go with `darktable-generate-cache` .

If you choose not to activate the disk backend and select too small a cache size you may observe adverse effects: Continuous regeneration of thumbnails when you navigate your collection, flickering of thumbnail images, or darktable may even become unresponsive. A good choice of cache size is 512MB or higher (see memory for more information).

All thumbnails are fully color managed if the corresponding option is activated in preferences > lighttable . Colors are rendered accurately on screen as long as your system is properly set up to hand over the right monitor profile to darktable. For more information on color management see the color management section.

skulls

If for some reason darktable is unable to generate a thumbnail, it displays an image of a skull (). Don't panic!

There are three main reasons that this could happen:

- *Missing Image File*: darktable remembers all images ever imported, as long as they have not been removed from your database. If 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 using the selected images module 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.
- *Invalid Image Format*: While the extension of an image may seem to be supported by darktable, its contents could be either an unsupported image format or a corrupt file.
- *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, especially on a 32-bit system. See memory for more information.

2.5.4. star ratings & 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 can be displayed over thumbnails in the lighttable view and filmstrip module.

star ratings

You can give an image a rating from zero to five stars. The quality criteria which lead to a rating are up to you. Whenever you import images, each image receives a default rating which you can define in preferences > import . You can also mark an image as “rejected”.

There are several ways to change a rating. While hovering the cursor over an image thumbnail, you can press a number key 0 – 5 to define the number of stars, or press R to “reject” an image. This is probably the fastest way to rate your images on first inspection of a film roll.

You can also directly click on the star icons that are overlayed on the thumbnails. Click the x to reject.

As Rejecting an image removes the currently-applied star rating, you can undo the rejection by clicking x or pressing R again.

Similarly you can click the first star for a second time to reset the image rating to unranked, or zero stars. This behavior can be changed in preferences > lighttable .

To rate multiple images at once, select those images in the lighttable or filmstrip and then press the appropriate shortcut key, or click the desired star rating in the bottom panel of the lighttable view.

You can filter images by star rating in the top panel .

color labels

Color labels are another way to classify 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 in the lighttable or filmstrip and then press the appropriate shortcut key or click the corresponding color button in the bottom panel. To remove all labels from the selected images, press the gray button.

You can filter images by color label in the collect images module.

2.5.5. image grouping

Grouping images helps to improve the 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) module, or by pressing Ctrl+G. Likewise, you can remove selected images from a group by clicking the “ungroup” button, or pressing Ctrl+Shift+G. Images generated by duplicating an existing image are automatically grouped. Similarly, if you import multiple images with the same base name, but different extensions (eg. `IMG_1234.CR2` and `IMG_1234.JPG`), these images will automatically form a group.

Images which are members of a group are labeled with a () symbol in their thumbnails.

The “group” button () in the top panel of the lighttable view toggles grouping mode on and off. If grouping is off, each image is displayed as an individual thumbnail. If grouping is on, images of a group are collapsed, and are represented by a single thumbnail image. This displayed thumbnail is called the group head. If you press the () symbol in the group’s thumbnail, only this group is expanded. If you then expand another group, the first group collapses. To collapse an expanded group again, just click again on the () symbol of its group head.

An expanded group in the filemanager mode of lighttable view is indicated by an orange frame which appears as soon as your mouse pointer hovers over one of the images. This frame surrounds all images in the group.

You can define which image constitutes the group head, while in an expanded view of a group, by clicking on the () symbol of the desired image. That symbol is shown only if grouping mode is enabled, so to change the group head, you need to first enable group mode, then expand the group you want to change and finally click the () symbol on the desired image. To determine which image is the group leader, hover over the () and a popup will identify the group leader.

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

Image groups are also 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: Simply 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.

Note: “duplicating images” only generates a copy of an image’s history stack, stored in another small XMP file. There is still only one raw file.

2.5.6. metadata and tagging

darktable allows you to store additional information about your images to allow them to be more easily searched and grouped. This information is stored in darktable’s database and XMP sidecar files and can also be included within exported images.

metadata

Metadata (e.g. title, description) is free-format text that usually differs for each image.

See the metadata editor module for details of how to add metadata to images.

tagging

Tags are usually shared between multiple images and are used to categorise and group them.

See the tagging module for details of how to add tags to images.

3. Darkroom

3.1. overview

The darkroom view is where you develop your images. The center panel contains the image currently being edited.

zoom

Middle-click on the center panel to zoom to 1:1 and double-middle-click to zoom to 2:1.

Alternatively you can zoom in and out between 1:1 and “fit to screen” by scrolling with your mouse. Scroll with your mouse while holding the Ctrl key pressed to extend the zoom range to between 2:1 and 1:10.

3.2. darkroom view layout

left panel

From top to bottom:

navigation
Navigate and zoom the center view

snapshots
Take and view snapshots for comparison with the current edit

duplicate manager
View and manage duplicates

global color picker
Select and display color information taken from parts of the image

tagging
Manage tags

image information
Display information about the current image

mask manager
View and edit drawn shapes

export
Export selected images to local files or external services.

right panel

From top to bottom:

histogram
A graphical depiction of the image’s light levels

module groups
Select module groups (if enabled)

search module
Search for a module (if enabled)

processing modules
The modules used to process an image

module order
Choose the order in which processing modules are executed in the pixelpipe

bottom panel

From left to right:

presets
Quick access menu for module presets. You can manage the contents of this menu by selecting “manage quick presets list...”

styles
Quick access menu for styles

second darkroom window
For multi-monitor setup, allows the user to display a second image on another screen

focus peaking
Toggle focus peaking mode

color assessment
Toggle the ISO12646 color assessment view

raw overexposed warning
Toggle raw overexposed indicators (right-click for options)

clipping warning
Toggle clipping warnings (right-click for options)

soft proof
Toggle softproofing overlays (right-click for options)

gamut check
Toggle gamut checking (right-click for options)

overlay line colors
Set colors for modes that overlay lines on the image (masks, crop guides etc.)

You can also enable the filmstrip module at the bottom of the screen to allow you select and interact with the images that are currently visible in the lighttable view.

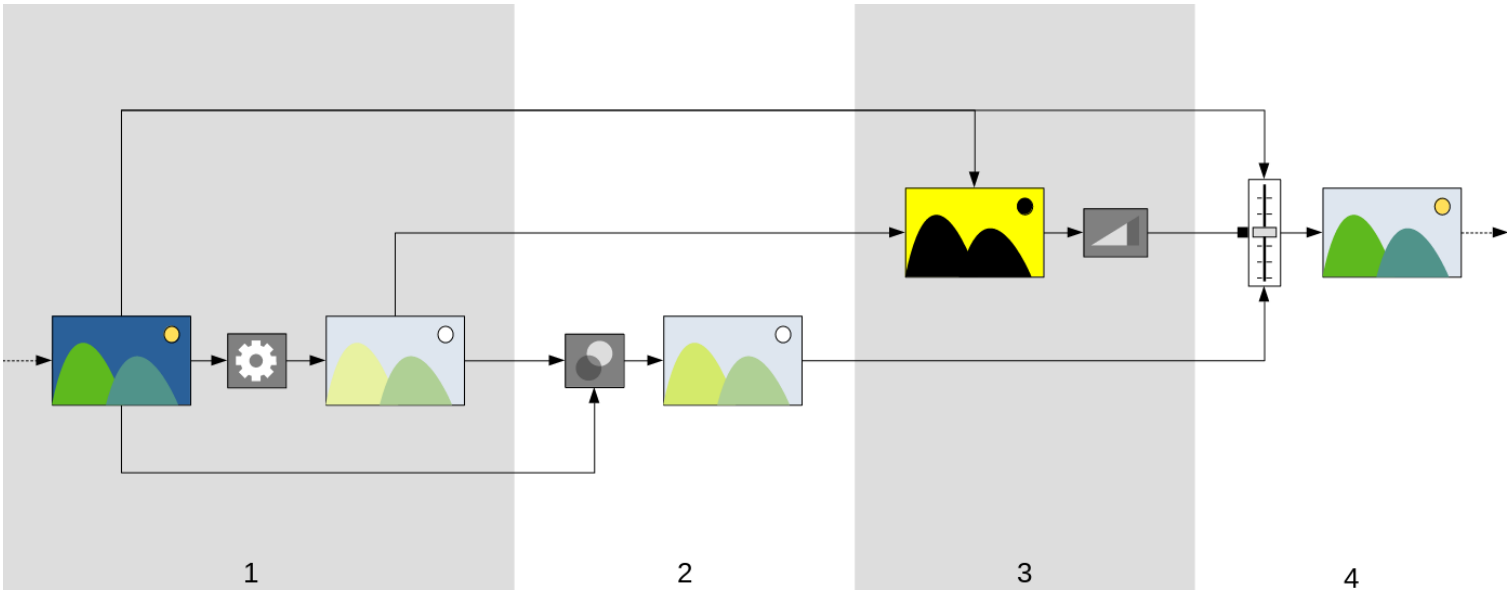
3.3. the pixelpipe

3.3.1. the anatomy of a processing module

The basic element of image processing in darktable is the processing module . In order to process a raw image a number of such modules act on the input image in sequence, each performing a different *operation* on the image data. For those familiar with Adobe Photoshop, the concept of a *processing module* in darktable is analagous to that of an *adjustment layer* in that both make an incremental adjustment to the image, building on top of the adjustments that came before.

darktable also provides utility modules , however these are not directly involved in image processing, and instead provide a GUI that allows you to manage your images, tag them, export them and so on.

Every processing module executes independently in a similar manner:



1. Receive the *module input* from the last executed module and perform an *operation* on it to produce the *processed output*. This *operation* is different for every processing module .

2. Combine the *module input* and *processed output* using a blending operator to produce the *blended output*. If no blending is performed, the output of this step is the same as the *processed output*.

3. Generate a *mask*, which defines an *opacity* for each pixel in the image. The *opacity* is later used to control how strongly the module's operation is applied to each part of the image.

   You may define your own mask by drawing shapes over the image or by using pixel properties from the *module input* or *processed output* (see masks for details). This mask may be further modified with a global opacity setting, which affects every pixel equally.

   If no drawn/parametric mask is used, the output of this step is a mask where every pixel has the same opacity (governed by the global opacity setting). If no opacity is defined (no blending is performed) a global opacity of 1.0 (or 100%) is assumed.

4. Combine the *module input* and *blended output* pixel-by-pixel using the *mask* as a mixing operator, to produce the *final output*. Where the mask opacity is 100%, the *final output* is the *blended output* for that pixel. Where the mask opacity is 0 the final output is the *module input* for that pixel. An intermediate opacity combines the *blended output* and *module input* proportionally. The *final output* is passed to the next module for further processing.

Steps 2 and 3 are optional and not supported by all modules. For example, the demosaic module must be applied to the entire raw file in order to produce a legible image so it does not make sense to mask or blend its output.

Each of the above steps is defined in more detail in subsequent sections.

3.3.2. the pixelpipe & module order

The ordered sequence of processing modules operating on an input file to generate an output image is known as the "pixelpipe".

The order of the pixelpipe is represented graphically by the order in which modules are presented in the user interface – the pixelpipe starts with a RAW image at the bottom of the list, and applies the processing modules one by one, piling up layer upon layer of processing from the bottom up, until it reaches the top of the list, where it outputs the fully processed image.

Note: The order in which processing modules are executed exactly matches the order in which the modules appear in darktable's user interface. **Changing the order of the modules in the user interface will change how your image is processed.**

---

module order and workflows

The order in which modules are executed within the pixelpipe has been carefully chosen to give the best output quality. In previous versions of darktable it was not possible to change the module order. However, there are a number of very specific use cases where the movement of some modules within the pixelpipe is advised.

One of the main reasons to change the module order came about with darktable version 3.0, which introduced the new *scene-referred* way of working. Version 3.2 formalised this by introducing the *display-referred* and *scene-referred* workflows, which are controlled by the preferences > processing > auto-apply pixel workflow defaults setting.

The main difference between these two workflows is that *display-referred* operates primarily in the low-dynamic-range of the user's screen, whereas *scene-referred* operates in the high-dynamic range of the original RAW file. The transition between *scene-referred* and *display-referred* is usually performed by a *tone mapping* module which takes the high-dynamic-range input from the camera and compresses it to fit the dynamic range of the output medium.

display-referred workflow

Prior to version 3.0 darktable's workflow was *display-referred* (auto-apply pixel workflow defaults = "display-referred") and this is still provided as the default workflow in the current version. In this workflow, the *base curve* or *filmic rgb* module performs tone mapping early in the pixelpipe and most other darktable modules operate on image data in the compressed *display-referred* space.

This worfklow enables the legacy (pre-darktable-3.0) module order and automatically switches on the *base curve* module for new images.

Pixel data within the *display-referred* space is non-linear and is not a physically realistic representation of the original scene. This can lead to various artifacts with some modules, hence the creation of the (now recommended) *scene-referred* workflow.

scene-referred workflow

The new *scene-referred* workflow (auto-apply pixel workflow defaults = "scene-referred") was introduced as part of darktable 3.0. The module order was entirely rearranged to place the *filmic rgb* and *base curve* tone mapping much later in the pixelpipe. This means that most pixel operations now operate in *linear rgb* space with only a few modules remaining within the non-linear *display-referred* space. Within this workflow it is now recommended that the majority of image processing takes place using the modules modules up to and including *filmic rgb* . Operations in this section of the pixelpipe, being truly linear, are much more physically realistic and produce fewer artifacts.

This workflow enables the new (darktable 3.0+) module order and automatically enables the *exposure* and *filmic rgb* modules with some presets designed to act as a reasonable starting point for scene-referred image editing.

changing module order

It remains highly recommended that users not change the order within the pixelpipe for a number of reasons:

- The sequence of modules has been selected with great care in order to give highest output quality. Changes to the sequence often worsen the result rather than improving it.
- Some processing modules simply don't make sense if they are shifted in the pixelpipe. For example, *highlight reconstruction* needs to be performed on raw data before *demosaic* , which itself needs to be performed before any *input color profile* can be applied. For this reason it is still not possible to move some of the modules that are placed early in the pixelpipe.
- Most of darktable's modules are designed to work within a specific color space (see the color management section for more details). Full flexibility would require modules to support different parallel algorithms depending on the color space they are working in, which would drastically increase complexity.

Despite the general recommendation to leave the pixelpipe order alone, it is possible to move modules within the pixelpipe by holding Ctrl+Shift and dragging and dropping the desired module to a new location. This should only be done by experienced users who understand the impact this will have on the image.

The module order can be manually changed back to either the *3.0* or *legacy* versions using the module order module, which can also be used to define your own custom module order presets.

3.3.3. the history stack

The *history stack* stores the entire editing history for a given image, in the order in which those edits were applied. It is saved to darktable's library database and the image's XMP sidecar file and persists between editing sessions.

Each time a processing module is enabled, disabled, moved or amended a new entry is added to the top of the *history stack*.

The history stack can be queried and modified within the history stack module in the darkroom.

Note: The history stack is not a representation of the order in which the modules are **executed** but the order in which they were **amended**. The execution order is represented by the order of the modules in the right-hand panel.

3.3.4. undo and redo

While you are editing your image, darktable records all of the modifications you make to that image. Thanks to this recording process it is possible to undo and redo changes to recover a previous editing state. Note that the undo/redo facility is unlimited in the number of steps while editing an image, but is reset each time the darkroom is switched to a new image.

Press Ctrl+Z to undo the last modification and Ctrl+Y to redo the last undone modification (if any).

3.4. processing modules

3.4.1. module header

At the top of each processing module is the *module header*.

Click on the module name to expand the module and display the parameters that control its operation.

By default darktable will only allow one processing module to be expanded at a time – if you click the header of another module, the previously-opened module's controls are collapsed. If you want to expand more than one module, you may expand further modules by Shift+clicking on the header and all previously expanded modules will remain open. This behaviour can be reversed via a setting in preferences > darkroom .

Note: Expanding a module does not cause it to be activated. See below for how to activate modules.

The module header contains the following controls in order from left to right:

on/off button
Click to toggle the module on or off. Some modules are essential for image processing and cannot be disabled (though their parameters may be amended). Similarly, some modules are not applicable for certain types of image and cannot be enabled.

Ctrl+click on the on/off button to toggle whether the module has focus. The focus state is usually used to activate any overlays that a module places over the image to control its functionality. For example, the *crop & rotate* module only shows the composition and crop guide lines on the image if it has focus.

module name
The module name consists of a description of the module's operation (which cannot be changed) followed by the module's instance name (which can). By default the first instance of a module has an empty instance name. If you create additional instances, the name of each new instance is initiated with a unique integer. For example, the second created instance of an exposure module will be automatically named exposure 1.

Ctrl+click on a module's name to manually amend its instance name.

mask toggle
This icon will appear in the header whenever a mask is active on a module. Hover over the icon to see what type of mask is enabled. Click it to display the current mask as a yellow overlay over a black-and-white version of the image. Solid yellow indicates an opacity of 100%; a fully visible gray background image (without yellow overlay) indicates an opacity of 0%. This toggle button can be disabled in preferences > darkroom > show mask indicator in module headers .

multiple instance menu
This drop-down menu allows you to create, delete, move and rename module instances. Right-click on this icon to directly create a new instance of the module. See the multiple instances section for more information.

reset parameters
Click to reset all parameters within the module to their default values. Ctrl+click to reapply any automatic presets for the module – if no automatic presets are applicable for this module, Ctrl+click will simply reset to default values (same as click).

presets menu
This menu allows you to apply, create and edit module presets. See the presets section for more information.

The visibility of the three icons to the right of the module name can be controlled through preferences > darkroom > show right-side buttons in darkroom module headers .

3.4.2. multiple instances

Many of darktable's modules can be applied more than once in the pixelpipe. Each instance of a module behaves independently, taking its input from the module below in the pixelpipe delivering its output to the module above.

As with the base instance of a module, all instances can be moved independently within the pixelpipe either by holding Ctrl+Shift while dragging & dropping or by choosing "move up" or "move down" in the *multiple instances* drop-down menu.

Instances can be renamed by Ctrl+clicking on the module header.

typical use cases

There are many occasions where it makes sense to have a module apply more than once in the pixelpipe. Here are some typical use cases.

- The *exposure* module can be used in combination with masks to lighten or darken parts of an image. A separate instance may be created to modify each part of the image.
- You may wish to handle luma and chroma noise independently. This can be accomplished by generating two instances of your chosen denoising module and using the first one only on luma (by selecting blend mode "lightness") and the second one only on chroma (by selecting blend mode "color").

Note: Each instance also adds to the workload of your pixelpipe. Generating too many instances – especially of the more demanding modules – will cause noticeable slow-down.

managing multiple instances

Click on the *multiple instance menu* in the module header to display a drop-down menu, with the following options. Right-click on the menu icon to create a new instance directly (same action as clicking on the “new instance” option of the menu).

new instance
Create a new instance of the current module with all of its parameters reset to default. The ‘instance name’ is automatically set to a unique integer so that it can be distinguished from its parent.

duplicate instance
Create a new instance of the current module with all of its parameters inherited from its parent. As with ‘new instance’ the ‘instance name’ is automatically set to a unique integer.

move up/down
Move the instance up or down in the pixelpipe

delete
Remove the current instance. This option is not available if only one instance is present.

rename
Rename the current instance. See the history stack section for more details on how the instance name impacts copying and pasting history stacks.

3.4.3. presets

Presets allow you to store commonly used module settings for future use. Some modules already come with pre-defined presets and you may also define your own. Both internal and user-defined presets can be shown by clicking the *presets menu* in the module header .

Most of the functionality described here applies to processing modules only. However, presets can also be used with some utility modules. When used with utility modules, the functionality to auto-apply or auto-show presets based on image Exif data is not available.

Please note that, for processing modules, the saved preset also includes the active state of the module. You can use this to create your own default settings, which you can activate on-demand. Simply set your desired defaults, disable the module, and save the preset.

the presets menu

The presets menu will contain one or more of the following entries depending on what presets are defined or selected for the current module:

preset list
A list of the presets available for the current module. The currently selected preset (if any) is shown in **bold**.

edit this preset
If a preset has been selected, click to edit that preset (see below)

delete this preset
If a preset has been selected, delete that preset.

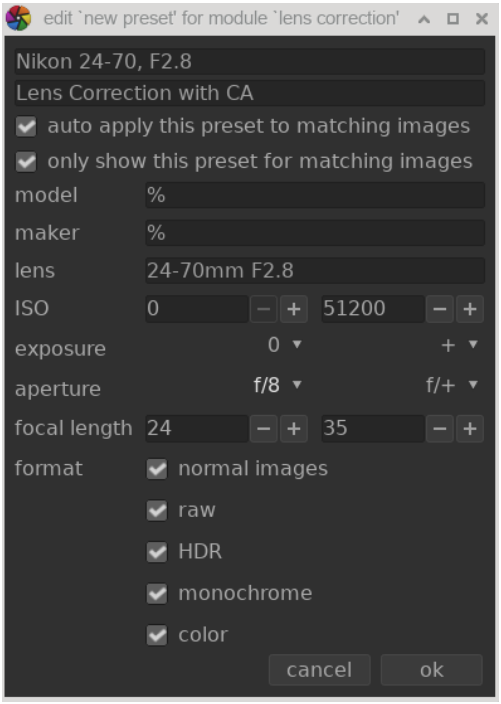
update preset [name]
Update the named preset to match the module’s current parameters.

store new preset
Create a new preset using the module’s current parameters.

Left-click on a preset name to apply the preset to the current instance of the module. Right-click on a preset name to create a new instance of the module and apply the selected preset to it. You can also apply a preset at any time while you are in the darkroom by pressing the shortcut key that has been assigned to it (see preferences > shortcuts).

creating and editing presets

When creating or editing presets, the following dialog is shown:



The following options can be set:

name
The name of the preset

description
A searchable description for the preset (optional)

auto apply this preset to matching images *(processing modules only)*
Check this box to automatically apply this preset to matching images when they are opened in the darkroom for the first time (you can reapply such automatic presets by Ctrl+clicking on the *reset* button in the module header). Additional controls will appear to allow you to define which images the preset will be applied to based on image Exif data (see below).

For example, 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. Leave all fields unchanged to auto-apply a preset to all images.

The example dialog above sets up following rules: if the lens name matches, the aperture is greater than or equal to f/8 and the focal length is between 24 and 35mm the preset will be automatically applied.

The image information module displays the camera model and lens name for each image. Use this to ensure you have the correct spelling.

only show this preset for matching images *(processing modules only)*
Check this box to automatically show the preset in the preset menu, using the same set of filters.

filter criteria

The following criteria can be used to auto-apply or auto-show presets for processing modules.

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).

format
Only apply the preset to certain types of image. Choose from “normal images”, “raw”, “HDR”, “monochrome” and “color”.

managing presets

Both user-created and pre-defined presets can be viewed and managed from within preferences > presets .

Note: If you create a user-defined preset with the same name as a built-in preset, your preset will override the built-in version, which will no longer be accessible.

If you delete a preset that has the same name as one of the built-in presets, then your user preset will be deleted, and that preset name will no longer appear in the preset menu at all. The next time you start darktable, the corresponding built-in preset will once again become visible.

3.4.4. module controls

sliders

Sliders offer five different methods of interaction, depending on the level of control you require.

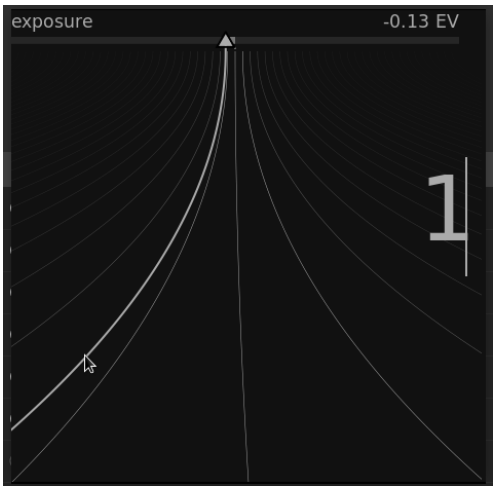
left-click
Click anywhere in the slider area to set the value. You can also click and drag to change it. You don’t have to aim for the triangle or even the line - you can click anywhere in the entire height of the slider including the label.

mouse wheel
Hover over the slider with your mouse, then use your mouse wheel to adjust the value.

keyboard arrow keys
When the slider has focus you can hover over the slider with your mouse, then use your keyboard’s arrow keys (←/↓ and →/↑) to adjust the value. In order to give focus to the widget without changing the current value you can right-click, then right-click again.

right-click

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



A bent line extending from the triangular marker moves with your mouse. The closer your mouse pointer is to the triangular marker the coarser the control you have over the value; the further away from the triangular marker the finer your control. Left-click with your mouse to accept the new value and and hide the pop-up.

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 expression which darktable will calculate for you – the previous value is referenced as “x”.

double-click

You can double-click on a slider or its label to reset to the default value.

In addition, the precision of mouse-wheel and arrow key-adjustments can be altered:

- hold down the Shift key while adjusting to *increase* the step size by a factor of 10.
- hold down the Ctrl key while adjusting to *decrease* the step size by a factor of 10.

Both of these multipliers can be amended in the `$HOME/.config/darktablerc` file:

```
darkroom/ui/scale_rough_step_multiplier=10.0
darkroom/ui/scale_precise_step_multiplier=0.1
```

comboboxes

Click on a combobox to open a list of available options which you can click to select. Occasionally the selection list will open close to the bottom or top of the screen meaning that only some of the items are visible – simply scroll with your mouse wheel to bring up the full list. Alternatively, you can also use the mouse wheel and keyboard arrow keys to select an option, or start typing to filter the combobox entries.

color pickers

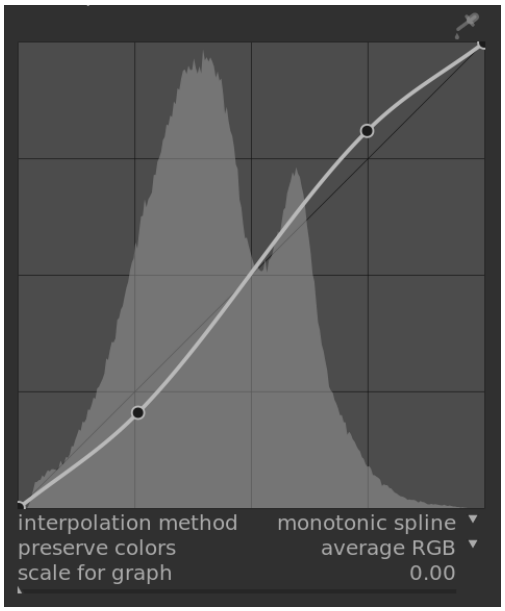
A number of modules allow parameters to be set using color pickers (identified by the  icon). These use a standard interface and most can operate in either point or area mode. Point mode can be activated by clicking on the color picker icon. Ctrl+click on the icon to activate area mode.

keyboard shortcuts

Module parameters can also be amended with keyboard shortcuts. See preferences > shortcuts for more information.

3.4.5. curves

Three processing modules (base curve , tone curve and rgb curve) use curves to control the tones in the image. These modules have some common features that deserve separate discussion.



nodes

In their default state, curves are straight lines, defined by two anchor nodes at the top-right and bottom-left of the curve graph. You can move the nodes to modify the curve or generate new nodes by clicking on the curve. Ctrl+click to generate a new node at the x-location of the mouse pointer and the corresponding y-location of the current curve – this adds a node without the risk of accidentally modifying the curve. Up to 20 nodes can be defined per curve. To remove a node, click on it and drag it out of the widget area.

curve controls

The following controls are common to two or more of the above processing modules and are therefore discussed separately here. Please see the individual module documentation for discussion of any additional module-specific controls.

interpolation method

tone curve and rgb curve modules only

Interpolation is the process by which a continuous curve is derived from a few nodes. As this process is never perfect, several methods are offered that can alleviate some of the issues you may encounter.

- Arguably, the most visually pleasing method is the “cubic spline” – since it gives smooth curves, the contrast in the image is better enhanced. However, this method is very sensitive to the nodes' position, and can produce cusps and oscillations when the nodes are too close to each other, or when there are too many of them. This method works best when there are only 4 to 5 nodes, evenly spaced.
- The “centripetal spline” method is designed specifically to avoid cusps and oscillations, but as a drawback it will follow the nodes more loosely. It is very robust, no matter the number of nodes and their spacing, but will produce a more faded and dull contrast.
- The “monotonic spline” method is designed specifically to give a monotonic interpolation, meaning that there will be none of the oscillations the cubic spline may produce. This method is most suitable when you are trying to build an analytical function from a node interpolation (for example: exponential, logarithm, power, etc.). Such functions are provided as presets. It is a good trade-off between the two aforementioned methods.

preserve colors

If a non-linear tone curve is applied to each of the RGB channels individually, then the amount of tone adjustment applied to each color channel may be different, and this can cause hue shifts. Therefore, the *preserve colors* combobox provides different methods of calculating the “luminance level” of a pixel. The amount of tone adjustment is calculated based on this luminance value, and then this same adjustment is applied to all three of the RGB channels. Different luminance estimators can affect the contrast in different parts of the image, depending on the specific characteristics of that image. The user can therefore choose a particular estimator that provides the best results for the given image. Some of these methods are discussed in detail in the **preserve chrominance** control in the *filmic rgb* module. The following options are available:

- *none*
- *luminance*
- *max RGB*
- *average RGB*
- *sum RGB*
- *norm RGB*
- *basic power*

scale for graph

tone curve and base curve modules only

The scale allows you to distort the graph display so that certain graphical properties emerge to help you draw more useful curves. Note that the scaling option only affects the curve display, not the actual parameters stored by the module.

By default, a “linear” scale is used (defined by a scale factor of 0). This scale uses evenly spaced abscissa and ordinates axes.

A logarithmic scale will compress the high values and dilate the low values, on both the abcissa and the axis of ordinates, so that the nodes in lowlights get more space on the graph and can be controlled more clearly.

Increase the ‘scale for graph’ slider to set the base of the logarithm used to scale the axes. This allows you to control the amount of compression/dilation operated by the scale. If you draw purely exponential or logarithmic functions from identity lines, setting this value defines the base of such functions.

3.4.6. wavelets

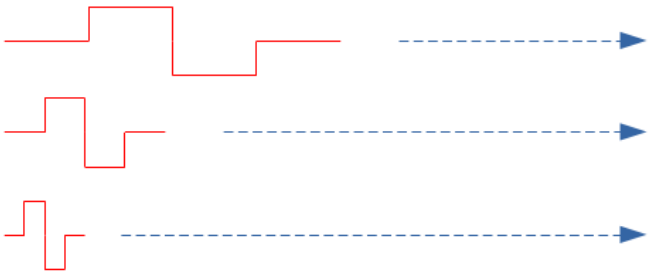
Wavelets are a technique used in image processing to separate (or *decompose*) an image into a number of distinct *layers*, each containing a different level of detail. After decomposing an image in this way, a module can limit its processing to one or more of these detail layers, and then piece the layers back together again at the end to form its output image. This allows us to be surgical about which features in the image we wish to impact when working with a module.

Some of the operations darktable can perform in this way are:

- noise removal (in the *denoise (profiled)* , *raw denoise* and *contrast equalizer* modules)
- contrast adjustment (in the *contrast equalizer* module)
- blurring or removal of unwanted detail (in the *retouch* module)

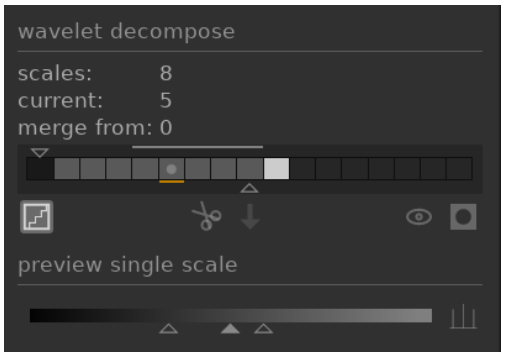
theory

A wavelet is a mathematical function that starts off at zero, oscillates up and down and then settles back to zero. The following diagram shows some simple wavelets of differing sizes.



These wavelet functions are used to scan across and down the image using a mathematical operation called *convolution*. This picks out details from the image that are on a similar scale to the size of a given wavelet, and builds a number of detail layers each corresponding to a different wavelet scale.

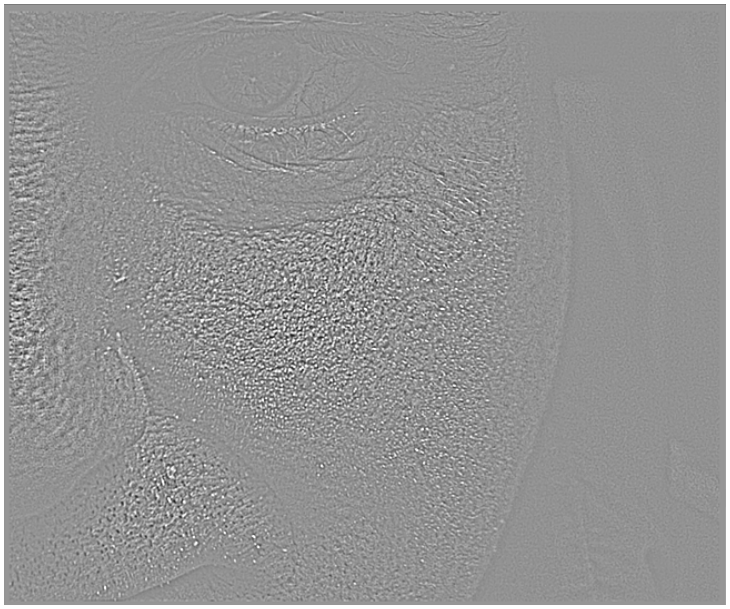
Below is an example where detail layers have been extracted from the image shown above. In this case, the images were produced using the *retouch* module, splitting the image into 8 different layers, and using the module's controls to visualise the details present at each of these wavelet scales:



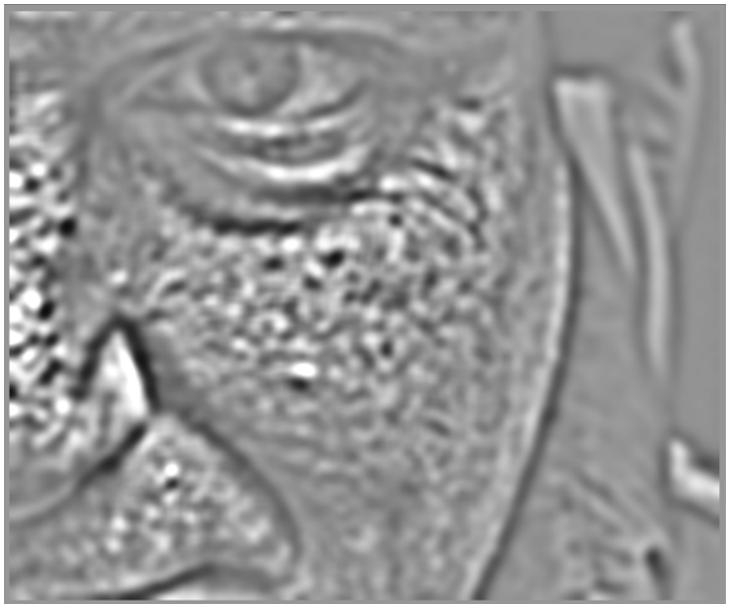
The bars in the *wavelet decompose* section indicate the layers that have been extracted at different wavelet scales. The darkest rectangle at the left represents the entire image (before decomposition) and the gray boxes represent each of the decomposed layers. Clicking on the staircase icon below the bar graph enables the layer visualisation overlay so that you can see what the currently selected layer looks like.

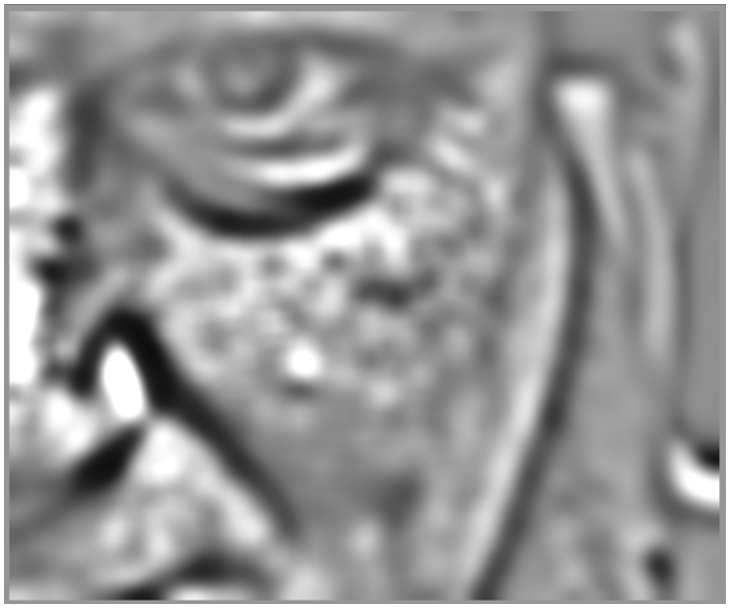
Let's take a look at some of the layers generated for the above image.

At scale #2, the image contains only very fine details, including the model’s eyebrows, eye lashes and the pores of his skin. It doesn’t include the coarser details of the image, because those details have been extracted to other layers:



At scales #5 and #6 we begin to see larger and larger features:



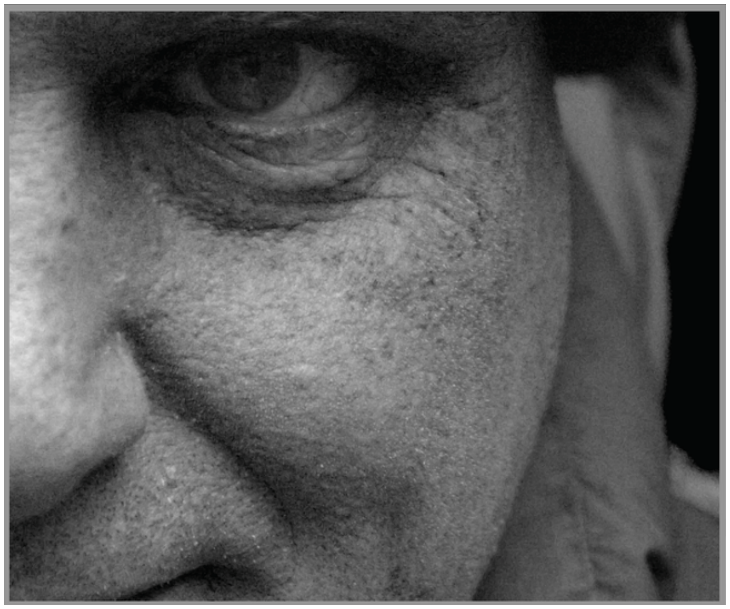


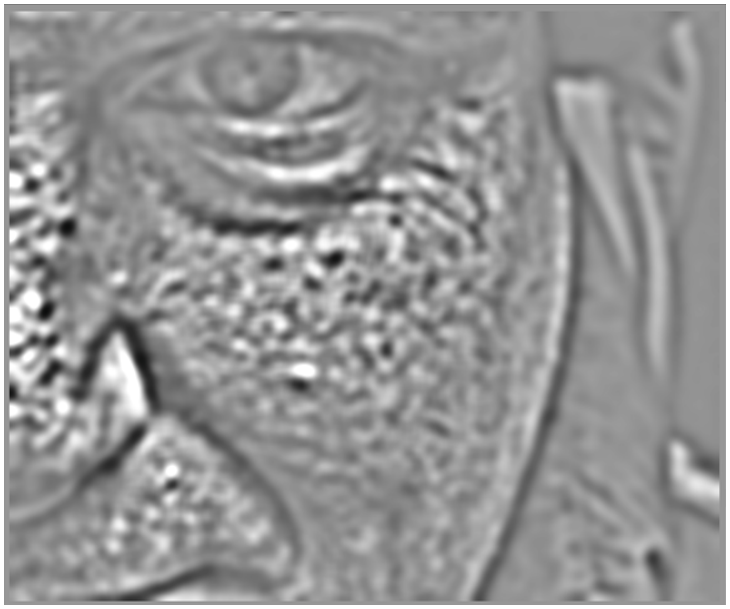
By scale #8 we only see very high-level features such as the overall shape of the model’s nose, eye and the cheek:

why use wavelets?

Suppose, in the above example, that we wanted to smooth out some of the blotchiness in the model’s skin, without losing any of the underlying skin texture. Wavelet decomposition makes this a trivial operation - we can simply use the retouch module to apply a Gaussian blur to only the ‘blotchy’ detail layer(s), leaving all other detail layers untouched. Once the adjustment is complete, the retouch module simply recombines that adjusted layer with the remaining untouched layers to produce the final image.

The sequence of images below shows (1) The original image; (2) The layer (scale 5) that we wish to blur; and (3) The final image after the scale 5 layer has been blurred and the layers recombined:





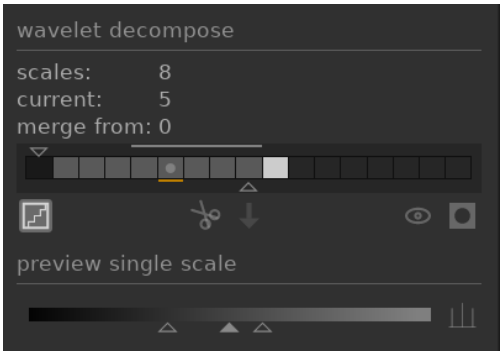
As you can see, the large-scale skin blotches have been removed, but the smaller-scale details remain untouched.

interacting with wavelet scales

There are two methods by which processing modules allow you to modify their operation using wavelet scales.

wavelet decomposition

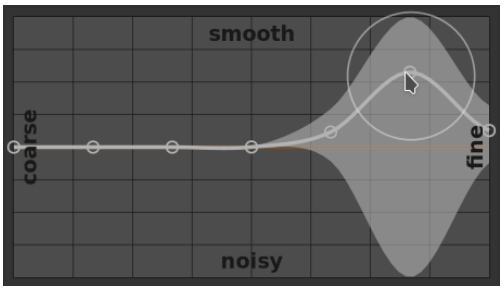
As discussed above, the *retouch* module allows you to choose how many detail levels to split your image into. It decomposes the image into separate layers and allows you to perform operations selectively on each individual layer or on the image as a whole:



See the *retouch* module documentation for more details.

spline controls

The *denoise (profiled)* , *raw denoise* and *contrast equalizer* modules allow their effects to be applied more or less to different wavelet scales using *splines*.



Here, each node in the graph represents a different level of detail in the image, from coarse detail on the left to fine detail on the right. You can raise or lower each of these nodes with your mouse to increase or decrease the module's effect, respectively, on that wavelet scale.

To modify the curve, click slightly above or below the line near to a node and drag up or down. You can change the width of your modification by scrolling with your mouse wheel, which increases or reduces the size of the circle displayed under your mouse pointer. A small circle indicates that the effect of dragging the curve up or down will be isolated mostly to the node being adjusted. A larger circle indicates that the effect will be more broad and will increasingly impact adjacent nodes. When you hover your mouse over the graph, shaded areas indicate the parts of the spline that will be impacted when you attempt to modify the curve.

3.5. masking & blending

3.5.1. overview

By default each module takes its input from the preceding module in the pixelpipe, performs its operation on the image data, and hands the output over to the next module in the pixelpipe.

A module's output data can optionally be reprocessed (combined) with its input data before being handed to the next module. This additional processing step is called *blending*. Input and output data is reprocessed using algorithms called blending operators or blend modes .

Each blend mode is further controlled by the *opacity* parameter, having a value between 0% and 100%, which defines how much the input and output images contribute to the final result. Typically an opacity of 0% outputs an image which is identical to the input image (the module has no effect) whereas an opacity of 100% delivers the maximum effect of the module.

The opacity can be the same for every pixel in the image (by simply combining a blend mode with the single opacity slider) – in this case blending acts uniformly on the entire image. Alternatively the opacity values can vary based on the properties or location of each pixel. This local modification of opacity is called a *mask*. Masks provide the user with fine control over which parts of an image are affected by a module and to what extent. At your choice you may activate a drawn mask , a parametric mask or a combination of the two .

Blending and masking functionality is controlled via a group of icons located at the bottom of each applicable module.

These icons enable the following masking and blending options, from left to right:

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

uniformly
Input and output images are reprocessed to the same extent for all pixels with the extent of the blending controlled by a single opacity slider. Additional controls to choose the blend mode and opacity 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 based on pixel location as defined with a drawn mask. Additional controls are displayed which allow you to draw a mask using one or more shapes. If no mask elements are drawn then all pixels have the same opacity, as defined by the opacity slider.

parametric mask

Reprocessing takes place with the chosen blend mode and opacity based on the properties of individual pixels. Additional controls are displayed that allow you to adjust the opacity on a per-pixel basis determined by pixel values.

drawn & parametric mask

Reprocessing takes place with the chosen blend mode and opacity based on a combination of a drawn mask parametric mask.

raster mask

Reprocessing takes place with the chosen blend mode and opacity based on a mask that was generated within a different module

blending options

Choose which color space to use when calculating the blending mask, and specify whether or not to allow a mask to be generated based on the module's output channels (normally a parametric mask is generated based on the input channels coming into the module). The following options are available:

- *reset to default blend colorspace*: Use the default color space for the module to specify the parametric mask.
- *Lab*: Use the Lab color space (where available) when specifying the parametric mask.
- *RGB (display)*: Use the display-referred RGB/HSL color space to specify the parametric mask.
- *RGB (scene)*: Use the scene-referred RGB/$J_zC_zh_z$ color space to specify the parametric mask.
- *show output channels*: Show the parametric mask output channel controls, so that the parametric mask can be defined in terms of the module's output channels.

Note: Not all of these blend options are available for every module.

3.5.2. blend modes

Blend modes define how the input and output of a module are combined (blended) together before a module's final output is passed to the next module in the pixelpipe.

Classic blending modes, designed for display-referred RGB (constrained to 0-100%) implicitly define a fulcrum at 50% (gray) or 100% (white) in their algorithms, depending on the blend mode. Because scene-referred is not subject to these restrictions, this fulcrum needs to be explicitly defined by the user when performing blending operations in the "RGB (scene)" color space. The additional *blend fulcrum* parameter will be presented to the user when using one of these blend modes in this color space. The effect depends on the operator used. For example, values above the fulcrum might be brightened and values below darkened, or vice versa.

The final output of a module is computed 'per-pixel' as follows:

```
final_output = (1.0 - opacity) * module_input + opacity * blended_output
```

where the `blended_output` is a combination of the input and output images, depending on the blend mode (below), and the `opacity` is defined 'per-pixel' by a combination of the mask and global opacity parameter.

normal modes

normal

The most commonly used blend mode, "normal" simply mixes input and output to an extent determined by the opacity parameter. This mode is commonly used to reduce the strength of a module's effect by reducing the opacity. This is also usually the blend mode of choice when applying a module's effect selectively with masks. This mode is also known as the "over" Porter-Duff alpha blending operator (see Alpha compositing for more details).

normal bounded

not available in the "RGB (scene)" color space

This blend mode is the same as "normal", except that the input and output data are clamped to a particular min/max value range. Out-of-range values are effectively blocked and are not passed to subsequent 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 to be properly handled later. The "normal" blend mode is therefore usually preferred.

arithmetic modes

addition
Add together the pixel values of the input and output images, lightening the output. When blending in the “RGB (scene)” color space, the pixel values of the output image are multiplied by a value proportional to the “blend fulcrum”.

subtract
Subtract the pixel value of the *output* from the *input*. When blending in the “RGB (scene)” color space, the pixel values of the output image are multiplied by a value proportional to the “blend fulcrum”. Pixel values less than 0 are set to 0.

subtract reverse
only available in the “RGB (scene)” color space
Subtract the pixel value of the *input* from the *output*. In this case, the “blend fulcrum” parameter of the “RGB (scene)” color space is applied to the input image.

multiply
Multiply the pixel values of the input and output together. When blending in display-referred color spaces, pixel values are between 0 and 1.0, the final output will be clamped and will always be darker. When blending in the “RGB (scene)” color space, this value is further multiplied by a value proportional to the “blend fulcrum”. In this case, values may be greater than 1.0 and therefore brighten the base image. This may have other side-effects, such as updating the white point in the filmic module.

Multiply blending simulates an optical variable density filter, where the density is defined by the output of the module. It has many applications, from blooming and local contrast enhancements (when used with a blur or low-pass filter) to dodging/burning and global contrast enhancements (when used with exposure). The fulcrum sets the output intensity threshold between darkening and brightening (any RGB value below fulcrum will darken).

multiply reverse
only available in the “RGB (scene)” color space
This mode works in the same way as the “multiply” mode except that in the `final_output` equation given at the top of this page, the `module_input` parameter is replaced with `module_output` (the output of the module before blending is applied).

divide
Divide the pixel values of the input by the output. When blending in the “RGB (scene)” color space, the pixel values of the output image are multiplied by a value proportional to the “blend fulcrum”.

Since this is the inverse of the multiply mode, it will darken where multiply brightens and vice versa. Everything else works in essentially the same way.

divide reverse
only available in the “RGB (scene)” color space
Divide the pixel values of the *output* by the *input*. In this case, the “blend fulcrum” parameter of the “RGB (scene)” color space is applied to the input image.

screen
not available in the “RGB (scene)” color space
Invert the input and output pixel values, multiply those values together and invert the result. This yields approximately the opposite effect to “multiply” mode – the resulting image is usually brighter, and sometimes “washed out” in appearance.

average
Return the arithmetic mean of the input and output pixel values.

difference
Return the absolute difference between the input and output pixel values.

geometric mean
Return the square root of the product of the input and output pixel values.

harmonic mean
Return the product of the input and output pixel values, multiplied by 2 and divided by their sum.

contrast enhancing modes

The following modes are not available in the “RGB (scene)” blending color space as they rely on an assumption of “50% mid gray” which only applies to display-referred and non-linear color spaces.

overlay
This mode combines the “multiply” and “screen” blend modes: The parts of the input where the output is brighter, become brighter; The parts of the image where the output is darker, become darker; Mid-gray is unaffected. This mode is often combined with the *lowpass* and *highpass* modules.

softlight
This mode is similar to “overlay”, except the results are softer and less bright. As with overlay, it is often combined with the *lowpass* and *highpass* modules.

hardlight
This mode is not related to “softlight” in anything but name. Like overlay mode it is a combination of “multiply” and “screen” modes and has a different effect above and below mid-gray. The results with hardlight blend mode tend to be quite intense and usually need to be combined with a reduced opacity.

vividlight
This mode is an extreme version of overlay/softlight. Values darker than mid-gray are darkened; Values brighter than mid-gray are brightened. You will probably need to tone down its effect by reducing the opacity

linearlight
This mode is similar to the effect of “vividlight”.

pinlight
This mode performs a darken and lighten blending simultaneously, removing mid-tones. It can result in artifacts such as patches and blotches.

color channel modes

Lab channels

The following are available for blending in the Lab color space only

Lab lightness
Mix the lightness from the input and output images, while taking the color channels (a and b) unaltered from the input image. In contrast to “lightness” this blend mode does not involve any color space conversion and does not clamp any data. In some cases this blend mode is less prone to artifacts than “lightness”.

Lab a-channel
Mix the Lab “a” color channel from the input and output images, while taking the other channels unaltered from the input image.

Lab b-channel
Mix the Lab “b” color channel from the input and output images, while taking the other channels unaltered from the input image.

Lab color
Mix the Lab color channels (a and b) from the input and output images, while taking the lightness unaltered from the input image. In contrast to “color” this blend mode does not involve any color space conversion and does not clamp any data. In some cases this blend mode is less prone to artifacts than “color”.

RGB channels

The following are available when blending in RGB color spaces only.

RGB red channel
Mix the “red” channel from the input and output images, while taking the other channels unaltered from the input image.

RGB green channel
Mix the “green” channel from the input and output images, while taking the other channels unaltered from the input image.

RGB blue channel
Mix the “blue” channel from the input and output images, while taking the other channels unaltered from the input image.

HSV channels

The following are available when blending in the “RGB (display)” color space only.

HSV lightness

Mix the lightness from the input and output images, while taking color unaltered from the input image. In contrast to “lightness” this blend mode does not involve clamping.

HSV color

Mix the color from the input and output images, while taking lightness unaltered from the input image. In contrast to “color” this blend mode does not involve clamping.

others

lightness

Mix lightness from the input and output images, while taking color (chroma and hue) unaltered from the input image.

chrominance

Mix chrominance from the input and output images, while taking lightness and hue unaltered from the input image. This blend mode uses RGB ratios, divided by a Euclidean norm.

lighten

not available in the “RGB (scene)” color space

Compare the pixel values of the input and output images, and output the lighter value.

darken

not available in the “RGB (scene)” color space

Compare the pixel values of the input and output images, and output the darker value.

hue

not available in the “RGB (scene)” color space

Mix hue (color tint) from the input and output images, while taking lightness and chroma unaltered from the input image.

color

not available in the “RGB (scene)” color space

Mix color (chroma and hue) from the input and output images while taking lightness 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.

coloradjustment

not available in the “RGB (scene)” color space

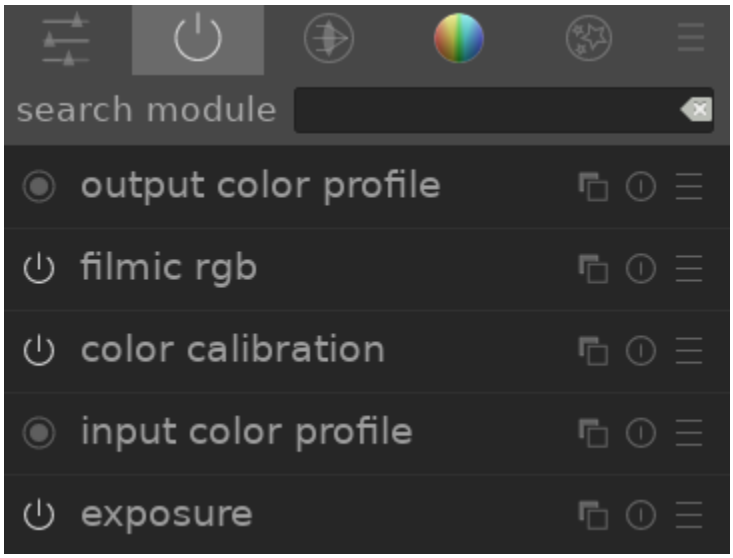
Some modules act predominantly on the tonal values of an image but also perform some color saturation adjustments (e.g. the *levels* and *tone curve* modules). This blend mode takes the lightness from the module’s output and mixes colors from input and output, enabling control over the module’s color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

3.5.3. Masken

3.6. organization

3.6.1. overview

Processing modules are organized and accessed via the right-hand panel in the darkroom:



Click on the icons at the top of this panel to reveal, from left to right,

 **quick access panel**

A customizable panel allowing quick access to commonly-used module controls.

 **active modules**

A group containing the modules that are currently active in the pixelpipe.

module groups

One or more groups of processing modules. These groups are user-defined but some default groupings are provided as presets.

presets menu

A menu that allows you to access stored module layout presets and create your own (via the “manage presets..” option).

Click once on a module group icon (including the active group) to show only the modules in that group. Click a second time to show all available modules.

You can change which widgets appear in the quick access panel, and which modules appear in the module groups, by right-clicking on the appropriate icon.

search

Below the module group icons is the search bar, which you can use to access any of the processing modules, regardless of whether they are currently in a group. Some modules have additional tags stored against them, which will also be queried when searching.

3.6.2. module groups

A number of pre-defined module groups are shipped with darktable and are selectable as presets. These are summarized below.

All of these presets (with the exception of *modules: deprecated* and *search only*) also include the quick access panel. All except the *modules: deprecated* group include the search bar.

modules: all

This preset contains all modules, sorted according to the traditional module groupings used prior to darktable 3.4, as follows:

 **base modules**

A minimal set of modules normally required to render a presentable image.

 **tone modules**

Other modules relating to tone levels and contrast

 **color modules**

Modules relating to color grading and color profiles

 **corrective modules**

Modules relating to correcting problems relating to lens distortions, sensor noise, sharpening, etc..

 **(special) effects modules**

Includes special effects such as retouch, liquify, bloom, sharpen, etc..

modules: default

This preset is the default module group layout from darktable 3.4 onwards and consists of a new simplified set of module groups, as follows:

 **technical modules**

Modules that deal with technical issues relating to the physics of sensors and denoising, lenses and associated corrections, color profiles, dynamic range and tone mapping, and recovering from damage to the image by physical limitations (hot pixels, clipped highlights, etc.)

 **grading modules**

Modules concerned with primary (corrective) and secondary (creative) subjective corrections of colors and tones

 **(special) effects modules**

“Special effect” modules such as retouch, liquify, bloom, sharpen, etc..

workflow: scene-referred & *workflow: display-referred*

These presets define groups of modules relevant to the scene-referred and display-referred workflows, sorted into groups as shown below:

 **base modules**

A basic set of modules to adjust the cropping/orientation, adjust the exposure, and apply tone mappings and contrast as appropriate to the workflow.

 **color modules**

Modules relating to color grading and color saturation.

 **corrective modules**

Modules relating to correcting problems relating to lens distortions, sensor noise, sharpening, retouching, etc..

 **(special) effects modules**

Includes special effects such as watermarks, framing, vignetting, etc..

workflow: beginner

This preset provides a minimal set of modules targeted as a starting point for beginners. It is suggested that beginners start by copying this minimal preset, and add to it as they gain experience with additional modules.

 **base modules**

A basic set of modules to adjust the cropping/orientation, adjust the exposure, and apply a basic tone mapping via base curve or basic adjustments.

 **grading modules**

Modules dealing with creative tone and color grading.

 **(special) effects modules**

Includes special effects such as retouch, sharpen, watermarks, etc..

previous config

These presets are automatically generated for users who have upgraded from a version of darktable prior to 3.4. Where you have previously set up favourites or altered the *hidden* flag on modules, these presets contains those customisations, retaining the legacy module groups (*previous config* preset) or new module groups (*previous config with new layout* preset).

If favourites were created in prior versions these will remain available in an additional group:

 **favourite modules**

This group was previously used by users to make it easier to find frequently-used modules, and is available under the “previous config” presets. New users can, of course, still create their own custom group and name it “favourites” if they so desire.

search only

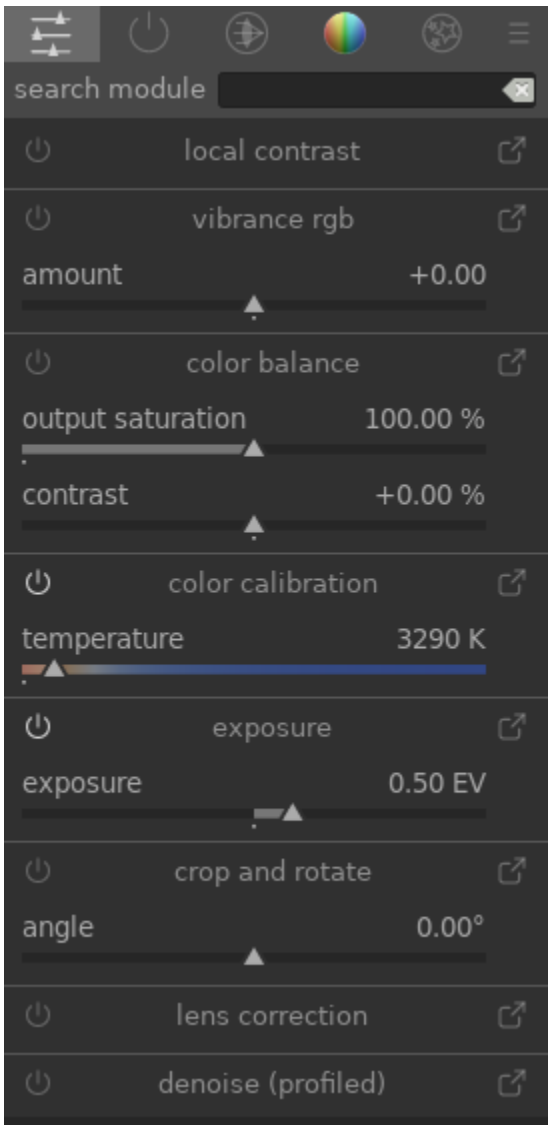
This preset does not include any module groupings. Modules may only be accessed using the search facility.

modules: deprecated

This preset contains a list of deprecated modules. This is the only way to access deprecated modules for new edits but be warned: these modules will be removed for new edits in the next release of darktable. This group cannot be duplicated and the modules within it cannot be added to user-created groups.

3.6.3. quick access panel

The quick access panel allows you to access widgets from a number of processing modules all in one place.



You can add new widgets by right-clicking on the icon at the top-left or by using the manage module layouts screen.

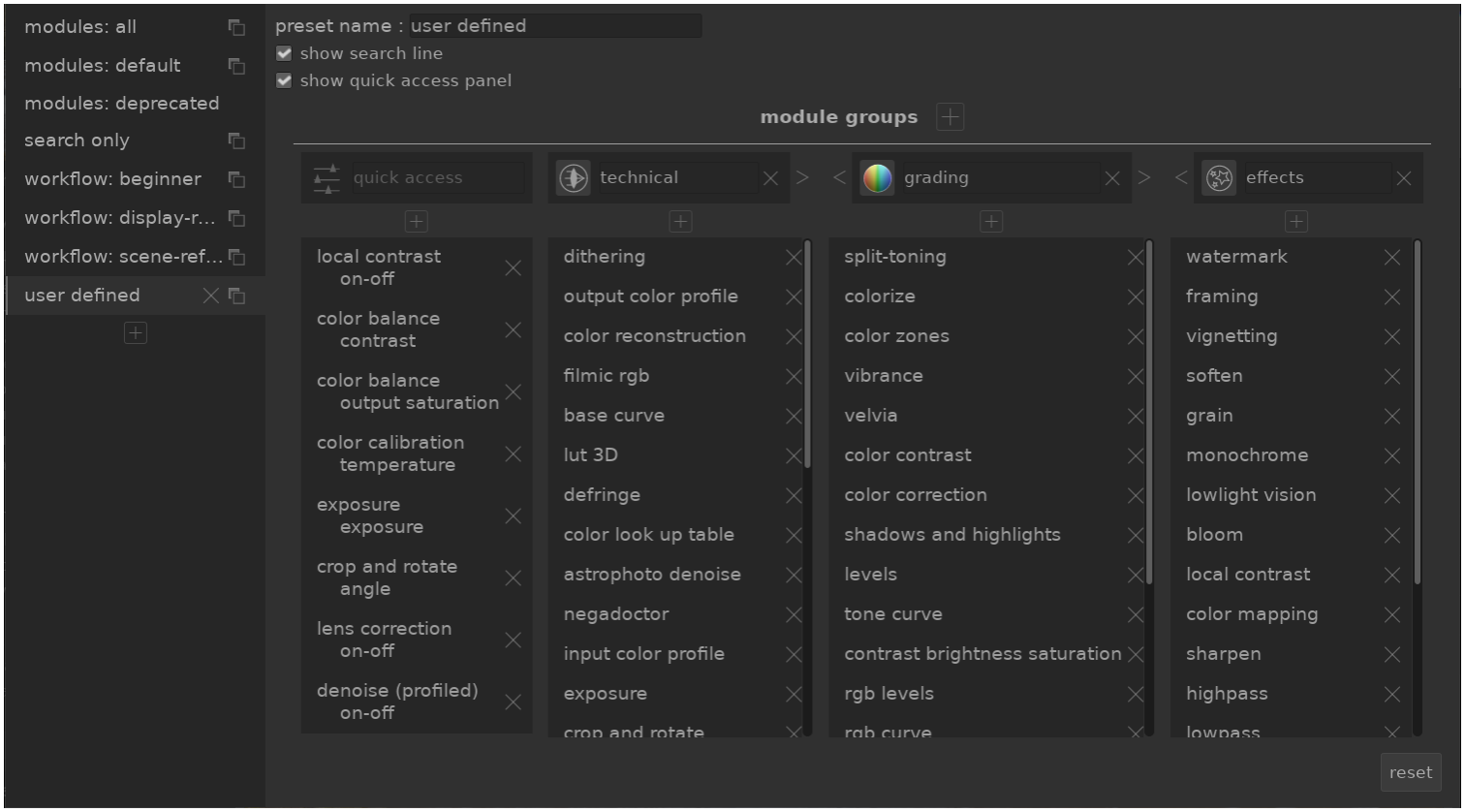
Click on the icon to the right of the module name to open the full version of that module. Click on the icon to the left of the module name to enable/disable the module.

Currently only some widgets may be added to this panel (check boxes, sliders, comboboxes) – more complex widgets such as curves are not supported.

If any module has multiple instances enabled you will no longer be able to manage that module's widgets through the quick access panel.

3.6.4. manage module layouts

Manage the layout and grouping of processing modules and the quick access panel.



This maintenance screen can be accessed from the *presets* menu beside the module search box or module group icons (below the histogram in the darkroom view).

module controls

preset list (left panel)

The left-hand panel lists the currently-defined module layout presets. Those at the top of the list are fixed and cannot be altered. User-defined presets are placed at the bottom of the list.

create
To create a new preset, populated with a minimal list of basic modules, click on the + button at the bottom of the left-hand panel. Alternatively click on the *duplicate* button to the right of one of the existing module presets to copy that preset to a new one. The above screenshot shows a new preset named “user defined” which has been created by duplicating the “modules: default” preset.

delete
Delete a user-defined preset with the X button.

rename
Rename a preset using the text entry box at the top of the right-hand panel. Right-click to bring up a menu which can be used to copy, paste, select all, delete or insert an emoji)

preset settings (right panel)

Change the settings for the selected preset

global controls

At the top of the right panel, the following options can be set

preset name
Choose a name for your preset.

show search line
Choose whether to display the search bar below the module group icons

show quick access panel
Choose whether to display the quick access panel. If checked, a new entry will appear in the bottom-right panel.

quick access & module groups

The bottom section of the right-hand panel allows you to create and amend module groups for the selected preset.

add a group
Click on the + sign beside the “module groups” header to add a new group.

remove a group
Remove a group by clicking on the X button beside the header of that module group.

add a module/widget
Add a module to a group, or a widget to the quick access panel, by clicking the + sign below the group name. Select the required module/widget from the list that appears.

remove a module/widget
Remove a module from a group, or a widget from the quick access panel, by clicking the X beside the module/widget name.

change a group’s icon
Change the icon assigned to a group by clicking on the existing group icon and selecting a new one from the drop-down list.

change the group order
Change order the groups are displayed in by clicking on the < and > buttons beside the group headers.

rename a group
Rename a module group using the text entry box at the top of the module list. Right-click to bring up a menu which can be used to copy, paste, select all, delete or insert an emoji)

Settings will be automatically saved when you exit the screen. Click *reset* to reset the settings to the start of your editing session.

4. Tethering

4.1. overview

The tethering view allows you to capture images directly into darktable from a connected camera.

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 your camera is mounted or viewed automatically, you will need to “unmount/eject” the camera before darktable can access it. This unlocks the camera so that darktable can take control of it – darktable will then re-lock the camera so that it cannot be used by other applications.

After the USB cable is connected, look at the import module in the lighttable view. If your camera is not visible in this module, click the “scan devices” button and it should appear with two functions available: “import from camera” and “tethered shoot”. Click “tethered shoot” to enter the tethering view.

darktable uses gphoto2 to interface with your camera. If you have problems finding the connected camera as described above, check the troubleshooting section in this chapter to verify that your camera has tethering support.

In the center view, images are shown while you capture them. You can capture an image by either using darktable’s user interface or by manually triggering a capture with your camera. If you are using Live View the image will be shown in darktable’s center view.

When entering tethering view, a film roll will be created using the same structure as defined for camera import (see preferences > import > session options). The job code will be predefined as “capture”.

If you want to group your captures into different film rolls, you should use the session module in the right-hand panel to set a different job code. When entering a new name and pressing Enter, a new film roll will be created and newly-captured images will be added into this new film roll.

darktable provides some useful tools to set up an image capture in the user interface. You can set up timelapse captures, brackets for HDR and even sequential captures of bracketed images. For more information read the documentation on the camera settings module and the examples later in this chapter.

4.2. tethering view layout

left panel

image information
Display image information

right panel

From top to bottom:

histogram
The histogram for the current image
session
Session settings
live view
Live view settings
camera settings
Camera settings
metadata editor
Edit metadata for selected images.
tagging
Tag selected images.

bottom panel

From left to right:

star ratings
Apply star ratings to images
color labels
Apply color categories to images

4.3. examples

This section contains examples of typical usages of tethering.

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 like, 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 preview the captures with the client instead of fumbling around with your camera.

Working in the tethering view 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.

capturing a timelapse

A timelapse is a video clip composed of images taken in a timed 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 . Now decide if you want to shoot in manual or auto mode. Only use auto in situations were the ambient light will change significantly during the time of the shoot. For example, shooting a timelapse over 24 hours might give you easier control of light in that kind of captured sequence.

The camera settings module 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 the center view.

4.4. troubleshooting

This troubleshooting guide will provide you with some steps to verify whether your camera can be used with tethering. This is done using the gphoto2 command-line tools. This is what darktable uses to interface with your camera.

Find your camera port name to use it in the following tests. Usually the port "`usb:`" will be enough and therefore is used in these examples.

is your camera detected?

The following command will verify that your camera is connected to the computer and detected by gphoto2.

```
env LANG=C gphoto2 --auto-detect
```

check the camera’s driver abilities

Execute the following command and ensure that the capture choices ability supports “Image” and configuration support is “yes”. darktable will check these two abilities to decide if the “tethered shoot” button should be shown or not.

```
env LANG=C gphoto2 --port usb: --abilities
```

remote capture

This step will verify that your camera can be remotely controlled – i.e. 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
```

tethered capture

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
```

now what?

If any of the above steps failed, there are problems with your specific camera and driver. Please report the issues to the gphoto2 mailing list for further help. You can find the mailing list at www.gphoto.org. 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 on the bugtracker. 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
```

5. Map

5.1. overview

The map view is where you geo-tag your images.

Map view shows a world map with the currently open image or film roll of images, pinned to their geo-tagged location. This requires that the image was geo-tagged 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 geotagging module, which is available in both the lighttable and map views.

center map view

In the center of the map view you will see a map.

Map data is taken from open map sources on the internet. New map data is only available if you are connected to the internet – darktable keeps a disk cache of previously loaded map data.

You can navigate within the map using your mouse. Left-click and drag to move the map. Use the scroll-wheel to zoom in and 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. Images close to each other are grouped and a count of grouped images is displayed on the bottom-left corner.

In order to assign geo coordinates to an image, activate the filmstrip on the lower panel (press Ctrl+F). You can assign a geo location to an image by dragging the image icon from the film-strip and positioning it on the map. darktable will record the new location (longitude and latitude) as part of the image metadata. This data will be included in exported images.

In order to remove location data from an image simply drag it from the map and drop it onto the filmstrip.

Images close to each other are grouped under a single image group. You can use the map settings module to control the grouping as needed. The number displayed on the bottom left of the thumbnail gives the number of images inside the group. A white number means that all images of the group are exactly at the same location. A yellow number means that the locations of images within the group are not all the same. Use the mouse scroll wheel while hovering over a group of images to scroll through the thumbnails of the images in that group.

Normally images in the center map view have black borders. If an image is selected in the filmstrip, then the corresponding image on the map will be highlighted with a white border.

Click+drag to adjust the position of an image. Shift+click to move a complete group of images.

On the left and right of the central map are panels that provide additional controls (see map view layout).

undo/redo

All image movements in the map view are recorded by darktable. It is possible to undo or redo such changes to recover a previous state. Note that this undo/redo facility is unlimited while moving images but it is reset each time you leave the map view.

Press Ctrl+Z to undo the last modification and Ctrl+Y to redo the last undone modification (if any).

5.2. map view layout

left panel

These modules are identical to the lighttable view. From top to bottom:

collect images
Filter the list of images displayed in the map view.
recently used collections
View recently used collections of images.
image information
Display image information.

right panel

Here you can find the modules specific to the map view. From top to bottom:

find location
Search for a place on map.
locations
Manage a hierarchical list of location tags and their corresponding regions on the map.
map settings
Set up your map overlay information and map providers.
tagging
Tag selected images.
geotagging
Apply GPX track data to selected images.

bottom panel

filmstrip
Drag images from the filmstrip onto the map as described in the *find location* module documentation.

6. Slideshow

6.1. overview

The slideshow view allows you to watch a slideshow of your current collection with the associated filtering rules and sort order applied.

To learn more about how to define the collection and filtering rules, see the section on collections . Select the sort criteria and sort order of the images in the top panel .

The next section provides more details on the usage of the slideshow view.

6.2. usage

The slideshow view is still in an early stage of development with only a basic set of features.

If you don’t need the auto-advance mode, you could even use the sticky preview feature instead.

spacebar	start and stop auto-advance mode which automatically switches to the next images every five seconds by default.
ESC	leave slideshow mode and return to lighttable view.
+ or up arrow	increase delay between each image.
- or down arrow	decrease delay between each image.
left-click or right arrow or right shift-key	switch to the next image of the collection.
right-click or left arrow or left shift-key	switch to the previous image of the collection.

Hint: To take full advantage of your screen size, put darktable into fullscreen mode by pressing F11 and hide the border-controls by pressing the B key.

---

7. Drucken

8. Modul Referenz

9. Preferences & Settings

9.1. overview

darktable comes with a number of user-configurable preferences. These can be adjusted in the preferences dialog, which can be reached by clicking the gear icon on the right of the top panel . Preferences are separated into tabs, each of which is described in more detail in the following sections.

When opened from within the lighttable or darkroom views, the appropriate tab will be loaded by default to allow you to modify that view's settings.

Any altered settings (i.e. those that differ from their default state) are highlighted with a bullet beside them. If you change a setting that needs a restart to take effect, a message will appear as a reminder after you exit the preferences dialog.

Double click on a setting's label to reset to its default value.

The preferences dialog can be closed by clicking on the close button in your window manager or pressing the Escape key.

9.2. general

The preferences in this tab control the overall look and feel of darktable.

interface language
Set the language of the user interface. The system default is marked with an * (needs a restart)

theme
Set the theme for the user interface. Apart from any aesthetic considerations, the recommended interface color for color evaluation is middle gray. Indeed, visual perception is affected by ambient brightness, and a low brightness of the interface causes all kinds of illusions. Using a dark interface to retouch photos can therefore lead to excessive retouching (abuse of contrast and saturation) and to a photo that is too dark once printed. It is therefore highly recommended that you use one of the "gray" themes for retouching work as these are designed so that the user interface approximates middle gray (default "darktable").

performance mode
Enable this option to render thumbnails and previews at a lower quality. This increases the rendering speed by a factor of 4, and is useful when working on slower computers (default off). This also improves the performance of the slideshow image rendering.

use system font size
Select this option to use the font size defined by your system. If unchecked, you may enter a custom font size in the box below (default on).

font size in points
If the "use system font size" option is switched off, enter a font size (in points) for darktable to use. The font size will be changed immediately.

GUI controls and text DPI
Adjust the global GUI resolution to rescale controls, buttons, labels, etc. Increase for a magnified GUI, decrease to fit more content in the window. Set to -1 to use the system-defined global resolution. Default is 96 DPI on most systems. (needs a restart)

CSS Theme Modifications

In addition to selecting a pre-built theme you can also apply additional CSS customisations of your own to tweak the look-and-feel of darktable. A text entry box is provided for this purpose.

When you have finished entering your CSS tweaks, click on the ‘save CSS and apply’ button. This will save your CSS to a file (`$HOME/.config/darktable/user.css`) and immediately apply it to the current darktable session.

If your changes cause any issues, you can uncheck the “modify selected theme with CSS tweaks below” check box. This will immediately restore the base theme but will leave your tweaks in the editor so that you can re-edit and try again. Simply press “save CSS and apply” again when you are ready to retry. This will automatically re-check the “modify selected theme with CSS tweaks below” checkbox and apply the new CSS.

If you have any issues with darktable please retry with this option unchecked to be certain that they were not caused by any of your changes.

9.3. import

This tab contains a number of default settings for the lighttable import module.

session options

These options define a naming pattern to organize images on disk when importing from a connected camera and when taking photos in the tethering view.

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:

```
$(HOME)              the home folder as defined by the system
$(PICTURES_FOLDER)   the pictures folder as defined by the system (usually “$HOME/Pictures”)
$(DESKTOP)           the desktop folder as defined by the system (usually “$HOME/Desktop”)
$(USERNAME)          your user account name on the system
$(FILE_NAME)         basename of the imported image
$(FILE_EXTENSION)    extension of the imported image
$(JOBCODE)           unique identifier of the import job
$(SEQUENCE)          a sequence number within the import job
$(MAX_WIDTH)         maximum image width to limit within export session
$(MAX_HEIGHT)        maximum image height to limit within export session
$(ID)                unique identification number of the image in darktable's database
$(YEAR)              year at the date of import
$(MONTH)             month at the date of import
$(DAY)               day at the date of import
$(HOUR)              hour at the time of import
$(MINUTE)            minute at the time of import
$(SECOND)            second at the time of import
$(EXIF_YEAR)         year the photo was taken (from Exif data)
$(EXIF_MONTH)        month the photo was taken (from Exif data)
$(EXIF_DAY)          day the photo was taken (from Exif data)
$(EXIF_HOUR)         hour the photo was taken (from Exif data)
$(EXIF_MINUTE)       minute the photo was taken (from Exif data)
$(EXIF_SECOND)       seconds the photo was taken (from Exif data)
$(EXIF_ISO)          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)`).

keep original filename
Check this box to keep original the filename instead of using the pattern below when importing from a camera or card (default off).

file naming pattern
The file name part of the naming pattern (default `$(YEAR)$(MONTH)$(DAY)_$(SEQUENCE).$(FILE_EXTENSION)`.

9.4. lighttable

The following options control functionality in the lighttable view and modules.

general

hide built-in presets for utility modules
If enabled, only user-defined presets will be shown in presets menu for utility modules – built-in presets will be hidden (default off).

use single-click in the collect module
Enable “single click” mode in the collect images module, which allows ranges to be selected (default off).

expand a single utility module at a time
Controls how utility modules are expanded. If this option is enabled, expanding a module by clicking collapses any other currently expanded panel. If you want to expand a panel without collapsing the others you can do so with Shift+click. Disabling this option inverts the meaning of click and Shift+click (default off).

scroll to utility modules when expanded/collapsed
With this option enabled the side panels will scroll a utility module to the top of the panel when it is expanded or collapsed (default off).

rating an image one star twice will not zero out the rating
Normally clicking a one star rating twice will set a zero star rating to that image. Check this option to disable this functionality (default off).

show scrollbars for center view
Should scrollbars be shown in the center view of the lighttable (default on).

thumbnails

color manage cached thumbnails
If activated, darktable generates thumbnails in a general color space (AdobeRGB) in order to render them independently of the individual monitor. Conversion to the monitor color space is undertaken at display time. If this option is not activated thumbnails are stored directly in a monitor-specific color space at generation time and are subsequently displayed without further corrections (default on).

use raw file instead of embedded JPEG from size
When generating thumbnails for images that have not yet been processed in the darkroom, if the thumbnail size is greater than this value, generate it by processing the raw image data. If the thumbnail is below this size, use the JPEG preview image embedded in the raw file. Once an image has been processed in the darkroom, thumbnails will always be generated from raw data (you can revert back to the JPEG preview by discarding history). To render thumbnails with the best quality choose “always”.

high quality processing from size
If the thumbnail size is greater than this value and is being generated from raw data, it will be processed using the full quality rendering path, which is better but slower (default 720p). To render thumbnails with the best quality, choose “always”.

delimiters for size categories
Size categories are used to allow different thumbnail overlays to be shown depending on the thumbnail size. A pipe delimited set of values defines at what image sizes the categories change. The default value of “120|400” means that there will be 3 categories of thumbnails: 0-120px, 120-400px and >400px.

pattern for the thumbnail extended overlay text
If the user has chosen to show extended overlay text over thumbnail images, this setting allows the user to define what information is displayed. This pattern can use any of the variables defined in the variables section.

pattern for the thumbnail tooltip (empty to disable)

Defines what information is displayed in the tooltip when the mouse hovers over image thumbnails. This pattern can use any of the variables defined in the variables section.

9.5. darkroom

The following options control functionality in the darkroom view and associated modules.

general

pen pressure control for brush masks

Controls how the pressure reading of a graphics tablet impacts newly generated drawn mask brush strokes. You can control the brush width, hardness and 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 drawn mask brush strokes. Stronger smoothing leads to fewer nodes and easier editing at the expense of lower accuracy.

pattern for the image information line

Set the information to be displayed in the image information line . You can use any variables in the variables section as well as `$(NL)` for a new line.

position of the image information line

Choose the darkroom panel in which the image information line is displayed. Choose between “top left” “top right” “top center” “bottom” and “hidden” (default “bottom”).

border around image in darkroom mode

Process the center image in darkroom mode with a border of the given number of pixels (default 20).

show scrollbars for center view

Should scrollbars be shown in the center view of the darkroom (default off).

demosaicing for zoomed out darkroom mode

Choose how to demosaic images in the darkroom view when not viewing the image at 1:1 zoom scale

- *always bilinear (fast)* is fastest, but not as sharp
- *at most PPG (reasonable)* uses PPG + interpolation modes
- *full (possibly slow)* will use exactly the settings for full-size export

(default “*at most ppg (reasonable)*").

reduce resolution of preview image

Reduce the resolution of the navigation preview image (choose from “original”, “1/2”, “1/3” or “1/4” size). This may improve the speed of the rendering but take care as it can also hinder accurate color-picking and masking (default “original”).

show right-side buttons in processing module headers

Choose whether to show the three buttons (multi-instance, reset, presets) on the right-hand-side of the module header for processing modules. These buttons will always appear when the mouse is over the module. At other times they will be shown or hidden according to this preference selection:

- *always*: always show all buttons
- *active*: only show the buttons when the mouse is over the module
- *dim*: buttons are dimmed when mouse is away
- *auto*: hide the buttons when the panel is narrow
- *fade*: fade out all buttons when the panel narrows
- *fit*: hide all the buttons if the module name doesn’t fit
- *smooth*: fade out all buttons in one header simultaneously
- *glide*: gradually hide individual buttons as needed

(default *always*)

show loading screen between images

Show gray loading screen when navigating between images in the darkroom. Switch this option off to just show a simple toast message and leave the previous image in place until the next image is loaded. Note that switching this option off can be very useful to quickly compare duplicate images, however, there might be issues with long loading times (leading you to think the next image has already loaded) and you may observe visual artifacts while the next image is loading (default on).

modules

display of individual color channels

Controls how individual color channels are displayed when activated in the parametric masks feature. You can choose between “false color” and “gray scale” (default “false color”).

hide built-in presets for processing modules

If enabled, only user-defined presets will be shown in presets menu for processing modules – built-in presets will be hidden (default off).

expand a single processing module at a time

Controls how processing modules are expanded in the darkroom. 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 can do so with Shift+click. Disabling this option inverts the meaning of click and Shift+click (default on).

only collapse modules in current group

When choosing to expand a single processing module at a time (using the logic defined in the previous setting), only collapse other modules that appear in the current visible group. Disable this option to ensure that modules in non-visible groups are also collapsed (default on).

expand the module when it is activated, and collapse it when disabled

Select this option for the darkroom to automatically expand or collapse processing modules when they are enabled or disabled. (default off)

scroll to processing modules when expanded/collapsed

With this option enabled the darkroom side panel will scroll a processing module to the top when it is expanded or collapsed (default on).

white balance slider colors

Controls the appearance of the sliders in the white balance module.

- *no color*: background of the sliders is not colored at all.
- *illuminant color*: slider colors represent the color of the light source, i.e. the color you are adjusting to in order to achieve neutral white
- *effect emulation*: slider colors represent the effect the adjustment would have had on the scene. This is how most other raw processors show temperature/tint sliders colors.

(default *no color*)

colorbalance slider block layout

Controls the appearance of the shadows/mid-tones/highlights sections in the color balance module.

- *list*: all sliders are shown in one long list (with headers)
- *tabs*: use tabs to switch between the blocks of sliders
- *columns*: the blocks of sliders are shown next to each other (in narrow columns)

(default *list*)

show mask indicator in module headers

If enabled, an icon will be shown in the header of any processing modules that have a mask applied (default on).

9.6. other views

The following options control functionality in the map and slideshow views.

map / geolocalisation

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).

slideshow

waiting time before each picture in slideshow
The number of seconds before displaying the next image (default 5).

9.7. processing

The following options control how images are processed.

always use LittleCMS 2 to apply output color profile
If this option is activated, darktable will use the LittleCMS 2 system library to apply the output color profile instead of its own internal routines. This is significantly slower than the default but might give more accurate results in some cases.

If the given ICC is LUT-based or contains both a LUT and a matrix, darktable will use LittleCMS 2 to render the colors regardless of this configuration parameter's value (default off).

pixel interpolator (warp)
The pixel interpolator used for rotation, lens correction, liquify, crop and final scaling.

Whenever we scale or distort an image we have to choose a pixel interpolation algorithm (see wikipedia for details). For warping modules, darktable offers bilinear, bicubic or lanczos2. In general, bicubic is a safe option for most cases and is the default value.

pixel interpolator (scaling)
The pixel interpolator used for scaling. The same options are provided as for the warp modules, but with the addition of lanczos3.

lanczos3 can cause pixel overshoots leading to artefacts but sometimes gives a more crisp visual appearance. This option is therefore only provided for transforming (scaling) algorithms and is the default value.

3D lut root folder
Define the root folder (and sub-folders) containing Lut files used by the *lut 3D* module

auto-apply chromatic adaptation defaults
Choose which module is responsible for performing white balance adjustments (chromatic adaptation) by default. Select "legacy" (default) to perform basic chromatic adaptation within the *white balance* module only. Select "modern" to use a combination of the *white balance* and *color calibration* modules to perform modern chromatic adaptation with improved color science. These settings are applied by default to new edits and will not impact old edits.

auto-apply pixel workflow defaults
Choose which modules and module order to apply to new images by default:

- *scene-referred* workflow is based on linear RGB modules. Selecting this option will automatically enable *filmic rgb* and *exposure* modules will set the pixelpipe order to the (scene-referred) order defined for darktable 3.0 and later.

  The *exposure* module will include an automatic adjustment of +0.5 EV to adjust the mid-gray to match that of the majority of SLR cameras. This adjustment can be overridden with an automatically-applied preset if the default produces consistently dark images for your camera.

  Finally, this setting automatically enables the "compensate camera exposure" option in the *exposure* module to adjust the global brightness appropriately in cases where the camera's exposure compensation dial was used to protect highlights in the image or Expose To The Right (ETTR) to optimally make use of the sensor's dynamic range.

- *display-referred* (default) workflow is based on Lab modules and is the legacy mode from darktable 2.6 and earlier. Selecting this option will automatically enable the *base curve* module and set the pixelpipe order to the legacy (display-referred) order used by default up to version 2.6.

- *none* will not enable any modules by default and will set the pixelpipe to the (scene-referred) order defined for darktable 3.0 and later.

auto-apply per camera basecurve presets
Use the per-camera basecurve by default instead of the generic manufacturer one if there is one available. This should only be used in conjunction with the *display referred* workflow defined above (default off).

auto-apply sharpen
Auto-apply the sharpen module to new images by default. This option is not recommended on cameras without a low-pass filter. (default on, requires a restart).

detect monochrome previews
Enable this to analyse images during import and tag them with the `darkroom|mode|monochrome` tag if they are found to be monochrome. The analysis is based on the preview image embedded within the imported file. This makes for a more convenient workflow when working with monochrome images, but it slows down the import, so this setting is disabled by default.

9.8. security

The following options allow the user to switch on warnings when certain activities are undertaken.

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 discarding history stack
Always ask the user before the history stack of an image is discarded (default on).

send files to trash when erasing images
Instead of physically deleting images from disk put them into the system's trash bin (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).

ask before removing empty folders
Always ask the user before removing any empty folder. This can happen after moving or deleting images (default off).

ask before deleting a tag
Always ask user before deleting a tag from an image (default on).

ask before deleting a style
Always ask user before deleting a style (default on).

ask before deleting a preset
Always ask user before deleting a preset (default on).

ask before exporting in overwrite mode
Always ask before exporting images in overwrite mode.

9.9. cpu/gpu/memory

This preference tab contains mostly performance-related settings:

memory in megabytes to use for thumbnail cache
In order to speed-up the display of film rolls, darktable stores image thumbnails in a cache file on disk (primary cache) and loads it into memory at startup. This setting controls the size of the cache in megabytes. It needs a restart if changed (default 256MB).

enable disk backend for thumbnail cache
If activated, darktable stores all thumbnails on disk as a secondary cache, and thereby keeps thumbnails accessible if they get dropped from the primary cache. This needs more disk space but speeds up the lighttable view as it avoids reprocessing of thumbnails (default on).

enable disk backend for full preview cache

If enabled, darktable writes full preview images to disk (`.cache/darktable/`) when evicted from the memory cache. Note that this can take a lot of storage (several gigabytes for 20k images) and darktable will never delete cached images. It's safe to delete these manually if you want. Enabling this option will greatly improve lighttable performance when zooming an image in full preview mode (default off).

number of background threads

This controls how many parallel threads are used to create thumbnails during import. 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. 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. Needs a restart if changed (default 16).

activate OpenCL support

darktable can use your GPU to significantly speed up processing. The OpenCL interface requires suitable hardware and matching OpenCL drivers on your system. If one of those is not found the option is grayed out. Can be switched on and off at any time and takes immediate effect (default on).

OpenCL scheduling profile

Defines how preview and full pixelpipe tasks are scheduled on OpenCL enabled systems:

- *default*: the GPU processes the center view pixelpipe; the CPU processes the preview pipe;
- *multiple GPUs*: both pixelpipes are processed in parallel on two different GPUs;
- *very fast GPU*: both pixelpipes are processed sequentially on the GPU.

9.10. storage

The following options are related to darktable's library database and XMP sidecar files .

database

check for database maintenance

Indicates when darktable should check for database fragmentation and perform maintenance. Options are "never", "on startup", "on close" and "on both". Each of these is also available with an additional "(don't ask)" option to perform the checks automatically without prompting (default "on close").

database fragmentation ratio threshold

Fragmentation ratio (in per cent) above which database maintenance should be performed (subject to the selection made in the option above) (default 25).

create database snapshot

Specifies how often darktable should create database snapshots. Options are "never", "once a month", "once a week", "once a day" and "on close" (default "once a week")

how many snapshots to keep

Number of snapshots to keep after creating a new snapshot, not counting database backups taken when moving between darktable versions. Enter "-1" to store an unlimited number of snapshots. (default 10)

xmp

write sidecar file for each image

XMP files provide a redundant method of saving the changes you have made to an image, independently of darktable's database. XMP files can later be re-imported into a different database, preserving your changes to the image. It's strongly recommended that you leave this option activated so that you don't lose data if your database becomes corrupted. 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 some output files on export. This option allows binary XMP tags to be compressed in order to save space. Available options are “never”, “always”, and “only large entries” (default).

look for updated xmp files on startup

Scan all XMP files on startup and check if any have been updated in the meantime by some other software. If updated XMP files are found, a menu is opened for the user to choose which of the XMP files to reload (replacing darktable’s database entries by the XMP file contents) and which of the XMP to overwrite from darktable’s database. Activating this option also causes darktable to check for text sidecar files that have been added after import time – see preferences > lighttable > overlay txt sidecar over zoomed images (default off).

9.11. miscellaneous

interface

sort built-in presets first

Choose how the presets menu is sorted. If this option is enabled, built-in presets are shown first. If the option is disabled, user presets are shown first (default on).

mouse wheel scrolls modules side panel by default

When enabled, the mouse wheel scrolls side panels by default and Ctrl+Alt+wheel scrolls data entry fields. When disabled, this behavior is reversed (default off).

always show panels' scrollbars

Defines whether the panel scrollbars should be always visible or activated only depending on the panel content (default on). (needs a restart)

method to use for getting the display profile

This option allows the user to force darktable to use a specific method to obtain the current display profile for color management . In the default setting “all”, darktable will choose to 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 two methods give different results. You can run the darktable-cmstest binary to examine your color management subsystem.

tags

omit hierarchy in simple tag lists

When exporting images any hierarchical tags are also added as a simple list of non-hierarchical tags to make them visible to some other programs. When this option is checked darktable will only include the last part of the hierarchy and ignore the rest. So `foo|bar|baz` will only add `baz`.

disable the entry completion

The entry completion is useful for those who enter tags using the keyboard only. For others the entry completion can be embarrassing (default off). (needs a restart)

keyboard shortcuts with multiple instances

It is possible to create multiple instances of many processing modules. In this scenario it is not always obvious which instance should be controlled by keyboard shortcut operations. The following options control rules that are applied (in order) to decide which module instance keyboard shortcuts should be applied to.

These rules are also used when clicking and dragging on the histogram to change exposure.

prefer expanded instances

If instances of the module are expanded, ignore collapsed instances (default off).

prefer enabled instances

After applying the above rule, if remaining instances of the module are active, ignore inactive instances (default off).

prefer unmasked instances

After applying the above rules, if remaining instances of the module are unmasked, ignore masked instances (default off).

selection order

After applying the above rules, apply the shortcut to the first or last instance remaining (default “last instance”).

other

do not show april 1st game
Don’t show a game when opening darktable on April 1st (default on).

password storage backend to use
The backend to use for password storage. Options: “auto” (default), “none”, “libsecret”, “kwallet”.

executable for playing audio files
Define 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”).

9.12. shortcuts

Much of the functionality of darktable can be accessed via shortcuts using the keyboard or keyboard/mouse combinations. These shortcuts are user-configurable via the *shortcuts* tab.

Many important shortcut actions are provided with default key combinations, but most must be manually configured by the user. Any key may be used for a keyboard shortcut, and may be combined with the Shift, Control or Alt modifier keys (or any combination thereof).

When you open the *shortcuts* tab you are initially presented with a hierarchical list of all actions that can be applied with a keyboard shortcut. At the top of this hierarchy is a short list of key *categories* which are defined below.

add or amend a shortcut

In order to add or amend a shortcut, first navigate to the action you want to change and double-click on it. You will be prompted to press the new key combination to be mapped to the selected action.

If a conflict is found you will be given the option to retain the existing shortcut or replace it. Depending on context, it is possible to use the same keyboard shortcut for multiple actions. For example the same key combination may be used for one action in the lighttable view and another in the darkroom view.

remove a shortcut

To remove a keyboard shortcut, single-click on the action you wish to remove it from and press the Backspace key.

search for a shortcut action

A search field is shown at the bottom of the *shortcuts* tab. Enter the text you wish to search for and press Enter or click the *search* button. Press Enter or *search* multiple times to cycle through all matching shortcut actions.

view currently assigned shortcuts

Press the H key in any darktable view to show a list of all shortcuts that are assigned for the current view.

import, export, reset

You can import your shortcut mappings from or export them to a file.

Press the *default* button to reset all shortcuts to their default state. *Take care when using this option as it is not possible to restore back to a previous state unless you have first exported existing shortcuts to a file or taken a backup of your configuration directory.*

shortcut categories

Keyboard shortcuts are categorized within a hierarchical list so that they can easily be found. The following sections summarize these categories and list some common options.

global

Shortcut actions in this section are applicable to all darktable views.

If you have created any user-defined styles, these will be available as actions within a “styles” sub-section.

views

A single section is provided for each darktable view. Shortcut actions are only applicable to the selected view.

processing modules

Shortcut actions for the processing modules in the darktable view. A section is provided for each processing module.

In addition, a separate “blending” section allows you to control the masking and blending options for the last module that you interacted with.

common shortcuts

Every processing module provides the following shortcut actions by default

enable module
Enable or disable the module, regardless of whether it is currently visible

show module
Expand or collapse the module. If the module is not currently displayed on the screen, darktable will switch to an appropriate module group (if applicable) before displaying it

focus module
Cause the module to receive or lose focus.

reset module parameters
Reset the module to its default state

show preset menu
Show the presets menu for the module

presets
An expandable category which lists all currently-defined presets for the module as possible actions. It will not be shown if there are no presets.

For comboboxes and sliders, some standard shortcut actions are provided, as described in the following sections.

In addition, other module-specific controls will be provided with their own shortcut actions.

sliders

All sliders in processing modules can be adjusted via keyboard shortcuts, regardless of whether the module is currently shown or enabled. The following shortcut actions are provided as standard for each slider:

increase/decrease
Separate shortcuts which allow you to increase or decrease the slider’s value by a single step.

dynamic
A single shortcut that can be used in combination with the mouse scroll wheel to increase and decrease slider values.
edit
A shortcut to bring up the slider's edit dialog within which you may key a value directly or modify the slider with the mouse.
reset
Reset the slider to its default value.

In addition, you can modify the precision of the increase/decrease operations with a keyboard shortcut (*shortcuts > views > darkroom > change keyboard shortcut slider precision*), choosing between fine, normal and coarse. See module controls for more details.

When performing increase/decrease and dynamic operations on sliders a toast message will appear at the top of the image to indicate the current value of the slider.

comboboxes

As with sliders, all comboboxes in processing modules can be adjusted via keyboard shortcuts. The following shortcut actions are provided as standard for each combobox:

next/previous
Separate shortcuts which allow you to change to the next or previous entry in the combobox
dynamic
A single shortcut that can be used in combination with the mouse scroll wheel to change to the next/previous entry in the combobox

If the end of a combobox list is reached, these shortcuts will cycle back to the beginning of the list. Similarly, if the beginning of the list is reached the shortcuts will cycle to the end.

multiple module instances

It is possible to create multiple instances of many processing modules. In this scenario it is not always obvious which instance should be controlled by keyboard shortcut operations.

See preferences > miscellaneous for some additional settings that allow you to control how keyboard shortcuts are handled when multiple instances of a processing module are present.

utility modules

Shortcut actions for the utility modules . These are modules that are not used for image processing and may appear on any panel. Some utility modules can be used in multiple views.

As with processing modules, some shortcut actions are provided by default for each module:

show module
Expand or collapse the module.
reset module parameters
Reset the module to its default state.
show preset menu
Show the presets menu for the module.

Some of the above actions may not be available for all utility modules.

9.13. presets

This menu provides an overview of the presets that are defined for darktable's modules, and allows you to modify some of their properties.

Pre-defined presets (those that are included by default within darktable) are shown with a lock symbol. Their properties cannot be changed.

User-defined presets can be imported from saved `.dtpreset` files using the "import" button at the bottom of the screen. You can export *all* user-defined presets to a single directory using the "export" button.

Delete a preset by selecting it and pressing the Delete key.

Edit a preset's properties by selecting it and pressing Enter or double-clicking. This will open a dialog that allows the preset to be edited, saved to an external `.dtpreset` file or deleted.

See the creating and editing presets section for more information about the other properties that can be edited in this dialog.

9.14. lua options

lua scripts installer don't show again
Check this box to hide the lua scripts installer in the lighttable if no lua scripts are installed.

10. Scripting with Lua

darktable comes with a versatile scripting interface language for functionality enhancement.

10.1. overview

Lua scripts can be used to define actions for darktable to perform when an event is triggered. One example might be calling an external application during file export in order to apply additional processing steps outside of darktable.

Lua 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 5.3. Describing the principles and syntax of Lua is beyond the scope of this usermanual. For a detailed introduction see the Lua reference manual .

The following sections will provide you with a brief introduction to how Lua scripts can be used within darktable.

10.2. basic principles: luarc files

At startup, darktable will automatically run the Lua scripts `$DARKTABLE/share/darktable/luarc` and `$HOME/.config/darktable/luarc` (where `$DARKTABLE` represents the darktable installation directory and `$HOME` represents your home directory).

This is the only time darktable will run Lua scripts by itself. These scripts can register callbacks to perform actions on various darktable events. This callback mechanism is the primary method of triggering lua actions.

10.3. a simple lua example

Let's start with a simple example which will print some code on the console. Create a file called `luarc` in darktable's configuration directory (usually `$HOME/.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 darktable-specific in the script. We simply use Lua's 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 darktable's Lua API reference manual.

```
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 by running darktable with "`darktable -d lua`" on the command line. This is the recommended way to do traces in a darktable lua script.

10.4. 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 `require 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 the `types_dt_lua_image_t` section of the API manual). 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. Here, we are interested in the “red” label. 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
```

This code should be quite simple to understand at this point, but it contains a few interesting aspects of 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’s 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 it should not be used for anything else.

10.5. 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 prints its parameters")
```

Now start darktable, go to “preferences > shortcuts > lua > A shortcut that prints its parameters”, assign a shortcut and try it. You should see a nice message printed on the screen.

Let’s look at the code in detail. We first define a function which takes two strings as input parameters. The first one is the type of event that is triggered (“shortcut”) and the second one is the name of the shortcut (“A shortcut that prints its parameters”). The function itself calls `darktable.print`, which 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 preferences window.

Let’s try a shortcut that is a little more interactive. This one will look at the images the user is currently interested in (selected or moused-over) and 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 that gives the star rating of any image (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 makes it easy to follow darktable's UI logic in lua.

If you select an image and press your shortcut a couple of times, it will work correctly at first but when you have reached five stars, darktable will start showing the following error on the console:

```
<![CDATA[
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 to the console.

10.6. exporting images with lua

So far we have learned to use lua to adapt darktable to our particular workflow. Let's look now at how to use lua to easily export images to an online service. If you are able to upload an image to a service via the command line then you can use lua to integrate this into darktable's user interface.

In this next example we will use lua to export via scp. A new storage type will appear in darktable's UI which will export images to a remote target via the copy mechanism in ssh.

```
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 not darktable.control.execute("scp "..filename.." "..
      darktable.preferences.read("scp_export",
         "export_path","string")) then
      darktable.print_error("scp failed for "..tostring(image))
    end
end)
```

`darktable.preferences.register` will add a new preference to darktable's preferences 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 the API manual relating to `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 type. The first argument is a name for the storage type, 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. We will detail how to do that in the next section
- We do not 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. `scp` can't be provided a password easily, so we will leave it that way
- There is no message displayed once the example is done, only the progress bar 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.

10.7. building user interface elements

Our previous example was a bit limited. In particular the use of a preference for the export path wasn't very nice. We can do better than that by adding elements to the user interface in the export dialog.

UI elements are created via the `darktable_new_widget` function. This function takes a type of widget as a parameter and returns a new object corresponding to that widget. You can then set various fields in that widget to set its parameters. You will then use that object as a parameter to various functions that will add it to the darktable UI. The following simple example adds a lib in the lighttable view with a simple label

```
local my_label = darktable.new_widget("label")
my_label.label = "Hello, world !"

dt.register_lib("test","test",false,{
    [dt.gui.views.lighttable] = {"DT_UI_CONTAINER_PANEL_LEFT_CENTER",20},
    },my_label)
```

There is a nice syntactic trick to make UI elements code easier to read and write. You can call these objects as functions with a table of key values as an argument. This allows the following example to work. It creates a container widget with two sub-widgets: a label and a text entry field.

```
   local my_widget = darktable.new_widget("box"){
      orientation = "horizontal",
      darktable.new_widget("label"){ label = "here => " },
      darktable.new_widget("entry"){ tooltip = "please enter text here" }
   }
```

Now that we know that, let's improve our script a bit.

```
darktable = require "darktable"

local scp_path = darktable.new_widget("entry"){
  tooltip ="Complete path to copy to. Can include user and hostname",
  text = "",
  reset_callback = function(self) self.text = "" end
}

darktable.register_storage("scp_export","Export via scp",
  function( storage, image, format, filename,
     number, total, high_quality, extra_data)
    if not darktable.control.execute(scp "..filename.." "..
      scp_path.text
    ) then
      darktable.print_error("scp failed for "..tostring(image))
    end
    end,
    nil, --finalize
    nil, --supported
    nil, --initialize
    darktable.new_widget("box") {
    orientation ="horizontal",
    darktable.new_widget("label"){label = "target SCP PATH "},
    scp_path,
})
```

10.8. sharing scripts

So far, all of our lua code has been 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 this 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/lua/ where CONFIGDIR
is your darktable configuration directory
* add the following line in the file $CONFIGDIR/luarc
  require "scp-storage"

USAGE
* select "Export via SCP" in the storage selection menu
* set the target directory
* export your images

LICENSE
GPLv2

]]
darktable = require "darktable"
```

```
darktable.configuration.check_version(...,{2,0,0})

local scp_path = darktable.new_widget("entry"){
  tooltip ="Complete path to copy to. Can include user and hostname",
  text = "",
  reset_callback = function(self) self.text = "" end
}

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.." "..
      scp_path.text
    ) then
      darktable.print_error("scp failed for "..tostring(image))
    end
    end,
    nil, --finalize
    nil, --supported
    nil, --initialize
    darktable.new_widget("box") {
    orientation ="horizontal",
    darktable.new_widget("label"){label = "target SCP PATH "},
    scp_path,
})
```

darktable will look for scripts (following the normal lua rules) in the standard directories plus `$CONFIGDIR/lua/*.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` so as not to 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 .

10.9. 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.

10.10. using darktable from a lua script

Warning: This feature is very experimental. It is known that several elements don't yet work in library mode. 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 relating 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 useful here if you don't want to work on your personal library.

10.11. lua API

darktable's Lua API is documented in its own manual with a detailed description of all data structures and functions. You can download the API manual from here .

11. Guides & Tutorials

11.1. developing monochrome images

Photography has a long history of producing monochrome images, and many still enjoy this aspect of photography. While there are some specialised/modified cameras with a truly monochrome sensor, most still use a regular camera to capture a color image, and turn it into a monochrome image during post-processing.

There are two main approaches this conversion:

- A *physical approach*, where we attempt to simulate how a silver-based photographic film emulsion would react to the light captured at the scene.
- A *perceptual approach*, where we develop a color image and reduce the color saturation in a perceptual color space such as *CIE Lab*.

These approaches and other monochrome-related features in darktable are discussed in the following sections.

importing and flagging images as monochrome

When importing an image, there are a number of properties that can be used to indicate that the image requires a monochrome treatment:

- If the image was taken using an achromatic camera, the image will be automatically flagged as monochrome.
- When you capture a scene from which you would like to produce a monochrome image, it can be helpful to put your camera into a “black & white” creative mode. This allows you to visualise what the scene would look like in monochrome through your camera’s liveview screen or electronic viewfinder. The camera will still capture the full color data in the raw file, but the embedded jpeg preview image will be monochrome. When you import such an image, darktable can automatically flag the image as monochrome based on the preview image.

  Checking whether the preview is monochrome slows down the import process, so this is disabled by default. You can enable this check in preferences > processing > detect monochrome previews
- When processing a raw file, one of the first steps is to *demosaic* the image. If you set the *demosaicing method* to “passthrough (monochrome)”, this discards color information during the demosaicing process, and darktable will flag the image as monochrome.

  Note: You should only use this for images taken on a camera where the Bayer filter has been removed.
- After you have imported the image, you can manually flag an image as monochrome (or not) using the *metadata* tab on the lighttable’s selected images module,

If any of the above methods result in an image being flagged as monochrome, darktable modules can use this information to present the user with some monochrome-specific module controls and/or apply special processing to the image.

The `darktable|mode|monochrome` tag will be automatically applied to any images flagged as monochrome, and if you have enabled any permanent overlay information on your lighttable thumbnails, such images will be marked with a visual indicator `B&W` next to the file type information. By automatically applying this tag and visual indicator, darktable makes it easy to set up filters to single out any images for monochrome treatment, and to see at a glance which images in the current collection bear the *monochrome* tag.

monochrome conversion

physical approach

This approach tends to work with linear scene-referred data from the sensor, and attempts to mimic the response of a photographic film with a silver emulsion. It consists of three steps:

1. Map the color channels from the sensor into a single monochrome channel. Different types of monochrome photographic film have different levels of sensitivity to various wavelengths of light, and this can be simulated by giving the three color channels different weightings when combining them together into a single monochrome channel. The *color calibration* module allows the three channels to be mixed into a gray channel by varying amounts, and it includes a number of presets that are designed to emulate the characteristics of some well-known types of photographic film.

2. Apply a luminosity saturation curve. As a piece of photographic film is exposed to more intense light, its response will fall off as the silver emulsion becomes saturated. This saturation curve is simulated by the *filmic rgb* module.

3. Developing a monochrome film in the darkroom traditionally involves “dodging and burning” to control the level of exposure across different parts of the image. This can be emulated in darktable by using either the *exposure* module with manually created masks , or by using the *tone equalizer* module, which will automatically generate a mask for you using a guided filter.

perceptual approach

The other option for producing a monochrome image is to reduce the color saturation in the image, which can be done in a linear colorspace, or in a color space oriented towards modelling human perception.

- The *color balance* module operates in linear RGB, and allows you to reduce the color saturation in the image using either the input or output color saturation slider – which you choose depends on whether you want to make any other adjustments to either the color or monochrome image in the color balance module. The color balance module tends to give a predictable and perceptually uniform result.

- The *monochrome* module works in Lab color space, and it allows the user to graphically define a weighted combination of colors to determine the density of the blacks in the monochrome image. The interface can be somewhat sensitive to the settings, with small changes producing large effects, and you may experience problems with the global contrast and/or black pixel artifacts.

- Other modules such as *color zones* can also be used to remove color saturation from the image, but these don’t offer any real advantage over the simplicity of the *color balance* module’s saturation sliders.

11.2. other resources

The official places to obtain information concerning darktable are the following:

- darktable.org
- GitHub wiki

The following articles provide some useful background information about darktable's scene-referred workflow:

- darktable 3.0 for dummies in 3 modules
- darktable 3.0 for dummies – hardcore edition
- darktable 3.0: RGB or Lab? Which modules? Help!
- darktable's filmic FAQ
- Introducing color calibration module

The following resources can provide a deeper understanding of the functionality of some specific processing modules:

- Image processing and the pixel pipeline in darktable 3.0
- Filmic RGB v3: remapping any dynamic range in darktable 3.0
- Filmic RGB v4: highlights reconstruction in darktable 3.2
- Dodging and burning with tone equalizer in darktable 3.0
- Noise reduction (profiled) with non-local means in darktable 3.0
- Noise reduction (profiled) with wavelets in darktable 3.0
- Contrast equalizer modules in darktable

Some other YouTube channels with recent content on using darktable are:

- Bruce Williams Photography
- Rico Richardson
- Boris Hajdukovic

For help and support, you can also ask questions on the main discussion forums at discuss.pixls.us .

12. Special Topics

12.1. color management

12.1.1. overview

darktable employs a fully color managed workflow:

- Input color specifications are taken from embedded or user-supplied ICC profiles or – in the case of raw files – from a library of camera-specific color matrices.
- darktable automatically reads the display profile of your monitor (if properly configured) for accurate on-screen color rendition. Multi-screen setups are fully supported as long as a system service like colord is in place and properly set up to inform darktable of the correct monitor profile.
- Output files can be encoded in one of darktable's built-in profiles, including sRGB and Adobe RGB, or into a color space supplied by the user as an ICC profile.

12.1.2. display profile

For darktable to faithfully render colors on screen it needs to find the correct display profile for your monitor. In general this requires your monitor to be properly calibrated and profiled, and it needs the profile to be correctly installed on your system. darktable queries your X display server's xatom as well as the system service colord (if available) for the right profile. If required you can enforce a specific method in preferences > miscellaneous .

To investigate your display profile configuration you can run the darktable-cmstest binary (Linux only) which prints out useful information (e.g. profile name per-monitor) and tells you if the system is correctly configured.

In rare cases you may need to manually select the display profile. This is possible from the soft proof and gamut check option dialogs in the darkroom view and the display profile dialog in the lighttable view.

Bear in mind that high-tier consumer-grade screens do not need a user-made display profile unless you need to perform soft-proofing with professional expectations, since they are properly calibrated to sRGB in the factory.

A poorly made display profile will be more harmful than sticking to the default sRGB profile, since the default might be slightly inaccurate but will at least be reliable. Advanced and professional users are advised to proceed with the production of custom display profiles only if they have the training to assess the quality of the resulting profile and understanding of the profiling options.

12.1.3. rendering method

darktable can render colors either with its internal algorithms or by using the external library LittleCMS2. darktable's internal method is, by an order of magnitude, faster than the external one. The external option gives you a choice of the rendering intent and might offer a slightly higher accuracy in some cases.

You can change the default method in preferences > processing > always use LittleCMS 2 to apply output color profile

---

Note: If the given ICC is LUT-based or contains both, a LUT and a matrix, darktable will use LittleCMS2 to render the colors regardless of the configuration parameter's value.

12.1.4. rendering intent

If rendering with LittleCMS2 is activated (see rendering method) you can define how out-of-gamut colors are handled when converting between color spaces. A selection box in the export module, the *output color profile* module, and soft proof gives you a choice of the following rendering intents:

perceptual
Suited to photographs as it maintains the relative position of colors. This is usually the best choice.

relative colorimetric
Out-of-gamut colors are converted to colors having the same lightness, but different saturation. Other colors remain unmodified.

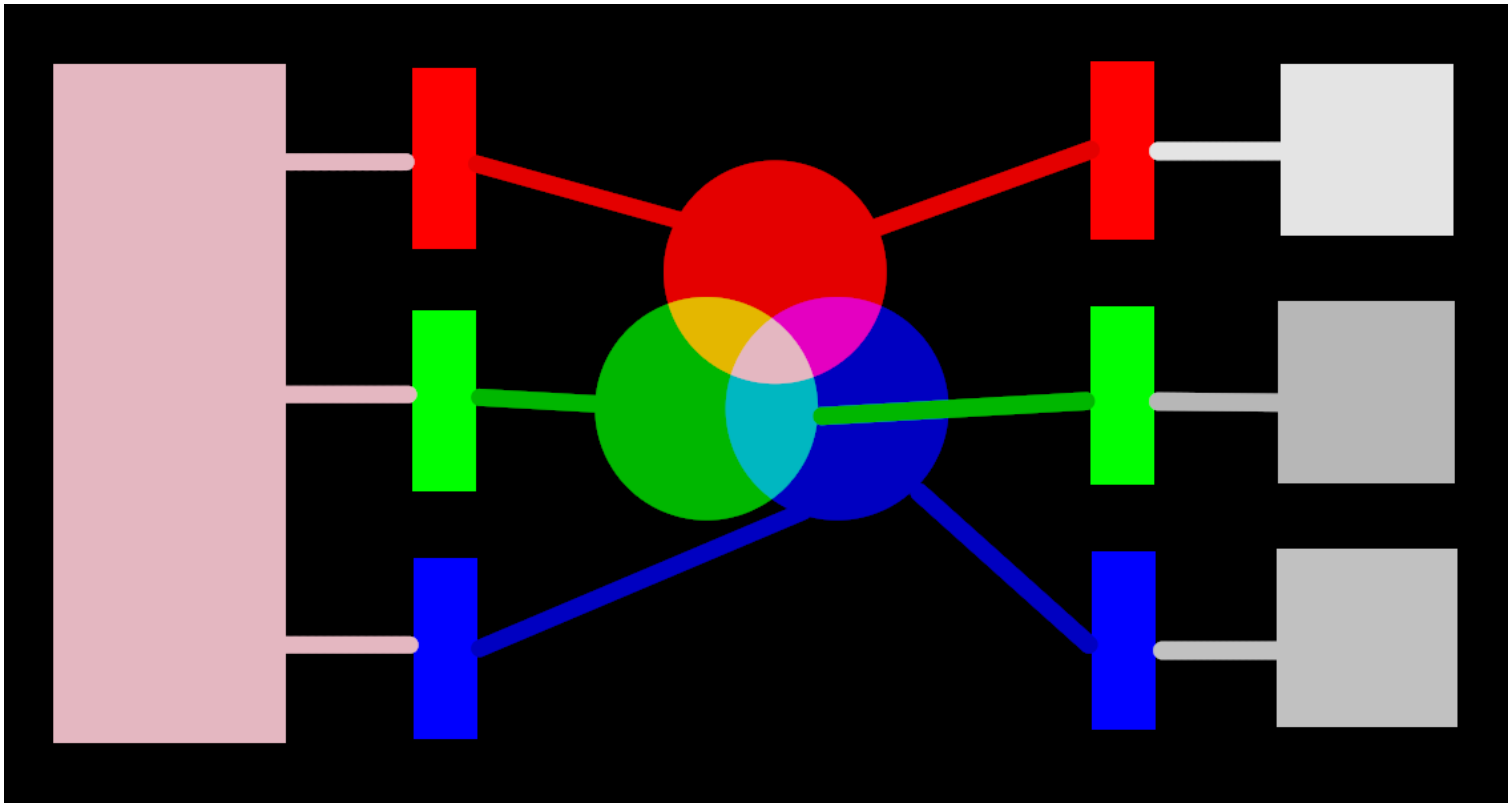
saturation
Saturation is retained but lightness is slightly changed.

absolute colorimetric
Keep the white point.

12.1.5. darktable's color spaces

darktable’s input images are either RGB files (like JPEGs or TIFFs) or camera RAWs. Both store visual information as a combination of primary colors (such as red, green and blue) which together describe a light emission to be later recreated by a display.

The following image illustrates this concept.



On the left-hand-side of the image is a colored light that we need to represent digitally. Using three ideal color filters, we can decompose this light into three colored primary lights at different intensities. In order to recreate the original colored light from our ideal decomposition, as illustrated in the center of the image, we simply need to recombine those 3 primary lights by addition.

It should be possible to reproduce this original light by taking a combination of white lights at the correct intensities and projecting that light through appropriately colored filters. This experiment can be performed at home using gels and dimmable white bulbs. This is roughly what old color CRT displays did and how video projectors still work.

In photography, the initial decomposition step is performed by the color filter array that sits on top of your camera’s sensor. This decomposition is not ideal, so it is not possible to recreate the original emission directly by a simple addition. Instead we need to provide some intermediate scaling to adjust the three intensities.

On screens, the LED bulbs are dimmed proportionally to each intensity, and the emissions of the three lights physically added to reconstruct the original emission. Digital images store the intensities of these primary lights as a set of 3 numbers for each pixel, depicted on the right of the above image as shades of gray.

While a set of display intensities can be easily combined to recreate an original light on a screen (for example, if we created a synthetic image in-computer) the set of captured intensities from a sensor needs some scaling in order for the on-screen light addition to reasonably reproduce the original light emission. This means that every set of intensities, expressed as an RGB set, must be linked to a set of filters (or primary LED colors) that define a *color space* – any RGB set only makes sense with reference to a color space.

Not only do we need to temper the captured intensities to make them summable again, but if we are to recompose the original light on a display that does not have the same colored filters or primaries as the space in which our RGB set belongs, these intensities need to be rescaled to take into account the differing filters on the display. The mechanism for this scaling is described in *color profiles*, usually stored within .icc files.

Note: Color is not a physical property of light – it exists only in the human brain, as a product of the decomposition of a light emission by the cone cells in the retina, again very similar in principle to the above filtering example. An “RGB” value should be understood as “light emissions encoded on 3 channels connected to 3 primaries”, but the primaries themselves may look different from what humans would call “red”, “green” or “blue”.

To summarize, the filters described here are overlapping band-pass filters. Since they overlap, summing them back together would not preserve the energy of the original spectrum so, long story short, we need to dial them down with regard to the retina cone response

Most of darktable’s actual image processing takes place in a large RGB “working profile” space, with some (mostly older) modules internally working in the CIELab 1976 color space (often just called “Lab”). The final output of the image processing pipeline is once again in an RGB space shaped for either the monitor display or the output file.

This process implies that the pixelpipe has two fixed 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 position in the pixelpipe which tells you which color space the module lives in:

- up to *demosaic* : The raw image information does not yet constitute an “image” but merely “data” about the light captured by the camera. Each pixel carries a single intensity for one single primary color, and camera primaries are very different from primaries used in models of human vision. Bear in mind that some of the modules in this part of the pipe can also act on non-raw input images in RGB format with full information on all three color channels.
- between *demosaic* and *input color profile* : Image is in RGB format within the color space of the specific camera or input file.
- between *input color profile* and *output color profile* : Image is in the space defined by the selected working profile (linear Rec2020 RGB by default). As darktable processes images in 4x32-bit floating point buffers, we can handle large working color spaces without risking banding or tonal breaks.
- after *output color profile* : Image is in RGB format as defined by the selected display or output ICC profile.

12.1.6. unbounded colors

Screens and most image file formats can only encode RGB intensities confined within a certain range. For example, images encoded on 8 bits can only contain values from 0 to 255, images on 10 bits from 0 to 1023, and so on… Graphic standards postulate that the maximum of that range, no matter its actual value, will always represent the maximum brightness that the display medium is able to render, usually between 100 and 160 Cd/m² (or nits) depending on the actual standard. We generally call this maximum “100 % display-referred”. The minimum of the range, encoded 0 no matter the bit-depth used, becomes then “0 % display-referred”. 100 % encodes pure white, 0 % encodes pure black.

This is a limitation for image processing applications, because it means that any pixel lying outside of this range will be clipped to the nearest bound, resulting in non-recoverable loss of data (colors and/or textures).

For the longest time, image processing software too was bounded to this limitation for technical reasons, and some still is, but now by design choice. As a result, they would clip RGB intensities at 100 % display-referred between image operations.

darktable uses floating-point arithmetic inside its color pipeline, which means it can handle any RGB value internally, even those outside the display-referred range, as long as it is positive. Only at the very end of the pipeline, before the image is saved to a file or sent to display, are the RGB values clipped if needed.

Pixels that can take values outside of the display range are said to have “unbounded colors”. One could choose to clamp (i.e. confine) those values to the allowed range at every processing step or choose to carry on with them, and clamp them only at the last step in the pipeline. However, it has been found that processing is less prone to artifacts if the unbounded colors are not clamped but treated just like any other color data.

At the end of the pipeline, modules like *filmic* can help you to remap RGB values to the display-referred range while maximizing the data preservation and avoiding hard clipping, which is usually not visually pleasing.

However, at all times in the pipeline, you must ensure that you do not create negative RGB values. RGB intensities encode light emissions and negative light does not exist. Those modules which rely on a physical understanding of light to process pixels will fail if they encounter a non-physical light emission. For safety, negative RGB values are still clipped whenever they might make the algorithms fail, but the visual result might look degraded. Negative values can be produced when abusing the *black level* in *exposure* or the *offset* in *color balance* and care should be taken when using these modules.

12.1.7. possible color artifacts

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, with the a and b channels purely representing 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 likely candidates for pixels with negative L values. If you are engaged in stage photography you should pay close attention to such lights appearing in images.

In order to mitigate this issue the *input color profile* module has a gamut clipping option. This option is switched off by default but can be activated if artifacts are observed. Depending on the settings, colors will be confined to one of the available RGB gamuts. In effect black pixel artifacts are prevented at the costs of losing some color dynamics.

12.1.8. darktable's color dimensions

This section defines the perceptual properties of color, both conceptually and quantitatively, in order to characterize and quantify the creative and corrective adjustments made on color in the software.

Definitions

Color properties like “saturation”, “brightness” or “lightness” have passed to common usage but are largely misused and often used to mean different things. In color science, each of these terms has a precise meaning.

There are two frameworks within which color properties may be analyzed and described:

- A scene-linear, physiological, framework, that mostly focuses on the response of the retina cone cells, using color spaces such as CIE XYZ 1931 or CIE LMS 2006,
- A perceptual, psychological, framework that stacks the brain’s corrections on top of the retina signal, using color spaces such as CIE Lab 1976, CIE Luv 1976, CIE CAM 2016 and JzAzBz (2017).

These two frameworks provide us with metrics and dimensions to analyze color and allow us to change some of its properties while preserving others unchanged.

The following 7 dimensions of color are used by darktable:

hue
An attribute of visual perception in which an area appears to be similar to one of the colors red, yellow, green, or blue, or to a combination of adjacent pairs of these colors considered in a closed ring. [1] Hue is a shared property between the perceptual and scene-linear frameworks.

luminance
The density of luminous intensity with respect to a projected area in a specified direction at a specified point on a real or imaginary surface. [2] Luminance is a property of scene-referred frameworks, and is expressed by the Y channel of the CIE XYZ 1931 space.

brightness
An attribute of visual perception according to which an area appears to emit, transmit or reflect more or less light. [3]

lightness
The brightness of an area judged relative to the brightness of a similarly illuminated area that appears to be white or highly transmitting. [4] Lightness is the perceptual, non-linear homologue of luminance (roughly equal to the cubic root of luminance Y). Lightness is expressed by the L channel in CIE Lab and Luv 1976 and the J channel in JzAzBz.

chroma
The colorfulness of an area judged as a proportion of the brightness of a similarly illuminated area that appears gray, white or highly transmitting. [5] *Warning: chroma is not short for chrominance, which is the color part of a video signal (the Cb and Cr channels in YCbCr, for example).*

brilliance
The brightness of an area judged relative to the brightness of its surroundings. [6]

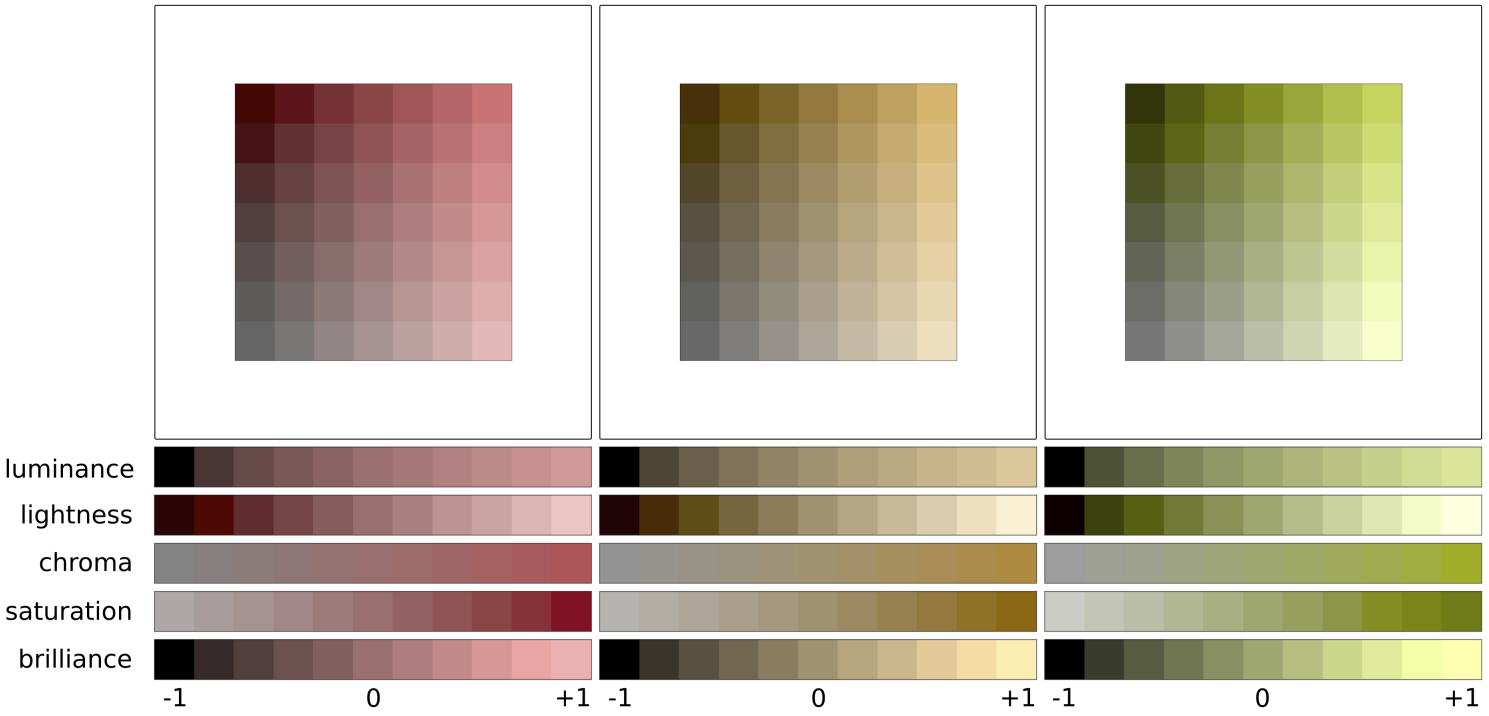
saturation

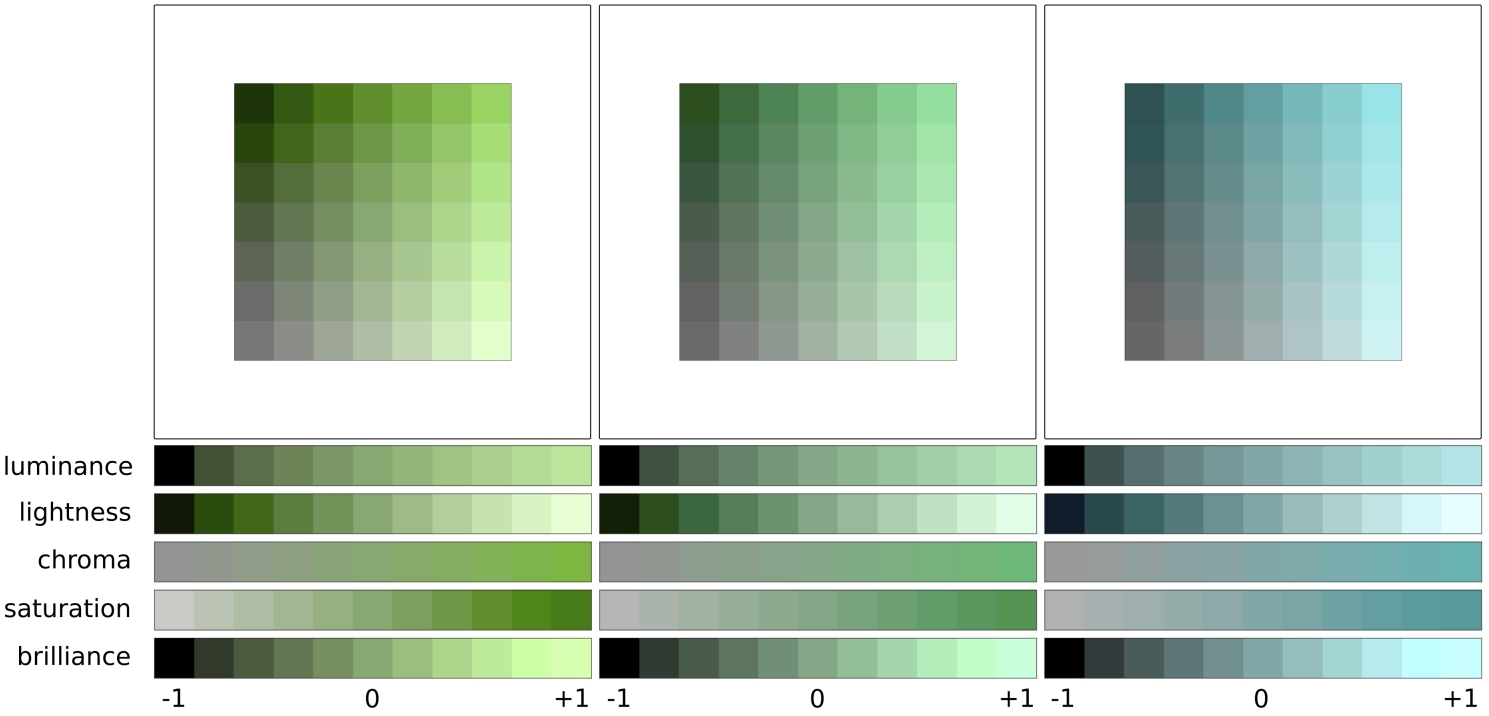
The colorfulness of an area judged in proportion to its brightness. [7]

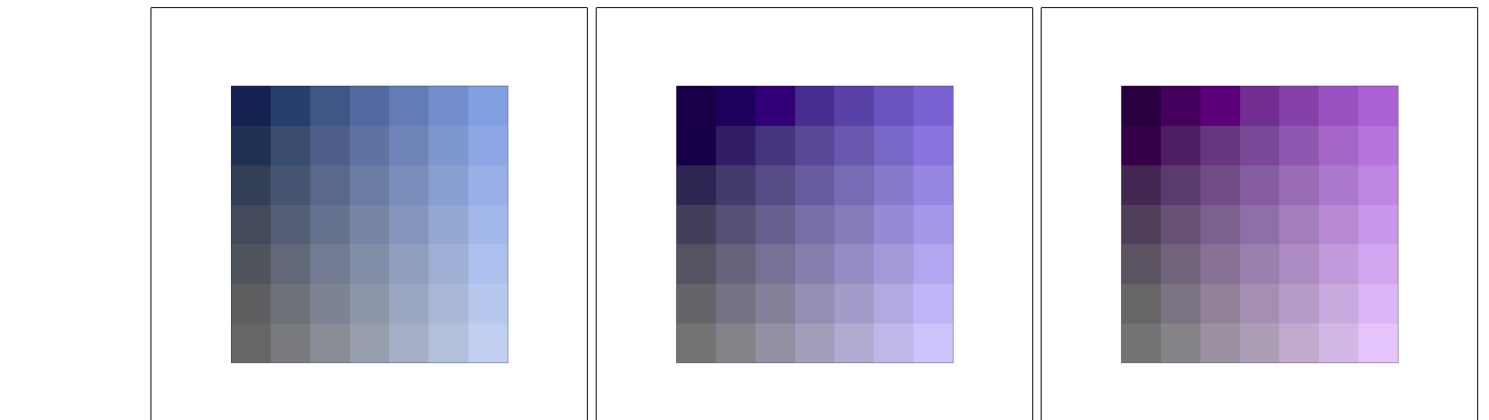
Colors can be described in many different color spaces, but no matter the color space, they need at least 3 components: some metric of luminance or brightness, and 2 metrics of chromaticity (hue and chroma, or opponent color coordinates).

Illustrations

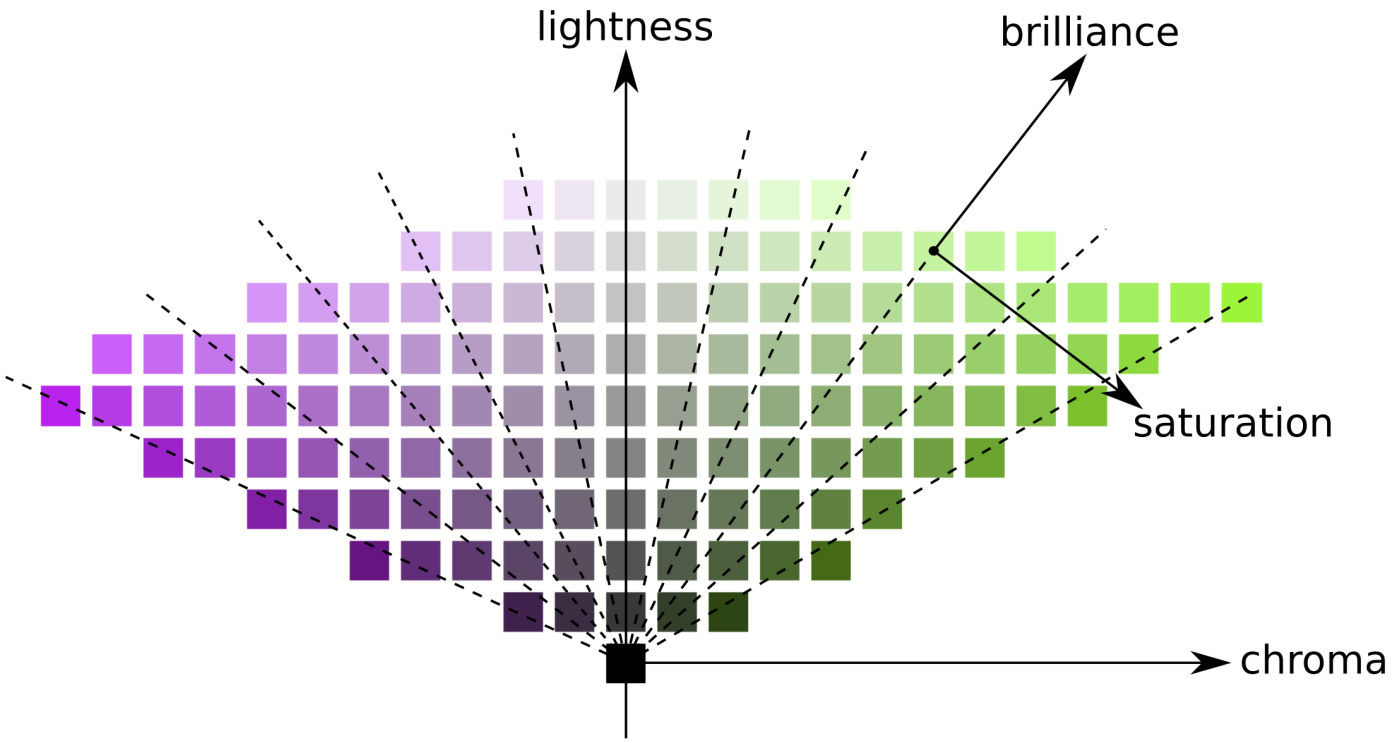
While the previous definitions are useful to give a meaning to the words, they don’t tell us what we should be looking at. The following charts show luminance, lightness, chroma, brilliance/brightness and saturation varying from a “0” base color and how the resulting colors degrade:


luminance
lightness
chroma
saturation
brilliance
-1
0
+1
-1
0
+1
-1
0
+1


luminance
lightness
chroma
saturation
brilliance
-1
0
+1
-1
0
+1
-1
0
+1


luminance

(Lightness + Chroma) or (Brilliance/Brightness + Saturation) are two different ways to encode the same reality. They are both orthogonal spaces that can be converted from one to another by a simple rotation of the base. This means that chroma evolves at constant lightness, saturation evolves at constant brilliance/brightness, and vice versa:



The lines of equal chroma are vertical (following the patches grid), meaning that chroma has the same direction for all colors in the gamut (see below). However, the lines of equal saturation are oblique (drawn dashed on the graph) and all go from black though each color patch, meaning their directions are particular to each color.

Increasing the chroma will therefore move all colors uniformly away from the central gray axis horizontally, while increasing the saturation will close or open the angle of the oblique dashed lines like a flower.

Similarly, increasing the lightness will move all colors uniformly up from the horizontal axis, while increasing brilliance/brightness will move them along the lines of equal saturation.

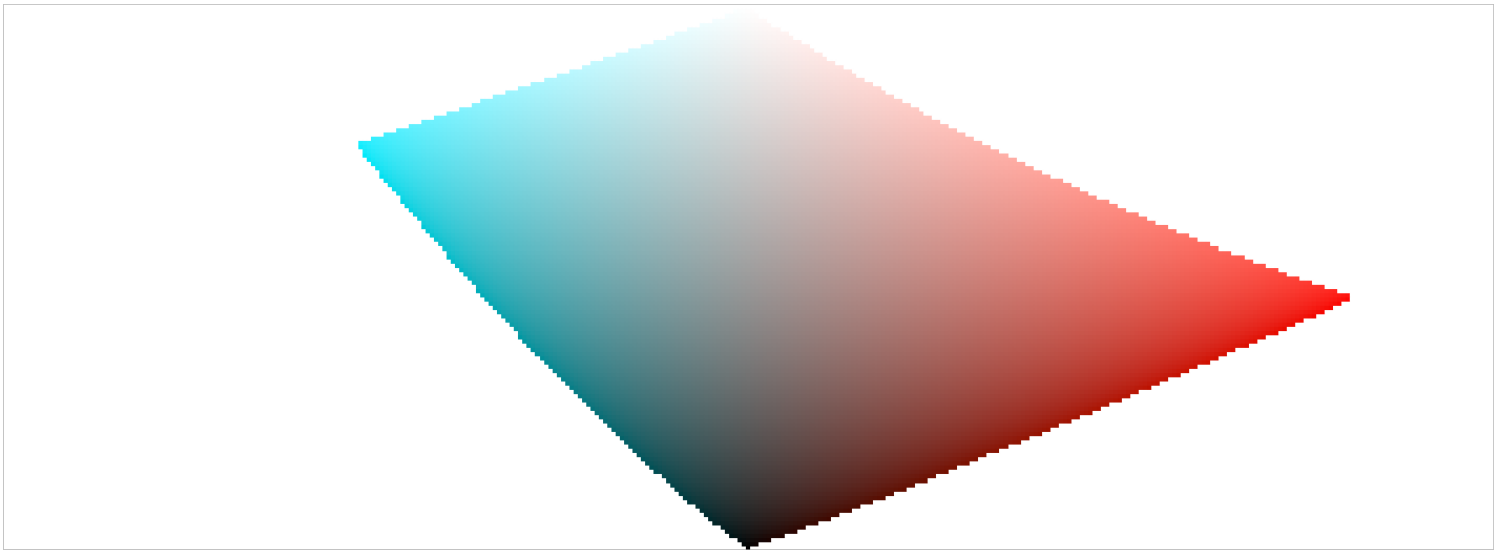
On both charts above, lightness, chroma, saturation and brilliance are drawn in the JzAzBz color space, which is a perceptual color space suited for an HDR signal, and is used in parametric masks and the color balance RGB module. Luminance is drawn in the CIE XYZ 1931 color space, and represents the effect of an exposure compensation. It shows the same behaviour as brilliance, except that the step size is not perceptually scaled.

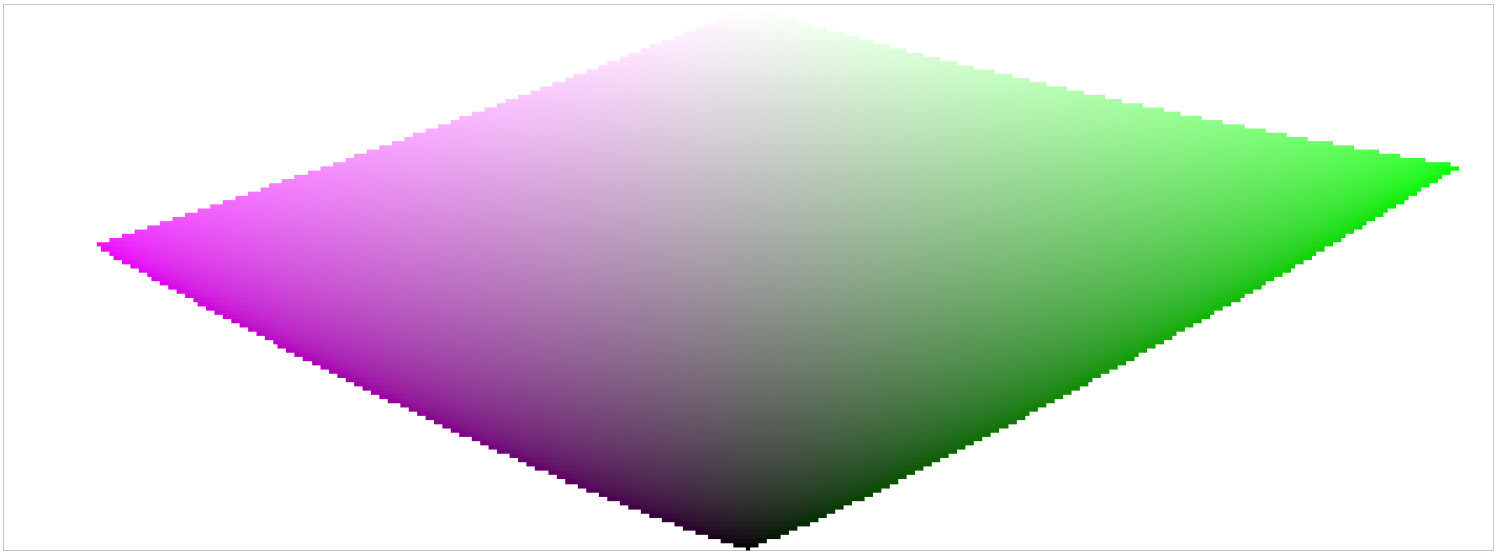
Note: In this section, *brilliance* and *brightness* are used simultaneously to describe the same dimension. In all rigor, brightness is an absolute metric, while brilliance is the brightness of some surface relative to the brightness of its surroundings (that is, how much a surface “pops” out of its surroundings and looks fluorescent). But in image editing, increasing the brightness of some surface will indeed increase its brilliance too, so the term *brilliance* is prefered in darktable’s user interface for clarity and in reference to its visual effect.

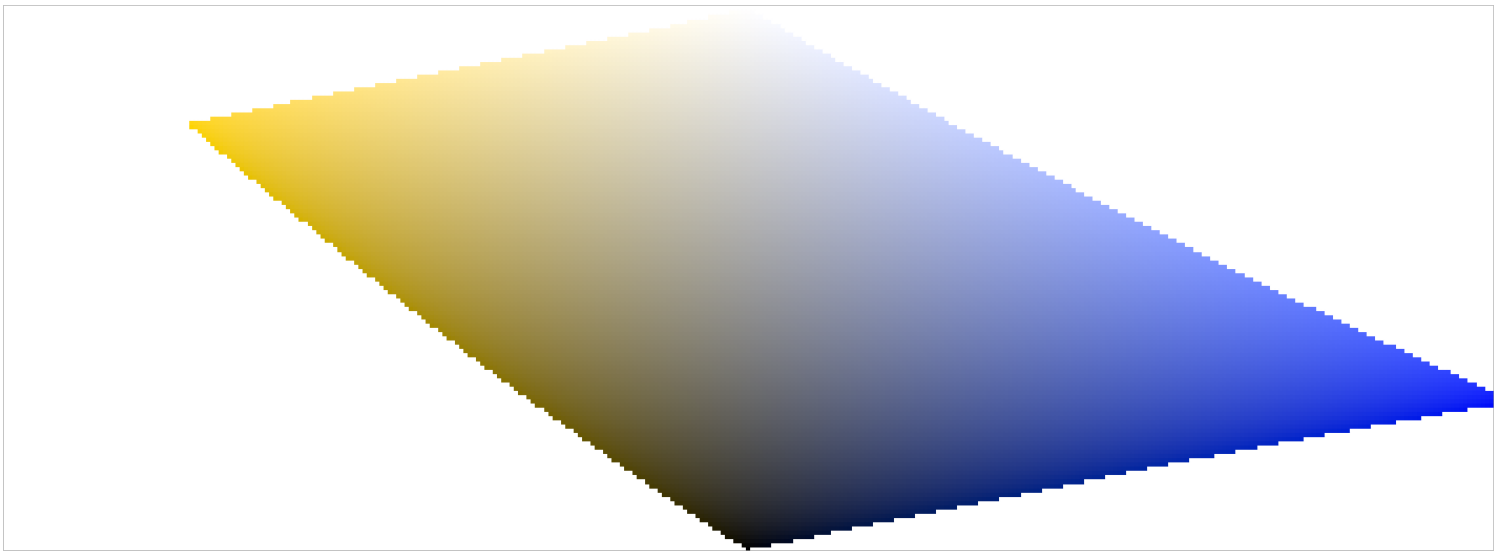
Color dimensions and gamut

The gamut is the volume of colors that a certain color space can encompass and encode. It is important to note that, once converted to perceptual spaces, the gamut of any RGB space is not uniform along hues.

The following examples show the gamut volume of the sRGB space on hue slices containing the primary red, green and blue lights of the sRGB space, over a lightness-chroma plane with a uniform scale:





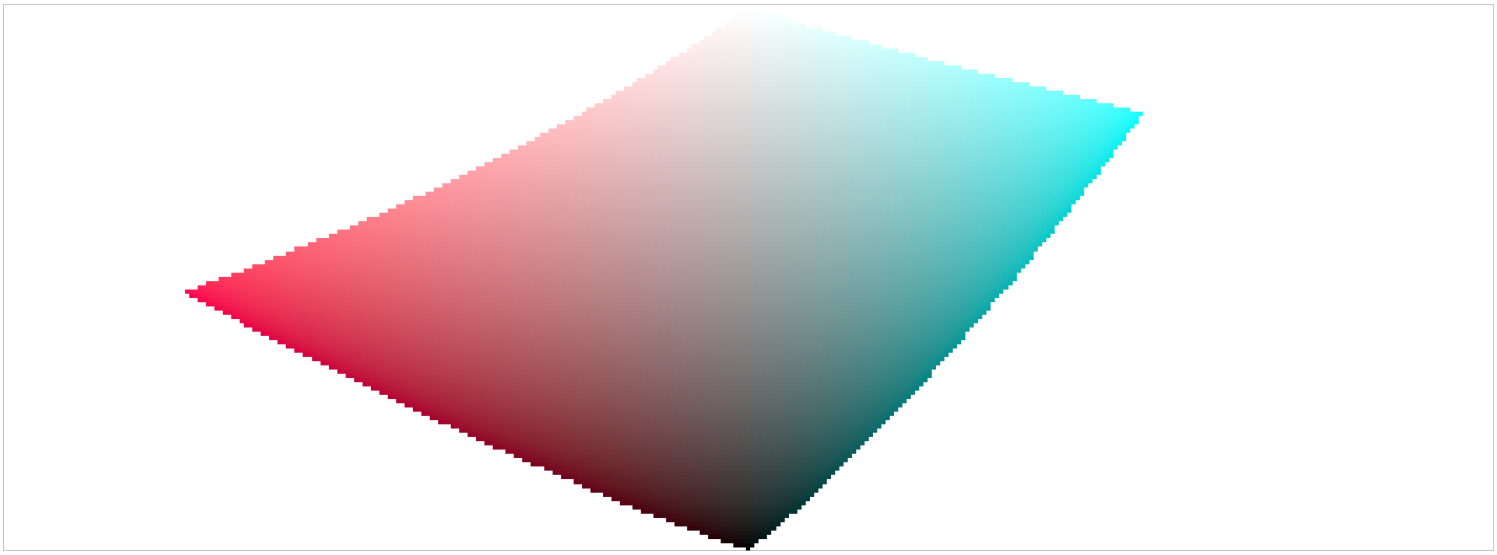


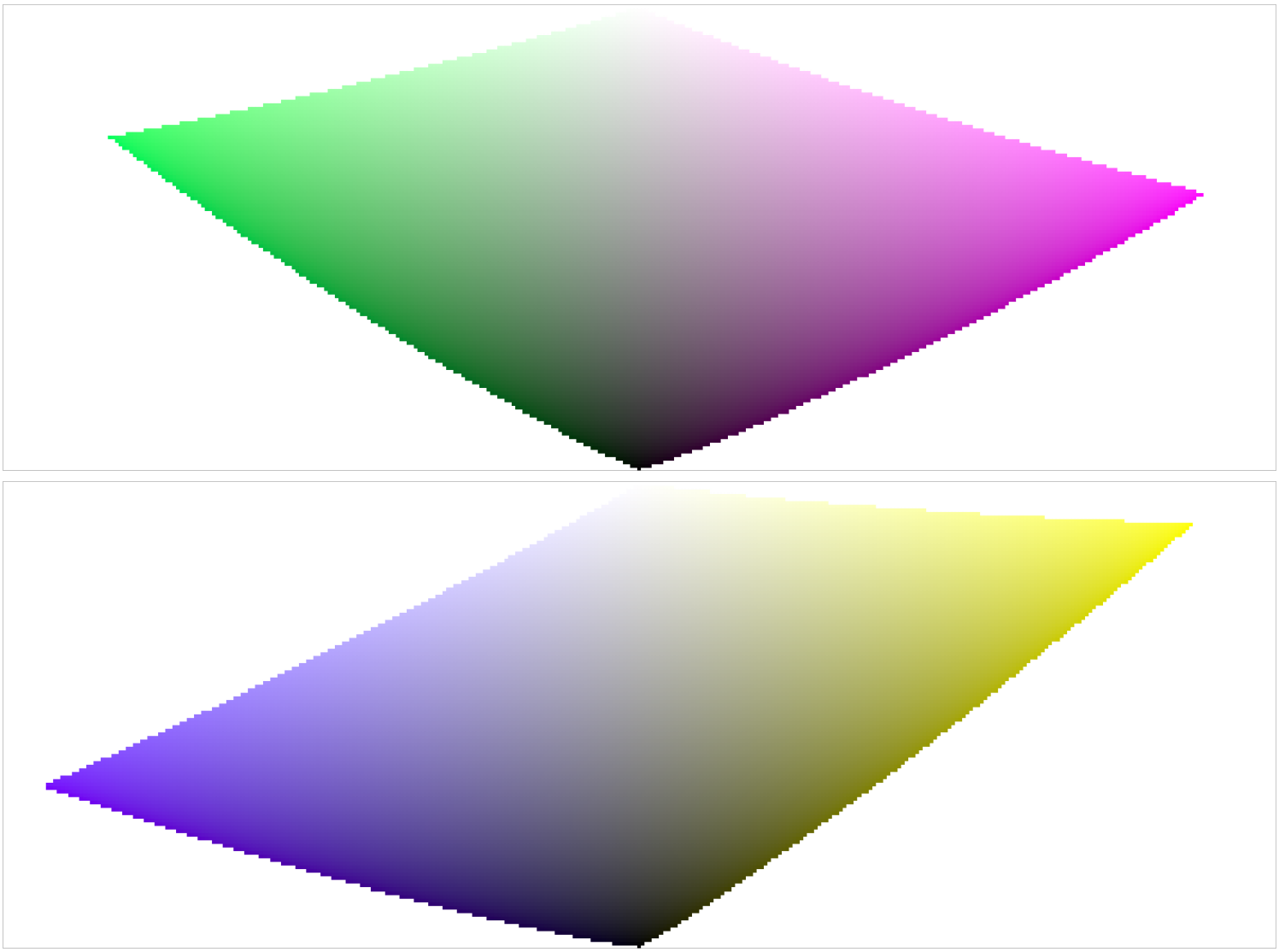
This shows that increasing the chroma (displacement over the horizontal axis) of some quantity can be safe for some hues at some lightness, but can push other hue-lightness coordinates way out of gamut. For example, we have much more margin in green or magenta than in cyan.

Many gamut issues, at export, are actually user-induced and the result of harsh chroma enlarging. For that reason, using brightness-saturation color models may be safer.

Color dimensions and complementary colors

Cyan, magenta, yellow (CMY) are complementary colors of red, green, blue (RGB). However, the complementary CMY spaces computed from RGB spaces are not perceptualy complementary. To show this, we create a CMY space from sRGB, where cyan has sRGB coordinates (0, 1, 1), magenta (1, 0, 1) and yellow (1, 1, 0), and display it in a lightness-chroma space:





By comparing with the hue slices of the primary colors in the previous section, it is easy to see not only that the gamuts don’t have the same shapes, but that the colors do not match.

This is one more reason to avoid using HSL/HSV spaces (derived from RGB spaces) to perform color editing – since these RGB spaces are not perceptually uniform in the first place, the resulting HSV/HSL spaces are not uniform either. While RGB spaces have some merit based on their connection to physical light, any process involving hue should go directly to perceptual spaces.

Color dimensions and settings

Many pieces of software, including darktable, call any settings that affect chroma “saturation” (for example, in color balance, “contrast/brightness/saturation”). This is a symptom of software trying to be accessible to non-professionals by using a common language. This is misleading, since saturation does exist and is quite different from chroma. In addition, many video specifications improperly call chroma “saturation”. Whenever darktable reuses such specifications, it uses the incorrect term from the specification rather than the proper color dimension term.

1. https://cie.co.at/eilvterm/17-22-067 ↩
2. https://cie.co.at/eilvterm/17-21-050 ↩
3. https://cie.co.at/eilvterm/17-22-059 ↩
4. https://cie.co.at/eilvterm/17-22-063 ↩
5. https://cie.co.at/eilvterm/17-22-074 ↩
6. https://doi.org/10.1002/col.20128 ↩
7. https://cie.co.at/eilvterm/17-22-073 ↩

12.2. memory

darktable's memory requirements are high. A simple calculation makes this clear. If you have a 20MPx image then, for precision reasons, darktable will store this internally as a 4 x 32-bit floating point cell for each pixel. Each full image of this size will therefore 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 merely to store and process image data. On top of that 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 needs a minimum of about 4GB of memory to run happily.

total system memory

From the above analysis, 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.

Theoretically, you could run darktable with lower amounts of physical RAM and balance this with enough swap space. However, you this could cause your system to "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. A solid-state drive can ease the pain slightly.

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, darktable splits the image into smaller parts for every processing module. This still requires one full input and output buffer, but intermediate buffers can be made small enough to fit everything into the hardware's limits.

memory fragmentation

Unfortunately this is not yet the full story. An effect called memory fragmentation can and will hit software that needs to perform 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. However this 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 separate 300MB areas. If there is no other free area of 1.5GB available, a subsequent 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's mip map cache allocates several small memory blocks per thumbnail, so this problem is even bigger. For this reason, as of darktable 2.0, 32-bit support is soft-deprecated.

further limitations

As if this were not challenging enough, there are further things that might limit darktable's access to memory. On some older boards you need to activate the "memory mapping" BIOS option 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 should consult your BIOS manual and the information about your Linux variant for further help.

setting up darktable on 32-bit systems

As we've seen, 32-bit systems are difficult environments for darktable. Still some users are 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 parameters that require adjustment in order 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, 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 thumbnails for many of your pictures.

If this is the case, take a minute to optimize your preference settings. You will find them in preferences > cpu/gpu/memory .

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 to run in parallel when importing film rolls or doing other background stuff. For obvious reasons on 32-bit systems you can only have one thread consuming resources at a time. So you need set this parameter to 1; anything higher will kill you.

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 thumbnail 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. Since darktable 2.0, the cache does allocate a few small buffers per thumbnail in cache, thus causing significant memory fragmentation. As explained before, this poses a problem for 32-bit systems. For this reason, as of darktable 2.0, 32-bit support is soft-deprecated.

darktable on 64-bit systems

There’s not much to be said here. Of course 64-bit systems also 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 cost. You can even run old 32-bit binaries on a 64-bit Linux. The only thing you need to do is 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.

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. If 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 uses all of them in parallel anyway – but hiding I/O latency.

One exception is worth mentioning. 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 cause even a well equipped 64-bit system to run out of memory. In that case lower the number of background threads to only one.

12.3. opencl

12.3.1. the background

Processing high resolution images is a demanding task needing a modern computer. Both in terms of memory requirements and CPU power, getting the best out of a typical 15, 20 or 25 Megapixel image can quickly take your computer to its limits.

darktable’s requirements are no exception. All calculations are performed on 4 x 32bit floating point numbers. This is slower than “ordinary” 8 or 16 bit integer algebra, but eliminates all problems of tonal breaks or loss of information.

A great deal of optimization has been undertaken 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 darktable to take advantage of the enormous power of modern graphics cards. Gamers' demand for highly detailed 3D worlds in modern shooters has fostered GPU development. AMD/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 that can 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. This is especially true when you want to repeat the same few processing steps millions of items. Typical use case: processing of high megapixel images.

12.3.2. how opencl works

As you can imagine, the hardware architecture of GPUs can vary significantly. There are different manufacturers, and even different generations of GPUs from the same manufacturer may not be comparable. At the same time GPU manufacturers don't normally disclose all the hardware details of their products to the public. One of the consequences of this 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 your GPU to be used 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 compiler at run-time. This way the application can use OpenCL on different GPU architectures (even at the same time). All of the 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 perform calculations for you.

12.3.3. activating 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. Most modern graphics cards from NVIDIA and AMD come with full OpenCL support. The OpenCL compiler is normally shipped as part of the proprietary graphics driver and is used as a dynamic library called `libOpenCL.so`. This library must be in a folder where it can be 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 for darktable to take advantage of the GPU. If that check passes, 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`) needs to be read and compiled and the included routines (called OpenCL kernels) need to be prepared for darktable's modules. If all that completes successfully, the preparation is finished.

By 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 preferences > cpu/gpu/memory . This configuration parameter is grayed out if the OpenCL initialization failed.

You can switch OpenCL support off and on at any time without requiring a restart. Depending on the type of modules you are using, you will notice the effect as a general speed-up during interactive work and 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, use modules like *shadows and highlights* , *sharpen* , *lowpass* , *highpass* or even more extreme *contrast equalizer* and *denoise (profiled)* .

If you are interested in profiling statistics, you can start darktable with command line parameters `-d opencl -d perf`. After each run of the pixelpipe you will be shown details of processing time for 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 for some rounding errors, the results are designed to be identical. If, for some reason, darktable fails to properly finish a GPU calculation, it will normally detect the failure and automatically (and transparently) fall back to CPU processing.

12.3.4. setting up opencl

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 loads `libOpenCL.so`, a system library which must be accessible to the system's dynamic loader (`ld.so`).
- `libOpenCL.so` reads 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/nvidiactl`, 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 by installing the `nvidia-modprobe` package.

A user account which needs to make use of OpenCL from within darktable must 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 this 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-uvm (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/nvidiactl
crw-rw-rw- 1 root root 250,   0 Jul 28 21:13 /dev/nvidia-uvm
```

Beware that the major/minor numbers (e.g. `250/0` for `/dev/nvidia-uvm` in this example) may vary depending on your system.

12.3.5. possible problems & solutions

darktable will detect OpenCL run-time errors automatically. On detecting an error, it will then reprocess everything on the CPU. While this will slow down processing it should not affect the end result.

There can be various reasons why OpenCL might fail during the initialization phase. OpenCL depends 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 will likely not be available.

In this case, the best thing to do is start darktable from a console with `darktable -d opencl`.

This will give additional debugging output about the initialization and use of OpenCL. First, if you find a line that starts with `[opencl_init] FINALLY ...` that should tell you whether 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 that have been observed in the past:

- darktable states 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 finds your `libOpenCL.so` but then tell you that it couldn't get a platform. NVIDIA drivers will often give error code -1001 in this case. This happens because `libOpenCL.so` is only a wrapper library. For the real work further vendor-specific libraries need to be loaded. This has 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 required libraries cannot be found by your system's dynamic loader. Giving full path names might help.
- darktable states that a context could not be created. This often indicates a version mismatch between the loaded graphics driver and libOpenCL. Check if you have left-over kernel modules or graphics libraries from an older installation and take appropriate action. When in doubt, perform a clean reinstall of your graphics driver. Sometimes, immediately after a driver update, the loaded kernel driver does not match the newly installed libraries. In this case reboot your system before trying again.
- darktable crashes during startup. This can happen if your OpenCL setup is completely broken or if your driver/library contains a severe bug. If you can't fix it, you can still use darktable with option `--disable-opencl`, which will skip the entire OpenCL initialization step.
- darktable fails to compile its OpenCL source files at run-time. In this case you will see a number of error messages looking like typical compiler errors. This could indicate an incompatibility between your OpenCL implementation and darktable's interpretation of the standard. In that case visit the developers in IRC in `#darktable` on FreeNode or on the developers mailing list at darktable-dev@lists.darktable.org 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.

A few on-CPU implementations of OpenCL also exist, coming as drivers provided by INTEL or AMD. We have observed that they do not provide any speed gain versus our hand-optimized CPU code. Therefore darktable simply discards these devices by default. This behavior can be changed by setting the configuration variable `opencl_use_cpu_devices` (in `$HOME/.config/darktablerc`) to TRUE.

12.3.6. amd/ati devices

While NVIDIA devices and most modern AMD/ATI devices will 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 part of their total GPU memory to darktable. For a 1GB device this typically amounts to only 512MB, a value which darktable in its standard configuration will refuse as insufficient. In 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 100 if this happens. 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) and 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 that this tends to cause darktable to crash sooner or later.

Our recommendation is therefore 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 can set in `$HOME/.config/darktable/darktablerc` to get things running. Here are the details:

opencl_memory_requirement
: Set this parameter to 500 so that darktable will accept your 512MB graphics memory as having sufficient memory.

opencl_memory_headroom
: This parameter controls how much graphics memory (from that reported by your card) darktable should leave untouched for driver and display use. Since for AMD/ATI devices we can only get half of the available RAM anyway, it's safe to set this to zero so that all of the 512MB can be used by darktable.

opencl_avoid_atomics
: Atomic operations in OpenCL are a special method 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 the CPU rather than accepting an ultra-slow GPU codepath. Therefore, set this parameter to TRUE if you experience slow processing of modules like *shadows and highlights* , *monochrome* , *local contrast* , or *global tonemap (deprecated)* or if you 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, though you might consider trying some of the performance optimization options described in the following section.

12.3.7. performance optimization

There are a number of configuration parameters in `$HOME/.config/darktable/darktablerc` that can 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 the pixelpipe. For a comfortable workflow it is essential to keep latency low.

In order to obtain profiling info you need to start darktable from a terminal with `darktable -d opencl -d perf`.

After each reprocessing of the pixelpipe – caused by module parameter changes, zooming, panning, etc. – you will see the total time and the time spent in each of the 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 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. These 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 flag controls how often darktable blocks the OpenCL pixelpipe to get a status on success/failure of the kernels that have been run. For optimum latency set this to TRUE, so that 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 that you can more easily isolate the problem. Problems have been reported with some older AMD/ATI cards, like the HD57xx, which can produce garbled output if this parameter is set to TRUE. If in doubt, leave it at its default of FALSE.

opencl_number_event_handles

Event handles are used so that darktable can monitor the success/failure of kernels and profiling info even if the pixelpipe is executed asynchronously. The number of event handles is a limited resource of your OpenCL driver. For sure they can be recycled but there is a limited number that can be used at the same time. Unfortunately, there is no way to find out what the resource limits are, so darktable needs to guess. The default value of 25 is quite conservative. You might want to see if higher values like 100 give better OpenCL performance. If your driver runs out of free handles you will experience failing OpenCL kernels with error code -5 (CL_OUT_OF_RESOURCES) or even crashes or system freezes. Reduce the number again if that happens. 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. This is 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 resource consuming operation, but can make sense depending on your GPU (including if the GPU is rather slow). In this 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 many cases this parameter should be set to “active module” (the default), which will only cache the input of the currently focused module.

opencl_micro_nap

In an ideal case you will keep your GPU busy at 100% when reprocessing the pixelpipe. That’s good. On the other hand your GPU may also be needed to do regular GUI updates. It might happen that there is no sufficient time left for this task. The consequence would by a jerky reaction of your GUI on panning, zooming or when moving sliders. To resolve this issue darktable can add small naps into its pixelpipe processing to have the GPU catch some breath and perform GUI related activities. The opencl_micro_nap parameter controls the duration of these naps in microseconds. You will 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. The default is 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.

12.3.8. scheduling profile

darktable can use the CPU and one or several OpenCL capable GPUs. Depending on the relative performance of these devices, users can choose among certain scheduling profiles to optimize performance. This is achieved by setting the configuration parameter preferences > cpu/gpu/memory > OpenCL scheduling profile which offers the following choices:

default

If an OpenCL capable GPU is found darktable uses it for processing the center image view while the navigation preview window is processed on the CPU in parallel. This is the preferred setting for systems with a reasonably fast CPU and a moderately fast GPU. The exact allocation of devices to the various pixelpipe types can be finetuned with the “opencl_device_priority” configuration parameter (see multiple devices).

very fast GPU

With this scheduling profile darktable processes the center image view and the preview window on the GPU sequentially. This is the preferred setting for systems with a GPU that strongly outperforms the CPU.

multiple GPUs

This setting addresses systems with multiple GPUs whose relative performance does not differ significantly. Whenever a processing job is started darktable uses any currently idle GPU but not the CPU. Users of systems with a variety of GPUs will need better control on their relative priority. They would be better off selecting the “default” profile and fine-tuning their system with the “opencl_device_priority” configuration parameter (see multiple devices).

On first start-up or after any detected change in the GPU configuration of your system darktable tries to identify the best suited profile for you. You can change it at any time in preferences > cpu/gpu/memory .

12.3.9. multiple devices

The scheduling of OpenCL devices on most systems can be optimized using the “OpenCL scheduling profile” settings. However, if your system is equipped with more than one GPU you might want to set the relative device priority manually. To do this you need to select the “default” scheduling profile and change the settings in the “opencl_device_priority” configuration parameter.

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 history stack – is run in a pixelpipe. There are four different types of pixelpipe in darktable. One type is responsible for processing the center image view (or full view) in darkroom mode, another pixelpipe processes the preview image (navigation window). There can be one of each of these two pixelpipes running at any one time – with the full and preview pixelpipes running in parallel. In addition there can be multiple parallel pixelpipes performing file exports as well as 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 varies significantly depending on the type of pixelpipe being executed. The preview image and thumbnails are low resolution and can be processed quickly. Processing the center image view is more demanding. A full export pixelpipe is more demanding still.

The 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, whereby the first available OpenCL device receives the 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 with the same name you need to use the device numbers in order to differentiate them.

A device specifier can be prefixed with an exclamation mark `!`, in which case the device is excluded from processing a given pixelpipe. You can also use an asterisk * as a wildcard, representing all devices not previously explicitly mentioned 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.

If a pixelpipe process is about to be started and all GPUs in the corresponding group are busy, darktable automatically processes the image on the CPU by default. You can enforce GPU processing by prefixing the list of allowed GPUs with a plus sign +. In this case darktable will not use the CPU but rather suspend processing until the next permitted OpenCL device is available.

darktable’s default setting for “opencl_device_priority” is `*/!0,*/*/*`.

Any detected OpenCL device is permitted to process the center view image. The first OpenCL device (0) is not permitted to process the preview pixelpipe. As a consequence, if there is only one GPU available on your system, the preview pixelpipe will always be processed on the CPU, keeping your single GPU exclusively for the more demanding center image view. This is a reasonable setting for most systems. No such 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 some time to 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.
```

Here, the GeForce GTS 450 is detected as the first device; the Radeon HD7950 ('Tahiti') as the second. 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 processing the center image pixelpipe; this is reserved to "all" other devices (i.e. the HD7950/Tahiti). Conversely, for the preview pixelpipe, the Tahiti is excluded, so that only the GTS450 is permitted to do the work.

For file export and thumbnail generation we want all hands on deck. However, darktable should first check whether the Tahiti device is free, because it's faster. If it is not free, then all other devices – in fact only the GTS450 – are checked.

12.3.10. opencl still does not run for me

As has been mentioned, OpenCL systems come with a huge variety of setups: different GPU manufacturers and models, varying amounts of GPU memory, different drivers, different distributions etc..

Many of the potential problems will only appear with very specific combinations of these factors. As the darktable developers only have access to a small fraction of these variations, please understand that we might not be able to fix your specific problem. There is not much we can do if we are unable to reproduce your issue.

If nothing else helps, the best option is probably to start darktable with

```
darktable --disable-opencl
```

In the end there is nothing in darktable which only runs on GPU. Don't let the lack of OpenCL discourage you – darktable's CPU code is also highly optimized for performance!

12.4. using darktable-chart

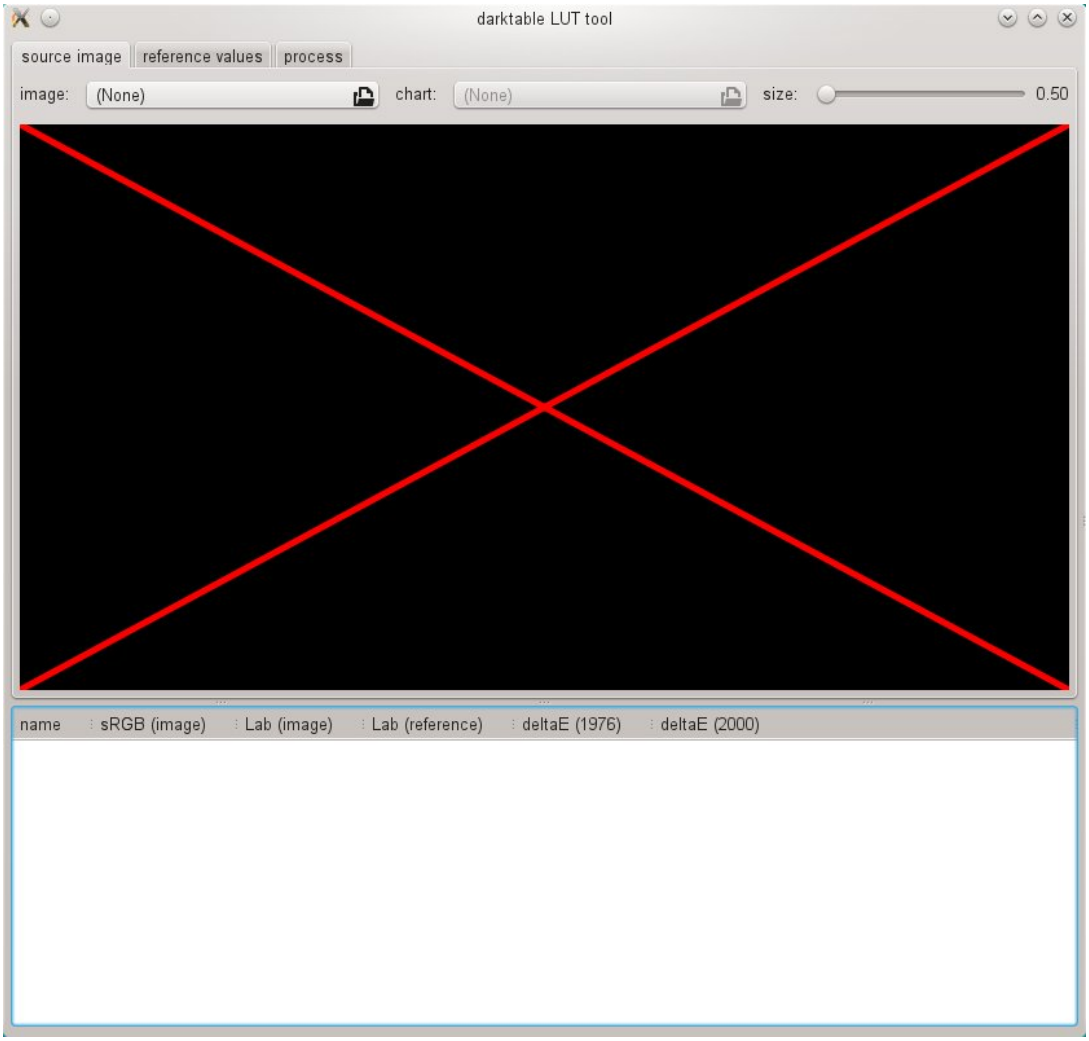
12.4.1. overview

`darktable-chart` is a tool for extracting luminance and color values out of images taken from color reference cards such as IT8.7/1 charts. Its main purpose is to compare a source image (typically a largely unprocessed raw image) to a target image (typically a JPEG image created in-camera) and produce a darktable style that is able to use the luminance and color values of the source image to produce the target image. This style employs the *tone curve* module, the *input color profile* module, and the *color look up table* module for that purpose.

Some cameras offer various film simulation modes of your choice. With the help of `darktable-chart` and the underlying modules you can create styles that replicate these film simulations from within darktable.

12.4.2. usage

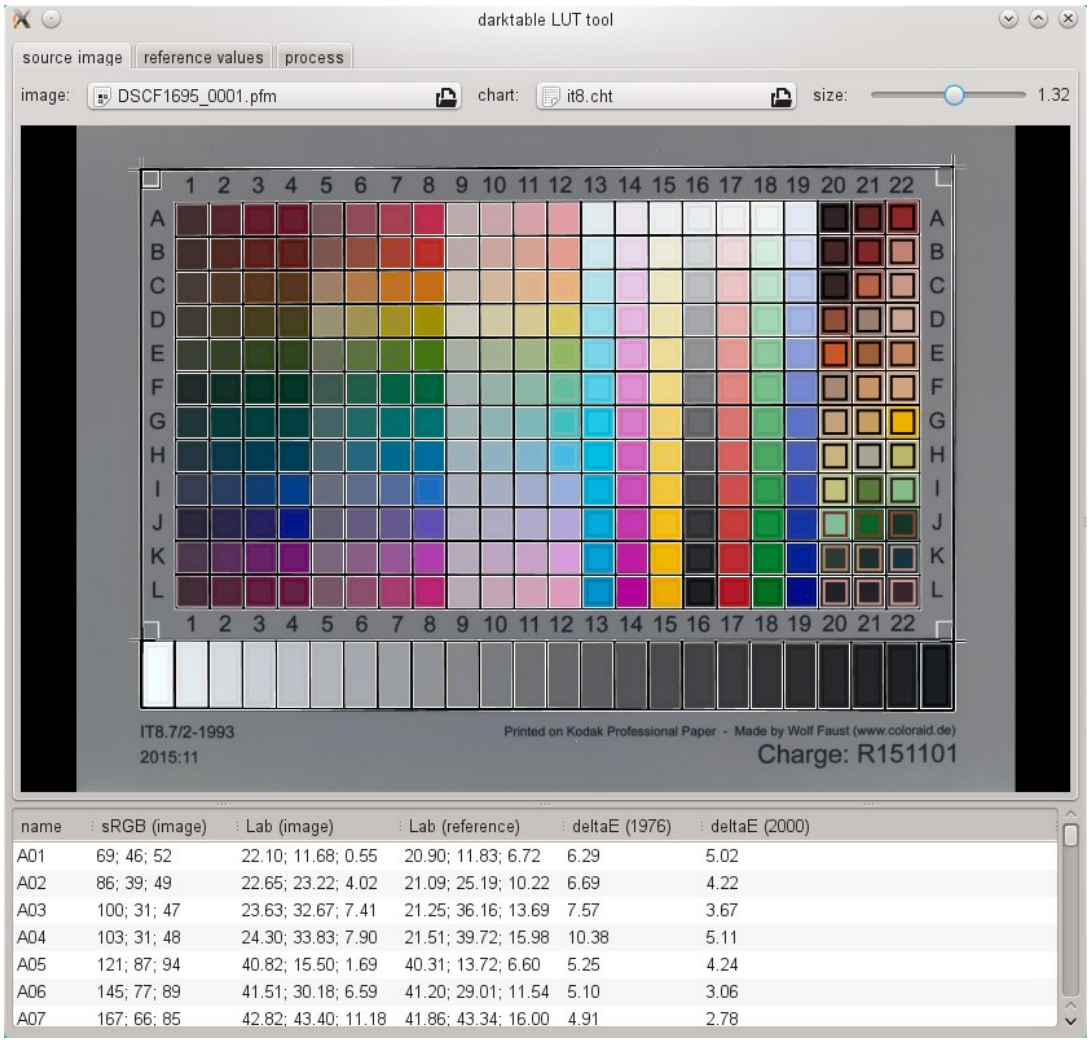
The tool is organized into three tabs in the upper part and a text output frame in the lower part.



The first tab is used to define the source image, the second tab defines the reference (target) and the third tab contains the controls to generate the resulting darktable style.

12.4.3. source image

In the “source image” tab you set your source image, which requires two elements. The first element is an input file in Lab Portable Float Map format (extension `.pfm`). The source file represents the largely unmodified data as the camera sees it. Information about how to take photos of a color reference card and produce a `.pfm` output file are described below. The second element is a chart file that contains a formal description of the underlying color reference card’s layout (extension `.cht`). Chart files are usually shipped with your color reference card or can be downloaded from the internet.



name	sRGB (image)	Lab (image)	Lab (reference)	deltaE (1976)	deltaE (2000)
A01	69; 46; 52	22.10; 11.68; 0.55	20.90; 11.83; 6.72	6.29	5.02
A02	86; 39; 49	22.65; 23.22; 4.02	21.09; 25.19; 10.22	6.69	4.22
A03	100; 31; 47	23.63; 32.67; 7.41	21.25; 36.16; 13.69	7.57	3.67
A04	103; 31; 48	24.30; 33.83; 7.90	21.51; 39.72; 15.98	10.38	5.11
A05	121; 87; 94	40.82; 15.50; 1.69	40.31; 13.72; 6.60	5.25	4.24
A06	145; 77; 89	41.51; 30.18; 6.59	41.20; 29.01; 11.54	5.10	3.06
A07	167; 66; 85	42.82; 43.40; 11.18	41.86; 43.34; 16.00	4.91	2.78

In real life the photo taken from the color reference card will show some perspective distortions relative to the layout defined in the chart file. For that reason the layout is displayed as a grid over the image and can be modified.

You can move the corners of the grid using the mouse to reach best alignment of grid and image.

A rectangular frame is displayed for each patch and defines the area from which `darktable-chart` will sample the required input data. You may need to modify the size of these rectangles so that the sampling area is big enough but does not overlap with neighboring patches. Use the “size” slider in the upper right part of the GUI. Higher values lead to smaller sizes.

12.4.4. reference values

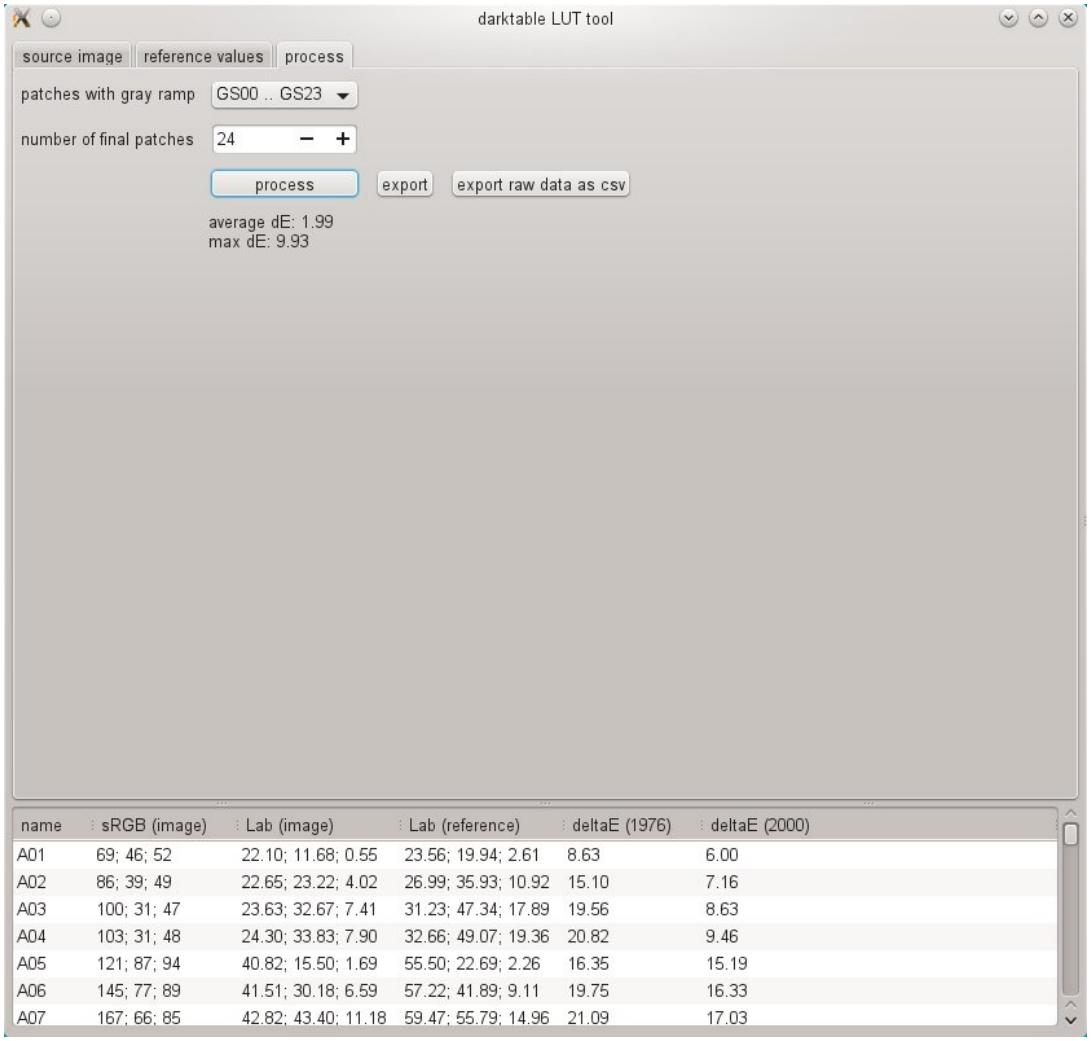
The “reference values” tab determines the target values to which the source image must to be modified by the resulting style. You can either supply reference values in the form of measured data of your color reference card (mode “cie/it8 file”), or you can supply a photographic image (mode “color chart image”) much in the same way as described above. This second image must also be supplied in Lab Portable Float Map format. There is no need to supply the chart file again as `darktable-chart` takes the same one as defined under “source image”. You only need to again align the layout grid and the image and potentially adjust the “size” slider.

In a typical use case the second image will be based on a JPEG file produced in-camera. This way you can create a style to simulate the in-camera processing within darktable.

In the lower text output frame you will see the color values extracted from the available data for each individual color patch. The first column gives the name of the patch, the second and third column show the corresponding color values of the source image in RGB and Lab format, respectively. The fourth column contains the Lab value coming from the reference (or from the chart file if no reference image has been given). Finally, the fifth and sixth columns display how strongly source and reference values deviate in terms of delta-E values.

12.4.5. process

If all of the required settings in the “source image” and “reference values” tabs are ready you can move on to the “process” tab.



First, you need to tell `darktable-chart` which of the patches represents the gray ramp. In the screenshot displayed above, the gray ramp is positioned in the lower part of the color reference chart, denoted as “GS00...GS23”.

The “number of final patches” input defines how many editable color patches the resulting style will use within the *color look up table* module.

Click the “process” button to start the calculation.

The quality of the result (in terms of average delta-E and maximum delta-E) is displayed below the button. This data shows how closely the resulting style (when applied to the source image) will be able to match the reference values – the lower the better.

Once you are happy with the result you can click on “export” to save the generated style.

Supply a style name and a style description under which the style will later appear in darktable. `darktable-chart` saves the style as a `.dtstyle` file which can be imported into darktable and shared with others. See styles .

The “export raw data as csv” button allows you to save the extracted raw data as a CSV file for debugging purposes or later use. `darktable-chart` offers a command line option to produce a style with the desired number of final patches from a supplied CSV file (see `darktable-chart`).

12.4.6. making input images for darktable-chart

To start with, you need a suitable photo of your color reference card in RAW+JPEG format. It goes beyond the scope of this manual to explain the details of how to take this photo, but in a nutshell you need to make the shot on a sunny day around midday with the light source (the sun) shining at an angle onto the card. You need to avoid any glare in the image. The neutral white color patch in the gray ramp (G00) should end up at the L value specified in the description of your card. Often this is L=92 and requires you to overexpose the shot by about 1/3 EV. Ideally you will make several shots with slightly different exposures and later select the right one in darktable. Make sure that the chart fills up most of the frame. Use a lens with a “normal” focal length (e.g. 50mm equivalent) and stop down a bit to avoid vignetting.

You then open the raw file in darktable and disable most modules, especially *base curve* . Select the standard input matrix in the *input color profile module* and disable gamut clipping. Select “camera white balance” in the *white balance* module.

There is a special situation if your camera automatically applies some lens corrections (namely vignetting correction) to the resulting JPEG file. In this case you need to activate the *lens correction* module in darktable so that raw processing matches the JPEG in this respect. However, since darktable’s vignetting correction may not exactly match the in-camera correction, it is better to disable this correction in-camera if possible.

To output your image go to the export module in the lighttable.

You will need to select “Lab” as the output color profile. This color space is not visible in the combobox by default. You first need to enable it by setting `allow_lab_output` to `TRUE` in `$HOME/.config/darktable/darktablerc`. Alternatively, you can start darktable with:

```
darktable --conf allow_lab_output=true
```

Then select “PFM (float)” as the output format and press “export” to generate the source image file.

You can prodce the corresponding reference (target) image from the JPEG in a similar way. This time you will need to disable all modules and export with the “Lab” output color profile in “PFM (float)” format.

12.5. program invocation

12.5.1. darktable

The `darktable` binary starts darktable with its GUI and full functionality. This is the standard way to use darktable.

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

```
darktable [-d {all,cache,camctl,camsupport,control,dev,
               fswatch,input,lighttable,lua,masks,memory,nan,
               opencl,perf,pwstorage,print,sql,ioporder,
               imageio,undo,signal}]
          [<input file>|<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>]
          [--noiseprofiles <noiseprofiles json file>]
          [--d-signal <signal>]
          [--d-signal-act <all,raise,connect,disconnect,print-trace>]
          [--conf <key>=<value>]
          [-t <num openmp threads>]
```

All parameters are optional. In most cases darktable should be started without any additional parameters, in which case darktable uses suitable defaults.

-d
{all,cache,camctl,camsupport,control,dev,fswatch,input,lighttable,lua,masks,memory,nan,opencl,perf,pwstorage,pri
Enable 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.

<input file>|<image folder>
Optionally supply the name of an image file or folder. 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
Print the darktable version number, a copyright notice, some other useful information, and then terminate.

--disable-opencl
Prevent darktable from initializing the OpenCL subsystem. Use this option if 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`. Use this option to provide 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 provide `:memory:` as the library file, in which case the database is kept in system memory – all changes are discarded when darktable terminates.

Whenever darktable starts, it will lock the library to the current user. It does this by writing the current process identifier (PID) into a lock file `<library file>.lock` next to the library specified. If darktable finds an existing lock file for the library, it will terminate immediately.

--datadir <data directory>
Define the directory where darktable finds its runtime data. The default place depends on your installation. Typical locations 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. This option tells darktable where to look for its shared libraries. The default location depends on your installation. Typical locations are `/opt/darktable/lib64/darktable/` and `/usr/lib64/darktable/`.

--tmpdir <tmp directory>
Define where darktable should store its temporary files. If this option is not supplied darktable uses the system default.

--configdir <config directory>
Define the directory where darktable stores the user specific configuration. The default location is `$HOME/.config/darktable/`.

--cachedir <cache directory>
darktable keeps a cache of image thumbnails for fast image preview and of precompiled OpenCL binaries for fast startup. By default the cache is located in `$HOME/.cache/darktable/`. Multiple thumbnail caches may exist in parallel – one for each library file.

--localedir <locale directory>
Define where darktable can find its language specific text strings. The default location depends on your installation. Typical locations 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.

--noiseprofiles <noiseprofiles json file>
Provide a json file that contains the camera specific noise profiles. The default location depends on your installation. Typical locations are `/opt/darktable/share/darktable/noiseprofile.json` and `/usr/share/darktable/noiseprofile.json`.

--d-signal <signal>
If `-d signal` or `-d all` is specified, specify signal to debug using this option. Specify `ALL` to debug all signals or specify signal using it’s full name. Can be used multiple times.

--d-signal-act <all,raise,connect,disconnect,print-trace>
If `-d signal` or `-d all` is specified, specify signal action to debug using this option.

--conf <key>=<value>
darktable supports a rich set of configuration parameters which the user defines in `$HOME/.config/darktable/darktablerc`. You may temporarily overwrite individual settings on the command line with this option – however, these settings will not be stored in “darktablerc” on exit.

-t <num openmp threads>
limit number of openmp threads to use in openmp parallel sections

12.5.2. darktable-cli

The `darktable-cli` binary starts the command line interface variant of darktable which allows images to be exported.

This variant does not open any display – it works in pure console mode without launching a GUI. This mode is particularly useful for servers running background jobs.

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

```
darktable-cli [<input file or folder>]
              [<xmp file>]
              <output file or folder>
              [--width <max width>]
              [--height <max height>]
              [--bpp <bpp>]
              [--hq <0|1|true|false>]
              [--upscale <0|1|true|false>]
              [--style <style name>]
              [--style-overwrite]
              [--apply-custom-presets <0|1|false|true>]
              [--out-ext <extension>]
              [--import <file or dir>]
              [--icc-type <type>]
              [--icc-file <file>]
              [--icc-intent <intent>]
              [--verbose]
              [--help [option]]
              [--core <darktable options>]
```

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

<input file or folder>
The name of the input file or folder (containing images) to be exported. If you wish to process multiple images or multiple folders use the `--import` option instead.

<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 provided darktable will search for an XMP file that belongs to the given input file(s).

<output file or folder>
The name of the output file or destination folder. The export file format is derived from the file extension or from `--out-ext` option. You can also use a number of variables in the output filename. For obvious reasons this parameter is mandatory if you use the program on an image folder containing multiple images. If you specify output folder it is recommended that you also specify the file format with `--out-ext`.

--width <max width>
Limit the width of the exported image to the given number of pixels.

--height <max height>
Limit the height of the exported image to the given number of pixels.

--bpp <bpp>
Define the bit depth of the exported image. Permitted values depend on the output file format.

Note: This option is not currently 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>
Define whether to use high quality resampling during export (see the export module for more details). Defaults to true.

--upscale <0|1|true|false>
Define whether allow upscaling during export. Defaults to false.

--style <style name>
Specify the name of a style to be applied during export. If a style is specified, the path to the darktable configuration directory must also be specified (i.e. `--core --configdir ~/.config/darktable`). By default no style is applied.

--style-overwrite
The specified style overwrites the history stack instead of being appended to it.

--apply-custom-presets <0|1|false|true>
Whether to load `data.db` which contains presets and styles. Disabling this option allows you to run multiple instances of `darktable-cli` at the cost of being unable to use the `--style` option. Defaults to true.

--out-ext <extension>
Set the output extension to use. If specified takes precedence over `<output file>`. By default this is extracted from `<output file>`. Defaults to `jpg` if `<output folder>` is specified.

--import <file or dir>
Specify input file or folder, can be used multiple times. This option cannot be combined with `<input file or folder>`.

--icc-type <type>
Specify the ICC profile type, which is the same as specifying `output profile` in `output color profile` module. Defaults to “image specified”. Use `--help icc-type` to obtain a list of the supported types. See output color profile for a more detailed description of the available options.

--icc-file <file>
Specify ICC profile filename. Defaults to an empty filename.

--icc-intent <intent>
Specify rendering intent. Defaults to “image specified”. Use `--help icc-intent` to obtain a list of the supported intents. See rendering intent for a more detailed description of the available options.

--verbose
Enables verbose output.

--help [option]
Prints usage and exits. If `option` is specified, additionally prints usage for the given option.

--core <darktable options>
All command line parameters following `--core` are passed to the darktable core and handled as standard parameters. See the darktable binary section for a detailed description.

12.5.3. darktable-generate-cache

The `darktable-generate-cache` binary updates darktable’s thumbnail cache. Invoke this program to generate all missing thumbnails in the background when your computer is idle.

`darktable-generate-cache` can be called with the following command line parameters:

```
darktable-generate-cache
              [-h, --help]
              [--version]
              [--min-mip <0-7>] [-m, --max-mip <0 - 7>]
              [--min-imgid <N>] [--max-imgid <N>]
              [--core <darktable options>]
```

All parameters are optional. If started without parameters `darktable-generate-cache` uses reasonable defaults.

-h, --help
Display usage information and terminate.

--version
Display copyright and version information and terminate.

--min-mip <0 - 7>, -m, --max-mip <0 - 7>
darktable can store thumbnails with up to eight different resolution steps for each image. These parameters define the maximum resolutionw which should be generated (defaults to a range of 0-2). There is normally no need to generate all possible resolutions here; missing ones will be automatically generated by darktable the moment they are needed. When asked to generate multiple resolutions at once, the lower-resolution images are quickly downsampled from the highest-resolution image.

--min-imgid <N>, --max-imgid <N>
Specifies the range of internal image IDs from the database to work on. If no range is given, `darktable-generate-cache` will process all images.

--core <darktable options>
All command line parameters following `--core` are passed to the darktable core and handled as standard parameters. See the darktable binary section for a detailed description.

12.5.4. darktable-chart

The `darktable-chart` binary is a dedicated utility to create styles out of pairs of images such as RAW+JPEG with in-camera processing. Details about its usage can be found in the using darktable-chart section.

`darktable-chart` can either start a GUI or be used as a command-line program.

```
darktable-chart
              [--help]
              [<input Lab pfm file>]
              [<cht file>]
              [<reference cgats/it8 or Lab pfm file>]
```

All parameters are optional, however, if you want to supply the second file name you also need to supply the first one etc. Starting darktable-chart this way opens a special GUI (details can be found in the using darktable-chart section).

--help
Provide usage information and terminate.

<input Lab pfm file>
Open the utility with the given file as source image. The input file needs to be in Lab Portable Float Map format.

<cht file>
Specify a chart file describing the layout of the used color reference chart.

<reference cgats/it8 or Lab pfm file>
Specify the reference values, either as measured values according to the CGATS standard, or as a reference image in Lab Portable Float Map format.

Alternatively `darktable-chart` can be used as a command line program to generate darktable style files out of previously saved CSV files.

```
darktable-chart
              --csv
              <csv file>
              <number patches>
              <output dtstyle file>
```

All parameters are mandatory.

<csv file>
A CSV file previously saved from within darktable-chart.

<number patches>
The number of color patches to be used in the color look up table settings of the created style.

<output dtstyle file>
The name of the style file to be created.

12.5.5. darktable-cltest

The `darktable-cltest` binary checks if there is a usable OpenCL environment on your system that darktable can use. It emits some debug output that is equivalent to calling `darktable -d opencl` and then terminates.

`darktable-cltest` is called without command line parameters.

12.5.6. darktable-cmstest

The `darktable-cmstest` binary (Linux only) investigates whether the color management subsystem of your computer is correctly configured and displays some useful information about the installed monitor profile(s).

`darktable-cmstest` is called without command line parameters.

12.6. variables

darktable supports variable substitution in a number of modules and preference settings. For example:

- Defining file names in the export module
- Displaying image information in the darkroom's image information line
- Displaying image information in the lighttable's overlays and tooltips (see preferences > lighttable)
- Placing text on an image in the *watermark* processing module

available variables

The following variables are available, though they may not all be available in all contexts:

```
$(ROLL_NAME)              roll of the input image
$(FILE_FOLDER)            folder containing the input image
$(FILE_NAME)              basename of the input image
$(FILE_EXTENSION)         extension of the input image
$(ID)                     the image id
$(VERSION)                the duplicate version number
$(VERSION_IF_MULTI)       same as $(VERSION) but null string if only one version exists
$(VERSION_NAME)           version name from metadata
$(SEQUENCE)               a sequence number within export job
$(MAX_WIDTH)              maximum image width to limit within export session
$(MAX_HEIGHT)             maximum image height to limit within export session
$(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
$(EXIF_ISO)               ISO value
$(EXIF_EXPOSURE)          Exif exposure
$(EXIF_EXPOSURE_BIAS)     Exif exposure bias
$(EXIF_APERTURE)          Exif aperture
$(EXIF_FOCAL_LENGTH)      Exif focal length
$(EXIF_FOCUS_DISTANCE)    Exif focus distance
$(LONGITUDE)              longitude
$(LATITUDE)               latitude
$(ELEVATION)              elevation
$(STARS)                  star rating
$(RATING_ICONS)           star rating (using star characters)
$(LABELS)                 colorlabels (text only)
$(LABELS_ICONS)           colorlabels (using colored bullet characters)
$(LABELS_COLORICONS)      colorlabels (using colored icons)
$(MAKER)                  camera maker
$(MODEL)                  camera model
$(LENS)                   lens
$(TITLE)                  title from metadata
```

```
$(DESCRIPTION)             description from metadata
$(CREATOR)                 creator from metadata
$(PUBLISHER)               publisher from metadata
$(RIGHTS)                  rights from metadata
$(TAGS)                    tags list (Xmp.dc.Subject)
$(CATEGORYn(category))     tag name of level n [0,9] of selected category (or tag)
$(SIDECAR_TEXT)            content of the text sidecar file (if any)
$(PICTURES_FOLDER)         pictures folder
$(HOME)                    home folder
$(DESKTOP)                 desktop folder
$(OPENCL_ACTIVATED)        whether OpenCL is activated
$(USERNAME)                user name defined by OS
$(NL)                      newline character
$(JOBCODE)                 internal jobcode of current job
```

string substitution

All of the variables support basic string substitution inspired by bash though some of the details differ.

All patterns are treated as simple string comparisons. There is no regex support.

The following string replacement functions are provided, where var is one of the variables listed above:

```
$(var-default)                  If var is empty, return "default"
$(var+alt_value)                If var is set, return "alt_value" else return empty string
$(var:offset)                   Return var starting from offset
                                If offset is negative count from the end of the string
$(var:offset:length)            Starting from offset, return at most length characters of var
                                If offset is negative the length is counted from the end of var
                                If length is negative this indicates the end of the result,
                                 counted from the end of var, and not an actual length
$(var#pattern)                  Remove "pattern" from the start of var
$(var%pattern)                  Remove "pattern" from the end of var
$(var/pattern/replacement)      Replace the first occurrence of "pattern" in var with "replacement"
                                If "replacement" is empty then "pattern" will be removed
$(var//pattern/replacement)     Replace all occurrences of "pattern" in var with "replacement"
                                If "replacement" is empty then "pattern" will be removed
$(var/#pattern/replacement)     If var starts with "pattern then "pattern" is replaced with "replacement"
$(var/%pattern/replacement)     If var ends with "pattern" then "pattern" is replaced with "replacement"
$(var^)                         Make the first character of var uppercase
$(var^^)                        Make all characters of var uppercase
$(var,)                         Make the first character of var lowercase
$(var,,)                        Make all characters of var lowercase
```

12.7. default module order

The following sections describe the default module order in the new scene-referred workflow and the legacy display-referred workflow. Note that in the following sections the module order goes from top (input file) to bottom (output image).

scene-referred module order

The default ordering of modules when using the new scene-referred workflow is as follows:

1. **raw black/white point**
2. invert (deprecated)
3. **white balance**
4. **highlight reconstruction**
5. chromatic aberrations
6. hot pixels
7. *raw denoise*
8. demosaic
9. denoise (profiled)
10. *surface blur*
11. *rotate pixels*
12. *scale pixels*
13. lens correction
14. haze removal
15. perspective correction
16. **orientation**
17. crop and rotate
18. liquify
19. spot removal
20. retouch
21. **exposure**
22. local tone mapping (deprecated)
23. tone equalizer
24. graduated density
25. unbreak input profile
26. *legacy equalizer*
27. **input color profile**
28. negadoctor
29. *astrophoto denoise*
30. color look up table
31. defringe
32. contrast equalizer
33. lowpass
34. highpass
35. sharpen
36. lut 3D
37. color mapping
38. channel mixer (deprecated)
39. basic adjustments
40. color balance
41. *rgb curve*
42. *rgb levels*
43. *base curve*
44. *filmic (legacy)*
45. **FILMIC RGB** – transition from scene-referred to display-referred space
46. *contrast brightness saturation*
47. *tone curve*
48. *levels*
49. *shadows and highlights*
50. *zone system (deprecated)*
51. *global tonemap (deprecated)*
52. *fill light (deprecated)*
53. local contrast
54. *color correction*
55. *color contrast*
56. velvia
57. vibrance
58. color zones
59. *bloom*
60. colorize
61. lowlight vision
62. *monochrome*
63. grain
64. soften

65. split-toning
66. vignetting
67. color reconstruction
68. **output color profile**
69. dithering
70. framing
71. watermark

key:

- *italic*: not recommended for scene-referred workflow
- **bold**: module on by default

legacy/display-referred module order

The default ordering of modules when using the legacy display-referred workflow is as follows:

1. **raw black/white point**
2. invert (deprecated)
3. **white balance**
4. **highlight reconstruction**
5. chromatic aberrations
6. hot pixels
7. raw denoise
8. demosaic
9. denoise (profiled)
10. local tone mapping (deprecated)
11. exposure
12. spot removal
13. retouch
14. lens correction
15. perspective correction
16. liquify
17. rotate pixels
18. scale pixels
19. orientation
20. crop and rotate
21. tone equalizer
22. graduated density
23. **BASE CURVE** – default transition between scene-referred and diaplay-refered space
24. surface blur
25. unbreak input profile
26. haze removal
27. **input color profile**
28. negadoctor
29. basic adjustments
30. color reconstruction
31. color look up table
32. defringe
33. legacy equalizer
34. vibrance
35. color balance
36. colorize
37. color mapping
38. bloom
39. astrophoto denoise
40. global tonemap (deprecated)
41. shadows and highlights
42. contrast equalizer
43. local contrast
44. color zones
45. lowlight vision
46. monochrome
47. filmic (legacy)
48. filmic rgb
49. contrast brightness saturation
50. zone system (deprecated)
51. tone curve
52. levels
53. rgb levels
54. rgb curve
55. fill light (deprecated)
56. color correction
57. **sharpen**
58. lowpass
59. highpass
60. grain
61. lut 3D
62. color contrast
63. **output color profile**
64. channel mixer (deprecated)

65. soften
66. vignetting
67. split-toning
68. velvia
69. dithering
70. framing
71. watermark

12.8. darktable's color pipeline

Most image processing applications come from the 1990s and/or inherit a 1990s workflow. These applications processed images encoded with 8 bit unsigned integers because it was more memory and computationally efficient. However, due to the use of an integer format (which implies rounding errors) they had to apply a “gamma” (essentially a transfer function applying a power 1/2.2 or 1/2.4 to encode the RGB values) and increase the bit-depth in the low-lights in order to reduce rounding errors there (humans are very sensitive to low-light details). The 8 bit integer formats are also technically limited to the 0-255 range. Anything outside of this range overflows and is clipped to the nearest bound.

These workflows, using bounded RGB representations and possibly non-linear transforms to encode RGB signals, are called “display-referred”. They rely on the assumption that the image has been prepared for display at an early stage in the processing pipeline, and embed hard-coded assumptions about the RGB values of black, middle-gray and white. Most of the image-processing algorithms used in these workflows have been tuned around these assumptions. For example, the darken and lighten blending modes expect a middle-gray encoded at 50% (or 128 in integer encoding).

Unfortunately the non-linear scaling, which is mandatory to make the integer encoding work, breaks the natural relationships between pixel values. Hue and saturation change in unpredictable ways, and value relationships between neighbouring pixels are dilated or compressed such that gradients are also altered unpredictably.

Display-referred pipelines therefore break optical filters (lens blurring or deblurring), alpha compositing (which rely on optical and geometrical definitions of occlusion), colors and gradients (local relationships between chrominance and luminance of pixels). They also don’t scale well to HDR images, which led to the development of many questionable local and global tonemapping methods and the infamous 2010s “HDR look”.

Modern computers are not tied to the same computational limitations as those from the 1990s, and can work on pixels whose values are completely unbounded (from 0 up to +infinity) and encoded as real numbers (using floating point formats). These possibilities enable what we call a “scene-referred” workflow, in which pixels can retain their original radiometric relationships along almost the entire processing pipe. In scene-referred workflow, pixels are prepared for display only at the last stage of the pipeline, in the display transform. This means that the RGB values of the pixels are kept proportional to the intensity of the light emission recorded by the camera on the scene, enabling accurate alpha compositing and optical filter emulations, while also scaling to any dynamic range through the same algorithm (SDR as well as HDR).

However, scene-referred pipelines lose the convenient fixed values of white, middle-gray and black which characterised display-referred pipelines, and setting these values, according to the scene and to the conditions of shooting, now becomes the responsibility of the user. This requires a more complex user interface.

Also, because scene-referred values are supposed to be physically-meaningful, pixels cannot have zero intensity. This would mean they have no light at all, and the existence of zero light breaks many physically-accurate algorithms. In fact, white and black mean nothing with regard to the original scene, which is only a collection of luminances at varying intensities. The scene-referred workflow simply aims at remapping some arbitrary scene luminances to what will appear white or black on the output medium.

Versions of darktable prior to 2.6 had a non-linear display-referred pipeline, assuming that a non-linear transform took place early in the pipe and middle-gray was thereafter encoded as 50%. However, not all modules and filters clipped pixel values above 100%, leaving open the possibility of recovering those values later in the pipe.

The *filmic* module’s view transform, introduced in darktable 2.6, was the first step toward a scene-referred pipeline, and deferred the mandatory, non-linear, display preparation to the end of the pipe, along with the ability to set custom black, gray and white values. The *color balance* module then introduced a way to deal with a variable definition of middle-gray.

Starting in darktable 3.2, users could choose between two workflows that defined consistent default settings, modules and pipeline order for both *display-referred* and *scene-referred* processing.

In darktable 3.4, a full scene-referred masking and blending option has been introduced, allowing masks to be defined for pixel values above 100% and using only unbounded blending operators.

Switching to *scene-referred* is a cognitive leap for most experienced users, who are used thinking in display-referred ways. In a display-referred workflow, it is customary to anchor the white value and let tone adjustments revolve around that point, trying to maximize brightness while avoiding clipping. In a scene-referred workflow, white and black values are fluid and adapted to the output medium. It is advised that users anchor middle-gray (which will be preserved as-is for any output medium) and let the view transform (*filmic*) dilate or contract the dynamic range around that point. Because 10 bit HDR white is 4 times as bright as 8 bit SDR white, any rigid definition of “white” becomes irrelevant. But anchoring for middle-gray is actually more convenient, since it keeps the average brightness of the picture unchanged through the view transform.

Some modules (*levels*, *rgb levels*, *tone curve*, *rgb curve*) are inherently incompatible with a scene-referred workflow, because their graphical interface implicitly suggests RGB values that are bounded within the 0-100% range. While the pixel operations they perform can be used in either scene-referred or display-referred workflows because they are unbounded internally, their controlling interface does not allow pixels to be selected outside of the 0-100% range.

Similarly, blending modes such as overlay, linear light, soft light, hard light, darken, brighten, etc. all have hard-coded thresholds that internally expect display-referred non-linear encoding.

In darktable 3.4, hovering the cursor over a module header shows a tooltip detailing the color spaces, ranges and encodings that the module expects, uses and produces. Here are the definitions of the terms used:

linear
: Pixel values are proportional to the scene radiometric emission, in a way that enables accurate emulation of physical filters.

non-linear
: Pixel values are re-scaled such that low-lights are given a larger encoding range, usually to remap the 18.45% reference middle-gray to a value between 46 and 50%.

display-referred
: Pixel values are expected to lie between 0 and 100% of the display range, where 100% is understood to be the luminance of a 20% reflective white surface (the white patch of a Color Checker) and 0% is understood to be the maximum density of the output medium (saturated black ink or minimum LED panel backlighting).

scene-referred
: Pixel values are expected to be greater than zero up to +infinity. The meaning of specific pixel values needs to be defined at runtime by the user. Scene-referred values don’t automatically imply a linear, radiometrically scaled, encoding.

12.9. contributing to dtdocs

This page defines the style guide for dtdocs and information about how to contribute to the project.

It is included in the user manual so that you can see how the page is rendered as well as how it is written. Please go to GitHub to see the source for this page.

The manual structure and content have been carefully considered based on the following critera:

1. The manual should be comprehensive – it should describe all of the functionality available in darktable
2. It should have a consistent and logical structure and every piece of functionality should have its own logical place within that structure
3. It should be as long as necessary but as short as possible – brevity is a must
4. It should be objective
5. Functionality should be explained once and only once (with the exception of the basic workflow guidelines in the overview section)
6. Images should be included only where necessary to improve understanding of key principles and should not contain text unless it is unavoidable

We are generally **not** interested in:

1. Restructuring the manual
2. Switching markup languages
3. Detailed workflow tutorials (though we are interested in publishing those on the blogs of either darktable.org or pixls.us)

We **are** interested in

1. Spelling and grammar corrections
2. Clarification of text
3. Documentation for new features

We are always extremely interested in hearing about which sections of the manual did not make sense to you and *why*, so that we can improve the documentation.

In general, if you wish to make a major change, please open an issue and discuss it with the maintainers first. This is to avoid doing work that wouldn’t be accepted.

format

This website is authored in pure markdown, using some extensions. It is initially designed to work with the Hugo SSG but indended to be portable enough that it can be easily rendered with another application if required.

File should be rendered in UTF-8 and should not include any column wrapping.

structure

The following shows the structure of an example main chapter with subsections in the dtdocs website.

```
example-chapter/
   _index.md
   section1-with-subsections/
      subsection1/
         image.png
      _index.md
      subsection1.md
      subsection2.md
   section2.md
   section3.md
```

A couple of notes on the above structure:

- `_index.md` files do not contain any content (they contain metadata only) and are used to render section headers and ToC entries. In the above example `example-chapter/_index.md` defines the title of the example chapter and the order in which it appears in the main table of contents. Similarly `example-chapter/section1-with-subsections/_index.md` defines metadata for the first section of the chapter.
- Media files should be contained in a directory with the same name as the page to which they relate. In this example, `example-chapter/section1-with-subsections/subsection1` contains media related to the `subsection1.md` page.

metadata

Metadata for the markdown files is presented at the head of the page using yaml. Any metadata may be defined – the module reference sections contain quite a lot of specific metadata – however the following defines some minimal metadata for the example page `example-chapter/section1-with-subsections/subsection1.md`.

```
---
title: Sub Section 1 Title
id: subsection1
weight: 10
---
```

title
This should contain the rendered title of your page. To include a colon within a title, enclose the title in double-quotes.

id
This is the id used to identify the page by Hugo. It should usually be the same name as the file (for content files) or the parent directory (for `_index.md` files).

weight
This is an optional metadata field used to define the order in which sections are presented in the Table of Contents. If the *weight* field is not included, pages will be rendered in alphabetic order by default. For example, to define the sections and subsections of the above example in reverse order, the following metadata would need to be set:

```
example-chapter/
   section1-with-subsections/
      _index.md              # weight: 30 (place section1 page at the end of example-chapter)
      subsection1.md            # weight: 20 (place subsection1 page at the end of section1)
      subsection2.md            # weight: 10 (place subsection2 page at the start of section1)
   section2.md               # weight: 20 (place section2 in the middle of example-chapter)
   section3.md               # weight: 10 (place section3 at the start of example-chapter)
```

content

general style guidance

- All content should be authored in plain markdown without shortcodes and HTML should be kept to an absolute minimum , if used at all
- Minimalism is an absolute must. Fewer words are preferred
- Markdown files should be as short as possible
- Follow the naming and capitalization norms present in the GUI of the application - namely all headers and titles are in lower case, except for the very top-level chapter names
- Headers in a file should not exceed level three (###)
- The primary authoring language is English
- Assume the reader has the application open while reading the user manual and only include images where they contribute to the explanation of complex functionality
- Use image callouts if you need to annotate an image (i.e. mark parts of the image with a letter or number and then explain the meaning in some text following the image). Do not place words directly into the image for annotations, as this makes localization difficult. See this page for an example.
- Changes to the content should be proposed via pull request or a similar mechanism
- Your submissions will be copy edited, don't take it personally

definition lists

The standard method of presenting information about darktable module controls is with the use of definition lists.

gui control name
A declaration of what the control does. For example "Set the exposure in EV units".

You can include as many paragraphs as you like, but try to limit to 2 or 3 where possible.

 **a control accessed through a button with an icon**
When a control is activated using an icon, take a screenshot of the icon using the standard darktable theme and add it before the name of the control

gui combobox name
Comboboxes often have multiple options that all need to be displayed with separate definitions. Use bulleted lists with *italics* for the combobox values.

- *the first value*: What the first value means
- *the second value*: What the second value means

Definition lists are also used throughout the document, wherever a named piece of functionality needs to be defined. See, for example, darktable-cli .

notes

If you wish to present an important note to the user, use the following format:

Note: This is an important note.

---

fixed-width fonts and code blocks

Fixed width fonts (using the ` character) should normally only be used for code blocks and when referencing file names and command line parameters.

links

Internal links must be relative to the current file and must point to a valid markdown (.md) file. Start links with either `./` to represent the current directory or `../` to represent the parent directory.

- Links to a processing module should be in italics, e.g. *exposure*
- Links to a utility module should be in plaintext, e.g. history stack
- Link to a top level section by referencing the `_index.md` file, e.g. module reference
- Link to a tab in the preferences dialog: preferences > general
- Link to a specific preference setting: preferences > general > interface language
- Each header within a page can be linked to directly with an anchor link: contributing/notes

images

When taking screenshots from the darktable application itself, use the default darktable theme.

Several filename suffixes can be used to control how an image is rendered.

icon

To insert an image as an icon, include #icon after the image name in the link. The markdown `![squirrel icon](./contributing/contributing.png#icon)` outputs the following: 

image width

You can set the image width to 25, 33, 50, 66, 75 or 100 per cent of the rendered page width by including #wxx after the image name in the link, where xx is the desired width. For example:

`![squirrel](./contributing/squirrel.png#w25)` outputs

`![squirrel](./contributing/squirrel.png#w75)` outputs

inline

With the exception of icons, images are included as block elements by default. You can override this by including `#inline` after the image name. This can be combined with the width setting as follows.

`![squirrel](./contributing/squirrel.png#w25#inline)` outputs 

default

By default images are presented as block elements with 100% width. So `![squirrel](./contributing/squirrel.png#w100)` and `![squirrel](./contributing/squirrel.png)` are equivalent and both output the following:

12.10. translating dtdocs

Translation of the darktable documentation is done via our Weblate instance .

You can either use Weblate’s web UI to translate the documentation or download the translation from Weblate to your computer, edit it, and then upload the changes.

Please do all translation work through Weblate. We will not accept pull requests directly on github to update PO files.

Making a new branch in git

1. Make a new branch to work on it git. For example: `git checkout -b fr-translation-init`

Adding a new language to Hugo

1. In the files `config.yaml` and `config-pdf.yaml`, locate the `languages:` line.
2. Add the language you wish to translate. For example, the English looks like this:

```
en-us:
  title: darktable 3.4 user manual
  weight: 1
```

3. Save the files.

Generating a PO file

Do the following steps if you want to update the POT and PO files from the markdown source.

1. Create an empty PO file for your language in the po folder with the file name `content.<language>.po`. For example: `touch po/content.fr-fr.po`
2. Run the script to populate the PO file: `cd tools/ && ./generate-translations.sh --no-translations`

Generating translated files

Do the following steps to generate the website files from a translation.

1. Generate the translated files: `cd tools/ && ./generate-translations.sh --no-update`.
2. Check the translation by starting hugo's internal server: `hugo server`
3. Open a web browser and check the changes. The URL is in the output of the `hugo server` command.
4. Remove the translated files, as we never check them into git: `cd tools/ && ./generate-translations.sh --rm-translations`.

Translating website and PDF strings

There are two themes for the darktable documentation: one for the HTML website and one for the PDF. You'll need to translate the strings for both.

1. Go to `themes/hugo-darktable-docs-themes/i18n`.
2. Copy content of the file `en.yaml` and name the new file `<your language>.yaml`.
3. Translate the content of the new yaml file.
4. Check the translated PO file into git, push it to github, and open a pull request to have your changes accepted.
5. Repeat the last four steps for the other theme, `themes/hugo-darktable-docs-pdf-theme`.

darktable 3.5 (dev) user manual
April 18, 2021